Chapter 4

Convolutional neural networks

4.1 2D images and some inductive bias

From the machine's point view an image is an array in two or three dimensions whose entries, a.k.a. *pixels* contain light intensities. In more detail, a black and white image is a matrix $I \in \mathbb{R}^{H \times L}$, whose entry (i, j) is an integer spanning from 0 (black) to 255 (white) and indicating the light intensity at the corresponding pixel location (see Figure 4.1). However, I is usually rescaled in such a way that each pixel belongs to the $[0, 1]$ interval or by other scaling techniques, so it is convenient to think about it as to a real matrix. Instead, a coloured image is a three dimensional array $I \in \mathbb{R}^{H \times L \times C}$ where C is the number of channels, usually $C = 3$. So I_{ij} is a vector in $\mathbb{R}^3$ containing the red, green and blue (RGB) intensities.

There are several learning problems related with images, such as segmentation[1], object detection, inpainting[2], generation, denoising and, of course, classification. All of them can be attacked by means of deep learning but Multilayer Perceptron alone might struggle to produce satisfying results. The first reason is that images definitely are high-dimensional data and even in the simple case of the MNIST digit in Figure 4.1, where $H = L = 28$, the input feature vector has size $D = 784$. Thence, fully connected neural networks come with a prohibitive number of parameters to infer from the data, even without considering the curse of dimensionality. The other reason has to do with the nature of the data itself: nearby pixels in a image are very likely

[1] https://en.wikipedia.org/wiki/Image_segmentation
[2] https://en.wikipedia.org/wiki/Inpainting

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	38	43	105	255	253	253	253	253	253	174	6	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	43	139	224	226	252	253	252	252	252	252	252	252	158	14	0	0	0	0	0
0	0	0	0	0	0	0	0	0	178	252	252	252	252	253	252	252	252	252	252	252	252	59	0	0	0	0	0
0	0	0	0	0	0	0	0	0	109	252	252	230	132	133	132	132	189	252	252	252	252	59	0	0	0	0	0
0	0	0	0	0	0	0	0	0	4	29	29	24	0	0	0	0	14	226	252	252	172	7	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	85	243	252	252	144	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	88	189	252	252	252	14	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	91	212	247	252	252	252	204	9	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	32	125	193	193	193	253	252	252	252	238	102	28	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	45	222	252	252	252	252	253	252	252	252	177	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	45	223	253	253	253	253	255	253	253	253	253	74	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	31	123	52	44	44	44	44	143	252	252	74	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	252	252	74	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	86	252	252	74	0	0	0	0	0	0	0	0
0	0	0	0	0	0	5	75	9	0	0	0	0	0	0	98	242	252	252	74	0	0	0	0	0	0	0	0
0	0	0	0	0	61	183	252	29	0	0	0	0	18	92	239	252	252	243	65	0	0	0	0	0	0	0	0
0	0	0	0	0	208	252	252	147	134	134	134	134	203	253	252	252	188	83	0	0	0	0	0	0	0	0	0
0	0	0	0	0	208	252	252	252	252	252	252	252	252	253	230	153	8	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	49	157	252	252	252	252	252	217	207	146	45	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	7	103	235	252	172	103	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4.1: A MNIST digit as an image (left) and as an array (right).

to share similar intensities (i.e. they are highly correlated). More generally, the structure of natural images suggests that I might live around a manifold having dimension much smaller than D. So our first intuition is: we need to look at smaller portions (or *patches*) of I in order to extract meaningful patterns. Moreover, when performing segmentation or object detection in order to separate (say) human bodies from the background, we don't want the way a pixel is classified depends on the body's location in the image, so a dedicated neural architecture should be translation *equivariant*. Similarly, when performing classification we do not want the predicted class to depend on the different position of the same key feature in the image: we also need translation *invariance*. Local correlation, translation invariance/equivariance are just a few inductive biases related to images and motivate the construction of an ad-hoc neural architecture to work with images.

4.2 Discrete convolution: I

As the name itself suggests, convolution is the key ingredient in CNNs and it allows us to inject some of the inductive biases seen so far into the neural net. Given two real (or complex) integrable functions $f, g : \mathbb{R} \to \mathbb{R}$ their convolution $f \star g$ is defined as

$$(\overline{f \star g})(t) := \int_{-\infty}^{\infty} f(x)g(t-x)dx. \tag{4.1}$$

(a) (b)

Figure 4.2: Illustration of 1D convolution for edge detection.

If f and g are probability density functions, the above equation can be rephrased as $(f \star g)(t) = \mathbb{E}_{X \sim f}[g(t - X)]$ and

Exercise 4.1. Show that if $X \sim f$ and $Y \sim g$ are two independent random variables, then the pdf of $X + Y$ is $f \star g$.

The discrete version of Eq. (4.1) reads

$$(\overline{x \star k})_i := \sum_{j=-\infty}^{\infty} x_{i-j} k_j,$$

with x and k now being real functions defined on $\mathbb{Z}$ and playing the role of g and f, respectively. If the above functions have finite support we can see them as vectors and, without loss of generality, one sets

$$(\overline{x \star k})_i = \sum_{j=1}^{L} x_{L+i-j} k_j, \tag{4.2}$$

for all $i \in \{1, \ldots, (N - L + 1)\}$, with N and L being the sizes of x and k respectively. The above equation defines 1D discrete convolution and k is called *kernel* or filter. However, for an unfortunate convention the deep learning community adopts the word convolution to refer to a related quantity, which is *cross-correlation*:

$$(x \star k)_i = \sum_{j=1}^{L} x_{i+j-1} k_j \tag{4.3}$$

where is $i \in \{1, \ldots, (N - L + 1)\}$. This is the actual main ingredient of CNNs.

Now, $(x \star k)$ is a vector of size $(N - L + 1)$ and it can easily be seen that

$$(x \star k) = \begin{bmatrix} k_1 & k_2 & \dots & k_L & 0 & \dots & 0 \\ 0 & k_1 & k_2 & \dots & k_L & \dots & 0 \\ \vdots & & & \ddots & & \vdots & \\ 0 & \dots 0 & & k_1 & k_2 & \dots & k_L \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}, \tag{4.4}$$

namely 1D cross-correlation can be formulated as a matrix product between a circulant matrix[3] and the signal to be convolved.

Exercise 4.2 After expressing the 1D convolution in Eq. (4.2) as a matrix product show the relation existing between convolution and cross-correlation.

From now on, with an unavoidable abuse of notation we refer to cross-correlation as to convolution. The main intuition behind Eq. (4.3) is: the convolved signal highlights the sub-sequences or *patches* of x correlating the most with k. For example in Figure 4.2 (a) one sees a signal of length 60 whose entries increase from 0 (black) to 1 (white). That signal is convolved with the filter $k = [-1, 0, 1]$ which encodes the notion of *edge*, i.e. an abrupt change in light intensity. The patch of x correlating the most with k is located in the middle as it can be seen in the convolved image (on the right). Depending on the features we want to detect on the main signal, we can adopt different filters. However, the main thing to keep in mind for the moment is

$$(x \star k) = W^{(k)} x,$$

where $W^{(k)}$ denotes the circulant matrix on the right hand side of Eq. (4.4) and $x = [x_1, \dots, x_N]^T$.

4.3 Discrete convolution: II

As we saw, images are 2 or 3 dimensional arrays and not vectors like x. However, what said in the previous section can be generalized. Consider for simplicity a black and white image $I \in \mathbb{R}^{H \times L}$ and a filter $K \in \mathbb{R}^{Q \times Q}$, with $Q << \min\{H, L\}$ and usually odd *odd*. Then

$$[I \star K]_{ij} = \sum_{l=1}^{Q} \sum_{m=1}^{Q} I_{i+l-1, j+m-1} K_{lm}$$

[3] https://en.wikipedia.org/wiki/Circulant_matrix

with $i \in \{1, \dots, (H - Q + 1)\}$ and $j \in \{1, \dots, (L - Q + 1)\}$.

Exercise 4.3 Given the image

$$I = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

and the kernel

$$K = \begin{bmatrix} x & y \\ z & k \end{bmatrix}$$

compute $I \star K$.

In this case too it is possible to formulate convolution as matrix multiplication. For example if one takes I and K as in the previous exercise, it is immediate to see that

$$\begin{aligned} \text{vect}(I \star K) &= \begin{bmatrix} x & y & 0 & z & k & 0 & 0 & 0 & 0 \\ 0 & x & y & 0 & z & k & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 0 & z & k & 0 \\ 0 & 0 & 0 & 0 & x & y & 0 & z & k \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \end{bmatrix} \\ &=: W^{(K)} \text{vect}(I), \end{aligned} \tag{4.5}$$

where $\text{vect}(I)$ denotes a vector obtained by stacking the rows of I in column. A simple reshape of the vector on the left hand side of the above equation will provide us with $I \star K$.

Remark. *Unlike what we saw for 1D convolution, the kernel matrix $W^{(K)}$ in the above equation is* not *exactly a circulant matrix: indeed when moving from row 2 to row 3 we need to add two zeros on the left of x. This is due to the fact that I now is a two dimensional array (a matrix) and linked with the* ***stride*** *(Section 4.4).*

Remark. *Being able to express a convolved image as a matrix/vector product allows us to write down convolutional neural networks (**CNNs**) in the very same form seen in Eq. (1.4), but with W_l now being the pseudo-circulant matrices like the one in Eq. (4.5). The parameters in $W_1, \ldots, W_L$ are learned from the data.*

As it can be seen $W^{(K)}$ is *sparse* and this is why CNNs are no longer fully connected neural nets (some links are deleted). Moreover, the non-null entries in $W^{(K)}$ are repeated (i.e. *shared*), so CNNs have a more parsimonious structure with respect to common MLPs. Finally, in order to extract several discriminant features from the input images, it is common to adopt more than one convolutional filter for each layer. So if C filters $K_1, \ldots, K_C$ are considered, the l-th hidden layer will consist of $W_l^{(K_1)}, \ldots, W_l^{(K_C)}$ arrays of learnable parameters. What said so far can easily be extended to coloured images.

4.4 Padding, stride and pooling