Some formulae

Course Project: Deep Learning

Name:	Matteo Calafà, Paolo Motta, Thomas Rimbot
Date:	May 21, 2022
Course:	Deep Learning EE-559
Instructor:	François Fleuret

1 Convolutional layer as linear layer

Take into consideration one single convolutional layer. Define:

- I as the 4D input tensor.
- W as the 2D weight tensor of the convolutional layer (that depends on the various parameters: kernel size, stride, padding ...).
- B as the 1D bias tensor of the convolutional layer (that depends on the various parameters: kernel size, stride, padding ...).
- O as the 4D output tensor.

Moreover, define:

- U as the 3D unfolded tensor of I.
- $\tilde{W}$ the 2D reshaped W according to the view `(out_channels, -1)`.
- $\tilde{B}$ the 3D reshaped B according to the view `(1, -1, 1)`.

Therefore, it holds:

$$\tilde{O} = \tilde{W} \otimes U + \tilde{B} \tag{1}$$

Explicitly in coordinates:

$$\tilde{O}_{p,q,r} = \sum_m \tilde{W}_{q,m} U_{p,m,r} + \tilde{B}_{0,q,0} \tag{2}$$

(in the instructions, the tensor product is directly calculated with `@`).

1.1 Gradient with respect to input

Suppose G is the gradient of the loss with respect to the output (therefore, same shape of O). We first need to pass from G to $\tilde{G}$ that is the same but with the same shape of $\tilde{O}$. Then, the (i,j,k) component of the gradient with respect to the unfolded input is:

$$\frac{\partial \mathcal{L}}{\partial U_{i,j,k}} = \sum_{p,q,r} \frac{\partial \mathcal{L}}{\partial \tilde{O}_{p,q,r}} \frac{\partial \tilde{O}_{p,q,r}}{\partial U_{i,j,k}} \overset{Def}{=} \sum_{p,q,r} \tilde{G}_{p,q,r} \frac{\partial \tilde{O}_{p,q,r}}{\partial U_{i,j,k}} \overset{(2)}{=} \sum_{p,q,r} \tilde{G}_{p,q,r} \tilde{W}_{q,j} \delta_{p,i} \delta_{r,k} = \sum_q \tilde{G}_{i,q,k} \tilde{W}_{q,j} \tag{3}$$

This product can be calculated with some `transpose` operations and then `torch.tensordot` or `@` (paying attention in the latter case). In this way, we get the derivative with respect to the unfolded input. To get the gradient with respect to the original input, since the unfold operation is a multiple copy operation, it is just needed to sum different components using the principle of *weight sharing*. Fortunately, this is already done by the `fold` operation as you can check in `check_fold_as_weight_sharing.py`.

$$\boxed{G \overset{\texttt{view}}{\to} \tilde{G} \overset{(3)}{\to} \frac{\partial \mathcal{L}}{\partial U} \overset{\texttt{fold}}{\to} \frac{\partial \mathcal{L}}{\partial I}}$$

1.2 Gradient with respect to weight

The (i,j) component of the gradient with respect to the reshaped weight $\tilde{W}$ is:

$$\frac{\partial \mathcal{L}}{\partial \tilde{W}_{i,j}} = \sum_{p,q,r} \frac{\partial \mathcal{L}}{\partial \tilde{O}_{p,q,r}} \frac{\partial \tilde{O}_{p,q,r}}{\partial \tilde{W}_{i,j}} \overset{Def}{=} \sum_{p,q,r} \tilde{G}_{p,q,r} \frac{\partial \tilde{O}_{p,q,r}}{\partial \tilde{W}_{i,j}} \overset{(2)}{=} \sum_{p,q,r} \tilde{G}_{p,q,r} U_{p,j,r} \delta_{q,i} = \sum_{p,r} \tilde{G}_{p,i,r} U_{p,j,r} \tag{4}$$

Again, this product can be calculated with `torch.tensordot` or `@` (paying attention in this latter case). To get instead the derivative with respect to the original weight, it is just needed to reshape this gradient to the original weight shape.

$$\boxed{G \overset{\texttt{view}}{\rightarrow} \tilde{G} \overset{(4)}{\rightarrow} \frac{\partial \mathcal{L}}{\partial \tilde{W}} \overset{\texttt{view}}{\rightarrow} \frac{\partial \mathcal{L}}{\partial W}}$$

1.3 Gradient with respect to bias

Since $\tilde{B}$ is just the reshaped version of B with view `(1,-1,1)` we can directly compute the gradient with respect to B as:

$$\frac{\partial \mathcal{L}}{\partial B_i} = \frac{\partial \mathcal{L}}{\partial \tilde{B}_{0,i,0}}$$

Then,

$$\frac{\partial \mathcal{L}}{\partial B_i} = \frac{\partial \mathcal{L}}{\partial \tilde{B}_{0,i,0}} = \sum_{p,q,r} \frac{\partial \mathcal{L}}{\partial \tilde{O}_{p,q,r}} \frac{\partial \tilde{O}_{p,q,r}}{\partial \tilde{B}_{0,i,0}} \overset{Def}{=} \sum_{p,q,r} \tilde{G}_{p,q,r} \frac{\partial \tilde{O}_{p,q,r}}{\partial \tilde{B}_{0,i,0}} \overset{(2)}{=} \sum_{p,q,r} \tilde{G}_{p,q,r} \delta_{q,i} = \sum_{p,r} \tilde{G}_{p,i,r} \tag{5}$$

This term can be calculated with `Tensor.sum()`.

$$\boxed{G \overset{\texttt{view}}{\rightarrow} \tilde{G} \overset{(5)}{\rightarrow} \frac{\partial \mathcal{L}}{\partial B}}$$