Proceedings of 11th IEEE International Conference
on CYBER Technology in Automation, Control, and Intelligent Systems
July 27 - 31, 2021, Jiaxing, China

2021 IEEE 11th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER) | 978-1-6654-2527-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/CYBER53097.2021.9588314

Neural Architecture Search Based on Tabu Search and Evolutionary Algorithm

Zhun Fan*, Zhoubin Long, Wenji Li, Zhaojun Wang, Zhi Yang and Liu Wang

***Abstract*— Most existing optimization methods for neural architecture search (NAS), including evolutionary algorithms, reinforcement learning and gradient-based approaches, have not employed memory strategies explicitly, which may lack of efficiency when searching neural architectures. To solve this issue, we propose a new NAS approach by using an evolutionary algorithm which employs a tabu mechanism to help to improve the search efficiency. To be more specific, the individuals of parent population are selected by tournament selection and tabu list. The tournament selection select parent population according to the accuracy of each individual. And the tabu mechanism builds a tabu list to record the chosen operations in the last previous search process, which employs a search memory mechanism to improve the efficiency explicitly. To confirm the superior performance of our approach, a well-designed surrogate model is used to accelerate the process of performance evaluation on CIFAR-10. The comprehensive experimental results show that the proposed method can reach to 2.48% error rate with about 2 GPU days, which demonstrates the superiority of the suggested method.**

I. Introduction

In recent years, deep convolutional neural networks have made great success in many computer vision fields, including semantic segmentation [1], object detection [2], [3], image classification and so on. However, most existing neural networks, such as AlexNet [4], GoogleNet [5] and Faster RCNN [2], are designed empirically, which need a lot of expertise experiences. For this reason, designing a neural network automatically is becoming a hot and promising research direction.

NAS is a typical bi-level optimization problem [6], where the upper level is to optimize the neural network structure, while the lower level is to optimize the weights of the neural network with a given architecture. To solve this problem, extensive research has been conducted for searching neural architectures by evolutionary algorithm (EA), reinforcement learning (RL) and gradient-based methods. However, most of these methods need thousands of GPU days to search an optimal architecture. For example, NASNet [7], a NAS method based on RL, costs 2000 GPU days in the search procedure. AmoebaNet [8], an EA base method, costs 3150 GPU days to search an optimal architecture. To accelerate the search procedure, weights sharing and neural architectures performance predictor mechanisms are proposed. For the weights sharing, it needs to pretrain a supernet that contains all the subnets in the search procedure, so the subnets can inherit weights from the supernet rather than training the model from scratch. For the performance predictor approach, it trains a predictor model to predict the performance of the neural network.

In this paper, we propose a new NAS method and the framework is shown in Fig. 1. To reduce the useless search, we apply a tabu mechanism to select solutions in the evolutionary algorithm. In the search process, the selection of parent population not only considers the accuracy of each individual but also considers whether the neighbors of current individual are selected in the previous search process. To further reduce the search time, a surrogate model [9] is applied to predict the performance of the individual.

The population consists of two parts, including the archive subpopulation and the tabu subpopulation. For the archive subpopulation, the individuals in the population are selected according to the accuracy on the validation set, and this subpopulation maintains the best individual during the search process. The tabu subpopulation is composed of the individuals whose neighbor individuals are not evaluated during the last few search iterations. In this work, we have searched the optimal individual on CIFAR-10 [10] dataset which is a very popular benchmark in NAS work, and we have evaluated the optimal architecture on CIFAR-10 by setting appropriate hyper-parameters.

This research was supported by the Construction Project of the International Center for Science and Technology Innovation of Greater Bay Area under grant 2021A0505030072.

Zhun Fan is the corresponding author and with Department of Key Lab of Digital Signal and Image Processing of Guangdong Province Engineering college, Shantou University, Shantou, Guangdong, China, ZFan@stu.edu.cn.

978-1-6654-2527-8/21/$31.00 © 2021 IEEE

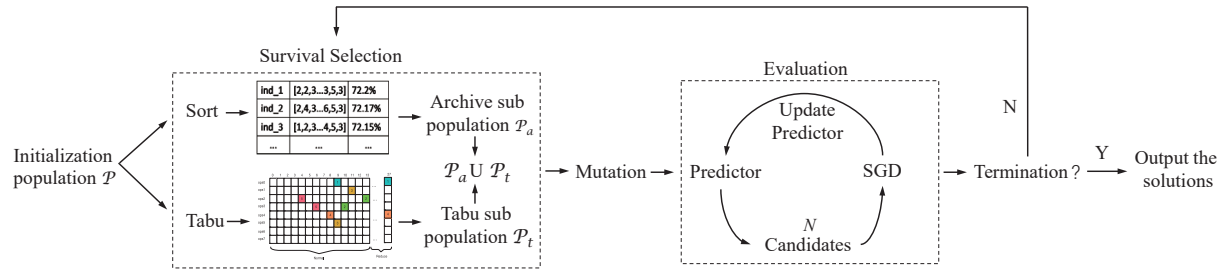


Fig. 1. The overview of proposed framework.

The extensive experimental results indicate that our proposed method is comparable to state-of-the-art methods. The key contributions of this paper are summarized as follows:

1. We combine the tabu mechanism and evolutionary algorithm in NAS, which is the first attempt in NAS work.

2. The extensive experiment results prove that the tabu mechanism can improve the performance of NAS.

II. Related Work

In recent years, with the development of NAS, searching a neural architectures for task-specific in automatically and efficiently is particulary attractive. The existing NAS methods can be broadly classified into RL, EA and gradient-based methods. For the NAS with RL methods, Zoph [11] first applied RL method in NAS. He used a recurrent neural network(RNN) to generate the model descriptions of neural networks and trained this RNN with RL on a validation set. The MetaQNN [12] search a neural network with a Q-learning agent, and the BlockQNN [13] extended this work by stacking the block that searched by the learning agent to construct the whole auto-generated network, resulting in a more general work that archives better results than its predecessor on CIFAR-10 [10] dataset. ENAS [14] improves the search efficiency by using parameter sharing mechanism and forcing all child models to share weights. For the gradient-based methods, transforming the discrete search space to be continuous space is the core. In DARTS [15], they proposed that the optimal network is the subnetwork of the supernet and used continuous architecture weights to parameterize each candidate operation, like 3×3 separable convolution, pooling and so on. GDAS [16] further sampled one operation each step rather than used the weighted sum of all operations, which can improve the search efficiency. PC-DARTS [17] searched in a sub-channel instead of operations. Beside, Amended-DARTS [18] used the mathematical methods to make the DARTS more stable in the search stage.

A. Evolutionary Neural Architecture Search

Evolutionary neural architecture search methods are usually population-based, and they can be divided into several steps, including population initialization, fitness evaluation, mutation, selecting next population and stoping criterion [19]. Genetic CNN [**?**] first applied the genetic algorithm to search the structure of neural network automatically. It used a fix-length binary string to represent a neural network, which shows the possibility of using evolutionary algorithms to search neural architectures. For example, AmoebaNet [8] used the age-based mechanism to keep the younger individuals in the population. It's the first comparison of EA and RL method in the large scale, and the work also proved that the evolutionary algorithm can get faster convergence than reinforcement learning in NAS. Recently, CARS [20] used the evolutionary algorithm to sample the neural network from the supernet. By using the weights sharing mechanism, the sampled neural networks can inherit the weights from the supernet, which can improve the search efficiency.

At the meanwhile, other evolutionary NAS methods are proposed. For example, CGP-CNN [21] used the cartesian genetic programming to design a neural network by existing blocks, including residual block, convolution block, pooling block and so on. Sun [22] applied the random forest to train a surrogate model to get the performance of the given architecture. NSGA-NET [23] used NSGA-II [24] to consider multiple objectives, including model size and model accuracy.

B. Tabu Algorithm

Tabu search algorithm [25] has been used to solved many practical problems, such as the travelling salesman problem (TSP), the vehicle routing problem

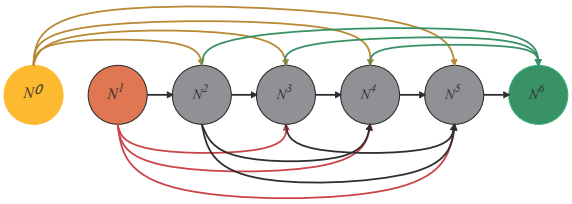


Fig. 2. An illustration of the connection in a cell. Each cell has two input nodes N^0 and N^1, and one output node N^6. The two input nodes N^0 and N^1 are the output of the two previous cells.

(VRP) and the quadratic assignment problem (QAP). The key idea of the tabu search is that it can maintain the information and knowledge such as mutation points and chosen operations about the selected solutions during the search by using the flexible memory structure called tabu list. The memory-based search mechanism can guide the local search method to continue its search and jump out from local optimum.

III. Proposed Approach

In this section, we introduce the proposed NAS approach based on tabu search and evolutionary algorithms. A pseudocode and a flowchart outlining the overall approach are shown in Algorithm 1 and Fig. 1, respectively. The details are described as follows.

Algorithm 1: Proposed Algorithm

Input : Initialized population *POP*
Output : *solutions*

```
Predictor← trainRBF(POP)
tabuList ← ∅
while n < N do
    aSubpop ← GetArchivePop(POP, nArchive)
    tSubpop ← GetTabuPop(POP, nTabu, tabuList)
    Q ← ∅
    while k < K do
        if rand() < ρ then
            p ← TournamentSelection(aSubpop)
        else
            p ← TournamentSelection(tSubpop)
        p ← Mutation(p)
        Q ← Q∪p; k ← k+1
    f ← Predictor(Q)
    c ← GetCandidate(f)
    c ← SGD(c)
    POP ← ExtendPOP(POP, c)
    tabuList ← UpdateTabuList(tabuList, Q)
    Predictor ← UpdateRBF(Predictor, c)
    n ← n+1
solutions ← GetSolutions(POP)
```

A. Search Space and Architecture Encoding

Search space is one of the most important part in NAS and a well-designed search space can improve

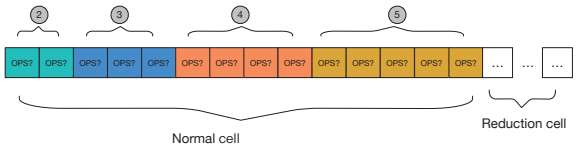


Fig. 3. Neural network is represented as a 28-integer string, the first 14-bits encode normal cell, and the last 14-bits encode reduction cell.

search efficiency significantly. Here, we adopt a cell-based search space which is very popular in recent NAS work. In this work, the neural network is stacked by two types of cells, including the normal cell and the reduction cell. To be more specific, the normal cell keeps the dimension of the output as same as that of the input. As for the reduction cell, it reduces the dimension of the input by setting the stride as two. A cell is represented by a directed acyclic graph, which can be seen in Fig. 2. Each cell contains nodes and edges. The nodes are feature maps and the edges are operations, such as convolution and pooling operations. To be more specific, each cell contain 7 nodes, the first two nodes represent the input nodes which are output from their two previous cells. The last node is represented as the output node, and the rest four nodes are defined as computation nodes. From node N^2 to N^5, the computation process is calculated according to Eq. (1). The candidate operations include skip-convolution, 3×3 max pooling, 3×3 average pooling, 3×3 separable convolution, 3×3 dilated convolution, 5×5 dilated convolution, 5×5 separable convolution and none operation.

$$I^{(j)} = \sum_{i<j} o_{i,j}\left(I^{(i)}\right) \tag{1}$$

With this search space, we adopt a 28-integer string to encode the neural network. The range of each bit is from 0 to 7, which is equal to the number of candidate operations. As shown in Fig. 3, the 28-integer string encoding has two parts, including the normal cell encoding and the reduction cell encoding. The first 14-bits represent the structure of the normal cell, and the last 14-bits represent the structure of the reduction cell. For example, in the normal encoding part, it includes four inner nodes named N^2 - N^5, each node's input comes from the previous output. The node N^2, which is a gray node in the Fig .3, has two inputs from nodes N^0 and N^1. The node's input of N^5 is form N^0 to N^4.

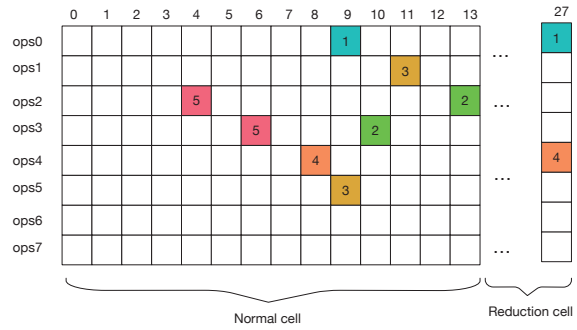


Fig. 4. An example of tabu list in this work. The number in each grid represent the tenure of current node and operation. And it will decrease to zero as iterations increase.

B. Mutation

In this work, we apply mutation to generate new architectures from parent population. We divide the mutation process into two steps. At first step, we randomly sample n bits from 28-bits encoding, the n bits named mutation points. Then, we randomly generate operations in each mutation points. More specifically, the number of n mutation points is decided by the accuracy of selected parent individual, n will be set large while the accuracy of given parent individual not performance well. On the contrary, n will be set small.

C. Tabu and Tabu list

In order to keep the better genes of the current population and reduce the useless search, we apply the tabu rule. The tabu rule can record the mutation points and operations from previous search process and save them into tabu list. From Fig .1, we can see that the selection of solutions not only considers the accuracy of individual, but also considers whether the tabu list consists of the current mutation points and operations. To be more specific, a parent population consists of two parts. One part is similar to the traditional evolutionary algorithm, which chooses the parent population by using tournament selection. The other part of the parent population is chosen by tabu algorithm which is designed in our work. The tabu algorithm consists of two stages, firstly we sort the current population according to the performance of each individual. Secondly, if an individual is better than the best found so far, the tabu rules are ignored by this individual. Otherwise we inquire the mutation points and operations of current individual in the tabu list, and ignore current individual if the mutation points and operations are used in previous search stage. We continue this process until the number of solutions is reached to a predefined size.

The tabu list is used to memory the mutation points and operations in the previous search procedure. In this work, the tabu list is a 8×28 two-dimensional array. The "8" represents the number of candidate operations, and "28" represent the 28-bits encoding. Tabu tenure is a variable to indicate the max number in the tabu list, and it decreases to zero with the increasing of the iteration times.

D. Performance Evaluation

When evaluating the performance of neural architectures, most existing methods use backpropagation algorithms. However, they are usually time-consuming. The reason is that a NAS method sometimes needs several thousand hours to search an optimal architecture. To overcome this issue, we use a Radial Basis Function (RBF) to predict the accuracy of the sampled model, which can improve the search efficiency. To train the RBF model, we sample a set of neural architectures and train them by backpropagation algorithms before the search process. In this work, we initialize the population by using the individuals which perform well in the first few search experiments. Notably, this population initialization process is carried out only once. In the search stage, we use the RBF to predict the performance of sampled neural architectures. To update and validate the performance of the predictor, we use the backpropagation algorithm to train candidate individuals which perform well in the predictor. Finally we use those candidate individuals to update the predictor. The RBF can predict the performance of given architecture in just few seconds, while the backpropagation may cost several hours to evaluate the given architecture. So the RBF improve the search process obviously.

IV. Experiments and Results Analysis

Generally, a NAS can be divided into two stages, including search stage and evaluation stage. In the search stage, we search the architecture on CIFAR-10 dataset by using surrogate model to accelerate this process. Then we select the optimal architecture and train it by SGD on CIFAR-10 dataset.

A. Experiments on CIFAR-10

In the search stage, we use the RBF surrogate model which needs to train offline and update in the search process to predict the performance of neural networks. To train and update the RBF, we use a mini batch size to train the neural networks by SGD. To be

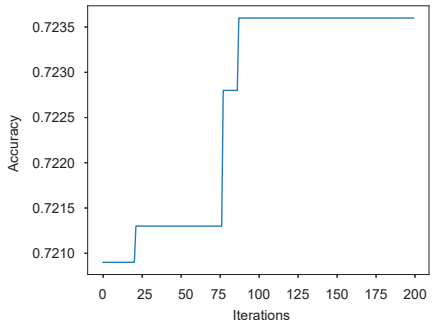


Fig. 5. The best accuracy of each iterations.

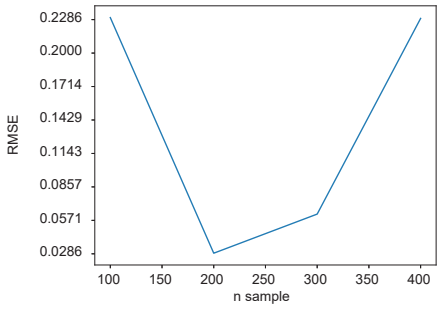


Fig. 6. The RBF performance with different sample architecture numbers on CIFAR-10.

more specific, we use the hyper parameters which are provided by Xiawu Zheng [26]. The epoch is set to 10. The batch size is set to 128. The learning rate is set to 0.003. The cell numbers is set to 6. The initialize channel is set to 8 and the image size is set to 16. Comparing with the RBF performance with different sample architectures which are shown in Fig. 6. We choose 200 architectures to train the RBF surrogate models before search process.

In this work, the tabu tenure is set to 5, and the population is set to 20. We randomly select parent individuals from two subpopulations. In this work, 10 individuals are selected by the tournament selection from the archive subpopulation, and the other 10 individuals are selected by tabu mechanism from the tabu subpopulation.

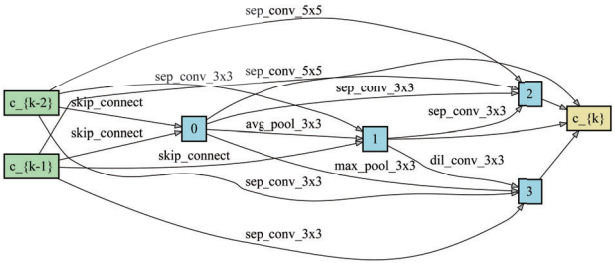
In order to verify the proposed memory strategies can improve the performance in the search stage, we search neural architecture by using only EA algorithm that the tabu subpopulation was set to 0 in the search stage. The experimental results are shown in Tab. I.

We choose the optimal architecture searched on CIFAR-10 in the search stage. At the evaluation stage, we use gradient descent to train the selected architecture with the batch size 96. To evaluate the

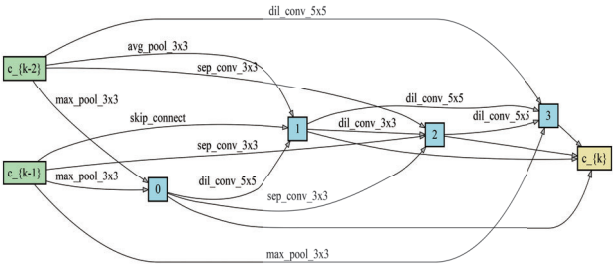
TABLE I

THE COMPARISON RESULTS OF EA AND EA THAT WITH MEMORY STRATEGIES IN THE SEARCH STAGE

Method	Accuracy (%)
EA	72.40
EA with memory strategies	72.47



(a) Normal cell found on CIFAR-10.



(b) Reduction cell found on CIFAR-10.

Fig. 7. The best normal cell and reduction cell found on CIFAR-10 in the search stage. The $c\{k\text{-}1\}$ and $c\{k\text{-}2\}$ denote the outputs of previous two cells before kth cell.

performance of the obtained architecture, a large network is constructed with 20 stacked cells and 36 initial channels. The network is trained with the same training settings as in the searching phase for 600 epochs on the complete CIFAR-10 training set. The Fig. 7 show the best normal cell and reduction that we find.

The corresponding test results on CIFAR-10 are shown in Tab. II. From the results, we can find that our proposed method can get a well trade-off between the error rate and the GPU days.

V. CONCLUSION

This paper proposed a novel neural architecture search algorithm for rapidly designing neural architectures. We combined the evolutionary algorithm and the tabu search algorithm. Two subpopulations were set in the population, which include the archive subpopulation and tabu subpopulation. The archive subpopulation and tabu subpopulation were selected by

TABLE II
RESULTS ON CIFAR-10 DATASET

Architectures	Test Error (%)	Params (M)	Search Cost (GPU days)	Search Method
NASNet-A+cutout [7]	2.65	3.3	1800	RL
ENAS + cutout [14]	2.89	4.6	4	RL
DARTS [15]	2.76	3.3	4	gradient-based
RC-DARTS [27]	2.81	3.3	1	gradient-based
AmoebaNet-A + cutout [8]	3.34	3.2	3150	EA
AmoebaNet-B + cutout [8]	2.55	**2.8**	3150	EA
CARS [20]	2.62	3.6	**0.4**	EA
NSGANet [23]	3.85	3.3	8	EA
OURS	**2.48**	4.6	2	EA

tournament selection and tabu mechanism respectively. By tournament selection we can maintain the better genes in the population, and the tabu mechanism can help us to avoid useless search. Finally, the RBF was used to predict the performance of individuals, which reduce the search time to 2 GPU days. The experimental results on CIFAR-10 dataset demonstrated that our proposed method can get comparability results than state-of-the-art methods.

REFERENCES

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: convolutional networks for biomedical image segmentation," in *Proceedings of International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, p. 91–99.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[6] J. Bracken and J. T. McGill, "Mathematical programs with optimization problems in the constraints," *Operations Research*, vol. 21, no. 1, pp. 37–44, 1973.

[7] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[8] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 4780–4789.

[9] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61–70, 2011.

[10] A. Krizhevsky, V. Nair, and G. Hinton, "Learning multiple layers of features from tiny imagess(technical report)," *University of Toronto*, 2009.

[11] I. Bello, B. Zoph, V. Vasudevan, and Q. V. Le, "Neural optimizer search with reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 70, 2017, pp. 459–468.

[12] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[13] Z. Zhong, Z. Yang, B. Deng, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "BlockQNN: efficient block-wise neural network architecture generation," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[14] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proceedings of International Conference on Machine Learning*. PMLR, 2018, pp. 4095–4104.

[15] H. Liu, K. Simonyan, and Y. Yang, "DARTS: differentiable architecture search," in *Proceedings of International Conference on Learning Representations*, 2019.

[16] X. Dong and Y. Yang, "Searching for a robust neural architecture in four gpu hours," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1761–1770.

[17] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong, "PC-DARTS: partial channel connections for memory-efficient architecture search," in *Proceedings of International Conference on Learning Representations*, 2020.

[18] K. Bi, C. Hu, L. Xie, X. Chen, L. Wei, and Q. Tian, "Stabilizing darts with amended gradient estimation on

architectural parameters," *arXiv preprint arXiv:1910.11831*, 2019.

[19] Y. Liu, Y. Sun, B. Xue, M. Zhang, and G. Yen, "A survey on evolutionary neural architecture search," *arXiv preprint arXiv:2008.10937*, 2020.

[20] Z. Yang, Y. Wang, X. Chen, B. Shi, C. Xu, C. Xu, Q. Tian, and C. Xu, "CARS: continuous evolution for efficient neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1829–1838.

[21] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in *Proceedings of the genetic and evolutionary computation conference*, 2017, pp. 497–504.

[22] Y. Sun, H. Wang, B. Xue, Y. Jin, G. G. Yen, and M. Zhang, "Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 350–364, 2019.

[23] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "NSGA-Net: neural architecture search using multi-objective genetic algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 419–427.

[24] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[25] F. Glover, "Tabu search—part i," *ORSA Journal on computing*, vol. 1, no. 3, pp. 190–206, 1989.

[26] X. Zheng, R. Ji, Q. Wang, Q. Ye, Z. Li, Y. Tian, and Q. Tian, "Rethinking performance estimation in neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 356–11 365.

[27] X. Jin, J. Wang, J. Slocum, M.-H. Yang, S. Dai, S. Yan, and J. Feng, "RC-DARTS: resource constrained differentiable architecture search," *arXiv preprint arXiv:1912.12814*, 2019.