Neural Architecture Search for Real-Time Driver Behavior Recognition

Jaeho Seong
Department of Future Automotive and IT Convergence
Kyungpook National University
Daegu, Republic of Korea
wogh3569@knu.ac.kr

Chaehyun Lee
School of Electronic and Electrical Engineering
Kyungpook National University
Daegu, Republic of Korea
hyeu333@knu.ac.kr

Dong Seog Han
School of Electronic and Electrical Engineering
Kyungpook National University
Daegu, Republic of Korea
dshan@knu.ac.kr

***Abstract*—Driver behavior recognition (DBR) helps to ensure driver safety by alerting drivers about potential hazards and minimizing them. In this paper, we use deep learning-based neural architecture search (NAS) to classify driver behavior. In the NAS method, a reinforcement learning algorithm is used, and the neural network architecture is quickly searched by sharing the weights of the parameters. Most DBR models focus on accuracy, while high processing speed is required in order to be applied to actual vehicles. In addition, since the driver monitoring system (DMS) includes complex algorithms based on deep learning, it requires a DBR model that takes this into account. We collect our own data set for driver behavior classification and recognize four common driving behaviors: general driving, mobile phone use, food intake, and smoking. The proposed model on our own data set collected through experiments has better performance and lower network cost than the previous lightweight classification model.**

Keywords—deep learning, neural network architecture search, driver behavior recognition

I. INTRODUCTION

The driver monitoring system (DMS) recognizes the driver's careless behavior and warns the driver to prevent traffic accidents. DMS includes algorithms such as driver's face detection, head pose estimation, gaze estimation, and risky behavior classification. Among them, driver behavior recognition (DBR) classifies behaviors such as smoking and eating that distract the driver while driving. Research on deep learning-based DBR using camera sensors is being actively conducted recently. However, deep learning-based models have a large amount of computation and embedded devices used in vehicle environments have limited resources. Therefore, in order to apply DBR in conjunction with algorithms used in DMS, not only high accuracy but also a lightweight classification model is required.

Recently, lightweight model architectures such as MobileNet [1], [2] and ShuffleNet [3], [4] have generally been adopted in devices with limited resources such as mobile or embedded model structures. Furthermore, remarkable progress has been made in the field of neural architecture search (NAS), which automatically generates and optimizes convolutional neural network (CNN). The classification model using NAS showed better performance than the existing human-designed networks [5]. NAS algorithms achieve high performance despite the high computational cost for searching for a network. Most of the existing NAS studies have analyzed model performance using datasets such as CIFAR-10 and ImageNet.

In this paper, we propose a NAS algorithm for DBR. The proposed NAS algorithm uses operations to optimize the neural network architecture through reinforcement learning and to consider weight reduction and accuracy in the architecture search space. The network architecture was searched for classifying driver behavior, and the proposed model not only achieved higher accuracy than the existing deep learning-based lightweight model but also significantly reduced the cost in terms of network size.

Following the introduction, this paper is structured as follows. Section 2 introduces the NAS algorithm based on reinforcement learning, and Section 3 explains the algorithm for real-time DBR. Section 4 explains the experimental results of the existing classification model and the proposed classification model, and Section 5 concludes.

II. RELATED WORK

The search space of the neural network architecture defines an architecture space to ensure the model's performance to be generated. The operator space of NAS includes convolution, pooling, and residual connections. In general, NAS requires very high computing costs. Reinforcement learning-based algorithms have proposed methods such as weight sharing and progressive search for efficient search. Most NAS do not consider model lightweight or use latency as a constraint for network lightweight, thus limiting the models that can be generated within the search space. In this paper, performance is improved by redefining the operator space of the NAS for accuracy improvement and weight reduction.

This paper follows ENAS [6] of the reinforcement learning algorithm-based NAS method. The neural network search space consists of a directed acyclic graph. Recurrent neural network (RNN) shares stored weight parameters before training the generated network to accelerate search time and applying previously trained weights to each network. Then, after training the network, the trained weights are stored. When the validation accuracy of the generated network is higher than the previously-stored validation accuracy, the stored cells are removed, and the

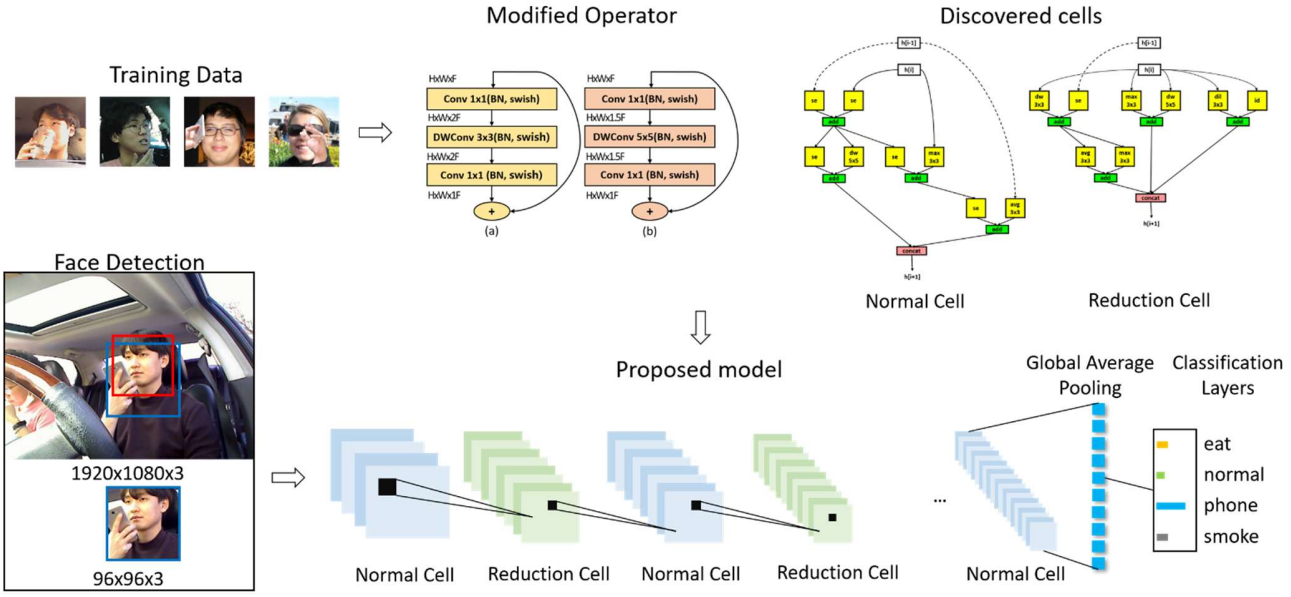

Fig. 1. Proposed DBR Algorithm.

searched cells are stored. Validation accuracy is used as a reward for the reinforcement learning algorithm to be trained in the direction that the RNN controller searches for the optimal network. This process is repeated to search for the optimal neural network architecture.

The learnable parameters are the parameter θ of the RNN and parameter w of the generated network. The policy of reinforcement learning is defined as $\pi(m;\ \theta)$. m is the model chosen from a policy. To optimize the generated network, we use stochastic gradient descent (SGD) to learn in the direction of minimization. The loss function $\mathcal{L}(m;\ w)$ uses cross-entropy, and the expected value of the loss function is $\mathbb{E}_{m\sim\pi}(m;\ \theta)\ [\mathcal{L}(m;\ w)]$. It is calculated using Monte Carlo estimation.

$$\nabla\mathbb{E}_{m\sim\pi}(m;\ \theta)\ [\mathcal{L}(m;\ w)]\ \approx \frac{1}{M}\sum_{i=1}^{M}\nabla_w\mathcal{L}(m_i, w) \qquad (1)$$

As the RNN trains, it learns in the direction of maximizing the expected value of $\mathbb{E}_{m\sim\pi}(m;\ \theta)\ [r(m,w)]$ the reward. The reward $r(m,w)$ uses the accuracy of the validation data set.

III. Proposed Algorithm

A. Data Pre-processing

This study proposes an efficient DBR algorithm for recognizing driver behavior from in-vehicle camera sensors and applying it to DMS based on deep learning. DMS consists of complex algorithms such as face detection, facial landmark, head pose estimation, gaze estimation, emotion classification, and behavior recognition. In particular, in the case of face detection, face landmarks, emotion classification, and behavior recognition, deep learning-based models with high accuracy are mainly used. For this reason, to apply a real-time DMS by linking a deep learning-based model with a large amount of computation and a DBR model, it is essential to reduce the model's weight. Face detection in DMS is an essential algorithm used before applying facial landmarks and emotion classification. Therefore, when the region of interest is extracted using the face detector, an object classification model may be used instead of object detection having a relatively large computational amount. As shown in Fig. 1, the original image is the size of 1920×1080×3. By removing the surrounding background, inference speed and classification accuracy can be improved. A region of interest, including the driver's face region, is designated through the face detector. In order to classify the driver's behavior, the region of interest is expanded. Since most of the driver's behavior range is from the face to the upper body, it expands to the lower region. The image is resized to 96×96×3 to reduce the amount of computation. Pre-processed images classify driver behavior into four classes: eat, normal, phone, smoke through the proposed CNN-based classifier.

B. Architecture Search Space

As shown in Fig. 2, the search space consists of a network, cells, and nodes. A node is a unit constituting a cell. Node is a specific tensor having an output value of operator. An input node is defined as N_i, an intermediate node is defined as N_j, and an output node Y_o. N_i is represented by N_1 and N_2. Input nodes N_1 and N_2 take as input the previous output tensor and the former output tensor, respectively, and are applied to the next first normal cell. N_j has $n-2$ output tensors from N_3 to N_n if the number of nodes is n. N_j outputs the operator defined to extract features from the RNN controller and the order for connecting nodes to each other. For all nodes N_n, if the connection between nodes is defined as (a, b) and the operation is defined as o, $o(a, b)$ can be defined as the connection between nodes N_a and N_b. Finally, N_j ($N_3 \sim N_n$) outputs Y_o by performing a concatenation operation. The RNN controller samples two operations for each node and a number to connect between the two nodes. Two output tensors are generated by the combination of each operation and node, and one output tensor is generated through element-wise addition operation.

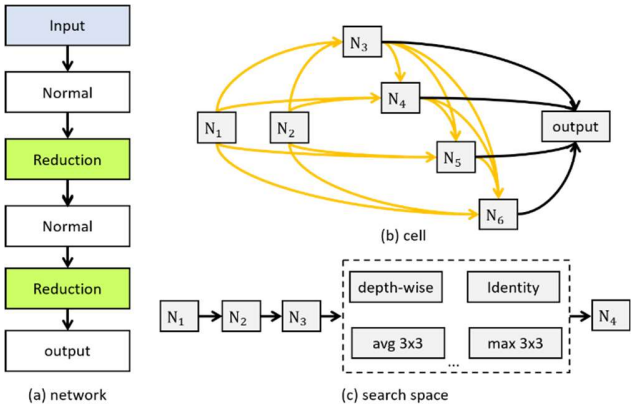

Fig. 2. Architecture Search Space. (a): Network. (b): Cell. (c): Search Space.

A cell is defined as a mapping of $H \times W \times F$ to $H' \times W' \times F'$. H is the height, W is the width, and F is the number of filters. For normal cell, $stride = 1$ is applied when the operation is applied. Therefore, it becomes $H' = H$, $W' = W$, $F' = F$. On the other hand, $stride = 2$ is applied to the reduction cell to $H' = H/2$, $W' = W/2$, and $F' = 2 \times F$ increases the number of filters.

The network consists of small modular normal cells and reduction cells. This cell-based architecture search is advantageous for weight sharing because the search cost is lower than designing the entire network, and the same architecture is repeated. The entire network is constructed by repeatedly stacking normal cells and reduced cells.

C. *Operator Space*

In general, the operator space in NAS includes convolution, pooling, and residual connection. For example, operators used in ENAS define a total of 7 operators using kernel-sized average pooling and max pooling of 3×3, kernel-sized convolution of 3×3 and 5×5, kernel-sized depthwise-separable convolution of 3×3 and 5×5 [6]. However, according to our experimental results of this paper, these operator definitions are not effective for both accuracy and weight reduction. In addition, regular convolution operations are not suitable for weight reduction models due to their high computational cost and do not include additional attention blocks [7]-[9].

Therefore, this paper proposes two operator spaces for weight reduction and performance improvement.

The first operator space is as follows:

- inverted bottleneck conv: 3×3, 5×5
- max-pooling: 3×3
- average-pooling: 3×3
- se-block expansion ratio =0.25
- skip connection (identity)
- dilated conv: 3×3

As shown in Fig. 3, the inverted bottleneck conv has kernel sizes of 3×3 and 5×5, and the expansion ratios use 2 and 1.5, respectively [2]. The se-block [7] channel reduction ratio was set to 0.25. The kernel size of the dilated convolution is 3×3 and is then replaced by regular convolution.

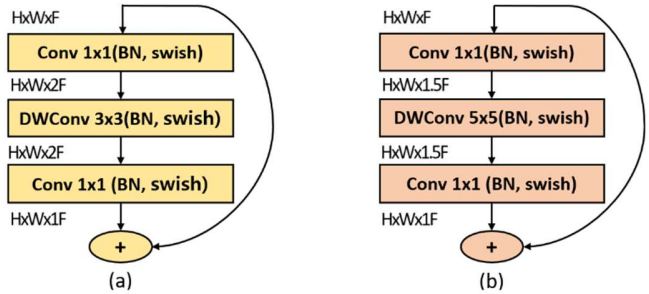

Fig. 3. First operator space for (a) 3×3 inverted bottleneck conv, (b) 5×5 inverted bottleneck conv.

The second operator space is as follows:

- inverted bottleneck conv: 3×3, 5×5
- depth-wise separable conv with se-block: 3×3, 5×5
- skip connection(identity)

As shown in Fig. 4(a) and Fig. 4(b), the architectures of the inverted bottleneck conv proposed by MobileNetV2. Depending on the kernel size, expansion ratio 4 was applied in 3×3 and 2 in 5×5 in consideration of the computation amount. Fig. 4(c) and Fig. 4(d) consist of a combination of depth-wise separable conv, se-block, and skip-connection. The reduction ratio of se-block is set to 0.25. The kernel sizes of conv were 3×3 and 5×5. The two-operator spaces in common include batch-normalization and activation functions after the convolution

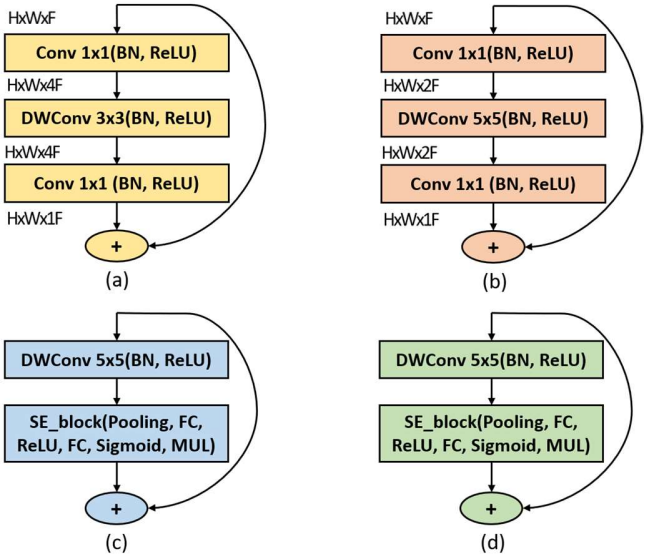

Fig. 4. Second operator space for (a) 3×3 inverted bottleneck conv, (b) 5×5 inverted bottleneck conv, (c) 3×3 depth-wise separable conv with se-block and (d) 5×5 depth-wise separable conv with se-block.

operation. Also, He normal initialization and l2 normalization were used for each kernel.

IV. Experimental Environment and Results

This Section explains the experimental environment and results. In order to collect driver behavior data, original RGB images were collected using a camera sensor. Then, the face area is designated as the region of interest and the crop of the image. In order to collect driver behavior data from various angles, cameras were installed in four places in the front interior side of the car, as shown in Fig. 5 Abko Apc720 Lite and Logitech Quickcam Pro 9000 were used as camera sensors, and the image sampling rate is 30 frames per second(fps). As a data collection environment, the CPU used Intel Core i7 2.3Ghz and Python Open CV. The collected images are used as training data after extracting the region of interest through the face detector. The overall configuration of the dataset is shown in Table I.

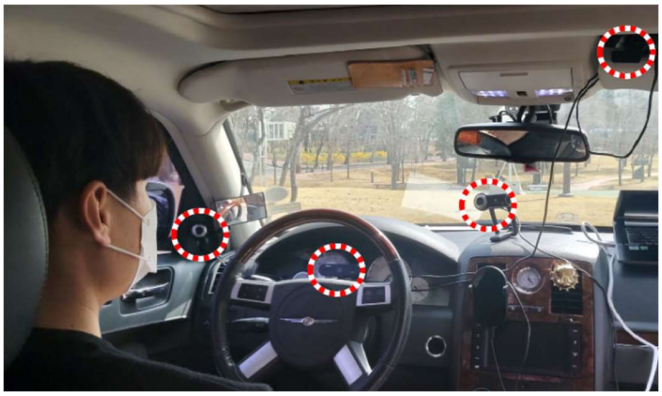

Fig. 5. Data collection environment.

TABLE I
Driver Behavior Data Set Used in The Experiment

Class	Training data	Validation data	Total
Eat	3,146	2,050	5196
Normal	4,782	1,130	5,912
Phone	3,204	1,400	4,604
Smoke	4,414	1,030	5,444

As the sensor used in the experiment, the Elp-Usbfhd05mt-kl170ir model was used as a single camera sensor. The Elp camera has a field of view of 180°, a resolution of 1920×1080 size, and a speed of up to 30 FPS. As an environment for NAS, Intel Core i7 3.60GHz, Nvidia GeForce RTX 3070 is used.

(1) Training for RNN: RNN consists of 32 LSTM units and uses Adam as an optimization algorithm to train the model. 0.00035, 0.9, 0.999, and 0.001 were used as the learning rate, the first momentum, the second momentum, and the L2 weight decay, respectively. The total number of nodes sampled in LSTM is 5 in the first operator space and 6 in the second operator space. The batch size of the RNN is 1, and the RNN is trained using the validation accuracy of the network generated for a total of 150 epochs as a reward.

(2) Training for the generated network: One network is generated per epoch for a total of 150 epochs. The generated network uses SGD, learning rate and momentum are 0.05 and 0.9, respectively, and the batch size is 8.

(3) Training for the optimal neural network: After the neural network search is completed, a network with the highest validation accuracy is generated. We use Adam to train the proposed model. The learning rate is 0.001, and if the validation accuracy of the model does not increase within 10 times, the learning rate is halved with a batch size of 8, and the model is trained for 100 epochs.

In this paper, Dropout [10], Stochastic Depth [11], and RandAugment [12] are used as techniques to prevent overfitting and increase model performance. First, dropout is applied only to inverted bottleneck conv, and the ratio is set to 0.1. Stochastic Depth is applied to the operator corresponding to convolution among operators and is removed while the operator is learning at a rate of 0.1. The magnitude of RandAugment is 7.

Table II shows the layer configuration of the proposed network. It was constructed by sequentially stacking normal cell and reduction cell. The number of convolutional filters starts with 8 and doubles the number of filters in the reduction cell. The input size is reduced from 96 to 6, and the driver's behavior is classified through the softmax function through global average pooling (GAP). In Fig. 6, if $h[i-1]$ has a larger input size than $h[i]$, the input size is reduced by half through 3×3 operation, and then used as the input for the next operation.

TABLE II
Proposed Architecture

Proposed Model	Filter	Feature map size
Normal Cell	8	96×96
Reduction Cell	16	48×48
Normal Cell	16	48×48
Reduction Cell	32	24×24
Normal Cell	32	24×24
Reduction Cell	64	12×12
Normal Cell	64	12×12
Reduction Cell	128	6×6
Normal Cell	128	6×6
GAP		1×1
1028-dim FC, softmax		

Table III compares the performance of our model with the existing deep learning classification model. The proposed model#1 is a model searched using the first proposed operator space, and the proposed model#2 is the second proposed operator space. For a fair experiment, all experiments were performed under the same conditions, and in the case of accuracy, five averages were measured. MobileNetV2, used in the experiment, reduced the input size to 96×96 and the number of layers for comparison with the proposed model and then compared the performance with the proposed model. The proposed model showed about 8% higher performance than MobileNetV2 despite having fewer of parameters.

TABLE III
Performance Comparison of Classification Model

Model	Params	Acc
MobileNetV2	0.92M	83.69
ENAS	3.6M	88.48
Proposed Model #1	0.96M	91.12
Proposed Model #2	0.77M	92.08

In addition, it showed higher performance when compared with ENAS, which is a conventional neural network structure search technique. Comparing the proposed models #1 and #2, they have similar performance, but varying the operator space shows slightly better performance.

V. Conclusions

In this paper, the operator of the architecture search space was modified to search the neural network structure for the driver behavior classification model. The proposed algorithm constructed a more efficient network in the trade-off relationship between accuracy and inference speed. As a result, it showed higher performance than the existing classification model on our data set collected through experiments. In a future study, performance verification when using the proposed model in an actual embedded device is required, and a method for generating driver behavior data to reduce overfitting will be required.

Acknowledgment

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2021-2020-0-01808) supervised by the IITP (Institute of Information & Communications Technology Planning & Evaluation); and the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2021R1A6A1A03043144).

References

[1] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).

[2] Sandler, Mark, et al. "Mobilenetv2: Inverted residuals and linear bottlenecks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[3] Zhang, Xiangyu, et al. "Shufflenet: An extremely efficient convolutional neural network for mobile devices." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[4] Ma, Ningning, et al. "Shufflenet v2: Practical guidelines for efficient CNN architecture design." Proceedings of the European conference on computer vision (ECCV). 2018.

[5] Elsken, Thomas, Jan Hendrik Metzen, and Frank Hutter. "Neural architecture search: A survey." The Journal of Machine Learning Research 20.1 (2019): 1997-2017.

[6] Pham, Hieu, et al. "Efficient neural architecture search via parameters sharing." International Conference on Machine Learning. PMLR, 2018.

[7] Hu, Jie, Li Shen, and Gang Sun. "Squeeze-and-excitation networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[8] Woo, Sanghyun, et al. "Cbam: Convolutional block attention module." Proceedings of the European conference on computer vision (ECCV). 2018

[9] Wang, Fei, et al. "Residual attention network for image classification." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[10] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The journal of machine learning research 15.1 (2014): 1929-1958.

[11] Huang, Gao, et al. "Deep networks with stochastic depth." European conference on computer vision. Springer, Cham, 2016.

[12] Cubuk, Ekin D., et al. "Randaugment: Practical automated data augmentation with a reduced search space." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops.

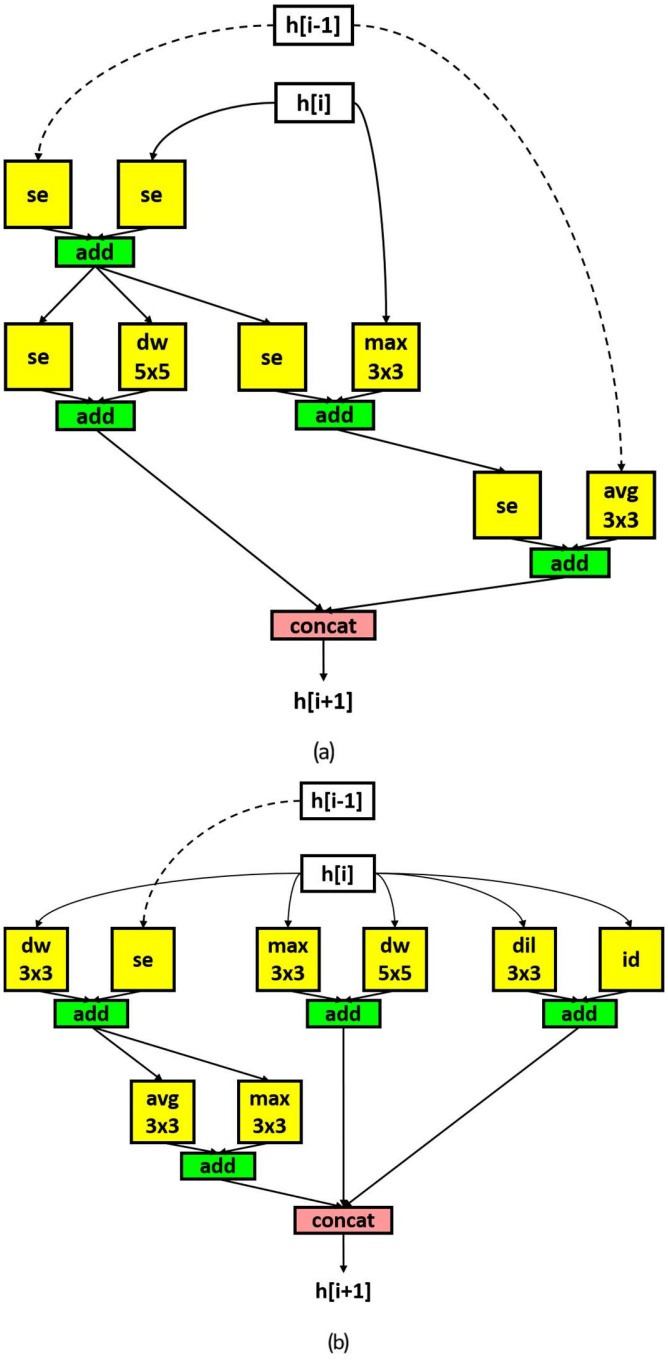

Fig. 6. Cells of proposed model#1 discovered in search space for (a) normal cell and (b) reduction cell.