Fundamentals of Convolutional Neural Networks Quiz Solutions

Anthony Shara
Aditya Ganapathi
Dohyun Cheon
Larry Yan
Richard Shuai

December 2, 2020

1 Filter Size

Given that an input is 256x256 and that the size of layer 1 is 224x224, what is the size of the first convolution filter? (Assume 0 padding and a stride of 1)

Solution: 33x33. Using the formula C = ((I - F + 2P)/S) + 1, where C = size of the convoluted matrix, I = size of the input matrix, F = size of the filter, P = size of the padding, S = stride applied, we have 224 = ((256 - F + 2∗0)/1) +1 = 256 - F + 1. Thus, F = 33.

2 Output size

With input size 32 x 32, a kernel size of 3x3, a stride of 3, and 2 on both sides, what is the result of the output feature map?

Solution: 11x11. Using the formula C = ((I - F + 2P)/S) + 1, where C = size of the convoluted matrix, I = size of the input matrix, F = size of the filter, P = size of the padding, S = stride applied, we have C = ((32 - 3 + 2∗2)/3) = 11.

3 Same padding

With input size 32 x 32, a kernel size of 7x7, and a stride of 1, what padding is necessary in order to achieve a "same" convolution? (A "same" convolution refers to a convolution which results in an output with the same shape of the original input).

Solution: Pad by 3 on all sides. Using the formula C = ((I - F + 2P)/S) + 1, where C = size of the convoluted matrix, I = size of the input matrix, F = size of the filter, P = size of the padding, S = stride applied, we see that the we need to pad with 3 zeroes on all sides to achieve a same convolution.

4 3x3 Filters

A 3x3 filter covers only 9 neurons while a 15x15 filter covers 225 neurons. How many 3x3 filter layers are required to achieve the same coverage as 1 15x15 filter?

Solution: Applying a 3x3 filter to a 15x15 image results in a 13x13 matrix. We see that applying a 3x3 filter on a matrix w by h results in a matrix with w-2 by h-2. Thus we will need (15 - 1) / 2 = 7 3x3 filters.

5 CNN Advantages

Why would we use convolutional neural networks as opposed to fully connected layers between two feature maps?

Solution: Convolutional neural networks save parameters by sharing parameters to extract features. CNNs can be thought of as regularized multilayer perceptrons, where we only use some weights of the fully connected layers to extract features.

6 Image Classification Intuition

For classification, why do many architectures use fully connected layers after the convolutional layers in order to make classification predictions?

Solution: Convolutional filters are efficient for extracting relevant features of the image for classification, while fully connected layers help to integrate all of this information together to classify an image.

7 HyperParameters

Which of the following are hyperparameters of a Convolutional Neural Network?

- Size of Filters
- Stride lengths
- Depth of the Network
- The values of the filters

Solution: A, B, and C.

8 Vanishing Gradient Problem

As more layers using ReLu activation functions are added to a CNN, the gradients of a loss function approaches zero. What technique is used in a famous CNN architecture to combat this vanishing gradient problem?

Solution: Residual blocks from the famous ResNet is a solution to this problem.

9 Pooling

Why is pooling important in CNN architectures?

Solution: Pooling allows you to downsample your feature maps so that when convolutions are applied to them, features are extracted from a larger receptive field. This contrasts with simply making kernel sizes larger, which can be very expensive in terms of the number of parameters.

10 Convolutions as Matrix-Vector Multiplications

Convolutional layers represent linear transformations, and they can be expressed as a matrix vector multiplication $A\vec{x}$ for some matrix A and some vector $\vec{x}$. Explain how to obtain these, and explain why convolutions aren't implemented as matrix-vector multiplications in practice.

Solution: The $\vec{x}$ vector would be the flattened version of the input feature map, while the A matrix would be Toeplitz matrix which maps $\vec{x}$ into the flattened version of the output feature map. The reason convolutions aren't implemented as matrix-vector multiplications in practice is because the resulting convolution matrices are extremely large and wasteful since they contain many zeroes. Seeing as fully connected layers can be expressed as matrix-vector multiplications, this relates to the concept that convolutions can be viewed as fully connected layers where most weights are 0s.