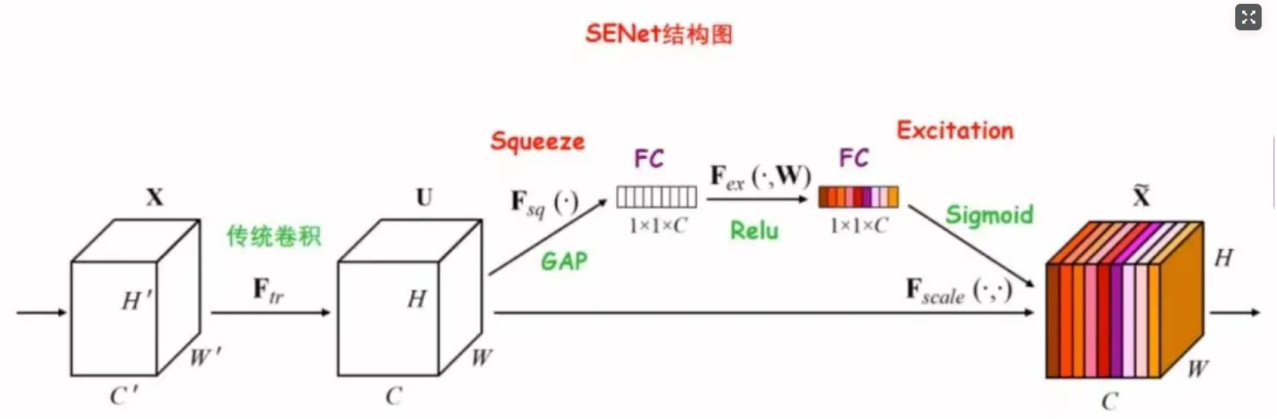
Squeeze-and-Excitation Networks

原理



输入x，经过一系列卷积操作后，得到特征图C，它的维度为b×c×h×w，通过如下3个操作：

1. 压缩操作（squeeze）

$$z_c = \mathbf{F}_{sq}(\mathbf{u}_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i,j). \qquad (2)$$

沿通道方向进行全局平均化，将h×w的二维特征变为一维，因此经压缩操作后，维度变化为b×c×1×1，每个1×1的实数某种程度上代表了当前通道的全局特征。

2. 激励操作（excitation）

$$\mathbf{s} = \mathbf{F}_{ex}(\mathbf{z}, \mathbf{W}) = \sigma(g(\mathbf{z}, \mathbf{W})) = \sigma(\mathbf{W}_2 \delta(\mathbf{W}_1 \mathbf{z})), \qquad (3)$$

where δ refers to the ReLU [63] function, $\mathbf{W}_1 \in \mathbb{R}^{\frac{C}{r} \times C}$ and $\mathbf{W}_2 \in \mathbb{R}^{C \times \frac{C}{r}}$. To limit model complexity and aid generalisation, we parameterise the gating mechanism by forming a bottleneck with two fully-connected (FC) layers around the non-linearity, i.e. a dimensionality-reduction layer with

得到Squeeze的1x1xC全局特征后，通过全连接层对每个通道的重要性进行预测。为了减少计算量，设置了压缩率ratio，第一个全连接层W1后跟ReLU，生成b×(c/ratio)×1×1；第二个全连接层W2后跟sigmoid，生成b×(c/ratio)×ratio×1×1（还原通道数）

3. 缩放操作（scale）
   得到excitation的b×c×1×1的通道权重后，通过放缩操作，得到b×c×h×w的注意力图。将它与输入x相乘，得到最终输出的特征图。

pytorch代码

PYTHON

```python
class SE(nn.Module):
    def __init__(self, c1, c2, r=16):
        super(SE, self).__init__()
        self.avgpool = nn.AdaptiveAvgPool2d(1)
        self.l1 = nn.Linear(c1, c1 // r, bias=False)
        self.relu = nn.ReLU(inplace=True)
        self.l2 = nn.Linear(c1 // r, c1, bias=False)
        self.sig = nn.Sigmoid()

    def forward(self, x):
        print(x.size())
        """
        b: batch size
        c: channels number
        _: width
        _: high

        可以这样理解:
        获取有多少个特征图，一个特征图有多少个通道，宽和
高
        """
        b, c, _, _ = x.size()

        # 每个通道进行全局平均化，最后结果只有一个数值
        # 有多少个通道，就有多少个数值
        y = self.avgpool(x).view(b, c)
        y = self.l1(y)
        y = self.relu(y)
        y = self.l2(y)
        y = self.sig(y)
        y = y.view(b, c, 1, 1)
```

```
        # y.expand_as(x)表示生成每一个通道的权重矩阵
        return x * y.expand_as(x)
```