ECA-Net

论文的研究内容

论文原文：

“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 crosschannel 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.”

总结：
作者认为SE block的两个FC层之间的降维是不利于channel attention的权重学习的，因此提出了一个基于1D Conv实现不降维的方法。

原理

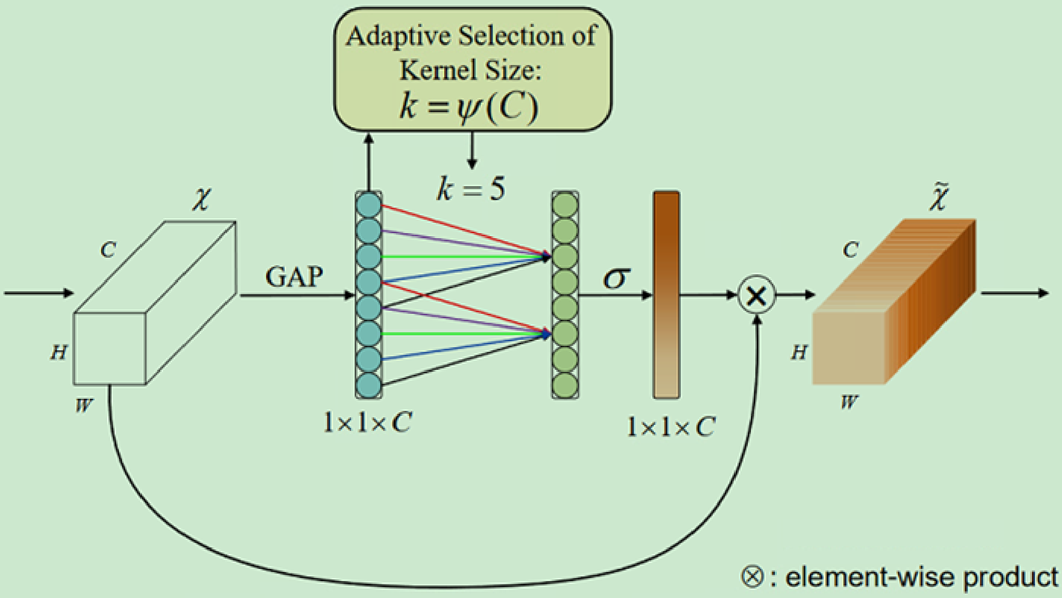


Figure 2. Diagram of our efficient channel attention (ECA) module. Given the aggregated features obtained by global average pooling (GAP), ECA generates channel weights by performing a fast $1D$ convolution of size k, where k is adaptively determined via a mapping of channel dimension C.

ECA首先沿通道方向进行avg_pool，将特征图的维度从b×c×h×w变成b×c×1×1。

ECA学习通道间的依存关系的策略：每次学习k个相邻通道之间的依存关系，而不是一次性学习所有通道之间的依存关系，并且每一次学习共享权重。例如，C个通道，标号从0开始，直到C。第一次学习0到k号通道的依存关系；第二次学习1到1+k号通道的依存关系......如此反复，截止到C号通道。

论文使用带状矩阵，来描述这一过程，如下：

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

为了方便计算，采用了上述的每次k个通道的依存关系学习 共享 权重，即：图中的$\omega\ldots$均为同一组权重。因此该过程就可以使用一维卷积来表示，卷积后跟sigmoid激活函数，如下：

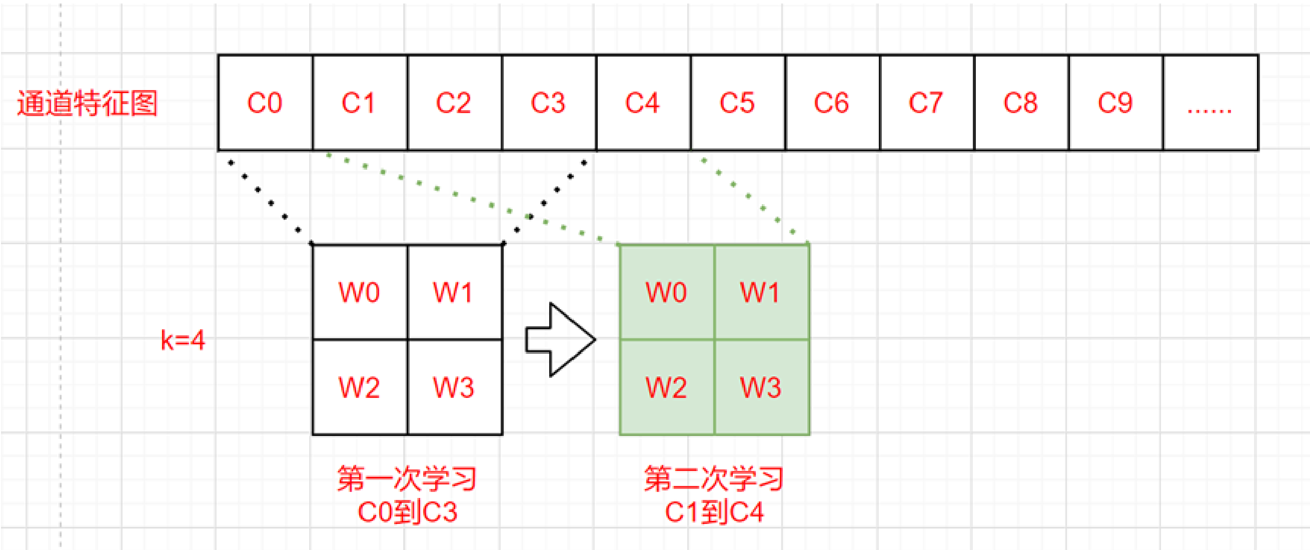
$$\omega_i = \sigma\left(\sum_{j=1}^{k} w^j y_i^j\right), \quad y_i^j \in \Omega_i^k. \tag{8}$$

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

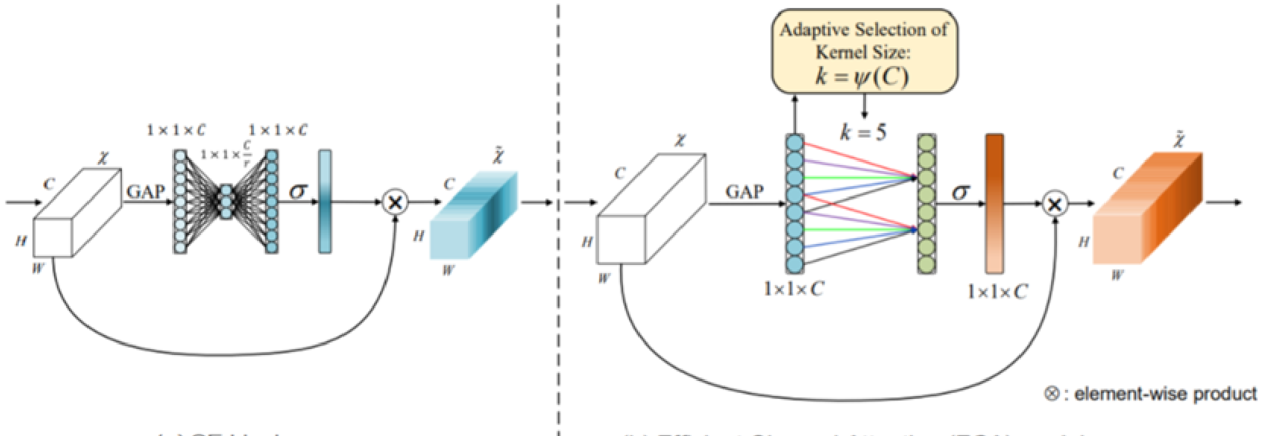
$$\boldsymbol{\omega} = \sigma(\mathrm{C1D}_k(\mathbf{y})), \tag{9}$$

where C1D indicates $1D$ convolution. Here, the method in

可以想象成一个一维卷积核（要学习的权重）在通道特征图中滑动，图解如下：



SE模块和ECA模块的结构差别：



(a) SE block
(b) Efficient Channel Attention (ECA) module

pytorch代码

PYTHON

```python
class ECA(nn.Module):
    def __init__(self, c1, c2, k_size=3):
        super(ECA, self).__init__()
        self.avg_pool = nn.AdaptiveAvgPool2d(1)
        self.conv = nn.Conv1d(1, 1,
kernel_size=k_size, padding=(k_size-1)//2,
bias=False)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        out = self.avg_pool(x)
        out = out.squeeze(-1)
        # 理解成矩阵转置，变成行向量
        out = out.transpose(-1, -2)
        out = self.conv(out)
        out = out.transpose(-1, -2)
        out = out.unsqueeze(-1)
        out = self.sigmoid(out)
        out = out.expand_as(x)

        return out * x
```