DS-GA 1008:
Deep Learning, Spring 2019
Homework Assignment 2
Yves Greatti - yg390

1. Fundamentals

1.1. Convolution

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

(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 $\boldsymbol{A}$ 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 in our case, we have 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 $\boldsymbol{A}$, and 3×3 convolution kernel $\boldsymbol{B}$, with P=0 and S=1, we have $O_{\text{width}} = \lfloor \frac{5+2*0-3}{1} \rfloor + 1 = 3 = O_{\text{height}}$

(c) Compute the output $\boldsymbol{C}$ 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 $\boldsymbol{B}$ to the upper corner of size 3×3 of the image $\boldsymbol{A}$, we have: $4\times3+5\times3+2\times3+3\times5+3\times5+2\times5+4\times2+3\times4+4\times3 = 109$ Similarly we can find the entries for the rest of the output $\boldsymbol{C}$:

$\boldsymbol{C}$ =

109	92	72
108	85	74
110	74	79

(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:

$$\begin{aligned}\frac{\partial E}{\partial \boldsymbol{A}_{ij}} &= \sum_{k,l=1,3} \frac{\partial E}{\partial \boldsymbol{C}_{kl}} \frac{\partial \boldsymbol{C}_{kl}}{\partial \boldsymbol{A}_{ij}} \\ &= \sum_{k,l=1,3} \frac{\partial \boldsymbol{C}_{kl}}{\partial \boldsymbol{A}_{ij}}\end{aligned}$$

Expanding this, we obtain:

$$\begin{aligned}\frac{\partial E}{\partial \boldsymbol{A}_{11}} &= \frac{\partial \boldsymbol{C}_{11}}{\partial \boldsymbol{A}_{11}} = \boldsymbol{B}_{11} \\ \frac{\partial E}{\partial \boldsymbol{A}_{12}} &= \frac{\partial \boldsymbol{C}_{11}}{\partial \boldsymbol{A}_{12}} + \frac{\partial \boldsymbol{C}_{12}}{\partial \boldsymbol{A}_{12}} = \boldsymbol{B}_{12} + \boldsymbol{B}_{11} \\ \frac{\partial E}{\partial \boldsymbol{A}_{13}} &= \frac{\partial \boldsymbol{C}_{11}}{\partial \boldsymbol{A}_{13}} + \frac{\partial \boldsymbol{C}_{12}}{\partial \boldsymbol{A}_{13}} + \frac{\partial \boldsymbol{C}_{13}}{\partial \boldsymbol{A}_{13}} = \boldsymbol{B}_{13} + \boldsymbol{B}_{12} + \boldsymbol{B}_{11} \\ &\vdots\end{aligned}$$

If we keep expanding each term we realize that each term is the result of a convolution between the loss gradient $\frac{\partial E}{\partial \boldsymbol{C}}$, which is a matrix of all 1s, and a 180-degree rotated filter $\boldsymbol{B}$. We have as result:

$\frac{\partial E}{\partial \boldsymbol{A}}$ =

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

$\boldsymbol{A}$ =

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

$\boldsymbol{B}$ =

4	3	3
5	5	5
2	4	3

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

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

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
  In regular max pooling, you downsize an input set by taking the maximum value of smaller N x N subsections of the set (often 2x2), and try to reduce the set by a factor of N, where N is an integer.
- AvgPool2d
  Performs the average pooling on the input.
- FractionalMaxPool2d
  Fractional max pooling is slightly different than regular max pooling. Fractional max pooling, as you might expect from the word "fractional", means that the overall reduction ratio N does not have to be an integer. The sizes of the pooling regions are generated randomly but are fairly uniform.
- LPPool2d
  LP-pooling performs a 2D power-average pooling over an input signal. Each LP-pooling defines a spherical shape in a non-Euclidean space whose metrics is defined by the lp-norm.
- AdaptiveMaxPool2d
  Adaptive max pooling is a max operation per channel. The tensor be-

fore the average pooling is supposed to have as many channels as your model has classification categories.

· AdaptiveAvgPool2d
Adaptive average pooling is similar to adaptive max-pooling where the operation is an average instead of the maximum.

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

· MaxPool2d

$$\boldsymbol{Y}^k_{i,j} = \max\{X^k_{l,m} | (l, m) \in S^k_{i,j}\}$$

· AvgPool2d

$$\boldsymbol{Y}^k_{i,j} = \frac{1}{|S^k_{i,j}|} \sum_{(l,m)\in S^k_{i,j}} X^k_{l,m} \text{ where } |S^k_{i,j}| \text{: number of elements (l,m) in } S^k_{i,j}$$

· LPPool2d

$$\boldsymbol{Y}^k_{i,j} = \Big(\sum_{(l,m)\in S^k_{i,j}} (X^k_{l,m})^p \Big)^{\frac{1}{p}}$$

Note that PyTorch defined LPPool2d as $(\sum_{x\in X} x^p)^{\frac{1}{p}}$

(c) Write out the result of applying a max-pooling module with kernel size of 2 and stride of 1 to $\boldsymbol{C}$ from Part 1.1.
Applying max-pooling with a kernel of size 2 and stride 1 to , we obtain the following output:

max-pooling($\boldsymbol{C}$) =

109	92
110	85

(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 $\|X\|_\infty = \max_{x_i \text{ component of X}}\{|x_i|\} = \lim_{p\to\infty} \|X\|_p$, thus LP-pooling is max-pooling for $p \to \infty$.
- Average-pooling: Reusing the notations in question 1.2.b, then for p=1:

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