Machine Learning for Neuroscience

Slides and notebooks: `https://ml4ns.github.io`

Notes by: *Nan Fletcher-Lloyd, Payam Barnaghi*

Lecture 7. Convolutional Neural Networks (CNNs)

In lecture 7, we cover the basic principles of Convolutional Neural Networks (CNNs). Expanding on our previous lecture on neural networks and multi-layer perceptrons, here we introduce convolutional neural networks (CNNs), a specialised kind of neural network for processing data that has a known grid-like topology, such as image data (a 2D grid of pixels) and time-series data (a 1D grid when samples are taken at regular time intervals).

We further expand on the structure of a CNN, including describing the process of convolution and pooling. In a convolutional layer, a sliding kernel is applied to an input to produce an output. In this lecture, we consider how to design our convolutional layers, including choice of padding, choice of stride, and the number of input and output channels.

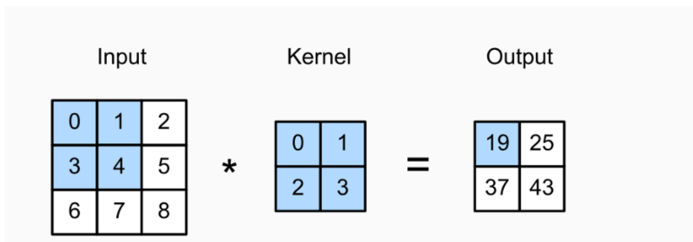

Figure 7.1. Example of a convolution
Source: Dive into Deep Learning, Aston Zhang *et al.*, https://d2l.ai/

Since we typically use small kernels for any given convolution, we risk losing pixels on the perimeter of the image. One solution to this problem is to augment/pad our input around the boundary, allowing us to retain the dimensions of the original input. This is known as the same padding. If we were to apply a kernel without padding, we would produce an output of the same dimensions as the kernel (as in Figure 7.1), and this is known as valid padding. Figure 7.1 illustrates a convolution that might occur in a CNN by taking an input and applying a sliding kernel (convolution window).

The pooling layer of a CNN is responsible for reducing the spatial size of the previous convolved layer. This is to decrease the computational power required to process the data through dimensionality reduction. Here, we introduce the two different types of pooling that can occur in a pooling layer: maximum pooling returns the maximum value from the portion of the input covered by the kernel, and average pooling returns the average of all the values from the portion of the input covered by the kernel. Figure 7.2 illustrates a 2 x 2 Maximum pooling

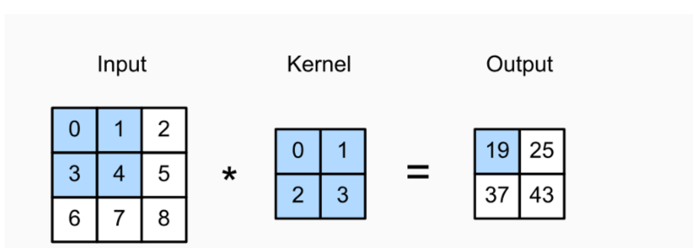

Figure 7.2. Example of maximum pooling
Source: Dive into Deep Learning, Aston Zhang *et al.*, https://d2l.ai/

Having successfully enabled the model to understand the features, we flatten the final output and feed it to a regular neural network for classification purposes (e.g. $Softmax$ classification). A fully connected layer is often added to learn non-linear combinations of the high-level features as represented by the output of the convolution layer.

Finally, we discuss the various architectures of CNNs that have been key in helping us to solve current machine learning and decision-making problems. The tutorial aims to further build your familiarity with $PyTorch$, as well as demonstrate the key techniques needed to build a CNN for classification of image data.

A corresponding assessment will help you to evaluate your understanding of CNNs and demonstrate the skills you have learned so far.