Integrating Multiple Receptive Fields through Grouped Active Convolution

Yunho Jeon and Junmo Kim, Member, IEEE

Abstract—Convolutional networks have achieved great success in various vision tasks. This is mainly due to a considerable amount of research on network structure. In this study, instead of focusing on architectures, we focused on the convolution unit itself. The existing convolution unit has a fixed shape, and is limited to observing restricted receptive fields. In an earlier work, we proposed the active convolution unit (ACU), which can freely define its shape and learn by itself. In this paper, we propose a detailed analysis of the proposed unit and show that it is an efficient representation of a sparse weight convolution. Furthermore, we expand the unit to a grouped ACU, which can observe multiple receptive fields in one layer. We found that the performance of a naive grouped convolution is degraded by increasing the number of groups; however, the proposed unit retains the accuracy even though the number of parameters reduces. Based on this result, we suggest a depthwise ACU, and various experiments have shown that our unit is efficient and can replace the existing convolutions.

Index Terms—Convolutional neural network (CNN), Multiple Receptive Fields, depthwise convolution, deep learning.

1 INTRODUCTION

CONVOLUTIONAL neural network (CNN) has become a major topic of deep learning, especially in the visual recognition tasks. After the great success at the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) of 2012 [1], many efforts have been made to improve accuracy while reducing computational budgets by using CNN. The major focus for this research was on designing network architectures [2], [3], [4], [5], [6], [7]. Recently, attempts were made to automatically generate efficient network architectures [8], [9], and the generated networks achieved better result than the conventional networks. This approach is yet very slow and difficult to train by using feasible number of resources but will affect the designing of networks. In such studies, components are considered as more important factors than network construction.

Some other studies have focused on components to improve the performance of the network. Such methods suggest replacing existing units with new components while retaining the network architectures. In the early years of the development of this method, researchers focused mainly on activation units [10], [11], [12], [13], [14]. Variants of pooling were also proposed in many studies [15], [16], [17], [18]. Although the convolution unit is a core component of CNN, research on this unit has only been under way in recent years. Many variants of convolutions have been proposed to overcome weaknesses of the conventional convolution [19], [20], [21], [22], [23], [24]. As the shape of a convolution unit is fixed, it can only observe restricted receptive fields. Chen et al. [21] and Yu and Koltun [22] suggested a dilated convolution, which can expand the receptive field by retaining the parameters; this was applied for the dense prediction of segmentation. However, its dilation was fixed and set on initialization. In our previous paper [25], we solved this problem by introducing position parameters, and defined the unit as an active convolution unit (ACU). The proposed unit could define any shape of convolution, and the shape is learnable and can be optimized through backpropagation.

Generally, CNN expands its receptive areas by stacking convolution layers. Attempts have been made to view multiple receptive fields by configuring several units in parallel. Inception [3], [4], [5] is a module block using different sizes of convolutions. Pyramid pooling [17] and atrous pooling [26], [27] have also been suggested to aggregate features at different scales. Although the ACU could optimize its shape, it could possess only one shape for each layer because that shape is shared for all output channels. To overcome this weakness, we developed a grouped ACU inspired from grouped convolution [28], [29] and depthwise convolution [30], [31], [32]. A grouped ACU can have multiple shapes while decreasing computational complexity. To the best of our knowledge, this is the first component that is able to view different receptive fields in a layer not combining multiple units.

In this paper, we further consolidate the results of a
previous research [25] through further analysis and experiments. Moreover, we extended the ACU to a grouped ACU, which can view multiple receptive fields at one layer. Through this study, we show that the proposed unit is an efficient component for the formation of network topologies. The key contributions of this paper are summarized as follows:

- In this study, we performed additional analysis on ACU and showed that the ACU is an efficient representation of a sparse weight convolution. Further experiments explain why ACU can achieve better results with similar numbers of parameters.
- The grouped ACU is proposed, which is the combination of the original ACU and grouped convolution. While the original ACU uses only one shape and shares it for all channels, the grouped ACU uses multiple position sets in a layer. It is able to receive multiple receptive fields at one layer while reducing the computation complexity. Moreover, we show that the Inception module can be simplified through this unit.
- With the application of our proposed ACUs in various network architectures, we show that the units are simple, efficient, and can replace the existing convolutions.

In the next section, we review the related works on network architectures and the variants of convolution units. In section 3, we revisit ACU, which was proposed in [25], and provide a detailed analysis of the characteristics of the unit. In section 4, we propose grouped ACU, which divides the input and output channels according to a group and the shared position parameters in each group, respectively, and expand on the proposed unit to generalize the Inception model. In section 5, experimental results are demonstrated to show the effectiveness of the proposed method.

## 2 Related Work

Our approach is based on the success of CNN for image classifications. The methodology of such a classification has spread to various other applications including semantic segmentation [21], [26], [33] and object detection [34], [35], [36], [37].

### CNN Architectures

Most research on CNN has focused on developing an architecture to achieve a better result. AlexNet [38] uses various types of convolutions and stacks some pooling layers to reduce spatial dimensions. VGG [2] is based on the idea that a stack of two $3 \times 3$ convolution layers is more effective than $5 \times 5$ layers. This network is used broadly for many applications owing to the simplicity of the topology. GoogleNet [3], [4], [5] introduced an Inception layer for the composition of various receptive fields. This network showed that a carefully crafted design can achieve better result while maintaining a constant computational budget. The residual network [6], [7], [39] solves the gradient vanishing problem by adding shortcut connections to implement identity mapping, allowing for deeper networks to be configured. Later, many variant of a residual network were proposed [40], [41], [42].

### Variants of Convolution

The dilated convolution [21], [22] was suggested to enhance the resolution of the result and reduce postprocessing in semantic segmentation tasks. In our previous work [25], we introduced the ACU, which is a generalization of the naive convolution. By introducing position parameters, any shape of a convolution can be defined, and the shape can be learned through formal backpropagation.

Some studies have attempted to use dynamic weights instead of fixed weights. Dynamic filter network [23] convolves the weight received from other networks. Deformable convolution [24] is another type of dynamic network, which receives the position parameter from other networks, and the convolution shape is changed dynamically. The concept of deformable convolution is similar to our ACU. The major difference is that ACU possesses intrinsic position parameters, while a deformable convolution receives the position parameters from other networks, and thus needs another network to generate them.

### Combination of Multiscale Features

Beyond extending receptive areas, there have been many attempts to combine multiple fields. Spatial pyramid pooling [17] was suggested for integrating receptive fields at different scales. GoogleNet [3], [4], [5] formed the Inception module composed of multiple sized convolutions; this can create better features by using less number of parameters. In the segmentation tasks, DeepLab [26], [27] used the atrous spatial-pyramid-pooling layer, which uses multiple filters with different dilation rates. All of this research comprised multiple operations, and no unified component was built that allows the viewing of multiple receptive fields at different scales.

### Grouped Convolution

Grouped convolution divides its input and output into small groups, and calculates the outputs by using only input channels within the same group. AlexNet [38] first introduced the grouped convolution not for the improving accuracy but for other practical reasons, i.e., for running a large network in restricted resources, convolution was divided into two groups. In ShuffleNet [29], a grouped convolution is used for reducing the computational cost while maintaining accuracy with limited resources on the embedded devices. ResNext [28] shows that increasing the number of groups is more efficient than expanding depth or width for gaining accuracy.

### Depthwise Convolution

Depthwise (or channelwise) convolution is a special case of grouped convolution, in which the number of input and output channels is the same as the number of groups. This implies that each output channel is calculated using only a corresponding input channel. Xception [30] uses separable convolution, which first applies the depthwise convolution, followed by the pointwise convolution. This operation reduces the number of weight parameters efficiently, and the corresponding network achieves better result with less parameters. MobileNet [31], [32] also employs the depthwise convolution to reduce the network size and run fast in embedded devices.

## 3 Active Convolution

In this section, we will revise the basic concept of the ACU proposed in an earlier work [25]. Naive convolution unit has
learnable weights \( W \), and each weight is convolved with its input (Eq. (1)). This can be written for one output value \( y_{m,n} \) in given location \((m,n)\), as shown in Eq. (2). We omitted a bias term and assumed one output channel for simplicity.

\[
Y = W * X
\]

(1)

\[
y_{m,n} = \sum_c \sum_{i,j} w_{c,i,j} \cdot x_{c,m+i,n+j}
\]

(2)

where \( c \) is the index of the input channel, and \( w_{c,i,j} \) and \( x_{c,m,n} \) are the weight and input value in the given channel \( c \) and position \((m,n)\), respectively. Index \( i, j \) defines the fixed convolution area (e.g. \( i, j = \{-1, 0, 1\} \) for \( 3 \times 3 \) convolution).

Unlike naive convolution, ACU has the additional position parameter \( \theta_p \), which defines the horizontal and vertical displacement \((\alpha_k, \beta_k)\) of the input from the center of the filter (Eq. (3)). We denote one acceptor of the ACU as synapse, and each synapse has its own weight and position.

\[
\theta_p = \{ p_k | 0 \leq k < K \},
\]

\[
p_k = (\alpha_k, \beta_k) \in \mathbb{R}^2
\]

(3)

where \( k \) is the index of the synapse. By using this position parameter, the ACU is defined as

\[
Y = W * X_{\theta_p}
\]

(4)

\[
y_{m,n} = \sum_c \sum_k w_{c,k} \cdot x_{c,m,n}
\]

(5)

\[
= \sum_c \sum_k w_{c,k} \cdot x_{c,m+\alpha_k,n+\beta_k}
\]

where \( x_{c,m,n} \) is the input value located at displacement \( p_k \) from origin \((m,n)\) at channel \( c \).

\( \alpha_k \) and \( \beta_k \) are not limited to integers but are expandable to real numbers. When these values are real, the location of input is placed in an inter-lattice point. The value of this point can be calculated through interpolation. By using bilinear interpolation, the output value is differentiable according to \( \alpha_k \) and \( \beta_k \), and these parameters are learnable through backpropagation. Backward calculation for the weight and bias is the same as that for the naive convolution, and the derivatives for the position parameter can be calculated easily. Please refer to [25] for more details.

### 3.1 What is an ACU?

In a previous study [25], we defined a new form of convolution by adding position parameters and showed that this new unit improves classification accuracy. Thus, the following questions must be answered: is ACU a new operation that is completely different from convolution? What characteristics of the ACU improve the performance of the network? In this section, we answer these questions and show that ACU is actually the efficient representation of a sparse weight convolution.

**Lemma 1.** The set containing all operations represented by convolution is the subset of the set containing all operations represented by ACU, i.e., \( C \subset A \)

where \( A \) is the set containing all operations represented by ACU, and \( C \) is the set containing all operations represented by convolution. Clearly, any convolution can be represented by an ACU. Given a convolution, we can convert it to an ACU by using the same weight, and assign position parameters based on the given shape. For instance, the conventional \( 3 \times 3 \) convolution is represented by the ACU with \( \theta_p = \{(−1,−1), (0,−1), (1,−1), (−1,0), (0,0), (1,0), (−1,1), (0,1), (1,1)\} \).

**Theorem 2.** Given position parameter \( \theta_p \) and weight \( W \), there exists an extrapolated weight \( \overline{W}_{\theta_p} \) which holds for

\[
W * X_{\theta_p} = \overline{W}_{\theta_p} * X
\]

The detailed proof of Theorem 2 is in Appendix A, and here we provide conceptual explanations. We first contemplated an ACU with one synapse. Fig. 2(a) shows the original calculation using interpolating inputs. Interpolated input \( x_{c,m,n}^{p_k} \) is derived using four neighbor input values, and weight \( w \) of the given synapse is multiplied with the interpolated input. In Fig. 2(b), instead of interpolating inputs, the weight is extrapolated to the nearest four neighbors. Each extrapolated weight \( w_{a,b}^{p_k} \) is multiplied with the original input value, and the summation of these four values provides the same result as that obtained through the previous calculation. This procedure can be regarded as a \( 2 \times 2 \) convolution.

---

**Fig. 2.** Comparison of two methods for calculating ACU. (a) Weights are multiplied with interpolated inputs. (b) Extrapolated weights are multiplied with original inputs, and summed up. These two methods obtain the same result.
This procedure can be generalized to an ACU with multiple synapses (Fig. 3). Each weight of a synapse extrapolated to the nearest points, and the extrapolated weights at the same point are summed up. This leads to the construction of one large convolution weight \( \mathbf{W}_{\theta_p} \), and the original calculation was noted to be the same as that of a conventional convolution using this weight. According to this result, any ACU can be converted to a naive convolution, thus leading to the derivation of Lemma 3.

**Lemma 3.** The set containing all operations represented by ACU is the subset of the set containing all operations represented by convolution, i.e., \( A \subset C \).

**Theorem 4.** Based on Lemma 1 and Lemma 3, the set containing all operations represented by ACU is equal to the set containing all operations represented by convolution, i.e., \( C = A \).

This shows that ACU has the same operational span as a convolution, only but has a different representation. Generally, extrapolated weight is sparse, and sparsity depends on the position parameters and number of synapses. In a naive convolution, the size should be large even though the weight is sparse, thus requiring more parameters. However, ACU reduces the number of total parameters by using a position set. Therefore, we can say that ACU is the efficient representation of a sparse weight convolution, and we can expand the size of the receptive field infinitely without exploding the weight parameters.

### 3.2 Discussion about ACU

In [25], we showed that changing the naive convolution to an ACU improves network accuracy, while retaining the similar number of parameters. In this study, we conducted further experiments on the CIFAR-10 dataset [43] to understand and analyze the effect of ACU. Base network is a 29-layer deep residual network using pre-activation bottleneck blocks [7]. The template for residual block is

\[
\begin{bmatrix}
1 \times 1, 32 \\
3 \times 3, 32 \\
1 \times 1, 128
\end{bmatrix}
\]

which comprises three stages, and a doubling width after each stage. The networks are trained for 64k iterations, and all results are an average of five runs. Please refer to Appendix B for details. The base network acquired 6.16% test error with 1.24M parameters.

#### 3.2.1 Parameter Efficiency

To verify the effect of ACU, we changed 3×3 convolutions in residual blocks to an ACU, and the total number of parameter were almost the same. For training the ACU, we used normalized gradient for position parameters and conducted warming-up of 10k iterations, following our previous work. This simple change could reduce 0.6% error (Table 1).

Fig 4 shows the learned positions of ACUs, and the displacement of all synapses are bounded below three. Therefore, as we proved earlier, this network is the subset of the network using 7×7 convolutions. To compare this with the ACU network, we changed 3×3 convolutions to 7×7. Although this network has more than three times the parameters of the ACU network, the performance is similar. This result shows that the ACU can expand the network capacity with a small number of parameters and optimize the network better than a large convolution. Therefore, we can say that an ACU is an efficient representation of sparse weight convolution.

#### 3.2.2 Factor of Improvement

To deeply understand the reason of the improvement brought about by the ACU, we conducted additional experiments, as shown in Table 2. Fig 4 shows that high layers tend to enlarge the fields of reception, and this is in agreement with the results in our previous work. Therefore, we might believe that the improvement is brought about by the enlargement of the receptive fields in higher layers, and we changed 3×3 convolution in the last stage to a

### Table 1

| Network   | Test Error (%) | Params  |
|-----------|----------------|---------|
| 3×3 Conv  | 6.16 ± 0.15    | 1.24M   |
| 3×3 ACU   | 5.56 ± 0.13    | 1.24M   |
| 7×7 Conv  | 5.49 ± 0.12    | 3.82M   |

Fig. 4. Example of learned position of ACUs. Higher layers tend to grow its receptive field. Sizes of all ACUs are bounded in the 7×7 area.
dilated convolution \cite{21, 22} with dilation 2 to enlarge its receptive field while retaining the same parameters (Base-D2). However, this result is even poorer than that of the base network, showing that the enlargement of the receptive field, by itself, is not helpful for the network.

If, as claimed earlier, the ACU is able to learn its shape through training, then, we must determine whether the final shape of the convolution is really the effectively optimized shape for the application. To answer this question, we initialized the shape of ACUs according to the final shape of the convolution of one sample of the ACU network, and fixed the shape (ACU-Fixed). Weights were randomly initialized for different runs, and trained from their initial state. The result obtained was even further improved from that of the ACU network; thus, we can say that the shape of the convolution learned by the ACU was optimized for the given application. We believe that this is the main factor of the improvement using ACU.

### 4 Grouped Active Convolution

In our previous work \cite{23}, we shared the position set for all output channels in a layer, with each layer having only one shape. To view multiple receptive fields for ACU, we might apply different shapes to some output channels while retaining the connections between inputs and outputs. In this approach, every input channel should be interpolated for all other shapes, whereas the interpolation for the input is required only once in the original ACU. This leads to numerous calculations, thus slowing down the process. Moreover, no significant improvement could be found to offset this overhead. Thus, the practical application of multiple shapes for an ACU has been difficult.

ResNeXt \cite{28} showed that a convolution with a group is more efficient than a fully connected convolution in residual networks. A grouped convolution divides input channels according to groups, and calculates the corresponding output by only using the input channels within a same group. This reduces lots of weights for a convolution. Recently, many studies \cite{29, 30, 31, 32} used grouped convolution for improving the accuracy and reducing the number of parameters. Xie et al. \cite{28} denoted the number of groups as a cardinality; we used the same terminology in this study.

Motivated by the grouped convolution, we propose the grouped ACU, which enables to use multiple position parameter sets without increasing computation complexity. The grouped ACU shares position parameters within a same group and uses different parameter sets for different groups; therefore, the number of position sets is the same as the cardinality. Eq. (3) can be extended to Eq. (6) in a grouped ACU.

$$\theta_p = \{\theta^g_p \mid 1 \leq g \leq G\},$$

$$\theta^g_p = \{p^g_k \mid 0 \leq k < K\},$$

$$p^g_k = (\alpha_k^g, \beta_k^g) \in \mathbb{R}^2$$

where G is the cardinality and K is the number of synapses, respectively. In addition, $\theta^g_p$ is the set of position parameters of each group g.

Figure 5(b) shows the concept of the grouped ACU. Compared to the original ACU, different position sets are applied to different groups in the grouped ACU. As this grouped ACU divides input channels according to groups, each input is interpolated only once regardless of the cardinality; therefore, the computation complexity for interpolation is the same as that of the original ACU. The total computation complexity is decreased owing to the reduction in the linkages from input to output channels. When considering the grouped ACU, convolutions of each group possess their individual shapes, and multiple receptive fields can be observed in a layer. The ACU we proposed in earlier work can be regarded as a grouped ACU with only one group.

The number of parameters for weight is the same as that for naive convolution. The additional parameters for a grouped ACU are $2 \times (K - 1) \times G$; each synapse has two parameters for defining positions, and parameters are not required at the origin because its position is fixed. As a result, the total number of parameters for the grouped ACU, except bias, include

$$\{C_I \times C_O \times K/G\} + \{2 \times (K - 1) \times G\}$$

where $C_I$ and $C_O$ are the numbers of input and output channels, respectively. Generally, as the width of channels is greater than the number of synapses, if cardinality $G$ is small, the first term is considerably larger than the second term. With the increase in cardinality, the second term also increases but the total number of parameters decreases.
4.1 Generalization of Inception

The Inception module [2] is composed of multiple types of convolutions and poolings, and achieves good accuracy while maintaining a constant computational budget. The key feature of this success is based on the composition of multiple receptive-field convolutions. However, it is difficult to design such an Inception structure because its components and parameters are carefully decided manually by considering questions such as how many different types of convolutions should be used, what type of convolution should be used, and how many channels must be applied for convolutions.

Chollet [30] simplified Inception by using $3 \times 3$ convolutions (Fig. 6(a)). However, this simplification is not enough to generalize an Inception module because it has only one receptive field. The important characteristic of Inception is the use of multiple receptive fields. If we change $3 \times 3$ convolutions to an ACU (Fig. 6(b)), this structure can receive multiple receptive fields, as each ACU is able to learn its own shape through training. Therefore, we claim that this structure can be a more general form of the Inception module. Fig. 6(b) is equivalently represented by a grouped ACU following a $1 \times 1$ convolution (Fig. 6(c)). This generalization solves the design difficulties by simplifying the Inception, while still taking advantage of it; only the width and cardinality of a grouped ACU should be selected, and an ACU will learn multiple receptive fields through training.

By further increasing the cardinality, we reached an extreme case, in which only one input channel is used for producing output. This kind of convolution is a special case of the grouped convolution and is called depthwise convolution. Fig. 6(d) shows the depthwise ACU, which applies the ACU to the depthwise convolution. In this unit, each output channel has a unique shape, and it accepts as many receptive fields as the number of channels available.

4.2 Effectiveness of Grouped ACU

To prove the effectiveness of the grouped ACU, we conducted experiments (Table 3). The base network is the same as that defined in section 3.2. This network can be converted to the ResNeXt [28] (16×4d) model, in which cardinality is 16, and the basic residual block is

$$\begin{bmatrix}
1 \times 1, 64 \\
3 \times 3, 64, & G = 16 \\
1 \times 1, 128
\end{bmatrix}.$$  

This network achieved less test error compared to the base network with similar number of parameters. By using this 16×4d model, we simply changed the grouped convolutions to grouped ACUs. As the cardinality is 16, ACUs use 16 position sets. This network achieved 4.93% error rate, which shows a 0.45% improvement compared to the naive convolution.

To examine the effect of using more multiple receptive fields, we increased cardinality. ResNeXt maintains the cardinality regardless of stage, and thus the number of input channels in a group is doubled after a stage. We changed this model slightly such that the new model retains the number of input channels per group instead of cardinality for all stages. In this modification, the cardinality is doubled after every stage. $\times 4d$ in Table 3 denotes four input channels per group. The cardinality of the $\times 4d$ network is the same as that of $16 \times 4d$ network for the first stage; however, at the second stage, the cardinality reaches 32. This reduces the total number of parameters because the linkage from input to output channels becomes sparser at the second and third stages.

We tested the number of input channels per group to 4, 2, and 1 (Table 3). As the cardinality increases, the number of parameters decreases. Fig. 7 summarizes the experimental results, showing an interesting phenomenon. That is, if we use naive convolution, the accuracy is decreased with the decrease in the input dimension for one group. This shows that an increase in the cardinality is not always helpful, and a grouped convolution possesses an optimal cardinality. The reduction in the input dimension implies that the available feature map for convolution is reduced, and naive convolution thus forms restricted combinations with a small number of input channels. This is why increasing cardinality decreases accuracy.

In contrast, in a grouped ACU, an increasing cardinality does not degrade the accuracy. In Fig. 7, the error rate remains almost the same while the input dimension reduces.
Naive convolution has a fixed shape, because of which the combination of weight is restricted. However, the corresponding ACU could expand its receptive field, and the combination is not restricted even though the input dimension is reduced. Fig. 8 shows an example of the ACU-learned shape of a convolution in a layer. The figure also shows diversity of receptive fields. In addition, Fig. 12 shows the visualization of filters (the filter of 3×3 convolution in the last residual block of 1×1d network in Table. 3). While we can see many duplications in naive convolution, the filter of ACU is more diverse and wide.

Actually, the degree of freedom for one filter in a grouped ACU is larger than that in a naive convolution owing to the additional position parameters. Therefore, it is usual to suspect that the characteristics, which maintain performance with low input dimensions, originate from large number of free parameters. To complete our analysis, we conducted further experiments by changing 3×3 convolutions to 5×5 convolutions, which have more or the same number of parameters than the corresponding ACU network. We observed the same tendency, i.e., with the decreasing number of input channels in a group, the accuracy is also decreased. A depthwise 5×5 convolution has exactly the same number of parameters as that in a depthwise ACU; however, the accuracy of this network is poorer than that of ACU.

Therefore, the benefit of using an ACU to develop useful features by only using a small number of inputs does not originate from the number of parameters but from its ability to change its shape. Considering that 1×1d has a small number of parameters than 4d, 1d (depthwise) is a good choice for the grouped ACU; thus, we used the depthwise ACU for the remainder of this study.

### 4.3 Discussion

#### 4.3.1 Effect of Warming up

In [25], we suggested the strategy of “warming-up,” which freezes the position parameters for some iterations to stabilize the initial weight; this showed an additional performance boost. However, in the depthwise ACU, warming-up is not needed, and is rather harmful. Table 4 shows the results of test accuracy with and without warming-up. When we use one group of ACU, the stability of its shape is important because the position parameter is shared among all channels. Therefore, the early movement of position with unstable weight is not good for the training, and warming-up of the weight is a beneficial option to reduce this effect.

However, in a depthwise ACU, each input-output channel pair does not share position parameters but uses its own set, and the position is optimized according to the initial weight. Thus, the early movement of position is not an unstable behavior. In contrast, warming-up prevents the movement of the position and restricts diversity in shapes. In this analysis, we did not use warming-up for depthwise ACUs.

---

### Table 3

Comparison of test error (%) with the increase in cardinality. The results are summarized in Fig. 7.

| Network | Conv3×3 Test Error(%) | Params(M) | Conv5×5 Test Error(%) | Params(M) | ACU Test Error(%) | Params(M) |
|---------|-----------------------|-----------|-----------------------|-----------|-------------------|-----------|
| 16×4d   | 5.38 ± 0.06           | 1.28      | 4.78 ± 0.1            | 1.54      | 4.93 ± 0.14       | 1.28      |
| ×4d     | 5.43 ± 0.13           | 1.18      | 4.96 ± 0.14           | 1.27      | 4.87 ± 0.19       | 1.19      |
| ×2d     | 5.59 ± 0.06           | 1.16      | 5.06 ± 0.13           | 1.2       | 4.83 ± 0.16       | 1.17      |
| ×1d     | 5.75 ± 0.17           | 1.15      | 5.18 ± 0.19           | 1.17      | 4.9 ± 0.08        | 1.17      |

---

Fig. 7. Test accuracy of networks in Table 3. By increasing cardinality, the accuracy of the network using naive convolution is degraded. However, the network using an ACU almost retains the accuracy even though the parameter size decreases.

Fig. 8. Example of learned position in a layer (ACU in the last residual block of 1×1d network in Table 3). This shows diversity of learned positions.
4.3.2 Effectiveness of Using Multiple Positions

A depthwise ACU uses same groups for convolution and position parameters. However, conceptually, we can separate the notion of the group for computing a convolution and sharing positions. For example, it is possible to use four groups for a convolution but two groups for positions: groups 0 and 1 share the same position parameters, while groups 1 and 2 use another set.

If a grouped ACU is applied, the improvement can be derived from the ACU itself or by using multiple positions of the ACU. To clarify the factor for the improvement, we examined another type of ACU, which operates similar to the depthwise ACU but uses only one set of parameters and shares them for all outputs. This implies that a layer has one receptive field regardless of the cardinality.

Even with the shared receptive field (×1d/ACU/Share in Table 5), a better result was achieved than that for the base network. This shows that the ACU is effective even though using only one set of position parameters. If we use multiple positions (∗1d/ACU/Multi), we can obtain additional improvement; this supports the effectiveness of using multiple position parameters.

4.3.3 Retraining with Trained Shapes

In section 3.2.2 we achieved better results by retraining the network using trained position parameters on naive ACU than training it from scratch. This strategy could also be thought to be applied to a depthwise ACU network; however, this is false. An ACU with one group shares position parameters for all output channels; thus, the optimized shape is generalized for that layer. Therefore, the trained shape can also be effective for re-initializing weight. However, in a depthwise ACU, each position set is specialized only for a particular weight. If we reassign the weights with an optimized shape, the weight would not match with the trained shape. As a result, obtaining a better accuracy than that of the original network is difficult. Nevertheless, we conducted experiments, and as expected, obtained worse results.

### Table 4

| Network          | w/o warming-up | w /warming-up | improvement |
|------------------|----------------|---------------|-------------|
| ResNet/ACU       | 5.64           | 5.56          | +0.08       |
| ×1d/ACU          | 4.9            | 5.04          | -0.14       |

### Table 5

| Network          | CIFAR10(%) | params  |
|------------------|------------|---------|
| ×1d              | 5.75 ± 0.17 | 1.15M   |
| ×1d/ACU/Share    | 5.2 ± 0.04  | 1.15M   |
| ×1d/ACU/Multi    | 4.9 ± 0.08  | 1.17M   |

5 EXPERIMENT

5.1 Experiment on ImageNet

We conducted experiments on the ImageNet classification task [1] on several networks to show the effectiveness of the proposed unit. To observe the effect of ACU itself, we retained all other parameters, and did not apply intensive image augmentation. We randomly cropped 224×224 from 256×256 images based on the method in [38], and flipped the images horizontally. The inputs were normalized, and no more augmentations were applied. To reduce the training time, we used linear decay learning-rate schedule with 60 epochs. The initial learning rate was 0.1. An extensive search was not conducted for finding the optimal parameters for the given networks. All the results were derived using a single network and performing single crop testing on a validation set, with an average of three runs.

5.1.1 Replacing Inception Modules

In section 3.1 we claim that a grouped ACU can generalize Inception modules, and we conducted the experiment to support this idea. We started with an Inception network [4] and applied BN [44] before all ReLUs to speed up the training. This is the base network and we achieved 72.0% Top-1 accuracy (Table 6). Then, from the base network, we simply changed all of the Inception modules to a (1×1 conv)-(3×3 depthwise ACU) block. When we retained the width of the Inception, this conversion reduced the number of parameters. To compare the performance by using a similar parameter size, we expanded the width of Inception by a factor of 1.4.

This change reduces the computational complexity in terms of the total number of Multiply–Adds (MAdds), due to the simplification of Inception modules. Further, this network achieved slightly better result compared to that using the original method. Fig. 9 shows the samples of learned position on each Inception block. Five channels were selected randomly and their position sets are represented by different colors. Diverse shapes of convolutions can be observed; therefore, multiple receptive fields can be applied in a layer like Inception module. This new block is simple and is not needed to decide which type of convolution or pooling should be used or how many channels must be assigned for each operation. This simplification can help develop effective architectures by reducing complexity of design choices.

5.1.2 Xception with Depthwise ACUs

Xception [30] is considered as an extreme version of the Inception module using depthwise separable convolutions.
As stated previously, the replacement of Inception with only $3 \times 3$ convolutions is not enough for generalization because such a replacement would not be able to show multiple receptive fields, which is an important characteristic of an Inception block.

For verifying that the use of a depthwise ACU is effective than the use of a naive convolution, we changed the depthwise convolutions to ACUs. Because we regarded a residual block as one Inception module, we only changed the last depthwise convolution to a depthwise ACU in each residual block. The last depthwise ACU summarizes features in a block by observing multiple receptive fields. This modification resulted in the addition of small amounts of parameters, and we achieved better result than the base network (Table 6). Fig. 10 illustrates the 2D histogram of synapse positions of ACU in each residual block. The distribution of positions varies according to the depth of a layer. Compared to naive depthwise convolutions, which can view only a fixed area, our unit can view more diverse receptive fields, and this result shows that our generalization is more effective.

5.1.3 Mobile application of Depthwise ACU

Depthwise convolution reduces not only the number of parameters but also the computational complexity. Accordingly, attempts have been made to apply it to mobile applications. MobileNet \cite{31, 32} is an up-to-date network which runs fast in mobile devices that maintaining a high accuracy. This network uses depthwise convolutions to achieve good accuracy with feasible running time.

Although the calculation of depthwise ACU is slightly complicated than that of a naive convolution because of the presence of interpolations, the calculation of a depthwise operation utilizes a small portion of the total network computation; the fully connected $1 \times 1$ convolution utilizes considerable amounts of computations over the entire network. Therefore, ACU can be applied for mobile applications; we examined this potential through experiments.

The base network we used in this study was the MobileNet v2 \cite{32}, which consists of repetitions of residual blocks with depthwise convolution. We changed all depthwise convolutions in the residual blocks, as mentioned in the previous section. Table 7 shows the experimental result. We performed tests by varying the width multiplier from 0.8 to 1.4; this controls the network width. Fig. 11 summarizes the results and shows that the network with ACU consistently achieved better accuracy with the same number of parameters.

To compare the performance of a network in terms of computational complexity, we calculated MAdds. As discussed earlier, the depthwise operation occupies a small portion of the total calculations. Even though ACU is computationally more complex than naive convolution owing to the interpolations, its overhead is not as much as that of the total computations. In addition, the ratio of overheads is reduced with the increasing network width. For example, MobileNet v2 ($\times 1.4$) and ACU ($\times 1.2$) achieved almost the same accuracies; however, the network with ACU has fewer parameters and complexity. This result shows the possibility that the ACU can also be used for mobile applications.
TABLE 7
Comparison of the Mobile network with ACU for ImageNet dataset. This is the validation error (%) evaluated using single-crop image of single model. The results are summarized in Fig. [1]. The networks using ACU achieve better result with fewer parameters and complexity.

| Network     | Top-1   | Top-5   | params | MAdds  |
|-------------|---------|---------|--------|--------|
| MobileNet v2(x=0.8) | 69.2    | 88.7    | 2.5M   | 201M   |
| MobileNet v2(x=1)  | 71.3    | 90.2    | 3.5M   | 314M   |
| MobileNet v2(x=2.12) | 72.4    | 90.8    | 4.7M   | 437M   |
| MobileNet v2(x=14) | 73.3    | 91.4    | 6.1M   | 586M   |
| ACU(×0.8)    | 70.2    | 89.3    | 2.5M   | 244M   |
| ACU(×1)      | 72.1    | 90.8    | 3.6M   | 369M   |
| ACU(×1.2)    | 73.3    | 91.4    | 4.8M   | 502M   |
| ACU(×1.4)    | 74.0    | 91.8    | 6.2M   | 663M   |

TABLE 8
Test error of CIFAR-10. Compared to ResNeXt-29, 8×64d model, the use of ACU achieved better result with a smaller number of parameters.

| Network          | Test Error(%) | params |
|------------------|---------------|--------|
| ResNeXt-29, 8×64d| 3.9 ± 0.12    | 34.4M  |
| Depthwise ACU-29 | 3.9 ± 0.09    | 16.13M |
| Depthwise ACU-47 | 3.65 ± 0.06   | 27.37M |

5.2 Experiment on CIFAR-10

We performed more experiments on CIFAR-10 [43] datasets. We initiated with the ResNeXt-29 architecture, which is 29-layer deep; the width of bottleneck is 512 and the base cardinality is eight. We reproduced this network but did not achieve the result reported in the previously mentioned studies. This is mainly due to the difference in the framework: we used Caffe [45] not Torch [46] used in the aforementioned studies. In addition, we constructed the residual network in a pre-activation style [7].

In the ResNeXt-29 network, we changed 3×3 grouped convolutions to depthwise ACUs (ACU-29). This reduces many network parameters (16.13M); however, the accuracy remains almost the same as that of the base network, as described in section 4.2. To increase the accuracy with an even deeper network, we stacked five blocks on each stage (ACU-47, 47 layer deep). This network uses 27.37M parameters, which are still less than those in ResNeXt-29, with an error rate of 3.63%. Although the reproduced result of the original network was not achieved, a similar result was achieved with small numbers of parameters.

6 Conclusion

In this paper, we revised the active convolution, which we proposed previously [25], and showed that ACU is an efficient representation of sparse weight convolution. By using an ACU, we can achieve a better result with small number of parameters than that used in naive convolutions. Further experiments have demonstrated that the improvement of accuracy with the use of ACU is due to the ability of training the optimal shape of convolutions.

Beyond ACU, we proposed the grouped ACU, which can use a number of position sets instead of one shared position. By applying groups of positions, multiple receptive fields can be observed in a layer. We also showed that the 1×1 convolution followed by the grouped ACU is a generalization of the Inception module. Furthermore, by increasing cardinalities, we observed that the network with an ACU retains a similar accuracy even while the number of parameters decreases. As a result, we suggest that a depthwise ACU is a good option because it greatly reduces computation cost as well as the number of parameters. Our experiments showed the effectiveness of the depthwise ACU for various benchmarks and networks, and it is considered an attractive unit for replacing naive convolutions.

APPENDIX A

Proof of Theorem 2

In Eq. (5), input \( x_{c,m+a_k,n+b_k} \) can be calculated through bilinear interpolation:

\[
x_{c,m+a_k,n+b_k} = Q_{c,k}^{11} \cdot (1 - \Delta a_k) \cdot (1 - \Delta b_k) + Q_{c,k}^{12} \cdot \Delta a_k \cdot (1 - \Delta b_k)
+ Q_{c,k}^{21} \cdot (1 - \Delta a_k) \cdot \Delta b_k + Q_{c,k}^{22} \cdot \Delta a_k \cdot \Delta b_k
\]  (8)

where \( Q_{c,k}^{ab} \) represents the four nearest integer points (Fig. 1):

\[
Q_{c,k}^{11} = x_{c,m_k^1,n_k^1}, Q_{c,k}^{12} = x_{c,m_k^1,n_k^2}, Q_{c,k}^{21} = x_{c,m_k^2,n_k^1}, Q_{c,k}^{22} = x_{c,m_k^2,n_k^2},
\]  (9)

where

\[
\Delta a_k = a_k - |a_k|, \quad \Delta b_k = b_k - |b_k|,
\]  (10)

\[
m_k^1 = m + |a_k|, \quad n_k^1 = m_k^1 + 1, \quad n_k^2 = n + |b_k|, \quad n_k^2 = n_k^2 + 1
\]  (11)

By using Eq. (8), Eq. (5) is converted to

\[
y_{m,n} = \sum_c \sum_k w_{c,k} \cdot (Q_{c,k}^{11} \cdot (1 - \Delta a_k) \cdot (1 - \Delta b_k) + Q_{c,k}^{12} \cdot \Delta a_k \cdot (1 - \Delta b_k) + Q_{c,k}^{21} \cdot (1 - \Delta a_k) \cdot \Delta b_k + Q_{c,k}^{22} \cdot \Delta a_k \cdot \Delta b_k)
\]  (12)

By substituting Eq. (12) with extrapolated weight \( w_{11}^{11}, w_{21}^{12}, w_{22}^{12} \) and \( w_{12}^{12} \), we get

\[
w_{11}^{11} = (1 - \Delta a_k) \cdot (1 - \Delta b_k) \cdot w_{c,k}
\]  (13)

\[
w_{21}^{12} = \Delta a_k \cdot (1 - \Delta b_k) \cdot w_{c,k}
\]  (13)

\[
w_{12}^{12} = (1 - \Delta a_k) \cdot \Delta b_k \cdot w_{c,k}
\]  (13)

\[
w_{22}^{12} = \Delta a_k \cdot \Delta b_k \cdot w_{c,k}
\]  (13)
Therefore, we can develop extrapolated weight \( W_{\theta_p} \), according to which Eq. (16) holds by using \( \overline{w}_{c,i,j} \).

\[
W \ast X_{\theta_p} = W_{\theta_p} \ast X
\]  

(16)

**APPENDIX B**

**Experiment Details**

We trained the network by using 50k images, and tested it by using 10k images. The inputs were normalized and randomly cropped to 32\( \times \)32 images, padded by four pixels on each side and flipped horizontally according to the method in [6], [28]. The weight parameters were initialized using the method proposed by He et al. [13]. The initial learning rate was 0.1, and was divided by 10 after 32k and 48k iterations. The networks were trained with 64k iterations by using stochastic gradient descent with the Nesterov momentum. The batch size was 128, and the momentum was 0.9. We used L2 regularization, and the weight decay was 5e-4. For training ACU, we initialized the positions of the synapses with the shape of a conventional convolution and used a normalized gradient with the learning rate of 1e-3 as a position parameter.

The networks consisted of three stages, and each stage has three residual blocks (Table 9). After each stage, the width was doubled, and a stride 2 to 3\( \times \)3 convolution was applied at the first block of each stage. We used a pre-activation style residual network [2] with a projection shortcut.

**REFERENCES**

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “Imagenet large scale visual recognition challenge,” International Journal of Computer Vision (IJCV), vol. 115, no. 3, pp. 211–252, Dec 2015.

[2] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” in International Conference on Learning Representations (ICLR), 2015, pp. 1–14.
Fig. 12. Visualization of the filter of the $3 \times 3$ convolution in the last residual block of $\times 1 d$ network in Table 3 (a) Owing to the restriction of its shape, many duplications of filters are shown in naive convolution. (b) To visualize the filter of ACU, we extrapolated filters and cropped a $9 \times 9$ area. The filters are shown as diverse shapes, and display many white areas indicating sparsity. Note that blue and red represent negative and positive values, respectively.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in neural information processing systems (NIPS), 2012, pp. 1097–1105.

[39] S. Zagoruyko and N. Komodakis, “Wide residual networks,” in Proceedings of the British Machine Vision Conference (BMVC), E. R. H. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[40] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.

[41] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng, “Dual path networks,” in Advances in neural information processing systems (NIPS), 2017, pp. 4470–4478.

[42] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” arXiv preprint arXiv:1704.04861, 2017.

[43] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3431–3440.

[44] S. Zagoruyko and N. Komodakis, “Wide residual networks,” in Proceedings of The 32nd International Conference on Machine Learning (ICML), 2015, pp. 448–456.

[45] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions,” in International Conference on Learning Representations (ICLR), 2016.

[46] X. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, “Deformable convolutional networks,” in Proceedings of the IEEE International Conference on Computer Vision (ICCV), Oct 2017, pp. 764–773.

[47] D. Han, J. Kim, and J. Kim, “Deep pyramidal residual networks,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 4961–4970.

[48] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788.

[49] S. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in European Conference on Computer Vision (ECCV). Springer, 2016, pp. 21–37.

[50] W. Liu, Y. Chen, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 5987–5995.

[51] Y. Zhou, Q. Ye, Q. Qiu, and J. Jiao, “Oriented response networks,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.
[45] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” in Proceedings of the 22nd ACM international conference on Multimedia. ACM, 2014, pp. 675–678.

[46] R. Collobert, K. Kavukcuoglu, and C. Farabet, “Torch7: A matlab-like environment for machine learning,” in NIPS Workshop, 2011.