Exploiting Operation Importance for Differentiable Neural Architecture Search

Yuan Zhou, *Senior Member, IEEE*, Xukai Xie, and Sun-Yuan Kung, *Life Fellow, IEEE*

***Abstract*— Recently, differentiable neural architecture search (NAS) methods have made significant progress in reducing the computational costs of NASs. Existing methods search for the best architecture by choosing candidate operations with higher architecture weights. However, architecture weights cannot accurately reflect the importance of each operation, that is, the operation with the highest weight might not be related to the best performance. To circumvent this deficiency, we propose a novel indicator that can fully represent the operation importance and, thus, serve as an effective metric to guide the model search. Based on this indicator, we further develop a NAS scheme for "exploiting operation importance for effective NAS" (EoiNAS). More precisely, we propose a high-order Markov chain-based strategy to slim the search space to further improve search efficiency and accuracy. To evaluate the effectiveness of the proposed EoiNAS, we applied our method to two tasks: image classification and semantic segmentation. Extensive experiments on both tasks provided strong evidence that our method is capable of discovering high-performance architectures while guaranteeing the requisite efficiency during searching.**

***Index Terms*— High-order Markov chain, image classification, neural architecture search (NAS), semantic segmentation.**

I. Introduction

DESIGNING appropriate network architecture for specific problems is a challenging task. Better network architecture usually leads to significant performance improvement. In recent years, the neural architecture search (NAS) technique [1]–[8] has demonstrated success in automating the design of neural architectures. Many architectures produced by NAS methods have achieved higher accuracy than those manually designed in tasks, such as image classification [1], [9], super-resolution [10], semantic segmentation [11]–[13], and object detection [14]. NAS methods not only boost the model performance but also free human experts from the tedious task of tweaking the architecture.

So far, three main frameworks have been predominantly used in NAS: evolutionary algorithm (EA)-based NAS [7], [16], [17], reinforcement learning (RL)-based NAS [1], [2], [18], and gradient-based NAS [6], [19], [20]. In both EA-based and RL-based approaches, the searching procedures require the validation accuracy of numerous architecture candidates, which is computationally expensive. For example, the RL method [1], [2] trains and evaluates more than 20 000 neural networks across 500 GPUs over four days. These approaches use a large number of computational resources, which is inefficient and unaffordable.

To eliminate this deficiency, gradient-based NAS methods [6], [15], [19], [20], such as DARTS [6] and GDAS [20], have been developed more recently. They construct a super network and relax the architecture representation by assigning architecture weights to the candidate operations. In DARTS, a computation cell is searched as the building block of the final architecture, and each cell is represented as a directed acyclic graph (DAG) consisting of an ordered sequence of N nodes. The concrete search space is then relaxed into a continuous one so that the architecture can be optimized with respect to its validation set performance by gradient descent. It achieved comparable performance to EA-based [16] and RL-based [1] methods while only requiring a search cost of a few GPU days. However, DARTS requires a large amount of memory because it optimizes the entire super network at each iteration. To reduce the cost of memory during the search procedure, GDAS [20] is proposed. It samples subgraphs according to the architecture weights and only optimizes one subgraph at each training iteration.

Existing gradient-based NAS methods select the candidate operations based on their architecture weights. Candidate operations with higher weights are more likely to be selected to derive the target architecture. However, we found that architecture weights did not accurately reflect the importance of each operation. To demonstrate this issue, we obtained the stand-alone architectures with different candidate operations by replacing a selected operation in a cell of the final searched architecture with all the other candidate operations, fully training them until convergence, and calculating the accuracy of each stand-alone model. The accuracy versus its corresponding architecture weights is plotted in Fig. 1. As can be seen, the operation with the highest architecture weight does not achieve the best accuracy. Fig. 2 illustrates the overall procedure of the gradient-based NAS methods.

Given the limitation of architecture weights, it is natural to ask the question: will we be able to improve architecture search performance if we apply a more effective indicator to guide the model search? To this end, we propose a simple

Manuscript received 17 March 2020; revised 6 October 2020 and 28 March 2021; accepted 31 March 2021. Date of publication 17 May 2021; date of current version 28 October 2022. This work was supported in part by the National Natural Science Foundation of China under Grant U2006211 and in part by the National Key Research and Development Project of China under Grant 2020YFC1523204. *(Corresponding author: Yuan Zhou.)*

Yuan Zhou and Xukai Xie are with the School of Electrical and Information Engineering, Tianjin University, Tianjin 300072, China (e-mail: zhouyuan@tju.edu.cn; xkxie@tju.edu.cn).

Sun-Yuan Kung is with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08540 USA (e-mail: kung@princeton.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3072950.

Digital Object Identifier 10.1109/TNNLS.2021.3072950

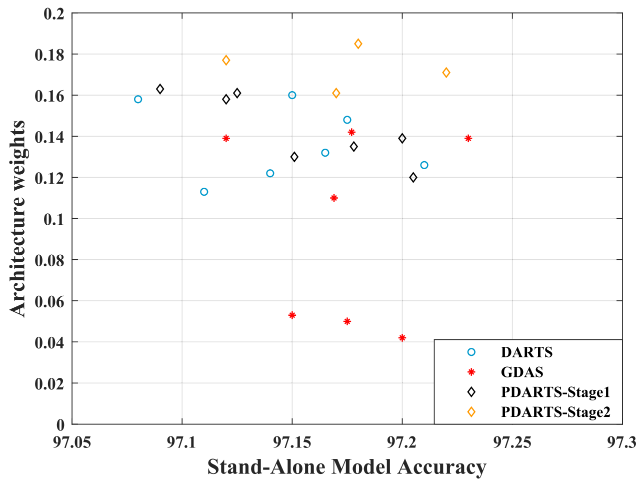


Fig. 1. Correlation between the stand-alone model accuracy and architecture weights. To obtain the stand-alone architectures with different candidate operations, we separately replace a selected operation in a cell of the final architecture with all the other candidate operations and fully train them until converge. PDARTS-Stage1 and PDARTS-Stage2 represent the different stages of the PDARTS [15] search process.

yet effective solution to NAS, termed the Exploring Operation Importance for Effective NAS (EoiNAS). The outline of our method is given as follows.

1) It has been demonstrated in [21] that operation A is better than operation B if A has fewer training epochs and higher validation accuracy during the search process. *Based on this criterion, a new indicator is proposed to fully exploit the operation importance and guide the model search*. In contrast to existing methods (such as MDENAS [21]) that use the differential of the training epochs and validation accuracy to update the architecture parameters, we calculate the ratio of validation accuracy and training epochs as an additional condition to judge the operation importance.
2) To further improve the search efficiency and accuracy, we propose a high-order Markov chain-based strategy to slim the search space. Existing methods only consider the current information to judge the operation importance during the operation pruning process, which leads to inaccurate judgments. By contrast, we use the high-order Markov chain to combine the historical importance of an operation with its current indicators to more accurately determine the importance of the operation. The most inferior operation is then pruned according to the importance calculated by the Markov chain. This process continues until only one operation remains; this operation can be regarded as the best operation to derive the final architecture. Due to this strategy, our super network exhibits better convergence than the state-of-the-art NAS methods.

The effectiveness of EoiNAS was verified on two tasks: image classification and semantic segmentation. For the image classification task, searching was performed on CIFAR-10, and evaluation was performed on both the CIFAR-10/100 and ImageNet datasets. EoiNAS could finish one searching procedure in several GPU hours and discovered a robust neural network with a test error of 2.50%. Moreover, the networks discovered on CIFAR-10 could be successfully transferred to ImageNet, achieving top-one and top-five errors of 25.6% and 8.3%, respectively. We have also demonstrated the effectiveness of our search algorithm for the task of semantic segmentation and achieved high performance even without ImageNet pretraining.

The remainder of this article is organized as follows. In Section II, we review the related recent work on NAS algorithms. In Section III, we describe our proposed operation importance indicator (OII) and the high-order Markov chain-based search space slimming (SSS) strategy. Section IV presents the experimental results of our proposed methods on the classification task and analyzes their effectiveness by comparing them with previous NAS algorithms. Section V demonstrates the effectiveness of our search algorithm on a semantic segmentation task. Finally, our conclusions are presented in Section VI.

II. Related Works

With the rapid development of deep learning, significant performance gains have been brought to a wide range of computer vision problems, for example, image classification and semantic segmentation, most of which are owed to manually designed network architectures [22]–[30]. Recently, a new research field named NAS [1]–[5], [31], [32] has been attracting increasing attention. The goal of NAS is to find automatic ways of designing neural architectures to replace conventional handcrafted ones. According to the heuristics of exploring a large architecture space, existing NAS approaches can be roughly divided into three categories, namely, EA-based approaches [7], [16], [17], RL-based approaches [1], [2], [18], [33] and gradient-based approaches [6], [19], [20].

A. Reinforcement Learning-Based NAS

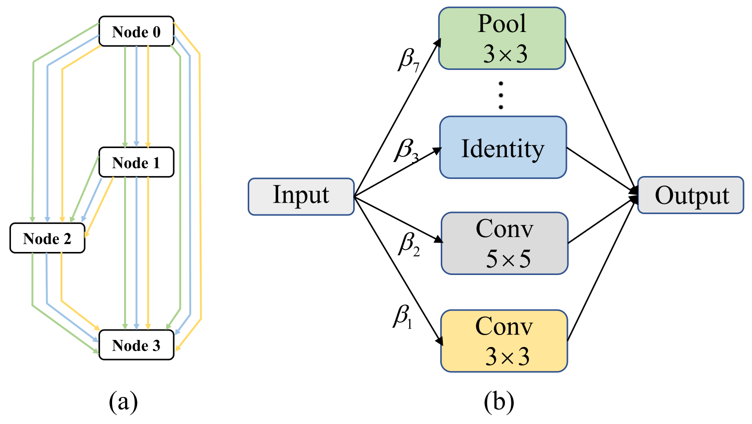
An RL-based approach was proposed by Zoph and Le [1] and Zoph *et al.* [2] for NAS. They used a recurrent network as a controller to generate the model description of a child neural network designated for a given task. The resulting architecture (NASNet) improved over the existing handcrafted network models at the time.

B. Evolutionary Algorithm-Based NAS

An alternative search technique was proposed by Real *et al.* [7] where an evolutionary (genetic) algorithm was used to find a neural architecture tailored for a given task. The evolved neural network (AmoebaNet) had an improved performance over NASNet. Although these works achieved state-of-the-art results on various classification tasks, their main disadvantage was the large number of computational resources that they demanded.

C. Gradient-Based NAS

In contrast to treating architecture search as a black-box optimization problem, gradient-based NAS methods [6], [15], [19]–[21], [34]–[38] utilize the gradient obtained in the training process to optimize neural architecture. DARTS [6] relaxed



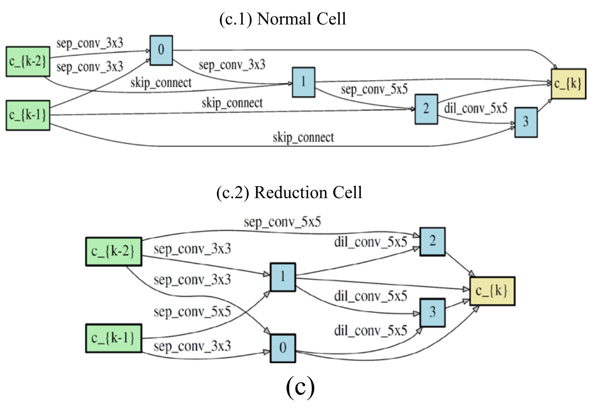


Fig. 2. Illustration of the gradient-based NAS procedure. (a) Cell structure is represented by a DAG. (b) Candidate operations between every two nodes are assigned architecture weights to represent their importance. (c) On each edge, the candidate operation with the highest weight is selected to derive the target architecture (selected architectures).

the search space to be continuous so that the architecture could be optimized with respect to its validation set performance by gradient descent. Therefore, gradient-based approaches successfully accelerate the architecture search procedure, and only several GPU days are required. Because DARTS optimizes the entire super network during the search process, it may suffer from discrepancies between the continuous architecture encoding and the derived discrete architecture. GDAS [20] has been suggested as an alternative method to alleviate this discrepancy. GDAS approaches the search problem as sampling from a distribution of architectures, where the distribution itself is learned in a continuous way. The distribution is expressed via slack softened one-hot variables that multiply the operations and make the sampling procedure differentiable. SNAS [19] applied a similar technique to constrain the architecture parameters to be one-hot in order to tackle the inconsistency in optimizing objectives between search and evaluation scenarios. In addition, MdeNAS [21] proposed a multinomial distribution learning method for extremely effective NAS, which considered the search space as a joint multinomial distribution. The distribution is optimized to have high-performance expectations. SGAS [37] greedily chooses and prunes candidate operations using heuristic criteria that take the edge importance, the selection certainty, and the selection stability into consideration. Such an approach alleviates the effect of the degenerate search-evaluation correlation problem and reflects the true ranking of architectures. StacNAS [38] points out that the correlation of similar operators is the contributing factor to the issue of the ranking and then introduces a new hierarchical search algorithm via operator clustering to differentiable NAS.

III. Methodology

We first describe our search space and continuous relaxation in general form in Section III-A, where the computation procedure for an architecture is represented as a DAG. We then propose a new indicator to fully exploit the importance of each operation in Section III-B. Finally, we design a high-order Markov chain-based SSS strategy to make the super network exhibit fast convergence and high training accuracy in Section III-C.

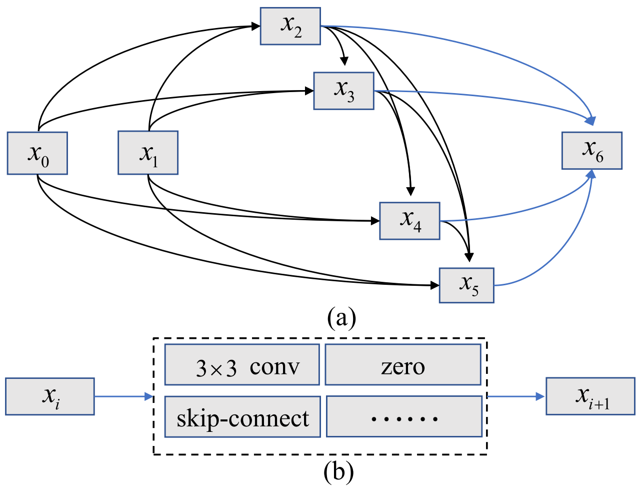


Fig. 3. Search space. (a) Cell contains seven nodes, two input nodes, four intermediate nodes that apply sampled operations on the input nodes and upper nodes, and an output node that concatenates the outputs of the four intermediate nodes. (b) Edge between two nodes denotes a possible operation sampled according to the discrete probability distribution $P\{O = o_n\}$ in the search space (candidate operation set).

A. Search Space and Continuous Relaxation

In this work, we leverage GDAS [20] as our baseline framework. Our goal is to search for optimal cells and construct a network comprising L cells. A cell is defined as a DAG of N nodes, where each node is a network layer, that is, performing a specific mathematical function. In this work, N was set to 7. As shown in Fig. 3, a cell contains seven nodes, including two input nodes, four intermediate nodes that apply sampled operations on the input nodes and upper nodes, and an output node that concatenates the outputs of the four intermediate nodes. We denote the operation space as O, where each element represents a candidate operation $o(\cdot)$. We denote the feature map of node i as x_i; the feature maps of the N nodes in a cell can then be represented as $\{x_0, x_1, \ldots, x_{N-1}\}$. An edge $f_{i,j}$ represents the information flow connecting nodes i and j, which consists of a set of operations weighted by the architecture weights $\beta_{i,j}$ and is, thus, formulated as

$$f_{i,j}(x_i) = \sum_{o\in O} \beta^{o}_{i,j} o(x_i) \tag{1}$$

$$\text{s.t. } \beta^{o}_{i,j} = \frac{\exp\left(\alpha^{o}_{i,j}\right)}{\sum_{o'\in O} \exp\left(\alpha^{o'}_{i,j}\right)} \tag{2}$$

where $\alpha^{o}_{i,j}$ is the oth element of an $|O|$-dimensional learnable vector $\alpha_{i,j} \in \mathbb{R}^{|O|}$, and $\beta_{i,j}$ encodes the sampling distribution of the function between nodes i and j, as discussed below. Intuitively, a well-learned $\boldsymbol{\beta} = \{\beta^{o}_{i,j}\}$ represents the relative importance of operation o for transforming the feature map of node i to that of node j. Similar to GDAS, between nodes i and j, we sample one operation from O according to a discrete probability distribution $P\{O = o(\cdot)\} = \beta^{o}_{i,j}, o \in O$. During the search, we calculate the feature map of each node in a cell as

$$x_j = \sum_{i<j} f_{i,j}(x_i) \tag{3}$$

where $f_{i,j}$ is sampled from candidate operation set O.

Because operation $f_{i,j}$ is sampled according to a discrete probability distribution, we cannot backpropagate gradients to optimize $\alpha_{i,j}$. To allow backpropagation, we use the Gumbel–Max trick [27], [39] and the softmax function [40] to reformulate (3) and (4), which provides an efficient way to draw samples from a discrete probability distribution in a differentiable manner

$$x_j = \sum_{i=1}^{j-1} \sum_{o=1}^{|O|} h^{o}_{i,j} f^{o}_{i,j}\left(x_i; w^{o}_{i,j}\right) \tag{4}$$

$$\text{s.t. } h^{o}_{i,j} = \frac{\exp\left(\left(\alpha^{o}_{i,j} + \tau_o\right)/T\right)}{\sum_{o'=1}^{|O|} \exp\left(\left(\alpha^{o'}_{i,j} + \tau_{o'}\right)/T\right)}. \tag{5}$$

Here, τ_o are i.i.d. samples drawn from Gumbel (0,1), $f^{o}_{i,j}$ indicates the oth function in O, $h^{o}_{i,j}$ is the oth element of $h_{i,j}$, $w^{o}_{i,j}$ is the weight of $f^{o}_{i,j}$ between node i and j, and T is the temperature parameter [27], which controls the Gumbel–Softmax distribution. As the parameter T approaches zero, the Gumbel–Softmax distribution becomes equivalent to the discrete probability distribution. T is annealed from 5.0 to 0.0 during our search procedure.

We use the same candidate function set (O) as in [20] for a fair comparison. Our candidate operation set O contains the following eight operations: 1) identity; 2) zero; 3) 3×3 separable convolutions; 4) 3×3 dilated separable convolutions; 5) 5×5 separable convolutions; 6) 5×5 dilated separable convolutions; 7) 3×3 average pooling; and 8) 3×3 max pooling. We searched for two kinds of cells, namely, normal cell and reduction cell. When searching for the normal cell, each operation in O has a stride of one. For the reduction cell, the stride of operations is set as two on input nodes and one on other nodes.

To obtain the best cells, we optimize the network parameters $\boldsymbol{w}$ and architecture parameters $\boldsymbol{\alpha}$ by alternating gradient descent performed on the training and validation sets, respectively. This optimization procedure is defined as a bilevel optimization problem

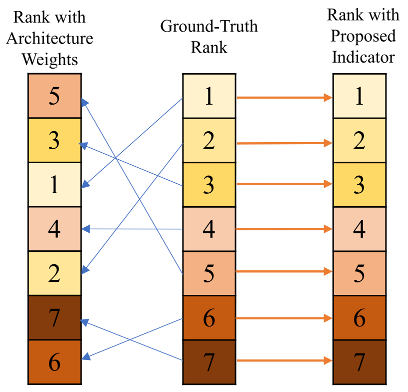


Fig. 4. We visualize the corresponding importance of each operation by the colors and numbers. A change in ranking occurs between architecture weight ranking and ground truth and our proposed indicator can accurately reflect the operation importance. Yang *et al.* [41] have shown that, for many NAS methods, the cell-based search space has a very narrow accuracy range such that the seed has a considerable impact on architecture rankings. We have therefore run EoiNAS four times with different random seeds to obtain four different models and picked the best model based on its validation performance.

$$\min_{\boldsymbol{\alpha}} \; L_{\text{valid}}(\boldsymbol{\alpha}, \boldsymbol{w}^*(\boldsymbol{\alpha})) \tag{6}$$

$$\text{s.t. } \boldsymbol{w}^*(\boldsymbol{\alpha}) = \operatorname*{argmin}_{\boldsymbol{w}} \; L_{\text{train}}(\boldsymbol{\alpha}, \boldsymbol{w}) \tag{7}$$

where L_{valid} is the validation loss and L_{train} is the training loss. Once we discover the best normal cell and reduction cell, we stack copies of these best cells to make up a neural network.

B. Operation Importance Indicator

In the existing gradient-based NAS method, operation importance is directly ranked by the architecture weights β, which is supposed to represent the relative importance of a candidate operation versus the other operations. When the search process is completed, the most important operation is selected, and other inferior operations are pruned according to the value of the architecture weights.

However, architecture weights cannot accurately reflect the importance of each operation, as discussed in Section I, and Fig. 4 illustrates this issue: a change in ranking occurs between architecture weight ranking and the true one.

1) Proposed Indicator: It has been demonstrated in [21] that operation A is better than operation B if A has fewer training epochs and higher validation accuracy during the search process. The absolute validation accuracy of each operation is not able to truly reflect the operation importance because one operation can obtain better performance if it is trained by more iterations. Therefore, the training epochs need to be combined with the validation accuracy to calculate a relative accuracy parameter. Specifically, the accuracy parameter is defined as the ratio of validation accuracy and training epochs, which is used as an additional condition to judge the operation importance. The accuracy parameter is represented as

$$C^{o}_{i,j} = \frac{A^{o}_{i,j}}{E^{o}_{i,j}} \tag{8}$$

where $A_{i,j}^o$ is the validation accuracy, and $E_{i,j}^o$ is the training epoch of each operation on each edge. We also use softmax to normalize the accuracy parameter and then obtain the accuracy weight of each operation as

$$S_{i,j}^o = \frac{\exp\left(C_{i,j}^o\right)}{\sum_{o' \in O} \exp\left(C_{i,j}^{o'}\right)}. \tag{9}$$

The accuracy weight is used as an additional operation importance judgment condition. Based on the architecture weight $\beta_{i,j}^o$ and accuracy weight $S_{i,j}^o$, we obtain a more effective indicator $\boldsymbol{I}$, as shown in (10), which can fully exploit the importance of each operation

$$I_{i,j}^o = S_{i,j}^o + \lambda \beta_{i,j}^o \tag{10}$$

where λ is a parameter to balance the two contributions, which is set to 0.5 in this work.

2) Analysis and Comparison: As shown in Fig. 4, compared to DARTS [6] and GDAS [20] that rank the operation importance directly by architecture weight, our proposed indicator can accurately reflect the operation importance, which can help to select the optimal operation and consequently achieve the highest accuracy. Compared to SGAS [37] that ranks the operations according to three selection criteria, our proposed indicator can more directly exploit the operation importance and guide the model search. Compared to StacNAS [38], our search strategy can combine the historical importance of an operation when ranking the operation importance, so the search results are robust.

We then analyze the difference between our search method and MdeNAS [21]. In the learning procedure of MdeNAS, the architecture parameters are updated through the differential of the training epochs and accuracy such that the probability of a bad operation is transferred to better operations. In our work, the architecture parameters are updated by stochastic gradient descent; the accuracy parameter determined by the ratio of the accuracy and training epochs is used as an additional operation importance judgment condition to better reflect the operation importance.

Based on this new indicator $\boldsymbol{I}$ and high-order Markov chain model, a SSS strategy is proposed during the search process to further improve the search efficiency and accuracy, as we will discuss next.

C. High-Order Markov Chain-Based Search Space Slimming Strategy

To further improve the search efficiency and accuracy, we propose a high-order Markov chain-based strategy to slim the search space. Existing operation pruning algorithms use architecture parameters to judge the importance of candidate operations during the training process [15]. However, the architecture parameters always fluctuate when the supernet has not yet stably converged; in this case, the architecture parameters cannot accurately reflect the operation importance.

To this end, we propose a new mechanism to better reflect the importance of the operation in order to make the operation pruning more robust. More specifically, we use a high-order Markov chain to synthesize the historical information from the previous iterations. Compared to the previous methods that only consider the current information to judge the operation importance during the operation pruning process [15], our high-order Markov chain-based SSS strategy can combine the historical importance of an operation with its current indicator, which determines the importance of the operation more accurately and robustness.

We denote the training of every k epochs as a step. In each step, we use an n-order Markov chain model to incorporate historical information to judge the operation importance more accurately. Specifically, we define two states for each candidate operation between every two nodes, namely, Live(L) and Die(D). Each candidate operation has a state transition matrix Q to update its corresponding state. The definite n-order Markov chain model is defined in (11)

$$X_i = \sum_{m=1}^{n} \sigma_m X_{i-m} Q_{i-m} \tag{11}$$

$$\text{s.t. } \sigma_m \geq 0, \quad \sum_{m=1}^{n} \sigma_m = 1 \tag{12}$$

where $X_i = (l_i, d_i)$ is the operation state at the ith iteration, and l_i and d_i represent the probability of L and D, respectively. $Q_i = \begin{pmatrix} q_{LL_i} & q_{LD_i} \\ q_{DL_i} & q_{DD_i} \end{pmatrix}$ is the corresponding state transition matrix, where q_{LL_i} and q_{DD_i} represent the probability of staying in the same state, q_{LD_i} represents the probability of moving from the L state to the D state, and q_{DL_i} represents the probability of moving from the D state to the L state. We set the initial operation state $X_0 = (l_0, d_0) = (0.5, 0.5)$ and the initial state transition matrix $Q_0 = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix}$.

We define a rule for updating the state transition matrix Q. As the indicator of importance increases, we increase q_{LL_i} and decrease q_{LD_i} both by ε_1, simultaneously increasing q_{DL_i} and decreasing q_{DD_i} by ε_2. Conversely, when the indicator decreases, we carry out the opposite updates. The operation with the highest D probability after updating is considered the most inferior operation. At each step, we judge the most inferior of the remaining operations and prune it. This process continues until only one operation remains. This operation can be regarded as the best operation to derive the final architecture.

Our searching algorithm is presented in Algorithm 1. In the first 20 epochs, we only perform gradient descent-based optimization over the network parameters. It helps to obtain architecture weights that are able to balance parameterized operations (e.g., convolution operation) and nonparameterized operations (e.g., skip-connect operation). Then, we jointly optimize the architecture parameters $\boldsymbol{\alpha}$ and network parameters $\boldsymbol{w}$ in an alternating manner via stochastic gradient descent with first-order approximation. Specifically, in an iterative manner, we optimize the network parameters by descending $\nabla_{\boldsymbol{w}} L_{\text{train}}(\boldsymbol{\alpha}, \boldsymbol{w}; T)$ on the training set and optimize the architecture parameters by descending $\nabla_{\boldsymbol{\alpha}} L_{\text{val}}(\boldsymbol{\alpha}, \boldsymbol{w}; T)$ on the validation set. The number of training epochs and the accuracy of validation were recorded to calculate the important indicator for each operation. The operation state X is then

Algorithm 1 Efficient NAS

```
Input: Training set: $D_T$; Validation set: $D_V$;
       Operation set: $O$
Init: Network parameters: $\boldsymbol{w}$; Architecture parameters: $\boldsymbol{\alpha}$;
      Validation accuracy: $\boldsymbol{A}$; Training epochs: $\boldsymbol{E}$;
      Operation state: $X$; State transition matrix: $Q$;
      Temperature parameter: $T$
1: while not converge do
2:     Sample a sub-graph to train according to Eq. (5);
3:     Update $\boldsymbol{w}$ by descending $\nabla_{\boldsymbol{w}} L_{train}(\boldsymbol{\alpha}, \boldsymbol{w}; T)$ on $D_T$;
4:     Update $\boldsymbol{\alpha}$ by descending $\nabla_{\boldsymbol{\alpha}} L_{valid}(\boldsymbol{\alpha}, \boldsymbol{w}; T)$ on $D_V$;
5:     Update $\boldsymbol{A}$, $\boldsymbol{E}$;
6:     Calculate operation importance $\boldsymbol{I}$ by Eq. (10);
7:     Update each operation state $X$ by Eq. (11);
8:     if epoch > 20 and epoch % 20 == 0 then
9:         Prune operation according to its state on each edge;
10:    end if
11: end while
12:     end if
        end while
13: Derive the final architecture based on the indicator $\boldsymbol{I}$;
14: Optimize the architecture on the training set
```

updated based on the high-order Markov chain-based SSS strategy. An operation will be pruned after 20 epochs if its corresponding Die probability is the highest. When the search procedure is completed, we derive the best cells by retaining two edges with the largest values of indicator I for each node in the cell.

IV. Experiments on Classification Task

To verify the effectiveness of our method, we applied EoiNAS to two vision tasks: image classification and semantic segmentation. This section focuses on the task of image classification.

We conducted experiments on three popular image classification datasets: CIFAR-10, CIFAR-100 [50], and ImageNet [51]. An architecture search was performed on CIFAR-10, and the discovered architectures were evaluated on all three datasets.

A. Datasets

Both CIFAR-10 and CIFAR-100 contain 50k training and 10k testing RGB images with a fixed spatial resolution of 32×32. These images are equally distributed over ten classes and 100 classes in CIFAR-10 and CIFAR-100, respectively. In the architecture search scenario, the training set is equally split into two subsets: one for updating the network parameters and the other for updating the architecture parameters. In the evaluation scenario, a standard training/testing split is used.

We used ImageNet to test the transferability of the architectures discovered on CIFAR-10. Specifically, we used a subset of ImageNet, ILSVRC2012, which contains 1000 object categories and 1.28M training and 50k validation images. Following conventions used in [2] and [6], we applied the mobile setting where the input image size is 224×224.

B. Implementation Details

Following the pipeline in GDAS [20], our experiments consisted of three stages. First, EoiNAS was applied to search for the best normal/reduction cells on CIFAR-10. Then, a larger network was constructed by stacking the learned cells and retrained on both CIFAR-10 and CIFAR-100. The performance of EoiNAS was compared with other state-of-the-art NAS methods. Finally, we transferred the cells learned on CIFAR-10 to ImageNet to evaluate their performance on larger datasets.

1) Network Configurations for Searching: The neural cells for CNN were searched on CIFAR-10 following the procedures used in [6], [19], and [20]. The candidate function set O had eight different candidate operations, as introduced in Section III-A. By default, we trained a small network of eight cells for 160 epochs in total and set the number of initial channels in the first convolution layer as 16. Cells located at 1/3 and 2/3 of the total depth of the network are reduction cells, in which all the operations adjacent to the input nodes are of stride two.

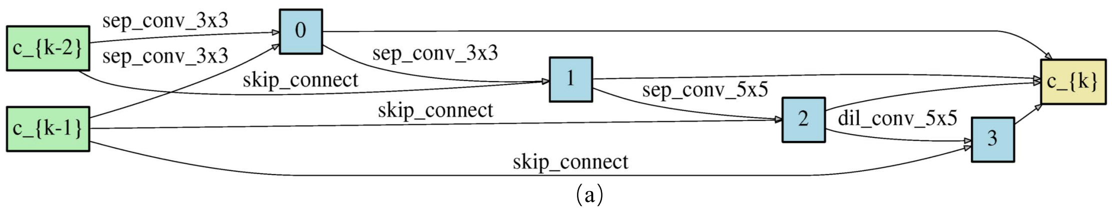
2) Parameter Settings for Searching: For the network parameters w, we used the SGD optimization. We started with a learning rate of 0.025 and reduced it to 0.001 following a cosine schedule. We used a momentum of 0.9 and a weight decay of 0.0003. For architecture parameter α, we used zero initialization, which implied an equal amount of attention over all possible operations. We used Adam optimization [56] with a learning rate of 0.0003, momentum (0.5; 0.999), and a weight decay of 0.001. To control the temperature parameter T of the Gumbel Softmax in (5), we used an exponentially decaying schedule. T was initialized as 5 and finally reduced to 0.

At the beginning of the search process, we set the hyperparameters $\sigma_1 = \sigma_2 = \sigma_3 = \sigma_4 = 0.25$ in the high-order Markov chain model. This is because the operation states fluctuate greatly at the beginning, and we cannot determine which historical state has more impact on the current state, so the values of σ_n were set to equal. With the progress of operation pruning, we gradually increased the value of σ_n that was close to the current state. The value of ϵ_1 was set to 0.02, and the value of ϵ_2 was set to 0.01. We chose ϵ_1 and ϵ_2 using the grid search, which could update the transition matrix in the best way and then obtain the most accurate operation states.

Following [20], we ran EoiNAS four times with different random seeds to obtain four different models and picked the best model based on its validation performance. For the best model, we conducted ten training runs from scratch and took the average value to obtain the results. The averaging operation eliminated noise and randomness during training. Our EoiNAS took approximately 0.6 GPU days to finish the search procedure on a single NVIDIA 1080Ti GPU. The best cells searched by EoiNAS are shown in Fig. 5.

C. Results on CIFAR-10 and CIFAR-100

For the evaluation of CIFAR, we built a network with 20 cells, as shown in Fig. 6(a), and trained it by 600 epochs with a batch size of 128. Cutout regularization [57] of length 16, drop path of probability 0.3, and auxiliary towers



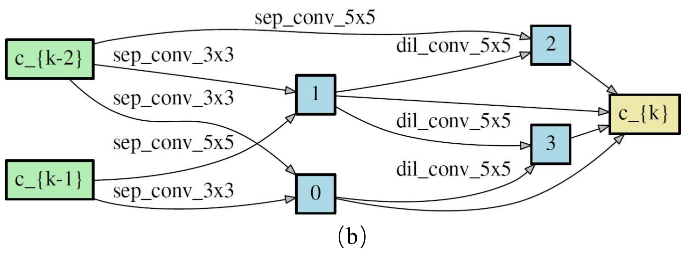


Fig. 5. Detailed structure of the best cells discovered on CIFAR-10 by our EoiNAS. (a) Normal cell. (b) Reduction cell. The definition of the operations on the edges is described in Section III-A. In the normal cell, the stride of each operation is one. For the reduction cell, the stride of operations is set as two on input nodes and one on other nodes.

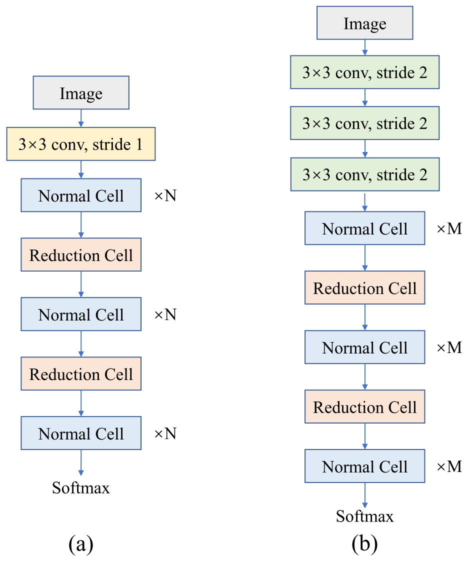


Fig. 6. Strategy to design (a) CIFAR-10/100 architecture and (b) ImageNet architecture based on the discovered normal and reduction cells.

of weight 0.4 [1] were applied. A standard SGD optimizer with a weight decay of 0.0003 and a momentum of 0.9 was used. The initial learning rate was 0.025, which was decayed to 0 following the cosine rule.

Evaluation results and comparison with state-of-the-art approaches are summarized in Table I. As demonstrated in Table I, EoiNAS achieved test errors of 2.50% and 17.3% on CIFAR-10 and CIFAR-100, respectively, with a search cost of only 0.6 GPU days. To obtain the same performance, AmoebaNet [7] spent four orders of magnitude more computational resources (0.6 GPU days versus 3150 GPU days). Our EoiNAS also outperformed GDAS [20] and SNAS [19] by a large margin. In addition, we compared our method to random search (RS) [6], which is considered to be a very strong baseline. Note that the accuracy of the model searched by EoiNAS was 0.7% higher than that of RS.

D. Results on ImageNet

The ImageNet dataset was used to test the transferability of architectures discovered on CIFAR-10. We adopted the same network configurations as GDAS [20]. As shown in Fig. 6(b), a network of 14 cells was constructed and trained by 250 epochs with batch size 128 on a single NVIDIA 1080Ti GPU, which took 12 days with the PyTorch [58] implementation. The network parameters were optimized by using an SGD optimizer with an initial learning rate of 0.1, a momentum of 0.9, and a weight decay of 3×10^{-5}. Additional enhancements, including label smoothing and auxiliary loss towers, were applied during training.

The evaluation results and comparison with state-of-the-art approaches are summarized in Table II. On ImageNet, the network achieved top-one and top-five errors of 25.6% and 8.3%, respectively. The architecture discovered by EoiNAS outperformed that discovered by GDAS by a large margin in terms of classification accuracy and model size. This demonstrates the transfer capability of the discovered architecture from a small dataset to a large dataset.

E. Ablation Studies

In addition, we have conducted a series of ablation studies that validate the importance and effectiveness of the proposed high-order Markov chain-based SSS strategy and an OII incorporated in the design of EoiNAS.

1) Search Space Slimming strategy: In Table III, we show ablation studies on CIFAR-10. The SSS refers to the high-order Markov chain-based SSS strategy, and OII is the proposed OII. All architectures were trained by 600 epochs. As shown by the results, our super network exhibits fast convergence and high training accuracy due to the high-order Markov chain-based SSS strategy.

2) Operation Importance Indicator: Table III also demonstrates the effectiveness of the proposed OII. The proposed indicator can better judge the importance of each operation and achieve higher accuracy. These results reveal the necessity of the OII. Furthermore, operations with the same OII are equally

TABLE I
CLASSIFICATION ERRORS OF OUR EoiNAS AND BENCHMARKS ON CIFAR-10 AND CIFAR-100

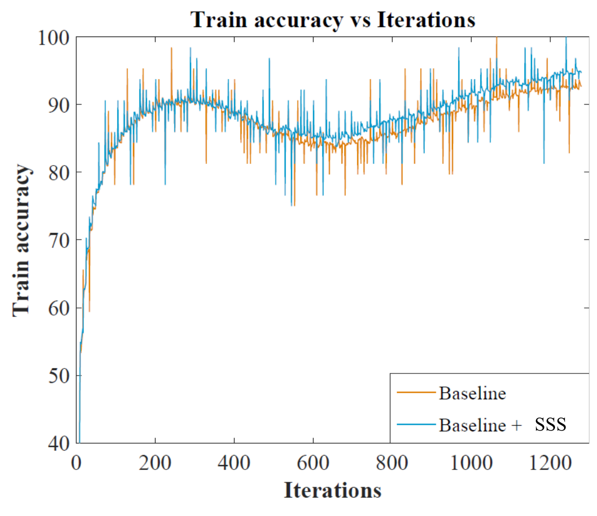
Type	**Architecture**	**GPUs**	**Times** (days)	**Params** (million)	**Test Error** C10(%)	C100(%)	**Search Method**
Human expert	ResNet + CutOut [22]	–	–	1.7	4.61	22.10	manual
	DenseNet [23]	–	–	25.6	3.46	17.18	manual
	SENet [42]	–	–	11.2	4.05	–	manual
Neural architecture search	MetaQNN [43]	10	8-10	11.2	6.92	27.14	RL
	NAS [1]	800	21-28	7.1	4.47	–	RL
	NAS + more filters [1]	800	21-28	37.4	3.65	–	RL
	NASNet-A [2]	450	3-4	3.3	3.41	–	RL
	NASNet-A + CutOut [2]	450	3-4	3.3	2.65	–	RL
	ENAS [18]	1	0.45	4.6	3.54	19.43	RL
	ENAS + CutOut [18]	1	0.45	4.6	2.89	–	RL
	Net Transformation [3]	5	2.0	19.7	5.70	–	RL
	SMASH [44]	1	1.5	16.0	4.03	–	RL
	AmoebaNet-A + CutOut [7]	450	7.0	3.2	3.34	18.93	evolution
	AmoebaNet-B + CutOut [7]	450	7.0	2.8	2.55	–	evolution
	Hierarchical NAS [16]	200	1.5	61.3	3.63	–	evolution
	CARS + CutOut [45]	1	0.4	3.0	2.86	–	evolution
	NSGANet [46]	1	8.0	3.3	3.85	–	evolution
	Progressive NAS [5]	100	1.5	3.2	3.63	19.53	SMBO
	DARTS (1st) + CutOut [6]	1	0.38	3.3	3.00	17.76	gradient-based
	DARTS (2nd) + CutOut [6]	1	1.0	3.4	2.82	17.54	gradient-based
	BayesNAS + cutout [47]	1	0.2	3.4	2.81	–	gradient-based
	SNAS + CutOut [19]	1	1.5	2.9	2.98	–	gradient-based
	GDAS [20]	1	1.0	3.4	3.87	19.68	gradient-based
	GDAS + CutOut [20]	1	1.0	3.4	2.93	18.38	gradient-based
	GDAS(FRC) + CutOut [20]	1	0.8	**2.5**	2.82	18.13	gradient-based
	GHN + CutOut [48]	1	0.84	5.7	2.84	–	gradient-based
	DATA + cutout [49]	1	1.0	3.4	2.59	–	gradient-based
	NAONet [8]	200	1.0	10.6	3.18	–	NAO
	MdeNAS + CutOut [21]	1	**0.16**	3.61	2.55	–	MDL
	Random Search + CutOut [6]	1	4.0	3.2	3.29	–	random
	EoiNAS	1	0.6	3.4	3.42	18.4	gradient-based
	EoiNAS + CutOut	1	0.6	3.4	**2.50**	**17.3**	gradient-based

TABLE II
COMPARISON WITH THE STATE-OF-THE-ART NETWORKS ON IMAGENET. ALL THE NAS NETWORKS ARE SEARCHED ON CIFAR-10 AND THEN DIRECTLY TRANSFERRED TO IMAGENET

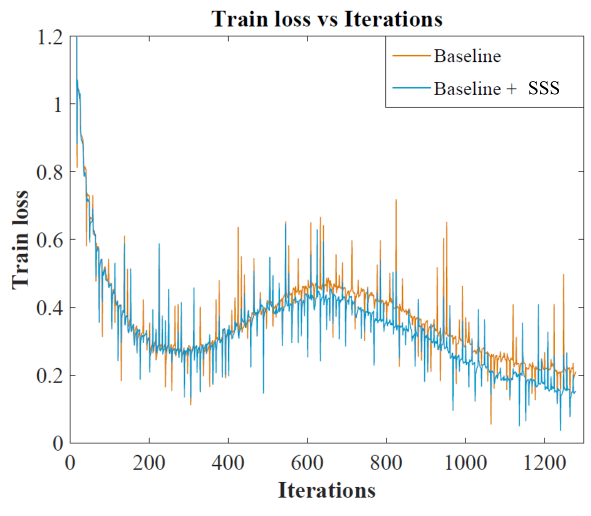
Type	**Architecture**	**GPUs**	**Times** (days)	**Params** (million)	**MAdds** (million)	**Test Error** (%) Top-1	Top-5	**Search Method**
Human expert	Inception-v1 [27]	–	–	6.6	1448	30.2	10.1	manual
	MobileNet-V1 [52]	–	–	4.2	569	29.4	10.5	manual
	MobileNet-V2 [53]	–	–	3.4	300	28.0	9.0	manual
	ShuffleNet-V1 [54]	–	–	5.0	524	29.1	9.2	manual
	ShuffleNet-V2 [30]	–	–	5.0	524	26.3	–	manual
	CondenseNet [29]	–	–	4.8	529	26.2	–	manual
Neural architecture search	NASNet-A [2]	450	3-4	5.3	564	26.0	8.4	RL
	NASNet-B [2]	450	3-4	5.3	**488**	27.2	8.7	RL
	NASNet-C [2]	450	3-4	4.9	558	27.5	9.0	RL
	AmoebaNet-A [7]	450	7.0	5.1	555	25.5	8.0	evolution
	AmoebaNet-B [7]	450	7.0	5.3	555	26.0	8.5	evolution
	AmoebaNet-C [7]	450	7.0	6.4	570	**24.3**	**7.6**	evolution
	CARS [45]	1	0.4	4.4	510	26.3	8.4	evolution
	Progressive NAS [5]	100	1.5	5.1	588	25.8	8.1	SMBO
	MdeNAS [21]	1	**0.16**	6.1	596	25.5	7.9	MDL
	GHN [48]	1	0.84	6.1	569	27.0	8.7	gradient-based
	DARTS (2nd) [6]	1	1.0	4.9	595	26.9	9.0	gradient-based
	PARSEC [36]	1	1.0	5.6	548	26.0	8.4	gradient-based
	BayesNAS [47]	1	0.2	**3.9**	–	26.5	8.9	gradient-based
	SNAS [19]	1	1.5	4.3	522	27.3	9.2	gradient-based
	GDAS [20]	1	1.0	5.3	581	26.0	8.5	gradient-based
	GDAS(FRC) [20]	1	0.8	4.4	497	27.5	9.1	gradient-based
	SETN [55]	1	1.8	5.2	597	26.7	8.6	gradient-based
	EoiNAS	1	0.6	5.0	570	25.6	8.3	gradient-based

important. We performed the corresponding experiments and found that, if the OII of the two candidate operations is the same, we can obtain a similar accuracy by pruning either of them.

We further show the loss and accuracy curves during the search process in Figs. 7 and 8. Fig. 7 demonstrates the effectiveness of SSS, while Fig. 8 proves the validity of OII. As discussed above, SSS and OII can substantially speed up

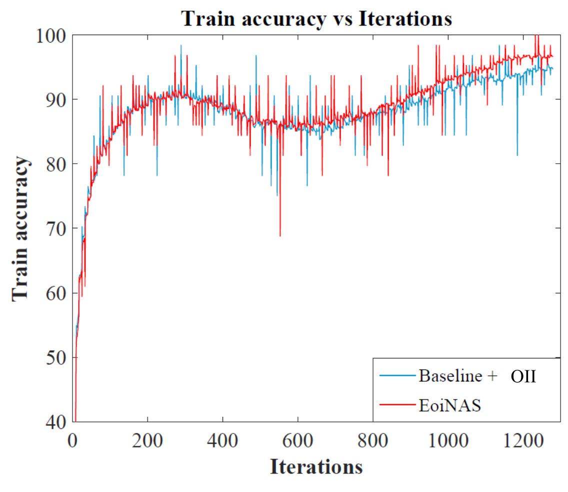


(a)

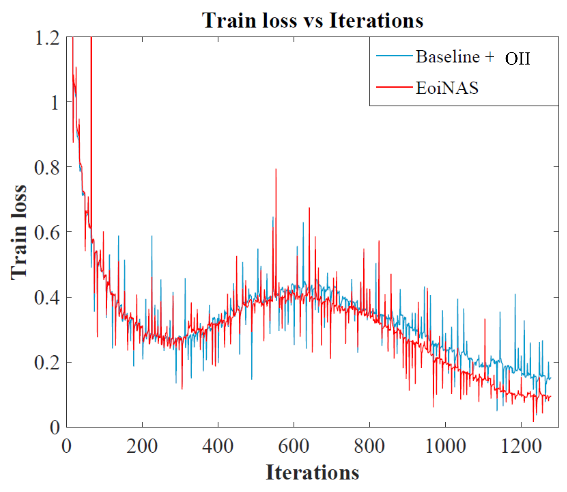


(b)

Fig. 7. Accuracy and loss curves with and without SSS. This figure compares (a) train accuracies and (b) train losses during the search stage with and without SSS.



(a)



(b)

Fig. 8. Accuracy and loss curves with and without OII. This figure compares (a) train accuracies and (b) train losses during the search stage with and without OII.

TABLE III
ABLATION STUDIES ON CIFAR-10. THE SSS MEANS THE HIGH-ORDER MARKOV CHAIN-BASED SSS STRATEGY; THE OII MEANS THE PROPOSED OII. ALL ARCHITECTURES ARE TRAINED BY 600 EPOCHS

Architecture	**SSS**	**OII**	**Times** (days)	**Params** (million)	**Error** (%)
EoiNAS without SSS and OII	×	×	1.0	3.4	2.93
EoiNAS without OII	✓	×	0.6	3.4	2.72
EoiNAS	✓	✓	0.6	3.4	2.50

the convergence during the search stage and achieve higher accuracy in comparison to the baseline.

F. Searched Architecture Analysis

In differentiable NAS methods, architecture weights are not able to accurately reflect the importance of each operation, as discussed in Sections I and III, because the accuracy of the fully trained stand-alone model and their corresponding architecture weights have low correlation. The proposed OII can better decide which operation should be kept on each edge and which edges should be the input of each node, especially for the selection of skip-connect.

The skip-connect operation plays an important role in the cell structure. As studied in [61] and [62], including a reasonable number and location of skip connections would make the gradient flow easier and optimize the deep neural network more stable. Comparing the searched results in Figs. 5 and 9, the architecture discovered by EoiNAS on CIFAR-10 tends to preserve skip-connect operations in a hierarchical manner, which can facilitate gradient back propagation and make the network have a better convergence.

In addition, compared with Figs. 5 and 9, we can see that EoiNAS encourages connections in a cell to cascade more

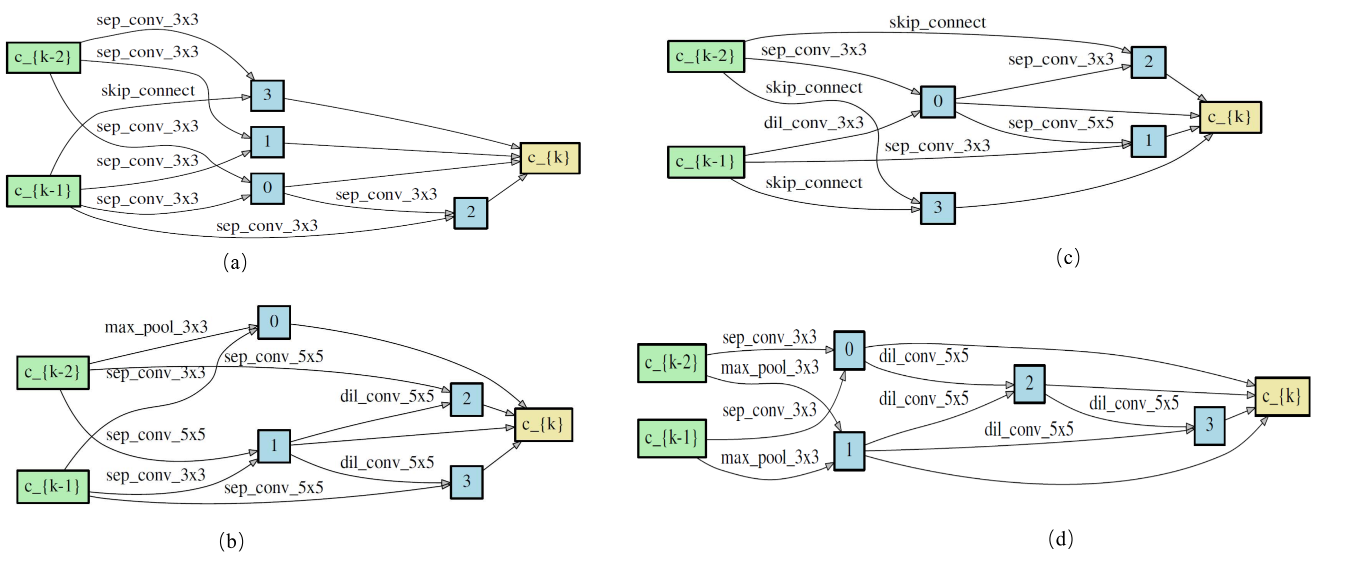


Fig. 9. Detailed structure of the best cells discovered on CIFAR-10 by baseline and baseline + SSS. (a) and (b) Best cells discovered on CIFAR-10 by baseline. (c) and (d) Best cells discovered by baseline + SSS. (a) and (c) Normal cell. (b) and (d) Reduction cell.

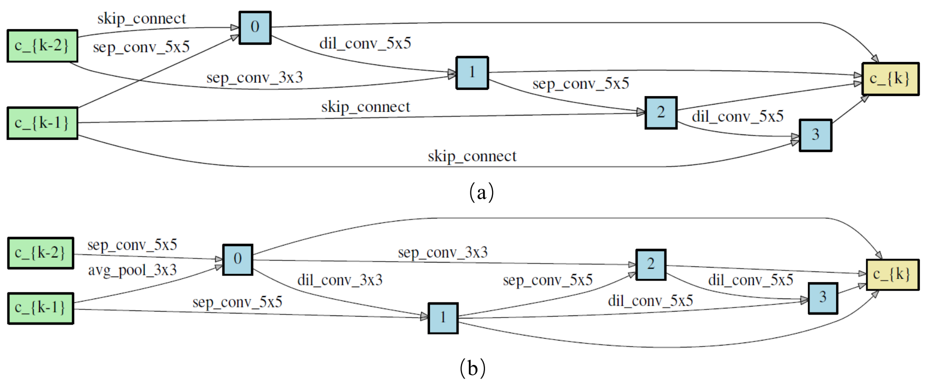


Fig. 10. Detailed structure of the best cells discovered on Cityscapes by our EoiNAS. (a) Normal cell. (b) Reduction cell. The definition of the operations on the edges is in Section III-A. In the normal cell, the stride of each operation is one. For the reduction cell, the stride of operations is set as two on input nodes and one on other nodes.

levels; in other words, there are more layers in the cell, making the evaluation network deeper and achieving better classification performance.

Finally, combining the OII with the SSS strategy can mutually enhance both of them. The indicator is able to accurately represent the importance of operation and determine the remaining and pruning operations. Meanwhile, by gradually pruning inferior operations, we can obtain a more accurate indicator.

V. Experiments on Semantic Segmentation Task

In this section, we further demonstrate the effectiveness of EoiNAS in the semantic segmentation task. Architecture search and evaluation are performed on the Cityscapes [61] dataset.

A. Datasets

Cityscapes [61] is a popular dataset for the semantic understanding of urban scenes. It contains high-quality pixel-level annotations of 5000 images collected in street scenes from 50 different cities. The image resolution was 1024 × 2048. Following the standard protocol [61] in the segmentation task, 19 semantic labels were used for evaluation. The training, validation, and testing sets contained 2975, 500, and 1525 images, respectively. An additional set of 23 473 coarse annotated images is also available in this dataset. Similar to the classification task, the training set was equally split into two subsets: one for updating network parameters and the other for updating the architecture parameters.

B. Implementation Details

Similar to the classification task, our experiments consisted of two stages. First, the EoiNAS was applied to the Cityscapes dataset to search for the normal/reduction cells constituting an architecture. Then, the architecture was retrained for evaluation.

1) Network Configurations for Searching: The neural cells for CNN were searched on Cityscapes following the method of [12]. The candidate function set O had the same eight different candidate operations as in the classification task.

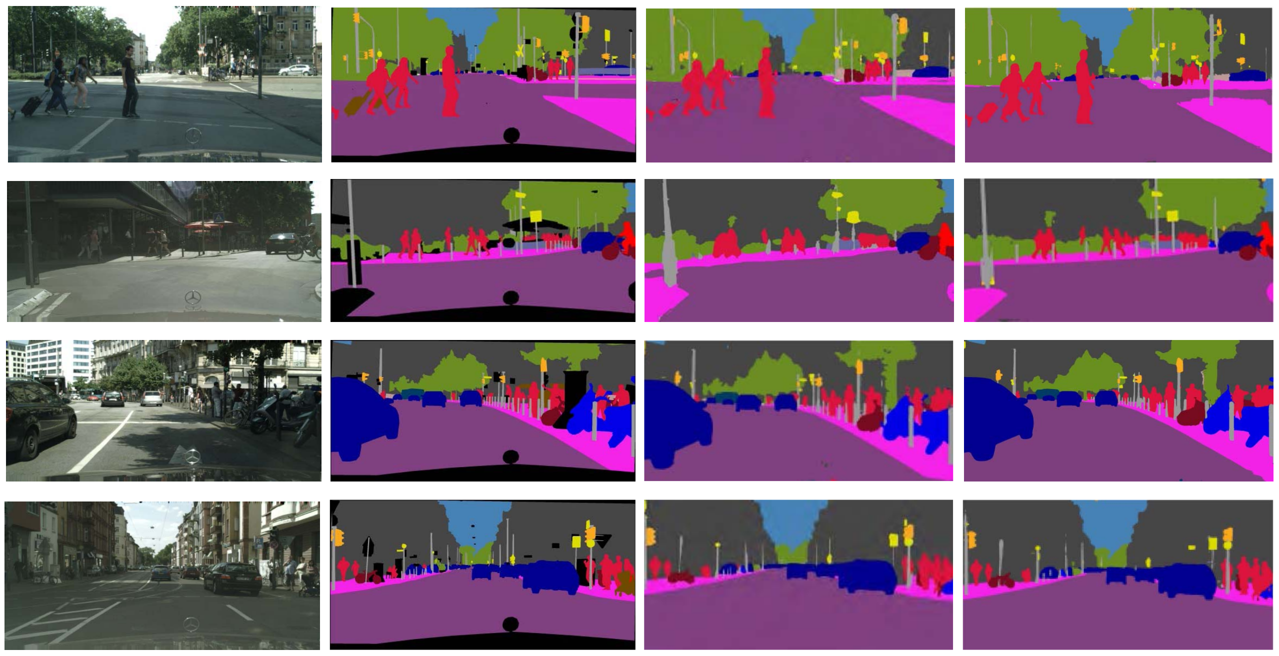


Fig. 11. Examples of EoiNAS results on the Cityscapes dataset. (a) Image. (b) Ground truth. (c) PSPNet. (d) EoiNAS.

TABLE IV
DETAILED NETWORK STRUCTURE ON THE SEARCHING STAGE, IN WHICH THE NORMAL AND REDUCTION CELL ARE SEARCHED BY OUR EoiNAS

Architecture	s	Feature map size	Output stride
Conv2d, 3×3	2	160×160	2
Conv2d, 3×3	1	160×160	2
Conv2d, 3×3	2	80×80	4
Normal cell$\times 3$	1	80×80	4
Reduction cell$\times 1$	2	40×40	8
Normal cell$\times 3$	1	40×40	8
Reduction cell$\times 1$	2	20×20	16
Normal cell$\times 2$	1	20×20	16
ASPP	–	20×20	16

The detailed network structure is shown in Table IV. Three convolution operations and ten cells formed a backbone. Cells located at 1/3 and 2/3 of the total depth of the network were reduction cells. An atrous spatial pyramid pooling (ASPP) [12] module was appended at the top of the final cell, and its output was bilinearly upsampled to the original resolution to produce the final prediction. By default, we trained the network for 160 epochs in total.

Here, we clarify why the network structure for semantic segmentation is different from that for image classification. The task of image classification focuses on the semantic aggregation of the entire image, which requires a contact feature map. Thus, the backbone network for image classification contained five spatial reductions, which resulted in a small feature map of 1/32 size of the original image. In contrast, semantic segmentation focused on the semantic aggregation of pixels. However, too much spatial reduction would lead to the loss of spatial information of pixels, thus degrading segmentation performance. To this end, the number of spatial reduction operations should be reduced for semantic segmentation. In our work, the spatial resolution of the final feature map was set only 16 times smaller than the input image resolution to balance the spatial density, semantics, and expensive computation. In addition to the first and third convolutional layers of the backbone network with strides of 2, the two reduction cells also served to downsample the feature map. Except for the reduction cells, the other cells were normal cells without reduction.

2) Parameter Settings for Searching: When learning network parameters w, we used the SGD optimizer with a momentum of 0.9, cosine learning rate that decays from 0.025 to 0.001, and weight decay of 0.0003. For architecture parameter α, we use zero initialization, which implies an equal amount of attention over all possible operations. We used the Adam optimization [56] with a learning rate of 0.0003, momentum (0.5; 0.999), and a weight decay of 0.001. To control the temperature parameter T of the Gumbel softmax in (5), we used an exponentially decaying schedule. T was initialized as 5 and, finally, reduced to 0. Our EoiNAS took approximately 0.8 GPU days to finish the search procedure on a single NVIDIA 1080Ti GPU. The best cells searched by EoiNAS are shown in Fig. 10.

C. Results on Cityscapes

To evaluate our searched architecture on the semantic segmentation task, we adopted the encoder–decoder structure. Specifically, the structure of the encoder is identical to the network structure in the searching stage, as shown in Table IV, while the decoder was the same as that in [46], which recovers the information of edges by exploiting the low-level features that have a downsample rate of 4. We first utilized the ImageNet dataset to pretrain the backbone network

TABLE V

PER-CLASS RESULTS ON THE CITYSCAPES DATASET. OUR EOINAS TRAINED WITHOUT IMAGENET PRETRAINING IS MARKED WITH "‡"

Method	road	swalk	build.	wall	fence	pole	tlight	sign	veg.	terrain	sky	person	rider	car	truck	bus	train	mbike	bike	mIoU
ENet [62]	96.3	74.2	85	32.2	33.2	43.5	34.1	44	88.6	61.4	90.6	65.5	38.4	90.6	36.9	50.5	48.1	38.8	55.4	58.3
CRF-RNN [63]	96.3	73.9	88.2	47.6	41.3	35.2	49.5	59.7	90.6	66.1	93.5	70.4	34.7	90.1	39.2	57.5	55.4	43.9	54.6	62.5
FCN [64]	97.4	78.4	89.2	34.9	44.2	47.4	60.1	65.0	91.4	69.3	93.9	77.1	51.4	92.6	35.3	48.6	46.5	51.6	66.8	65.3
SiCNN+CRF [65]	96.3	76.8	88.8	40.0	45.4	50.1	63.3	69.6	90.6	67.1	92.2	77.6	55.9	90.1	39.2	51.3	44.4	54.4	66.1	66.3
DPN [66]	97.5	78.5	89.5	40.4	45.9	51.1	56.8	65.3	91.5	69.4	94.5	77.5	54.2	92.5	44.5	53.4	49.9	52.1	64.8	66.8
Dilation10 [67]	97.6	79.2	89.9	37.3	47.6	53.2	58.6	65.2	91.8	69.4	93.7	78.9	55.0	93.3	45.5	53.4	47.7	52.2	66.0	67.1
LRR [68]	97.7	79.9	90.7	44.4	48.6	58.6	68.2	72.0	92.5	69.3	94.7	81.6	60.0	94.0	43.6	56.8	47.2	54.8	69.7	69.7
DeepLabv2 [69]	97.9	81.3	90.3	48.8	47.4	49.6	57.9	67.3	91.9	69.4	94.2	79.8	59.8	93.7	56.5	67.5	57.5	57.7	68.8	70.4
Piecewise [70]	98.0	82.6	90.6	44.0	50.7	51.1	65.0	71.7	92.0	72.0	94.1	81.5	61.1	94.3	61.1	65.1	53.8	61.6	70.6	71.6
FRRN [71]	98.2	83.3	91.6	45.8	51.1	62.2	69.4	72.4	92.6	70.0	94.9	81.6	62.7	94.6	49.1	67.1	55.3	53.5	53.5	71.8
RefineNet [72]	98.3	83.7	91.8	47.2	52.3	62.4	69.8	73.1	92.9	70.2	94.9	82.5	64.8	94.8	64.2	73	58.4	59.6	64.5	73.6
TuSimple [73]	98.4	84.8	92.4	54.3	54.3	62.8	70.2	75.9	93.2	70.9	94.7	84.1	66.5	95.3	68.3	78	63.9	64.8	73.6	76.1
PSPNet [74]	98.6	86.2	92.9	50.8	58.8	64.0	75.6	79.0	93.4	72.3	95.4	86.5	71.3	95.9	68.2	79.5	73.8	69.5	77.2	78.4
DeepLabv3+ [75]	98.4	86.4	93.0	57.6	62.5	64.3	76.0	79.6	94.0	71.5	96.2	86.5	70.0	96.2	71.2	85.6	77.2	68.4	77.0	79.5
EoiNAS‡	97.9	81.5	91.4	50.5	52.7	59.4	69.6	72.7	92.5	70.1	95.0	81.3	60.1	94.3	51.2	67.7	61.3	59.5	71.2	72.6
EoiNAS	98.6	86.6	93.2	58.1	63.0	64.5	75.2	79.2	93.4	72.1	95.1	86.3	71.4	96.0	73.5	90.4	80.3	69.9	76.9	**80.2**

(three convolution operations and ten cells). Then, the whole encoder–decoder network was initialized with the ImageNet pretrained weights and trained on the Cityscapes dataset. Our model was not trained on coarsely annotated images.

During the training, we adopted a polynomial learning rate schedule [75], [76] with an initial learning rate of 0.007. The crop size was 769×769. The model was trained by 90k iterations with a batch size of 4 due to the limitations of GPU memory.

Fig. 11 shows some examples of EoiNAS results on the Cityscapes dataset. The segmentation results are listed in Table V. Our EoiNAS achieved a higher mIOU on the Cityscapes dataset, even without ImageNet pretraining.

VI. CONCLUSION

In this article, we presented EoiNAS, a simple yet efficient architecture search algorithm for convolutional networks, in which a new indicator was proposed to fully exploit the operational importance to guide the model search. A high-order Markov chain-based search space reducing strategy was proposed to further improve the search efficiency and accuracy. To verify the effectiveness, we applied EoiNAS on two vision tasks: image classification and semantic segmentation. The EoiNAS drastically reduced the computation consumption while achieving excellent model accuracies, which outperformed the human-designed networks and other state-of-the-art NAS methods for both vision tasks.

REFERENCES

[1] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016, *arXiv:1611.01578*. [Online]. Available: http://arxiv.org/abs/1611.01578

[2] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8697–8710.

[3] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–8.

[4] T. Elsken, J. Hendrik Metzen, and F. Hutter, "Neural architecture search: A survey," 2018, *arXiv:1808.05377*. [Online]. Available: http://arxiv.org/abs/1808.05377

[5] C. Liu *et al.*, "Progressive neural architecture search," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 19–34.

[6] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," 2018, *arXiv:1806.09055*. [Online]. Available: http://arxiv.org/abs/1806.09055

[7] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 4780–4789.

[8] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, "Neural architecture optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 7816–7827.

[9] Y. Chen, K. Zhu, L. Zhu, X. He, P. Ghamisi, and J. Atli Benediktsson, "Automatic design of convolutional neural network for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 9, pp. 7048–7066, Sep. 2019.

[10] X. Chu, B. Zhang, H. Ma, R. Xu, and Q. Li, "Fast, accurate and lightweight super-resolution with neural architecture search," 2019, *arXiv:1901.07261*. [Online]. Available: http://arxiv.org/abs/1901.07261

[11] L.-C. Chen *et al.*, "Searching for efficient multi-scale architectures for dense image prediction," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 8699–8710.

[12] C. Liu *et al.*, "Auto-DeepLab: Hierarchical neural architecture search for semantic image segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 82–92.

[13] Y. Zhang, Z. Qiu, J. Liu, T. Yao, D. Liu, and T. Mei, "Customizable architecture search for semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, p. 11.

[14] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "NAS-FPN: Learning scalable feature pyramid architecture for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7036–7045.

[15] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," 2019, *arXiv:1904.12760*. [Online]. Available: http://arxiv.org/abs/1904.12760

[16] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," 2017, *arXiv:1711.00436*. [Online]. Available: http://arxiv.org/abs/1711.00436

[17] L. Xie and A. Yuille, "Genetic CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1379–1388.

[18] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," 2018, *arXiv:1802.03268*. [Online]. Available: http://arxiv.org/abs/1802.03268

[19] S. Xie, H. Zheng, C. Liu, and L. Lin, "SNAS: Stochastic neural architecture search," 2018, *arXiv:1812.09926*. [Online]. Available: http://arxiv.org/abs/1812.09926

[20] X. Dong and Y. Yang, "Searching for a robust neural architecture in four GPU hours," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1761–1770.

[21] X. Zheng, R. Ji, L. Tang, B. Zhang, J. Liu, and Q. Tian, "Multinomial distribution learning for effective neural architecture search," 2019, *arXiv:1905.07529*. [Online]. Available: http://arxiv.org/abs/1905.07529

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[23] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[26] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[27] E. J. Gumbel, *Statistical Theory of Extreme Values and Some Practical Applications: A Series of Lectures*, vol. 33. Washington, DC, USA: U.S. Government Printing Office, 1948.

[28] S.-H. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. Torr, "Res2Net: A new multi-scale backbone architecture," 2019, *arXiv:1904.01169*. [Online]. Available: http://arxiv.org/abs/1904.01169

[29] G. Huang, S. Liu, L. V. D. Maaten, and K. Q. Weinberger, "CondenseNet: An efficient DenseNet using learned group convolutions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2752–2761.

[30] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 116–131.

[31] D. Stamoulis *et al.*, "Single-path NAS: Designing hardware-efficient ConvNets in less than 4 hours," 2019, *arXiv:1904.02877*. [Online]. Available: http://arxiv.org/abs/1904.02877

[32] Z. Guo *et al.*, "Single path one-shot neural architecture search with uniform sampling," 2019, *arXiv:1904.00420*. [Online]. Available: http://arxiv.org/abs/1904.00420

[33] M. Tan *et al.*, "MnasNet: Platform-aware neural architecture search for mobile," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2820–2828.

[34] B. Wu *et al.*, "FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, p. 10.

[35] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," 2018, *arXiv:1812.00332*. [Online]. Available: http://arxiv.org/abs/1812.00332

[36] F. Paolo Casale, J. Gordon, and N. Fusi, "Probabilistic neural architecture search," 2019, *arXiv:1902.05116*. [Online]. Available: http://arxiv.org/abs/1902.05116

[37] G. Li, G. Qian, I. C. Delgadillo, M. Muller, A. Thabet, and B. Ghanem, "SGAS: Sequential greedy architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1620–1630.

[38] G. Li *et al.*, "Hierarchical neural architecture search via operator clustering," 2019, *arXiv:1909.11926*. [Online]. Available: http://arxiv.org/abs/1909.11926

[39] C. J. Maddison, D. Tarlow, and T. Minka, "A* sampling," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1–10.

[40] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," 2016, *arXiv:1611.01144*. [Online]. Available: http://arxiv.org/abs/1611.01144

[41] A. Yang, P. M. Esperanca, and F. M. Carlucci, "NAS evaluation is frustratingly hard," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–13.

[42] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.

[43] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," 2016, *arXiv:1611.02167*. [Online]. Available: http://arxiv.org/abs/1611.02167

[44] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "SMASH: One-shot model architecture search through HyperNetworks," 2017, *arXiv:1708.05344*. [Online]. Available: http://arxiv.org/abs/1708.05344

[45] Z. Yang *et al.*, "CARS: Continuous evolution for efficient neural architecture search," 2019, *arXiv:1909.04977*. [Online]. Available: http://arxiv.org/abs/1909.04977

[46] Z. Lu *et al.*, "NSGA-net: Neural architecture search using multi-objective genetic algorithm," 2018, *arXiv:1810.03522*. [Online]. Available: http://arxiv.org/abs/1810.03522

[47] H. Zhou, M. Yang, J. Wang, and W. Pan, "BayesNAS: A Bayesian approach for neural architecture search," 2019, *arXiv:1905.04919*. [Online]. Available: http://arxiv.org/abs/1905.04919

[48] C. Zhang, M. Ren, and R. Urtasun, "Graph HyperNetworks for neural architecture search," 2018, *arXiv:1810.05749*. [Online]. Available: http://arxiv.org/abs/1810.05749

[49] J. Chang *et al.*, "Data: Differentiable architecture approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 874–884.

[50] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep. 7, 2009.

[51] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[52] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: http://arxiv.org/abs/1704.04861

[53] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[54] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6848–6856.

[55] X. Dong and Y. Yang, "One-shot neural architecture search via self-evaluated template network," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3681–3690.

[56] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[57] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," 2017, *arXiv:1708.04552*. [Online]. Available: http://arxiv.org/abs/1708.04552

[58] A. Paszke *et al.*, "Automatic differentiation in Pytorch," in *Proc. Adv. Neural Inf. Process. Syst. Workshop*, Long Beach, CA, USA, Dec. 2017, pp. 1–4.

[59] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 630–645.

[60] K. A. Sankararaman, S. De, Z. Xu, W. Ronny Huang, and T. Goldstein, "The impact of neural network overparameterization on gradient confusion and stochastic gradient descent," 2019, *arXiv:1904.06963*. [Online]. Available: http://arxiv.org/abs/1904.06963

[61] M. Cordts *et al.*, "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3213–3223.

[62] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "ENet: A deep neural network architecture for real-time semantic segmentation," 2016, *arXiv:1606.02147*. [Online]. Available: http://arxiv.org/abs/1606.02147

[63] S. Zheng *et al.*, "Conditional random fields as recurrent neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1529–1537.

[64] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.

[65] I. Krešo, D. Čaušević, J. Krapac, and S. Šegvić, "Convolutional scale invariance for semantic segmentation," in *Proc. German Conf. Pattern Recognit.* Cham, Switzerland: Springer, 2016, pp. 64–75.

[66] Z. Liu, X. Li, P. Luo, C.-C. Loy, and X. Tang, "Semantic image segmentation via deep parsing network," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1377–1385.

[67] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," 2015, *arXiv:1511.07122*. [Online]. Available: http://arxiv.org/abs/1511.07122

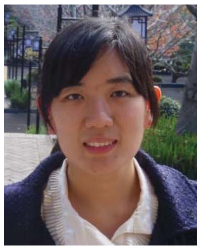
[68] G. Ghiasi and C. C. Fowlkes, "Laplacian pyramid reconstruction and refinement for semantic segmentation," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 519–534.

[69] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.

[70] G. Lin, C. Shen, A. van den Hengel, and I. Reid, "Efficient piecewise training of deep structured models for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3194–3203.

[71] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, "Full-resolution residual networks for semantic segmentation in street scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4151–4160.

[72] G. Lin, A. Milan, C. Shen, and I. Reid, "RefineNet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1925–1934.
[73] P. Wang *et al.*, "Understanding convolution for semantic segmentation," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2018, pp. 1451–1460.
[74] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2881–2890.
[75] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 801–818.
[76] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017, *arXiv:1706.05587*. [Online]. Available: http://arxiv.org/abs/1706.05587



Yuan Zhou (Senior Member, IEEE) received the B.Eng., M.Eng., and Ph.D. degrees from Tianjin University, Tianjin, China, in 2006, 2008, and 2011, respectively, all in electronic engineering and communication engineering.

Since 2011, she has been a Faculty Member with the School of Electronic Information Engineering, Tianjin University, where she is currently an Associate Professor. From 2013 to 2014, she was a Visiting Scholar with the School of Mechanical and Electrical Engineering, University of Southern Queensland, Toowoomba, QLD, Australia. From 2016 to 2017, she was a Visiting Scholar with the Department of Electrical Engineering, Princeton University, Princeton, NJ, USA. Her current research interests include computer vision and image/video communications.

Xukai Xie received the B.Eng. degree from the Hebei University of Technology, Tianjin, China, in 2018. He is currently pursuing the M.Eng. degree with the School of Electrical and Information Engineering, Tianjin University, Tianjin.

His research interests are in neural architecture search and model compression.

Sun-Yuan Kung (Life Fellow, IEEE) is currently a Professor with the Department of Electrical Engineering, Princeton University, Princeton, NJ, USA. He has authored or coauthored over 500 technical publications and numerous textbooks, including *VLSI Array Processors* (Prentice-Hall, 1988), *Digital Neural Networks* (Prentice-Hall, 1993), *Principal Component Neural Networks* (Wiley, 1996), *Biometric Authentication: A Machine Learning Approach* (Prentice-Hall, 2004), and *Kernel Methods and Machine Learning* (Cambridge University Press, 2014). His current research interests include machine learning, data mining and privacy, statistical estimation, system identification, wireless communication, VLSI array processors, signal processing, and multimedia information processing.

Prof. Kung was a founding member of several technical committees of the IEEE Signal Processing Society and the first Associate Editor in VLSI area and the first Associate Editor in Neural Network for the IEEE TRANSACTIONS ON SIGNAL PROCESSING in 1984 and 1991, respectively. He has served as a member of the Board of Governors of the IEEE Signal Processing Society from 1989 to 1991. He was a recipient of the IEEE Signal Processing Society's Technical Achievement Award for the contributions on parallel processing and neural network algorithms for signal processing in 1992, the IEEE Signal Processing Society's Best Paper Award for his publication on principal component neural networks in 1996, and the IEEE Third Millennium Medal in 2000. He was a Distinguished Lecturer of the IEEE Signal Processing Society in 1994. Since 1990, he has been the Editor-in-Chief of the *Journal of VLSI Signal Processing Systems*.