ECA-Net: Efficient Channel Attention for Deep Convolutional Neural Networks

Qilong Wang[1], Banggu Wu[1], Pengfei Zhu[1], Peihua Li[2], Wangmeng Zuo[3], Qinghua Hu[1,*]

[1] Tianjin Key Lab of Machine Learning, College of Intelligence and Computing, Tianjin University, China
[2] Dalian University of Technology, China   [3] Harbin Institute of Technology, China

Abstract

Recently, channel attention mechanism has demonstrated to offer great potential in improving the performance of deep convolutional neural networks (CNNs). However, most existing methods dedicate to developing more sophisticated attention modules for achieving better performance, which inevitably increase model complexity. To overcome the paradox of performance and complexity trade-off, this paper proposes an Efficient Channel Attention (ECA) module, which only involves a handful of parameters while bringing clear performance gain. By dissecting the channel attention module in SENet, we empirically show avoiding dimensionality reduction is important for learning channel attention, and appropriate cross-channel interaction can preserve performance while significantly decreasing model complexity. Therefore, we propose a local cross-channel interaction strategy without dimensionality reduction, which can be efficiently implemented via 1D convolution. Furthermore, we develop a method to adaptively select kernel size of 1D convolution, determining coverage of local cross-channel interaction. The proposed ECA module is efficient yet effective, e.g., the parameters and computations of our modules against backbone of ResNet50 are 80 vs. 24.37M and 4.7e-4 GFLOPs vs. 3.86 GFLOPs, respectively, and the performance boost is more than 2% in terms of Top-1 accuracy. We extensively evaluate our ECA module on image classification, object detection and instance segmentation with backbones of ResNets and MobileNetV2. The experimental results show our module is more efficient while performing favorably against its counterparts.

1. Introduction

Deep convolutional neural networks (CNNs) have been widely used in computer vision community, and have
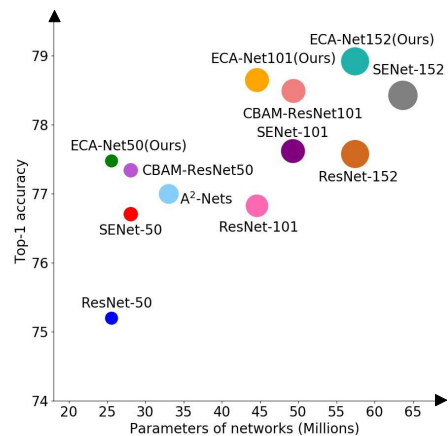
Figure 1. Comparison of various attention modules (i.e., SENet [14], CBAM [33], A^2-Nets [4] and ECA-Net) using ResNets [11] as backbone models in terms of classification accuracy, network parameters and FLOPs, indicated by radiuses of circles. Note that our ECA-Net obtains higher accuracy while having less model complexity.

achieved great progress in a broad range of tasks, *e.g.*, image classification, object detection and semantic segmentation. Starting from the groundbreaking AlexNet [17], many researches are continuously investigated to further improve the performance of deep CNNs [29, 30, 11, 15, 19, 20, 32]. Recently, incorporation of channel attention into convolution blocks has attracted a lot of interests, showing great potential in performance improvement [14, 33, 13, 4, 9, 18, 7]. One of the representative methods is squeeze-and-excitation networks (SENet) [14], which learns channel attention for each convolution block, bringing clear performance gain for various deep CNN architectures.

Following the setting of squeeze (i.e., feature aggregation) and excitation (i.e., feature recalibration) in SENet [14], some researches improve SE block by capturing more sophisticated channel-wise dependencies [33, 4, 9, 7] or by combining with additional spatial attention [33, 13, 7]. Although these methods have achieved
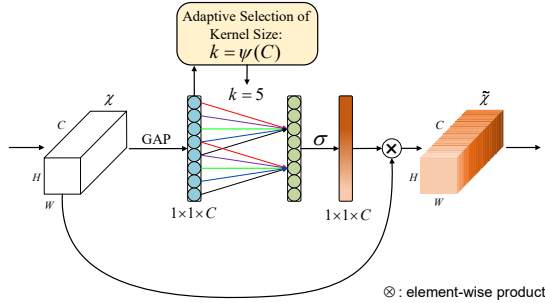
Figure 2. Diagram of our efficient channel attention (ECA) module. Given the aggregated features obtained by global average pooling (GAP), ECA generates channel weights by performing a fast $1D$ convolution of size k, where k is adaptively determined via a mapping of channel dimension C.

Model	No DR	Cross-channel Interaction	Lightweight
SENet [14]	×	√	−
CBAM [33]	×	√	×
GE-θ^- [13]	√	×	√
GE-θ [13]	√	×	×
GE-θ^+ [13]	×	√	×
A^2-Net [4]	×	√	×
GSoP-Net [9]	×	√	×
ECA-Net (Ours)	√	√	√

Table 1. Comparison of existing attention modules in terms of whether no channel dimensionality reduction (No DR), cross-channel interaction and less parameters than SE (indicated by lightweight) or not.

higher accuracy, they often bring higher model complexity and suffer from heavier computational burden. Different from the aforementioned methods that achieve better performance at the cost of higher model complexity, this paper focuses instead on a question: *Can one learn effective channel attention in a more efficient way?*

To answer this question, we first revisit the channel attention module in SENet. Specifically, given the input features, SE block first employs a global average pooling for each channel independently, then two fully-connected (FC) layers with non-linearity followed by a Sigmoid function are used to generate channel weights. The two FC layers are designed to capture non-linear cross-channel interaction, which involve dimensionality reduction for controlling model complexity. Although this strategy is widely used in subsequent channel attention modules [33, 13, 9], our empirical studies show dimensionality reduction brings side effect on channel attention prediction, and it is inefficient and unnecessary to capture dependencies across all channels.

Therefore, this paper proposes an *Efficient Channel Attention* (ECA) module for deep CNNs, which avoids dimensionality reduction and captures cross-channel interaction in an efficient way. As illustrated in Figure 2, after channel-wise global average pooling without dimensionality reduction, our ECA captures local cross-channel interaction by considering every channel and its k neighbors. Such method is proven to guarantee both efficiency and effectiveness. Note that our ECA can be efficiently implemented by fast $1D$ convolution of size k, where kernel size k represents the coverage of local cross-channel interaction, i.e., how many neighbors participate in attention prediction of one channel. To avoid manual tuning of k via cross-validation, we develop a method to adaptively determine k, where coverage of interaction (i.e., kernel size k) is proportional to channel dimension. As shown in Figure 1 and Table 3, as opposed to the backbone models [11], deep CNNs with our ECA

module (called ECA-Net) introduce very few additional parameters and negligible computations, while bringing notable performance gain. For example, for ResNet-50 with 24.37M parameters and 3.86 GFLOPs, the addtional parameters and computations of ECA-Net50 are 80 and 4.7e-4 GFLOPs, resecptively; meanwhile, ECA-Net50 outperforms ResNet-50 by 2.28% in terms of Top-1 accuracy.

Table 1 summarizes existing attention modules in terms of whether channel dimensionality reduction (DR), cross-channel interaction and lightweight model, where we can see that our ECA module learn effective channel attention by avoiding channel dimensionality reduction while capturing cross-channel interaction in an extremely lightweight way. To evaluate our method, we conduct experiments on ImageNet-1K [6] and MS COCO [23] in a variety of tasks using different deep CNN architectures.

The contributions of this paper are summarized as follows. (1) We dissect the SE block and empirically demonstrate avoiding dimensionality reduction and appropriate cross-channel interaction are important to learn effective and efficient channel attention, respectively. (2) Based on above analysis, we make an attempt to develop an extremely lightweight channel attention module for deep CNNs by proposing an *Efficient Channel Attention* (ECA), which increases little model complexity while bringing clear improvement. (3) The experimental results on ImageNet-1K and MS COCO demonstrate our method has lower model complexity than state-of-the-arts while achieving very competitive performance.

2. Related Work

Attention mechanism has proven to be a potential means to enhance deep CNNs. SE-Net [14] presents for the first time an effective mechanism to learn channel attention and achieves promising performance. Subsequently, development of attention modules can be roughly divided into two directions: (1) enhancement of feature aggregation; (2) combination of channel and spatial attentions. Specifically, CBAM [33] employs both average and max pooling to ag-

gregate features. GSoP [9] introduces a second-order pooling for more effective feature aggregation. GE [13] explores spatial extension using a depth-wise convolution [5] to aggregate features. CBAM [33] and scSE [27] compute spatial attention using a $2D$ convolution of kernel size $k \times k$, then combine it with channel attention. Sharing similar philosophy with Non-Local (NL) neural networks [32], GC-Net [2] develops a simplified NL network and integrates with the SE block, resulting in a lightweight module to model long-range dependency. Double Attention Networks (A^2-Nets) [4] introduces a novel relation function for NL blocks for image or video recognition. Dual Attention Network (DAN) [7] simultaneously considers NL-based channel and spatial attentions for semantic segmentation. However, most above NL-based attention modules can only be used in a single or a few convolution blocks due to their high model complexity. Obviously, all of the above methods focus on developing sophisticated attention modules for better performance. Different from them, our ECA aims at learning effective channel attention with low model complexity.

Our work is also related to efficient convolutions, which are designed for lightweight CNNs. Two widely used efficient convolutions are group convolutions [36, 34, 16] and depth-wise separable convolutions [5, 28, 37, 24]. As given in Table 2, although these efficient convolutions involve less parameters, they show little effectiveness in attention module. Our ECA module aims at capturing local cross-channel interaction, which shares some similarities with channel local convolutions [35] and channel-wise convolutions [8]; different from them, our method investigates a $1D$ convolution with adaptive kernel size to replace FC layers in channel attention module. Comparing with group and depth-wise separable convolutions, our method achieves better performance with lower model complexity.

3. Proposed Method

In this section, we first revisit the channel attention module in SENet [14] (i.e., SE block). Then, we make a empirical diagnosis of SE block by analyzing effects of dimensionality reduction and cross-channel interaction. This motivates us to propose our ECA module. In addition, we develop a method to adaptively determine parameter of our ECA, and finally show how to adopt it for deep CNNs.

3.1. Revisiting Channel Attention in SE Block

Let the output of one convolution block be $\mathcal{X} \in \mathbb{R}^{W \times H \times C}$, where W, H and C are width, height and channel dimension (i.e., number of filters). Accordingly, the weights of channels in SE block can be computed as

$$\boldsymbol{\omega} = \sigma(f_{\{\mathbf{W_1},\mathbf{W_2}\}}(g(\mathcal{X}))), \qquad (1)$$

where $g(\mathcal{X}) = \frac{1}{WH}\sum_{i=1,j=1}^{W,H}\mathcal{X}_{ij}$ is channel-wise global average pooling (GAP) and σ is a Sigmoid function. Let

Methods	Attention	#.Param.	Top-1	Top-5
Vanilla	N/A	0	75.20	92.25
SE	$\sigma(f_{\{\mathbf{W_1},\mathbf{W_2}\}}(\mathbf{y}))$	$2 \times C^2/r$	76.71	93.38
SE-Var1	$\sigma(\mathbf{y})$	0	76.00	92.90
SE-Var2	$\sigma(\mathbf{w} \odot \mathbf{y})$	C	77.07	93.31
SE-Var3	$\sigma(\mathbf{W}\mathbf{y})$	C^2	77.42	93.64
SE-GC1	$\sigma(\text{GC}_{16}(\mathbf{y}))$	$C^2/16$	76.95	93.47
SE-GC2	$\sigma(\text{GC}_{C/16}(\mathbf{y}))$	$16 \times C$	76.98	93.31
SE-GC3	$\sigma(\text{GC}_{C/8}(\mathbf{y}))$	$8 \times C$	76.96	93.38
ECA-NS	$\sigma(\boldsymbol{\omega})$ with Eq. (7)	$k \times C$	77.35	93.61
ECA (Ours)	$\sigma(\text{C1D}_k(\mathbf{y}))$	$k = 3$	**77.43**	**93.65**

Table 2. Comparison of various channel attention modules using ResNet-50 as backbone model on ImageNet. #.Param. indicates number of parameters of the channel attention module; $\odot$ indicates element-wise product; GC and C1D indicate group convolutions and $1D$ convolution, respectively; k is kernel size of C1D.

$\mathbf{y} = g(\mathcal{X})$, $f_{\{\mathbf{W_1},\mathbf{W_2}\}}$ takes the form

$$f_{\{\mathbf{W_1},\mathbf{W_2}\}}(\mathbf{y}) = \mathbf{W_2}\text{ReLU}(\mathbf{W_1}\mathbf{y}), \qquad (2)$$

where ReLU indicates the Rectified Linear Unit [25]. To avoid high model complexity, sizes of $\mathbf{W_1}$ and $\mathbf{W_2}$ are set to $C \times (\frac{C}{r})$ and $(\frac{C}{r}) \times C$, respectively. We can see that $f_{\{\mathbf{W_1},\mathbf{W_2}\}}$ involves all parameters of channel attention block. While dimensionality reduction in Eq. (2) can reduce model complexity, it destroys the direct correspondence between channel and its weight. For example, one single FC layer predicts weight of each channel using a linear combination of all channels. But Eq. (2) first projects channel features into a low-dimensional space and then maps them back, making correspondence between channel and its weight be indirect.

3.2. Efficient Channel Attention (ECA) Module

After revisiting SE block, we conduct empirical comparisons for analyzing effects of channel dimensionality reduction and cross-channel interaction on channel attention learning. According to these analyses, we propose our efficient channel attention (ECA) module.

3.2.1 Avoiding Dimensionality Reduction

As discussed above, dimensionality reduction in Eq. (2) makes correspondence between channel and its weight be indirect. To verify its effect, we compare the original SE block with its three variants (i.e., SE-Var1, SE-Var2 and SE-Var3), all of which do not perform dimensionality reduction. As presented in Table 2, SE-Var1 with no parameter is still superior to the original network, indicating channel attention has ability to improve performance of deep CNNs. Meanwhile, SE-Var2 learns the weight of each channel independently, which is slightly superior to SE block while involving less parameters. It may suggest that channel and its weight needs a direct correspondence while avoiding di-

mensionality reduction is more important than consideration of nonlinear channel dependencies. Additionally, SE-Var3 employing one single FC layer performs better than two FC layers with dimensionality reduction in SE block. All of above results clearly demonstrate avoiding dimensionality reduction is helpful to learn effective channel attention. Therefore, we develop our ECA module without channel dimensionality reduction.

3.2.2 Local Cross-Channel Interaction

Given the aggregated feature $\mathbf{y} \in \mathbb{R}^C$ without dimensionality reduction, channel attention can be learned by

$$\boldsymbol{\omega} = \sigma(\mathbf{W}\mathbf{y}), \tag{3}$$

where $\mathbf{W}$ is a $C \times C$ parameter matrix. In particular, for SE-Var2 and SE-Var3 we have

$$\mathbf{W} = \begin{cases} \mathbf{W}_{var2} = \begin{bmatrix} w^{1,1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & w^{C,C} \end{bmatrix}, \\ \mathbf{W}_{var3} = \begin{bmatrix} w^{1,1} & \cdots & w^{1,C} \\ \vdots & \ddots & \vdots \\ w^{1,C} & \cdots & w^{C,C} \end{bmatrix}, \end{cases} \tag{4}$$

where $\mathbf{W}_{var2}$ for SE-Var2 is a diagonal matrix, involving C parameters; $\mathbf{W}_{var3}$ for SE-Var3 is a full matrix, involving $C \times C$ parameters. As shown in Eq. (4), the key difference is that SE-Var3 considers cross-channel interaction while SE-Var2 does not, and consequently SE-Var3 achieves better performance. This result indicates that cross-channel interaction is beneficial to learn channel attention. However, SE-Var3 requires a mass of parameters, leading to high model complexity, especially for large channel numbers.

A possible compromise between SE-Var2 and SE-Var3 is extension of $\mathbf{W}_{var2}$ to a block diagonal matrix, i.e.,

$$\mathbf{W}_G = \begin{bmatrix} \mathbf{W}_G^1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{W}_G^G \end{bmatrix}, \tag{5}$$

where Eq. (5) divides channel into G groups each of which includes C/G channels, and learns channel attention in each group independently, which captures cross-channel interaction in a local manner. Accordingly, it involves C^2/G parameters. From perspective of convolution, SE-Var2, SE-Var3 and Eq. (5) can be regarded as a depth-wise separable convolution, a FC layer and group convolutions, respectively. Here, SE block with group convolutions (SE-GC) is indicated by $\sigma(\mathrm{GC}_G(\mathbf{y})) = \sigma(\mathbf{W}_G\mathbf{y})$. However, as shown in [24], excessive group convolutions will increase memory access cost and so decrease computational efficiency. Furthermore, as shown in Table 2, SE-GC with varying groups

bring no gain over SE-Var2, indicating it is not an effective scheme to capture local cross-channel interaction. The reason may be that SE-GC completely discards dependences among different groups.

In this paper, we explore another method to capture local cross-channel interaction, aiming at guaranteeing both efficiency and effectiveness. Specifically, we employ a band matrix $\mathbf{W}_k$ to learn channel attention, and $\mathbf{W}_k$ has

$$\begin{bmatrix} w^{1,1} & \cdots & w^{1,k} & 0 & 0 & \cdots & \cdots & 0 \\ 0 & w^{2,2} & \cdots & w^{2,k+1} & 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & w^{C,C-k+1} & \cdots & w^{C,C} \end{bmatrix}. \tag{6}$$

Clearly, $\mathbf{W}_k$ in Eq. (6) involves $k \times C$ parameters, which is usually less than those of Eq. (5). Furthermore, Eq. (6) avoids complete independence among different groups in Eq. (5). As compared in Table 2, the method in Eq. (6) (namely ECA-NS) outperforms SE-GC of Eq. (5). As for Eq. (6), the weight of y_i is calculated by only considering interaction between y_i and its k neighbors, i.e.,

$$\omega_i = \sigma\left(\sum_{j=1}^{k} w_i^j y_i^j\right), \ y_i^j \in \Omega_i^k, \tag{7}$$

where Ω_i^k indicates the set of k adjacent channels of y_i.

A more efficient way is to make all channels share the same learning parameters, i.e.,

$$\omega_i = \sigma\left(\sum_{j=1}^{k} w^j y_i^j\right), \ y_i^j \in \Omega_i^k. \tag{8}$$

Note that such strategy can be readily implemented by a fast $1D$ convolution with kernel size of k, i.e.,

$$\boldsymbol{\omega} = \sigma(\mathrm{C1D}_k(\mathbf{y})), \tag{9}$$

where C1D indicates $1D$ convolution. Here, the method in Eq. (9) is called by efficient channel attention (ECA) module, which only involves k parameters. As presented in Table 2, our ECA module with $k = 3$ achieves similar results with SE-var3 while having much lower model complexity, which guarantees both efficiency and effectiveness by appropriately capturing local cross-channel interaction.

3.2.3 Coverage of Local Cross-Channel Interaction

Since our ECA module (9) aims at appropriately capturing local cross-channel interaction, so the coverage of interaction (i.e., kernel size k of $1D$ convolution) needs to be determined. The optimized coverage of interaction could be tuned manually for convolution blocks with different channel numbers in various CNN architectures. However, manual tuning via cross-validation will cost a lot of computing resources. Group convolutions have been successfully

```python
def EfficientChannelAttention(x, gamma=2, b=1):
    # x: input features with shape [N, C, H, W]
    # gamma, b: parameters of mapping function

    N, C, H, W = x.size()

    t = int(abs((log(C, 2) + b) / gamma))
    k = t if t % 2 else t + 1

    avg_pool = nn.AdaptiveAvgPool2d(1)
    conv = nn.Conv1d(1, 1, kernel_size=k, padding=int(k/2),
                     bias=False)

    y = avg_pool(x)
    y = conv(y.squeeze(-1).transpose(-1, -2))
    y = y.transpose(-1, -2).unsqueeze(-1)

    return x * y.expand_as(x)
```

Figure 3. PyTorch code of our ECA module.

adopted to improve CNN architectures [36, 34, 16], where high-dimensional (low-dimensional) channels involve long range (short range) convolutions given the fixed number of groups. Sharing the similar philosophy, it is reasonable that the coverage of interaction (i.e., kernel size k of $1D$ convolution) is proportional to channel dimension C. In other words, there may exist a mapping ϕ between k and C:

$$C = \phi(k). \tag{10}$$

The simplest mapping is a linear function, i.e., $\phi(k) = \gamma * k - b$. However, the relations characterized by linear function are too limited. On the other hand, it is well known that channel dimension C (i.e., number of filters) usually is set to power of 2. Therefore, we introduce a possible solution by extending the linear function $\phi(k) = \gamma * k - b$ to a non-linear one, i.e.,

$$C = \phi(k) = 2^{(\gamma * k - b)}. \tag{11}$$

Then, given channel dimension C, kernel size k can be adaptively determined by

$$k = \psi(C) = \left| \frac{log_2(C)}{\gamma} + \frac{b}{\gamma} \right|_{odd}, \tag{12}$$

where $|t|_{odd}$ indicates the nearest odd number of t. In this paper, we set γ and b to 2 and 1 throughout all the experiments, respectively. Clearly, through the mapping ψ, high-dimensional channels have longer range interaction while low-dimensional ones undergo shorter range interaction by using a non-linear mapping.

3.3. ECA Module for Deep CNNs

Figure 2 illustrates the overview of our ECA module. After aggregating convolution features using GAP without dimensionality reduction, ECA module first adaptively determines kernel size k, and then performs $1D$ convolution followed by a Sigmoid function to learn channel attention. For applying our ECA to deep CNNs, we replace SE block by our ECA module following the same configuration in [14]. The resulting networks are named by ECA-Net. Figure 3 gives PyTorch code of our ECA.

4. Experiments

In this section, we evaluate the proposed method on large-scale image classification, object detection and instance segmentation using ImageNet [6] and MS COCO [23], respectively. Specifically, we first assess the effect of kernel size on our ECA module, and compare with state-of-the-art counterparts on ImageNet. Then, we verify the effectiveness of our ECA-Net on MS COCO using Faster R-CNN [26], Mask R-CNN [10] and RetinaNet [22].

4.1. Implementation Details

To evaluate our ECA-Net on ImageNet classification, we employ four widely used CNNs as backbone models, including ResNet-50 [11], ResNet-101 [11], ResNet-512 [11] and MobileNetV2 [28]. For training ResNets with our ECA, we adopt exactly the same data augmentation and hyper-parameter settings in [11, 14]. Specifically, the input images are randomly cropped to 224×224 with random horizontal flipping. The parameters of networks are optimized by stochastic gradient descent (SGD) with weight decay of 1e-4, momentum of 0.9 and mini-batch size of 256. All models are trained within 100 epochs by setting the initial learning rate to 0.1, which is decreased by a factor of 10 per 30 epochs. For training MobileNetV2 with our ECA, we follow the settings in [28], where networks are trained within 400 epochs using SGD with weight decay of 4e-5, momentum of 0.9 and mini-batch size of 96. The initial learning rate is set to 0.045, and is decreased by a linear decay rate of 0.98. For testing on the validation set, the shorter side of an input image is first resized to 256 and a center crop of 224×224 is used for evaluation. All models are implemented by PyTorch toolkit[1].

We further evaluate our method on MS COCO using Faster R-CNN [26], Mask R-CNN [10] and RetinaNet [22], where ResNet-50 and ResNet-101 along with FPN [21] are used as backbone models. We implement all detectors by using MMDetection toolkit [3] and employ the default settings. Specifically, the shorter side of input images are resized to 800, then all models are optimized using SGD with weight decay of 1e-4, momentum of 0.9 and mini-batch size of 8 (4 GPUs with 2 images per GPU). The learning rate is initialized to 0.01 and is decreased by a factor of 10 after 8 and 11 epochs, respectively. We train all detectors within 12 epochs on train2017 of COCO and report the results on val2017 for comparison. All programs run on a PC equipped with four RTX 2080Ti GPUs and an Intel(R)

---
[1]https://github.com/BangguWu/ECANet

Method	Backbone Models	#.Param.	FLOPs	Training	Inference	Top-1	Top-5
ResNet [11]		**24.37M**	**3.86G**	**1024 FPS**	**1855 FPS**	75.20	92.52
SENet [14]		26.77M	3.87G	759 FPS	1620 FPS	76.71	93.38
CBAM [33]		26.77M	3.87G	472 FPS	1213 FPS	77.34	93.69
A^2-Nets [4][†]	ResNet-50	33.00M	6.50G	N/A	N/A	77.00	93.50
GCNet [2]		28.08M	3.87G	N/A	N/A	77.70	93.66
GSoP-Net1 [9]		28.05M	6.18G	596 FPS	1383 FPS	77.68	**93.98**
AA-Net [1][†,◊]		25.80M	4.15G	N/A	N/A	**77.70**	93.80
ECA-Net (Ours)		**24.37M**	**3.86G**	785 FPS	1805 FPS	77.48	93.68
ResNet [11]		**42.49M**	**7.34G**	**386 FPS**	**1174 FPS**	76.83	93.48
SENet [14]		47.01M	7.35G	367 FPS	1044 FPS	77.62	93.93
CBAM [33]	ResNet-101	47.01M	7.35G	270 FPS	635 FPS	78.49	94.31
AA-Net [1][†,◊]		45.40M	8.05G	N/A	N/A	**78.70**	**94.40**
ECA-Net (Ours)		**42.49M**	7.35G	380 FPS	1089 FPS	78.65	94.34
ResNet [11]		**57.40M**	**10.82G**	**281 FPS**	**815 FPS**	77.58	93.66
SENet [14]	ResNet-152	63.68M	10.85G	268 FPS	761 FPS	78.43	94.27
ECA-Net (Ours)		**57.40M**	10.83G	279 FPS	785 FPS	**78.92**	**94.55**
MobileNetV2 [28]		**3.34M**	**319.4M**	**711 FPS**	**2086 FPS**	71.64	90.20
SENet	MobileNetV2	3.40M	320.1M	671 FPS	2000 FPS	72.42	90.67
ECA-Net (Ours)		**3.34M**	319.9M	676 FPS	2010 FPS	**72.56**	**90.81**

Table 3. Comparison of different attention methods on ImageNet in terms of network parameters (#.Param.), floating point operations per second (FLOPs), training or inference speed (frame per second, FPS), and Top-1/Top-5 accuracy (in %). †: Since the source code and models of A^2-Nets and AA-Net are publicly unavailable, we do not compare their running time. ◊: AA-Net is trained with Inception data augmentation and different setting of learning rates.

Xeon Silver 4112 CPU@2.60GHz.

4.2. Image Classification on ImageNet-1K

Here, we first assess the effect of kernel size on our ECA module and verify the effectiveness of our method to adaptively determine kernel size, then we compare with state-of-the-art counterparts and CNN models using ResNet-50, ResNet-101, ResNet-152 and MobileNetV2.

4.2.1 Effect of Kernel Size (k) on ECA Module

As shown in Eq. (9), our ECA module involves a parameter k, i.e., kernel size of $1D$ convolution. In this part, we evaluate its effect on our ECA module and validate the effectiveness of our method for adaptive selection of kernel size. To this end, we employ ResNet-50 and ResNet-101 as backbone models, and train them with our ECA module by setting k be from 3 to 9. The results are illustrated in Figure 4, from it we have the following observations.

Firstly, when k is fixed in all convolution blocks, ECA module obtains the best results at $k = 9$ and $k = 5$ for ResNet-50 and ResNet-101, respectively. Since ResNet-101 has more intermediate layers that dominate performance of ResNet-101, it may prefer to small kernel size. Besides, these results show that different deep CNNs have various optimal k, and k has a clear effect on performance of ECA-Net. Furthermore, accuracy fluctuations ($\sim$0.5%) of ResNet-101 are larger than those ($\sim$0.15%) of ResNet-50, and we conjecture the reason is that the deeper net-
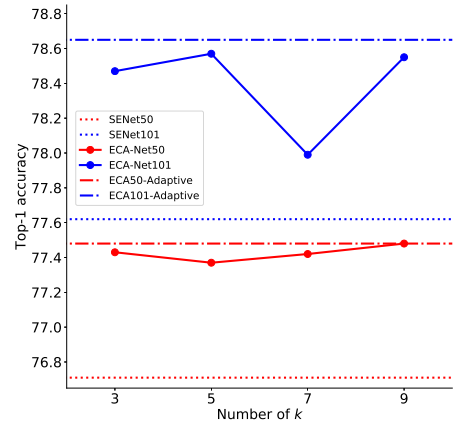

Figure 4. Results of our ECA module with various numbers of k using ResNet-50 and ResNet-101 as backbone models. Here, we also give the results of ECA module with adaptive selection of kernel size and compare with SENet as baseline.

works are more sensitive to the fixed kernel size than the shallower ones. Additionally, kernel size that is adaptively determined by Eq. (12) usually outperforms the fixed ones, while it can avoid manual tuning of parameter k via cross-validation. Above results demonstrate the effectiveness of our adaptive kernel size selection in attaining better and stable results. Finally, ECA module with various numbers of k consistently outperform SE block, verifying that avoiding dimensionality reduction and local cross-channel interaction have positive effects on learning channel attention.

4.2.2 Comparisons Using Different Deep CNNs

ResNet-50 We compare our ECA module with several state-of-the-art attention methods using ResNet-50 on ImageNet, including SENet [14], CBAM [33], A^2-Nets [4], AA-Net [1], GSoP-Net1 [9] and GCNet [2]. The evaluation metrics include both efficiency (i.e., network parameters, floating point operations per second (FLOPs) and training/inference speed) and effectiveness (i.e., Top-1/Top-5 accuracy). For comparison, we duplicate the results of ResNet and SENet from [14], and report the results of other compared methods in their original papers. To test training/inference speed of various models, we employ publicly available models of the compared CNNs, and run them on the same computing platform. The results are given in Table 3, where we can see that our ECA-Net shares almost the same model complexity (i.e., network parameters, FLOPs and speed) with the original ResNet-50, while achieving 2.28% gains in Top-1 accuracy. Comparing with state-of-the-art counterparts (i.e., SENet, CBAM, A^2-Nets, AA-Net, GSoP-Net1 and GCNet), ECA-Net obtains better or competitive results while benefiting lower model complexity.

ResNet-101 Using ResNet-101 as backbone model, we compare our ECA-Net with SENet [14], CBAM [33] and AA-Net [1]. From Table 3 we can see that ECA-Net outperforms the original ResNet-101 by 1.8% with almost the same model complexity. Sharing the same tendency on ResNet-50, ECA-Net is superior to SENet and CBAM while it is very competitive to AA-Net with lower model complexity. Note that AA-Net is trained with Inception data augmentation and different setting of learning rates.

ResNet-152 Using ResNet-152 as backbone model, we compare our ECA-Net with SENet [14]. From Table 3 we can see that ECA-Net improves the original ResNet-152 over about 1.3% in terms of Top-1 accuracy with almost the same model complexity. Comparing with SENet, ECA-Net achieves 0.5% gain in terms of Top-1 with lower model complexity. The results with respect to ResNet-50, ResNet-101 and ResNet-152 demonstrate the effectiveness of our ECA module on the widely used ResNet architectures.

MobileNetV2 Besides ResNet architectures, we also verify the effectiveness of our ECA module on lightweight CNN architectures. To this end, we employ MobileNetV2 [28] as backbone model and compare our ECA module with SE block. In particular, we integrate SE block and ECA module in convolution layer before residual connection lying in each 'bottleneck' of MobileNetV2, and parameter r of SE block is set to 8. All models are trained using exactly the same settings. The results in Table 3 show our ECA-Net improves the original MobileNetV2 and SENet by about 0.9% and 0.14% in terms of Top-1 accuracy, respectively. Furthermore, our ECA-Net has smaller model size and faster training/inference speed than SENet. Above results verify the efficiency and effectiveness of our ECA module again.

CNN Models	#.Param.	FLOPs	Top-1	Top-5
ResNet-200	74.45M	14.10G	78.20	94.00
Inception-v3	25.90M	5.36G	77.45	93.56
ResNeXt-101	46.66M	7.53G	78.80	94.40
DenseNet-264 (k=32)	31.79M	5.52G	77.85	93.78
DenseNet-161 (k=48)	27.35M	7.34G	77.65	93.80
ECA-Net50 (Ours)	24.37M	3.86G	77.48	93.68
ECA-Net101 (Ours)	42.49M	7.35G	78.65	94.34

Table 4. Comparisons with state-of-the-art CNNs on ImageNet.

4.2.3 Comparisons with Other CNN Models

At the end of this part, we compare our ECA-Net50 and ECA-Net101 with other state-of-the-art CNN models, including ResNet-200 [12], Inception-v3 [31], ResNeXt [34], DenseNet [15]. These CNN models have deeper and wider architectures, and their results all are copied from the original papers. As presented in Table 4, ECA-Net101 outperforms ResNet-200, indicating that our ECA-Net can improve the performance of deep CNNs using much less computational cost. Meanwhile, our ECA-Net101 is very competitive to ResNeXt-101, while the latter one employs more convolution filters and expensive group convolutions. In addition, ECA-Net50 is comparable to DenseNet-264 (k=32), DenseNet-161 (k=48) and Inception-v3, but it has lower model complexity. All above results demonstrate that our ECA-Net performs favorably against state-of-the-art CNNs while benefiting much lower model complexity. Note that our ECA also has great potential to further improve the performance of the compared CNN models.

4.3. Object Detection on MS COCO

In this subsection, we evaluate our ECA-Net on object detection task using Faster R-CNN [26], Mask R-CNN [10] and RetinaNet [22]. We mainly compare ECA-Net with ResNet and SENet. All CNN models are pre-trained on ImageNet, then are transferred to MS COCO by fine-tuning.

4.3.1 Comparisons Using Faster R-CNN

Using Faster R-CNN as the basic detector, we employ ResNets of 50 and 101 layers along with FPN [21] as backbone models. As shown in Table 5, integration of either SE block or our ECA module can improve performance of object detection by a clear margin. Meanwhile, our ECA outperforms SE block by 0.3% and 0.7% in terms of AP using ResNet-50 and ResNet-101, respectively.

4.3.2 Comparisons Using Mask R-CNN

We further exploit Mask R-CNN to verify the effectiveness of our ECA-Net on object detection task. As shown in Table 5, our ECA module is superior to the original ResNet by 1.8% and 1.9% in terms of AP under the settings of

Methods	Detectors	#.Param.	GFLOPs	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-50		**41.53 M**	**207.07**	36.4	58.2	39.2	21.8	40.0	46.2
+ SE block		44.02 M	207.18	37.7	60.1	40.9	22.9	41.9	**48.2**
+ ECA (Ours)	Faster R-CNN	**41.53 M**	207.18	**38.0**	**60.6**	**40.9**	**23.4**	**42.1**	48.0
ResNet-101		**60.52 M**	**283.14**	38.7	60.6	41.9	22.7	43.2	50.4
+ SE block		65.24 M	283.33	39.6	62.0	43.1	23.7	44.0	**51.4**
+ ECA (Ours)		**60.52 M**	283.32	**40.3**	**62.9**	**44.0**	**24.5**	**44.7**	51.3
ResNet-50		**44.18 M**	**275.58**	37.2	58.9	40.3	22.2	40.7	48.0
+ SE block		46.67 M	275.69	38.7	60.9	42.1	23.4	42.7	**50.0**
+ 1 NL		46.50 M	288.70	38.0	59.8	41.0	N/A	N/A	N/A
+ GC block	Mask R-CNN	46.90 M	279.60	**39.4**	**61.6**	**42.4**	N/A	N/A	N/A
+ ECA (Ours)		**44.18 M**	275.69	39.0	61.3	42.1	**24.2**	**42.8**	49.9
ResNet-101		**63.17 M**	**351.65**	39.4	60.9	43.3	23.0	43.7	51.4
+ SE block		67.89 M	351.84	40.7	62.5	44.3	23.9	45.2	52.8
+ ECA (Ours)		**63.17 M**	351.83	**41.3**	**63.1**	**44.8**	**25.1**	**45.8**	**52.9**
ResNet-50		**37.74 M**	**239.32**	35.6	55.5	38.2	20.0	39.6	46.8
+ SE block		40.23 M	239.43	37.1	57.2	**39.9**	21.2	40.7	**49.3**
+ ECA (Ours)	RetinaNet	**37.74 M**	239.43	**37.3**	**57.7**	39.6	**21.9**	**41.3**	48.9
ResNet-101		**56.74 M**	**315.39**	37.7	57.5	40.4	21.1	42.2	49.5
+ SE block		61.45 M	315.58	38.7	59.1	41.6	22.1	43.1	**50.9**
+ ECA (Ours)		**56.74 M**	315.57	**39.1**	**59.9**	**41.8**	**22.8**	**43.4**	50.6

Table 5. Object detection results of different methods on COCO val2017.

50 and 101 layers, respectively. Meanwhile, ECA module achieves 0.3% and 0.6% gains over SE block using ResNet-50 and ResNet-101 as backbone models, respectively. Using ResNet-50, ECA is superior to one NL [32], and is comparable to GC block [2] using lower model complexity.

4.3.3 Comparisons Using RetinaNet

Additionally, we verify the effectiveness of our ECA-Net on object detection using one-stage detector, i.e., RetinaNet. As compared in Table 5, our ECA-Net outperforms the original ResNet by 1.8% and 1.4% in terms of AP for the networks of 50 and 101 layers, respectively. Meanwhile, ECA-Net improves SE-Net over 0.2% and 0.4% for ResNet-50 and ResNet-101, respectively. In summary, the results in Table 5 demonstrate that our ECA-Net can well generalize to object detection task. Specifically, ECA module brings clear improvement over the original ResNet, while outperforming SE block using lower model complexity. In particular, our ECA module achieves more gains for small objects, which are usually more difficult to be detected.

4.4. Instance Segmentation on MS COCO

Then, we give instance segmentation results of our ECA module using Mask R-CNN on MS COCO. As compared in Table 6, ECA module achieves notable gain over the original ResNet while performing better than SE block with less model complexity. For ResNet-50 as backbone, ECA with lower model complexity is superior one NL [32], and is comparable to GC block [2]. These results verify our ECA module has good generalization ability for various tasks.

Methods	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-50	34.1	55.5	36.2	16.1	36.7	50.0
+ SE block	35.4	57.4	**37.8**	17.1	38.6	51.8
+ 1 NL	34.7	56.7	36.6	N/A	N/A	N/A
+ GC block	**35.7**	**58.4**	37.6	N/A	N/A	N/A
+ ECA (Ours)	35.6	58.1	37.7	**17.6**	**39.0**	**51.8**
ResNet-101	35.9	57.7	38.4	16.8	39.1	53.6
+ SE block	36.8	59.3	39.2	17.2	40.3	53.6
+ ECA (Ours)	**37.4**	**59.9**	**39.8**	**18.1**	**41.1**	**54.1**

Table 6. Instance segmentation results of different methods using Mask R-CNN on COCO val2017.

5. Conclusion

In this paper, we focus on learning effective channel attention for deep CNNs with low model complexity. To this end, we propose an efficient channel attention (ECA) module, which generates channel attention through a fast $1D$ convolution, whose kernel size can be adaptively determined by a non-linear mapping of channel dimension. Experimental results demonstrate our ECA is an extremely lightweight plug-and-play block to improve the performance of various deep CNN architectures, including the widely used ResNets and lightweight MobileNetV2. Moreover, our ECA-Net exhibits good generalization ability in object detection and instance segmentation tasks. In future, we will apply our ECA module to more CNN architectures (e.g., ResNeXt and Inception [31]) and further investigate incorporation of ECA with spatial attention module.

References

[1] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V. Le. Attention augmented convolutional networks. *arXiv:1904.09925*, 2019.

[2] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *ICCV Workshops*, 2019.

[3] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv:1906.07155*, 2019.

[4] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. A^2-Nets: Double attention networks. In *NIPS*, 2018.

[5] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

[7] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *CVPR*, 2019.

[8] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Channelnets: Compact and efficient convolutional neural networks via channel-wise convolutions. In *NeurIPS*, 2018.

[9] Zilin Gao, Jiangtao Xie, Qilong Wang, and Peihua Li. Global second-order pooling convolutional networks. In *CVPR*, 2019.

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *ICCV*, pages 2980–2988, 2017.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.

[13] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Andrea Vedaldi. Gather-excite: Exploiting feature context in convolutional neural networks. In *NeurIPS*, 2018.

[14] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

[15] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[16] Yani Ioannou, Duncan Robertson, Roberto Cipolla, and Antonio Criminisi. Deep roots: Improving cnn efficiency with hierarchical filter groups. In *CVPR*, 2017.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[18] Huayu Li. Channel locality block: A variant of squeeze-and-excitation. *arXiv*, 1901.01493, 2019.

[19] Peihua Li, Jiangtao Xie, Qilong Wang, and Wangmeng Zuo. Is second-order information helpful for large-scale visual recognition? In *ICCV*, 2017.

[20] Yanghao Li, Naiyan Wang, Jiaying Liu, and Xiaodi Hou. Factorized bilinear models for image recognition. In *ICCV*, 2017.

[21] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 936–944, 2017.

[22] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

[24] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet V2: Practical guidelines for efficient CNN architecture design. In *ECCV*, 2018.

[25] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[26] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(6):1137–1149, 2017.

[27] Abhijit Guha Roy, Nassir Navab, and Christian Wachinger. Recalibrating fully convolutional networks with spatial and channel "squeeze and excitation" blocks. *IEEE Trans. Med. Imaging*, 38(2):540–549, 2019.

[28] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

[29] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[30] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[31] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[32] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

[33] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: Convolutional block attention module. In *ECCV*, 2018.

[34] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

[35] Dong-Qing Zhang. Clcnet: Improving the efficiency of convolutional neural network using channel local convolutions. In *CVPR*, 2018.

[36] Ting Zhang, Guo-Jun Qi, Bin Xiao, and Jingdong Wang. Interleaved group convolutions. In *ICCV*, 2017.

[37] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018.