浅谈视觉Attention

李沛霖

什么是注意力机制

食物

天敌

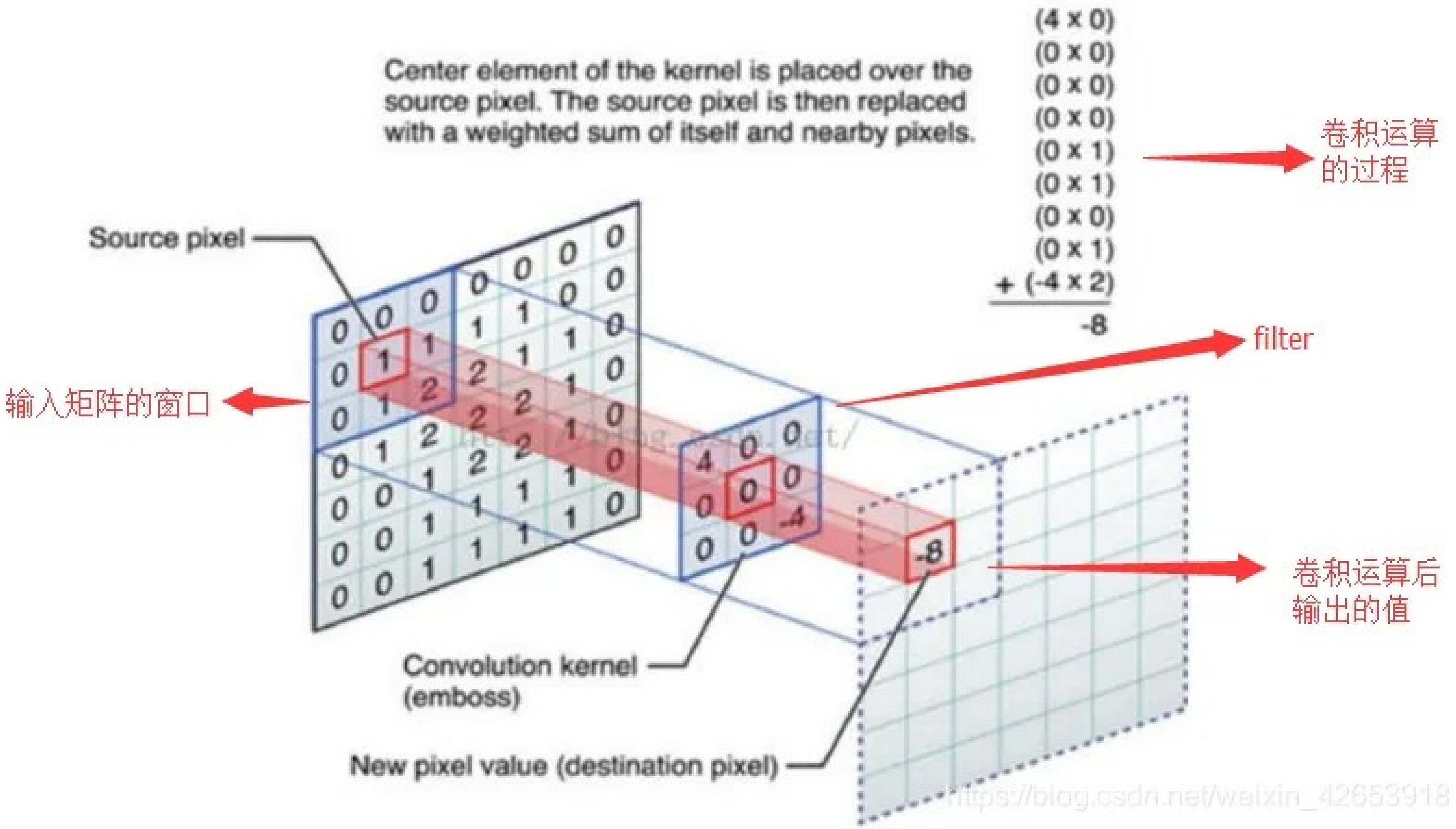
不随意线索

随意线索

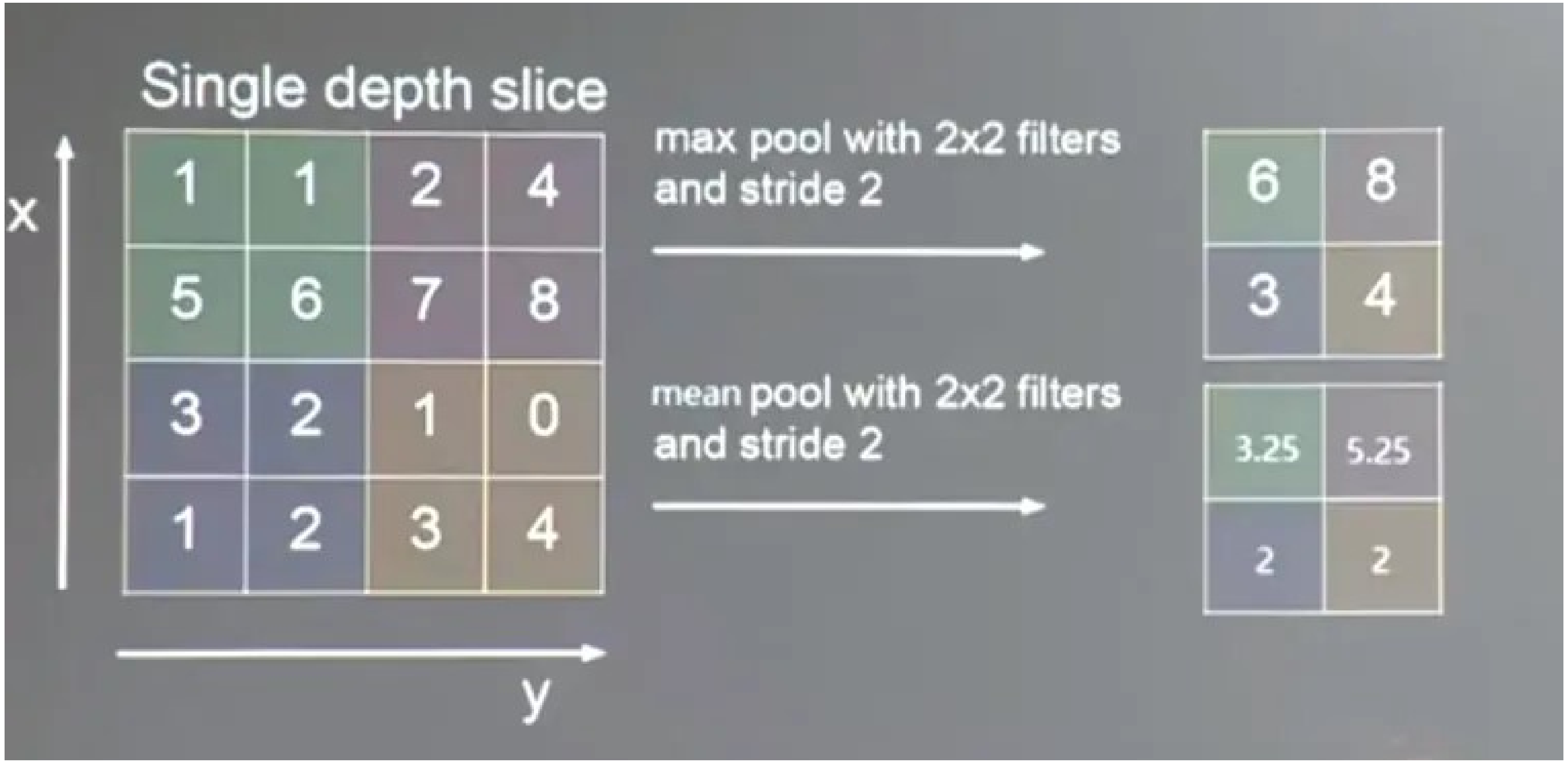
不随意线索

随意线索

神经网络中的不随意线索



神经网络中的不随意线索



什么是注意力机制

食物

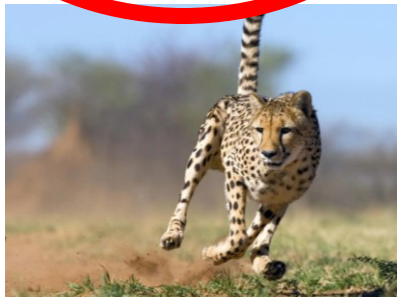
天敌

不随意线索

随意线索（query）

（key，value）

不随意线索



随意线索

Attention的雏形

给定数据 $(x_i, y_i), i = 1, \ldots, n$ ，对某一确定的x，要预测对应的y值

最简单的办法：$f(x) = \frac{1}{n}\sum_i y_i$

更好的方案：Nadaraya-Watson核回归

Attention的雏形

Nadaraya-Watson核回归：

$$f(x)=\sum_{i=1}^{n}\frac{K(x-x_i)}{\sum_{j=1}^{n}K(x-x_j)}y_i$$

query

key

value

$$K(u)=\frac{1}{\sqrt{2\pi}}\exp(-\frac{u^2}{2})$$ （高斯核）

Attention的雏形

Details of the softmax classifier

$$p(y|x) = \frac{\exp(W_y.x)}{\sum_{c=1}^{C}\exp(W_c.x)}$$

Nadaraya-Watson核回归:

非参数化

$$f(x) = \sum_{i=1}^{n} \frac{\exp(-\frac{1}{2}(x - x_i)^2)}{\sum_{j=1}^{n}\exp(-\frac{1}{2}(x - x_j)^2)} y_i$$

$$= \sum_{i=1}^{n} \text{softmax}(-\frac{1}{2}(x - x_i)^2) y_i$$

Attention的雏形

Details of the softmax classifier

$$p(y|x) = \frac{\exp(W_y . x)}{\sum_{c=1}^{C} \exp(W_c . x)}$$

Nadaraya-Watson核回归（改）：

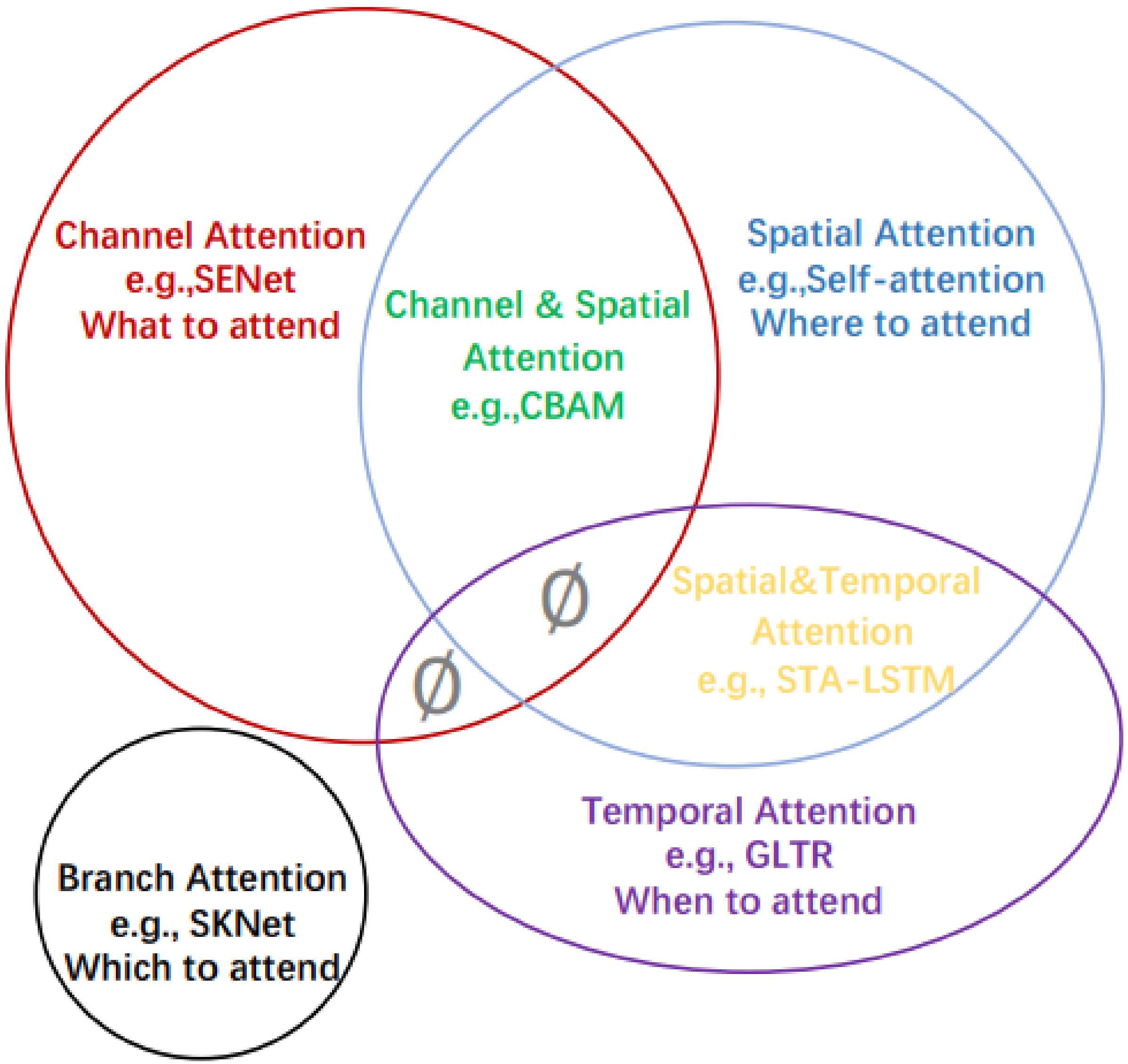
$\boxed{\text{参数化}}$

$$f(x) = \sum_{i=1}^{n} \frac{\exp(-\frac{1}{2}((x - x_i)w)^2)}{\sum_{j=1}^{n} \exp(-\frac{1}{2}(x - x_j)^2)} y_i$$

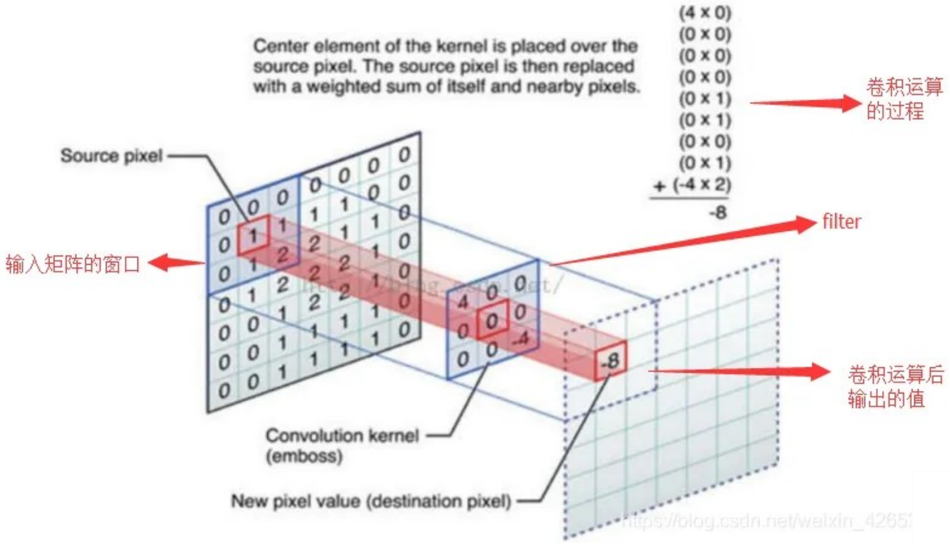
$$= \sum_{i=1}^{n} \text{softmax}(-\frac{1}{2}((x - x_i)w)^2) y_i$$

视觉领域的Attention

六大类，四个基本类：通道注意力、空间注意力、时间注意力和分支通道注意力，以及两个混合组合类别：通道与空间注意和空间与时间注意。



视觉领域的Attention



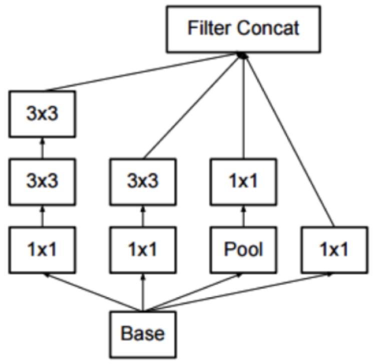


Figure 5. Inception modules where each 5 × 5 convolution is replaced by two 3 × 3 convolution, as suggested by principle 3 of Section 2.

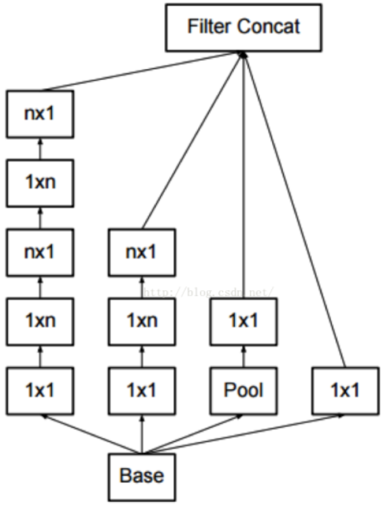


Figure 6. Inception modules after the factorization of the $n \times n$ convolutions. In our proposed architecture, we chose $n = 7$ for the 17×17 grid. (The filter sizes are picked using principle 3)

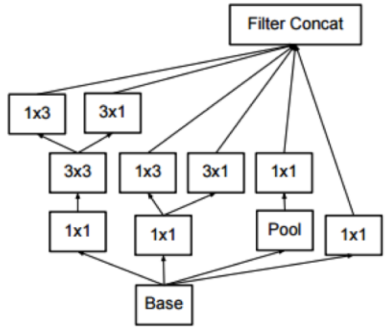
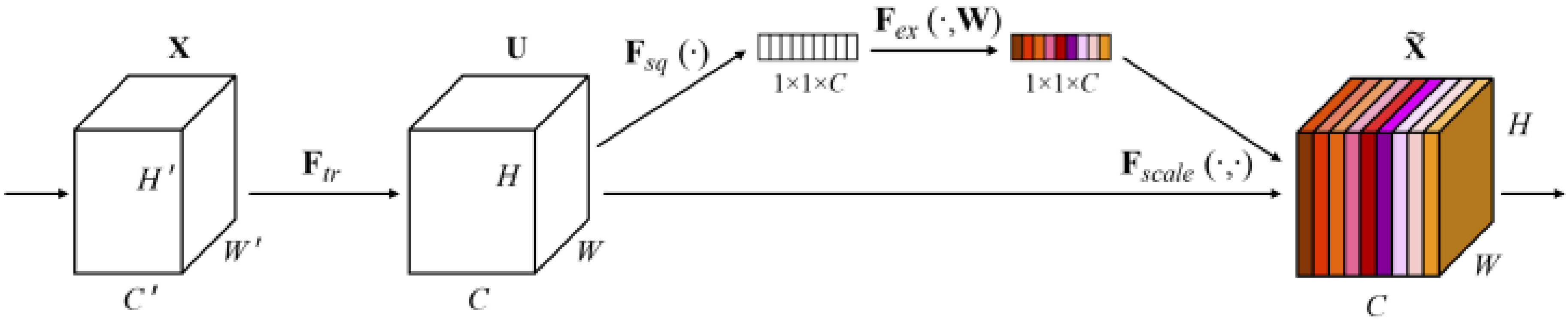
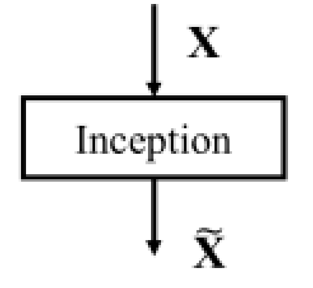


Figure 7. Inception modules with expanded the filter bank outputs. This architecture is used on the coarsest (8 × 8) grids to promote high dimensional representations, as suggested by principle 2 of Section 2. We are using this solution only on the coarsest grid, since that is the place where producing high dimensional sparse representation is the most critical as the ratio of local processing (by 1 × 1 convolutions) is increased compared to the spatial aggregation.

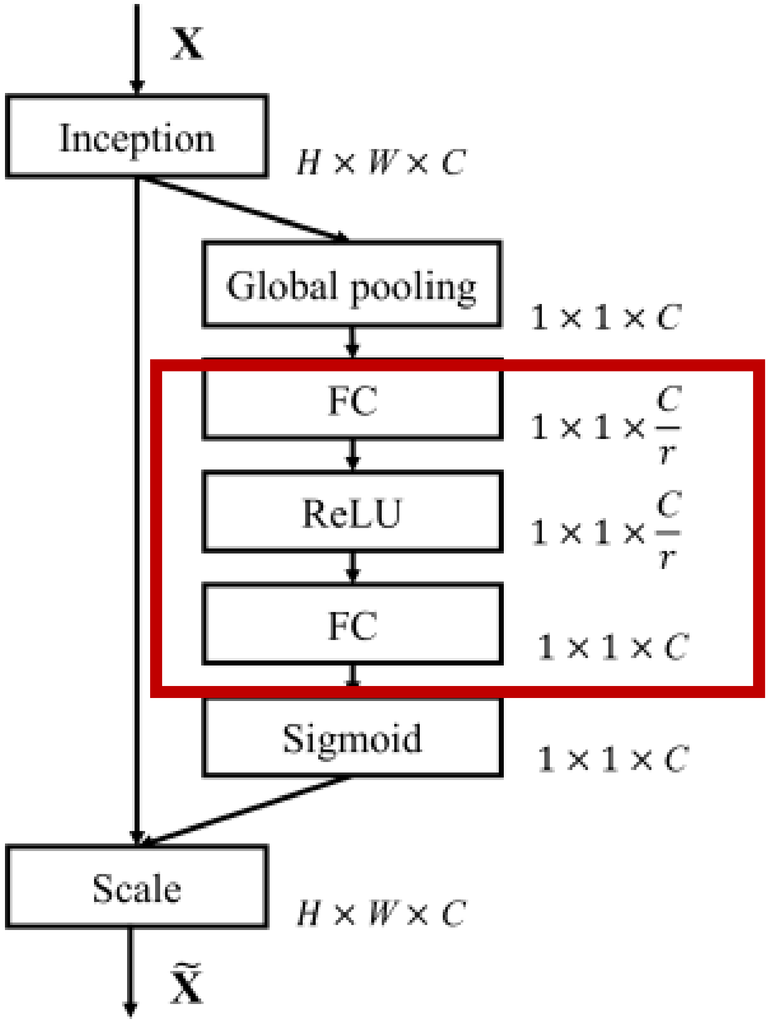
Channel Attention：SENet



Channel Attention：SENet



Inception Module



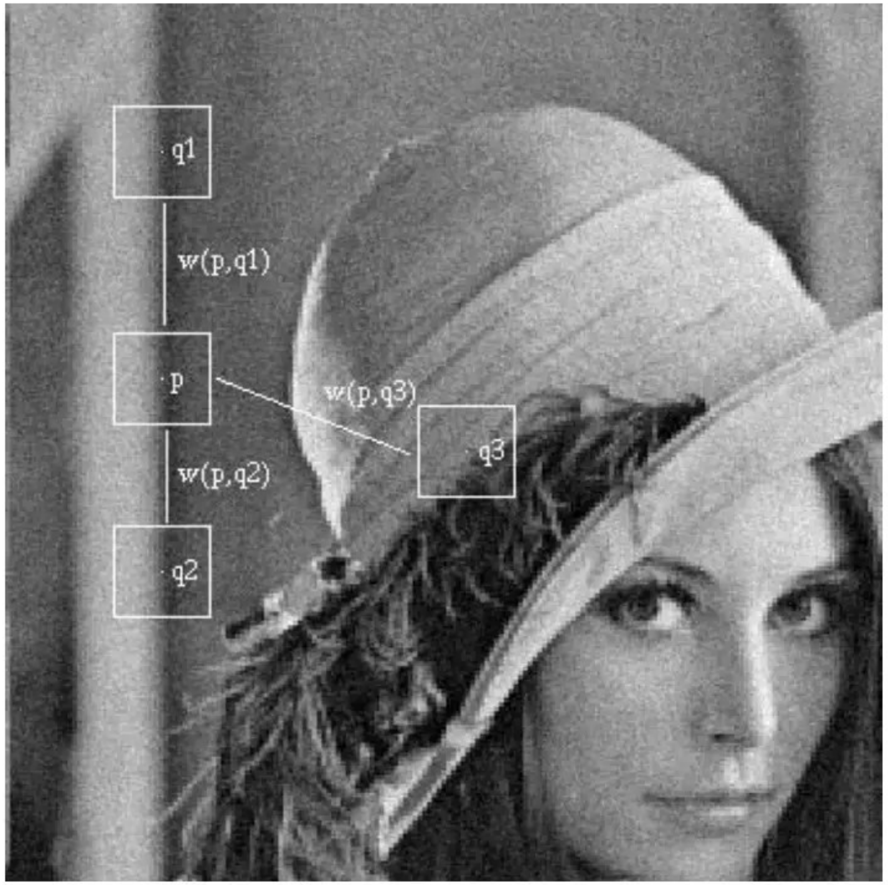
SE-Inception Module

好处：

1）具有更多的非线性，可以更好地拟合通道间复杂的相关性；

2）极大地减少了参数量和计算量

Spatial Attention：Non-local Network



Non local means filter

Spatial Attention：Non-local Network

（1）定义通用non-local操作：

$$\mathbf{y}_i = \frac{1}{\mathcal{C}(\mathbf{x})} \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j) g(\mathbf{x}_j).$$

（2）pairwise函数选择：

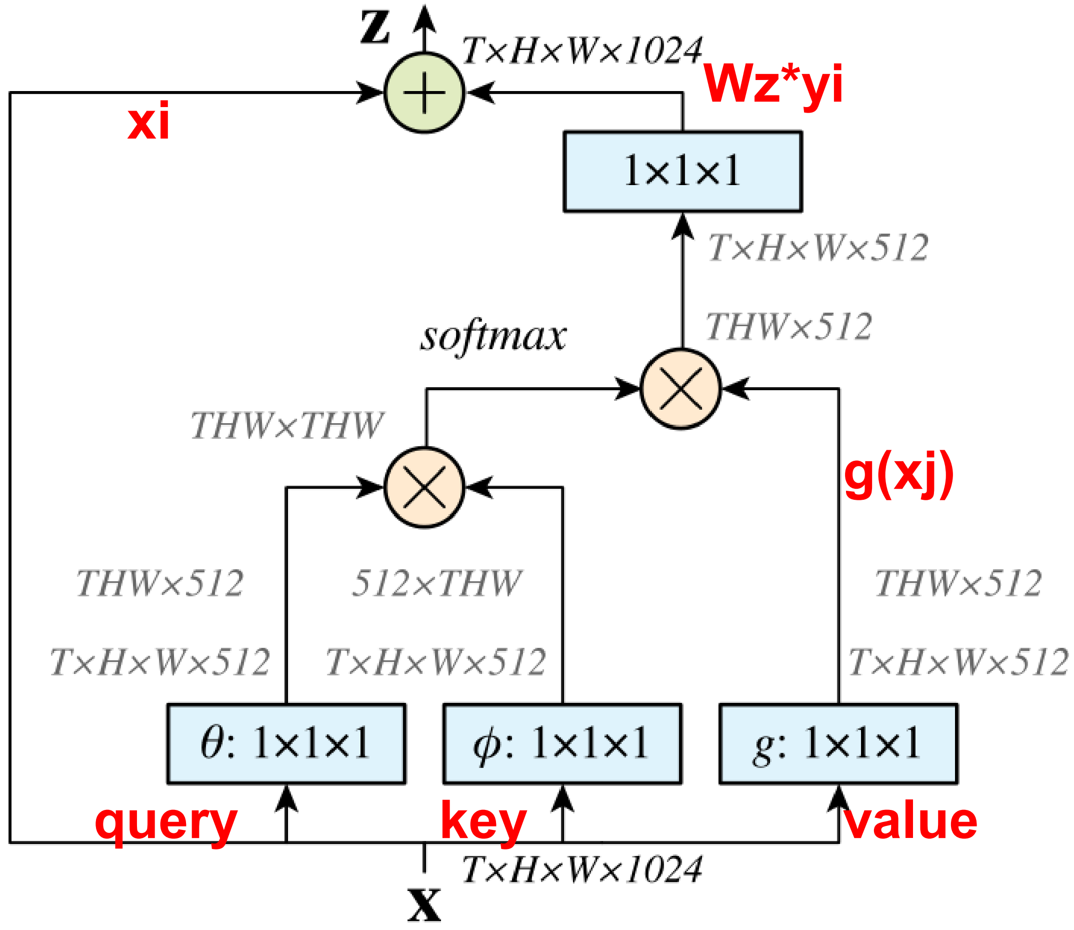
$$f(\mathbf{x}_i, \mathbf{x}_j) = e^{\mathbf{x}_i^T \mathbf{x}_j} \qquad \mathcal{C}(\mathbf{x}) = \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j).$$

$$f(\mathbf{x}_i, \mathbf{x}_j) = e^{\theta(\mathbf{x}_i)^T \phi(\mathbf{x}_j)} \qquad \mathcal{C}(\mathbf{x}) = \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j).$$

$$f(\mathbf{x}_i, \mathbf{x}_j) = \theta(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \qquad \mathcal{C}(\mathbf{x}) = N$$

$$f(\mathbf{x}_i, \mathbf{x}_j) = \mathrm{ReLU}(\mathbf{w}_f^T [\theta(\mathbf{x}_i), \ \phi(\mathbf{x}_j)]) \qquad \mathcal{C}(\mathbf{x}) = N$$

Spatial Attention：Non-local Network

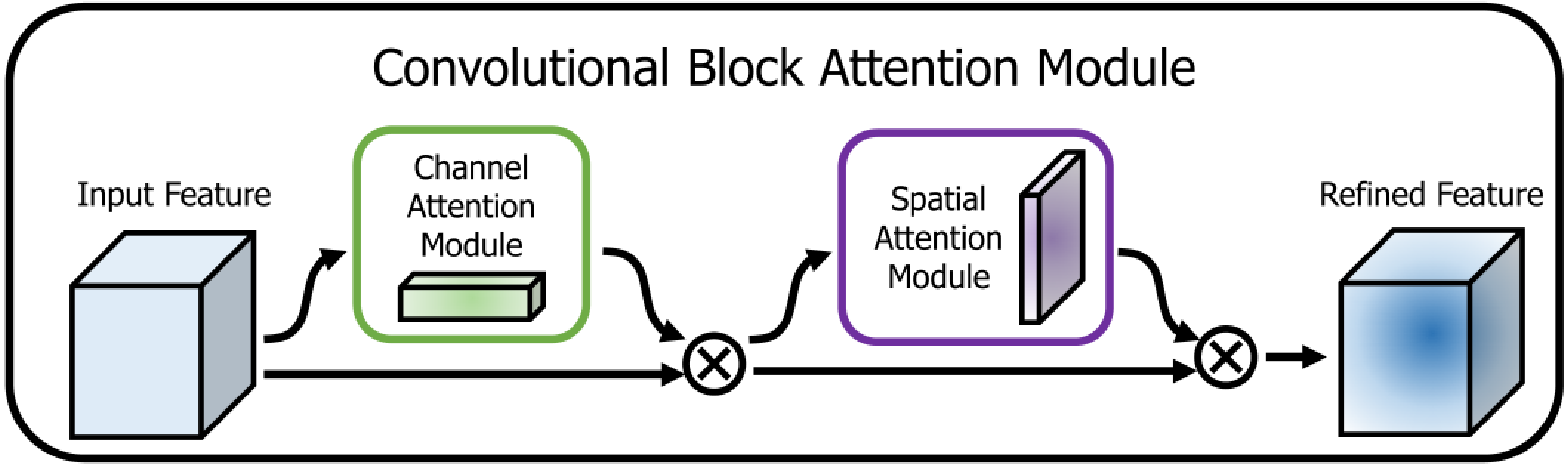


$$z_i = W_z y_i + x_i$$

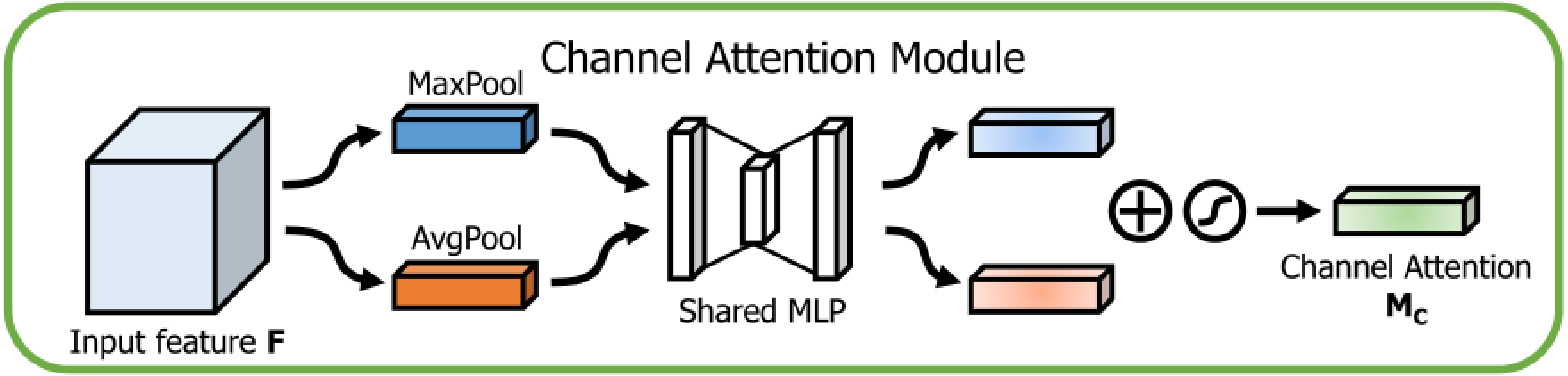
$$y_i = \mathrm{softmax}(\theta(x_i)^T \phi(x_j)) g(x_j)$$

$$= \frac{1}{\sum_{\forall j} e^{\theta(x_i)^T \phi(x_j)}} e^{\theta(x_i)^T \phi(x_j)} W_g x_j$$

Chn&Spa Attention：CBAM



Chn&Spa Attention：CBAM

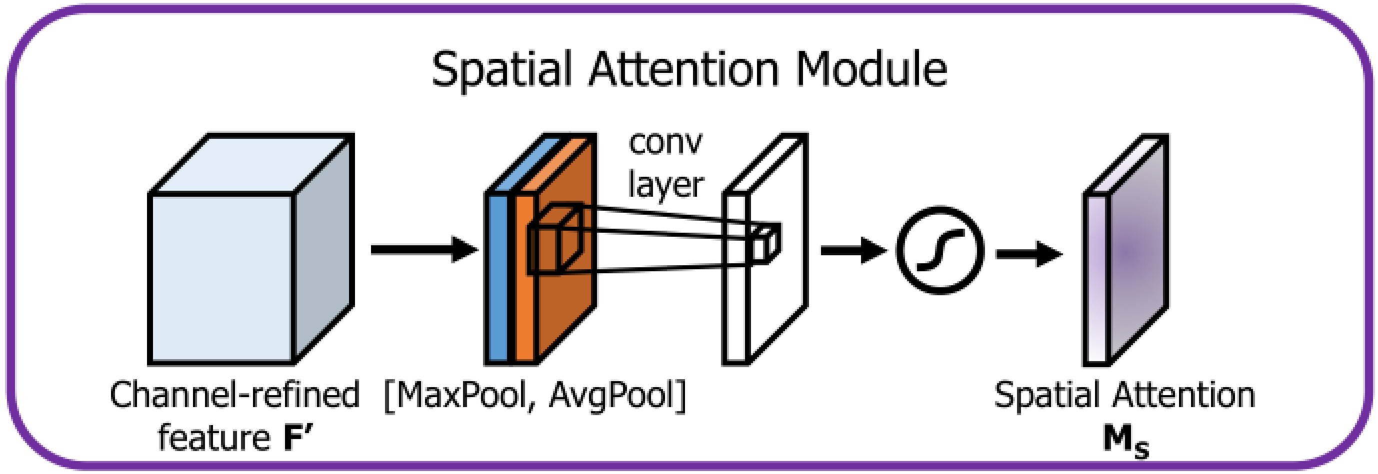


$$F : C * H * W$$

$$\begin{aligned} M_c(F) &= \sigma(MLP(AvgPool(F)) + MLP(MaxPool(F))) \\ &= \sigma(W_1(W_0(F^c_{avg})) + W_1(W_0(F^c_{max}))) \end{aligned}$$

, $\mathbf{W_0} \in \mathbb{R}^{C/r \times C}$, and $\mathbf{W_1} \in \mathbb{R}^{C \times C/r}$

Chn&Spa Attention：CBAM



$$\begin{aligned}\mathbf{M_s}(\mathbf{F}) &= \sigma(f^{7\times7}([AvgPool(\mathbf{F}); MaxPool(\mathbf{F})])) \\ &= \sigma(f^{7\times7}([\mathbf{F^s_{avg}}; \mathbf{F^s_{max}}])),\end{aligned}$$

σ代表sigmoid函数，f^{7*7}代表用尺寸为7*7的卷积核进行卷积运算。

Chn&Spa Attention：CBAM

```
class ChannelAttention(nn.Module):
    def __init__(self, in_planes, ratio=16):
        super(ChannelAttention, self).__init__()
        #平均池化
        self.avg_pool = nn.AdaptiveAvgPool2d(1)
        #最大池化
        self.max_pool = nn.AdaptiveMaxPool2d(1)

        #MLP  除以16是降维系数
        self.fc1   = nn.Conv2d(in_planes, in_planes // 16, 1, bias=
False) #kernel_size=1
        self.relu1 = nn.ReLU()
        self.fc2   = nn.Conv2d(in_planes // 16, in_planes, 1, bias=
False)

        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        avg_out = self.fc2(self.relu1(self.fc1(self.avg_pool(x))))
        max_out = self.fc2(self.relu1(self.fc1(self.max_pool(x))))
        #结果相加
        out = avg_out + max_out
        return self.sigmoid(out)
```

Chn&Spa Attention：CBAM

```
#空间注意力
class SpatialAttention(nn.Module):
    def __init__(self, kernel_size=7):
        super(SpatialAttention, self).__init__()
        #声明卷积核为 3 或 7
        assert kernel_size in (3, 7), 'kernel size must be 3 or 7'
        #进行相应的same padding填充
        padding = 3 if kernel_size == 7 else 1

        self.conv1 = nn.Conv2d(2, 1, kernel_size, padding=padding, bias
=False)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        avg_out = torch.mean(x, dim=1, keepdim=True)  #平均池化
        max_out, _ = torch.max(x, dim=1, keepdim=True) #最大池化
        #拼接操作
        x = torch.cat([avg_out, max_out], dim=1)
        x = self.conv1(x) #7x7卷积填充为3，输入通道为2，输出通道为1
        return self.sigmoid(x)
```

Chn&Spa Attention：CBAM

Description	Parameters	GFLOPs	Top-1 Error(%)	Top-5 Error(%)
ResNet50 (baseline)	25.56M	3.86	24.56	7.50
ResNet50 + AvgPool (SE [28])	25.92M	3.94	23.14	6.70
ResNet50 + MaxPool	25.92M	3.94	23.20	6.83
ResNet50 + AvgPool & MaxPool	25.92M	4.02	**22.80**	**6.52**

Description	Top-1 Error(%)	Top-5 Error(%)
ResNet50 + channel (SE [28])	23.14	6.70
ResNet50 + channel + spatial	**22.66**	**6.31**
ResNet50 + spatial + channel	22.78	6.42
ResNet50 + channel & spatial in parallel	22.95	6.59

Thanks.

Thanks