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]College of Intelligence and Computing, Tianjin University, China
[2]School of Information and Communication Engineering, Dalian University of Technology, China
[3]School of Computer Science and Technology, Harbin Institute of Technology, China

Abstract

Channel attention has recently 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 to achieve better performance, inevitably increasing the computational burden. To overcome the paradox of performance and complexity trade-off, this paper makes an attempt to investigate an extremely lightweight attention module for boosting the performance of deep CNNs. In particular, we propose an *Efficient Channel Attention* (ECA) module, which only involves k ($k \leq 9$) parameters but brings clear performance gain. By revisiting the channel attention module in SENet, we empirically show avoiding dimensionality reduction and appropriate cross-channel interaction are important to learn effective channel attention. Therefore, we propose a local cross-channel interaction strategy without dimension reduction, which can be efficiently implemented by a fast $1D$ convolution. Furthermore, we develop a function of channel dimension to adaptively determine kernel size of $1D$ convolution, which stands for coverage of local cross-channel interaction. Our ECA module can be flexibly incorporated into existing CNN architectures, and the resulting CNNs are named by ECA-Net. We extensively evaluate the proposed ECA-Net on image classification, object detection and instance segmentation with backbones of ResNets and MobileNetV2. The experimental results show our ECA-Net is more efficient while performing favorably against its counterparts. The source code and models can be available at https://github.com/BangguWu/ECANet.

Introduction

Deep convolutional neural networks (CNNs) have been widely used in artificial intelligence, and have achieved great progress in a broad range of tasks, e.g., image classification, object detection and semantic segmentation. Starting from the groundbreaking AlexNet (Krizhevsky, Sutskever, and Hinton 2012), many researches are continuously investigated to further improve the performance of deep CNNs (Simonyan and Zisserman 2015; Szegedy et al. 2015; He et al. 2016a; Huang et al. 2017; Li et al. 2017a; 2017b; Wang et al. 2018). Recently, incorporation of attention mechanism into convolution blocks has attracted a lot of attentions, showing great potential for performance improvement (Hu, Shen, and Sun 2018; Woo et al. 2018; Hu et al. 2018; Chen et al. 2018; Gao et al. 2019; Fu et al. 2019). Among these methods, one of the representative works is squeeze-and-excitation networks (SENet) (Hu, Shen, and Sun 2018), which learns channel attention for each convolution block, bringing clear performance gain over various deep CNN architectures.

Following the setting of squeeze (i.e., feature aggregation) and excitation (i.e., feature recalibration) in SENet (Hu, Shen, and Sun 2018), some researches improve SE block by capturing more sophisticated channel-wise dependencies (Woo et al. 2018; Chen et al. 2018; Gao et al. 2019; Fu et al. 2019) or by combining with additional spatial attention (Woo et al. 2018; Hu et al. 2018; Fu et al. 2019).

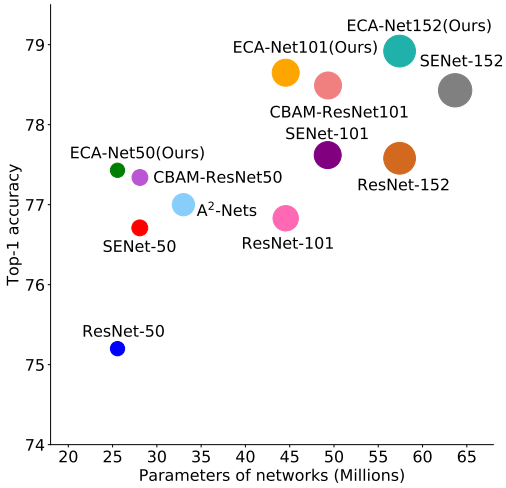


Figure 1: Comparison of various attention modules (i.e., SENet (Hu, Shen, and Sun 2018), CBAM (Woo et al. 2018), A^2-Nets (Chen et al. 2018) and ECA-Net) using ResNets (He et al. 2016a) as backbone models in terms of accuracy, network parameters and FLOPs. Sizes of circles indicate model computation (FLOPs). Clearly, our ECA-Net obtains higher accuracy while having less model complexity.

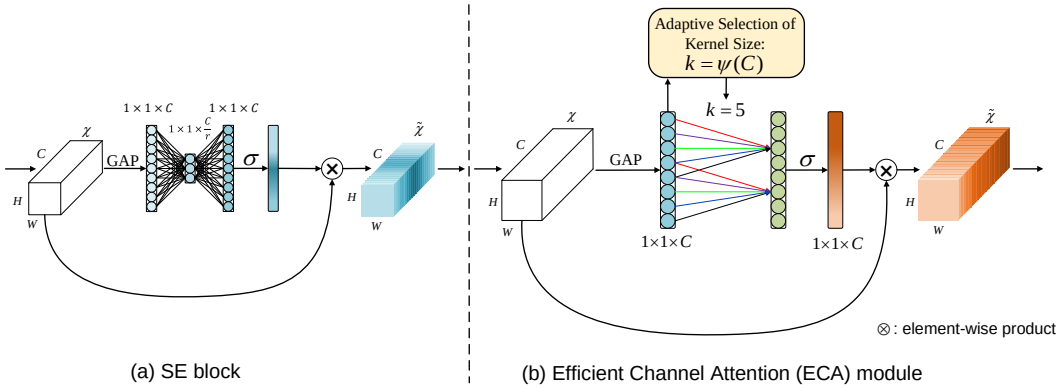


Figure 2: Comparison of (a) SE block and (b) our efficient channel attention (ECA) module. Given the aggregated feature using global average pooling (GAP), SE block computes weights using two FC layers. Differently, ECA generates channel weights by performing a fast $1D$ convolution of size k, where k is adaptively determined via a function of channel dimension C.

Although these methods have achieved 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 weight of each channel. The two FC layers are designed to capture non-linear cross-channel interaction, which involve dimensionality reduction for avoiding too high model complexity. Although this policy is widely used in the subsequent channel attention modules (Woo et al. 2018; Hu et al. 2018; Gao et al. 2019), our empirical analyses demonstrate dimensionality reduction will bring side effect on prediction of channel attention, and it is inefficient and unnecessary to capture dependencies across all channels.

Based on the above analyses, avoiding dimensionality reduction and appropriate cross-channel interaction are suggested to play a vital role in developing channel attention mechanisms. Therefore, this paper proposes an *Efficient Channel Attention* (ECA) module for deep CNNs based on above two properties. As illustrated in Figure 2 (b), after channel-wise global average pooling without dimensionality reduction, our ECA captures local cross-channel interaction by considering every channel and its k neighbors. As such, our ECA can be efficiently implemented by a fast $1D$ convolution of size k. The kernel size k represents the coverage of local cross-channel interaction, i.e., how many neighbors participate in attention prediction of one channel. Clearly, it will affect both efficiency and effectiveness of ECA. It is reasonable that coverage of interaction is in connection with channel dimension, so we propose a function associated with channel dimension to adaptively determine k. As shown in Figure 1 and Table 2, as opposed to the backbone models (He et al. 2016a), 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. To evaluate our method, we conduct experiments on ImageNet-1K (Deng et al. 2009) and MS COCO (Lin et al. 2014) using different deep CNN architectures and tasks. The contributions of this paper are summarized as follows.

- We empirically demonstrate avoiding dimensionality reduction and appropriate cross-channel interaction are important to learn efficient and effective channel attention for deep CNNs.
- We make an attempt to develop an extremely lightweight channel attention module for deep CNNs by proposing a novel *Efficient Channel Attention* (ECA), which increases little model complexity but brings clear improvement.
- 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.

Related Work

Attention mechanism has proven to be a potential means to reinforce deep CNNs. SE-Net (Hu, Shen, and Sun 2018) 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 (Woo et al. 2018) employs both average and max pooling to aggregate features. GSoP (Gao et al. 2019) introduces a second-order pooling for more effective feature aggregation. GE (Hu et al. 2018) explores spatial

extension using a depth-wise convolution (Chollet 2017) to aggregate features. scSE (Roy, Navab, and Wachinger 2019) and CBAM (Woo et al. 2018) 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 neural networks (Wang et al. 2018), Double Attention Networks (A^2-Nets) (Chen et al. 2018) introduces a novel relation function for image or video recognition, while Dual Attention Network (DAN) (Fu et al. 2019) and Criss-Cross Network (CCNet) (Huang et al. 2019) simultaneously consider non-local channel and non-local spatial attentions for semantic segmentation. Analogously, Li et al. propose an Expectation-Maximization Attention (EMA) module for semantic segmentation (Li et al. 2019). However, these non-local attention modules can only be used in one 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 CNN architectures. The two most widely used efficient convolutions are group convolutions (Zhang et al. 2017; Xie et al. 2017; Ioannou et al. 2017) and depth-wise separable convolutions (Chollet 2017; Sandler et al. 2018; Zhang et al. 2018; Ma et al. 2018). As given in Table 1, 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 (Zhang 2018) and channel-wise convolutions (Gao, Wang, and Ji 2018); different from them, our method focuses on proposing 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 results with lower model complexity.

Proposed Method

In this section, we first revisit the channel attention module in SENet (Hu, Shen, and Sun 2018). Then, we make an empirical comparison to analyze the effect of dimensionality reduction and cross-channel interaction, which motivate us to propose our efficient channel attention (ECA) module. In addition, we introduce an adaptive kernel size selection for our ECA and finally show how to adopt it for deep CNNs.

Revisiting Channel Attention

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), respectively. As shown in Figure 2 (a), the weights of channel attention in SE block can be computed as

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

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

$$f_{\{\mathbf{W_1},\mathbf{W_2}\}}(\mathbf{y}) = \mathbf{W_2}\mathrm{ReLU}(\mathbf{W_1}\mathbf{y}), \tag{2}$$

where ReLU indicates the Rectified Linear Unit (Nair and Hinton 2010). To avoid too 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[1].

Table 1: Comparison of various channel attention modules using ResNet-50 as backbone model on ImageNet. Param. indicates number of parameters involved by each channel attention module. $\odot$ means dot product. GC and C1D indicate group convolutions and $1D$ convolution, respectively. k is kernel size of C1D.

Methods	Attention	Param.	Top-1	Top-5
Vanilla	N/A	0	75.30	92.20
SE	$\sigma(f_{\{\mathbf{W_1},\mathbf{W_2}\}}(\mathbf{y}))$	$2 * C^2/r$	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	$\sigma(\mathrm{GC}_{16}(\mathbf{y}))$	$C^2/16$	76.95	93.47
SE-GC2	$\sigma(\mathrm{GC}_{C/16}(\mathbf{y}))$	$16 \times C$	76.98	93.31
SE-GC3	$\sigma(\mathrm{GC}_{C/8}(\mathbf{y}))$	$8 \times C$	76.96	93.38
ECA-NS	$\sigma(\boldsymbol{\omega})$ with Eq. (4)	kC	77.35	93.61
ECA (Ours)	$\sigma(\mathrm{C1D}_k(\mathbf{y}))$	$k = 3$	**77.43**	**93.65**

Efficient Channel Attention (ECA) Module

In this subsection, we make an empirical comparison for deeper analysis on the effect of channel dimensionality reduction and cross-channel interaction on learning channel attention. According to these analyses, we propose our efficient channel attention (ECA) module.

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 1, 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 dimensionality reduction is more important than consideration of nonlinear channel dependencies. Additionally, SE-Var3 employing one single FC layer

[1]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.

performs better than two FC layers with dimensionality reduction in SE block. All of above results clearly demonstrate the importance of avoiding dimensionality reduction in attention module. Therefore, we develop our ECA module without channel dimensionality reduction.

Local Cross-Channel Interaction Although both of SE-Var2 and SE-Var3 keep channel dimension unchanged, the latter one achieves better performance. The main difference is that SE-Var3 captures cross-channel interaction while SE-Var2 does not. It indicates that cross-channel interaction is helpful to learn effective attention. However, SE-Var3 involves a mass of parameters, leading to too high model complexity. From perspective of efficient convolutions (Zhang et al. 2017; Xie et al. 2017), SE-Var2 can be regarded as a depth-wise separable convolution (Chollet 2017). Naturally, group convolutions as another kind of efficient convolutions also can be used to capture cross-channel interaction. Given a FC layer, group convolutions divide it into multiple groups and perform linear transform in each group independently. SE block with group convolutions (SE-GC) is written as

$$\sigma(\mathrm{GC}_G(\mathbf{y})) = \sigma(\mathbf{W}_G\mathbf{y}), \tag{3}$$

where $\mathbf{W}_G = \begin{bmatrix} \mathbf{W}_G^1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{W}_G^G \end{bmatrix}$ is a block diagonal matrix, whose number of parameters is C^2/G and G is number of groups. However, as shown in Table 1, SE-GC with varying groups bring no gain over SE-Var2, indicating that group convolution is not an effective scheme for exploiting cross-channel interaction. Meanwhile, excessive group convolutions will increase memory access cost (Ma et al. 2018).

By visualizing channel features $\mathbf{y}$, we find that they usually exhibit a certain local periodicity (please refer to Appendix A1 for details). Therefore, different from the above methods (i.e., depth-wise separable convolutions, group convolutions and FC layers), we aim at capturing local cross-channel interaction, i.e., only considering interaction between each channel and its k neighbors. Thus, the weight of y_i can be calculated as

$$\omega_i = \sigma\Big(\sum_{j=1}^{k} \alpha_i^j y_i^j\Big), \ y_i^j \in \Omega_i^k, \tag{4}$$

where Ω_i^k indicates the set of k adjacent channels of y_i. Clearly, Eq. (4) captures local cross-channel interaction, and such locality constraint avoids interaction across all channels, which allows high model efficiency. In this way, each channel attention module involves $k * C$ parameters. To further reduce model complexity and improve efficiency, we let all channels share the same leaning parameters, i.e.,

$$\omega_i = \sigma\Big(\sum_{j=1}^{k} \alpha^j y_i^j\Big), \ y_i^j \in \Omega_i^k. \tag{5}$$

As such, our efficient channel attention (ECA) module can be readily implemented by a fast $1D$ convolution with kernel size of k, i.e.,

$$\boldsymbol{\omega} = \sigma(\mathrm{C1D}_k(\mathbf{y})), \tag{6}$$

```
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.

where C1D indicates $1D$ convolution. As listed in Table 1, by introducing local cross-channel interaction, our ECA achieves similar results with SE-var3 and ECA-NS in Eq. (4) (i.e., ECA without shared parameters), while has much lower model complexity (it only involves k parameters). In Table 1, k is set to 3.

Adaptive Selection of Kernel Size k In our ECA module (Eq. (6)), kernel size k is a key parameter. Since $1D$ convolution is used to capture local cross-channel interaction, k determines the coverage of interaction, which may vary against convolution blocks with different channel numbers and various CNN architectures. Albeit k could be tuned manually, it will cost a lot of computing resources. It is reasonable that k is in connection with channel dimension C. In general, it is expected that larger size of channels favor long-range interaction while smaller size of channels prefer short-term interaction. In other words, there may exist a certain mapping ϕ between k and C:

$$C = \phi(k). \tag{7}$$

Here, the optimal formulation of mapping ϕ usually is unknown. However, based on above analysis, k is suggested to be nonlinear proportional to C, so the parameterized exponential function is a feasible choice. Meanwhile, for the classical kernel tricks (Boser, Guyon, and Vapnik 1992; Mika et al. 1998), exponential family functions (e.g., Gaussian) as kernel functions are most widely used to handle the issues of unknown mappings. Therefore, we approximate the mapping ϕ using an exponential function, i.e.,

$$C = \phi(k) \approx \exp(\gamma * k - b). \tag{8}$$

Furthermore, since channel dimension C (i.e., number of filters) usually is set to integral power of 2, we replace $\exp(\gamma * k - b)$ [2] by $2^{(\gamma * k - b)}$. Then, given channel dimension C, kernel size k can be adaptively determined by

$$k = \psi(C) = \left| \frac{log_2(C)}{\gamma} + \frac{b}{\gamma} \right|_{odd}, \tag{9}$$

[2] Note that $\exp(\gamma * k - b) \approx 2.72^{(\gamma * k - b)}$.

where $|t|_{odd}$ indicates the nearest odd number of t. In this paper, we set γ and b to 2 and 1, respectively. Clearly, the mapping function ψ makes larger size of channels have long-range interaction and vice versa.

ECA for Deep CNNs

Figure 2 compares our ECA module with the SE block. For adopting our ECA to deep CNNs, we exploit exactly the same configuration with SENet (Hu, Shen, and Sun 2018), and just replace SE block by our ECA module. The resulting networks are named by ECA-Net. Figure 3 gives PyTorch code of our ECA, which is easy to be reproduced.

Experiments

In this section, we evaluate the proposed method on large-scale image classification and object detection using ImageNet (Deng et al. 2009) and MS COCO (Lin et al. 2014), 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 module on object detection using Faster R-CNN (Ren et al. 2017) and Mask R-CNN (He et al. 2017).

Implementation Details

To evaluate our ECA-Net on ImageNet classification, we employ three widely used CNNs as backbone models, including ResNet-50 (He et al. 2016a), ResNet-101 (He et al. 2016a), ResNet-152 (He et al. 2016a) and MobileNetV2 (Sandler et al. 2018). For training ResNet-50, ResNet-101 and ResNet-152 with our ECA, we adopt exactly the same data augmentation and hyper-parameter settings in (He et al. 2016a; Hu, Shen, and Sun 2018). 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 (Sandler et al. 2018), 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[3].

We further evaluate our method on MS COCO using Faster R-CNN (Ren et al. 2017) and Mask R-CNN (He et al. 2017), where ResNet-50 and ResNet-101 along with FPN (Lin et al. 2017) are used as backbone models. We implement all detectors by using MMDetection toolkit (Chen et al. 2019) 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 are run on a PC equipped with four RTX 2080Ti GPUs and an Intel(R) Xeon Silver 4112 CPU@2.60GHz.

[3]https://github.com/pytorch/pytorch

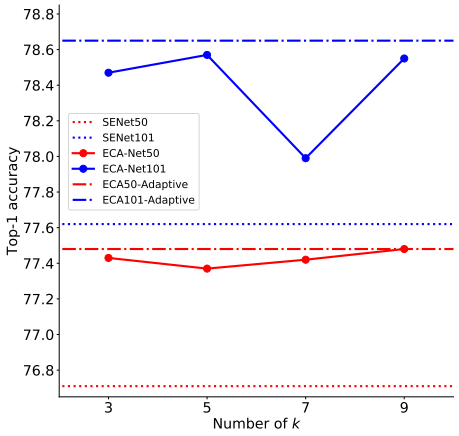


Figure 4: Results of our ECA module with various numbers of k using ResNet-50 and ResNet-101 as backbone models. Here, we also give the results of ECA module with adaptive selection of kernel size and compare with SENet as baseline.

Large-scale Image Classification on ImageNet-1K

Here, we first access the effect of kernel size on our ECA module and effectiveness of adaptive kernel size selection, then compare with state-of-the-art counterparts and CNN models using ResNet-50, ResNet-101, ResNet-152 and MobileNetV2.

Effect of Kernel Size and Adaptive Kernel Size Selection As shown in Eq. (6), 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 the proposed 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, so it may prefer to small kernel size. Furthermore, these results show that different deep CNNs have various optimal numbers of k, and k has a clear effect on performance of ECA-Net. Secondly, our adaptive selection of kernel size tries to find the optimal number of k for each convolution block, which can alleviate effect of depth of deep CNNs and avoid manual tuning of parameter k. Moreover,

Table 2: 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 public unavailable, we do not compare their running time. ♢: AA-Net is trained with Inception data augmentation and different setting of learning rates.

Method	Backbone Models	Param.	FLOPs	Training	Inference	Top-1	Top-5
ResNet (He et al. 2016a)	ResNet-50	24.37M	3.86G	1024 FPS	1855 FPS	75.20	92.52
SENet (Hu, Shen, and Sun 2018)		26.77M	3.87G	759 FPS	1620 FPS	76.71	93.38
CBAM (Woo et al. 2018)		26.77M	3.87G	472 FPS	1213 FPS	77.34	93.69
A^2-Nets (Chen et al. 2018)†		33.00M	6.50G	N/A	N/A	77.00	93.50
GSoP-Net1 (Gao et al. 2019)		28.05M	6.18G	596 FPS	1383 FPS	77.68	93.98
AA-Net (Bello et al. 2019)†,♢		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 (He et al. 2016a)	ResNet-101	42.49M	7.34G	386 FPS	1174 FPS	76.83	93.48
SENet (Hu, Shen, and Sun 2018)		47.01M	7.35G	367 FPS	1044 FPS	77.62	93.93
CBAM (Woo et al. 2018)		47.01M	7.35G	270 FPS	635 FPS	78.49	94.31
AA-Net (Bello et al. 2019)†,♢		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 (He et al. 2016a)	ResNet-152	57.40M	10.82G	281 FPS	815 FPS	77.58	93.66
SENet (Hu, Shen, and Sun 2018)		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 (Sandler et al. 2018)	MobileNetV2	3.34M	319.4M	711 FPS	2086 FPS	71.64	90.20
SENet		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**

it usually brings further improvement, demonstrating the effectiveness of adaptive selection of kernel size. Finally, ECA module with various numbers of k consistently outperform SE block, indicating that avoiding dimensionality reduction and local cross-channel interaction indeed exert positive effects on learning channel attention.

Comparisons using ResNet-50 Next, we compare our ECA module with several state-of-the-art attention methods using ResNet-50 on ImageNet, including SENet (Hu, Shen, and Sun 2018), CBAM (Woo et al. 2018), A^2-Nets (Chen et al. 2018), AA-Net (Bello et al. 2019) and GSoP-Net1 (Gao et al. 2019). The evaluation metrics concern 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 a fair comparison, we duplicate the results of all compared methods from their original papers, except training/inference speed. To test training/inference speed of various models, we employ publicly available models for the compared CNNs, and run them on the same computing platform. The results are given in Table 2, 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 terms of Top-1 accuracy. Comparing with state-of-the-art counterparts (i.e., SENet, CBAM, A^2-Nets, AA-Net and GSoP-Net1), ECA-Net obtains better or competitive performance while benefiting lower model complexity.

Comparisons using ResNet-101 Using ResNet-101 as backbone model, we compare our ECA-Net with SENet (Hu, Shen, and Sun 2018), CBAM (Woo et al. 2018) and AA-Net (Bello et al. 2019). From Table 2 we can see that ECA-Net outperforms the original ResNet-101 by 1.8% in terms of Top-1 accuracy 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.

Table 3: Comparisons with other state-of-the-art CNN models on ImageNet.

CNN Models	Param.	FLOPs	Top-1	Top-5
ResNet-152	57.40M	10.82G	77.58	93.66
SENet-152	63.68M	10.85G	78.43	94.27
ResNet-200	74.45M	14.10G	78.20	94.00
ResNeXt-101	46.66M	7.53G	78.80	94.40
DenseNet-264	28.78M	5.15G	77.85	93.78
ECA-Net50 (Ours)	24.37M	3.86G	77.48	93.68
ECA-Net101 (Ours)	42.49M	7.35G	78.65	94.34

Comparisons using ResNet-152 Using ResNet-101 as backbone model, we compare our ECA-Net with SENet (Hu, Shen, and Sun 2018). From Table 2 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 while outperforming SENet by 0.5% in terms of Top-1 accuracy 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.

Table 4: Object detection results of different methods on COCO val2017.

Methods	Detectors	Param.	GFLOPs	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-50	Faster R-CNN	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)		**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	Mask R-CNN	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**
+ 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**

Table 5: Instance segmentation results of different methods using Mask R-CNN on COCO val2017.

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
+ 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**

Comparisons using MobileNetV2 Besides ResNet architectures, we also verify the effectiveness of our ECA module on lightweight CNN architectures. To this end, we employ MobileNetV2 (Sandler et al. 2018) 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 2 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. All above results demonstrate the efficiency and effectiveness of our ECA module in deep CNNs again.

Comparisons with Other CNN Models At the end of this part, we compare our ECA-Net with other state-of-the-art CNN models, including ResNet-152 (He et al. 2016a), SENet-152 (Hu, Shen, and Sun 2018), ResNet-200 (He et al. 2016b), ResNeXt (Xie et al. 2017) and DenseNet-264 (Huang et al. 2017). These CNN models have deeper and wider architectures, and their results all are copied from the original papers. As listed in Table 3, our ECA-Net50 is comparable to ResNet-152 while ECA-Net101 outperforms SENet-152 and 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, 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.

Object Detection on MS COCO

In this subsection, we evaluate our ECA-Net on object detection task using Faster R-CNN (Ren et al. 2017) and Mask R-CNN (He et al. 2017). Here, we compare our ECA-Net with the original ResNet and SENet. All CNN models are first pre-trained on ImageNet, and then are transferred to MS COCO by fine-tuning.

Comparisons using Faster R-CNN Using Faster R-CNN as the basic detector, we employ ResNets of 50 and 101 layers along with FPN (Lin et al. 2017) as backbone models. As shown in Table 4, 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. Furthermore, our ECA module has lower model complexity than SE block. It is worth mentioning that our ECA module achieves more gains for small objects, which are usually harder to be detected.

Comparisons using Mask R-CNN We further exploit Mask R-CNN to verify the effectiveness of our ECA-Net on object detection task. As listed in Table 4, our ECA module is superior to the original ResNet by 1.8% and 1.9% in terms of AP under the settings of 50 and 101 layers, respectively. Meanwhile, ECA module achieves 0.3% and 0.6% gains over SE block using ResNet-50 and ResNet-101, respectively. The results in Table 4 demonstrate that our ECA module can be well generalized to object detection and is more suitable for detecting small objects.

Instance Segmentation on MS COCO

Finally, we give instance segmentation results of our ECA module using Mask R-CNN on MS COCO. As compared in Table 5, ECA module achieves notable gain over the original ResNet while performing better than SE block with less model complexity. These results verify our ECA module has good generalization ability to various tasks.

Conclusion

In this paper, we focus on learning channel attention for deep CNNs with low model complexity. To this end, we propose a novel efficient channel attention (ECA) module, which generates channel attention through a fast $1D$ convolution, whose kernel size can be adaptively determined by a function 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 adopt our ECA module to more CNN architectures (e.g., ResNeXt and Inception (Szegedy et al. 2016)) and further investigate the interaction between ECA and spatial attention module.

Appendix A1. Visualization of Global Average Pooling of Convolution Activations

Here, we visualize the results of global average pooling of convolution activations, which are fed to attention modules for learning channel weights. Specifically, we first train ECA-Net50 on the training set of ImageNet. Then, we randomly select some images from ImageNet validation set. Given a selected image, we first get it through ECA-Net50 and compute the global average pooling of activations from different convolution layers. The selected images are illustrated in left side of Figure 6 and we visualize the values of global average pooling of activations computed from conv_2_3, conv_3_2, conv_3_4, conv_4_3, conv_4_6 and conv_5_3, which are indicated by GAP_2_3, GAP_3_2, GAP_3_4, GAP_4_3, GAP_4_6 and GAP_5_3, respectively. Here, conv_2_3 indicates 3-th convolution layer of 2-th stage. As shown in Figure 6, we can observe that different images have similar trend in the same convolution layer, while these trends usually exhibit a certain local periodicity. Some of them are indicated by red rectangular boxes. This phenomenon may suggest that we can capture channel interaction in a local manner.

Appendix A2. Visualization of Weights Learned by ECA Modules and SE Blocks

To further analyze the effect of our ECA module on learning channel attention, we visualize the weights learned by ECA modules and compare with SE blocks. Here, we employ ResNet-50 as backbone model, and illustrate weights of different convolution blocks. Specifically, we randomly sample four classes from the ImageNet, which are *hammerhead shark*, *ambulance*, *medicine chest* and *butternut squash*, respectively. Some example images are illustrated in Figure 5. After training the networks, for all images of each class collected from ImageNet validation, we compute the channel weights of convolution blocks on average. Figure 7 visualizes the channel weights of conv_i_j, where i indicates i-th stage and j is j-th convolution block in i-th stage. Besides the visualization results of four random sampled classes, we also give the distribution of the average weights across $1K$ classes as reference. The channel weights learned by ECA modules and SE blocks are illustrated in bottom and top of each row, respectively.

Figure 5: Example images of four random sampled classes on ImageNet, including *hammerhead shark*, *ambulance*, *medicine chest* and *butternut squash*.

From Figure 7 we have the following observations. Firstly, for both ECA modules and SE blocks, the distributions of channel weights for different classes are very similar at the earlier layers (i.e., ones from conv_2_1 to conv_3_4), which may be caused by that the earlier layers aim at capturing the basic elements (e.g., boundaries and corners) (Zeiler and Fergus 2014). These features are almost similar for different classes. Such phenomenon also was described in the extended version of (Hu, Shen, and Sun 2018)[4]. Secondly, for the channel weights of different classes learned by SE blocks, most of them tend to be the same (i.e., 0.5) in conv_4_2 $\sim$ conv_4_5 while the differences among various classes are not obvious. On the contrary, the weights learned by ECA modules are clearly different across various channels and classes. Since convolution blocks in 4-th stage prefer to learn semantic information, so the weights learned by ECA modules can better distinguish different classes. Finally, convolution blocks in the final stage (i.e., conv_5_1, conv_5_2 and conv_5_3) capture high-level semantic features and they are more class-specific. Obviously, the weights learned by ECA modules are more class-specific than ones learned by SE blocks. Above results clearly demonstrate that the weights learned by our ECA modules have better discriminative ability.

[4]https://arxiv.org/abs/1709.01507

References

Bello, I.; Zoph, B.; Vaswani, A.; Shlens, J.; and Le, Q. V. 2019. Attention augmented convolutional networks. *arXiv:1904.09925*.

Boser, B. E.; Guyon, I.; and Vapnik, V. 1992. A training algorithm for optimal margin classifiers. In *COLT*, 144–152.

Chen, Y.; Kalantidis, Y.; Li, J.; Yan, S.; and Feng, J. 2018. A^2-Nets: Double attention networks. In *NIPS*.

Chen, K.; Wang, J.; Pang, J.; Cao, Y.; Xiong, Y.; Li, X.; Sun, S.; Feng, W.; Liu, Z.; Xu, J.; Zhang, Z.; Cheng, D.; Zhu, C.; Cheng, T.; Zhao, Q.; Li, B.; Lu, X.; Zhu, R.; Wu, Y.; Dai, J.; Wang, J.; Shi, J.; Ouyang, W.; Loy, C. C.; and Lin, D. 2019. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv:1906.07155*.

Chollet, F. 2017. Xception: Deep learning with depthwise separable convolutions. In *CVPR*.

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

Fu, J.; Liu, J.; Tian, H.; Li, Y.; Bao, Y.; Fang, Z.; and Lu, H. 2019. Dual attention network for scene segmentation. In *CVPR*.

Gao, Z.; Xie, J.; Wang, Q.; and Li, P. 2019. Global second-order pooling convolutional networks. In *CVPR*.

Gao, H.; Wang, Z.; and Ji, S. 2018. Channelnets: Compact and efficient convolutional neural networks via channel-wise convolutions. In *NeurIPS*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep residual learning for image recognition. In *CVPR*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Identity mappings in deep residual networks. In *ECCV*.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. B. 2017. Mask R-CNN. In *ICCV*, 2980–2988.

Hu, J.; Shen, L.; Albanie, S.; Sun, G.; and Vedaldi, A. 2018. Gather-excite: Exploiting feature context in convolutional neural networks. In *NeurIPS*.

Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-excitation networks. In *CVPR*.

Huang, G.; Liu, Z.; van der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *CVPR*.

Huang, Z.; Wang, X.; Huang, L.; Huang, C.; Wei, Y.; and Liu, W. 2019. CCNet: Criss-cross attention for semantic segmentation. In *ICCV*.

Ioannou, Y.; Robertson, D.; Cipolla, R.; and Criminisi, A. 2017. Deep roots: Improving cnn efficiency with hierarchical filter groups. In *CVPR*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet classification with deep convolutional neural networks. In *NIPS*.

Li, P.; Xie, J.; Wang, Q.; and Zuo, W. 2017a. Is second-order information helpful for large-scale visual recognition? In *ICCV*.

Li, Y.; Wang, N.; Liu, J.; and Hou, X. 2017b. Factorized bilinear models for image recognition. In *ICCV*.

Li, X.; Zhong, Z.; Wu, J.; Yang, Y.; Lin, Z.; and Liu, H. 2019. Expectation-maximization attention networks for semantic segmentation. In *ICCV*.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft COCO: Common objects in context. In *ECCV*.

Lin, T.; Dollár, P.; Girshick, R. B.; He, K.; Hariharan, B.; and Belongie, S. J. 2017. Feature pyramid networks for object detection. In *CVPR*, 936–944.

Ma, N.; Zhang, X.; Zheng, H.; and Sun, J. 2018. Shufflenet V2: practical guidelines for efficient CNN architecture design. In *ECCV*.

Mika, S.; Schölkopf, B.; Smola, A. J.; Müller, K.; Scholz, M.; and Rätsch, G. 1998. Kernel PCA and de-noising in feature spaces. In *NIPS*, 536–542.

Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*.

Ren, S.; He, K.; Girshick, R. B.; and Sun, J. 2017. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* 39(6):1137–1149.

Roy, A. G.; Navab, N.; and Wachinger, C. 2019. Recalibrating fully convolutional networks with spatial and channel "squeeze and excitation" blocks. *IEEE Trans. Med. Imaging* 38(2):540–549.

Sandler, M.; Howard, A. G.; Zhu, M.; Zhmoginov, A.; and Chen, L. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*.

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

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *CVPR*.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *CVPR*.

Wang, X.; Girshick, R.; Gupta, A.; and He, K. 2018. Non-local neural networks. In *CVPR*.

Woo, C.; Park, J.; Lee, J.-Y.; and Kweon, I. S. 2018. CBAM: Convolutional block attention module. In *ECCV*.

Xie, S.; Girshick, R. B.; Dollár, P.; Tu, Z.; and He, K. 2017. Aggregated residual transformations for deep neural networks. In *CVPR*.

Zeiler, M. D., and Fergus, R. 2014. Visualizing and understanding convolutional networks. In *ECCV*, 818–833.

Zhang, T.; Qi, G.-J.; Xiao, B.; and Wang, J. 2017. Interleaved group convolutions. In *ICCV*.

Zhang, X.; Zhou, X.; Lin, M.; and Sun, J. 2018. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*.

Zhang, D. 2018. Clcnet: Improving the efficiency of convolutional neural network using channel local convolutions. In *CVPR*.

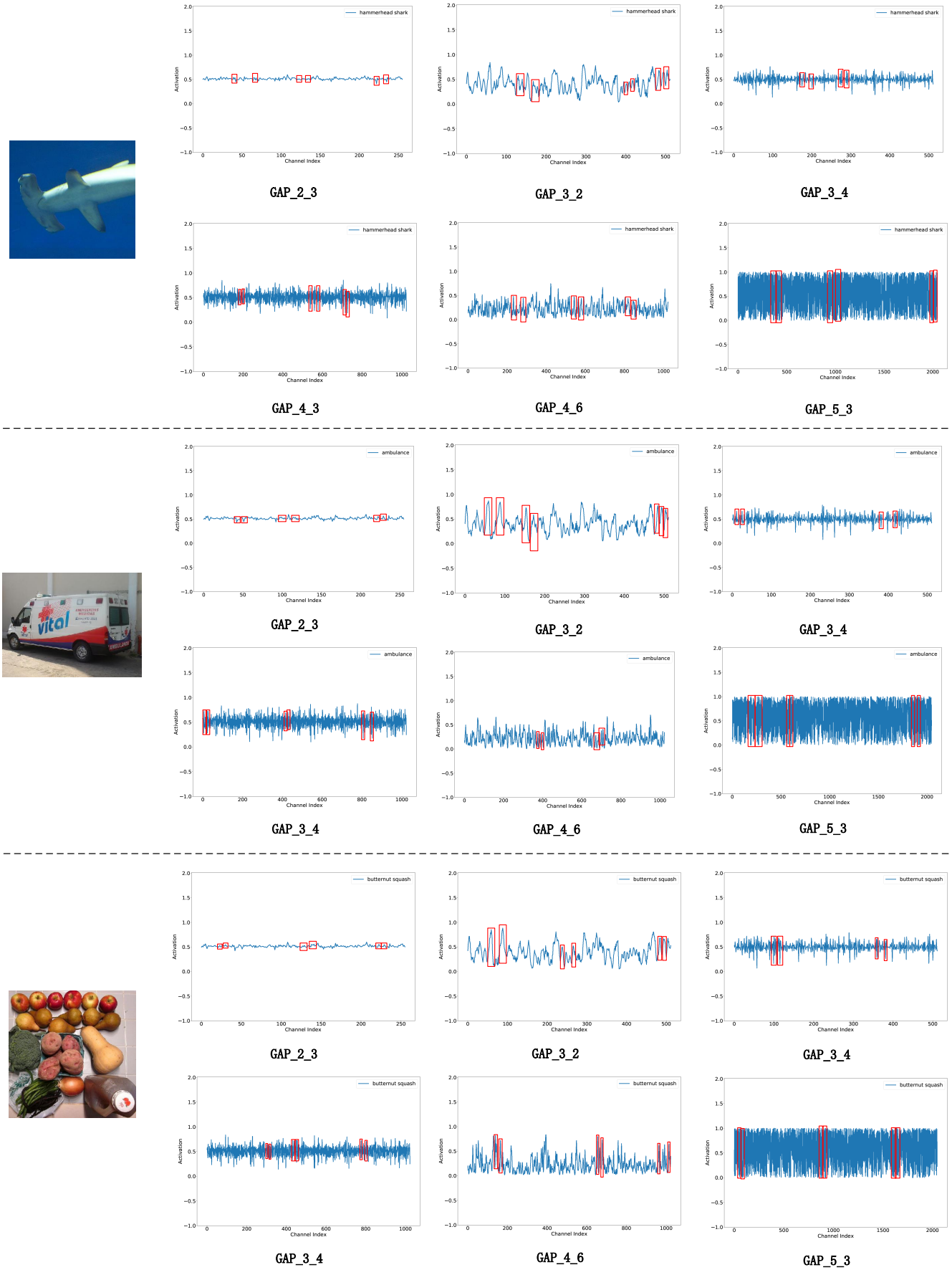
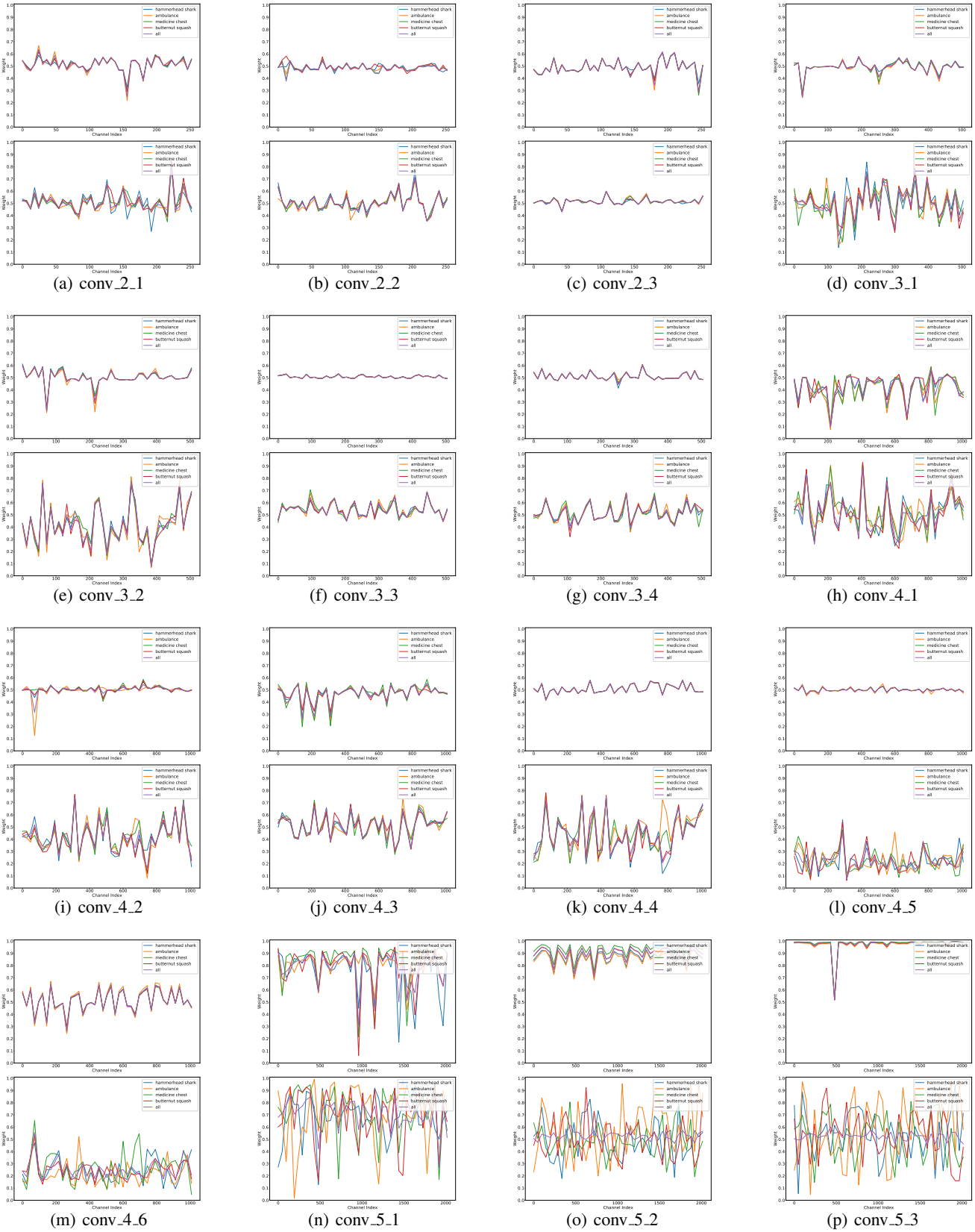


Figure 6: Visualization of the values of global average pooling on activations in different convolution layers, where different images have similar trend in the same convolution layer. Meanwhile, these trends present a certain kind of local periodicities, and some of them are indicated by red rectangular boxes. Better view with zooming in.



(a) conv_2_1 (b) conv_2_2 (c) conv_2_3 (d) conv_3_1

(e) conv_3_2 (f) conv_3_3 (g) conv_3_4 (h) conv_4_1

(i) conv_4_2 (j) conv_4_3 (k) conv_4_4 (l) conv_4_5

(m) conv_4_6 (n) conv_5_1 (o) conv_5_2 (p) conv_5_3

Figure 7: Visualization the channel weights of conv_i_j, where i indicate i-th stage and j is j-th convolution block in i-th stage. The channel weights learned by ECA modules and SE blocks are illustrated in bottom and top of each row, respectively. Better view with zooming in.