eCornell

# Explore Pooling Layers

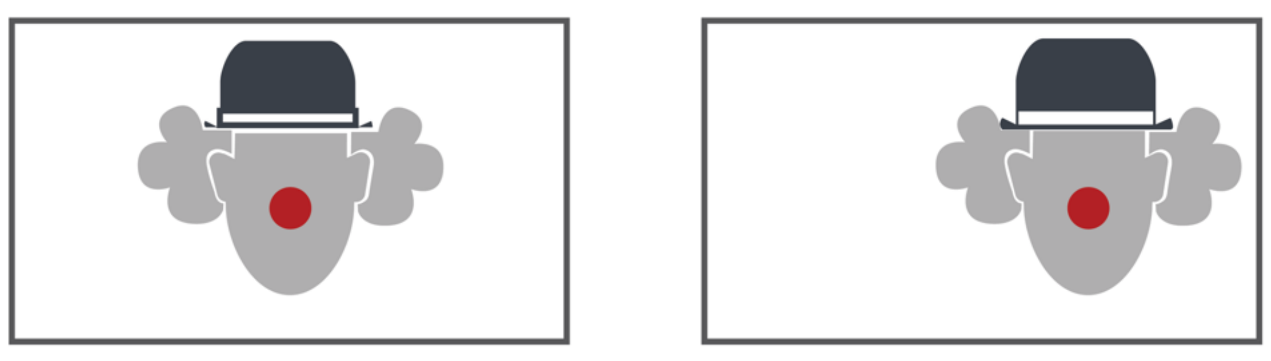
☆ Key Points

**Translation invariance is the ability to withstand small shifts of an object in a photo.**

**A pooling layer looks at little patches of the image and introduces translation invariance into ConvNets.**

**A good model that uses pooling layers would be able to recognize that the image is the same even when in a different position (i.e., a person centered in a photo and the same person on the right of another photo).**

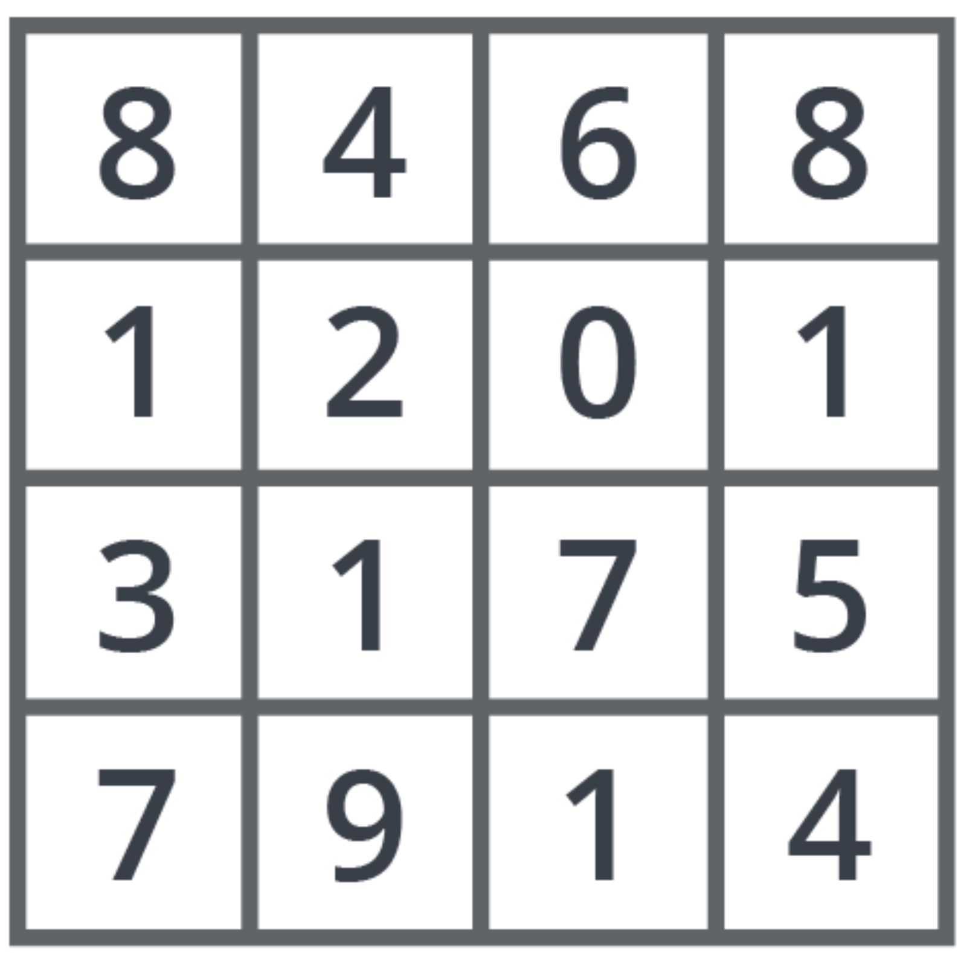
Suppose we are trying to classify whether an image contains a clown and we have the image on the left below as part of our training data.



Now, if we shift the clown to the right (right picture above), a good model should be able to recognize that the clown in the image on the left is the same as the clown in the image on the right. A model that is capable of doing this is *translation invariant*: it is able to withstand small translations of the object in the image.

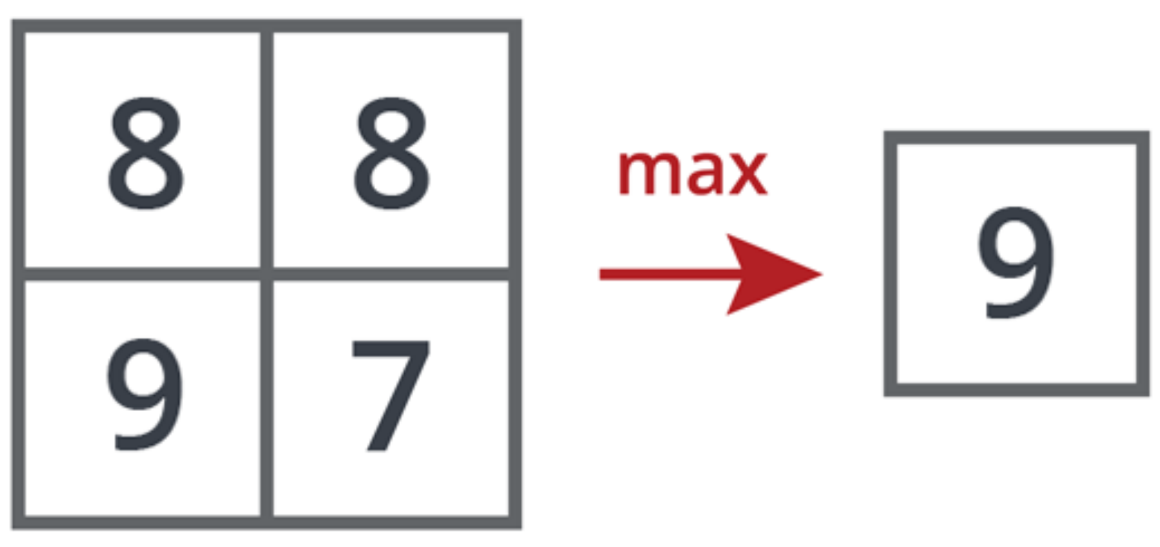
The pooling layer essentially introduces translation invariance into ConvNets. A pooling layer looks at small patches of the input image and then outputs the maximum value (max pooling) or average value (average pooling) of the patch.

Suppose we have the following array, which is a small 4 × 4 pixel region of an image:

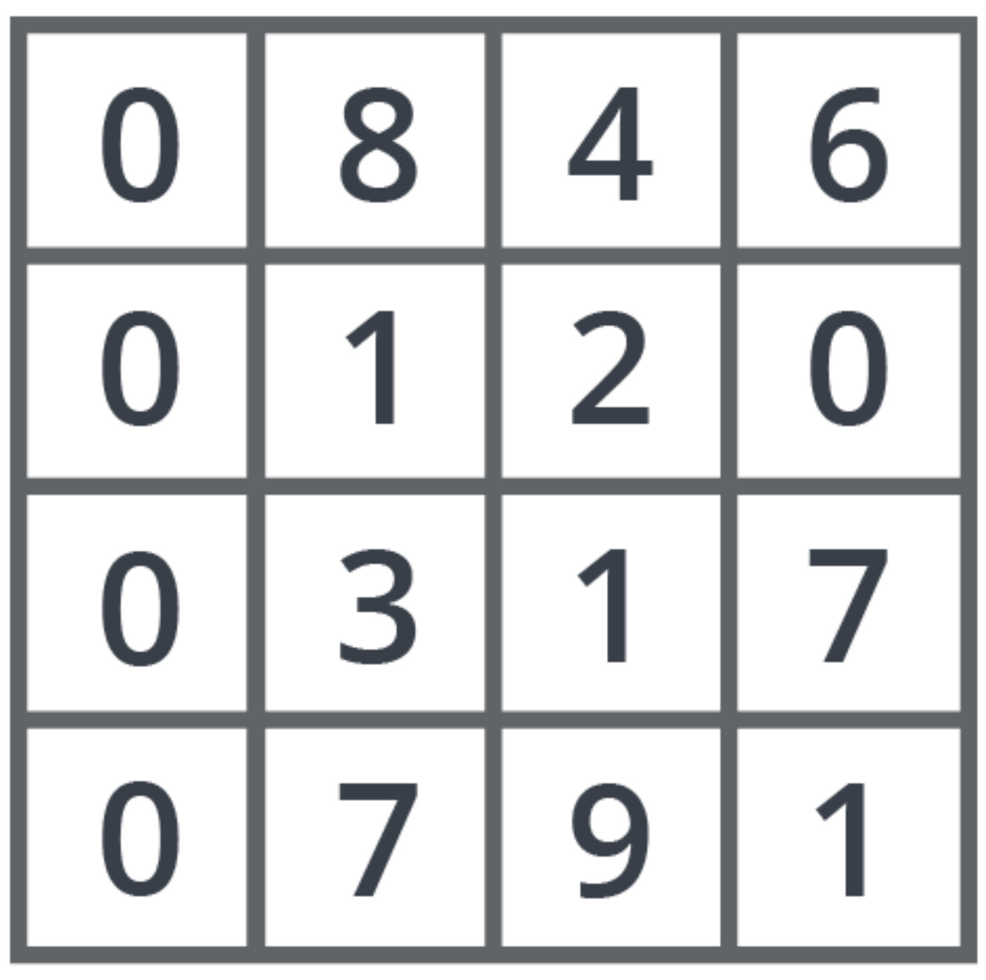


| | | | |
|---|---|---|---|
| 8 | 4 | 6 | 8 |
| 1 | 2 | 0 | 1 |
| 3 | 1 | 7 | 5 |
| 7 | 9 | 1 | 4 |

If we apply a max pooling of size 2 to this array, we'll capture the largest value in the four quadrants of the array. For example, the top-left quadrant contains 8, 4, 2, and 1. The largest value, 8, is captured in our smaller pooling array in the top-left location. If we do this to reduce the entire 4 × 4 into a 2 × 2 array, we will get:

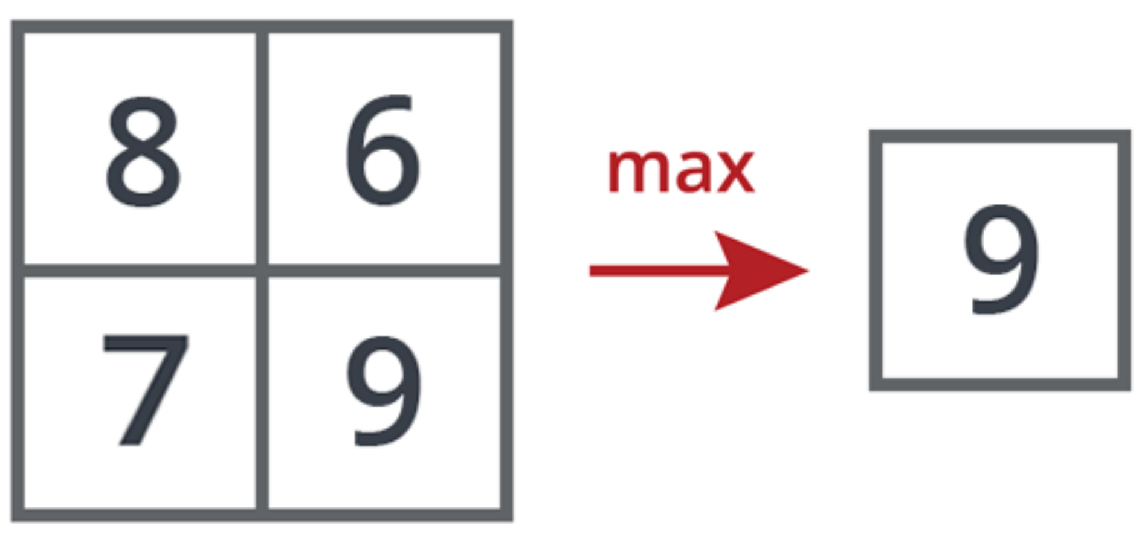


Essentially, the max pooling of size 2 outputs the maximum value inside each 2 × 2 region; hence, the max from the first example is 9. Now, suppose the object in the image is shifted by one pixel to the right. For the same 4 × 4 location of the image, we might get:



| | | | |
|---|---|---|---|
| 0 | 8 | 4 | 6 |
| 0 | 1 | 2 | 0 |
| 0 | 3 | 1 | 7 |
| 0 | 7 | 9 | 1 |

If we apply max pooling to size 2 to this array, we will obtain:



It is not entirely identical, but several values remained unchanged and the max output was still 9. The key is that the very large numbers will always survive, as they will remain the max values in their 2 × 2 sub-grids. If there are multiple rounds of pooling, the final output will likely be identical even if the image is translated.

## Increase of Receptive Field

Another nice property of pooling layers is that they shrink the image size. In the case above, with a 2 × 2 pooling window, we half the size of the activations. This can be very beneficial in several ways. First, it speeds up processing downstream, and second, it increases the receptive field (or the span) of the next convolution layer. Imagine, for example, in the example above, the final output is inserted into a 2 × 2 convolutional filter. This filter will convolve the 7 and 9, which are originally 4 pixels apart from each other. This increase in the receptive field is particularly useful for object recognition if the object spans the whole image.

## Nonlinearity

Last but not least, max pooling also introduces a nonlinearity into the network. Similar to a fully connected network, which uses ReLUs as nonlinear transition functions, max pooling can have a similar nonlinear effect in convolutional networks. (Note that convolutional filters are linear transformations.)

◂ Previous

Next ▸