

Section 09: Solutions

1. Convolutional Neural Networks

- (a) Discuss the advantages of a convolutional layer compared to a fully connected one. **Solution:**

Convolutional layers are more flexible than fully connected ones since not all input neurons affect all output neurons. In addition, the number of weights per layer is smaller than that of linear layers, which would ease computation with high-dimensional data.

- (b) Discuss the advantages of maxpooling in CNN. **Solution:**

Pooling layers are used to downsample feature maps, which make processing more efficient by reducing the number of parameters to learn.

2. Shapes in Convolutional Neural Networks

When designing a convolutional neural network, it's important to think about the shape of the data flowing through the network. In this problem you will get gain experience with thinking about the shapes in a neural network and a better intuition for why convolutional neural networks require so few parameters compared to fully connected layers.

Shape of a convolutional layer / maxpooling output: For a $n \times n$ input, $f \times f$ filter, padding p and stride s , the output size is $o \times o$ where:

$$o = \frac{n - f + 2p}{s} + 1$$

We will use Pytorch Conv2d to represent a 2D convolution, and Pytorch MaxPool2d to represent a 2D max pooling. Take a look at the documentation on [Conv2d](#) and [MaxPool2d](#).

- (a) Assume your input is a batch of N 64×64 RGB images. The input tensor your neural network receives will have shape $(N, 3, 64, 64)$. For each of the following operations, determine the new shape of the tensor as it flows through the network. Note that activations are omitted since they don't change the shape of the data as they act coordinate-wise.

1. `Conv2D(in_channels=3, out_channels=16, kernel_size=3, stride=1, padding=1)`

Solution:

$(N, 16, 64, 64)$

2. `MaxPool2d(kernel_size=2, stride=2, padding=0)`

Solution:

$(N, 16, 32, 32)$

3. `Conv2D(in_channels=16, out_channels=32, kernel_size=3, stride=1, padding=0)`

Solution:

$$(N, 32, 30, 30)$$

4. `MaxPool2d(kernel_size=2, stride=2, padding=1)`

Solution:

$$(N, 32, 16, 16)$$

5. `Conv2D(in_channels=32, out_channels=8, kernel_size=1, stride=1, padding=0)`

Solution:

$$(N, 8, 16, 16)$$

6. `Conv2D(in_channels=8, out_channels=4, kernel_size=5, stride=1, padding=0)`

Solution:

$$(N, 4, 12, 12)$$

7. `Flatten`

Solution:

$$(N, 576)$$

8. `Linear(in_features=576, out_features=10)`

Solution:

$$(N, 10)$$

(b) Again assume your input is a batch of N 64×64 RGB images. Now compute the number of parameters that each layers has. For the convolutional layers, also compute the number of parameters a fully connected layer mapping from the flattened input channels to the flattened output channels would have. It is okay to leave the number of parameters as products and additions such as $64 \cdot 32 + 16$.

1. `Conv2D(in_channels=3, out_channels=16, kernel_size=3, stride=1, padding=1)`

Solution:

$$\text{Conv: } 3 \cdot 16 \cdot 3 \cdot 3 + 16 = 448$$

$$\text{Fully connected: } 3 \cdot 64 \cdot 64 \cdot 16 \cdot 64 \cdot 64 + 16 \cdot 64 \cdot 64 = 805306512$$

2. `MaxPool2d(kernel_size=2, stride=2, padding=0)`

Solution:

0

3. Conv2D(in_channels=16, out_channels=32, kernel_size=3, stride=1, padding=0)

Solution:

$$\text{Conv: } 16 \cdot 32 \cdot 3 \cdot 3 + 32 = 4640$$

$$\text{Fully connected: } 16 \cdot 32 \cdot 32 \cdot 32 \cdot 30 \cdot 30 + 32 \cdot 30 \cdot 30 = 471888000$$

4. MaxPool2d(kernel_size=2, stride=2, padding=1)

Solution:

0

5. Conv2D(in_channels=32, out_channels=8, kernel_size=1, stride=1, padding=0)

Solution:

$$\text{Conv: } 32 \cdot 8 + 8 = 264$$

$$\text{Fully connected: } 32 \cdot 16 \cdot 16 \cdot 8 \cdot 16 \cdot 16 + 8 \cdot 16 \cdot 16 = 16779264$$

6. Conv2D(in_channels=8, out_channels=4, kernel_size=5, stride=1, padding=0)

Solution:

$$\text{Conv: } 8 \cdot 4 \cdot 5 \cdot 5 + 4 = 804$$

$$\text{Fully connected: } 8 \cdot 16 \cdot 16 \cdot 4 \cdot 12 \cdot 12 + 4 \cdot 12 \cdot 12 = 1180224$$

7. Flatten

Solution:

0

8. Linear(in_features=576, out_features=10)

Solution:

$$576 \cdot 10 + 10 = 5770$$