東北大學秦皇島分校
Northeastern University at Qinhuangdao

CBAM:卷积块注意力模块

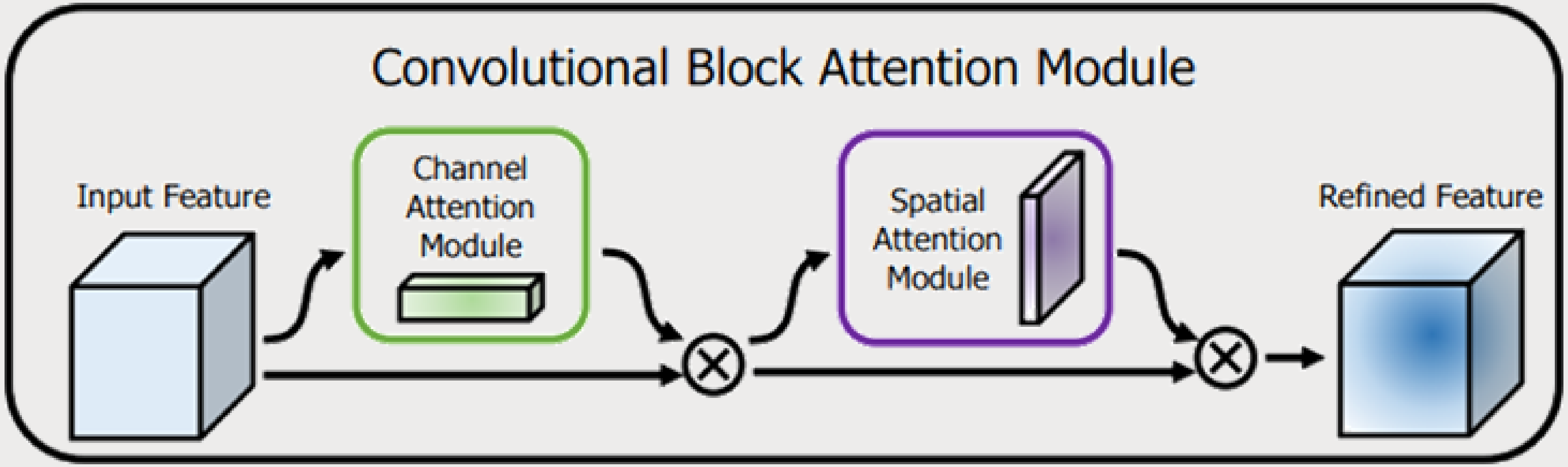
CBAM: Convolutional Block Attention Module

摘要

- 提出了卷积块注意模块(CBAM)，这是一种简单有效的卷积神经网络注意模块。在给定一个中间特征图的情况下，我们的模块沿着通道和空间两个单独的维度依次推断注意图，然后将注意图乘到输入特征图中进行自适应特征细化。因为CBAM是一个轻量级的通用模块，它可以无缝地集成到任何CNN架构中，损失可以忽略不计，并且可以与基本CNNs一起进行端到端培训。我们通过对ImageNet-1K、MS COCO检测和VOC 2007检测数据集的大量实验验证了CBAM的有效性。实验结果表明，不同的模型在分类和检测性能上都有一定的提高，说明了CBAM的广泛适用性。代码和模型将公开可用。
- 关键词：目标识别，注意力机制，封闭的卷积

论文地址：https://arxiv.org/abs/1807.06521

CBAM的框架图



特点:

- block轻量化，参数和计算量在加入神经网络时可忽略;
- 增加channel(通道)与spatial(空间)的分支，目的使各个分支可以学习“什么”和“在哪里”；
- 适用性广泛。

神经网络主要研究的三个方面：

- ➢Depth 深度
- ➢Width 宽度
- ➢Cardinality 基数

与深度有关研究：Vggnet, Resnet
与宽度有关研究：Googlenet
与基数有关研究：Xception, ResNeXt

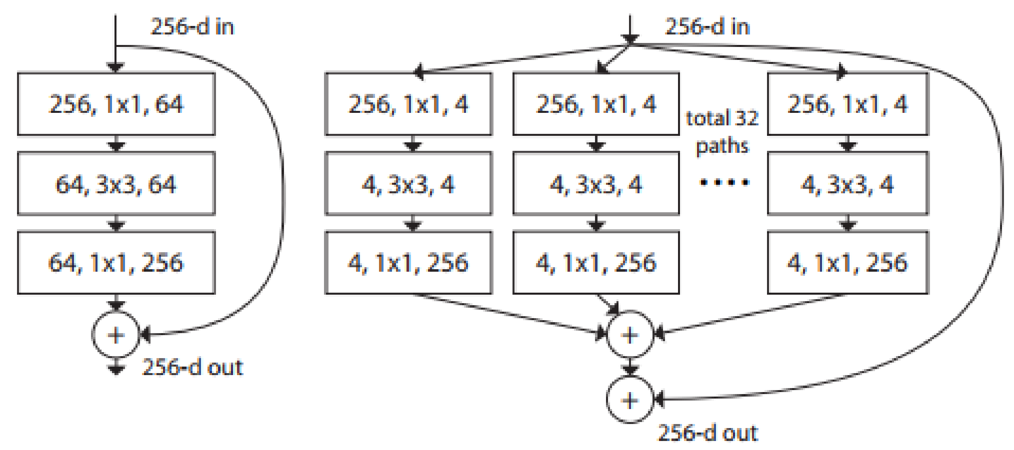
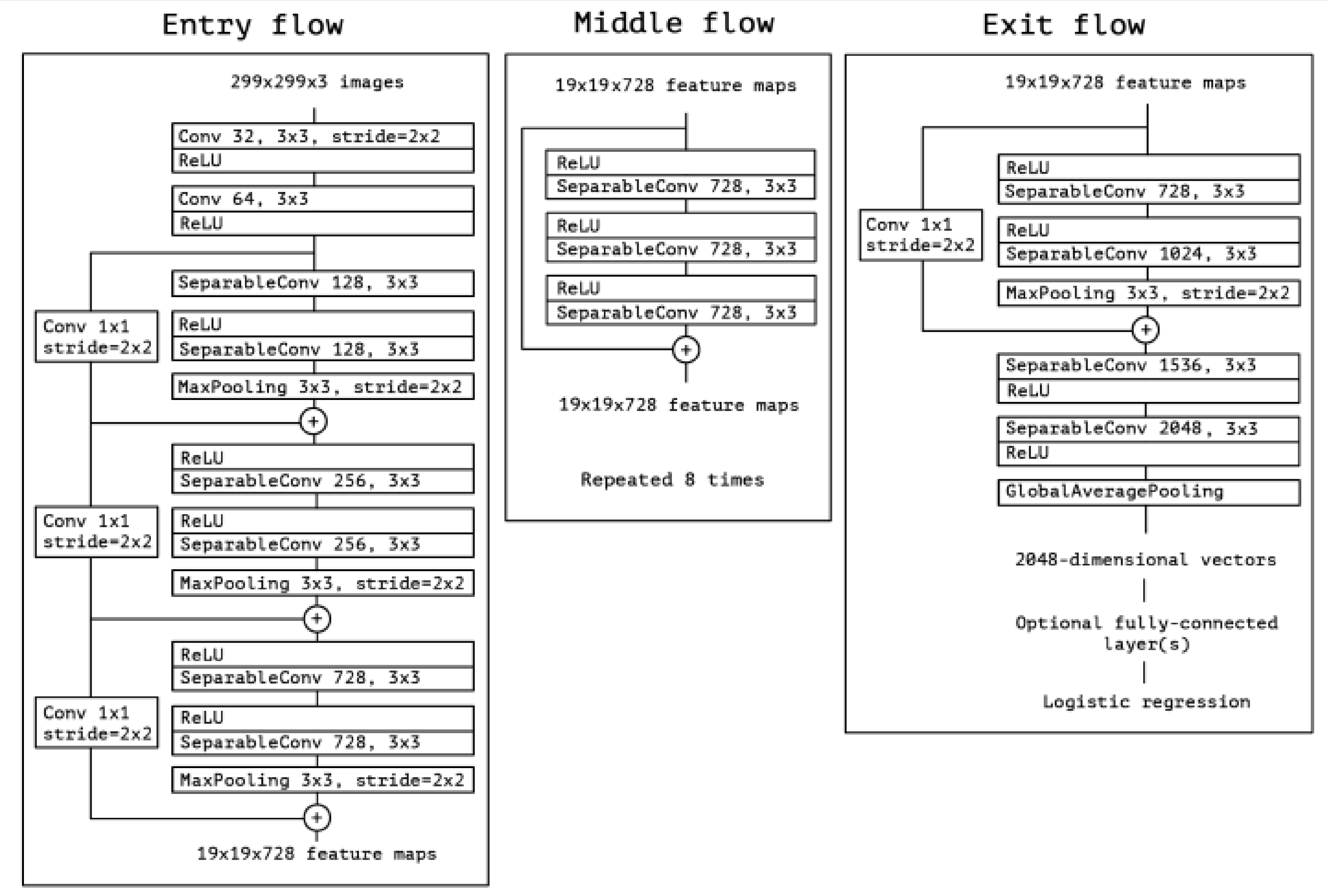


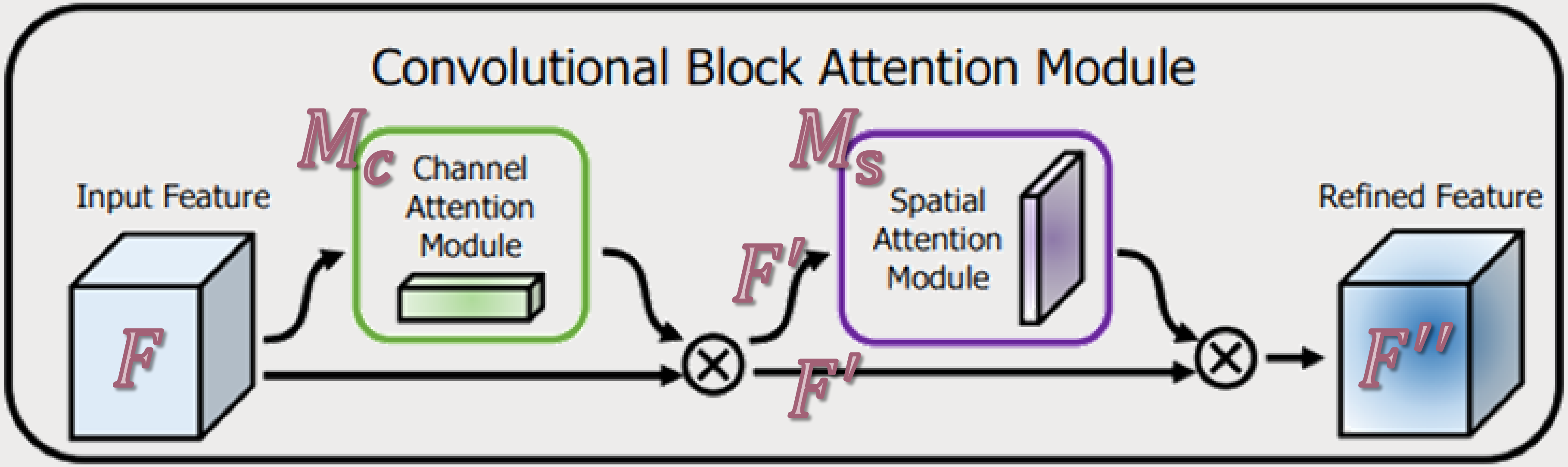
Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

ResNeXt结构



Xception网络结构

CBAM



公式：

$F \in \mathbb{R}^{C \times H \times W}$

$M_c \in \mathbb{R}^{C \times 1 \times 1}$

$M_s \in \mathbb{R}^{1 \times H \times W}$

$F' = M_c(F) \otimes F$

$F'' = M_s(F') \otimes F'$

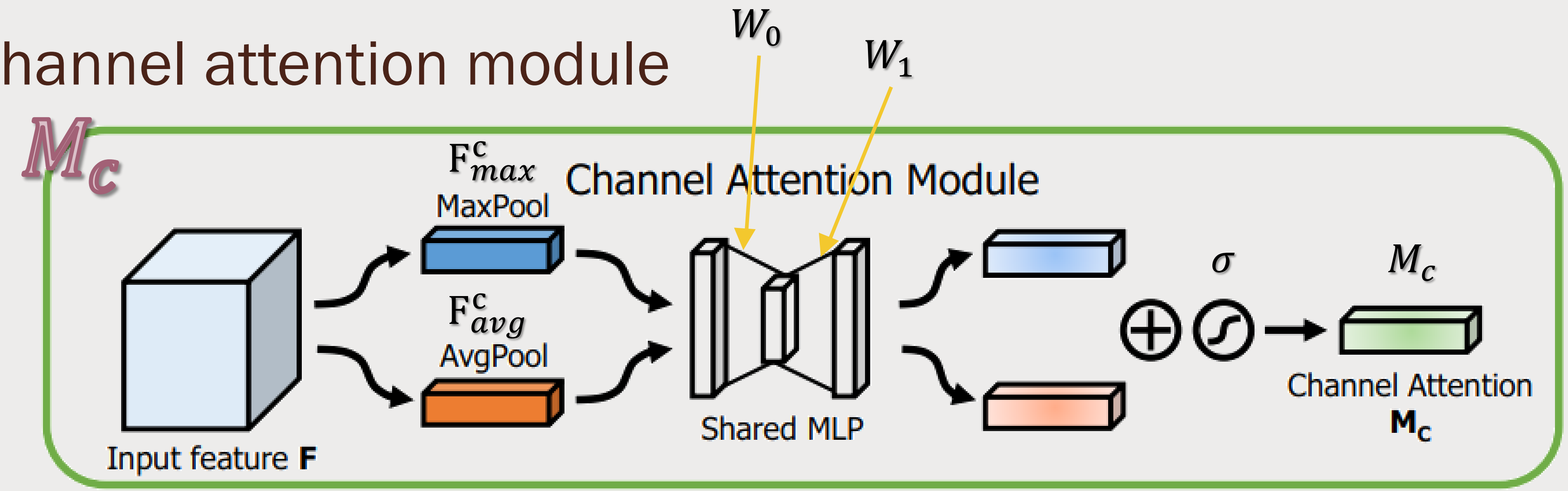
输入Input Feature应该是一个三维的张量；

$\otimes$ 代表非标乘法，相同索引数字相乘，而非矩阵式乘法；

经过通道注意力模型（Channel Attentionl Module），输出应为一维张量；

经过空间注意力模型（Spatial Attention Module），输出应为一个二维张量。

Channel attention module



- 输入特征是一个三维张量，经过最大池化与平均池化，将只存在channel，也就是变化为一维张量；
- 经最大池化与平均池化后的特征向量，通过同一个权重的，含有1个隐层的多层感知机（使用relu激活函数，**仅对隐层加**），再进行标量相加；
- 隐层的尺寸应设定为*C/r*，*C*为特征向量长度，*r*为减速比。
- *通过sigmoid函数激活，输出通道注意力。*

$$M_c(F) = \sigma\left(MLP\left(AvgPool(F)\right) + MLP\left(MaxPool(F)\right)\right)$$
$$= \sigma(W_1\left(W_0(F^c_{avg})\right) + W_1(W_0(F^c_{max})))$$

$M_c \in \mathbb{R}^{C\times1\times1}$
$W_0 \epsilon \mathbb{R}^{C/r\times C}$
$W_1 \epsilon \mathbb{R}^{C\times C/r}$

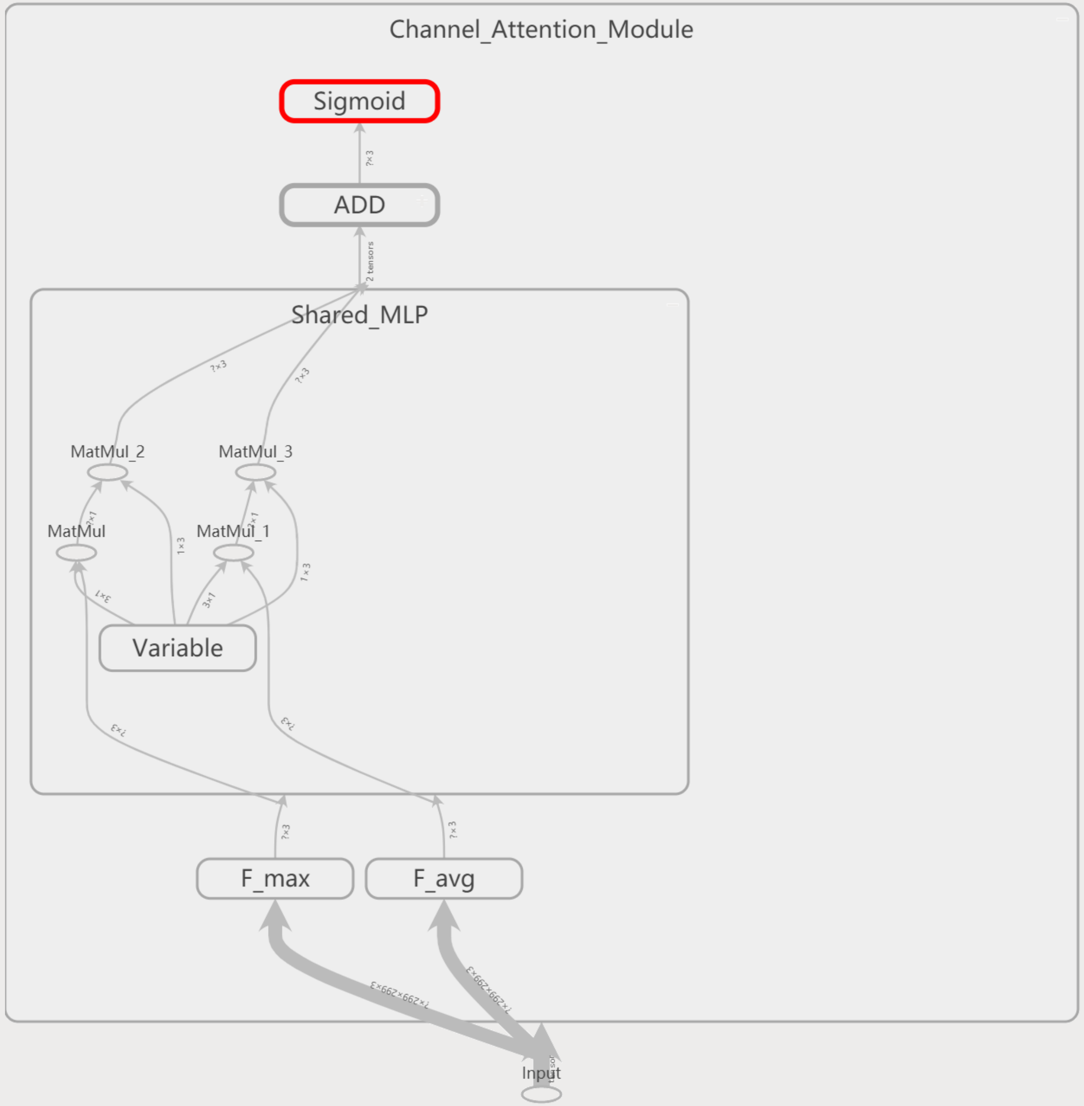
Channel attention module

参考代码:

```
def channel_attention_module(Input_feature, r):
    # image tensor shape: [batch, height, width, channel]
    with tf.name_scope('Channel_Attention_Module')
        tensor_shape = Input_feature.get_shape().as_list()
        height = tensor_shape[-3]
        width = tensor_shape[-2]
        channel = tensor_shape[-1]
        with tf.name_scope('F_max'):
            F_max = tf.nn.max_pool(Input_feature, ksize = [1,height,width,1], strides = [1,1,1,1], padding = 'VALID')
            F_max = tf.reshape(F_max, [-1, channel])
        with tf.name_scope('F_avg'):
            F_avg = tf.nn.avg_pool(Input_feature, ksize = [1,height,width,1], strides = [1,1,1,1], padding = 'VALID')
            F_avg = tf.reshape(F_avg, [-1, channel])
        with tf.name_scope('Shared_MLP'):
            with tf.name_scope('Variable'):
                W0 = weight_variable([channel, int(channel/r)], name='W0')
                W1 = weight_variable([int(channel/r), channel], name='W1')
            hidden_layer_m = tf.nn.relu(tf.matmul(F_max, W0))
            hidden_layer_a = tf.nn.relu(tf.matmul(F_avg, W0))
            layer_m = tf.matmul(hidden_layer_m, W1)
            layer_a = tf.matmul(hidden_layer_a, W1)
        with tf.name_scope('ADD'):
            channel_attention = layer_m + layer_a
        with tf.name_scope('Sigmoid'):
            channel_attention = tf.nn.sigmoid(channel_attention)
    return channel_attention
```

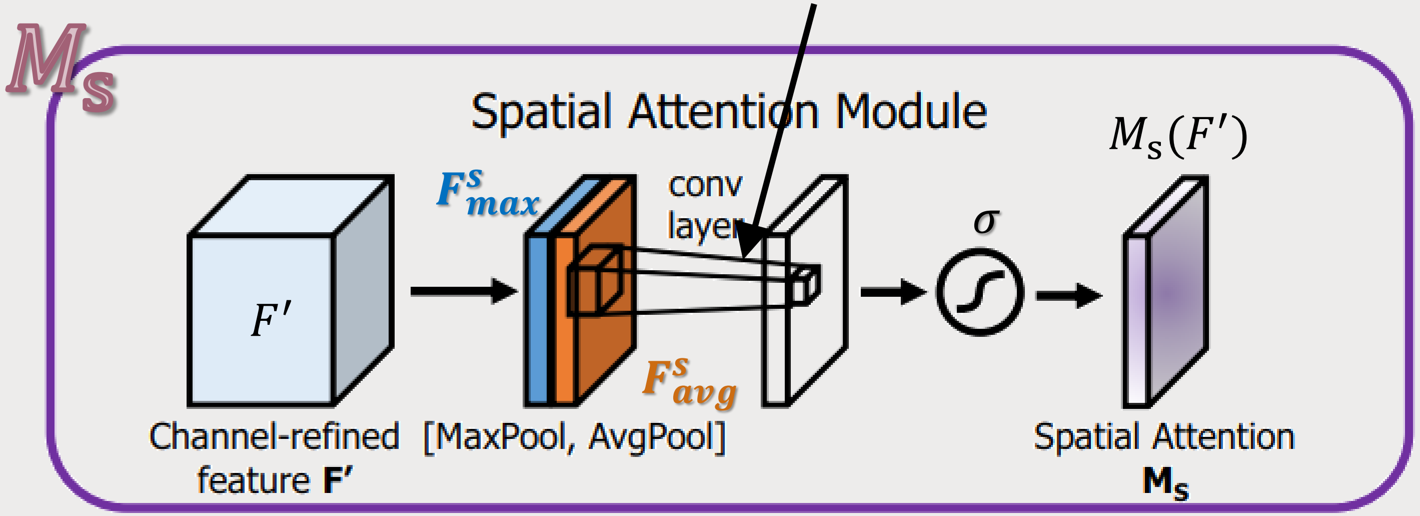
Channel attention module

Tensorboard可视化结果：



Spatial attention module

kernel size 7×7(SAME)



- 空间注意力模型集中于注意“哪里”；
- 平均池化与最大池化是沿着通道轴方向进行的，最终变二维张量；
- 拼接到一起的特征块通过一个7×7×2卷积核，大小不发生变化，通道变为1；
- 通过sigmoid函数激活，输出空间注意力。

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

$$M_s(F) \in \mathbb{R}^{H\times W}$$
$$F^s_{avg} \epsilon \mathbb{R}^{1\times H\times W}$$
$$F^s_{\max} \epsilon \mathbb{R}^{1\times H\times W}$$

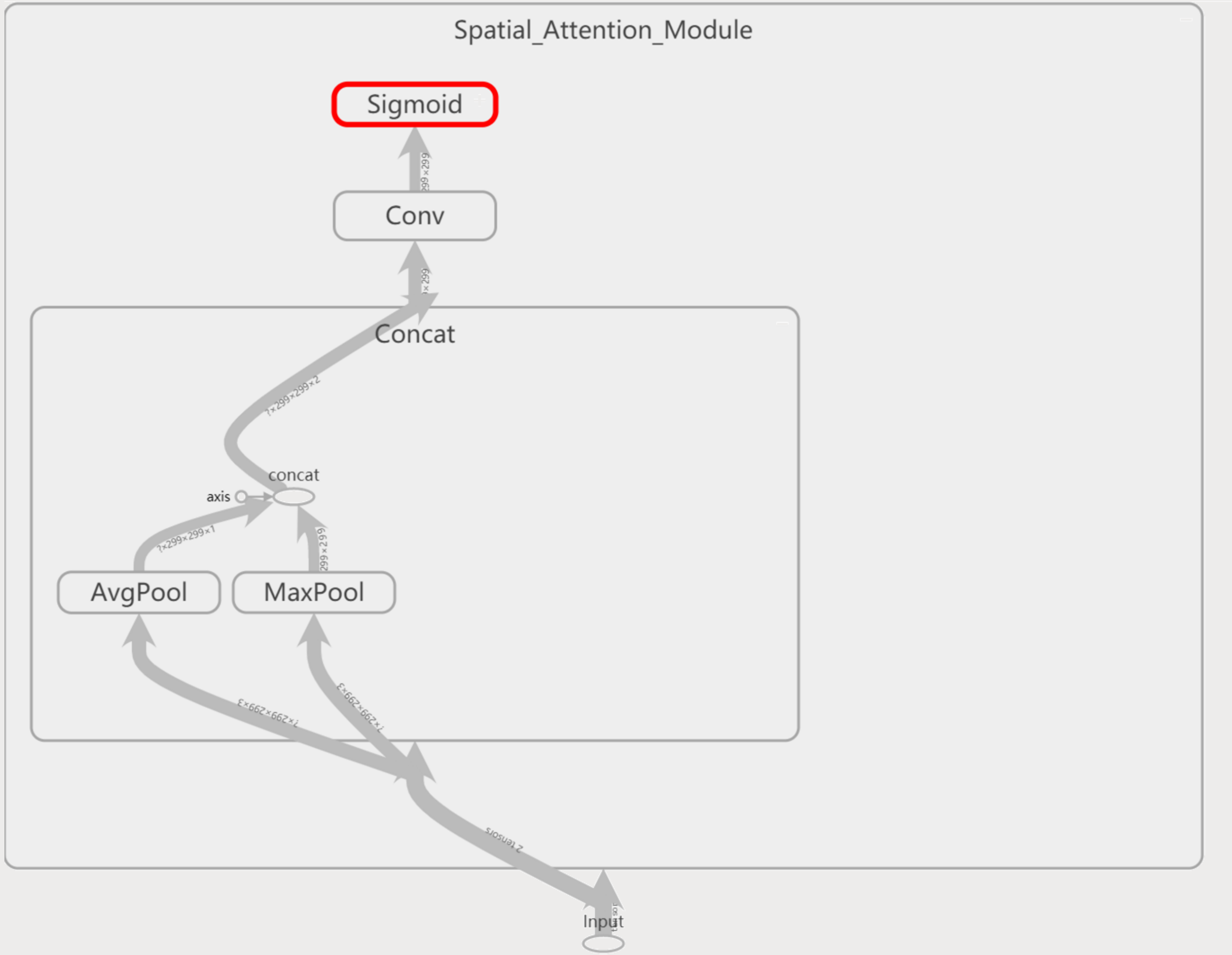
Spatial attention module

参考代码:

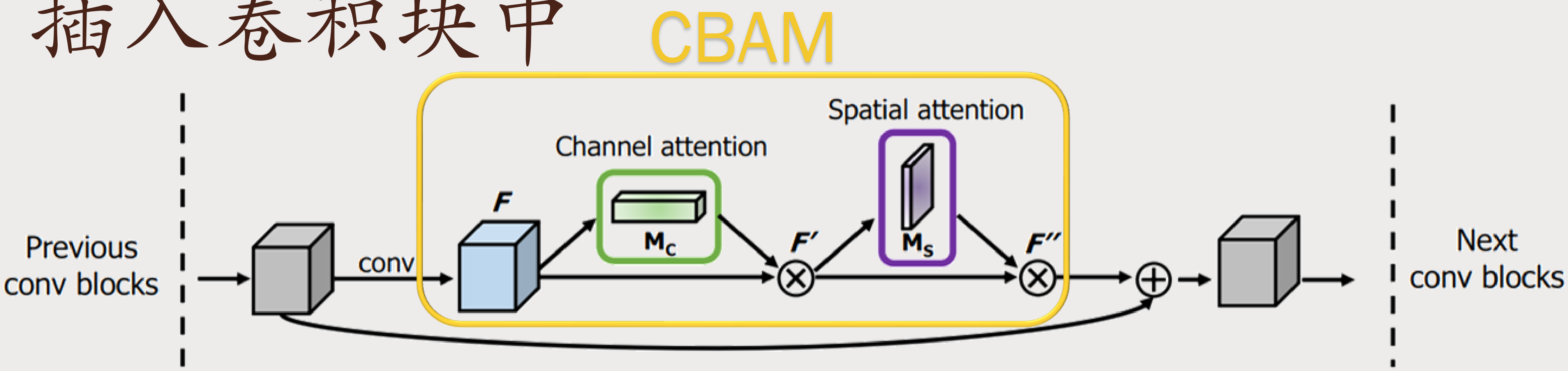
```
def spatial_attention_module(channel_refined_feature):
     with tf.name_scope('Spatial_Attention_Module'):
          tensor_shape = channel_refined_feature.get_shape().as_list()
          height = tensor_shape[-3]
          width = tensor_shape[-2]
          channel = tensor_shape[-1]
          with tf.name_scope('Concat'):
               with tf.name_scope('MaxPool'):
                    F_max = tf.nn.max_pool(channel_refined_feature, ksize = [1,1,1,channel], strides = [1,1,1,1], padding = 
'VALID')
               with tf.name_scope('AvgPool'):
                    F_avg = tf.reduce_mean(channel_refined_feature, axis=3)
                    F_avg = tf.reshape(F_avg, [-1,height,width,1])
               Fs = tf.concat([F_avg, F_max], 3)
          with tf.name_scope('Conv'):
               with tf.name_scope('Variable'):
                    filters = weight_variable([7,7,2,1], name='filter')
                    bias = weight_variable([1], name='bias')
               with tf.name_scope("Convolution"):
                    layer = conv2d(Fs, filters, strides=[1,1,1,1], padding = 'SAME') + bias
          with tf.name_scope('Sigmoid'):
               spatial_attention = tf.nn.sigmoid(layer)
     return spatial_attention
```

Spatial attention module

Tensorboard可视化结果:

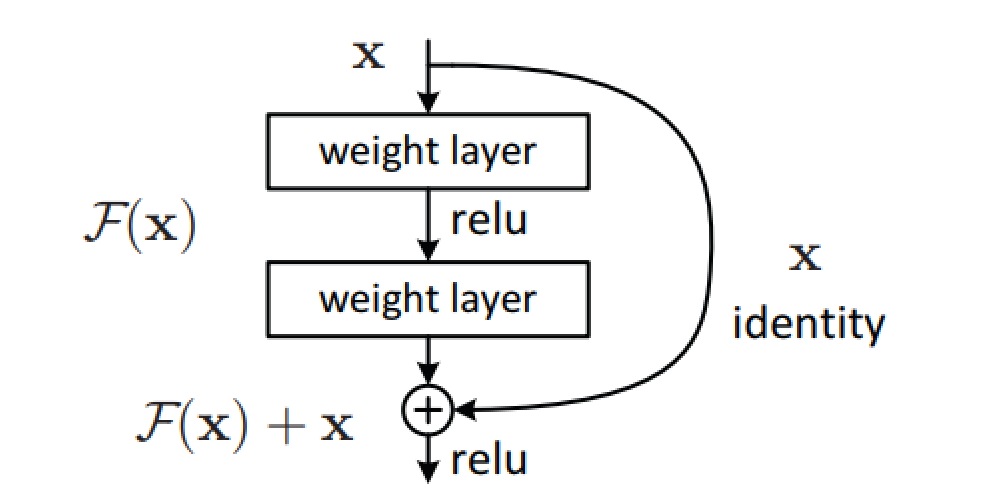


插入卷积块中



ResBlock + CBAM

- CBAM与ResBlock结合的方式如图

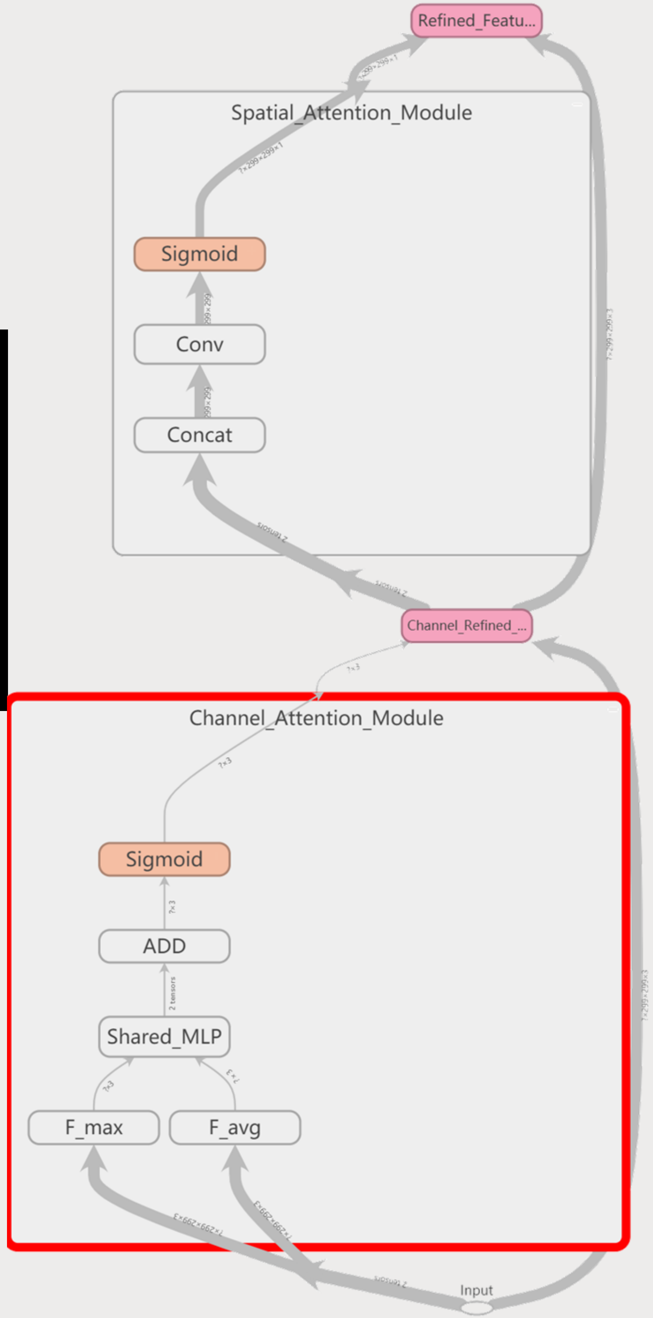


ResBlock

CBAM实现代码：

```
def CBAM(F, r):
    channel_attention = channel_attention_module(F, r)
    with tf.name_scope('Channel_Refined_Feature’):
        channel_refined_feature = channel_attention * F
        spatial_attention = spatial_attention_module(channel_refined_feature)
    with tf.name_scope('Refined_Feature’):
        refined_feature = spatial_attention*channel_refined_feature
    return refined_feature
```

完整代码：https://github.com/CExplorer/CBAM-tf



完整的框架图