Miniproject 2: Implementation of a Convolutional Neural Network for Denoising Images

Mehrsa Pourya and Farzad Pourkamali
EE559 (Deep Learning) Course Project, EPFL

I. Introduction

The goal of this project is to implement a denoiser Convolutional Neural Network (CNN) framework from scratch using only Pytorch [1] tensor operations.

In particular, this frameowork consists of the implementation of the following modules with the associated *forward* and *backward* passes:

- 2-dimensional convolution (`Conv2d`) and 2-dimensional transposed convolution (`TransposeConv2d`) layers
- Upsampling layers: each (`Upsampling`) layer is wrapper around the (`TransposeConv2d`) layer
- ReLU and Sigmoid activation functions
- Sequential module which combines all the modules involved in the framework
- Mean Squared Error (MSE) loss
- Optimization module: Stochastic Gradient Descent (SGD)

II. Modules

A. Convolutional Layers

In this section, first, we discuss how a convolution operator and corresponding gradients can be implemented by linear operations. We explicitly discuss the implementation of the two modules `Conv2d` and `TransposeConv2d`.

The method discussed below is based on a series of posts on convolutional layers [2], [3]. For simplicity, we present the method for *square* kernel matrices and *symmetric* paddings, stride, and dilation. The general case of non-square kernel matrices and non-symmetric paddings, stride, and dilation can be handled similarly with slight modifications.

*Notation:*We use capital bold letters for matrices ($\boldsymbol{X}$), lower case bold letters for vectors ($\boldsymbol{x}$). $\text{vec}(.) : \mathbb{R}^{n\times n} \to \mathbb{R}^{n^2}$ is a linear map which stacks the columns of a matrix $\boldsymbol{X} \in \mathbb{R}^{n\times n}$ on top of another to get a vector in $\mathbb{R}^{n^2}$. The (usual) convolution and transposed convolution of two matrices are denoted by $*$ and $*^T$, respectively.

Let us consider the simplest case with padding = 0 ($p = 0$), stride = 1 ($s = 1$), and dilation = 1 ($d = 1$), and without the bias term. Let the input matrix (image) be $\boldsymbol{X} \in \mathbb{R}^{n\times m}$, and the kernel be $\boldsymbol{W} \in \mathbb{R}^{k\times k}$. As explained in A1, convolution can be expressed as

$$\text{vec}\big(\boldsymbol{X} * \boldsymbol{W}\big)^T = \text{vec}(\boldsymbol{W})^T \big[\boldsymbol{x}^{(1)} | \dots | \boldsymbol{x}^{(L)}\big] \tag{1}$$

where T is the size of the output as a vector ($L = (n-k+1)(m-k+1)$), and $\boldsymbol{x}^{(i)}$ is the vectorization of $k \times k$ sub-matrices of $\boldsymbol{X}$, e.g. for $k = 2$, $\boldsymbol{x}^{(1)} = \text{vec}\Big(\begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix}\Big) = [X_{11}, X_{12}, X_{21}, X_{22}]^T$. Therefore, by a linear operation (1) followed by a matricization operation, we can perform a convolution task. The general case of convolution can be handled in the following way:

- If padding is non-zeros, the input is padded with zeros on all 4 sides, and the new matrix is the input in (1).
- If stride is not one, the sub-matrices (which constructs the columns of the second matrix in the right-hand side of (1)) are chosen $s-1$ distance. (see Example 1)
- If dilation is not one, by definition, $d-1$ zeros columns/rows are inserted between columns and rows of $\boldsymbol{W}$, and the convolution operation can be performed as in (1).

If there is a bias term b, a term $b \times \boldsymbol{J}$ is added to the output of the convolution $\boldsymbol{X} * \boldsymbol{W}$, where $\boldsymbol{J}$ is an all-one matrix of the proper size (as the output size of convolution).

Denote the output of the convolution by $\boldsymbol{O} = \boldsymbol{X} * \boldsymbol{W} (+b \times \boldsymbol{J})$. Let $\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}$ be the gradient of loss with respect to (w.r.t.) $\boldsymbol{O}$, where $\big(\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}\big)_{i,j} = \frac{\partial \mathcal{L}}{\partial O_{i,j}}$. Then, the gradient of loss w.r.t. the kernel can be written as a usual convolution:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}} = \boldsymbol{X} * \frac{\partial \mathcal{L}}{\partial \boldsymbol{O}} \tag{2}$$

and the convolution operation can be carried out as discussed before with the following considerations (note that the above convolution should be performed with stride one in all the cases):

- If padding (of the original convolution operation) is non-zero, the matrix $\boldsymbol{X}$ in (2) should be the padded input.
- If stride is not one, the convolution is performed with the dilated version of $\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}$ with dilation equal to the stride of the original convolution operation, see A2. *Moreover, note that along dilating $\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}$ one might need to add zeros columns/rows to it to match the sizes for the convolution in* (2).
- If dilation (of the original convolution operation) is not one, the left-hand side (LHS) of (2) is the gradient of the loss w.r.t. the dilated kernel (zeros between rows and columns), so the gradient w.r.t. the original kernel should be appropriately extracted from the matrix in the LHS.

The gradient of the loss w.r.t. the bias term can be easily written as

$$\frac{\partial \mathcal{L}}{\partial b} = \boldsymbol{J} * \frac{\partial \mathcal{L}}{\partial \boldsymbol{O}} = \sum_{i,j} \frac{\partial \mathcal{L}}{\partial O_{i,j}} \tag{3}$$

where $\boldsymbol{J}$ is the all-one matrix of the same size as $\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}$.

As as explained in A4, the gradient w.r.t. the input can also be expressed as a convolution:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{X}} = \big(\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}\big)^{(\tilde{\mathrm{p}})} * \boldsymbol{W}^0 \tag{4}$$

where $\boldsymbol{W}^0$ is the $180°$-rotated version of $\boldsymbol{W}$. $\big(\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}\big)^{(\tilde{\mathrm{p}})}$ is the padded version of $\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}$, in which the paddings on each side are not the equal and is done to match the size for the convolution.

- If padding (of the original convolution operation) is non-zero, the matrix in LHS of (4) is the gradient of the loss w.r.t. the padded input, so the gradient w.r.t. the input should be extracted properly from the matrix in the LHS.
- If stride is not one, $\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}$ should be dilated with d equal to the stride of the original convolution operation and then padded properly to match the size.
- If dilation (of the original convolution operation) is not one, the convolution in (4) is performed with the dilated (then rotated) version of $\boldsymbol{W}$.

Therefore, the gradient of a convolution operator w.r.t. weights, bias, and input can be expressed as a usual convolution operation, which is linear due to (1).

The 2d transposed convolution be expressed as usual convolution:

$$\boldsymbol{X} *^T \boldsymbol{W} = \boldsymbol{X}^{(\mathrm{p})} * \boldsymbol{W}^0 \tag{5}$$

where $\boldsymbol{X}^{(\mathrm{p})}$ is the properly padded input, and $\boldsymbol{W}^0$ is the $180°$-rotated version of kernel weights. Therefore, the gradient w.r.t. weights, bias, and input can be written as convolution operation as discussed earlier, which we omit here due to the space limit, see B.

Implementation of convolution using Pytorch: As we have seen, both convolution and transposed convolution and corresponding gradients can be expressed as the convolution of two matrices. Now, we discuss implementing the convolution operation using Pytorch. By (1), convolution can be seen as a matrix multiplication, which can be implemented easily. It remains to construct proper matrices for the convolution. In all the cases, the second matrix in the RHS of (1) is constructed from the input of the convolution (the first matrix) using `torch.nn.Unfold`, and kernel matrix (the first matrix in RHS of (1)) is reshaped using `.reshape`. Moreover, the last step, which is constructing the output matrix from its vectorization, is done using `torch.nn.Fold`.

1) `Conv2d`: This module performs a 2-dimensional convolution on the input of the size $(N \times C_{\mathrm{in}} \times H_{\mathrm{in}} \times W_{\mathrm{in}})$, where N is the batch size and C_{in} is the number of the input channel. The output is of the size $(N \times C_{\mathrm{out}} \times H_{\mathrm{out}} \times W_{\mathrm{out}})$, where C_{out} is the number of output channels.

The inputs of the module are $C_{\mathrm{in}}, C_{\mathrm{out}}$, kernel size $((k_1, k_2)$, stride, padding, dilation (all can be of the type `int` or `tuple`), and bias which is a binary variable. The parameters are kernel weights of the size $C_{\mathrm{out}} \times C_{\mathrm{in}} \times k_1 \times k_2)$ and bias of the size C_{out}.

For each sample in the batch and each input channel, the input matrix is convolved with the corresponding kernel to get the output of the output channel.

2) `TransposeConv2d`: This module performs a 2-dimensional transposed convolution on the input of the size $(N \times C_{\mathrm{in}} \times H_{\mathrm{in}} \times W_{\mathrm{in}})$, where N is the batch size and C_{in} is the number of the input channel. The output is of the size $(N \times C_{\mathrm{out}} \times H_{\mathrm{out}} \times W_{\mathrm{out}})$, where C_{out} is the number of output channels.

The inputs of the module are $C_{\mathrm{in}}, C_{\mathrm{out}}$, kernel size $((k_1, k_2)$, stride, padding, dilation (all can be of the type `int` or `tuple`), and bias which is a binary variable. The parameters are kernel weights of the size $C_{\mathrm{in}} \times C_{\mathrm{out}} \times k_1 \times k_2$ and bias of the size C_{out}.

For both modules, the `forward` method return the output of the module for the given input and the `backward` return the gradients in of the input while accumulating the gradients of the parameters. In addition the method `param` returns a list including the parameters and their gradient as paired tuples.

B. Activation Functions

We have implemented two activation functions in our framework:

1) `ReLU`: ReLU is defined as:

$$\mathrm{ReLU} : \mathbb{R} \to \mathbb{R} \qquad \mathrm{ReLU}(x) = \max(0, x)$$

and its derivative w.r.t. the input reads

$$\mathrm{ReLU}'(x) = \mathbb{I}(x > 0)$$

where $\mathbb{I}$ is the indicator function. ReLU acts element-wise on its input, $\boldsymbol{O} = \mathrm{ReLU}(\boldsymbol{X})$. In the *backward* pass, the gradient of loss w.r.t. the input reads:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{X}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{O}} \odot \frac{\partial \boldsymbol{O}}{\partial \boldsymbol{X}} = \mathbb{I}(\boldsymbol{X} > 0) \odot \frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}$$

where $\odot$ denotes the *Hadamrd*(element-wise) product, and $\mathbb{I}(.)$ acts element-wise on $\boldsymbol{X}$.

2) `Sigmiod`: Sigmoid is defined as :

$$\sigma : \mathbb{R} \to \mathbb{R} \qquad \sigma(x) = \frac{1}{1 + e^{-x}}$$

and its derivative w.r.t. the input reads

$$\sigma'(x) = \frac{e^{-x}}{\big(1 + e^{-x}\big)^2}$$

Sigmoid acts element-wise on its input, $\boldsymbol{O} = \sigma(\boldsymbol{X})$. In the *backward* pass, the gradient of loss w.r.t. the input reads:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{X}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{O}} \odot \frac{\partial \boldsymbol{O}}{\partial \boldsymbol{X}} = \sigma'(\boldsymbol{X}) \odot \frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}$$

C. Sequential

This module combines other given modules in a sequential structure. The constructor of this module initiates the network layers by a tuple based on the given input. Its `forward` method iterates over the layers, passing the forward output of one layer to the input of the next one. The `backward` does the same thing in the reverse direction for the gradients. The

`param` method returns the parameters of all layers and their gradients as a list of paired tuples. In addition, this module has a `update_parameters` which updates the parameters of each layer using the new given values. This method is used in the optimizing process to keep track of the changes in each module.

D. Loss

Based on our experiments in miniproject-1, we have only implemented the mean squared error (MSE) loss, which we denote by $\mathcal{L}$. Consider a batch of size N of matrices $\{\boldsymbol{X}_i\}_{i=1}^N, \boldsymbol{X}_i \in \mathbb{R}^{H\times W}$ and $\{\boldsymbol{Y}_i\}$ their corresponding targets. Let $\{\hat{\boldsymbol{X}}_i\}$ be the output of the network, the loss $\mathcal{L}$ is defined as:

$$\mathcal{L}: \mathbb{R}^{N\times H\times W} \times \mathbb{R}^{N\times H\times W} \to \mathbb{R}$$

$$\mathcal{L}\big(\{\hat{\boldsymbol{X}}_i, \boldsymbol{Y}_i\}_{i=1}^N\big) = \frac{1}{N}\frac{1}{HW}\sum_{i=1}^{N} \|\hat{\boldsymbol{X}}_i - \boldsymbol{Y}_i\|_F^2$$

where $\|.\|_F$ is the *Frobenius* norm. Derivative of the loss w.r.t. its input $\hat{\boldsymbol{X}}_i$ reads:

$$\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{X}}_i} = \frac{2}{N}\frac{1}{HW}(\hat{\boldsymbol{X}}_i - \boldsymbol{Y}_i)$$

E. Stochastic Gradient Descent (SGD)

This module takes the network parameters and the learning rate λ as the input in the constructor. It has a method `step` which updates the each parameter w by a gradient descent step using its gradient $\frac{\partial \mathcal{L}}{\partial w}$ as follows:

$$w = w - \lambda \times \frac{\partial \mathcal{L}}{\partial w}.$$

The mentioned gradient is calculated by a backward pass of the network using a batch of input samples; in that sense, SGD is different from conventional gradient descent. This module has a `zero_grad` method that sets all the gradients to zero, which helps to abstain dependency of gradients associated to different batches. In addition, SGD has a `update_parameters` method that updates its parameters and gradients with the given input. This is similar to the method with the same name of Sequential to keep track of updates of the parameters. Whenever a step of the gradient descent is done, the parameters of both the sequential and the optimizer are updated.

III. Performance

Using all the described modules we implement the network of the instruction. After trying different hyperparameters, we chose the kernel size as 2. The number of the channels of the convolutional layers is illustrated in Fig. 1. We use the SGD optimizer with a learning rate of 10 and the batch size is 32.

```
self.net = Sequential(Conv2d(3, 64, 2,  stride = 2, device=self.device),
                      ReLU(),
                      Conv2d(64, 256, 2,  stride = 2, device=self.device),
                      ReLU(),
                      Upsampling(256, 64, 2,  stride = 2, device=self.device),
                      ReLU(),
                      Upsampling(64, 3, 2,  stride = 2, device=self.device),
                      Sigmoid())
```

Fig. 1: Implemented network

An epoch of our training on NVIDIA GeForce RTX 3090 takes about 1 minute and 30 seconds and achieves the PSNR of 23.13dB. For the best model, we run 20 epochs of training which results in a PSNR of **24.28**dB on the validation dataset.

References

[1] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[2] P. Solai, "Convolutions and backpropagations," 2018. [Online]. Available: https://pavisj.medium.com/convolutions-and-backpropagations-46026a8f5d2c

[3] M. Kaushik, "Part 1:backpropagation for convolution with strides," 2019. [Online]. Available: https://medium.com/@mayank.utexas/backpropagation-for-convolution-with-strides-8137e4fc2710

[4] ——, "Part 2:backpropagation for convolution with strides," 2019. [Online]. Available: https://medium.com/@mayank.utexas/backpropagation-for-convolution-with-strides-fb2f2efc4faa

APPENDIX

A. Convolution as a linear operator

In this section, we explain how 2-dimensional convolution and gradients with respect to (w.r.t.) the input, weights, and bias can be expressed as tensor/matrix factorization.

1) Convolution: Let $\boldsymbol{X} \in \mathbb{R}^{3\times 3}$ be the input matrix and $\boldsymbol{W} \in \mathbb{R}^{2\times 2}$ be the kernel matrix. The convolution of $\boldsymbol{X}$ with $\boldsymbol{W}$ reads:

$$\boldsymbol{X} * \boldsymbol{W} = \begin{bmatrix} X_{11}W_{11} + X_{12}W_{12} + X_{21}W_{21} + X_{22}W_{22} & X_{12}W_{11} + X_{13}W_{12} + X_{22}W_{21} + X_{23}W_{22} \\ X_{21}W_{11} + X_{22}W_{12} + X_{31}W_{21} + X_{32}W_{22} & X_{22}W_{11} + X_{23}W_{12} + X_{32}W_{21} + X_{33}W_{22} \end{bmatrix} \tag{6}$$

One can see that, each entry of $\boldsymbol{X} * \boldsymbol{W}$ can be written as the inner product of the vectorization of $\boldsymbol{W}$ (denoted by $\boldsymbol{w} \in \mathbb{R}^4$) and vectorization of 2×2 sub-matrices of $\boldsymbol{X}$, e.g. $\boldsymbol{x}^{(1)} = \mathrm{vec}\Big(\begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \Big) = [X_{11}, X_{12}, X_{21}, X_{22}]^T$.

$$\begin{aligned} \boldsymbol{X} * \boldsymbol{W} &= \begin{bmatrix} [X_{11}, X_{12}, X_{21}, X_{22}].\boldsymbol{w} & [X_{12}, X_{13}, X_{22}, X_{23}].\boldsymbol{w} \\ [X_{21}, X_{22}, X_{31}, X_{32}].\boldsymbol{w} & [X_{22}, X_{22}, X_{32}, X_{33}].\boldsymbol{w} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{w}^T \mathrm{vec}\Big(\begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \Big) & \boldsymbol{w}^T \mathrm{vec}\Big(\begin{bmatrix} X_{12} & X_{13} \\ X_{22} & X_{23} \end{bmatrix} \Big) \\ \boldsymbol{w}^T \mathrm{vec}\Big(\begin{bmatrix} X_{21} & X_{22} \\ X_{31} & X_{32} \end{bmatrix} \Big) & \boldsymbol{w}^T \mathrm{vec}\Big(\begin{bmatrix} X_{22} & X_{23} \\ X_{32} & X_{33} \end{bmatrix} \Big) \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{w}^T \boldsymbol{x}^{(1)} & \boldsymbol{w}^T \boldsymbol{x}^{(2)} \\ \boldsymbol{w}^T \boldsymbol{x}^{(3)} & \boldsymbol{w}^T \boldsymbol{x}^{(4)} \end{bmatrix} \end{aligned} \tag{7}$$

Considering the vectorization of the both sides, we get

$$\mathrm{vec}(\boldsymbol{X} * \boldsymbol{W}) = \boldsymbol{w}^T [\boldsymbol{x}^{(1)} | \boldsymbol{x}^{(2)} | \boldsymbol{x}^{(3)} | \boldsymbol{x}^{(4)}] \tag{8}$$

Therefore, by a linear operation (8) followed by a reshaping ($\mathbb{R}^4 \to \mathbb{R}^{2\times 2}$, which is linear) we can perform a convolution task. This procedure can be done in the same way for the arbitrary size of input and kernel matrices to obtain (1).

- If padding is non-zeros, the input is padded with zeros on all 4 sides, and the new matrix is the input in (8).
- If stride is not one, the sub-matrices (which constructs the columns of the second matrix in the right-hand side of (8)) are chosen $s-1$ distance.
- If dilation is not one, by definition, $d-1$ zeros columns/rows are inserted between columns and rows of $\boldsymbol{W}$, and the convolution operation can be performed as in (8).

Example 1. *Consider $\boldsymbol{X} \in \mathbb{R}^{2\times 2}, \boldsymbol{W} \in \mathbb{R}^{2\times 2}$, and the convolution is with $p = 1, s = 2, d = 1$. Denote the padded input by $\boldsymbol{X}^{(\mathrm{p})} \in \mathbb{R}^{4\times 4}$*

$$\begin{aligned} \boldsymbol{X}^{(\mathrm{p})} * \boldsymbol{W} &= \begin{bmatrix} [X_{11}^{(\mathrm{p})}, X_{12}^{(\mathrm{p})}, X_{21}^{(\mathrm{p})}, X_{22}^{(\mathrm{p})}].\boldsymbol{w} & [X_{13}^{(\mathrm{p})}, X_{14}^{(\mathrm{p})}, X_{23}^{(\mathrm{p})}, X_{24}^{(\mathrm{p})}].\boldsymbol{w} \\ [X_{31}^{(\mathrm{p})}, X_{32}^{(\mathrm{p})}, X_{41}^{(\mathrm{p})}, X_{42}^{(\mathrm{p})}].\boldsymbol{w} & [X_{33}^{(\mathrm{p})}, X_{34}^{(\mathrm{p})}, X_{43}^{(\mathrm{p})}, X_{44}^{(\mathrm{p})}].\boldsymbol{w} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{w}^T \mathrm{vec}\Big(\begin{bmatrix} X_{11}^{(\mathrm{p})} & X_{12}^{(\mathrm{p})} \\ X_{21}^{(\mathrm{p})} & X_{22}^{(\mathrm{p})} \end{bmatrix} \Big) & \boldsymbol{w}^T \mathrm{vec}\Big(\begin{bmatrix} X_{13}^{(\mathrm{p})} & X_{14}^{(\mathrm{p})} \\ X_{23}^{(\mathrm{p})} & X_{24}^{(\mathrm{p})} \end{bmatrix} \Big) \\ \boldsymbol{w}^T \mathrm{vec}\Big(\begin{bmatrix} X_{31}^{(\mathrm{p})} & X_{32}^{(\mathrm{p})} \\ X_{41}^{(\mathrm{p})} & X_{42}^{(\mathrm{p})} \end{bmatrix} \Big) & \boldsymbol{w}^T \mathrm{vec}\Big(\begin{bmatrix} X_{33}^{(\mathrm{p})} & X_{34}^{(\mathrm{p})} \\ X_{43}^{(\mathrm{p})} & X_{44}^{(\mathrm{p})} \end{bmatrix} \Big) \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{w}^T \boldsymbol{x}^{(\mathrm{p})(1)} & \boldsymbol{w}^T \boldsymbol{x}^{(\mathrm{p})(2)} \\ \boldsymbol{w}^T \boldsymbol{x}^{(\mathrm{p})(3)} & \boldsymbol{w}^T \boldsymbol{x}^{(\mathrm{p})(4)} \end{bmatrix} \end{aligned}$$

2) Gradient w.r.t. weights: Denote the output of the convolution by $\boldsymbol{O} = \boldsymbol{X} * \boldsymbol{W} (+ b \times \boldsymbol{J})$. In the case explained in previous section, let $\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}$ be the gradient of loss with respect to (w.r.t.) $\boldsymbol{O}$, where $\big(\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}\big)_{i,j} = \frac{\partial \mathcal{L}}{\partial O_{i,j}}$. Then, using chain rule, gradient of the loss w.r.t. the kernel reads

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}} &= \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial W_{11}} & \frac{\partial \mathcal{L}}{\partial W_{12}} \\ \frac{\partial \mathcal{L}}{\partial W_{21}} & \frac{\partial \mathcal{L}}{\partial W_{22}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial O_{11}} \frac{\partial O_{11}}{\partial W_{11}} + \frac{\partial \mathcal{L}}{\partial O_{12}} \frac{\partial O_{12}}{\partial W_{11}} + \frac{\partial \mathcal{L}}{\partial O_{21}} \frac{\partial O_{21}}{\partial W_{11}} + \frac{\partial \mathcal{L}}{\partial O_{22}} \frac{\partial O_{22}}{\partial W_{11}} & \frac{\partial \mathcal{L}}{\partial O_{11}} \frac{\partial O_{11}}{\partial W_{12}} + \frac{\partial \mathcal{L}}{\partial O_{12}} \frac{\partial O_{12}}{\partial W_{12}} + \frac{\partial \mathcal{L}}{\partial O_{21}} \frac{\partial O_{21}}{\partial W_{12}} + \frac{\partial \mathcal{L}}{\partial O_{22}} \frac{\partial O_{22}}{\partial W_{12}} \\ \frac{\partial \mathcal{L}}{\partial O_{11}} \frac{\partial O_{11}}{\partial W_{21}} + \frac{\partial \mathcal{L}}{\partial O_{12}} \frac{\partial O_{12}}{\partial W_{21}} + \frac{\partial \mathcal{L}}{\partial O_{21}} \frac{\partial O_{21}}{\partial W_{21}} + \frac{\partial \mathcal{L}}{\partial O_{22}} \frac{\partial O_{22}}{\partial W_{21}} & \frac{\partial \mathcal{L}}{\partial O_{11}} \frac{\partial O_{11}}{\partial W_{22}} + \frac{\partial \mathcal{L}}{\partial O_{12}} \frac{\partial O_{12}}{\partial W_{22}} + \frac{\partial \mathcal{L}}{\partial O_{21}} \frac{\partial O_{21}}{\partial W_{22}} + \frac{\partial \mathcal{L}}{\partial O_{22}} \frac{\partial O_{22}}{\partial W_{22}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial O_{11}} X_{11} + \frac{\partial \mathcal{L}}{\partial O_{12}} X_{12} + \frac{\partial \mathcal{L}}{\partial O_{21}} X_{21} + \frac{\partial \mathcal{L}}{\partial O_{22}} X_{22} & \frac{\partial \mathcal{L}}{\partial O_{11}} X_{12} + \frac{\partial \mathcal{L}}{\partial O_{12}} X_{13} + \frac{\partial \mathcal{L}}{\partial O_{21}} X_{22} + \frac{\partial \mathcal{L}}{\partial O_{22}} X_{23} \\ \frac{\partial \mathcal{L}}{\partial O_{11}} X_{21} + \frac{\partial \mathcal{L}}{\partial O_{12}} X_{22} + \frac{\partial \mathcal{L}}{\partial O_{21}} X_{31} + \frac{\partial \mathcal{L}}{\partial O_{22}} X_{32} & \frac{\partial \mathcal{L}}{\partial O_{11}} X_{22} + \frac{\partial \mathcal{L}}{\partial O_{12}} X_{23} + \frac{\partial \mathcal{L}}{\partial O_{21}} X_{32} + \frac{\partial \mathcal{L}}{\partial O_{22}} X_{33} \end{bmatrix} \end{aligned} \tag{9}$$

Comparing with (6), one can see that they have the same form. Therefore, gradient of the loss w.r.t. to the kernel weights can be expressed as the convolution of the input ($\boldsymbol{X}$) with the carried gradient matrix ($\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}$).

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}} = \boldsymbol{X} * \frac{\partial \mathcal{L}}{\partial \boldsymbol{O}} \tag{10}$$

and the convolution operation can be carried out as discussed before with the following considerations (note that the above convolution should be performed with stride equal to one in all the cases):

- If padding (of the original convolution operation) is non-zeros, the matrix $\boldsymbol{X}$ in (10) should be the padded input.
- If stride is not one, writing the gradient explicitly as in (9) for all elements, one can see that some terms vanish due to the stride of the original convolution operation. Therefore, the convolution should be performed with the dilated version of $\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}$ with d equal to the stride of the original convolution operation, see [4] for more details. *Moreover, note that along dilating $\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}$ one might need to add zeros columns/rows to it to match the sizes for the convolution in* (10).
- If dilation (of the original convolution operation) is not one, the left-hand side (LHS) of (10) is the gradient of the loss w.r.t. the dilated kernel (zeros between rows and columns), so the gradient w.r.t. the original kernel should be extracted properly from the matrix in the LHS.

Example 2. *Consider $\boldsymbol{X} \in \mathbb{R}^{2\times 2}, \boldsymbol{W} \in \mathbb{R}^{2\times 2}$, and the convolution is with $p = 1, s = 2, d = 1$. Denote the padded input by $\boldsymbol{X}^{(\mathrm{p})} \in \mathbb{R}^{4\times 4}$, then the gradient w.r.t. weights can written as*

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}} &= \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial W_{11}} & \frac{\partial \mathcal{L}}{\partial W_{12}} \\ \frac{\partial \mathcal{L}}{\partial W_{21}} & \frac{\partial \mathcal{L}}{\partial W_{22}} \end{bmatrix} \\
&= \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial O_{11}}\frac{\partial O_{11}}{\partial W_{11}} + \frac{\partial \mathcal{L}}{\partial O_{12}}\frac{\partial O_{12}}{\partial W_{11}} + \frac{\partial \mathcal{L}}{\partial O_{21}}\frac{\partial O_{21}}{\partial W_{11}} + \frac{\partial \mathcal{L}}{\partial O_{22}}\frac{\partial O_{22}}{\partial W_{11}} & \frac{\partial \mathcal{L}}{\partial O_{11}}\frac{\partial O_{11}}{\partial W_{12}} + \frac{\partial \mathcal{L}}{\partial O_{12}}\frac{\partial O_{12}}{\partial W_{12}} + \frac{\partial \mathcal{L}}{\partial O_{21}}\frac{\partial O_{21}}{\partial W_{12}} + \frac{\partial \mathcal{L}}{\partial O_{22}}\frac{\partial O_{22}}{\partial W_{12}} \\ \frac{\partial \mathcal{L}}{\partial O_{11}}\frac{\partial O_{11}}{\partial W_{21}} + \frac{\partial \mathcal{L}}{\partial O_{12}}\frac{\partial O_{12}}{\partial W_{21}} + \frac{\partial \mathcal{L}}{\partial O_{21}}\frac{\partial O_{21}}{\partial W_{21}} + \frac{\partial \mathcal{L}}{\partial O_{22}}\frac{\partial O_{22}}{\partial W_{21}} & \frac{\partial \mathcal{L}}{\partial O_{11}}\frac{\partial O_{11}}{\partial W_{22}} + \frac{\partial \mathcal{L}}{\partial O_{12}}\frac{\partial O_{12}}{\partial W_{22}} + \frac{\partial \mathcal{L}}{\partial O_{21}}\frac{\partial O_{21}}{\partial W_{22}} + \frac{\partial \mathcal{L}}{\partial O_{22}}\frac{\partial O_{22}}{\partial W_{22}} \end{bmatrix} \\
&= \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial O_{11}}X^{(\mathrm{p})}_{11} + \frac{\partial \mathcal{L}}{\partial O_{12}}X^{(\mathrm{p})}_{13} + \frac{\partial \mathcal{L}}{\partial O_{21}}X^{(\mathrm{p})}_{31} + \frac{\partial \mathcal{L}}{\partial O_{22}}X^{(\mathrm{p})}_{33} & \frac{\partial \mathcal{L}}{\partial O_{11}}X^{(\mathrm{p})}_{12} + \frac{\partial \mathcal{L}}{\partial O_{12}}X^{(\mathrm{p})}_{14} + \frac{\partial \mathcal{L}}{\partial O_{21}}X^{(\mathrm{p})}_{32} + \frac{\partial \mathcal{L}}{\partial O_{22}}X^{(\mathrm{p})}_{34} \\ \frac{\partial \mathcal{L}}{\partial O_{11}}X^{(\mathrm{p})}_{21} + \frac{\partial \mathcal{L}}{\partial O_{12}}X^{(\mathrm{p})}_{23} + \frac{\partial \mathcal{L}}{\partial O_{21}}X^{(\mathrm{p})}_{41} + \frac{\partial \mathcal{L}}{\partial O_{22}}X^{(\mathrm{p})}_{43} & \frac{\partial \mathcal{L}}{\partial O_{11}}X^{(\mathrm{p})}_{22} + \frac{\partial \mathcal{L}}{\partial O_{12}}X^{(\mathrm{p})}_{24} + \frac{\partial \mathcal{L}}{\partial O_{21}}X^{(\mathrm{p})}_{42} + \frac{\partial \mathcal{L}}{\partial O_{22}}X^{(\mathrm{p})}_{44} \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & X_{11} & X_{12} & 0 \\ 0 & X_{21} & X_{22} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial O_{11}} & 0 & \frac{\partial \mathcal{L}}{\partial O_{12}} \\ 0 & 0 & 0 \\ \frac{\partial \mathcal{L}}{\partial O_{21}} & 0 & \frac{\partial \mathcal{L}}{\partial O_{22}} \end{bmatrix} \\
&= \boldsymbol{X}^{(\mathrm{p})} * \frac{\partial \mathcal{L}}{\partial \boldsymbol{O}} \qquad \textit{with dilation} = 2
\end{aligned}$$

3) Gradient w.r.t. bias: The gradient of the loss w.r.t. the bias term can be easily written as

$$\frac{\partial \mathcal{L}}{\partial b} = \boldsymbol{J} * \frac{\partial \mathcal{L}}{\partial \boldsymbol{O}} = \sum_{i,j} \frac{\partial \mathcal{L}}{\partial O_{i,j}} \tag{11}$$

where $\boldsymbol{J}$ is the all-one matrix of the same size as $\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}$.

4) Gradient w.r.t. input: Consider the case explained in previous sections, $\boldsymbol{X} \in \mathbb{R}^{3\times 3}, \boldsymbol{W} \in \mathbb{R}^{2\times 2}$. To compute the gradient of the loss w.r.t. the input $\boldsymbol{X}$, using chain rule, we can write the gradient explicitly as

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \boldsymbol{X}} &= \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial X_{11}} & \frac{\partial \mathcal{L}}{\partial X_{12}} & \frac{\partial \mathcal{L}}{\partial X_{13}} \\ \frac{\partial \mathcal{L}}{\partial X_{21}} & \frac{\partial \mathcal{L}}{\partial X_{22}} & \frac{\partial \mathcal{L}}{\partial X_{23}} \\ \frac{\partial \mathcal{L}}{\partial X_{31}} & \frac{\partial \mathcal{L}}{\partial X_{32}} & \frac{\partial \mathcal{L}}{\partial X_{33}} \end{bmatrix} \\
&= \begin{bmatrix} \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}}\frac{\partial O_{ij}}{\partial X_{11}} & \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}}\frac{\partial O_{ij}}{\partial X_{12}} & \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}}\frac{\partial O_{ij}}{\partial X_{13}} \\ \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}}\frac{\partial O_{ij}}{\partial X_{21}} & \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}}\frac{\partial O_{ij}}{\partial X_{22}} & \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}}\frac{\partial O_{ij}}{\partial X_{23}} \\ \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}}\frac{\partial O_{ij}}{\partial X_{31}} & \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}}\frac{\partial O_{ij}}{\partial X_{32}} & \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}}\frac{\partial O_{11}}{\partial X_{33}} \end{bmatrix} \\
&= \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial O_{11}}W_{11} & \frac{\partial \mathcal{L}}{\partial O_{11}}W_{12} + \frac{\partial \mathcal{L}}{\partial O_{12}}W_{11} & \frac{\partial \mathcal{L}}{\partial O_{12}}W_{12} \\ \frac{\partial \mathcal{L}}{\partial O_{11}}W_{21} + \frac{\partial \mathcal{L}}{\partial O_{21}}W_{11} & \frac{\partial \mathcal{L}}{\partial O_{11}}W_{22} + \frac{\partial \mathcal{L}}{\partial O_{12}}W_{21} + \frac{\partial \mathcal{L}}{\partial O_{21}}W_{12} + \frac{\partial \mathcal{L}}{\partial O_{22}}W_{11} & \frac{\partial \mathcal{L}}{\partial O_{12}}W_{22} + \frac{\partial \mathcal{L}}{\partial O_{22}}W_{12} \\ \frac{\partial \mathcal{L}}{\partial O_{21}}W_{21} & \frac{\partial \mathcal{L}}{\partial O_{21}}W_{22} + \frac{\partial \mathcal{L}}{\partial O_{22}}W_{21} & \frac{\partial \mathcal{L}}{\partial O_{22}}W_{22} \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{\partial \mathcal{L}}{\partial O_{11}} & \frac{\partial \mathcal{L}}{\partial O_{12}} & 0 \\ 0 & \frac{\partial \mathcal{L}}{\partial O_{21}} & \frac{\partial \mathcal{L}}{\partial O_{22}} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} W_{22} & W_{21} \\ W_{12} & W_{11} \end{bmatrix}
\end{aligned} \tag{12}$$

Therefore, we have:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{X}} = \big(\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}\big)^{\tilde{(\mathrm{p})}} * \boldsymbol{W}^0 \tag{13}$$

where $\boldsymbol{W}^0$ is the rotated version of $\boldsymbol{W}$. $\big(\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}\big)^{\tilde{(\mathrm{p})}}$ is the padded version of $\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}$, the padding on dimensions is not the same and is done to match the size for the convolution.

- If padding (of the original convolution operation) is non-zero, the matrix in LHS of (13) is the gradient of the loss w.r.t. the padded input, so the gradient w.r.t. the input should be extracted properly from the matrix in the LHS.
- If stride is not one, writing the gradient explicitly as in (12), one can see that some terms vanish due to the stride of the original convolution operation, still, the gradient w.r.t. the input can be written as (13) but first $\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}$ should be dilated with d equal to the stride of the original convolution operation, and then padded properly to match the size.
- If dilation (of the original convolution operation) is not one, the convolution in (13) is performed with the dilated (then rotated) version of $\boldsymbol{W}$.

Example 3. *Consider $\boldsymbol{X} \in \mathbb{R}^{2\times 2}, \boldsymbol{W} \in \mathbb{R}^{2\times 2}$, and the convolution is with $p = 1, s = 2, d = 1$. Denote the padded input by $\boldsymbol{X}^{(\mathrm{p})} \in \mathbb{R}^{4\times 4}$, then the gradient w.r.t. weights can written as*

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \boldsymbol{X}^{(\mathrm{p})}} &= \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial X^{(\mathrm{p})}_{11}} & \frac{\partial \mathcal{L}}{\partial X^{(\mathrm{p})}_{12}} & \frac{\partial \mathcal{L}}{\partial X^{(\mathrm{p})}_{13}} & \frac{\partial \mathcal{L}}{\partial X^{(\mathrm{p})}_{14}} \\ \frac{\partial \mathcal{L}}{\partial X^{(\mathrm{p})}_{21}} & \frac{\partial \mathcal{L}}{\partial X^{(\mathrm{p})}_{22}} & \frac{\partial \mathcal{L}}{\partial X^{(\mathrm{p})}_{23}} & \frac{\partial \mathcal{L}}{\partial X^{(\mathrm{p})}_{24}} \\ \frac{\partial \mathcal{L}}{\partial X^{(\mathrm{p})}_{31}} & \frac{\partial \mathcal{L}}{\partial X^{(\mathrm{p})}_{32}} & \frac{\partial \mathcal{L}}{\partial X^{(\mathrm{p})}_{33}} & \frac{\partial \mathcal{L}}{\partial X^{(\mathrm{p})}_{34}} \\ \frac{\partial \mathcal{L}}{\partial X^{(\mathrm{p})}_{41}} & \frac{\partial \mathcal{L}}{\partial X^{(\mathrm{p})}_{42}} & \frac{\partial \mathcal{L}}{\partial X^{(\mathrm{p})}_{43}} & \frac{\partial \mathcal{L}}{\partial X^{(\mathrm{p})}_{44}} \end{bmatrix} \\
&= \begin{bmatrix} \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}} \frac{\partial O_{ij}}{\partial X^{(\mathrm{p})}_{11}} & \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}} \frac{\partial O_{ij}}{\partial X^{(\mathrm{p})}_{12}} & \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}} \frac{\partial O_{ij}}{\partial X^{(\mathrm{p})}_{13}} & \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}} \frac{\partial O_{ij}}{\partial X^{(\mathrm{p})}_{14}} \\ \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}} \frac{\partial O_{ij}}{\partial X^{(\mathrm{p})}_{21}} & \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}} \frac{\partial O_{ij}}{\partial X^{(\mathrm{p})}_{22}} & \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}} \frac{\partial O_{ij}}{\partial X^{(\mathrm{p})}_{23}} & \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}} \frac{\partial O_{ij}}{\partial X^{(\mathrm{p})}_{24}} \\ \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}} \frac{\partial O_{ij}}{\partial X^{(\mathrm{p})}_{31}} & \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}} \frac{\partial O_{ij}}{\partial X^{(\mathrm{p})}_{32}} & \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}} \frac{\partial O_{ij}}{\partial X^{(\mathrm{p})}_{33}} & \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}} \frac{\partial O_{ij}}{\partial X^{(\mathrm{p})}_{34}} \\ \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}} \frac{\partial O_{ij}}{\partial X^{(\mathrm{p})}_{41}} & \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}} \frac{\partial O_{ij}}{\partial X^{(\mathrm{p})}_{42}} & \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}} \frac{\partial O_{ij}}{\partial X^{(\mathrm{p})}_{43}} & \sum_{i,j=1}^{2} \frac{\partial \mathcal{L}}{\partial O_{ij}} \frac{\partial O_{ij}}{\partial X^{(\mathrm{p})}_{44}} \end{bmatrix} \\
&= \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial O_{11}} W_{11} & \frac{\partial \mathcal{L}}{\partial O_{11}} W_{12} & \frac{\partial \mathcal{L}}{\partial O_{12}} W_{11} & \frac{\partial \mathcal{L}}{\partial O_{12}} W_{12} \\ \frac{\partial \mathcal{L}}{\partial O_{11}} W_{21} & \frac{\partial \mathcal{L}}{\partial O_{11}} W_{22} & \frac{\partial \mathcal{L}}{\partial O_{12}} W_{21} & \frac{\partial \mathcal{L}}{\partial O_{12}} W_{22} \\ \frac{\partial \mathcal{L}}{\partial O_{21}} W_{11} & \frac{\partial \mathcal{L}}{\partial O_{21}} W_{12} & \frac{\partial \mathcal{L}}{\partial O_{22}} W_{11} & \frac{\partial \mathcal{L}}{\partial O_{22}} W_{12} \\ \frac{\partial \mathcal{L}}{\partial O_{21}} W_{12} & \frac{\partial \mathcal{L}}{\partial O_{21}} W_{22} & \frac{\partial \mathcal{L}}{\partial O_{22}} W_{21} & \frac{\partial \mathcal{L}}{\partial O_{22}} W_{22} \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\partial \mathcal{L}}{\partial O_{11}} & 0 & \frac{\partial \mathcal{L}}{\partial O_{12}} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\partial \mathcal{L}}{\partial O_{21}} & 0 & \frac{\partial \mathcal{L}}{\partial O_{22}} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} W_{22} & W_{21} \\ W_{12} & W_{11} \end{bmatrix}
\end{aligned}$$

B. Transposed convolution

In this section, we discuss how transposed convolution can be expressed as usual convolution operation, and gradients can be carried out using the previous section.

1) Transposed convolution: Consider the matrix $\boldsymbol{X} \in \mathbb{R}^{2\times 2}$ as the input and $\boldsymbol{W} \in \mathbb{R}^{2\times 2}$ as the kernel, we have:

$$\begin{aligned}
\boldsymbol{X} *^T \boldsymbol{W} &= \begin{bmatrix} X_{11}W_{11} & X_{11}W_{12} + X_{12}W_{11} & X_{12}W_{12} \\ X_{11}W_{21} + X_{21}W_{11} & X_{11}W_{22} + X_{12}W_{21} + X_{21}W_{12} + X_{22}W_{11} & X_{12}W_{22} + X_{22}W_{12} \\ X_{21}W_{21} & X_{21}W_{22} + X_{22}W_{21} & X_{22}W_{22} \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & X_{11} & X_{12} & 0 \\ 0 & X_{21} & X_{22} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} W_{22} & W_{21} \\ W_{12} & W_{11} \end{bmatrix}
\end{aligned} \tag{14}$$

Generalizing the idea for arbitrary size of input and kernel, we obtain:

$$\boldsymbol{X} *^T \boldsymbol{W} = \boldsymbol{X}^{(\mathrm{p})} * \boldsymbol{W}^0 \tag{15}$$

where $\boldsymbol{X}^{(\mathrm{p})}$ is the properly padded input, and $\boldsymbol{W}^0$ is the $180°$-rotated version of kernel weights. Note that the above convolution should be performed with a stride equal to one in all the cases.

- For the convolution in (15), the input is padded by $k-p-1$ on each side, where k is the kernel-size and p is the padding.
- If stride is not one, zero columns/rows of width $s-1$ are inserted between columns/rows of the input. Then, it will padded according to the previous point.
- If dilation is not one, by definition, $d-1$ zeros columns/rows are inserted between columns and rows of $\boldsymbol{W}$, and the convolution operation can be performed as in (15).

2) Gradient w.r.t. weights: Denote the output of the convolution by $\boldsymbol{O} = \boldsymbol{X} * \boldsymbol{W}(+b \times \boldsymbol{J})$. As explained in A2, the gradient w.r.t. (rotated) the weights can be written as:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^0} = \boldsymbol{X}^{(\mathrm{p})} * \frac{\partial \mathcal{L}}{\partial \boldsymbol{O}} \tag{16}$$

So, the gradient w.r.t. weights can be extracted by rotating the matrix in LHS of (16).

- $\boldsymbol{X}^{(\mathrm{p})}$ is the properly padded matrix as in (15).
- If stride is not one, zero columns/rows of width $s-1$ are inserted between columns/rows of the input. Then, it will padded according to the previous point. (As in (15))
- If dilation (of the original convolution operation) is not one, the LHS of (16) is the gradient w.r.t. dilated , rotated kernel, so the the gradient should be extracted properly. (By first rotating then removing the columns/rows.)

3) Gradient w.r.t. bias: The gradient of the loss w.r.t. the bias term can be easily written as

$$\frac{\partial \mathcal{L}}{\partial b} = \boldsymbol{J} * \frac{\partial \mathcal{L}}{\partial \boldsymbol{O}} = \sum_{i,j} \frac{\partial \mathcal{L}}{\partial O_{i,j}} \tag{17}$$

where $\boldsymbol{J}$ is the all-one matrix of the same size as $\frac{\partial \mathcal{L}}{\partial \boldsymbol{O}}$.

4) Gradient w.r.t. input: The gradient w.r.t. the input can be written as:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{X}^{(\mathrm{p})}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{O}} * \boldsymbol{W} \tag{18}$$

where $\boldsymbol{X}^{(\mathrm{p})}$ is the padded input. Thus, the gradient should be extracted properly from the LHS of (18).

- If stride is not one, after removing the extra columns/rows from the LHS (which are due to the padding), then inserted zero columns/rows should also be removed.
- If dilation (of the original convolution operation) is not one, then $\boldsymbol{W}$ is the dilated kernel.