EECS 182 Deep Neural Networks
Fall 2025 Anant Sahai and Gireeja Ranade

Discussion 5

1. Coding Question: Principles of CNN

Open the notebook at https://tinyurl.com/cs182-dis05-code. In this notebook, you'll study principles and properties of CNN. Run the code cells and answer the written questions in the notebook.

Note: The second part of this question may take a few minutes to train. Feel free to move on to the next question while you wait for the training to finish.

Solution: Solutions to the notebook questions:

Part A: Translational Equivariance

i The output of the CNN admits a similar pattern compared to the image input of the CNN. Any translation in the input image also translates to the output of the CNN. In contrast, the output from the MLP is not very interpretable. The key difference between CNN and MLP is that CNN is translationally equivariant, while MLP is not. This is because CNN has parameter sharing spatially.

ii Such properties are from the architecture of CNN not from the trained weights. We will explore the inductive biases of CNNs more in the homework this week.

Part B: Training CNN and MLP on CIFAR10

i CNN performs better than MLP. This is because CNN's inductive biases: sparse interactions, parameter sharing, and translational equivariance, are more suitable for the vision modality than MLP.

ii See the solution in the notebook.

2. 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).

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

(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**?

Solution: The output dimension is 9 - 3 + 1 = 7. The first element is $k_1x_1 + k_2x_2 + k_3x_3$ and the last element is $k_1x_7 + k_2x_8 + k_3x_9$.

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

Solution: The output dimension is 9 (same as the input dimension) by the definition of same padding. To accomplish this, we would pad the input vector with a single zero at the front of the vector and a single zero at the back of the vector before applying the filter. Therefore, the first element is $k_2x_1 + k_3x_2$ and the last element is $k_1x_8 + k_2x_9$.

(c) Convolution is a linear operation and we can express it in the form of linear layers, ie. $\mathbf{h} = \mathbf{K}\mathbf{x}$ (assume the bias term is zero). 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.

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

Solution: This is a Topelitz matrix corresponding to discrete convolution. The output size shrinks by k-1, which is 2 in this case, so $\mathbf{K}$ has a dimension of $\mathbb{R}^{7\times 9}$.

$$\begin{bmatrix} k_1 & k_2 & k_3 & 0 & 0 & \dots & 0 \\ 0 & k_1 & k_2 & k_3 & 0 & \dots & 0 \\ & & & \vdots & & & \\ 0 & \dots & 0 & k_1 & k_2 & k_3 & 0 \\ 0 & \dots & 0 & 0 & k_1 & k_2 & k_3 \end{bmatrix}$$

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

Solution: Since the only difference is the "same padding", we will be adding one row at the beginning and the end respectively. The output size is $\mathbb{R}^{9\times 9}$.

$$\begin{bmatrix} k_2 & k_3 & 0 & 0 & 0 & \dots & 0 \\ k_1 & k_2 & k_3 & 0 & 0 & \dots & 0 \\ 0 & k_1 & k_2 & k_3 & 0 & \dots & 0 \\ & & & \vdots & & & \\ 0 & \dots & 0 & k_1 & k_2 & k_3 & 0 \\ 0 & \dots & 0 & 0 & k_1 & k_2 & k_3 \\ 0 & \dots & 0 & 0 & 0 & k_1 & k_2 \end{bmatrix}$$

(e) Suppose that we no longer used weight sharing: for each location within the input, we apply a different filter. **How does this change our $\mathbf{K}$ matrix? How many parameters do we have now?**

Solution:

We will still end up with a "sparse" weight matrix, but each row now represents a new kernel filter.

In the case where no padding is applied, we end up with $(9-3+1)*3 = 21$ non-zero weights as shown below.

$$\begin{bmatrix} k_1 & k_2 & k_3 & 0 & \dots & 0 & 0 & 0 \\ 0 & k_4 & k_5 & k_6 & \dots & 0 & 0 & 0 \\ & & & \vdots & & & & \\ 0 & \dots & 0 & 0 & k_{16} & k_{17} & k_{18} & 0 \\ 0 & \dots & 0 & 0 & 0 & k_{19} & k_{20} & k_{21} \end{bmatrix}$$

In the case where "SAME" padding is applied, we need 4 more weights.

$$\begin{bmatrix} k_{-1} & k_0 & 0 & 0 & \dots & 0 & 0 & 0 \\ k_1 & k_2 & k_3 & 0 & \dots & 0 & 0 & 0 \\ 0 & k_4 & k_5 & k_6 & \dots & 0 & 0 & 0 \\ & & & \vdots & & & & \\ 0 & \dots & 0 & 0 & k_{16} & k_{17} & k_{18} & 0 \\ 0 & \dots & 0 & 0 & 0 & k_{19} & k_{20} & k_{21} \\ 0 & \dots & 0 & 0 & 0 & 0 & k_{22} & k_{23} \end{bmatrix}$$

(f) We want to know the general formula for computing the output dimension of a convolution operation. Suppose we have an input vector of dimension W_{in} and a filter of size K. If we assume stride of 1 and no padding, **what's the output dimension W_{out} ? If we applied stride of s and padding of size p, how would the dimension change?**

Solution:

With no padding, the output dimension will be $W_{out} = W_{in} - K + 1$.

With padding p and stride s, the output dimension will be $\left\lfloor \frac{W+2p-K}{s} \right\rfloor + 1$.

(g) Now, consider a convolution over a 2-dimensional image using a 2-dimensional filter. If our input image is square with dimension $W_{in} \times W_{in}$ and we apply square filter with dimension $K \times K$, stride 1, and no padding, **what are the dimensions of the resulting output? If we applied stride s and padding p, what would the output dimension be?**

Solution: The formula from the previous part can be applied to each dimension separately. This means that without padding, the output is square with each side having dimension $W_{out} = W_{in} - K + 1$. With padding p and stride s, we again get a square, this time with each side having dimension $W_{out} = \left\lfloor \frac{W+2p-K}{s} \right\rfloor + 1$.

Let's take what we've learned into actual applications on image tasks. Suppose our input is a 256×256 RGB image. You are tasked to design a convolution layer that takes in 256×256 images as input and outputs a 32-channel feature map using convolution kernels each with a kernel size of 5.

(h) Conventionally in frameworks such as PyTorch, images are arranged in 3-D tensors of shape `[channels, height, width]`. Our convolutional layer takes in the image as input and outputs a feature map arranged in 3-D tensors of shape `[32, height, width]`. **What is the shape of our input tensor (fill in the numerical value)? What is the shape of the tensor that stores all the convolution kernels in the convolution layer?**

(*Hint: the convolution kernel tensor contains a convolution kernel for each pair of input channel and output channel.*)

Solution: The shape of the input tensor is $3 \times 256 \times 256$. The convolution kernel shape is $32 \times 3 \times 5 \times 5$ or $3 \times 32 \times 5 \times 5$ or $96 \times 5 \times 5$ (any solution is valid. PyTorch uses the first format)

(i) 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 parameters do we have**?

If instead of using a convolution layer, we use a fully connected layer (flattening the image to treat it as a single vector and sized so that the output vector has the same number of activations as the output of the convolutional layer), **how many parameters does the fully connected layer contain?** Feel free to use a calculator for this question.

Solution: For the CNN layer: using the formula we derived in part (f), the output shape can be computed as $\left\lfloor \frac{256-5}{2} \right\rfloor + 1 = 126$, so the output dimension is $32 \times 126 \times 126$. In total we have $32 \times (3 \times 5 \times 5) = 2400$ parameters.

If we used a fully connected layer instead, we would first need to flatten the input image into a vector. This input vector would have dimension $d_{in} = 3 \times 256 \times 256 = 196,608$. to have the same number of activations as the CNN layer, our output vector would need to have dimension $d_{out} = 32 \times 126 \times 126 = 508,032$. The resulting fully connected layer will have $d_{in} \times d_{out} = 196,608 \times 508,032 \approx 100B$ parameters.

Contributors:

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

