前言

CBAM（ Convolutional Block Attention Module ）是一种轻量级注意力模块的提出于 2018 年，它可以在空间维度和通道维度上进行 Attention 操作。论文在 Resnet 和 MobileNet 上加入 CBAM 模块进行对比，并针对两个注意力模块应用的先后进行实验，同时进行 CAM 可视化，可以看到 Attention 更关注目标物体。

1.什么是 CBAM?

CBAM（Convolutional Block Attention Module）是轻量级的卷积注意力模块，它结合了通道和空间的注意力机制模块。

上图可以看到，CBAM 包含 CAM（Channel Attention Module）和 SAM（Spartial Attention Module）两个子模块，分别进行通道和空间上的 Attention。这样不只能够节约参数和计算力，并且保证了其能够做为即插即用的模块集成到现有的网络架构中去。

由上图所示，有输入、通道注意力模块、空间注意力模块和输出组成。输入特征，然后是通道注意力模块一维卷积，将卷积结果乘原图，将 CAM 输出结果作为输入，进行空间注意力模块的二维卷积，再将输出结果与原图相乘。

（1）Channel attention module（CAM）

通道注意力模块：通道维度不变，压缩空间维度。该模块关注输入图片中有意义的信息(分类任务就关注因为什么分成了不同类别)。

图解：将输入的 feature map 经过两个并行的 MaxPool 层和 AvgPool 层，将特征图从 C*H*W 变为 C*1*1 的大小，然后经过 Share MLP 模块，在该模块中，它先将通道数压缩为原来的 1/r（Reduction，减少率）倍，再扩张到原通道数，经过 ReLU 激活函数得到两个激活后的结果。将这两个输出结果进行逐元素相加，再通过一个 sigmoid 激活函数得到 Channel Attention 的输出结果，再将这个输出结果乘原图，变回 C*H*W 的大小。

通道注意力公式：

CAM 与 SEnet 的不同之处是加了一个并行的最大池化层，提取到的高层特征更全面，更丰富。论文中也对为什么如此改动做出了解释。

AvgPool & MaxPool 对比实验

在 channel attention 中，表 1 对于 pooling 的使用进行了实验对比，发现 avg & max 的并行池化的效果要更好。这里也有可能是池化丢失的信息太多，avg&max 的并行连接方式比单一的池化丢失的信息更少，所以效果会更好一点。

（2）Spatial attention module（SAM）

空间注意力模块：空间维度不变，压缩通道维度。该模块关注的是目标的位置信息。

图解：将 Channel Attention 的输出结果通过最大池化和平均池化得到两个 1*H*W 的特征图，然后经过 Concat 操作对两个特征图进行拼接，通过 *7*7 卷积变为 1 通道的特征图（实验证明 7*7 效果比 3*3 好），再经过一个 sigmoid 得到 Spatial Attention 的特征图，最后将输出结果乘原图变回* C*H*W 大小。

空间注意力公式：

（3）CAM 和 SAM 组合形式

通道注意力和空间注意力这两个模块能够以并行或者串行顺序的方式组合在一块儿，关于通道和空间上的串行顺序和并行作者进行了实验对比，发现先通道再空间的结果会稍微好一点。具体实验结果如下：

由表三可以看出，基于 ResNet 网络，两个 Attention 模块按 Channel Attention + Spatial Attention 的顺序效果会更好一些。

2.消融实验

（1）Channel attention

首先是不同的通道注意力结果比较，平均池化、最大池化和两种联合使用并使用共享 MLP 来进行推断保存参数，结果如下：

首先，参数量和内存损耗差距不大，而错误率的提高，显然两者联合的方法更优。

（2）Spatial attention

对比 *7*7 卷积核和 3*3 卷积核的效果，结果 7*7 卷积核效果更好

（3）Channel attention+spatial attention

对比 SEnet、CAM 和 SAM 并行、SAM+CAM 和 CAM+SAM 的效果，最终 CAM+SAM 效果最好 。

3.图像分类

再数据集 ImageNet-1K 上使用 ResNet 网络进行对比实验

4.目标检测

数据集：MS COCO 和 VOC 2007 如下表所示： MS COCO 上，CBAM 在识别任务上泛化性能较基线网络有了显著提高。

如下表： VOC 2007 中，采用阶梯检测框架，并将 SE 和 CBAM 应用于检测器。CBAM 极大的改善了所有强大的基线与微不足道的额外参数。

5.CBAM 可视化

本文利用 Grad CAM 对不一样的网络进行可视化后，能够发现，引入 CBAM 后，特征覆盖到了待识别物体的更多部位，而且最终判别物体的几率也更高，这表示注意力机制的确让网络学会了关注重点信息。

6.Pytorch 代码实现

代码实现：

```
import torch
import torch.nn as nn

class CBAMLayer(nn.Module):
def **init**(self, channel, reduction=16, spatial_kernel=7):
super(CBAMLayer, self).**init**()

        # channel attention 压缩H,W为1
        self.max_pool = nn.AdaptiveMaxPool2d(1)
        self.avg_pool = nn.AdaptiveAvgPool2d(1)

        # shared MLP
        self.mlp = nn.Sequential(
            # Conv2d比Linear方便操作
            # nn.Linear(channel, channel // reduction, bias=False)
            nn.Conv2d(channel, channel // reduction, 1, bias=False),
            # inplace=True直接替换，节省内存
            nn.ReLU(inplace=True),
            # nn.Linear(channel // reduction, channel,bias=False)
            nn.Conv2d(channel // reduction, channel, 1, bias=False)
        )

        # spatial attention
        self.conv = nn.Conv2d(2, 1, kernel_size=spatial_kernel,
                              padding=spatial_kernel // 2, bias=False)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        max_out = self.mlp(self.max_pool(x))
        avg_out = self.mlp(self.avg_pool(x))
        channel_out = self.sigmoid(max_out + avg_out)
        x = channel_out * x

        max_out, _ = torch.max(x, dim=1, keepdim=True)
        avg_out = torch.mean(x, dim=1, keepdim=True)
        spatial_out = self.sigmoid(self.conv(torch.cat([max_out, avg_out],
dim=1)))
        x = spatial_out * x
        return x

x = torch.randn(1,1024,32,32)
net = CBAMLayer(1024)
y = net.forward(x)
print(y.shape)
```

总结

CBAM 中作者对两个注意力机制使用的先后顺序做了实验，发现通道注意力在空间注意力之前效果最好。 作者在实验中应用了 grad-CAM 来可视化 feature map，是个非常有用的可视化手段，在图像分类任务中可以观察

feature map 的特征，解释了为什么模型将原图分类到某一类的结果。 加入 CBAM 模块不一定会给网络带来性能上的提升，受自身网络还有数据等其他因素影响，甚至会下降。如果网络模型的泛化能力已经很强，而你的数据集不是 benchmarks 而是自己采集的数据集的话，不建议加入 CBAM 模块。CBAM 性能虽然改进的比 SE 高了不少，但绝不是无脑加入到网络里就能有提升的。也要根据自己的数据、网络等因素综合考量。

- 原文链接