Att-DARTS: Differentiable Neural Architecture Search for Attention

Kohei Nakai*, Takashi Matsubara† and Kuniaki Uehara‡
*Graduate School of System Informatics, Kobe University
1-1 Rokkodai, Nada, Kobe, Hyogo, 657-8501 Japan.
†Graduate School of Engineering Science, Osaka University
1-3, Machikaneyama, Toyonaka, Osaka, 560-8531 Japan.
‡Faculty of Business Administration, Osaka Gakuin University
2-36-1, Kishibe-Minami, Suita, Osaka, 564-8511 Japan.
Emails: k.nakai@ai.cs.kobe-u.ac.jp, matsubara@sys.es.osaka-u.ac.jp, kuniaki.uehara@ogu.ac.jp

***Abstract*—Neural architecture search (NAS) is a promising method to automatically identify neural network architectures. Differentiable architecture search (DARTS) is method that significantly reduces search time and finds architectures that can achieve state-of-the-art performance. For computer vision tasks, DARTS searches convolutional neural networks (CNNs) via stacking convolution layers and pooling operations. Recent studies on neural architectures indicate that attention modules can improve the performances of CNNs by discarding information of no interest, while existing NAS methods have put little focus on it. In this study, we propose Att-DARTS, which searches attention modules as well as convolution and pooling operations simultaneously. In our experiments on CIFAR-10 and CIFAR-100 datasets, we demonstrate that Att-DARTS can find architectures that achieve lower classification error rates and require fewer parameters compared to those found by DARTS.**

***Index Terms*—neural architecture search, object recognition, attention mechanism**

I. Introduction

Deep learning has enabled us to solve various tasks with high performance thanks to its flexibility and rich expressive ability. The flexibility and expressive ability come from its architecture designed for targeted tasks. For instance, convolutional neural networks (CNNs) and their extensions are commonly used for computer vision tasks [1]. For building a neural architecture, one has to design it manually by following a trial-and-error approach. Manually designing architectures requires a considerable amount of expertise and time to ensure that they achieve state-of-the-art performances.

The goal of neural architecture search (NAS) is to automate this time-consuming and error-prone process [2]. The recent development of NAS has enabled us to automatically design an architecture that achieves good performance in tasks such as semantic image segmentation [3] and person re-IDentification [4]. A notable difficult of NAS is ensuring the discreteness of the search space over operations such as convolution and pooling. A selection of one operation among several candidate operations needs to be a discrete and indifferentiable action. Therefore, many of the earliest NAS works used reinforcement learning and evolutionary algorithms [5]–[8].

This study was partially supported by the MIC/SCOPE #172107101 and JSPS KAKENHI (19H04172, 19K20344).

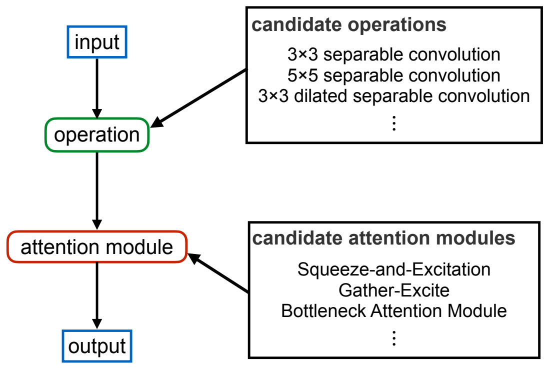

input
operation
attention module
output
candidate operations
3×3 separable convolution
5×5 separable convolution
3×3 dilated separable convolution
⋮
candidate attention modules
Squeeze-and-Excitation
Gather-Excite
Bottleneck Attention Module
⋮

Fig. 1: An overview of Att-DARTS. We propose applying not only an operation (convolution or pooling) but also an attention module after the operation is applied. Att-DARTS chooses both operations and attention modules from each candidate space.

These methods trained a new candidate architecture from scratch and compared its performance with existing ones; however, these processes are computationally expensive and time consuming, often requiring thousands of GPU days. ENAS reduced search time to within a GPU day by sharing parameters among all candidate architectures [9]. A breakthrough was led by gradient-based approaches such as DARTS, which builds a neural network as a weighted sum of all candidate operations and trains the relative weights among the candidates and their internal parameters [10]. Compared with ENAS, DARTS achieved better results with slightly longer search times.

Of the three dimensions for NAS: search space, search strategy, and performance estimation strategy [2], most earlier NAS works focus on the search strategy. In this study, our focus is on the search space. For computer vision tasks, we use CNNs, whose main operation is convolution. A neural network can capture the spatial and channel-wise features of an image applying a convolution. However, the neural network considers all spatial and channel-wise features including the ones that are not useful, when using only convolutions. To discard such useless features and focus on important features, attention modules have been proposed [11]–[17]. Studies that employed CNNs with attention modules reported better performances with a comparable number of parameters. We consider that attention modules are promising candidates for neural architectures and can be employed in the search space.

In this work, we propose a novel search method called Att-DARTS that can find architectures using attention modules. Att-DARTS assumes a CNN composed of repeatedly stacked cells similar to most existing NAS works; however, it inserts an attention module after each operation. During the search process, Att-DARTS optimizes the relative weights of both operations and attention modules in addition to internal parameters using gradient descent like DARTS [10]. After the search process, Att-DARTS chooses the operations and attention modules with the strongest relative weights and builds a final cell architecture.

We searched the architecture with attention modules on an image dataset, CIFAR-10, and evaluated the identified cells on CIFAR-10 and CIFAR-100 [18]. In our experiments, we found that Att-DARTS found a cell that achieves a 2.54% test error on CIFAR-10 (around 10% lower than that of DARTS) and a 16.54% test error rate on CIFAR-100. Simultaneously, Att-DARTS reduces the number of parameters by 0.1M compared to DARTS.

The remainder of this paper is organized as follows. Section II is devoted to introducing related works on the neural architecture search and attention modules. Section III shows the details of our proposed method, Att-DARTS. Section IV provides details on the experimental results on CIFAR-10 and CIFAR-100. In Section V, our experiments demonstrates that Att-DARTS outperforms DARTS and many comparative methods. Finally, Section VI provides the conclusion of the paper.

II. Related Work

A. Neural Architecture Search

A NAS aims to automate the design of neural network architectures and to reduce experts' efforts. Early NAS works are based on reinforcement learning or evolutionary algorithms [5]–[8]. Zoph *et al.* [5] generated optimal hyperparameters of each convolutional layer (e.g., filter size, and stride) and the number of layers using a controller recurrent neural network (RNN). The RNN is trained with reinforcement learning to maximize the accuracy of a child network. Other studies employed evolutionary algorithms such as AmoebaNet [7].

These methods train a new candidate architecture from scratch and compare it with the existing ones; these processes are computationally expensive and time consuming. Some works reduced the search time by using a one-shot model, which is a single model composed of all candidate operations [9], [19], [20]. The model facilitated sharing weight parameters among all candidate architectures. At the search stage, we only need to train this single model instead of all candidate models from scratch. Instead of designing an entire neural network architecture, NASNet [6] designed an architecture composed of repeated small networks, each of which is called a cell, and this helped to drastically reduce the search space.

Many of these methods used reinforcement learning or evolutionary algorithms to design a neural network architecture topology, whereas we train a neural network using the gradient descent method. Hence, architecture search and parameter optimization can be performed in alternatively, which increases computational time. To bridge this gap, gradient-based methods such as DARTS [10] have been proposed.

Gradient-based methods make the operation search space continuous by implementing an operation as a weighted sum of all candidate operations. DARTS jointly trains the relative weights of the operations and their internal parameters using the gradient descent method, and it chooses the cells with the strongest relative weights. The best cells found by DARTS achieved results comparable with those found by methods based on reinforcement learning or evolutionary algorithms.

Various methods inspired by DARTS have been proposed. To reduce memory consumption during the search space, DARTS employed a shallower network architecture at the search stage than at the evaluation stage. P-DARTS [21] bridged this depth gap by reducing the pattern of candidate operations step-by-step. PC-DARTS [22] proposed a partial channel connection and improved memory efficiency. Some works focus on the issue that DARTS tends to choose many skip connections [21], [23], [24]. However, these studies focused on the search strategy. In contrast, our study focuses on the search space.

B. Differentiable Architecture Search (DARTS)

In this section, we provide detailed information on DARTS [10] for comparing it with our proposal. DARTS searches for a computation cell, and the entire architecture obtained using DARTS is a chain-structured network, where L cells are stacked. Each cell c_k is defined as a directed acyclic graph with N nodes, where node x_i is connected to nodes x_j that have larger indices $i < j$. Each node x_i is a feature map. Two of their nodes are inputs, a node is an output, and the remaining $N-3$ nodes are intermediate nodes. The input nodes are obtained from the output nodes of the previous two cells c_{k-1} and c_{k-2}. The output node is defined as the depth-wise concatenation of all intermediate nodes.

During the architecture search, DARTS finds an operation in the operation space $\mathcal{O}$ for each edge. When the focus is on the edge between nodes x_i and x_j, each candidate operation $o(\cdot) \in \mathcal{O}$ has a relative weight $s_o^{(i,j)}$, which is defined by the softmax operation over the operation parameters $\alpha_o^{(i,j)}$ of all candidate operations:

$$s_o^{(i,j)} = \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})}. \tag{1}$$

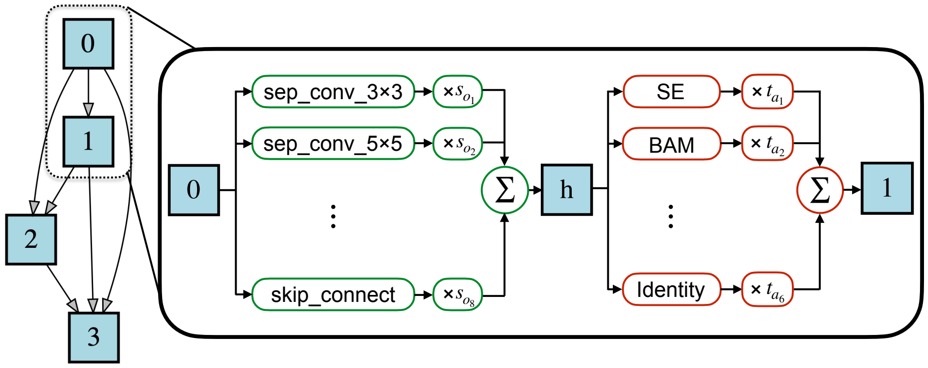


Fig. 2: Illustration of Att-DARTS.

The information flow from node x_i to node x_j is computed as the weighted sum of all candidate operations:

$$\bar{o}^{(i,j)}(x_i) = \sum_{o \in \mathcal{O}} s_o^{(i,j)} o(x_i) \tag{2}$$

Node x_j is computed as the sum of all flows from the source nodes in the cell:

$$x_j = \sum_{i<j} \bar{o}^{(i,j)}(x_i). \tag{3}$$

Then, DARTS jointly learns operation parameters α and weight parameters w (e.g., convolution kernels) using the gradient descent method by solving a bilevel optimization problem, where operation parameters α are upper-level variables and weight parameters w are lower-level variables:

$$\min_{\alpha} \ \mathcal{L}_{val}(w^*(\alpha), \alpha) \tag{4}$$

$$s.t. \ w^*(\alpha) = \underset{w}{\operatorname{argmin}} \ \mathcal{L}_{train}(w, \alpha). \tag{5}$$

$\mathcal{L}_{train}$ and $\mathcal{L}_{val}$ denote the training and validation losses.

After the architecture search, DARTS chooses an operation $o(\cdot)$ according to the relative weight $s_o^{(i,j)}$. For each intermediate node x_j, DARTS chooses the operation o with the strongest weight $s_o^{(i,j)}$ for each edge, and it then chooses two edges with the strongest weights $s_o^{(i,j)}$ from all edges connected to node x_j. Hence, DARTS keeps $2 \times (N-3)$ operations in a cell, and the information flow to node x_i is defined as

$$x_j = \sum_{(o,x_i) \in C_j} o^{(i,j)}(x_i). \tag{6}$$

where C_j is a set of chosen edges and their operations connected to node x_j.

C. Attention Modules

Convolutional neural networks have achieved significant results in image classification tasks. The expressive power of CNNs increases with the network size, and many studies have been conducted to investigate a trainable deeper and wider architecture [1]. The main operation of CNNs is convolution, and by repeating convolutions, CNNs can extract a larger amount of spatial and channel-wise features, some of which are often useless or conflict with important features.

Attention modules have been proposed to discard such features and focus on important ones; further, attention modules help CNNs achieve a better performance and make them robust against disturbances [11]–[17].

Wang et al. [11] proposed a residual attention network, which is composed of trunk and mask branches. The trunk branch extracts features similar to ordinary convolutions. The mask branch builds an attention mask and multiplies the output of the trunk branch. Many following studies proposed attention modules that focus on the mask and are used after a convolution operation. Squeeze-and-excitation (SE) [12] proposed a lighter attention module, which has channel attention that enables the network to focus on important channels. Gather-excite (GE) [13] added a depth-wise convolution to SE and increased its flexibility. BAM [14] and CBAM [15] have spatial attention and channel attention. BAM builds a mask as a sum of its channel and spatial attentions, whereas CBAM sequentially applies channel and spatial attentions. A^2-Net [16] has a bilinear pooling operation to reflect the different needs in different locations.

III. Method

A. Architecture Search with Attention Modules

In this section, we propose a method—Att-DARTS—that searches for a neural architecture using attention modules, as illustrated in Fig. 2. As with many NAS works, the entire network consists of repeated cells, and Att-DARTS searches the good cells. Each cell c_k is expressed as a directed acyclic graph with N nodes, two of which are inputs and one is the output. The remaining $N-3$ nodes are intermediate nodes. Each node x_i is a feature map. The input nodes are obtained from the output nodes of the previous two cells c_{k-1} and c_{k-2}. The output node is defined as the depth-wise concatenation of all intermediate nodes in the cell.

Att-DARTS searches the cell including attention modules as well as operations. Original studies on attentions proposed several locations to insert attention modules; however, the most popular location is immediately after a convolution operation (e.g., see [12], [15]). Following this approach, Att-DARTS assumes an edge composed of an operation preceding an attention module.

At the search stage, Att-DARTS identifies an operation in the operation space $\mathcal{O}$, and an attention module in the attention module space $\mathcal{A}$. When the focus is on the edge from node x_i to node x_j, each candidate operation $o(\cdot) \in \mathcal{O}$ has a relative weight $s_o^{(i,j)}$, which is defined by the softmax operation over the operation parameters $\alpha_o^{(i,j)}$ of all candidate operations:

$$s_o^{(i,j)} = \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})}. \tag{7}$$

Each candidate attention module $a(\cdot) \in \mathcal{A}$ has a relative weight $t_a^{(i,j)}$, which is also defined by the softmax operation over the attention module parameters $\beta_a^{(i,j)}$ of all candidate attention modules:

$$t_a^{(i,j)} = \frac{\exp(\beta_a^{(i,j)})}{\sum_{a' \in \mathcal{A}} \exp(\beta_{a'}^{(i,j)})}. \tag{8}$$

We introduce a new node $h_{i,j}$ on the edge from node x_i to node x_j. Similar to that in DARTS, the information flow from node x_i to node $h_{i,j}$ is computed as the weighted sum of all candidate operations:

$$h_{i,j}(x_i) = \sum_{o \in \mathcal{O}} s_o^{(i,j)} o(x_i). \tag{9}$$

The information flow from $h_{i,j}$ to x_j is computed as the weighted sum of all candidate attention modules:

$$\bar{a}^{(i,j)}(h_{i,j}) = \sum_{a \in \mathcal{A}} t_a^{(i,j)} a(h_{i,j}) \tag{10}$$

Node x_j is computed as the sum of all flows from source nodes $h_{i,j}$ in the cell:

$$x_j = \sum_{i<j} \bar{a}^{(i,j)}(h_{i,j}) \tag{11}$$

Then, we learn the operation parameters α, attention parameters β, and weight parameters w (e.g., convolution kernels) simultaneously, using the gradient descent by solving the bilevel optimization problem, where the operation parameters α and the attention parameters β are the upper-level variables and the weight parameters w are the lower-level variables.

$$\min_{\alpha,\beta} \ \mathcal{L}_{val}(w^*(\alpha,\beta),\alpha,\beta) \tag{12}$$

$$s.t. \ w^*(\alpha,\beta) = \operatorname*{argmin}_{w} \ \mathcal{L}_{train}(w,\alpha,\beta) \tag{13}$$

$\mathcal{L}_{train}$ and $\mathcal{L}_{val}$ denote the training and validation losses, respectively.

After the search stage, Att-DARTS chooses operation $o(\cdot)$ for each edge from node x_i to x_j with the strongest relative weight $s_o^{(i,j)}$. Further, Att-DARTS chooses attention module $a(\cdot)$ for each edge with the strongest relative weight $t_a^{(i,j)}$. Then, Att-DARTS chooses two edges with the strongest operation weights $s_o^{(i,j)}$ from all edges connected to node x_j. Att-DARTS maintains $2 \times (N-3)$ operations and attention modules in a cell. The information flow to node x_j is defined as

$$x_j = \sum_{(o,a,x_i) \in C_j} a^{(i,j)}(o^{(i,j)}(x_i)), \tag{14}$$

where C_j is a set of the chosen edges connected to node x_j including their operations and attention modules.

IV. Experiments

A. Experimental Details

We searched a neural architecture on CIFAR-10 and evaluated it on CIFAR-10 and CIFAR-100 [18]. CIFAR-10 consists of 10 class RGB images, and CIFAR-100 consists of 100 class RGB images. Both have 50,000 training images and 10,000 test images; the images have a resolution of 32×32 pixels.

At the search stage, we searched for cells. We followed the same experimental setting as that of the original study on DARTS unless otherwise stated [10]. We independently executed Att-DARTS 4 times with different random seeds for 50 epochs with batch size 64. We randomly split 50,000 training images into training and validation sets of equal sizes. We set the number of cells $L = 8$; two reduction cells and six normal cells. The reduction cells were inserted into the 1/3 and 2/3 locations of the entire network. Operations were with stride 1 in the normal cells and with stride 2 in the reduction cells; hence, the image size was halved at the reduction cells. Each cell consisted of $N = 7$ nodes. The initial number of channels was set to 16, and it was doubled at the reduction cells.

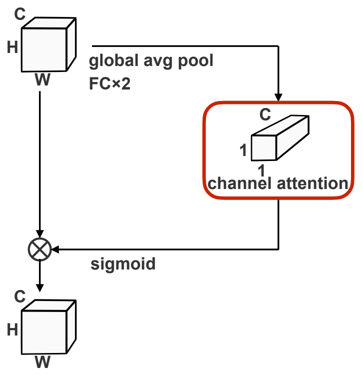
We used zero initialization for both operation and attention parameters. We used momentum SGD to optimize weight parameters w with an initial learning rate 0.025, momentum 0.9, and weight decay 3.0×10^{-4}. The learning rate was annealed down to zero following a cosine schedule [25]. We used the Adam optimizer [26] for the operation parameters α and attention parameters β with an initial learning rate 3.0×10^{-4}, momentum $(0.5, 0.999)$, and weight decay 1.0×10^{-3}.

After the search stage, we obtained candidate cells for each iteration by choosing operations and attention modules according to the operation parameters α and attention parameters β. We built a neural network composed of candidate cells from scratch and trained it with the same experimental setting as the evaluation stage. Then, we chose the best cells among all four runs based on the best validation accuracy within the 600 epochs, while DARTS chose the best cells based on the validation accuracy at the 100th epoch.

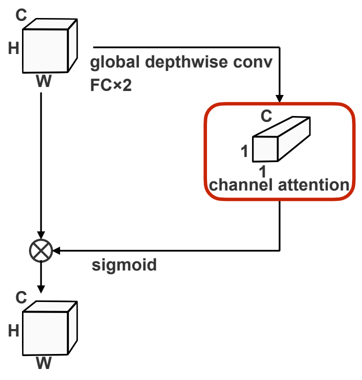
The evaluation is performed as follows. We used all 50,000 training images and set the number of cells $L = 20$; 2 reduction cells and 18 normal cells. The initial number of channels was set to 36, and they doubled at the reduction cells. We then applied cutout [27], path dropout [28] of probability 0.2, and auxiliary towers [29] of weight 0.4. We independently trained the best cells 5 times for 600 epochs with batch size 96, and then reported the mean accuracy and standard deviation.

B. Architecture Search Space

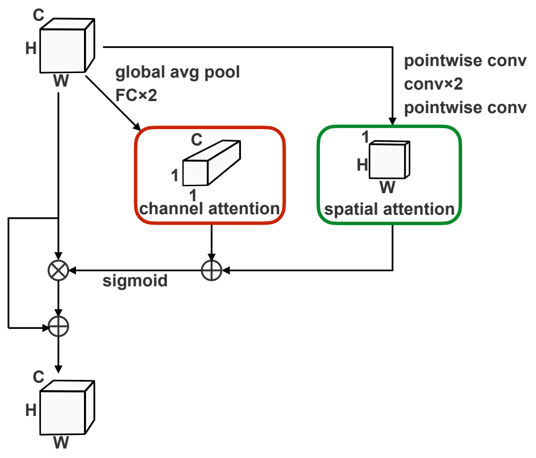
The operation space $\mathcal{O}$ was the same as that of DARTS: Identity, 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated separable convolutions, 3×3 max pooling, 3×3 average pooling, and *zero*. *Zero* indicates the absence of a connection



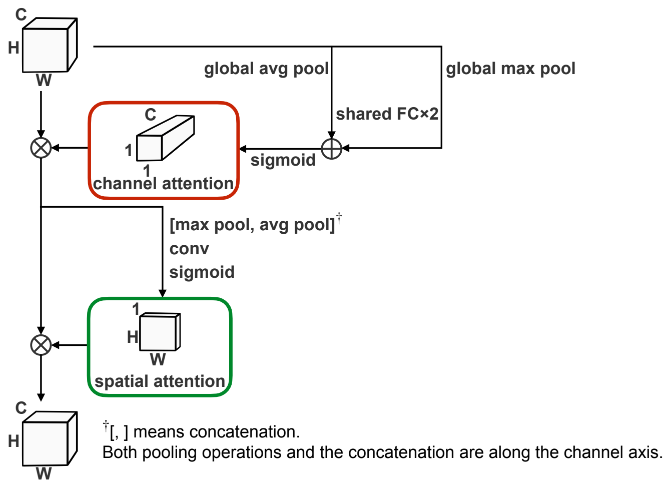
(a) Squeeze-and-Excitation (SE).



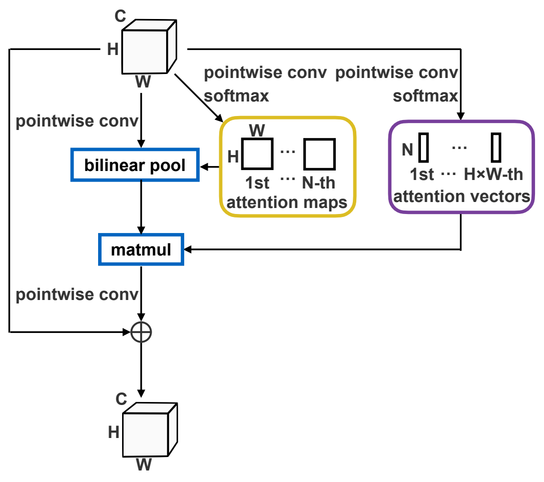
(b) Gather-Excite (GE-θ^{+}).



(c) Bottleneck Attention Module (BAM).



(d) Convolutional Block Attention Module (CBAM).



(e) Double-attention block (A^2-block).

Fig. 3: Overview of each attention module. $\otimes$ and $\oplus$ mean the element-wise multiplication and the element-wise summation, respectively.

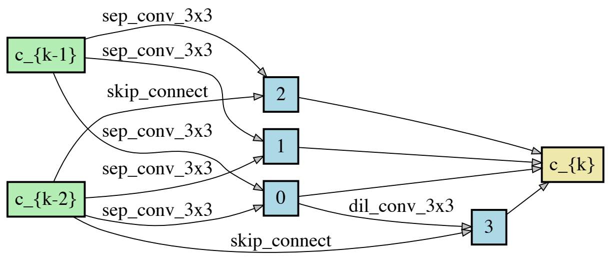
and it is excluded from the chosen operations. A depth-wise separable convolution [30], [31] is the computational- and memory-efficient version of a convolution. It splits the convolution into two layers: depth-wise convolution and point-wise convolution. The depth-wise convolution is applied to each channel independently; the pointwise convolution is a convolution with a kernel size of 1. A dilated convolution [32] is a convolution whose kernel is applied to every l space points (l is a dilation factor).

The attention space $\mathcal{A}$ includes Identity, SE [12], GE-θ^{+} [13], BAM [14], CBAM [15], and double-attention block (A^2-block) [16]. We show overviews of each attention module in Fig. 3.

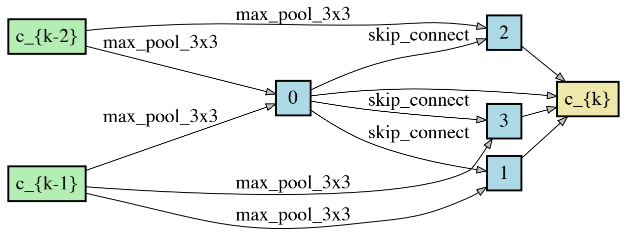
As shown in Fig. 3 (a), SE has a channel attention that enables the network to focus on important channels. The SE gathers spatial features of the input feature map by using a global averaging pooling, applies two fully connected layers, and employs the sigmoid function, thereby obtaining an attention mask with a spatial size of 1. The fully connected layers reduce the number of channels from C to C/r, where r is a reduction ratio. GE-θ^{+} applies a global depth-wise convolution instead of the global averaging pooling as shown in Fig. 3 (b).

BAM is a combination of channel and spatial attentions (see Fig. 3 (c)). The spatial attention mask has the same size as that of the input, while its number of channels is 1; the spatial attention enables a network to focus on important spatial positions. In particular, spatial attention is composed of four layers. The first point-wise convolution reduces the number of channels from C to C/r, two dilated convolutions with a dilation value d are applied, and then, the second one reduces it from C/r to 1. The channel attention of BAM is the same as that of SE. The attention mask of BAM is the sum of the outputs of the spatial and channel attentions. BAM multiplies the input tensor using the attention mask element, and it adds the output to the input tensor.

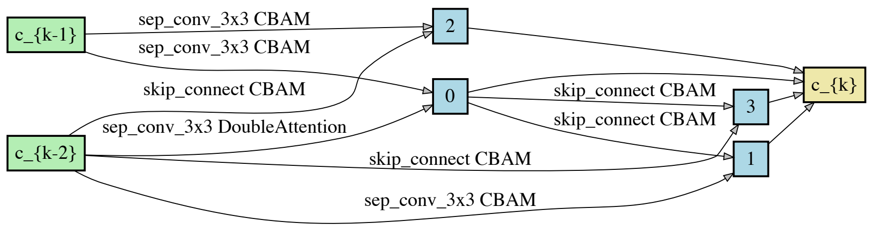
CBAM is also a combination of channel and spatial attentions as shown in Fig. 3 (d). CBAM gathers spatial features of the input feature map by max pooling in addition to global



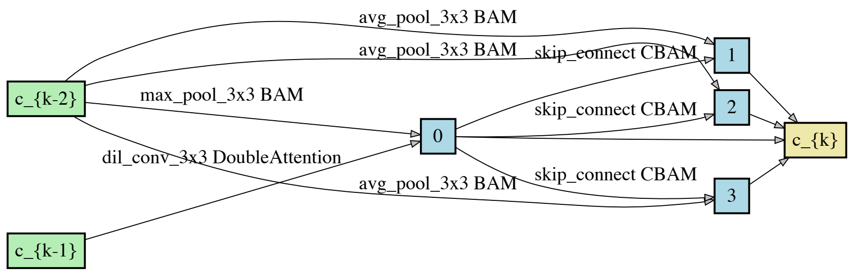
(a) Normal cell by DARTS.



(b) Reduction cell by DARTS.



(c) Normal cell by Att-DARTS.



(d) Reduction cell by Att-DARTS.

Fig. 4: Illustrations of cells found by DARTS and Att-DARTS. Att-DARTS can find cells wherein the attention module is inserted after each operation. sep_conv_3x3 and sep_conv_5x5 denote 3×3 and 5×5 separable convolutions respectively. dil_conv_3x3 and dil_conv_5x5 denote 3×3 and 5×5 dilated separable convolutions respectively. max_pool_3x3 denotes 3×3 max pooling. avg_pool_3x3 denotes 3×3 average pooling.

average pooling, and applies a shared MLP. The shared MLP has the same size as the one used in the channel attention in SE. CBAM applies the sigmoid function to the sum of the outputs of the shared MLP, and it obtains the channel attention mask. After channel attention, CBAM applies spatial attention, which applies the max and average poolings along with the channel axis, concatenates the outputs, applies a convolution and the sigmoid function, and obtains the spatial attention mask.

Fig. 3 (e) shows a double-attention block (A^2-block). The double-attention block uses two groups of attentions: attention maps and attention vectors. Each attention is obtained by the softmax function following a pointwise convolution that reduces the number of channels from C to N. The softmax function is applied over the spatial direction for the attention maps, and over the channel direction for the attention vectors. Further, double-attention block applies a pointwise convolution to the input to reduce the number of channels from C to M, and it applies bilinear pooling using the attention maps. Then, double-attention block applies the output matrix multiplication using the attention vectors. Finally, pointwise convolution restores the number of channels from M to C. Note that the double-attention block adds the output to the input, whereas the other attention modules multiply the input by the output. We use the same hyperparameter set of each attention as the one in the original papers. For example, the reduction ratio r of SE was set to 16. Dilation value d of a convolution in the spatial attention of BAM was 4. Reduced channel numbers M and N in the double-attention block were set to $C/4$.

V. Results and Discussion

A. Chosen Architecture

We illustrate the found cells in Fig. 4. The skip connection is chosen as an operation more frequently by Att-DARTS than by DARTS, which means that attention modules are repeatedly applied without convolution operations (e.g., from node c_{k-1} to node 1 via node 0). These attention modules have different parameters, and the network can focus on more specific features. In contrast, Identity is never chosen as an attention module despite it corresponding to a skip connection as an operation. Hence, one can say that Att-DARTS focuses on the attention modules.

TABLE I: Comparison with DARTS on CIFAR-10 and CIFAR-100

Architecture	**Test Error on CIFAR-10 (%)**	**Test Error on CIFAR-100 (%)**	**Params (M)**	**Search Method**
DARTS + cutout [10]	2.76 ± 0.09	$16.69 \pm 0.28^{\dagger}$	3.3	gradient
Att-DARTS + cutout	$\mathbf{2.54} \pm 0.10$	$\mathbf{16.54} \pm 0.40$	$\mathbf{3.2}$	gradient

†Results by the best cells for CIFAR-10 reported in the original paper [10] as is the case with our Att-DARTS.

TABLE II: Comparison with state-of-the-art architectures on CIFAR-10

Architecture	**Test Error (%)**	**Params (M)**	**Search Method**
DenseNet-BC [33]	3.46	25.6	manual
NASNet-A + cutout [6]	2.65	3.3	RL†
AmoebaNet-B + cutout [7]	2.55 ± 0.05	2.8	evolution
Hierarchical Evolution [8]	3.75 ± 0.12	15.7	evolution
PNAS [34]	3.41 ± 0.09	3.2	SMBO†
ENAS + cutout [9]	2.89	4.6	RL†
DARTS + cutout [10]	2.76 ± 0.09	3.3	gradient
SNAS(moderate) + cutout [35]	2.85 ± 0.02	2.8	gradient
BayesNAS + cutout [36]	2.81 ± 0.04	3.4	gradient
PC-DARTS + cutout [22]	2.57 ± 0.07	3.6	gradient
Att-DARTS + cutout	$\mathbf{2.54} \pm 0.10$	3.2	gradient

†RL means reinforcement learning, and SMBO means sequential model-based optimization.

CBAM and BAM are the most and second-most frequently selected attention modules, while SE and GE-θ^{+} are never selected. SE and GE-θ^{+} have only channel attentions, and their contributions are limited compared to CBAM and BAM, which have both channel and spatial attentions. This result emphasizes the importance of spatial attention. Despite CBAM and BAM having many convolution layers, Att-DARTS reduces the number of weight parameters; the entire network requires 3.2M weight parameters with Att-DARTS and 3.3M with DARTS. This is because the number of channels and spatial resolution are reduced in the attention modules.

DARTS and Att-DARTS chose separable convolutions as operations most frequently. A separable convolution can be considered as a convolution whose kernel is factorized and requires less parameters. DARTS chose only max pooling among pooling operations in the reduction cells, whereas Att-DARTS chose average pooling. Max pooling selects the most significant pixel in its kernel, as it is similar to an attention module. Att-DARTS learns to select features by the attention modules and does not need max pooling.

B. Architecture Evaluation

We summarize the evaluation results on CIFAR-10 in Table I. We report the average and standard deviations of five independent runs for CIFAR-10. Att-DARTS achieved a 2.54% test error with 3.2M parameters; Att-DARTS improved the error rate by 0.22% and reduced the number of parameters by 0.1M from DARTS. We also evaluated the best cells on CIFAR-100 as summarized in Table I. We report the average and standard deviations of five independent runs for CIFAR-100. Att-DARTS also improved the error rate by 0.15%.

We summarize the comparison with state-of-the-art architectures on CIFAR-10 in Table II. Att-DARTS achieved the state-of-the-art results among NAS methods. AmoebaNet-B and PC-DARTS achieved comparable results, while the former requires thousands of GPU days and the latter requires 13% more parameters. Moreover, Att-DARTS's approach to search attention modules can be combined with any gradient-based NAS methods such as PC-DARTS and potentially improves their performances.

Note that the architecture search space $\mathcal{O} \times \mathcal{A}$ of Att-DARTS is the product of the operation space $\mathcal{O}$ and the attention module space $\mathcal{A}$. Hence, architecture search based on reinforcement learning and evolutionary algorithm encounters difficulty in finding a good architecture compared with the case only in the operation space $\mathcal{O}$.

C. Comparison with An Alternative

We conducted additional experiments using another search method. The main purpose of the attention modules is to emphasize the information of interest in feature maps while discarding useless information. For this purpose, most attention modules multiply the given feature maps by their attention masks as follows.

$$\bar{a}^{(i,j)}(h_{i,j}) = a_m(h_{i,j}) \otimes h_{i,j} \tag{15}$$

where $a_m(\cdot)$, $h_{i,j}$ and $\otimes$ denote an attention mask, an input feature map, and the element-wise product respectively. During the search stage, each candidate attention module is applied independently to the output of the preceding operation, and the output is calculated as the weighted sum of all candidate attention modules, as expressed in Eq. (10). In other words, the attention mask is the weighted sum of all candidate masks as follows.

$$\bar{a}^{(i,j)}(h_{i,j}) = \left(\sum_{a \in \mathcal{A}} t_a^{(i,j)} a_m(h_{i,j}) \right) \otimes h_{i,j} \tag{16}$$

TABLE III: Comparison of Att-DARTS architectures on CIFAR-10

Architecture	Test Error (%)	Params (M)
Att-DARTS-S + cutout	$\mathbf{2.54} \pm 0.10$	3.2
Att-DARTS-P + cutout	2.92 ± 0.10	$\mathbf{2.7}$

where $t_a^{(i,j)}$ denotes a relative weight for a candidate attention module $a(\cdot)$, as shown in Eq. (8). If two candidate attention modules learn to discard different channels, both channels are discarded and retained halfway. To emphasize the contribution of masks, we also propose an alternative weighted sum in the log-scale as follows.

$$\bar{a}^{(i,j)}(h_{i,j}) = \exp\left(\sum_{a \in \mathcal{A}} t_a^{(i,j)} \log(a_m(h_{i,j}))\right) \otimes h_{i,j}. \quad (17)$$

If one attention module learns to discard a channel, the channel is always discarded. We name this as Att-DARTS-P and name the previous one in Eq. (10) Att-DARTS-S. We excluded the double-attention block because it adds its mask unlike others.

We report the result in Table III. Unfortunately, Att-DARTS-P is inferior to Att-DARTS-S. When an attention module learns to discard a channel, the channel is always discarded from the output and is never restored during the architecture search, potentially converging to a steep local minima. The fact that the number of parameters was drastically reduced supports this hypothesis.

VI. Conclusion

In this work, we proposed Att-DARTS, a differentiable architecture search that finds cells with attention modules. Att-DARTS assumes a CNN composed of repeatedly stacked cells similar to most existing NAS works; however, it inserts an attention module after each operation, unlike the previous works. The experimental results on CIFAR-10 and CIFAR-100 demonstrated that Att-DARTS found an architecture that achieves a better classification error rates and requires less parameters than that found by its non-attention counterpart, DARTS. Future works include evaluations on a larger CNN and a larger dataset, and a more flexible cell topology.

References

[1] K. He *et al.*, "Deep Residual Learning for Image Recognition," in *CVPR*, 2016, pp. 770–778.

[2] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *Journal of Machine Learning Research*, vol. 20, 2019.

[3] C. Liu *et al.*, "Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation," *CVPR*, 2019.

[4] R. Quan *et al.*, "Auto-ReID: Searching for a Part-aware ConvNet for Person Re-Identification," in *ICCV*, 2019.

[5] B. Zoph and Q. V. Le, "Neural Architecture Search with Reinforcement Learning," in *ICLR*, 2017.

[6] B. Zoph *et al.*, "Learning Transferable Architectures for Scalable Image Recognition," in *CVPR*, 2018, pp. 8697–8710.

[7] E. Real *et al.*, "Regularized Evolution for Image Classifier Architecture Search," in *AAAI*, vol. 33, 2019, pp. 4780–4789.

[8] H. Liu *et al.*, "Hierarchical representations for efficient architecture search," in *ICLR*, 2018.

[9] H. Pham *et al.*, "Efficient Neural Architecture Search via parameter Sharing," in *ICML*, vol. 9, 2018, pp. 6522–6531.

[10] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable Architecture Search," in *ICLR*, 2019.

[11] F. Wang *et al.*, "Residual attention network for image classification," in *CVPR*, 2017, pp. 6450–6458.

[12] J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks," in *CVPR*, 2018, pp. 7132–7141.

[13] J. Hu *et al.*, "Gather-excite: Exploiting feature context in convolutional neural networks," in *NeurIPS*, 2018, pp. 9401–9411.

[14] J. Park *et al.*, "BAM: Bottleneck Attention Module," in *BMVC*, 2018.

[15] S. Woo *et al.*, "CBAM: Convolutional block attention module," in *ECCV*, 2018.

[16] Y. Chen *et al.*, "A^2-Nets: Double Attention Networks," in *NeurIPS*, 2018, pp. 352–361.

[17] I. Bello *et al.*, "Attention Augmented Convolutional Networks," in *ICCV*, 2019.

[18] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," *University of Toronto*, 2012.

[19] A. Brock *et al.*, "SmaSH: One-shot model architecture search through hypernetworks," in *ICLR*, 2018.

[20] G. Bender *et al.*, "Understanding and simplifying one-shot architecture search," in *ICML*, vol. 2, 2018, pp. 883–893.

[21] X. Chen *et al.*, "Progressive Differentiable Architecture Search: Bridging the Depth Gap between Search and Evaluation," in *ICCV*, 2019.

[22] Y. Xu *et al.*, "PC-DARTS: Partial Channel Connections for Memory-Efficient Differentiable Architecture Search," in *ICLR*, 2020.

[23] H. Liang *et al.*, "DARTS+: Improved Differentiable Architecture Search with Early Stopping," 2019.

[24] X. Chu *et al.*, "Fair DARTS: Eliminating Unfair Advantages in Differentiable Architecture Search," *arXiv preprint arXiv:1911.12126*, 2019.

[25] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," in *ICLR*, 2017.

[26] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *ICLR*, 2015.

[27] T. DeVries and G. W. Taylor, "Improved Regularization of Convolutional Neural Networks with Cutout," 2017.

[28] N. Srivastava *et al.*, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[29] C. Szegedy *et al.*, "Going deeper with convolutions," in *CVPR*, 2015, pp. 1–9.

[30] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017.

[31] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *CVPR*, 2017, pp. 1800–1807.

[32] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *ICLR*, 2016.

[33] G. Huang *et al.*, "Densely connected convolutional networks," in *CVPR*, 2017, pp. 2261–2269.

[34] C. Liu *et al.*, "Progressive Neural Architecture Search," in *ECCV*, 2018.

[35] S. Xie *et al.*, "SNAS: Stochastic neural architecture search," in *ICLR*, 2019.

[36] H. Zhou *et al.*, "BayesNAS: A Bayesian Approach for Neural Architecture Search," in *ICML*, vol. 97, 2019, pp. 7603–7613.