EECS 182 Deep Neural Networks Fall 2023 Anant Sahai Discussion 4

1. Weight Sharing in CNN

In this question we will look at the mechanism of weight sharing in convolutions. Let's start with a 1-dimension example. Suppose that we have a 9 dimensional input vector and compute a 1D convolution with the kernel filter that has 3 weights (parameters).

$$\mathbf{k} = \begin{bmatrix} k_1 & k_2 & k_3 \end{bmatrix}^T, \mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & ... & x_9 \end{bmatrix}^T$$

(a) What's the **output dimension** if we apply filter $\mathbf{k}$ with no padding and stride of 1? What's the **first element** of the output? What's the **last element**?

(b) What's the **output dimension** if we apply $\mathbf{k}$ with "same padding" and stride of 1? What's the **first element** of the output? What's the **last element**?

(c) Recall that in lecture, we discussed that CNN filters have the property of **weight sharing**, meaning that different portions of an image can share the same weight to extract the same set of features. Turns out convolution is a linear operation and we can express it in the form of linear layers, ie. $\mathbf{x'} = \mathbf{Kx}$ (assume the bias term is zero).

Find $\mathbf{K}$, the linear transformation matrix corresponding to the convolution applied in part (a). (*Hint: What is the dimension of* $\mathbf{K}$*?*)

(d) **Find K for part (b), where we apply "same padding".** (*Hint: You should only be adding rows to your answer in part c.*)

(e) Suppose that we no longer want to share weights spatially over the input, ie. we go though the same mechanics as convolution "sliding window", but for different locations within the input, we apply different kernel. **How does this change our matrix? How many weights do we have now?**

(f) We want to know the general formula for computing the output dimension of a convolution operation. Suppose we have a square input image of dimension $W \times W$ and a $K \times K$ kernel filter. If we assume stride of 1 and no padding, **what's the output dimension W' ? What if we applied stride of s and padding of size p, how would the dimension change?**

Let's take what we've learnt into actual applications on image tasks. Suppose our input is a 256 by 256 RGB image. We are also given a set of 32 filters, each with kernel size of 5.

(g) Conventionally in frameworks such as PyTorch, images are 3D tensors arranged in the format of `[channels, height, width]`. In practice, it's more common to have an additional `batch_size` dimension at the front, but here we ignore that to simplify the math. **What's the shape our input tensor**? **What's the shape of each kernel filter**? (*Hint: Both your answers should have 3 dimensions.)*

(h) Now apply convolution on our image tensor with no padding and stride of 2. **What is the output tensor's dimension?** Considering all kernel filters, **how many weights do we have**? Had we not use CNN but MLP instead (with flattened image), **how many weights does that linear layer contain?** Feel free to use a calculator for this question.

2. Coding Question: Principles of CNN

Look at the CNNPrinciples.ipynb. In this notebook, you'll study principles and properties of CNN. You will see the three problems below in the notebook as well. Write down your answers in the notebook.

(a) Report the result. Which model performs better? Explain why.

(b) What do you observe? Why is it different from the case of MLP?

(c) Note that even though CNN is not trained, the feature maps not only are still translationally equivariant but also extract a quite good features. Why is it so?

(d) Explain the results. Why do you think the performance is better than the MLP?

Contributors:

- Jake Austin.
- Suhong Moon.
- Kevin Li.
- Anant Sahai.