Deep Learning - Spring 2019
Homework 2

Daniel Rivera Ruiz
Department of Computer Science
New York University
drr342@nyu.edu

1. Fundamentals

1.1. Convolution

Figure 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?
Solution.
We can align the convolution kernel with the first three elements of the input matrix $\boldsymbol{A}$ (trying to align it with he fourth element would result in the kernel going beyond $\boldsymbol{A}$ and the convolution operation would be undefined). Therefore, the dimensionality of the output matrix is 3×3.

(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.
Solution.
The formula is

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

which is consistent with the previous answer for $I = 5$, $P = 0$, $K = 3$ and $S = 1$.

(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.
Solution.
The output matrix $\boldsymbol{C}$ is defined as follows:

$$\boldsymbol{C} = [\boldsymbol{C}_{ij}] = [\boldsymbol{A}[ij : (i+2)(j+2)] * \boldsymbol{B} + b_{ij}] \quad i, j \in \{1, 2, 3\}$$

where $\boldsymbol{A}[ij : (i+2)(j+2)]$ is the submatrix of $\boldsymbol{A}$ starting at A_{ij} and ending at $A_{(i+2)(j+2)}$, and b_{ij} is the bias term. Since we know that $b_{ij} = 0 \; \forall \; i, j$, we only need to compute the convolution which is defined as follows:

$$\boldsymbol{A} * \boldsymbol{B} = \sum_i \sum_j \boldsymbol{A}_{ij} \cdot \boldsymbol{B}_{ij}$$

Using this formula we arrive to the final expression for $\boldsymbol{C}$:

$$\boldsymbol{C} = \begin{bmatrix} 109 & 92 & 72 \\ 108 & 85 & 74 \\ 110 & 74 & 79 \end{bmatrix}$$

$$\boldsymbol{A} = \begin{bmatrix} 4 & 5 & 2 & 2 & 1 \\ 3 & 3 & 2 & 2 & 4 \\ 4 & 3 & 4 & 1 & 1 \\ 5 & 1 & 4 & 1 & 2 \\ 5 & 1 & 3 & 1 & 4 \end{bmatrix} \qquad \boldsymbol{B} = \begin{bmatrix} 4 & 3 & 3 \\ 5 & 5 & 5 \\ 2 & 4 & 3 \end{bmatrix}$$

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

(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.

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!

Solution.
First we notice that $\boldsymbol{A}_{ij}$ will (potentially) appear in the expression for $\boldsymbol{C}_{kl}$ as follows:

$$\boldsymbol{C}_{kl} = \ldots + \boldsymbol{A}_{ij} \cdot \boldsymbol{B}_{(i-k+1)(j-l+1)} + \ldots$$

We know by definition that $i, j \in \{1, 2, 3, 4, 5\}$ and $k, l \in \{1, 2, 3\}$. However, the indices for $\boldsymbol{B}$ should also fall within the values $\{1, 2, 3\}$. If this is not the case, it means that the element $\boldsymbol{A}_{ij}$ we are considering is in fact outside the scope of the convolution. With this restriction we can write the partial derivative for $\boldsymbol{C}_{kl}$ as follows:

$$\frac{\partial \boldsymbol{C}_{kl}}{\partial \boldsymbol{A}_{ij}} = \begin{cases} \boldsymbol{B}_{(i-k+1)(j-l+1)} & 1 \leq i-k+1 \leq 3 \wedge 1 \leq j-l+1 \leq 3 \\ 0 & \text{otherwise} \end{cases}$$

Using the fact that $\frac{\partial E}{\partial \boldsymbol{C}_{ij}} = 1$ and applying the chain rule we can write:

$$\begin{aligned} \frac{\partial E}{\partial \boldsymbol{A}_{ij}} &= \sum_{k=1}^{3} \sum_{l=1}^{3} \left(\frac{\partial E}{\partial \boldsymbol{C}_{kl}} \cdot \frac{\partial \boldsymbol{C}_{kl}}{\partial \boldsymbol{A}_{ij}} \right) \\ &= \sum_{k=1}^{3} \sum_{l=1}^{3} \boldsymbol{B}_{(i-k+1)(j-l+1)} \mathbb{1}(1 \leq i-k+1 \leq 3 \wedge 1 \leq j-l+1 \leq 3) \end{aligned}$$

where $\mathbb{1}$ is the indicator function. This definition tells us that once we set an element $\boldsymbol{A}_{ij}$ from the input matrix, we can calculate the partial derivative of the error E as a sum of elements from the kernel $\boldsymbol{B}$. If at some point during the summation we encounter an index that is out of bounds for $\boldsymbol{B}$, we simply add 0.

Thus, the desired backpropagated gradient is

$$\begin{aligned} \frac{\partial E}{\partial \boldsymbol{A}} &= \left[\frac{\partial E}{\partial \boldsymbol{A}_{ij}} \right] \quad i, j \in \{1, 2, 3, 4, 5\} \\ &= \begin{bmatrix} 4 & 7 & 10 & 6 & 3 \\ 9 & 17 & 25 & 16 & 8 \\ 11 & 23 & 34 & 23 & 11 \\ 7 & 16 & 24 & 17 & 8 \\ 2 & 6 & 9 & 7 & 3 \end{bmatrix} \end{aligned}$$

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.
Solution.

- `torch.nn.MaxPool2d`
- `torch.nn.AvgPool2d`
- `torch.nn.FractionalMaxPool2d`
- `torch.nn.LPPool2d`
- `torch.nn.AdaptiveMaxPool2d`
- `torch.nn.AdaptiveAvgPool2d`

(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.
Solution.
For the max-pooling, average pooling and LP-pooling (p-norm) modules we have:

$$\boldsymbol{Y}^k_{\max} = \left[\boldsymbol{Y}^k_{i,j}\right]_{i \in H_{\text{out}}, j \in W_{\text{out}}} = \max_{s \in S^k_{i,j}} \left(\boldsymbol{X}^k[s]\right)$$

$$\boldsymbol{Y}^k_{\text{avg}} = \left[\boldsymbol{Y}^k_{i,j}\right]_{i \in H_{\text{out}}, j \in W_{\text{out}}} = \frac{1}{|S^k_{i,j}|} \sum_{s \in S^k_{i,j}} \boldsymbol{X}^k[s]$$

$$\boldsymbol{Y}^k_{LP_p} = \left[\boldsymbol{Y}^k_{i,j}\right]_{i \in H_{\text{out}}, j \in W_{\text{out}}} = \left(\sum_{s \in S^k_{i,j}} \left(\boldsymbol{X}^k[s]\right)^p \right)^{\frac{1}{p}}$$

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

$$\boldsymbol{C}_{\max} = \begin{bmatrix} 109 & 92 \\ 110 & 85 \end{bmatrix}$$

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

- Max-pooling.
  If we look at the definition of LP-pooling with $p \to \infty$ we get:

$$\boldsymbol{Y}_{LP_\infty} = \lim_{p \to \infty} \left(\sum_{s \in S^k_{i,j}} \left(\boldsymbol{X}[s]\right)^p \right)^{\frac{1}{p}}$$

  Let $s_j = \text{argmax}(\boldsymbol{X}[s])$. It is clear that as $p \to \infty$, $\left(\boldsymbol{X}[s_j]\right)^p \gg \left(\boldsymbol{X}[s_i]\right)^p \ \forall\ i \neq j$, and therefore the summation can be rewritten as

$$\sum_{s \in S^k_{i,j}} \left(\boldsymbol{X}[s]\right)^p = \left(\boldsymbol{X}[s_j]\right)^p$$

If we replace this in the expression for $\boldsymbol{Y}_{LP_\infty}$ we get:

$$\begin{aligned}\boldsymbol{Y}_{LP_\infty} &= \lim_{p\to\infty}((\boldsymbol{X}[s_j])^p)^{\frac{1}{p}} = \lim_{p\to\infty}\boldsymbol{X}[s_j] = \boldsymbol{X}[s_j] \\ &= \max_{s\in S}(\boldsymbol{X}[s]) \\ &= \boldsymbol{Y}_{\max}\end{aligned}$$

– Average pooling.
In this case it is straightforward that with $p = 1$ we have:

$$\boldsymbol{Y}_{\text{avg}} = \frac{1}{|\boldsymbol{S}|}\cdot\boldsymbol{Y}_{LP_1}$$