卷积神经网络的复杂度分析

在梳理CNN经典模型的过程中，我理解到其实经典模型演进中的**很多创新点都与改善模型计算复杂度紧密相关**，因此今天就让我们对卷积神经网络的复杂度分析简单总结一下下。

> 本文主要关注的是**针对模型本身的复杂度分析**（其实并不是很复杂啦～）。如果想要进一步评估模型**在计算平台上的理论计算性能**，则需要了解 Roofline Model 的相关理论，欢迎阅读本文的进阶版： Roofline Model与深度学习模型的性能分析。

“复杂度分析”其实没有那么复杂啦~

---

1. 时间复杂度

即模型的运算次数，可用 $\textbf{FLOPs}$ 衡量，也就是浮点运算次数（FLoating-point OPerations）。

> ### 1.1 单个卷积层的时间复杂度

$$\textbf{Time} \sim O(M^2 \cdot K^2 \cdot C_{in} \cdot C_{out})$$

- M 每个卷积核**输出**特征图 $(Feature\ Map)$ 的边长
- K 每个卷积核 $(Kernel)$ 的边长
- C_{in} 每个卷积核的通道数，也即输入通道数，也即上一层的输出通道数。
- C_{out} 本卷积层具有的卷积核个数，也即输出通道数。
- 可见，每个卷积层的时间复杂度由**输出**特征图面积 M^2 、卷积核面积 K^2 、输入 C_{in} 和输出通道数 C_{out} 完全决定。
- 其中，**输出**特征图尺寸本身又由**输入**矩阵尺寸 X 、卷积核尺寸 K 、$Padding$、 $Stride$ 这四个参数所决定，表示如下：

$$M = (X - K + 2 * Padding)/Stride + 1$$

- 注1：为了简化表达式中的变量个数，这里统一假设输入和卷积核的形状都是正方形。
- 注2：严格来讲每层应该还包含 1 个 $Bias$ 参数，这里为了简洁就省略了。

1.2 卷积神经网络整体的时间复杂度

$$\textbf{Time} \sim O\left(\sum_{l=1}^{D} M_l^2 \cdot K_l^2 \cdot C_{l-1} \cdot C_l\right)$$

- D 神经网络所具有的卷积层数，也即**网络的深度。**
- l 神经网络第 l 个卷积层
- C_l 神经网络第 l 个卷积层的输出通道数 C_{out} ，也即该层的卷积核个数。
- 对于第 l 个卷积层而言，其输入通道数 C_{in} 就是第 $(l-1)$ 个卷积层的输出通道数。
- 可见，CNN整体的时间复杂度并不神秘，只是所有卷积层的时间复杂度累加而已。
- 简而言之，层内连乘，层间累加。

示例：用 Numpy 手动简单实现二维卷积

假设 Stride = 1, Padding = 0, img 和 kernel 都是 np.ndarray.

```
def conv2d(img, kernel):
    height, width, in_channels = img.shape
    kernel_height, kernel_width, in_channels, out_channels = kernel.shape
    out_height = height - kernel_height + 1
    out_width = width - kernel_width + 1
    feature_maps = np.zeros(shape=(out_height, out_width, out_channels))
    for oc in range(out_channels):              # Iterate out_channels (# of kernels)
        for h in range(out_height):             # Iterate out_height
            for w in range(out_width):          # Iterate out_width
                for ic in range(in_channels):   # Iterate in_channels
                    patch = img[h: h + kernel_height, w: w + kernel_width, ic]
                    feature_maps[h, w, oc] += np.sum(patch * kernel[:, :, ic, oc])

    return feature_maps
```

2. 空间复杂度

空间复杂度（访存量），严格来讲包括两部分：总参数量 + 各层输出特征图。

- **参数量**：模型所有带参数的层的权重参数总量（即**模型体积**，下式第一个求和表达式）
- **特征图**：模型在实时运行过程中每层所计算出的输出特征图大小（下式第二个求和表达式）

$$\mathbf{Space} \sim O\left(\sum_{l=1}^{D} K_l^2 \cdot C_{l-1} \cdot C_l + \sum_{l=1}^{D} M^2 \cdot C_l\right)$$

- 总参数量只与卷积核的尺寸 K 、通道数 C 、层数 D 相关，而**与输入数据的大小无关。**
- 输出特征图的空间占用比较容易，就是其空间尺寸 M^2 和通道数 C 的连乘。
- 注：实际上有些层（例如 ReLU）其实是可以通过原位运算完成的，此时就不用统计输出特征图这一项了。

---

3. 复杂度对模型的影响

- 时间复杂度决定了模型的训练/预测时间。如果复杂度过高，则会导致模型训练和预测耗费大量时间，既无法快速的验证想法和改善模型，也无法做到快速的预测。
- 空间复杂度决定了模型的参数数量。由于维度诅咒的限制，模型的参数越多，训练模型所需的数据量就越大，而现实生活中的数据集通常不会太大，这会导致模型的训练更容易过拟合。
- 当我们需要裁剪模型时，由于卷积核的空间尺寸通常已经很小（3x3），而网络的深度又与模型的表征能力紧密相关，不宜过多削减，因此模型裁剪通常最先下手的地方就是通道数。