PRE-NAS: Predictor-assisted Evolutionary Neural Architecture Search

Yameng Peng
RMIT University
Melbourne, Victoria, Australia
yameng.peng@student.rmit.edu.au

Andy Song
RMIT University
Melbourne, Victoria, Australia
andy.song@rmit.edu.au

Vic Ciesielski
RMIT University
Melbourne, Victoria, Australia
vic.ciesielski@rmit.edu.au

Haytham M. Fayek
RMIT University
Melbourne, Victoria, Australia
haytham.fayek@ieee.org

Xiaojun Chang
University of Technology Sydney
Sydney, New South Wales, Australia
xiaojun.chang@uts.edu.au

ABSTRACT

Neural architecture search (NAS) aims to automate architecture engineering in neural networks. This often requires a high computational overhead to evaluate a number of candidate networks from the set of all possible networks in the search space during the search. Prediction of the networks' performance can alleviate this high computational overhead by mitigating the need for evaluating every candidate network. Developing such a predictor typically requires a large number of evaluated architectures which may be difficult to obtain. We address this challenge by proposing a novel evolutionary-based NAS strategy, Predictor-assisted E-NAS (PRE-NAS), which can perform well even with an extremely small number of evaluated architectures. PRE-NAS leverages new evolutionary search strategies and integrates high-fidelity weight inheritance over generations. Unlike one-shot strategies, which may suffer from bias in the evaluation due to weight sharing, offspring candidates in PRE-NAS are topologically homogeneous, which circumvents bias and leads to more accurate predictions. Extensive experiments on NAS-Bench-201 and DARTS search spaces show that PRE-NAS can outperform state-of-the-art NAS methods. With only a single GPU searching for 0.6 days, competitive architecture can be found by PRE-NAS which achieves 2.40% and 24% test error rates on CIFAR-10 and ImageNet respectively.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; **Search methodologies**; **Discrete space search**; • **Networks** → **Network performance modeling**.

KEYWORDS

Evolutionary algorithm, architecture search, performance predictor

1 INTRODUCTION

Despite the overwhelming success of deep learning, the high computational cost associated with model development remains a challenge in the field, especially in real-world applications which often require carefully constructing complicated structures such as ResNet [15], DenseNet [16] and manually tuning the hyperparameters [8, 15, 19, 30, 35]. One approach to address this challenge is NAS (neural architecture search). The aim is to automatically create a competitive neural network for a given task. A fully-machine-designed neural network [43] can achieve a test accuracy of 96.35%

on the CIFAR-10 dataset, in comparison with 96.54% from DenseNet [16], a hand-designed neural network. However, in this task, NAS requires 800 Tesla K40 GPUs parallel running for nearly one month. This high computational cost compromises the saving on manual architecture design. Hence, reducing of the computational costs is one of the key targets in NAS.

One way to reduce the cost in NAS is to cut down evaluation, for example, using fewer epochs or a small portion of data to train the networks [26, 40, 44]. However, this approach may lead to inadequate training, and therefore inaccurate results. Another approach, learning curve extrapolation methods [1, 18], predicts the tendency of network optimisation based on early epochs in training. Similarly, a surrogate model can be built to predict the performances of candidate networks [21]. Predictor-based evaluation methods require their own training, hence there is a need to sample networks from the search space with associated high computational cost. Weights sharing provides a good alternative, in particular, the one-shot model [4, 22, 25], which treats all candidate networks as sub-networks of an over-parameterised super-network. Sub-networks within the same search space can share weights, hence computational costs can be reduced. However sharing weights between heterogeneous architectures is problematic, and can easily lead to incorrect ranking of the candidate networks [2, 39].

Hence, this work addresses the balance between reducing computational cost and improving network ranking by proposing a predictor-assisted evolutionary architecture search algorithm (PRE-NAS). It is driven by an evolutionary algorithm that is based on a population of candidate networks rather than one network. We introduce several new strategies to train predictors more effectively, especially with extremely limited training samples. Firstly, elitist evolution is introduced to maintain a good pool of candidates. Secondly, multi-mutation and a representative selection strategy are introduced. These strategies can improve the training set for the predictor by heuristically increasing the number of mutations and sampling representative candidates from each generation. So the predictor can evaluate multiple candidate networks with no significant cost increase. In addition, a high-fidelity weight inheritance is incorporated to reduce the computational cost further.

The contributions of this work are summarised as follows:

(1) We propose a predictor-assisted evolutionary search algorithm (PRE-NAS) which outperforms several mainstream

NAS algorithms on benchmark and real-world search spaces in terms of efficiency and performance.
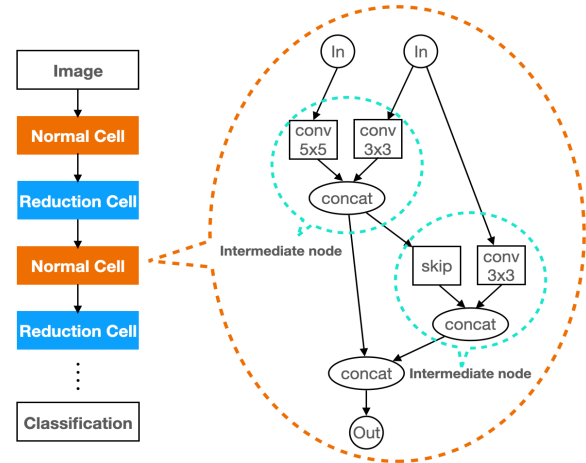
(2) We introduce a representative selection strategy that can train a good performance predictor using an extremely limited number of training samples in the NAS context. Moreover, we apply a multi-mutation strategy to utilise the performance predictor maximally in the evolutionary NAS.

(3) We report a 2.40% test error rate on CIFAR-10 and a 24% top-1 test error rate on ImageNet (mobile setting), with only 0.6 GPU days to search.

Figure 1: Illustration of a cell-based network. Left-hand side demonstrates the complete stacking-constructed network. The micro-structure in the orange dash line represents a cell network. An inside circle represents an intermediate node.

2 RELATED WORK

NAS is generally studied from three perspectives: search algorithm, search space, and network evaluation strategy. Typical search algorithms include reinforcement learning (RL), evolutionary algorithm (EA), gradient-based and Bayesian optimisation (BO). Search space is continuous in gradient-based methods like DARTS [22]. Thus, the search space and evaluation strategy are tightly coupled. In contrast, search space in evolutionary-based [21, 23, 26, 27, 38, 42] and reinforcement learning-based [34, 43] methods is usually discrete, offering better flexibility and compatibility for network evaluation strategies. RL is more costly in comparison. Hence we leverage the advantages of EA as the search algorithm in this NAS study.

There are many successful evolutionary or predictor-assisted NAS work [21, 23, 26, 27, 33, 37, 38, 42]. AmoebaNet [26] adopts a regularised evolutionary algorithm called Aging Evolution. It uses one mutation operation to generate a new architecture and discards the oldest architecture from the population in each search cycle. However, it is not efficient. Duplicate offspring are not checked so it is possible to see identical architectures in two different search cycles. The oldest individual could be the best-performing one in the population, so adding more unnecessary cost onto the search. PNAS [21] adopts a sequential model-based optimisation (SMBO) search algorithm and a multi-layer perceptron (MLP) ensemble predictor. PNAS search starts from shallow cell networks and progresses to complex ones. During the architecture search, a predictor is trained to predict the performance of candidate networks without needing to train all of the candidate networks. After evaluation, top-k candidates are selected for further training [21]. This strategy needs to define a suitable k which may change for different tasks.

Besides, in order to calibrate the performance predictor during architecture search, we adopt weight inheritance training for a few representative candidate networks. Unlike other work [27, 36] inheriting weights between variable-size of generation, our work adopts fixed-size weight inheritance. The topology of parent and offspring networks are similar, hence helpful for function preservation during weight inheritance. In PRE-NAS, we propose a percentile representative selection strategy that is more generalizable than the top-k strategy. PRE-NAS incorporates the flexibility of the EA, efficiency of performance prediction and weight inheritance to address the aforementioned issues in NAS.
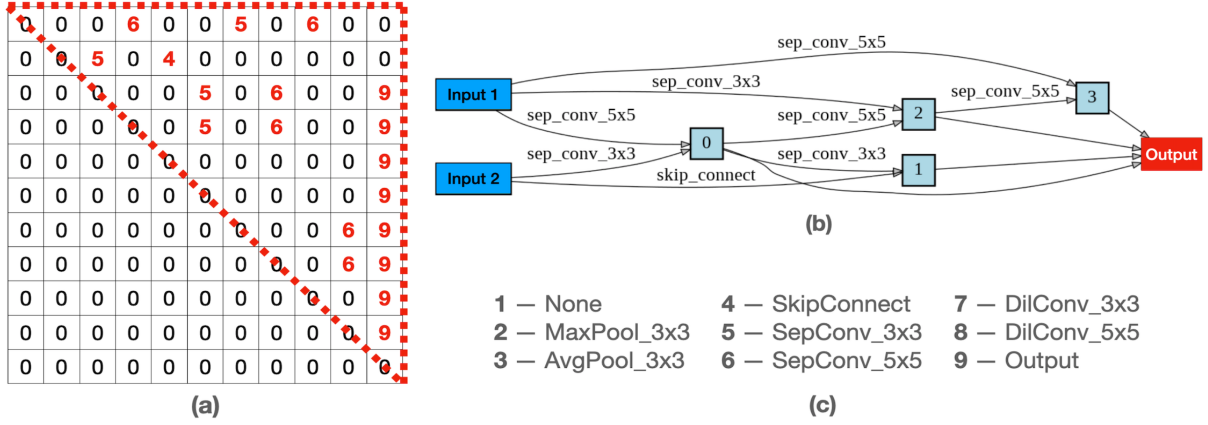
3 PREDICTOR-ASSISTED EVOLUTIONARY NAS

The details of PRE-NAS are described here, including the search space, search algorithm, multi-mutation, representative selection, performance predictor, and weight inheritance training.

3.1 Search Space of Cell-based Networks

The search space is defined by the specific task and the chosen representation. Two search spaces, NAS-Bench-201 [11] and DARTS [22] are studied here. They are both cell-based search spaces, as many well-performing networks have been manually designed based on this representation. Fig. 1 is an illustration of the construction of a cell-based network by stacking repeated modules together [15, 16]. The shallow network in the oval callout on the right is called a Cell Network. Each cell network consists of several pre-defined operations. A complete network can be constructed by stacking these cells (Fig. 1 left). The number in a stack, e.g. the depth of the network, depends on the difficulty of the target task. Thus, the search algorithm only needs to focus on the microstructure of a cell network.

The search space of NAS-Bench-201 [11] is relatively simple as it aims to provide a fair environment for comparison between NAS algorithms. The elements are four nodes (one input node, two intermediate nodes and one output node) and five pre-defined network operations (none, skip-connection, 1x1 convolution, 3x3 convolution, 3x3 average pooling) connecting these nodes. So the search space consists of $5^6 = 15625$ architectures. Thus, it is possible to exhaustively train all 15625 architectures for 200 epochs on three different image datasets and record their individual performance. Then the accuracy of a candidate network can be obtained by querying the records instead of training from scratch.

Figure 2: Illustration of upper triangular adjacency matrix representation for architecture encoding. The corresponding architecture is visualised on the top right demonstrating a cell network (b). The list (c) on the bottom right shows the available network operations and their indices, which are the cell values of the matrix (a) on the left.

DARTS space appears much more recently [5, 17, 28, 29, 38]. It has a much larger and more complex search space than NAS-Bench-201. There are eight different operations available in the search space of DARTS: 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated convolutions, 3×3 max pooling, 3×3 average pooling, skip connection and none. Each cell network has 2 input nodes, 4 intermediate nodes and 1 output node, leading to $\prod_{i=1}^{4} \frac{(i+1)i}{2} \times 7^2 \approx 10^9$ possible architectures. Thus, more complex and powerful neural architectures can be found in this search space. This study adopts a similar setup as DARTS, except that DARTS searches for a Reduction Cell with a different structure from the Normal Cell. Our Reduction Cell and Normal Cell have the same structure, similar to the strategy in PNAS [21] and BONAS [28].

The encoding of the above two search spaces both use upper triangular adjacency matrices, as demonstrated in Fig. 2. A cell network is represented as an upper triangular matrix where the number in a cell indicates the network operation connecting the two nodes represented in the row and in the column, similar to the encoding schemes in [11, 28]. This matrix is also the input to our performance predictor. Note that the output (No. 9) in the operation list is to demonstrate the encoding scheme, but it is not involved in the actual architecture search, e.g, mutations or predictor training.

3.2 Evolutionary Search Algorithm

As the evolutionary search algorithms [12, 23, 26, 27, 31, 33, 38] have shown competitiveness in flexibility and performance perspectives over RL, BO, and gradient-based methods [18, 22, 28, 43], it is the basis of our PRE-NAS. But unlike other works such as AmoebaNet [26] which uses aging evolution, PRE-NAS adopts an elitism strategy. Note that, EA algorithms in NAS usually manipulate and update only a few networks in a population as the cost of operating on all networks can be prohibitive [7].

The details are shown in **Algorithm 1**. Steps 4-8 comprise the initialisation phase which prepares the population by randomly generating P architectures (Step 5) and using conventional training to obtain validation accuracy on the target task (Step 6). A

Algorithm 1 PRE-NAS

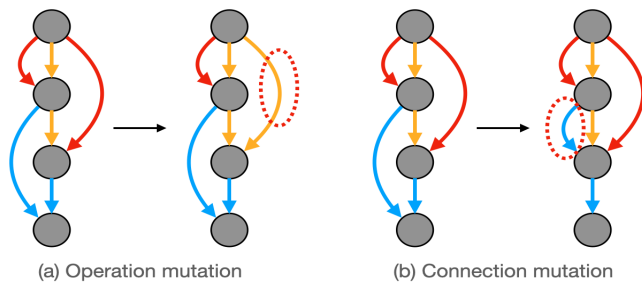**Require:** Population size P, Search cycle C, Sample size S
1: $population \leftarrow \emptyset$
2: $history \leftarrow \emptyset$
3: $Children \leftarrow \emptyset$
4: **while** $population < P$ **do**
5:     $model.arch \leftarrow RandomGenerateArchitectures()$
6:     $model.valid\_accuracy \leftarrow Train(model.arch)$
7:     Add $model$ to the $population$
8:     Add $model$ to $history$
9: **end while**
10: **while** C not fulfilled **do**
11:     $Predictor \leftarrow$ training the predictor
12:     $candidates \leftarrow$ randomly sample S architectures from the $population$
13:     $parent \leftarrow$ best-performing one in the $candidates$
14:     **while** $mutation\_times$ not fulfilled **do**
15:         $child.arch \leftarrow Mutate(parent.arch)$
16:         $child.valid\_accuracy \leftarrow Predictor(child.arch)$
17:         Add $child$ to $Children$
18:     **end while**
19:     $Representatives \leftarrow$ select **few** representatives from $Children$
20:     **for all** $child \in Representatives$ **do**
21:         $child.valid\_accuracy \leftarrow Inheritance\_Train(child.arch)$
22:         Add $child$ to $history$
23:     **end for**
24:     Calculate **Spearman coefficient** between predicted and trained accuracy of $Representatives$;
25:     Enlarge $mutation\_times$ if **Spearman coefficient** higher than previous
26:     Add the best-performing $child$ to the $population$
27:     Remove the worst $architecture$ from the $population$
28: **end while**
29: **end**

model contains architecture and its accuracy will be added into the *population* and *history* (Steps 7&8). The training samples for the performance predictor are stored in the *history*. After the initialisation, the search begins. The key differences compared to other EAs are: (1) multi-mutation is used to generate a group of child networks (Step 14); (2) the performance predictor is used to predict the accuracy of child networks (Step 16) which allows us to evaluate many architectures with a much lower cost; (3) a representative selection strategy is adopted to pick distinctive architectures (Step 19); (4) weight inheritance is used in offspring training (Step 21). The Spearman coefficient is calculated between predicted and trained accuracy (Step 24). A high Spearman coefficient indicates the predictor can better predict the tendency of the child networks.

(a) Operation mutation   (b) Connection mutation

Figure 3: Illustration of the two mutation strategies of PRE-NAS. Grey circles represent nodes of the cell network and coloured lines represent different network operations. (a) An offspring network generated by operation mutation, the red dashed circle indicates that the network operation at this position has been mutated. (b) An offspring network generated by connection mutation, the red dash line circle indicates the connection between two nodes has been mutated.

3.3 Multi-mutation

Fig. 3 shows two mutation operators in PRE-NAS. Each graph is a cell network and a grey circle is a network node. Coloured lines represent different network operations, e.g. convolution or pooling. An arrow represents the direction of the tensor flow. The operation mutation (Fig. 3a) generates an offspring network from a parent network by randomly choosing a network operation and replacing it with another randomly picked operation. The connection mutation (Fig. 3b) is similar but changes the destination node of an operation. Theoretically, it is possible to explore the entire space by alternating these two mutation operators. The AmoebaNet [26] suffers from high computational overhead, as it only generates one offspring by mutating the parent in each search cycle. In PRE-NAS, a multi-mutation strategy is used to take advantage of the predictor so all possible offspring networks of a given parent can be generated and evaluated in each search cycle to better cover the search space. The total number of possible offsprings from a parent network can be calculated by $2N(O-1) + 2N!$, where N represents the number of intermediate nodes and O represents the number of network operations. As discussed in Section 3.1, the DARTS space consists

of 4 intermediate nodes and 8 operations, which means a parent network can generate up to 76 different offsprings.

3.4 Representative Selection

After mutation, the performances of offspring networks will be predicted by our performance predictor. Good candidate networks can be picked by the top-k strategy [21, 28, 37]. However, this strategy is not adequate to cover the entire distribution of different scenarios, thus, it is hard to train a predictor with good generalisation. Besides, it needs a way to determine the optimal K. To better maintain diversity and coverage, we propose a representative selection strategy that selects representative architectures from the *children*, not just the best ones. More specifically, the architectures are selected based on the statistical percentile of predicted validation accuracy: the architectures at the maximum percentile, 75% point, 50% point, 25% point, and the minimum percentile. After the selection, the ground-truth accuracy of these representative candidates can be obtained by training them on the target dataset. The Spearman Ranking Correlation Coefficients (a monotonic relationship between two groups) are calculated between their predicted accuracy and ground-truth accuracy of the representative architectures. The higher the Spearman value, the better ranking of the predictor. Further, the representative architectures and their accuracy values will be added to the *history* to be the samples for further training of the performance predictor. More exhibitions and discussions on the advantages of our representative selection strategy will be introduced in Section 4.1.

3.5 Performance Predictor

The desired characteristics of a PRE-NAS predictor are: (1) it can learn from limited training samples, as the cost of obtaining the target value (validation accuracy) is expensive; (2) it contains only a few model parameters, as training a big model will slow down the overall search process. The encoding scheme described in the Section 3.1, architectures are encoded as upper triangular adjacency matrices. Multi-mutations are applied to this encoding, which will also be the input to the performance predictor. Note that architectures from different search spaces would have matrices of different sizes. For instance, architectures from the search space of NAS-Bench-201 will be encoded as a 4×4 matrix, architectures from DARTS will be encoded as an 11×11 matrix (Fig. 2).
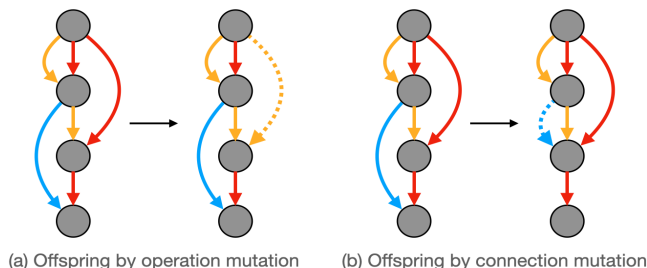
The performance predictors we considered are Random Forest Regressor, Support Vector Regressor, Bayesian Ridge Regressor, Kernel Ridge, Linear Regressor, and Multi-Layer Perceptron, due to their computational efficiency. These predictors are tested on the NAS-Bench-201. To simulate the evaluation process during the architecture search, we randomly sample 100 architectures and their accuracy as the training data (less than 1% of the total amounts of the search space) and randomly sample another 100 pairs from the rest of 15525 architectures as the test data. We record the Spearman coefficient between predicted and ground-truth accuracy. 500 independent experiments show that Random Forrest is the best performing and most consistent predictor when facing extremely limited training samples. The results are shown in Table 1.

Predictor	Spearman Coefficient
Random Forest	0.65±0.08
Support Vector	0.39±0.11
Bayesian Ridge	0.04±0.11
Kernel Ridge	0.06±0.10
Linear Regressor	0.04±0.11
Multi-Layer Perceptron	0.06±0.10

Table 1: Results of 6 regressors trained with 100 samples. Each regressor is trained with the same data in each experiment. We record the Spearman coefficient between predicted and ground-truth accuracy, and report the mean and standard deviation based on 500 independent experiments.
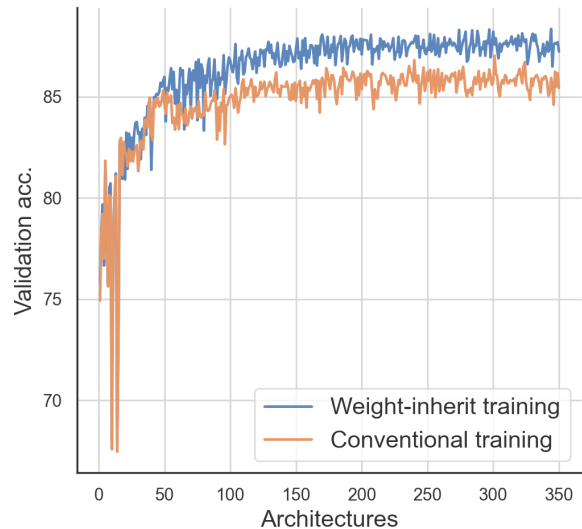
3.6 High-Fidelity Weight Inheritance

Similar to the well known one-shot model [22, 27], weight inheritance is also based on the weight sharing technique. One-shot allows sharing the weights across the entire search space, even for heterogeneous architectures. Weight inheritance however only shares weights between topologically homogeneous networks, e.g., parent and offspring networks. Thus, evolutionary search is naturally suitable for the weight inheritance training, because offspring networks are highly similar in terms of topology as they are generated from parent networks by mutating particular operations or connections (Fig. 4). Unlike previous work on the variable size of networks [27, 36], we adopt a fixed size weight inheritance strategy to best preserve the network function. Moreover, we use Kaiming Norm [14] to initialise the mutated connection or operation weights.

(a) Offspring by operation mutation   (b) Offspring by connection mutation

Figure 4: Illustration of weight inheritance, grey circles represent nodes of the cell network, and colour lines represent different network operations. (a) and (b) demonstrate two offspring networks generated by operation and connection mutations. Solid lines indicate that these operations will inherit the weights from the parent network. Dash lines indicate that the weights of these operations will reset.

To investigate the effectiveness of our weight inheritance, we generated 350 candidate networks by implementing multi-mutation on randomly generated parent networks from the search space of DARTS. Both conventional (from scratch) and weight inheritance training were employed on the CIFAR-10 dataset. Both training runs were optimised with momentum SGD, and a batch size of 128. The hyper-parameters for the former strategy were set as follows:

Figure 5: Results of weight inheritance and conventional training on 350 candidate networks. The network order is sorted by the CIFAR-10 validation accuracy. The Spearman Correlation Coefficient between these two groups is 0.83.

learning rate was 0.025 (annealed via cosine strategy), momentum was 0.9, weight decay was 0.0001, and number of training epochs was 100. For weight inheritance training, as offsprings will partially inherit trained weights from parent networks, both learning rate and training epochs were reduced as follow: the learning rate was set to 0.01 and the number of training epochs was set to 50.

Fig. 5 shows the CIFAR-10 validation accuracy of these 350 networks obtained by (1) weight inheritance and (2) conventional training. The Spearman Coefficient between these two groups is 0.83, which indicates the high fidelity of inherited weights. This is significantly better than the one-shot model that shows poor correlation with the ground-truth accuracy [39]. The weight inheritance strategy can obtain both training efficiency and high ranking correlation, as it required much fewer training epochs to reach a similar performance when compared with conventional training.

4 EXPERIMENTS AND RESULTS

We have implemented our experiments on two search spaces which are NAS-Bench-201 [11] and DARTS [22]. We use NAS-Bench-201 space to implement a large scale of experiments, it aims to prove the stability of search performance of PRE-NAS. We use DARTS space to implement real-world architecture search, which can generate high-performance neural networks.

4.1 Search on NAS-Bench-201

As the search space of NAS-Bench-201 has been exhaustively evaluated and the performance of every network within the search space is known, there is no need for training the initial population or offspring candidates, but query the record. On this benchmark, PRE-NAS was repeated 500 times in order to reduce the variance between
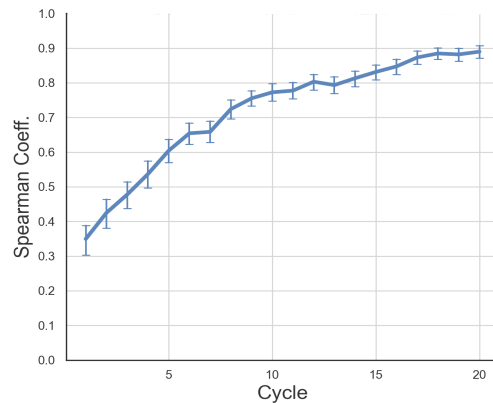
Algorithm	CIFAR-10		CIFAR-100		ImageNet-16-120	
	Validation	Test	Validation	Test	Validation	Test
RSPS [20]	84.16±1.69	87.16±1.69	59.00±4.60	58.33±4.34	31.56±3.28	31.14±3.88
DARTS-V1 [22]	39.77±0.00	54.30±0.00	15.03±0.00	15.61±0.00	16.43±0.00	16.32±0.00
DARTS-V2 [22]	39.77±0.00	54.30±0.00	15.03±0.00	15.61±0.00	16.43±0.00	16.32±0.00
GDAS [10]	90.00±0.21	93.51±0.13	71.14±0.27	70.61±0.26	41.70±1.26	41.84±0.90
SETN [9]	82.25±5.17	86.19±4.63	56.86±7.59	56.87±7.77	32.54±3.63	31.90±4.07
ENAS [25]	39.77±0.00	54.30±0.00	15.03±0.00	15.61±0.00	16.43±0.00	16.32±0.00
AmoebaNet [26]	91.19±0.31	93.92±0.30	71.81±1.12	71.84±0.99	45.15±0.89	45.54±1.03
Random Search [3]	90.93±0.36	93.70±0.36	70.93±1.09	71.04±1.07	44.45±1.10	44.57±1.25
REINFORCE [43]	91.09±0.37	93.85±0.37	71.61±1.12	71.71±1.09	45.05±1.02	45.24±1.18
BOHB [13]	90.82±0.53	93.61±0.52	70.74±1.29	70.85±1.25	44.26±1.36	44.42±1.49
PRE-NAS (ours)	91.37±0.28	94.04±0.34	71.95±1.21	72.02±1.22	45.16±1.00	45.34±1.03
*Optimal	91.61	94.37	73.49	73.51	46.77	47.31

Table 2: Search results of our proposed PRE-NAS and other search algorithms on the NAS-Bench-201 search space. Each algorithm is searching under a similar computational budget and repeated 500 times. We recorded the CIFAR-10 validation accuracy of the best-performing architecture which found in each experiment and reported the mean and standard deviation over 500 independent experiments. *Optimal indicates the accuracy of best-performing architecture recorded in the benchmark.

runs. The computational budget was set according to the experiments in [11]. We set the search *Cycle C* to 20, *Population_size P* to 20, *Sample_size S* to 10, and initial *mutation_times* equal to the *Sample_size S*. This means that the algorithm runs mutation operations 10 times to generate 10 offspring networks. Once the child networks are evaluated by the performance predictor, then the percentile selection will apply. There is one additional hyper-parameter called *mutation_factor*, which is used to increase *mutation_times*. For instance, if the predictor performed well in the current search cycle, *mutation_times* will be increased by *mutation_factor*. Thus, the predictor could gradually evaluate more architectures if it performs well. This parameter was set to 1.2. This setup ensures a similar computational budget which is the total number of architectures that have been queried by using the benchmark. Note that the total number of training samples for the performance predictor is around 100 under this setup (approx. 10^{-4} of the search space), which is similar to our simulation experiment shown in Section 3.5.

4.1.1 NAS-Bench-201 Results. Table 2 shows the results from PRE-NAS and respective SOTA algorithms over 500 independent experiments. The columns under CIFAR-10, CIFAR-100 and ImageNet-16-120 indicate the average validation and test accuracy with the standard deviation of the best architecture by each algorithm. The last row shows the best accuracy that the architecture in this search space could possibly reach. The large scale of independent architecture search experiments clearly show that the overall performance of PRE-NAS dominates all other algorithms, given a similar computational budget.

We also recorded the Spearman coefficient between predicted and ground-truth CIFAR-10 validation accuracy at each search cycle. Fig. 6 shows the average value of the Spearman coefficient over 500 independent experiments. Even with extremely limited training samples, the performance predictor was still continuously improving, taking advantage of multi-mutation and representatives selection. Moreover, we have investigated two extra candidate selection
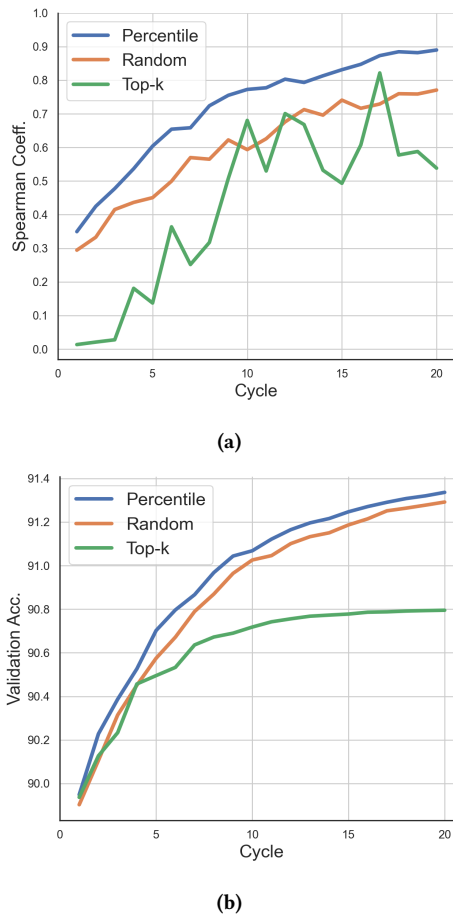

Figure 6: Result of Spearman coefficient between predicted and ground-truth accuracy. The line indicates the average value of the Spearman coefficient at each search cycle based on 500 independent experiments.

strategies, top-*k* (e.g. top-5) and random. The experimental setups are the same as for our percentile strategy. Fig. 7 demonstrate that our percentile-based representative selection strategy is superior on both predictor training and architecture search. Based on these results, we think that percentile or even random-based representative selection strategies are more likely to cover the distribution of the search space than the top-*k* strategy. Thus, it constructed richer training samples for the performance predictor.

4.2 Search on DARTS

DARTS search space contains approximately 10^9 possible architectures, much larger than NAS-Bench-201. We follow a similar procedure to the experiments on NAS-Bench-201 that use the

(a)

(b)

Figure 7: (a) Spearman coefficient between predicted and ground-truth accuracy under three candidates selection strategies. (b) Ground-truth accuracy under three candidates selection strategies. Each line indicates the average value at each search cycle based on 500 independent experiments.

CIFAR-10 as the proxy task dataset, except that it requires training the networks (shown in **Algorithm 1**). Specifically, we randomly generate architectures to form the initial population and use conventional training to obtain their accuracy. Then the offspring networks can inherit weights from them. Therefore we can use weight inheritance training to train the following offspring networks with a much less computational cost. We set the search *Cycle C* to 100, *Population_size P* to 64, *Sample_size S* to 32, initial *mutation_times* equals the *Sample_size S*, and the *mutation_factor* to 1.2. For the initial population, we use conventional training and set the epoch number to 100, momentum to 0.9, learning rate to 0.025 (anneal cosine strategy) and weight decay to 0.0001. We use weight inheritance training for the offspring candidate networks and set the learning rate to 0.01, epoch number to 50 which is the same as the setup in Section 3.6. Both conventional and weight inheritance training are optimised by momentum SGD and batch size to 128. We set the network layer to 1 (single cell) and the initial channel to 16 during the architecture search, as training

the shallow networks would significantly reduce the computational cost. Under this setup, there are 64 networks (initial population) trained by the conventional way and approximately 500 networks (offspring networks) trained by weight inheritance. Thus, the theoretical number of training samples for the performance predictor is 564 (approx. 10^{-8} of the search space). PRE-NAS would evaluate around 7000 architectures by implementing the multi-mutation. Each network would take approximately 10 minutes on a single Tesla V100 GPU if training from scratch. This leads to approximate 48 days to complete the search. By utilising the performance predictor and weight inheritance training, PRE-NAS reduces the computational cost to 0.6 GPU days.

4.2.1 Result on CIFAR-10. After PRE-NAS found the promising architecture, we use it to construct a full-size network as discussed in Section 3.1. For the training of our full-size network, we apply a similar setup with DARTS [22], i.e., stacked the cell network with 20 layers. Table 3 shows the results of the comparison between our full-size network and the networks from other methods. As the CIFAR-10 results have high variance between different runs, we trained our full-size model 10 times and reported the mean and standard deviation. From the results, the architecture found by PRE-NAS is significantly better than most state-of-the-art which also searched on DARTS space. PRE-NAS only took 0.6 GPU days for the search. From the perspective of search cost, our method is still not inferior to the mainstream one-shot NAS. In evolution-based search algorithms, PRE-NAS also shows a huge advantage in both search efficiency and performance.

4.2.2 Result on ImageNet. As the ImageNet dataset contains approximate 13k large size images with 1000 classes, directly searching on ImageNet will cause gigantic computational overhead. The performance transferability of cell networks have been proved in most of the previous work [6, 22, 39, 44]. Our full-size network for ImageNet is also constructed from the architecture found on CIFAR-10, we apply a similar setup with DARTS [22] to build a slightly larger model. Table 4 shows the results of comparison between our model and the networks from other work, which is consistent with the results shown in Table 3.

5 CONCLUSION

In this paper, we proposed a predictor-assisted evolutionary search algorithm called PRE-NAS. We demonstrated that the predictor-based evolutionary NAS is highly competitive with mainstream methods. By adopting the multi-mutation and representative selection strategies, we can train a good performance predictor with extremely limited data (e.g. approx. 10^{-8} to 10^{-4} of the total size of the search space). The predicted accuracy shows a strong correlation with the ground-truth accuracy. By utilising the high-fidelity weight inheritance strategy, we can further reduce the cost of candidate network training. On DARTS search space, the search cost is reduced from 48 GPU days to 0.6 GPU days. Extensive experiments show that the network found by PRE-NAS can outperform several state-of-the-art search algorithms on the search spaces of NAS-Bench-201 and DARTS.

In this work, we only used network accuracy as the search objective. Introducing multi-objectives could be a promising direction

Algorithm	Test Error (%)	Params (M)	GPU	Search Cost (GPU Days)	Search Method	Evaluation	Year
ENAS [25]	2.89★	4.6	-	0.45	Reinforce	One-shot	ICML2018
PNAS [21]	3.34±0.09	3.2	-	225	SMBO	Predictor	ECCV2018
RandomNAS [20]†	2.85±0.08	4.3	P100&V100	2.7	Random	One-shot	UAI2020
DARTS [22]†	3.00±0.14	3.3	4 1080Ti	4	Gradient	One-shot	ICLR2019
P-DARTS [5]†	2.50★	3.4	-	0.3	Gradient	One-shot	ICCV2019
FairDARTS [6]†	2.54★	2.8	1 V100	0.42	Gradient	One-shot	ECCV2020
CGP-ResSet [32]	5.01★	1.7	2 1080Ti	4	Evolution	Conventional	IJCAI2018
AmoebaNet-A [26]	3.34±0.06	3.2	450 K40	3150	Evolution	Conventional	AAAI2019
Lemonade [12]	3.05★	4.7	16 Titan X	56	Evolution	Conventional	ICLR2019
EENA [42]	2.56★	8.47	1 Titan X	0.65	Evolution	Weight Inheritance	ICCV2019
NSGA-Net [24]	2.75★	3.3	1 1080Ti	4	Evolution	Predictor	GECCO2019
NSGANetV1-A4 [24]	2.02★	4.0	8 2080Ti	27	Evolution	Predictor	TEC2020
EcoNAS [41]†	2.62±0.02	2.9	1 1080Ti	8	Evolution	Conventional	CVPR2020
CARS [38]†	2.62★	3.6	-	0.4	Evolution	One-shot	CVPR2020
EvNAS [31]†	2.47±0.06	3.6	1 2080 Ti	3.83	Evolution	One-shot	GECCO2021
PRE-NAS (ours)†	2.49±0.09	4.5	1 V100	0.6	Evolution	Predictor+w/i	

Table 3: Performance comparison between the networks found by PRE-NAS and other search algorithms on CIFAR-10, the lower test error rate is better. † means the method has also been used on DARTS search space. ★ indicates the original paper only provided their best performance. Dash means the original paper has not provided the information. 'w/i' is short for the weight inheritance training.

Algorithm	Test Error(%) Top-1 / Top-5	Params (M)	GPU	Search Cost (GPU Days)	Search Method	Evaluation	Year
PNAS [21]	25.8 / 8.1	5.1	-	225	SMBO	Predictor	ECCV2018
DARTS [22]†	26.7 / 8.7	4.7	4 1080Ti	4	Gradient	One-shot	ICLR2019
P-DARTS [5]†	24.4 / 7.4	4.9	-	0.3	Gradient	One-shot	ICCV2019
FairDARTS [6]†	26.3 / 8.3	5.3	1 V100	0.42	Gradient	One-shot	ECCV2020
AmoebaNet-A [26]	25.5 / 8	5.1	450 K40	3150	Evolution	Conventional	AAAI2019
EcoNAS [41]†	25.2 / -	4.3	1 1080Ti	8	Evolution	Conventional	CVPR2020
CARS [38]†	24.8 / 7.5	5.1	-	0.4	Evolution	One-shot	CVPR2020
NSGANetV1-A3 [24]	23.8 / 7	5.0	8 2080Ti	27	Evolution	Predictor	TEC2020
EvNAS [31]†	24.4 / 7.4	5.1	1 2080 Ti	3.83	Evolution	One-shot	GECCO2021
PRE-NAS (ours)†	24 / 7.8	6.2	1 V100	0.6	Evolution	Predictor+w/i	

Table 4: Performance comparison between the networks found by PRE-NAS and other search algorithms on the ImageNet (mobile setting), the lower test error rate is better. † means the method has also been used on DARTS search space. Dash means the original paper has not provided the information. 'w/i' is short for the weight inheritance training.

for future work as multiple requirements, e.g. accuracy and model size, then can be considered simultaneously during architecture search. Furthermore, it can be observed that the performance of evolutionary algorithms depends on the quality of the initial population. A high-quality initial population is more likely to generate good offspring networks. In near future, we will investigate ways to better initialize the initial populations.

REFERENCES

[1] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. 2017. Practical Neural Network Performance Prediction for Early Stopping. *CoRR* abs/1705.10823 (2017).

[2] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. 2018. Understanding and Simplifying One-Shot Architecture Search. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80),* Jennifer Dy and Andreas Krause (Eds.).

[3] James Bergstra and Yoshua Bengio. 2012. Random Search for Hyper-Parameter Optimization. *J. Mach. Learn. Res.* 13 (2012), 281–305.

[4] Han Cai, Ligeng Zhu, and Song Han. 2018. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. *CoRR* abs/1812.00332 (2018). arXiv:1812.00332

[5] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. 2019. Progressive Differentiable Architecture Search: Bridging the Depth Gap Between Search and Evaluation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).*

[6] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. 2020. Fair DARTS: Eliminating Unfair Advantages in Differentiable Architecture Search. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XV,* Vol. 12360. Springer, 465–480.

[7] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* 6, 2 (2002), 182–197.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and*

PMLR, 550–559.

Short Papers). Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186.

[9] Xuanyi Dong and Yi Yang. 2019. One-Shot Neural Architecture Search via Self-Evaluated Template Network. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019.* IEEE, 3680–3689.

[10] Xuanyi Dong and Yi Yang. 2019. Searching for a Robust Neural Architecture in Four GPU Hours. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019.* Computer Vision Foundation / IEEE, 1761–1770.

[11] Xuanyi Dong and Yi Yang. 2020. NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.*

[12] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Efficient Multi-Objective Neural Architecture Search via Lamarckian Evolution. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019.*

[13] Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research, Vol. 80),* Jennifer G. Dy and Andreas Krause (Eds.). PMLR, 1436–1445.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *CoRR* abs/1502.01852 (2015). arXiv:1502.01852

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[16] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. 2017. Densely Connected Convolutional Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017.* IEEE Computer Society, 2261–2269.

[17] Andrew Hundt, Varun Jain, and Gregory D. Hager. 2019. sharpDARTS: Faster and More Accurate Differentiable Architecture Search. *CoRR* abs/1903.09900 (2019).

[18] Aaron Klein, Stefan Falkner, Jost Tobias Springenberg, and Frank Hutter. 2017. Learning curve prediction with Bayesian neural networks. *International Conference on Learning Representations* (2017).

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25.* 1097–1105.

[20] Liam Li and Ameet Talwalkar. 2020. Random Search and Reproducibility for Neural Architecture Search. In *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference (Proceedings of Machine Learning Research, Vol. 115).* 367–377.

[21] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. 2018. Progressive Neural Architecture Search. In *Proceedings of the European Conference on Computer Vision (ECCV).*

[22] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019. DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations (ICLR).*

[23] Zhichao Lu, Kalyanmoy Deb, Erik D. Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. 2020. NSGANetV2: Evolutionary Multi-objective Surrogate-Assisted Neural Architecture Search. In *Proceedings of the European Conference on Computer Vision (ECCV),* Vol. 12346. Springer, 35–51.

[24] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh D. Dhebar, Kalyanmoy Deb, Erik D. Goodman, and Wolfgang Banzhaf. 2019. NSGA-Net: neural architecture search using multi-objective genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2019, Prague, Czech Republic, July 13-17, 2019,* Anne Auger and Thomas Stützle (Eds.). ACM, 419–427.

[25] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. 2018. Efficient Neural Architecture Search via Parameters Sharing. In *Proceedings of the 35th International Conference on Machine Learning (ICML).* 4095–4104.

[26] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. 2019. Regularized Evolution for Image Classifier Architecture Search. In *AAAI Conference on Artificial Intelligence,* Vol. 33. 4780–4789.

[27] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. 2017. Large-Scale Evolution of Image Classifiers. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70).* International Convention Centre, Sydney, Australia, 2902–2911.

[28] Han Shi, Renjie Pi, Hang Xu, Zhenguo Li, James T. Kwok, and Tong Zhang. 2020. Bridging the Gap between Sample-based and One-shot Neural Architecture Search with BONAS. In *Advances in Neural Information Processing Systems,* Vol. 33.

[29] Julien Siems, Lucas Zimmer, Arber Zela, Jovita Lukasik, Margret Keuper, and Frank Hutter. 2020. NAS-Bench-301 and the Case for Surrogate Benchmarks for Neural Architecture Search. *CoRR* abs/2008.09777 (2020).

[30] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations (ICLR).*

[31] Nilotpal Sinha and Kuan-Wen Chen. 2021. Evolving neural architecture using one shot model. In *GECCO '21: Genetic and Evolutionary Computation Conference, Lille, France, July 10-14, 2021,* Francisco Chicano and Krzysztof Krawiec (Eds.). ACM, 910–918.

[32] Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. 2018. A Genetic Programming Approach to Designing Convolutional Neural Network Architectures. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18.* International Joint Conferences on Artificial Intelligence Organization, 5369–5373.

[33] Yanan Sun, Handing Wang, Bing Xue, Yaochu Jin, Gary G. Yen, and Mengjie Zhang. 2020. Surrogate-Assisted Evolutionary Deep Learning Using an End-to-End Random Forest-Based Performance Predictor. *IEEE Trans. Evol. Comput.* 24, 2 (2020), 350–364.

[34] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. 2019. MnasNet: Platform-Aware Neural Architecture Search for Mobile. In *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR).* 2820–2828.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems,* Vol. 30. 5998–6008.

[36] Tao Wei, Changhu Wang, Yong Rui, and Chang Wen Chen. 2016. Network Morphism. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016 (JMLR Workshop and Conference Proceedings, Vol. 48),* Maria-Florina Balcan and Kilian Q. Weinberger (Eds.). 564–572.

[37] Wei Wen, Hanxiao Liu, Yiran Chen, Hai Helen Li, Gabriel Bender, and Pieter-Jan Kindermans. 2020. Neural Predictor for Neural Architecture Search. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIX,* Vol. 12374. Springer, 660–676.

[38] Zhaohui Yang, Yunhe Wang, Xinghao Chen, Boxin Shi, Chao Xu, Chunjing Xu, Qi Tian, and Chang Xu. 2020. CARS: Continuous Evolution for Efficient Neural Architecture Search. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020.* IEEE, 1826–1835.

[39] Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. 2020. Evaluating The Search Phase of Neural Architecture Search. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.*

[40] Arber Zela, Aaron Klein, Stefan Falkner, and Frank Hutter. 2018. Towards Automated Deep Learning: Efficient Joint Neural Architecture and Hyperparameter Search. In *ICML 2018 AutoML Workshop.*

[41] Dongzhan Zhou, Xinchi Zhou, Wenwei Zhang, Chen Change Loy, Shuai Yi, Xuesen Zhang, and Wanli Ouyang. 2020. EcoNAS: Finding Proxies for Economical Neural Architecture Search. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020.* IEEE, 11393–11401.

[42] Hui Zhu, Zhulin An, Chuanguang Yang, Kaiqiang Xu, Erhu Zhao, and Yongjun Xu. 2019. EENA: Efficient Evolution of Neural Architecture. In *2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019, Seoul, Korea (South), October 27-28, 2019.* IEEE, 1891–1899.

[43] Barret Zoph and Quoc V. Le. 2017. Neural Architecture Search with Reinforcement Learning. In *International Conference on Learning Representations (ICLR).*

[44] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. 2018. Learning Transferable Architectures for Scalable Image Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*