

1. Fundamentals

1.1. Convolution

Table 1 depicts two matrices. The one on the left represents a 5×5 single-channel image \mathbf{A} . The one on the right represents a 3×3 convolution kernel \mathbf{B} .

- (a) What is the dimensionality of the output if we forward propagate the image over the given convolution kernel with no padding and stride of 1?

If you apply the convolution kernel with no padding and a stride of 1 over the input image \mathbf{A} we obtain an output image of size (3,3). This is illustrated in the animations from ["A technical report on convolution arithmetic in the context of deep learning"](#) but with in our case, no padding and a stride of one.

- (b) Give a general formula of the output width O in terms of the input width I , kernel width K , stride S , and padding P (both in the beginning and in the end). Note that the same formula holds for the height. Make sure that your answer in part (a) is consistent with your formula.

The padding P being at the beginning and at the end, and having a stride of S , we obtain a general formula for the output width which will be similar to the one for the height :

$$O = \left\lfloor \frac{I + 2 * P - K}{S} \right\rfloor + 1$$

For 5×5 image \mathbf{A} , and 3×3 convolution kernel \mathbf{B} , we have $O_{\text{width}} = \left\lfloor \frac{5+2*0-3}{1} \right\rfloor + 1 = 3 = O_{\text{height}}$

- (c) Compute the output \mathbf{C} of forward propagating the image over the given convolution kernel. Assume that the bias term of the convolution is zero.

If we apply the kernel B to the upper corner of size 3×3 of the image A , we have: $4 \times 3 + 5 \times 3 + 2 \times 3 + 3 \times 5 + 3 \times 5 + 2 \times 5 + 4 \times 2 + 3 \times 4 + 4 \times 3 = 109$
Similarly we can find the entries for the rest of the output C :

$$C = \begin{array}{|c|c|c|} \hline 109 & 92 & 72 \\ \hline 108 & 85 & 74 \\ \hline 110 & 74 & 79 \\ \hline \end{array}$$

- (d) Suppose the gradient backpropagated from the layers above this layer is a 3×3 matrix of all 1s. Write the value of the gradient (w.r.t. the input image) backpropagated out of this layer.

Using the chain rule, we have:

$$\begin{aligned} \frac{\partial E}{\partial A_{ij}} &= \sum_{k,l=1,3} \frac{\partial E}{\partial C_{kl}} \frac{\partial C_{kl}}{\partial A_{ij}} \\ &= \sum_{k,l=1,3} \frac{\partial C_{kl}}{\partial A_{ij}} \end{aligned}$$

Expanding this, we obtain:

$$\begin{aligned} \frac{\partial E}{\partial A_{11}} &= \frac{\partial C_{11}}{\partial A_{11}} = B_{11} \\ \frac{\partial E}{\partial A_{12}} &= \frac{\partial C_{11}}{\partial A_{12}} + \frac{\partial C_{12}}{\partial A_{12}} = B_{12} + B_{11} \\ \frac{\partial E}{\partial A_{13}} &= \frac{\partial C_{11}}{\partial A_{13}} + \frac{\partial C_{12}}{\partial A_{13}} + \frac{\partial C_{13}}{\partial A_{13}} = B_{13} + B_{12} + B_{11} \\ &\vdots \end{aligned}$$

If we keep expanding each term we realize that each term is the result of a convolution between a 180-degree rotated filter B and the loss gradient $\frac{\partial E}{\partial C}$ which is a matrix of all 1s. Or the gradient backpropagated loss is the convolution of the loss to the output with the kernel B : $\frac{\partial E}{\partial A} = \frac{\partial E}{\partial C} * B$ which results in:

Hint: You are given that $\frac{\partial E}{\partial C_{ij}} = 1$ for some scalar error E and $i, j \in \{1, 2, 3\}$.
You need to compute $\frac{\partial E}{\partial A_{ij}}$ for $i, j \in \{1, \dots, 5\}$. The chain rule should help!

$\frac{\partial E}{\partial \mathbf{A}} =$					4	7	10	6	3
					9	17	25	16	8
					11	23	34	23	11
					7	16	24	17	8
					2	6	9	7	3

$\mathbf{A} =$					4	5	2	2	1
					3	3	2	2	4
					4	3	4	1	1
					5	1	4	1	2
					5	1	3	1	4

$\mathbf{B} =$			4	3	3
			5	5	5
			2	4	3

Table 1: Image Matrix (5×5) and a convolution kernel (3×3).

1.2. Pooling

The pooling is a technique for sub-sampling and comes in different flavors, for example max-pooling, average pooling, LP-pooling.

- (a) List the `torch.nn` modules for the 2D versions of these pooling techniques and read on what they do.

The 2D pooling layers are

- `MaxPool2d`
 - `AvgPool2d`
 - `FractionalMaxPool2d`
 - `LPPool2d`
 - `AdaptativeMaxPool2d`
 - `AdaptativeAvgPool2d`
- (b) Denote the k -th input feature maps to a pooling module as $\mathbf{X}^k \in \mathbb{R}^{H_{\text{in}} \times W_{\text{in}}}$ where H_{in} and W_{in} represent the input height and width, respectively. Let $\mathbf{Y}^k \in \mathbb{R}^{H_{\text{out}} \times W_{\text{out}}}$ denote the k -th output feature map of the module where H_{out} and W_{out} represent the output height and width, respectively. Let $S_{i,j}^k$ be a list of the indexes of elements in the sub-region of \mathbf{X}^k used for generating $\mathbf{Y}_{i,j}^k$, the (i,j) -th entry of \mathbf{Y}^k . Using this notation, give formulas for $\mathbf{Y}_{i,j}^k$ from three pooling modules.

- MaxPool2d

$$\mathbf{Y}_{i,j}^k = \max\{X_{l,m}^k | (l, m) \in S_{i,j}^k\}$$

- AvgPool2d

$$\mathbf{Y}_{i,j}^k = \frac{1}{|S_{i,j}^k|} \sum_{(l,m) \in S_{i,j}^k} X_{l,m}^k \text{ where } |S_{i,j}^k|: \text{ number of elements } (l,m) \text{ in } S_{i,j}^k$$

- LPPool2d

$$\mathbf{Y}_{i,j}^k = \left(\sum_{(l,m) \in S_{i,j}^k} (X_{l,m}^k)^p \right)^{\frac{1}{p}}$$

- (c) Write out the result of applying a max-pooling module with kernel size of 2 and stride of 1 to \mathbf{C} from Part 1.1.

Applying max-pooling with a kernel of size 2 and stride 1 to \mathbf{C} , we obtain the following output:

$$\text{max-pooling}(\mathbf{C}) = \begin{array}{|c|c|} \hline 109 & 92 \\ \hline 110 & 85 \\ \hline \end{array}$$

- (d) Show how and why max-pooling and average pooling can be expressed in terms of LP-pooling.

- Max-pooling: Short answer: if we apply max-pooling to the feature map X , since $\|X\|_\infty = \lim_{p \rightarrow \infty} \|X\|_p$ thus max-pooling is LP-pooling for $p \rightarrow \infty$.
- Average-pooling: Reusing the notations in question 1.2.b, then for $p=1$:

$$\begin{aligned} \text{LP-pooling}(S_{i,j}^k) &= \sum_{(l,m) \in S_{i,j}^k} X_{l,m}^k \\ &= |S_{i,j}^k| \text{Average-pooling}(S_{i,j}^k) \end{aligned}$$