Fundamentals of Convolutional Neural Networks Quiz Solutions

Anthony Shara
Aditya Ganapathi
DoHyun Cheon
Larry Yan
Richard Shuai

December 11, 2020

1 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.

2 Weight savings

2.1 Fully connected layer

For an input with dimensions $256 \times 256 \times 3$, calculate the number of weights required for a fully connected layer with 1000 neurons.

Solution: 196608000. Since the layer is fully connected, there are $256 \times 256 \times 3 = 196608$ weights to each neuron. Multiply by 1000 neurons to get the total.

2.2 Convolutional layer

If we were to instead use a convolutional layer with filter size 7, sufficient padding and stride to result in an output of size 84x84, and a depth of 32, calculate the number of weights required for this layer.

Solution: 4704. Due to weight sharing, all of the weights for a given layer are the same. There are $7 \times 7 \times 3 = 147$ weights for each filter (remember that a filter extends through the depth of the input!) and 32 filters. Their product is 4704. Note that the convolutional layer, even though it had more neurons ($84 \times 84 \times 32$), used much fewer weights than the fully connected layer. This is why we use convolutional networks for images.

3 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.

4 3x3 Filters

A 3x3 filter covers only 9 neurons while a 15x15 filter covers 225 neurons. How many 3x3 filters are required to achieve the same coverage as a single 15x15 filter? What percentage of weights do we save using only 3x3 filters compared to a 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. Note that this only requires $9 \times 7 = 63$ weights, compared to the 225 weights of the 15x15 filter. We see that we need $(225 - 63)/225 * 100 = 72\%$ fewer weights.

5 Same convolution 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.

6 Convolutions as Matrix-Vector Multiplications

Convolutions represent linear transformations, and we have seen how they can be expressed as a matrix vector multiplication for some convolution matrix and some input vector. Let the input image be the 4x4 matrix X given below. We then perform a convolution with a $3x3$ kernel K that is also provided below.

$$X = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 0 \\ 3 & 3 & 0 & 0 \\ 4 & 0 & 0 & 0 \end{bmatrix} \qquad K = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

a) Express this convolution as a matrix-vector multiplication $\mathbf{A}\vec{x}$, writing out explicitly what the entries of A and $\vec{x}$ are. *Hint: Think about what the dimensions of the input and output image will be. What must the dimensions of A be?*

b) Give a reason for why convolutions are not usually implemented as a matrix-vector multiplication in practice.

Solution: **a)** First note that the convolution results in a $2x2$ output image. $\vec{x} \in \mathbb{R}^{16}$ would be the unraveled version of the input feature map, while the A matrix would be the matrix mapping $\vec{x} \in \mathbb{R}^{16}$ to an unraveled output feature map $\vec{y} \in \mathbb{R}^4$. The explicit forms of $\vec{x}$ and A are:

$$\vec{x} = \begin{bmatrix} 1 & 1 & 1 & 1 & 2 & 2 & 2 & 0 & 3 & 3 & 0 & 0 & 4 & 0 & 0 & 0 \end{bmatrix}^T$$

$$A = \begin{bmatrix} 1 & 2 & 3 & 0 & 4 & 5 & 6 & 0 & 7 & 8 & 9 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 & 4 & 5 & 6 & 0 & 7 & 8 & 9 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 3 & 0 & 4 & 5 & 6 & 0 & 7 & 8 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 & 0 & 4 & 5 & 6 & 0 & 7 & 8 & 9 \end{bmatrix}$$

b) As you can might tell, the reason convolutions aren't implemented as matrix-vector multiplications in practice is because the resulting convolution matrices are extremely large and repetitive due to the number of zeroes. However, seeing as fully connected layers can be expressed as matrix-vector multiplications, we can also see how this relates to the concept that convolutions can be viewed as fully connected layers where most weights are 0s.

7 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, especially with increased depth.

8 Convolutions vs Cross correlation

In practice, most CNNs use cross correlation instead of convolutions in their convolutional layers (don't flip filters). Given two implementations of CNNs, one using cross correlation and one using convolutions that are otherwise identical, with weights initialized to zero and given the same training set, what differences would there be in their outputs and filters, if any?

Solution: Both networks would give the same predictions, as the weights are learned and the same connections are still present. The only difference would be that all the filters are flipped, as the leftmost weight for the convolution implementation would be the same as the rightmost weight for the cross correlation implementation, and vice versa. Otherwise, the network is unchanged.

9 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.

10 Vanishing Gradient Problem

As more layers using ReLU activation functions are added to a CNN, the gradients of a loss function approaches zero. This means that adding more layers to a CNN produces diminishing returns on accuracy, as later layers are unable to learn the function effectively. What technique is used in a famous CNN architecture to combat this vanishing gradient problem?

Solution: Residual blocks or skip connections from the famous ResNet is a solution to this problem. In residual blocks, the output of a layer acts as input into both the next layer and one 2-3 layers forward. These skip connections allow for the loss function to propagate much further, enabling the network to learn its weights more accurately in response to the losses.