卷积神经网络

前面介绍了传统神经网络，接下来我们将正式开始介绍深度学习网络的环节。

从全连接层的参数说起

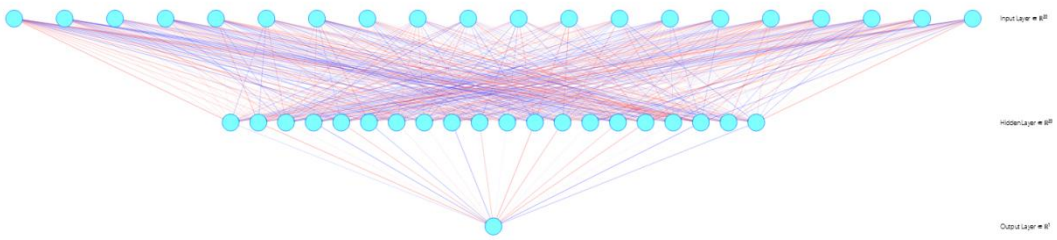
我们首先先来回顾一下全连接层(Fully connected layers, FC)，原因有二：一是全连接层在之后的深度学习网络中都有用到，常常作为网络层最后的特征优化及分类器来使用；二是说明一下全连接层的缺点是什么，以致深度学习网络不全用全连接层，而是使用卷积层来代替。

回顾一下神经元的数学公式，看一下参数数量的计算：

$$y_j = \sum_{i=1}^{N} w_{ij} x_i + b_j$$

式中，y_j表示全连接层输出向量的第 j 个值，N 表示全连接层上一层的输入长度，x_i表示输入的第 i 个特征值，w_{ij}表示第 j 个输出值所对应的第 i 个输入特征值的权重，b_j表示输出的第 j 个偏置值。

前面两章节介绍的全连接层，为了说明和计算的方便，每一层节点的个数只取了 2 或 3，或许感受不是很强烈，下图提高了一个数量级，可以看到，当输入层和隐藏层节点数取到 20 时，就已经是密密麻麻的了，第一层网络的参数数量也达到了 20*20=400。



20*20 的输入层和隐藏层

以简单的 480*480 分辨率的图片来说，全连接层输入特征值即为 480*480=230400，再假设输出的特征值为 100，那总参数量为 230400*100=23040000，即参数量已达到了千万级。更不用说更高尺度的图像特征了，参数数量将是迅猛爆炸的。

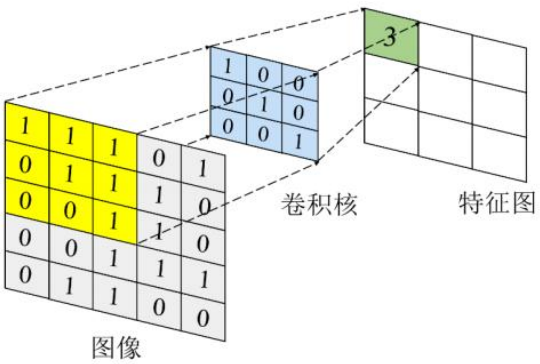
为什么呢？核心原因就在于全连接层相邻两层之间每个神经元都是相互连接的，每一个“连接”都表示一个不同的参数，导致参数量过大。这不仅带来了网络模型容易过拟合的问题，也降低了网络训练速度。为了克服这些问题，研究人员提出了卷积神经网络。

1.1 卷积层

权值共享与局部感受野

卷积神经网络 (Convolutional Neural Networks, CNN)的核心是提出了卷积层，形式也参照图像特征设置为了 a*a 这种形式，称为卷积核，后层所有神经元和前层神经元连接的参数一样称为权值共享，后层的每个神经元只和前层位置相近的神经元相关称为局部感受野，权值共享与局部感受野是卷积层的两大特性，使得网络训练的参数量大大减少，降低了网络训练的复杂度。

注意卷积层是参数层，里面是参数的数量。



以图中的输入图像和输出特征图为例，对于传统神经网络（全连接层），输出特征图的每个像素点（神经元）都是由输入图像的所有像素点乘上特定的权重

值并求和得到的，也就意味着每个输出层的神经元所对应的权重数是输入图像的所有像素点（神经元）数，而输出层的神经元数也是庞大的，因此总的权重数是庞大的。

相对于全连接层，卷积层使用了卷积核来减少参数。也就是把层与层之间密密麻麻的权重，抽象成大小远小于输入图像尺寸的卷积核（图中中间部分），卷积核上的每个像素点都代表权重值，这样对应于输出特征图的每一个像素点，只需要对覆盖区域进行卷积操作并产生对应输出，然后将卷积核按照一定的步长在输入图像上移动，最后得到高层特征图。也就是说，整张输入图像是共享该卷积核的权重的，即一个卷积核遍历整个图片（后面我们将看到，一个图片可以使用多个卷积核）。相比于全连接，卷积层的神经元的参数是由卷积核的数量和大小决定的。

所谓局部感受野，就是每个神经元都是利用卷积核感知的图像的局部区域（全连接层是感受的输入图像的全部），而该区域也就是该像素点的局部感受野。如图 2-8 所示，在 3×3 的输出特征图上的每个像素点都能映射到输入图像上 3×3 大小的局部感受野（若是传统神经网络，则输出特征图上的每个像素点都能映射到输入图像的所有像素点）。

记 X 为输入图像，Y 为输出特征图，那么卷积操作如下式所示：

$$Y = WX + b$$

式中，W 表示卷积核的权重，b 表示卷积偏置量，二维图像卷积公式可以写为：

$$y_{ij} = \sum_{m=0}^{h-1} \sum_{n=0}^{w-1} w_{mn} x_{i+m,j+n} + b$$

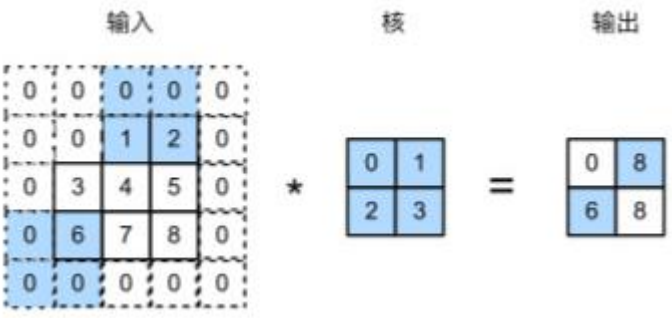
其中，y_{ij}表示二维输出特征图中坐标为(i,j)的像素点，w_{mn}是卷积核中坐标为(m,n)所对应的权重大小，h和w分别为卷积核的高和宽。

填充和步幅

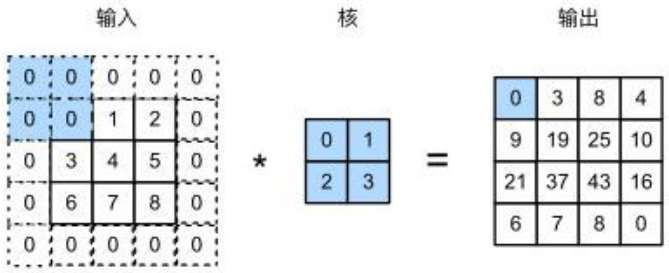
前面我们讲到卷积层是利用卷积核在前面的特征图上滑动遍历得到后面的特征图，问题就来了：每次滑动多少个像素点？

如下图所示，卷积核窗口从输入特征图的最左上方开始，按从左往右、从上往下的顺序，依次在输入特征图上滑动。我们将每次滑动的行数或列数称为步幅（stride）。控制步幅>1 可以减小输出特征图的尺寸。

下图是在特征图高和宽上步幅分别为 3 和 2 的计算过程：



填充（padding）是指在输入高和宽的两侧填充元素（通常是 0 元素），填充的核心目的是为了保证输出特征图尺寸和输入一致。上图里我们在原输入高和宽的两侧分别添加了值为 0 的元素，使得输入高和宽从 3 变成了 5，并导致输出高和宽由 2 增加到 4。



卷积后图像尺寸计算公式

卷积运算之后输出特征图相比输入图像尺寸变小了，尺寸计算公式为（不考虑空洞卷积）：

$$O = \frac{I - k + 2 \times p}{s} + 1$$

其中，输入图片大小是 I*I，卷积核大小为 k*k，步长 stride 为 s，padding 的像素数为 p。

可以利用边沿填充 (Padding)使卷积运算前后的尺寸保持不变，Padding 中填充大小 p 的计算公式如下所示

$$p = \frac{k - 1}{2}$$

式中，k 为卷积核大小。

完整版卷积尺寸考虑了空洞卷积，计算公式：

$$O = \frac{I - d \times (k - 1) + 2 \times p - 1}{s} + 1$$

其中，输入图片大小是 I*I，卷积核大小为 k*k，步长 stride 为 s，padding 的像素数为 p，dilation 为d。

多通道

这里再强调一下输出特征图及其通道数的概念，深度学习的核心就是数据和模型，而模型的核心就是寻找好的方式来提取传递数据的特征。对于输入的 RGB 图，通道数是 3,，那么中间的特征图的通道数如何计算呢？在卷积神经网络中，“一个”卷积核遍历输入图便是提取该输入图的某种特征，而一般深度学习网络

也会设置不同的卷积核个数来提取不同的特征，卷积核个数就是输出特征图的通道数。

问题来了，很多文章图示的时候，展示的卷积核是 3*3*1，不过即使第一层的原始图片也是 RGB 三通道的，中间网络层也都会设置多个卷积核来提取不同的特征，那么卷积核的“通道”是多少呢？卷积核的“通道”是不是也应该和输入通道数是对应的呢？那又是怎么计算的呢？

先来看一下 pytorch 里代码里卷积类里的参数：

```
nn.Conv2d(self, in_channels, out_channels, kernel_size, stride=1,
                    padding=0, dilation=1, groups=1,
                    bias=True, padding_mode='zeros')
```

in_channels：输入通道数

out_channels：输出通道数，等价于卷积核个数

kernel_size：卷积核尺寸

stride：步长

padding：填充宽度，主要是为了调整输出的特征图大小，一般把 padding 设置合适的值后，保持输入和输出的图像尺寸不变。

dilation：空洞卷积大小，默认为 1，这时是标准卷积，常用于图像分割任务中，主要是为了提升感受野

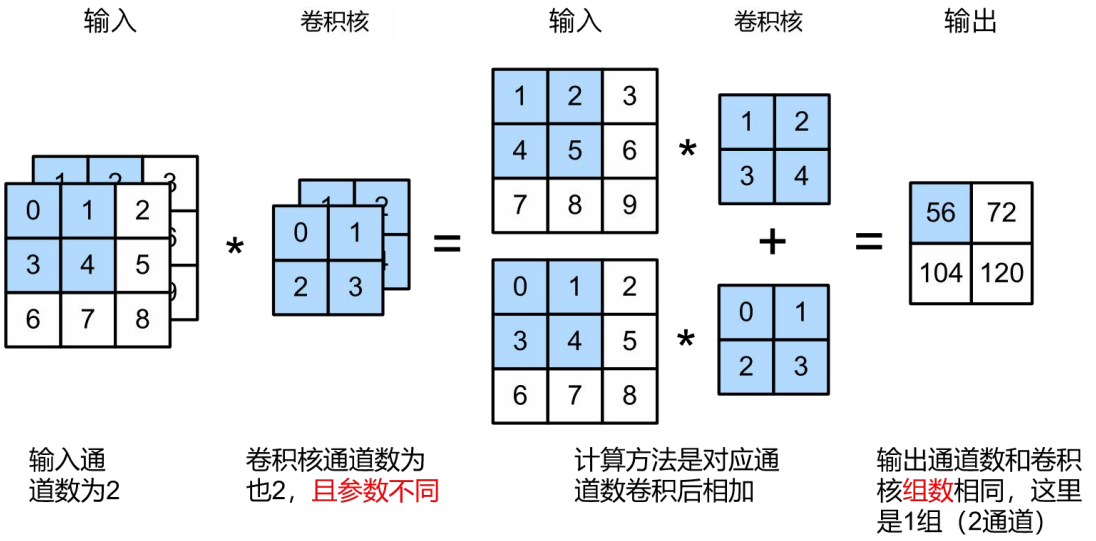
groups：分组卷积设置，主要是为了模型的轻量化，如在 ShuffleNet、MobileNet、SqueezeNet 中用到

bias：偏置

接下来以一个具体例子介绍说明一下通道数的几个概念，如下图，输入通道数（in_channels）为 2，输出通道数（out_channels）为 1，卷积核尺寸为 2*2，代入函数里即 nn.Conv2d(2, 1,2)。

尤为注意的是，输出通道数是卷积核的个数（事实上用组数更合适），这里是一组卷积核，因为输入通道数是 2，所以这一组卷积核对应的通道数也是 2，且参数各不相同（可以通过 conv_layer.weight.shape 验证，查看卷积核的 shape 是(1, 2, 2, 2)，对应是(output_channel, input_channel, kernel_size, kernel_size)），输出特征图的计算方法就是卷积核和对应通道的输入特征图卷积，然后得到多通道的输出，再相加合并成一个输出特征图。

若输出通道数设为 3，则应该有 3 组卷积核，每组仍是 2 通道，每一组会输出一个特征图，合起来共输出 3 通道的特征图。



卷积神经网络参数计算

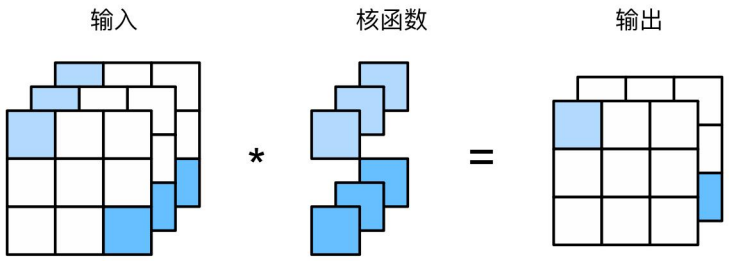
前面讲完了卷积层的基本原理，知道了参数就是由卷积核决定的，而且在多通道里又说明了不同通道的卷积核参数是不一样的，因此一个卷积层的参数计算应为：输入通道数*k*k*输出通道数，其中 k 为卷积核的尺寸。

不同于全连接层每个神经元对应一个偏置，卷积层的偏置项对应的是卷积核输出通道。前面已经讲过，输出通道数就是卷积核的组数，每组卷积核数对应输入通道数，所以每个通道卷积核对输入通道的特征图加权求和操作之后，再加上一个常数项，偏置项在激活函数之前。

1*1 卷积层

理解了前面多通道的概念，最后我们讨论一下卷积核尺寸为（1*1）的多通道卷积层。我们通常称之为 1*1 卷积层， 并将其中的卷积运算称为 1*1 卷积。

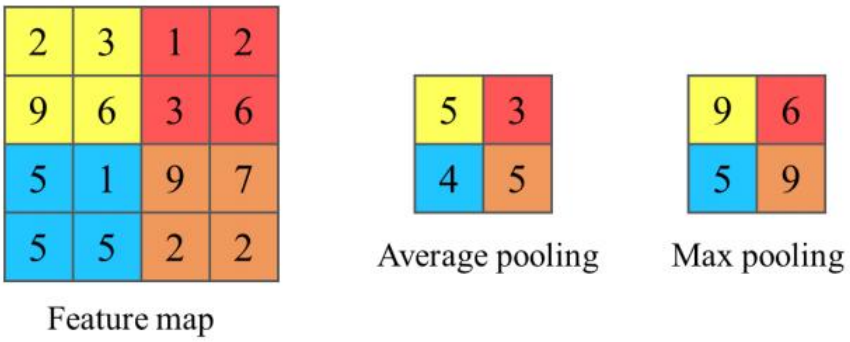
下图展示了使用输入通道数为 3、输出通道数为 2 的 1*1 卷积核计算。可以看到，输入和最终的输出具有相同的高和宽，输出的通道数取决于卷积核的组数，每一通道的输出特征图由一组卷积核对输入特征不同通道相同位置的元素按权重（就是卷积核参数值）累加得到。具体到图中，输出特征图中浅蓝色的神经元由输入通道对应位置的浅蓝色的神经元经过一组浅蓝色的卷积核（权重参数）加权求和得到，这其实是一种特殊的“全连接层”，不过卷积核是固定不变的。



因为卷积核尺寸为 1*1，所以 1*1 卷积不再能感受相邻元素，也就意味着计算只在通道维度上发生，这也代表着 1*1 卷积的主要作用是用来升维和降维。

1.2 池化层

前面详细介绍了卷积层，接下来介绍一下深度学习中另一重要的网络层——池化层。为什么需要池化层呢？简单地说，为了减少特征图的尺寸。池化层通过将前面 a*a 区域内的像素点取最大值或求平均值变为 1 个像素点，对应着最大值池化(Max pooling)和平均值池化(Average pooling)。如图所示，平均池化是在输入特征图的 2×2 区域内计算平均值作为输出特征；最大池化是在输入特征图的 2×2 区域内选择最大值作为输出特征。



代码里的参数为：

nn.MaxPool2d(kernel_size, stride=None, padding=0, dilation=1, return_indices=False, ceil_mode=False)

主要参数：

- kernel_size：池化核尺寸
- stride：步长
- padding ：填充个数
- dilation：池化核间隔大小
- ceil_mode：尺寸向上取整
- return_indices：记录池化像素索引

不同于全连接以及卷积操作，池化层不会引入新的参数，只是前面的特征图基于自身的计算。池化操作一般位于卷积操作之后，对卷积产生的多通道特征图进行降尺寸操作。

池化操作后图像维度计算公式：

$$O = \frac{I - k}{s} + 1$$

其中，O 是池化后输出维度，I 是输入图片的维度，池化层大小为 k*k，s 是步长 stride。

在处理多通道输入数据时，池化层对每个输入通道分别池化，而不是像卷积层那样将各通道的输入按通道相加。这意味着池化层的输出通道数与输入通道数相等。

在卷积操作中，卷积核的滑动步长一般要小于卷积核的尺寸，导致相邻的两次卷积操作的卷积核覆盖区域在输入图像上有所重叠，使得产生的高层特征中存在信息冗余。而在池化操作中，池化区域之间一般是不存在重叠的，直接将该区域的多个特征映射为一个特征，具有防止过拟合的作用。

1.3 卷积神经网络中的全连接层

这里再提一下全连接层，虽然当前不再用传统神经网络，但是深度学习网络中经常用全连接层做最后的特征提取和分类输出。这时，不同于卷积层和池化层是对输入图像上的局部感受野计算输出特征，全连接层输出的感受野是整张输入图像，而且需要将多通道特征图展平为一维向量作为输入。

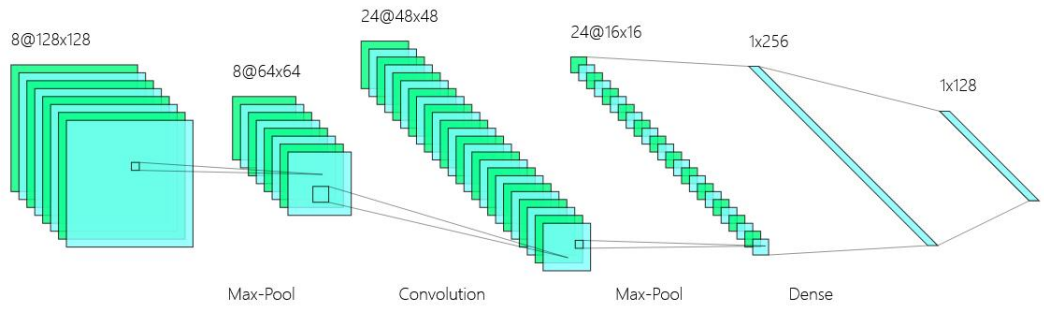
在 PyTorch 中，全连接层通常由 nn.Linear 类实现。这个类接受两个主要参数：输入特征数量（或输入尺寸）和输出特征数量（或输出尺寸）。例如，如果我们想在一个全连接层中将 10 个特征的输入转化为 5 个特征的输出，可以如下定义：

```
import torch.nn as nn
fc = nn.Linear(10, 5)
```

这个全连接层将包含 10*5=50 个参数，即权重和偏差。权重参数用于描述输入特征和输出特征之间的关系，而偏差参数则描述了输出特征的初始偏移量。

1.4 卷积神经网络结构小结

至此，我们已经搭建了卷积神经网络结构的骨架，下图为 LENET 的基本网络结构，包括卷积层，池化层以及全连接层，输出最后的结果。后面我们会以此介绍卷积神经网络的初步应用和代码实现。



参考资料

https://zh.d2l.ai/index.html