Shift-Invariant Convolutional Network Search

Nannan Li Yaran Chen Zixiang Ding Dongbin Zhao
State Key Laboratory of Management and Control for Complex Systems,
Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
School of artificial intelligence, University of Chinese Academy of Sciences, Beijing 101408, China
linannan2017@ia.ac.cn chenyaran2013@ia.ac.cn dingzixiang2018@ia.ac.cn dongbin.zhao@ia.ac.cn

***Abstract*—The development of Neural Architecture Search (NAS) makes Convolutional Neural Networks (CNN) more diverse and effective. But previous NAS approaches don't pay attention to the shift-invariant of CNN. Without the shift-invariant, convolutional network is not robust enough when input data is disturbed or damaged. Besides, taking accuracy as the only optimization goal of NAS cannot meet the increasingly diverse needs. In this paper, we propose Shift-Invariant Convolutional Network Search (SICNS). It uses one-shot NAS to search for shift-invariant convolutional network by incorporating the low-pass filter into the one-shot model. Furthermore, SICNS optimizes multiple indicators simultaneously through the multi-objective evolutionary algorithm. Through training one-shot model and evolving the architecture, we obtain convolutional networks which are robust and powerful on image classification task. Especially, our work can achieve 4.52% test error on CIFAR-10 with 0.7M parameters. And in case the input data are disturbed, the accuracy of searched network is 2.96% higher than network without low-pass filter.**

***Index Terms*—Neural architecture search, shift-invariant, multi-objective, low-pass filter, image classification**

I. Introduction

The machine learning has shown its powerful learning ability in a lot of areas [1]–[6], etc. As one of the methods of Automated Machine Learning (AutoML), neural architecture search has made a great contribution in the progress of machine learning. And benefiting from this, more and more novel and high-performance models are widely applied, especially convolution models [7]–[11]. Although both human-designed and automated-searched convolutional networks have achieved excellent results on image processing tasks, these convolutional networks don't consider about the shift-invariant of the network. To be specific, they tend to choose max pooling which performs better but is not anti-aliased. As a result, when the input image of convolutional network is disturbed (i.e. rotates or shifts), great change will be reflected on the output [12] [13]. For example, let [0, 0, 1, 1] as the input, the output after max pooling (stride as 2 and kernel size as 2) is [0, 1]. But when input shifts, the output is [1, 1]. The shift of input will have a great impact on output. The anti-aliasing is working on making convolutional network shift-invariant, and it can be achieved by adding a low-pass filter. In order to joint the superiority of max pooling and low-pass filter, Zhang inserts a low-pass filter between two operations of max pooling and makes convolution network become shift-invariant again [14]. These two operations are max operator evaluation and subsampling respectively. Concretely speaking, we use a blurry low-pass filter for subsampling. And the outputs of above example logically change to [0.5, 1] and [0.75, 0.75] due to the blurry low-pass filter. This reduces the effect of shift on input obviously. Not only that, this trick also achieves the similar effect in average pooling and strided-convolution. Therefore we put the low-pass filter included in the supernet (i.e. one-shot model) of one-shot NAS to search for networks with shift-invariant.

This work is supported by Youth research fund of the state key laboratory of complex systems management and control No. 20190213 and No. GJHZ1849 International Partnership Program of Chinese Academy of Sciences, and No FA2018111061SOW12 Program of the Huawei Technologies Co Ltd.

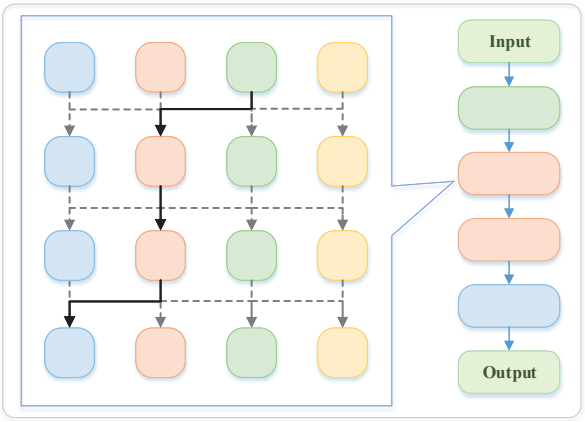

Fig. 1. The mini example of subnet and supernet. The left is a four-layer supernet with four candidate operations, and the right is a sampled single-path subnet.

One-shot neural architecture search is to speed up the performance evaluation of NAS. In one-shot NAS, the one-shot model training and sampled architecture optimization are critical. One-shot NAS treats all possible networks as subnets of supernet as shown in Fig. 1. The left of Fig. 1 shows an one-shot model with four layers, and every layer has four candidate operations. The right of Fig. 1 is the sampled network from the one-shot model. Besides, searched networks inherit weights of supernet [15], which significantly accelerates the performance evaluation process of NAS. We use single-path one-shot NAS [16], which optimizes weights of supernet and architecture sequentially instead of jointly (DARTs, [9]) or nestedly (ENAS, [7]). Thus it decouples two optimization processes, and makes architecture optimization process perform better in flexibility

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

and efficiency. And it uses uniform sampling to sample single-path network which is trained to update the supernet's weights. Combined with the above, the process of single-path one-shot NAS is similar to the idea of meta learning [17]. The prior tasks are sampled networks in weight optimization progress while the configuration is weights of supernet. And sampled networks in architecture optimization process are the new tasks. The accuracy of image (CIFAR-10) classification is the evaluation. Hence we formulate it as a meta learning problem to optimize the weights of supernet. And the one-shot model training can be more fair through training each node fairly.

As the application requirements of end devices for convolutional networks increasing, the networks need to meet multiple performance indicators at the same time instead of a single, such as accuracy, parameters, FLOPs, latency and so on. So in order to meet increasingly complex requirements of the model, we treat architecture optimization as a multi-objective optimization problem. And we use multi-objective evolutionary algorithm to evolve sampled networks. We take Non dominated Sorting Genetic Algorithm-II (NSGA-II) [18] as the search strategy to address this problem. Optimizing multiple objectives simultaneously, we can obtain high-performance network with few parameters. As shown in Fig. 2, incorporating the low-pass filter into one-shot model, and formulating single-path one-shot NAS as the meta learning problem, SICNS can search for shift-invariant convolution model through NSGA-II. And we discuss the effect of low-pass filter in convolutional network. Our approach can achieve 4.52% test error on CIFAR-10 with 0.7M parameters. And the accuracy of searched network 2.96% higher than accuracy network without low-pass filter when input data are shifted. In summary, our contributions are as follows:

1) We incorporate the low-pass filter into supernet and make the searched convolution networks shift-invariant. Consequently when the input image is disturbed, the convolutional network performs stably.
2) We formulate single-path one-shot neural architecture search as the meta learning problem and make the weight optimization more fair.
3) The convolutional network we searched achieves competitive results on CIFAR-10. And compared with the networks without the low-pass filter, the searched network has better performance.

II. Related work

In this section, we introduce the shift-invariant and one-shot neural architecture search respectively. And we review their related research work.

Shift-Invariant: Shift-invariant of convolution means that when input image of convolution shifts, output result (label) should be invariant. But the subsampling operation (pooling operation, the strided-convolution, etc) cause convolutional network lose shift-invariant [19]. Removing subsampling operation and using dilated convolution [20] [21] can improve the shift-invariant, but this is very computationally intensive. In addition, improving the subsampling operation can also address this problem to some extent. Due to the average pooling perform better than max pooling in shift-invariant but worse in effectiveness [22], Lee et al. [23] propose the pooling operation which combines the above-mentioned two pooling operations, but this cannot be completely anti-aliased. Then Adobe obtains shift-invariant convolutional network through blurring between max and subsampling of max pooling [14], this work add the low-pass filter before subsampling.

One-Shot Neural Architecture Search: As one of the methods to accelerate performance evaluation, one-shot NAS builds a supernet and shares weights among the subnets that sampled from the supernet. Such that training supernet only once, we can directly inherit the weights during the architecture optimization process. There are some one-shot NAS works with superior performance in recent years [16], [24]–[26]. SMASH learns a auxiliary hypernetwork which can output proper weights for model [24], while Graph Hyper-Networks (GHN) get weights by using graph neural network [25]. Single-path one-shot NAS trains single-path supernet to obtain suitable weights [16]. Self-Evaluated Template Network (SETN) analyzes the weight sharing of one-shot NAS and proves that one shot NAS can work well even only with gradient descent [26]. The above approaches can directly evaluate the performance of networks with no training [16], [24], [25].

Further, there are some multi-objective NAS works which have achieved excellent results, such as NSGA-Net [27], LEMONADE [28], etc. And as a kind of multi-objective genetic algorithm with the most influence and the widest application range, NSGA-II is used as search strategy of our work.

III. Method

Our work, shift-invariant convolutional network search, can be treated as a meta learning framework which is a two-stage training process as shown in Fig. 3. We train weights θ of one-shot model using sampled networks in the first stage (meta training stage), and the weights can be adapted in the new sampled networks in second stage (network optimization). Training is performed on CIFAR-10 during overall process.

In our framework, the meta training is to optimize the weights θ which is used to be adapted in network optimization process. Concretely speaking, we sample network a from supernet, and the weights is updated through Stochastic Gradient Descent (SGD). Thus the sampled network is equivalent to meta-task, and we train every meta-task only one step to update weights. The benefit of this is that we can sample more networks in a certain period of time. Furthermore, to some extent, this guarantees the fairness of the weights on a new task. With the meta-trained one-shot model, we can adapt the trained weights θ_* in new sampled network a_{new}, then we use SGD update θ_* into the weights $\theta_*^{'}$ of network a_{new}.

Therefore, this section is divided into three parts: i) one-shot model, the meta training to get θ_*, ii) network evolution, adapting θ_* into a_{new}, and iii) network training, updating θ_* into $\theta_*^{'}$ of network a_{new}.

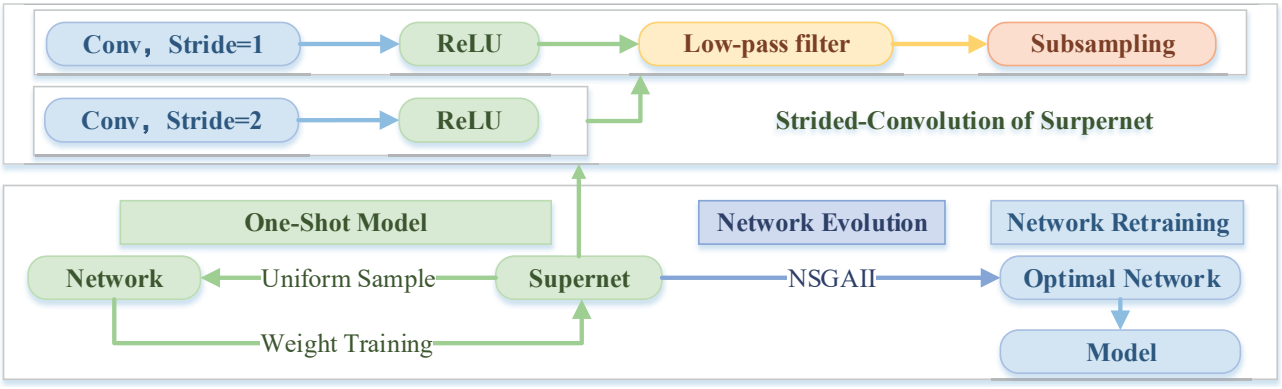

Fig. 2. The overview of our framework. Firstly, we use uniform sampling to sample network from supernet, and we train sampled network to update the weights of supernet. We incorporate low-pass filter into supernet through adding low-pass filter before subsampling in strided-convolution. Then weights are adapted into network which is optimized by NSGA-II, so we can search for optimal network efficiently. Finally we retrain the optimal network and obtain final model.

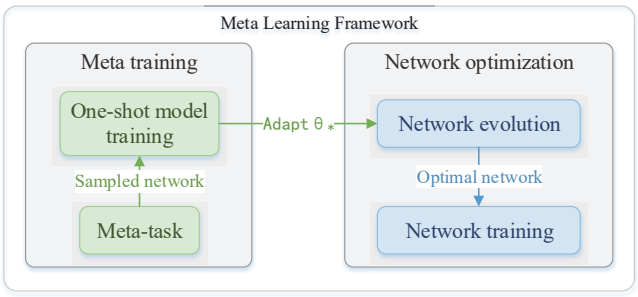

Fig. 3. The meta learning framework.

TABLE I
OPERATION SPACE OF ONE-SHOT MODEL. THE LPFILTER IS THE LOW-PASS FILTER, AND WITH REPRESENTS NETWORK THAT ADDS LOW-PASS FILTER BEFORE SUBSAMPLING WHILE WITHOUT MEANS NOT ADDING.

ID	Kernel size	Expand ratio	LPfilter
0	3	3	Without
1	3	3	With
2	5	3	Without
3	5	3	With
4	7	3	Without
5	7	3	With
6	3	6	Without
7	3	6	With
8	5	6	Without
9	5	6	With
10	7	6	Without
11	7	6	With
12	1	-	-

A. One-Shot Model

One-shot model (i.e. supernet) is critical in our approach. The building of one-shot model decides the paradigm of searched network. And the training of one-shot model can affect the evolution of network. Thus below we introduce the building and training of one-shot model respectively.

1) One-Shot Model Building: We choose block of MobileNetV3 [29], which combines architecture search and network design, as the basic block of one-shot model due to its outstanding performance. To be specific, the block contains squeeze-and-excite [30], inverted residual and linear bottleneck [31]. The kernel size and expand ratio of inverted residual are candidate variables of block. To search the shift-invariant convolutional network, whether to add a low-pass filter is also a candidate variable when subsampling. Besides, for the purpose of making the search space more flexible, we add a conv2D as candidate operation to search for networks with different sizes. To sum up, the complete operation space of one-shot model is as TABLE I, where the LPfilter is the low-pass filter. It has 13 candidate operations, which contains 12 basic blocks with three candidate variables (kernel size as 3, 5 and 7, expand ratio as 3, 6 and LPfilter as With, Without) and 1 conv2D.

Triangle filter is used as the low-pass filter in our supernet. Triangle filter, as the name implies, its filter function shape is a triangle. The weight at the center is the maximal, the other weights gradually decrease linearly within the radius of the filter kernel. Such that the triangle filter essentially implements linear interpolation. This makes triangle filter suitable for getting blurry features when subsampling. [14] compares three different filter kernels (2, 3 and 5) and shows that the large kernel size means better shift-invariant while small kernel size improve accuracy more. So we set the kernel size of triangle filter as 3 for improve both of accuracy and consistency. The specific values of the convolution kernel are as follows:

$$f = [1, 2, 1],$$

$$filt = f^T \times f = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix},$$

$$filt = filt/sum(filt).$$

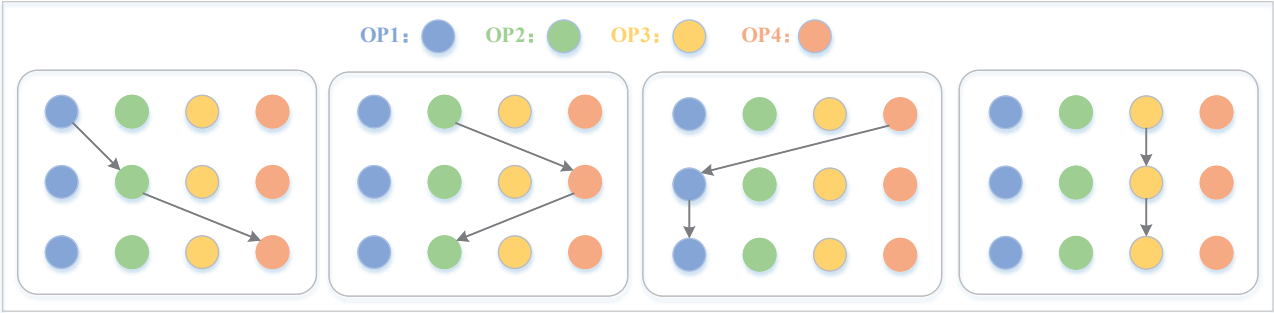

Fig. 4. The example of sampling without replacement. The one-shot model has four candidate operations in every layer. And four networks are sampled at a time, and each network has different operation in same layer.

Where f is the initial template vector for filter weights, *filt* is the weights matrix, and *sum* is the sum of each element of the matrix. We get the weight matrix *filt* throng template vector, and then we normalize *filt*. Different types of filter kernels have different vector templates. We will discuss the search for filter types in the subsequent research work.

2) One-Shot Model Training: In the one-shot model training process, we sample i networks by uniform sampling without replacement firstly. Fig. 4 is an example with i as 4, in which i is the number of candidate operations. In our work, i is 13. Uniform sampling is to make the sampled network not biased. Through sampling without replacement and sampling i (number of candidate operations) networks, we can ensure that each node of one-shot model is sampled and trained. Then to obtain suitable and fair weights, we train every sampled network only one step as mentioned above. Further, after computing the gradients of all sampled networks, we sum their losses and update the weights θ, instead of updating θ each time computing gradient. This is expressed as below:

$$\theta = \underset{\theta}{\operatorname{argmin}} \sum_{k=1}^{k=i} [L_{train}(N(a_k, \theta))],$$

where i is the number of candidate operations (here is 13), a_k is the kth network, $N(a_k, \theta)$ is the model which is network a_k with weights θ, and $L_{train}(.)$ is the training loss function on CIFAR-10.

The goal of this is to guarantee that every node's weights of one-shot model are updated the same times. Finally, when the maximum number of iterations is satisfied, we can get the trained weights θ_* of one-shot model. The specific one-shot model training algorithm is shown as Alg. 1.

Algorithm 1 One-Shot Model Training Algorithm

Input: CIFAR-10 dataset,
Sampled network a_k, k is from 1 to 13
Output: Weights θ_*
Initialize the weights θ
for each training step t **do**
  Train all nodes of one-shot model
  for each training epoch **do**
    Sample a set of networks a_0, ..., a_i without replacement
    Train a_0, ..., a_i on a batch training data
    $\theta \leftarrow \underset{\theta}{\operatorname{argmin}} \sum_{k=1}^{k=13} [L_{train}(N(a_k, \theta))]$
  end for
  $\theta_* \leftarrow \theta$
  Store weights θ_* for one-shot model
end for

B. Network Evolution

Given θ_*, we can adapt θ_* into a new sampled network in network evolution process. Thus we can search task-oriented network efficiently. In order to search for high-performance network and apply it to resource-constrained devices, the searched network need to meet multiple constraints. We use multi-objective evolutionary algorithm NSGA-II to optimize classification accuracy, amount of parameters, FLOPs and inference time simultaneously. The specific steps are as follows:

Step 1: Non-dominated sort stratifies the networks according to the network's non-inferior solution (accuracy, parameter, etc) level, and to guide the search towards the Pareto optimal solution set. If aforementioned objective function values of network A are better than the other network B, the network A is assigned to the first non-dominated layer and the network B is assigned to the next layer. Networks within the same layer have the same non-dominated order a_{rank}.

Step 2: Individual crowding distance d_i calculate. This step is to sort networks with the same a_{rank}. The individual crowding distance of the network a_i is calculated from the objective function values of its neighboring networks a_{i-1} and a_{i+1}. After determining the individual crowding distance d_i of each network a_i, we preferentially choose the network with the larger individual crowding distance.

Step 3: Elitist strategy selection keeps the excellent networks in the parent directly into the child to prevent the obtained Pareto optimal solution from being lost. In other words, we keep the network with higher non-dominated order a_{rank}. If the non-dominated order

is equivalent, we keep the network with individual larger crowding distance d_i. We stop selecting network to be retained until the number of reserved networks reaches the set number.

Step 4: Repeating Step 1 to 3 until the max epoch is met.

In this process we can get the Pareto optimal solution set, i.e. a set of networks. To obtain the optimal network, we retrain them in next process and get weights $\theta_*^{'}$ of networks in the Pareto optimal solution set.

C. Network Retraining

With a set of Pareto optimal networks, we can train each network to obtain its weights $\theta_*^{'}$ and final performance. The weights θ_* it inherited from one-shot model is updated to $\theta_*^{'}$ through using SGD. Then we rank them based on their performance and select optimal network for task requirements. Due to Conv2D operation in our candidate operations, we can flexibly search for the networks with different size. Thus we can select network with different scales to meet the task requirements. Specific experimental results are shown in the next section.

As is well-known, anti-aliasing can be achieved by low-pass filtering the image. And [14] proves that adding low-pass filter before subsampling can make convolutional network shift-invariant again. Our work aims to find out whether the network is more likely to opt for low-pass filter before subsampling through NAS. Such we do a set of comparative experiments to discuss the effect of low-pass filter on network performance and robustness to input disturbances. And we analyze the results of comparative experiments in next section.

IV. Experiment

A. Details

Our experiments are performed on CIFAR-10. The data are split into two parts: 50000 training images and 10000 validation images. We preprocessed the data through randomly flipping, centrally padding and randomly cropping.

In the training of one-shot model and retraining of the optimal network after network evolution, we adapt the same settings (e.g. learning rate schedule, etc). The batch size is 128. The epoch of one-shot model training is 120, while the optimal network retaining is 600.

In the process of the network evolution, the configuration settings of NSGA-II are as follows. The population size is set to 10, i.e. there are 10 networks per generation. The number of generations is set to 50. The objectives is set to 4 (test error, parameter, FLOPs, and inference time).

Furthermore, we do a set of comparative experiments: a) searched network, b) searched network without low-pass filter. We compare different inputs (inputs with shift and without shift) for each experiment. Besides, in previous convolutional network training, the data augmentation is applied to deal with perturbation in input data, e.g. data shifting. Such that we train the network with data augmentation and without data augmentation respectively.

TABLE II
PERFORMANCE COMPARISON BETWEEN OUR WORK AND SOME EXISTING WORKS ON CIFAR-10.

Model	Error(%)	Params(M)	FLOPs(M)
ResNet [32]	6.97	0.9	-
CondenseNet-86 [33]	5.00	0.5	65
DPP-Net [34]	4.62	0.5	64
LEMONADE [28]	4.57	0.5	-
SICNS(Ours)	**4.52**	0.7	92

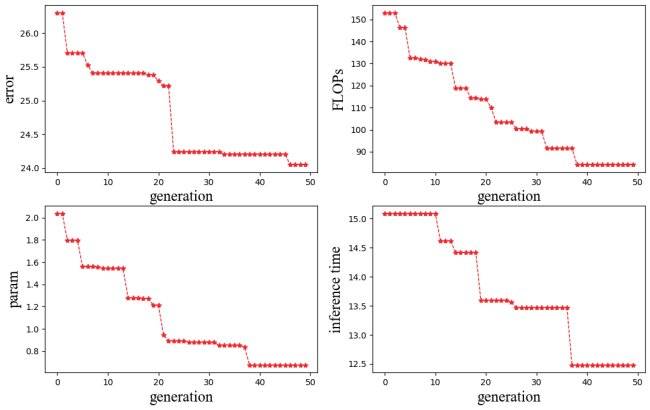

Fig. 5. Training curves for architecture evolution. X-axis is the training generations. Y-axis is the performance of four objective, the upper left is test error, the upper right is Flops, the lower left is parameters, and the lower right is inference time.

B. Results

Fig. 5 shows the training curves of architecture evolution. It can be seen that the network evolution process is gradually converging. Four objectives (test error, parameter, FLOPs, and inference time) are all getting smaller until convergence. Thus it can be seen that we can search for optimal network with few parameters and FLOPs, and the network is efficient for its shorter inference time. And results of optimal network retraining are divided into two parts, performance comparison and shift-invariant analysis.

1) performance comparison: TABLE II shows the performance comparison between our work and some existing works which include both human-designed models (the first block) and multi-objective NAS models (the second block) on CIFAR-10. As shown in TABLE II. Our approach can obtain competitive model. And to the best of my knowledge, SICNS can achieve state of the art performance in models with parameters less than 1M. The optimal model is shown in Fig.6, which performs 4.52% test error on CIFAR-10. And the model has 0.7M parameters, 92M FLOPs, and 15.35ms inference time on a single NVIDIA 2080Ti.

2) shift-invariant analysis: TABLE III is the comparison of experimental results about effect of low-pass filter on CIFAR-10 classification task. We compared accuracy for intuitive contrast. Our experiment shows that the searched networks are more likely to choose to add low-pass filter before subsampling. As shown in TABLE III, we can know that performance

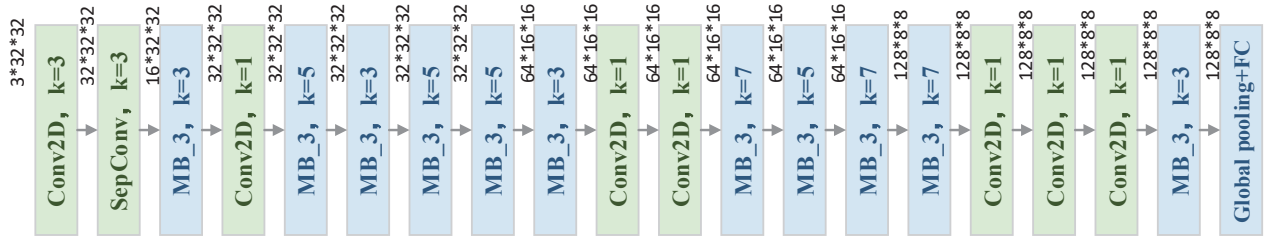

Fig. 6. The optimal model achieved by SICNS. k is the kernel size, and MB_3 is MobileNetV3 block with expand ratio as 3.

TABLE III
COMPARE THE IMPACT OF LOW-PASS FILTERS ON CLASSIFICATION ACCURACY ON CIFAR10.

Train	**With data augmentation**		**Without data augmentation**	
Valid	**Without shift**	**With shift**	**Without shift**	**With shift**
Searched network	95.48%	95.34%	91.14%	89.24%
Searched network without low-pass filter	95.31%	95.01%	89.50%	86.28%

has not much difference between valid data with shift and without shift when training data with data augmentation. And the low-pass filter can improve performance slightly. When training data without data augmentation, the performance difference is obvious. The classification accuracy of network with low-pass filter is 1.64% higher than that without low-pass filter when valid data are not shifted. But if the valid data are shifted, the classification accuracy of the network with low-pass filter has an advantage of 2.96%. Such that when input data are disturbed, the robustness of network is improved by adding low-pass filter before subsampling.

V. CONCLUSION

The paper proposes the shift-invariant convolutional network search, which uses sing-path one-shot neural architecture search. And by incorporating low-pass filter into one-shot model, we can search for high-performance network which is shift-invariant. Further, we experimentally compare the effectiveness of the low-pass filter on convolutional network. We prove that SICNS can get the competitive convolutional network that is robust when input is disturbed. Concretely speaking, SICNS can obtain 4.52% test error on CIFAR-10 with 0.7M parameters. And when input data shift, there is an advantage of 2.96% on accuracy than network without low-pass filter. In the future, we will discuss the types of low-pass filters through NAS and consider various disturbances to the input image.

REFERENCES

[1] Y. Chen, D. Zhao, L. Lv, and Q. Zhang, "Multi-task learning for dangerous object detection in autonomous driving," *Information Sciences*, vol. 432, pp. 559–571, 2018.

[2] L. Lv, D. Zhao, and K. Shao, "Deep sparse representation-based mid-level visual elements discovery in fine-grained classification," *Soft Computing*, vol. 23, no. 18, pp. 8711–8722, 2019.

[3] D. Zhao, Y. Chen, and L. Lv, "Deep reinforcement learning with visual attention for vehicle classification," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 9, no. 4, pp. 356–367, 2016.

[4] K. Shao, D. Zhao, N. Li, and Y. Zhu, "Learning battles in vizdoom via deep reinforcement learning," in *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2018, pp. 1–4.

[5] K. Shao, Y. Zhu, and D. Zhao, "Starcraft micromanagement with reinforcement learning and curriculum transfer learning," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 3, no. 1, pp. 73–84, 2018.

[6] D. Li, D. Zhao, Y. Chen, and Q. Zhang, "Deepsign: Deep learning based traffic sign recognition," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–6.

[7] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," *arXiv preprint arXiv:1802.03268*, 2018.

[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, 2019, pp. 4780–4789.

[9] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[10] Z. Ding, Y. Chen, N. Li, and D. Zhao, "Simplified space based neural architecture search," in *The 2019 Chinese Automation Congress*, accepted.

[11] N. Li, Y. Chen, Z. Ding, and D. Zhao, "Multi-objective neural architecture search for light-weight model," in *The 2019 IEEE Symposium Series on Computational Intelligence*, accepted.

[12] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, "A rotation and a translation suffice: Fooling cnns with simple transformations," *arXiv preprint arXiv:1712.02779*, 2017.

[13] A. Azulay and Y. Weiss, "Why do deep convolutional networks generalize so poorly to small image transformations?" *arXiv preprint arXiv:1805.12177*, 2018.

[14] R. Zhang, "Making convolutional networks shift-invariant again," *arXiv preprint arXiv:1904.11486*, 2019.

[15] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *arXiv preprint arXiv:1808.05377*, 2018.

[16] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single path one-shot neural architecture search with uniform sampling," *arXiv preprint arXiv:1904.00420*, 2019.

[17] J. Vanschoren, "Meta-learning: A survey," *arXiv preprint arXiv:1810.03548*, 2018.

[18] 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.

[19] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger, "Shiftable multiscale transforms," *IEEE transactions on Information Theory*, vol. 38, no. 2, pp. 587–607, 1992.

[20] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[21] F. Yu, V. Koltun, and T. Funkhouser, "Dilated residual networks," in

Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 472–480.

[22] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *International conference on artificial neural networks*. Springer, 2010, pp. 92–101.

[23] C.-Y. Lee, P. W. Gallagher, and Z. Tu, "Generalizing pooling functions in convolutional neural networks: Mixed," *Gated, and Tree*, vol. 1509, 2015.

[24] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Smash: one-shot model architecture search through hypernetworks," *arXiv preprint arXiv:1708.05344*, 2017.

[25] C. Zhang, M. Ren, and R. Urtasun, "Graph hypernetworks for neural architecture search," *arXiv preprint arXiv:1810.05749*, 2018.

[26] G. Bender, "Understanding and simplifying one-shot architecture search," 2019.

[27] 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*. ACM, 2019, pp. 419–427.

[28] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," *arXiv preprint arXiv:1804.09081*, 2018.

[29] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," *arXiv preprint arXiv:1905.02244*, 2019.

[30] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[31] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition. corr abs/1512.03385 (2015)," 2015.

[33] G. Huang, S. Liu, L. Van der Maaten, and K. Q. Weinberger, "Condensenet: An efficient densenet using learned group convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2752–2761.

[34] J.-D. Dong, A.-C. Cheng, D.-C. Juan, W. Wei, and M. Sun, "Dpp-net: Device-aware progressive search for pareto-optimal neural architectures," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 517–531.