

# CS-233 Theoretical Exercise 9

April 2024

## 1 Convolution - Computation

Let us denote an image as  $X$ . We can index a pixel  $x$  in this image as  $x = X_{ijc}$  (or equivalently in a more programming-oriented format  $X[i, j, c]$ ), where  $i, j, c$  are the indices for height ( $H$ ), width ( $W$ ), and channel ( $C$ ), respectively.

Now let us consider a 2D convolution kernel  $K$ , with parameters  $\{k \mid k = K_{ijc}\}$  and height  $H_K$ , width  $W_K$ , and number of channels  $C_K$ . We want to compute the convolution of this kernel with the input image  $X$ . Assume that we use stride  $s$  and padding  $p$ .

**Question 1 [feature map size]:** Give the expressions of the height, width, and number of channels of the output feature map. Assume that the stride  $s$  is carefully chosen so that the kernel convolves the image entirely, without any margin.

**Question 2 [computation cost]:** Now instead of a single kernel, we have  $C_{out}$  convolution kernels. If we just consider multiplication as one operation, how many operations are needed for the overall computation? Compare it to the number of operations we need if the output feature map is produced by a fully-connected layer instead (you can omit the bias).

**Question 3 [alternative view of convolution]:** Now consider a different problem: We have a black canvas of size  $50 \times 50$  with two white points at positions  $[20, 40]$  and  $[40, 20]$  (index starting from 0, shown in Fig. 1). We would like to obtain two Gaussian blobs (of size  $10 \times 10$ ) centered at the two white points, as shown in Fig. 2. Write pseudo-code to do so with only a convolution.

## 2 Convolution - Optimization

**Question 4 [David's CNN]:** David designed a CNN for dog/cat classification. He stacked multiple convolutional, average pooling, and fully-connected layers. However, he didn't use any non-linear activation functions, except for the last softmax operation. He believes that his CNN will surely outperform a simple logistic regression in terms of training loss. Explain why this is not correct.

Now, let us go back to the basic case of convolution between an input image  $X$  and one filter  $K$ , with no padding and a stride of 1. The output feature map is denoted as  $F$ . To simplify the case even further, let us assume that the input image only has 1 channel.

To optimize the filter  $K$ , we need to compute  $\frac{dL}{dK}$ , which is a function (denoted as  $g_K$ ) of  $\frac{dL}{dF}$  and  $X$ . To perform an adversarial attack on the input  $X$ , we need to compute  $\frac{dL}{dX}$ , which is a function (denoted as  $g_X$ ) of  $\frac{dL}{dF}$  and  $K$ . Note that all these gradients are in matrix form, *e.g.*,  $\frac{dL}{dK}$  is a matrix with elements  $\frac{dL}{dK_{ij}}$ .

**Question 5 [gradient of the filter]:** As mentioned above,  $\frac{dL}{dK} = g_K(\frac{dL}{dF}, X)$ . What is  $g_K$  (expressed in terms of  $\frac{dL}{dF}$  and  $X$ )? You should be able to get a very simple expression. **Hint:** Consider a  $3 \times 3$  input with a  $2 \times 2$  kernel, write down the expression of the elements in the output feature map ( $2 \times 2$ ), and compute the

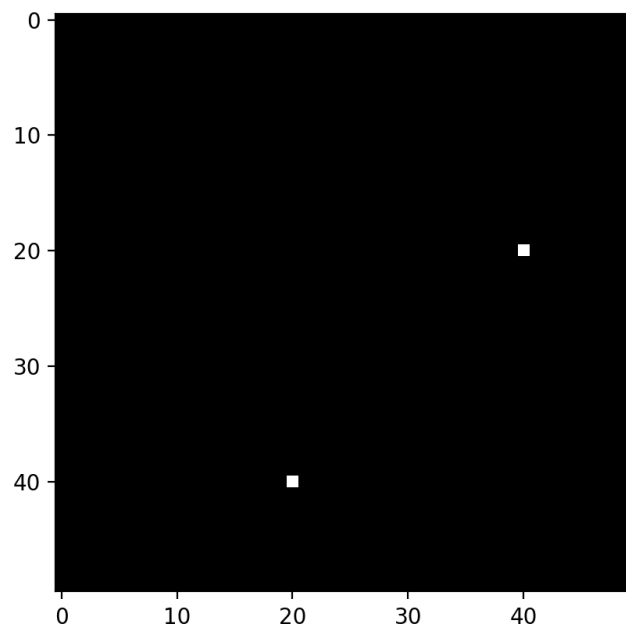


Figure 1: Black canvas with two white points

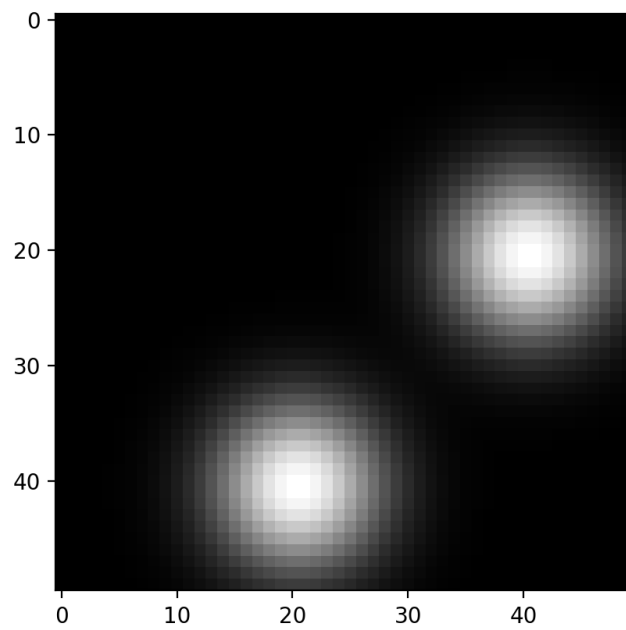


Figure 2: Desired output

gradient. This should give you something interesting.

**Question 6 [gradient of the input image]:** As mentioned above,  $\frac{dL}{dX} = g_X(\frac{dL}{dF}, K)$ . What is  $g_X$ ? You should obtain a slightly more complex expression than in the last question. **Hint:** Consider a kernel, denoted as  $K_{rot}$ , corresponding to  $K$  rotated by 180 degrees.