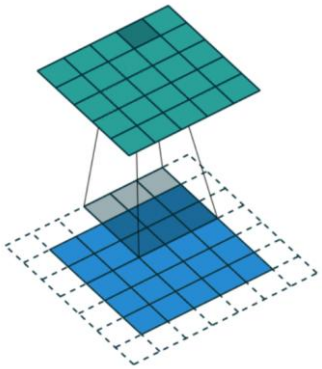
作业 4 矩阵运算在 CNN 的应用

卷积神经网络（CNN）是深度学习领域革命性的模型，尤其在图像分类这样的计算机视觉领域中。核心在于卷积核的提出，模仿了生物视觉中的感受野，该结构的参数可以学习，提取图像中的特征。本报告将介绍 CNN 的基本原理，探讨矩阵运算在这个过程的作用。

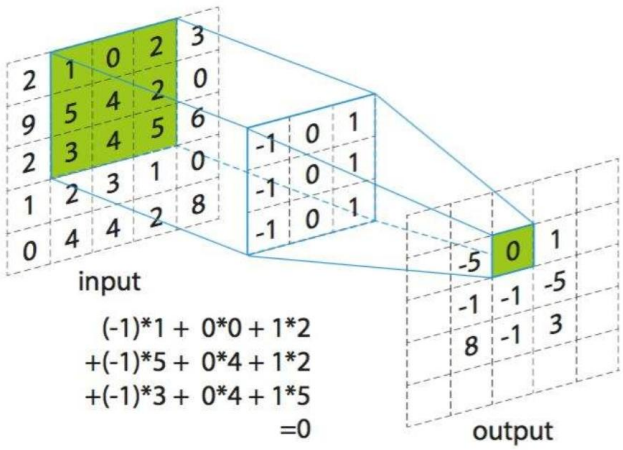
1　CNN 基本原理

卷积神经网络包括卷积层、激活层、池化层、全连接层。

卷积层：特征提取。该层的操作是对图像进行离散卷积，这是该层的命名来源和计算方法，使用一个小的、可学习的矩阵，称为卷积核，在输入数据上按一定步长滑动。



卷积核覆盖输入数据的一个局部区域（例如 3x3），这个区域称为当前输出值的“感受野”。然后将卷积核中的每个权重系数与其覆盖的输入数据对应位置的元素逐点相乘，将所有乘积结果相加，然后加上一个可学习的偏置项。这个结果就是输出特征图在当前位置的值。然后卷积核按照设定的步长在输入数据的宽度和高度方向滑动。每滑动到一个新位置，重复点积、求和、加偏置的过程。遍历完整个输入区域后，就生成了一张二维的特征图。这张图上的每个点代表了输入数据中对应局部区域与卷积核所检测特征的匹配程度。



如果像全连接层那样，每个输出神经元都连接到输入的每一个像素，对于高分辨率图像，参数量会爆炸。因此，在 CNN 的卷积层中，每个输出神经元只连接到输入数据的一个小局部区域（即其感受野），显著减少了连接数和参数数量。如果每个位置的感受野都使用不同的卷积核，参数量依然很大，因此，同一个卷积核在整个输入数据的所有位置上共享使用。无论卷积核滑动到图像的左上角还是右下角，它使用的都是同一组权重系数。例如，一个 3x3 的卷积核，无论输入图像多大，其参数数量固定在 3x3 + 1(bias) = 10 个（单通道）。这可以大幅减少参数量

输入通道：输入数据通常具有多个通道。例如，RGB 图像有 3 个通道 (Red, Green, Blue)。特征图也可以是多通道的（来自上一层的多个卷积核输出）。一个卷积核的“深度”必须与输入数据的通道数匹配。例如，处理 RGB 图像的卷积核尺寸通常是 K x K x 3 (K 是卷积核尺寸)。对于输入数据的一个位置，卷积核会覆盖所有通道上的对应局部区域。计算时，先在每个通道上进行点积运算，然后将所有通道上的点积结果求和，再加上偏置，得到该位置在一个输出通道上的值。

输出通道： 一个卷积层会使用多个（设 m 个）不同的卷积核。每个卷积核独立地在所有输入通道上滑动卷积，生成一张自己的特征图。因此，卷积层的输出是 m 个特征图。m 就是这个卷积层定义的输出通道数。

激活层：引入非线性，对卷积层输出的每一个元素应用一个非线性函数 f(•)。如果没有非线性激活函数，无论堆叠多少层卷积和全连接操作，整个网络都等价于一个单一的线性变换，表达能力极其有限，无法拟合复杂的非线性关系。常用激活函数包括 ReLU 函数，即 f(x) = max(0, x)。

池化层：空间降维，该层的目的是降低空间维度，减少特征图的高度和宽度，从而减少后续层的计算量和参数数量。该层可以扩大后续层的感受野，通过降低分辨率，后续卷积层中固定大小的卷积核能“看到”原始输入中更大的区域。

操作方式类似于卷积层，使用一个滑动窗口（通常 2x2 或 3x3）在输入特征图上滑动，但没有可学习的参数，步长通常等于窗口大小，保证不重叠或小于窗口大小，有重叠。

全连接层，通常位于 CNN 的末端（在多次卷积、激活、池化层之后）。它的任务是将前面层提取到的、分布在空间上的高级语义特征整合起来，用于

最终的分类或回归预测。

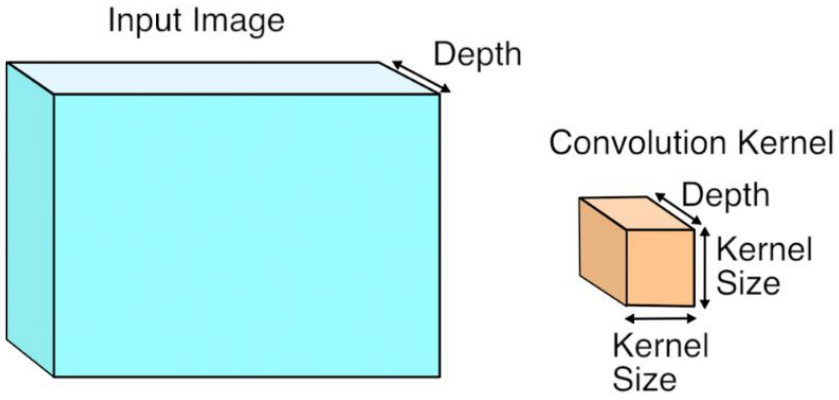
CNN 通过堆叠多个“卷积-激活-池化”模块，形成一种由浅入深的层次的特征提取结构，浅层学习检测低级、通用的局部特征，例如基本的纹理。更加深层则进一步学习高级的、具有高度语义的特征，代表整个目标或场景的抽象概念。

2 矩阵运算

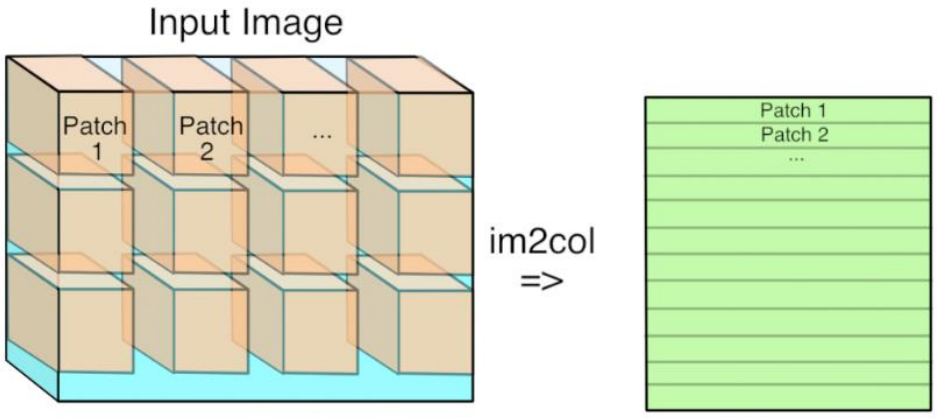
CNN 建立在卷积操作之上，但直接实现滑动窗口卷积在计算上极其低效，矩阵运算，特别是通用矩阵乘法，是解决这一问题的关键，主要应用于卷积层的高效实现，将卷积操作转化为矩阵乘可以有效的利用 GPU 对于矩阵运算极其高效的特性，因此可以使 GPU 训练 CNN 变得很高效。

实现这一转化的经典算法是 im2col 以及一些变种（im2row）。基本思想是将数据重组，使卷积操作能通过一次或几次大型矩阵乘法完成。

对于一个输入张量 Input（尺寸 H_in x W_in x C_in(深度)），使用卷积核 Kernel（尺寸 K_h x K_w x C_in），步长 (S_h, S_w)，填充 (P_h, P_w)。输出特征图尺寸为 H_out x W_out。每个输出位置 (i, j) 对应输入上的一个局部区域（感受野），尺寸 K_h x K_w x C_in。

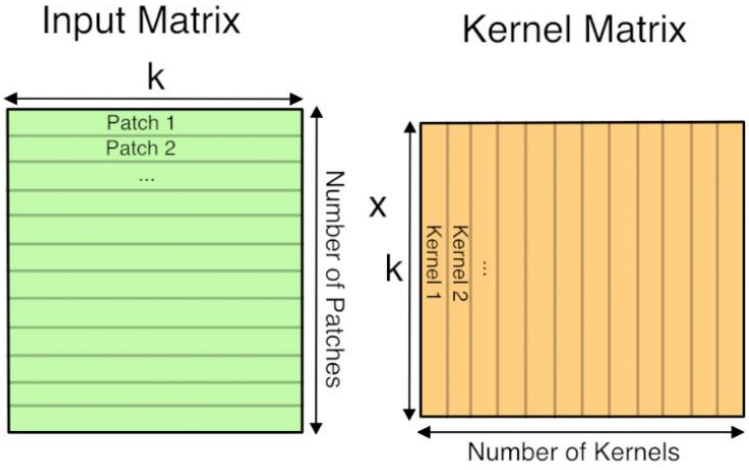


对于输出特征图上的每一个空间位置 (i, j)，找到该位置在输入 Input 上对应的感受野区域（考虑填充后）。这个区域是一个 K_h x K_w x C_in 的小立方体，将这个 K_h x K_w x C_in 的小立方体沿着空间和通道维度完全展平，变成一个列向量。这个向量的长度是 K_h * K_w * C_in。然后将所有 H_out * W_out 个这样的列向量按顺序排列，组合成一个巨大的二维矩阵，称为 Col 矩阵。



卷积层有 C_out 个不同的卷积核（每个生成一个输出通道）。将每一个卷积核也沿着空间和通道维度完全展平，变成一个向量。这个向量的长度同样是 K_h * K_w * C_in。将所有 C_out 个这样的行向量堆叠起来，形成一个二维矩阵，称为权重矩阵 W。

然后执行一次矩阵乘法：OutputMatrix = Col * W



然后将 OutputMatrix 重塑成(C_out, H_out, W_out)，这就是最终的输出特征图，最后给每个通道加上偏置。

im2col 的主要缺点是显著增加了临时内存占用，但是现在的常用框架，例如 PyTorch, TensorFlow 通常不会物理上创建一个完整的 Col 矩阵，而是使用一些更高级的技术，例如一边生成矩阵一部分元素，一边进行计算。

3 总结

矩阵运算，尤其是通用矩阵乘，是 CNN 在现代硬件上得以高效运行的基石。im2col 算法巧妙地将滑动窗口的卷积操作转化为标准的矩阵乘法，从而能够利用 CPU/GPU 上经过数十年极致优化的 BLAS 库（如 cuBLAS, MKL），使计算效率大幅提升。