Miscellaneous stuff about convolutional neural networks

Feb. 24, 2020

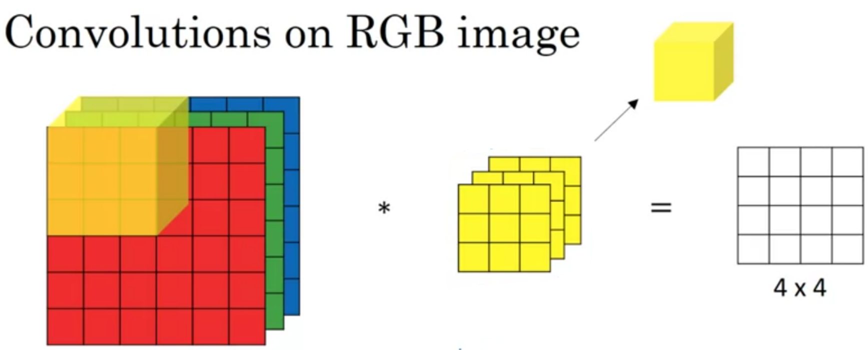
Convolutions with multiple channels

Set up for multiple channels

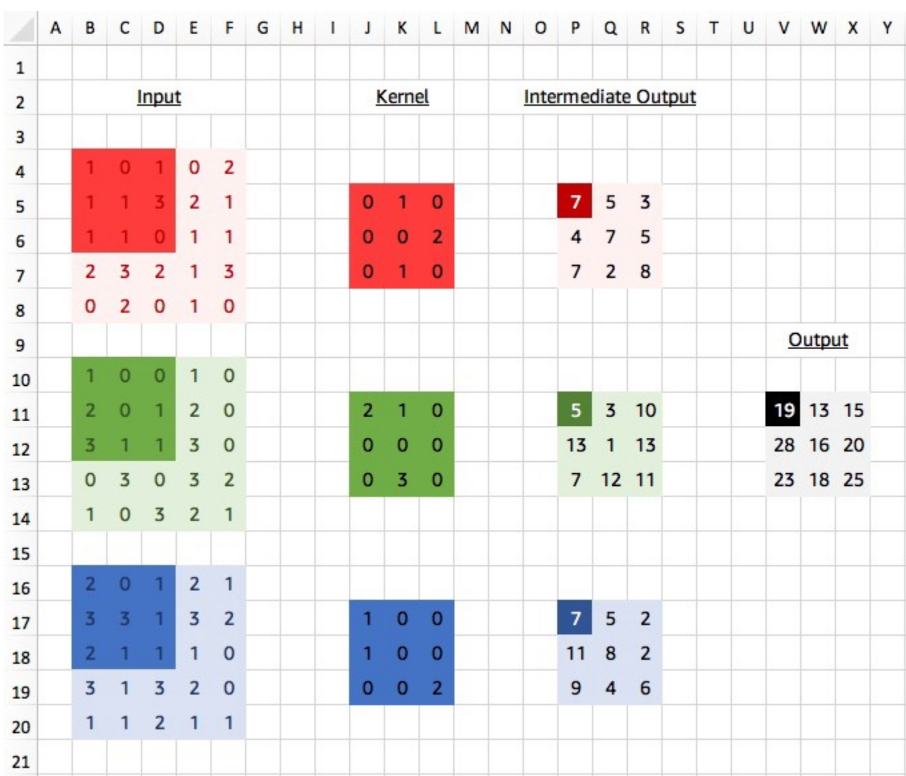
- We will work with 3-dimensional arrays of shape $(n_H^{[l]}, n_W^{[l]}, n_C^{[l]})$: H for height, W for width, and C for channels
- Starting point is often $n_C^{[0]} = 3$ for the Red/Green/Blue encoding of color images
- Later layers will have $n_C^{[l]} =$ the number of filters applied to the previous layer, $l - 1$
- The filter matrix W has shape $(f^{[l]}, f^{[l]}, n_C^{[l-1]})$: depth is the number of channels in the layer the filter is applied to.

How convolutions are calculated

- Picture from Andrew Ng (original source: https://www.youtube.com/watch?v=KTB_OFoAQcc&list=PLkDaE6sCZn6Gl29AoE31iwdVwSG-KnDzF&index=6):

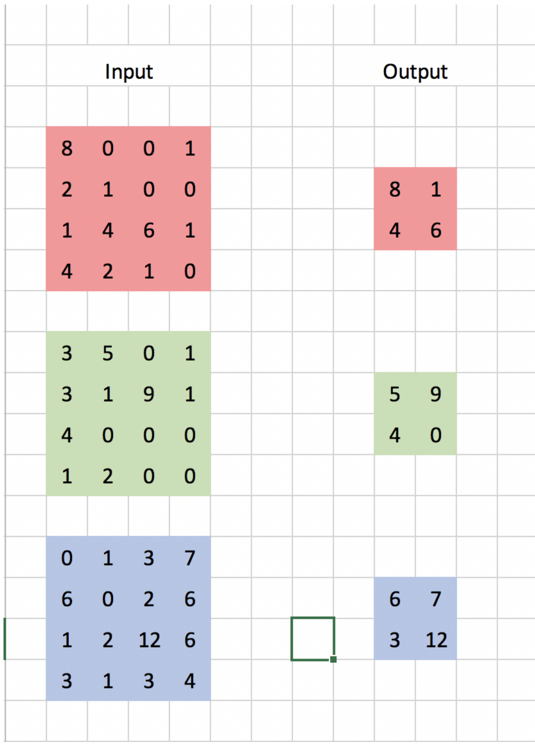


- Picture from Thom Lane (original source: https://medium.com/apache-mxnet/multi-channel-convolutions-explained-with-ms-ex



Pooling Layers

- A pooling layer does two things:
  - Answers the question "was a filter activated in a patch of the previous layer?"
  - Reduces the height and width of a representation of the input
- **Almost always:** max pooling with a 2×2 grid and stride of 2:



Why use convolutions?

There are two closely related reasons that convolutions are really helpful:

1. Translation invariance:
    - Suppose you have a 3 by 3 filter ($9 \cdot n_C^{[l-1]}$ estimated parameters) that does vertical edge detection.
    - This same filter does vertical edge detection anywhere in the image.
2. Reduced number of parameters:
    - Suppose the input image is 256 pixels by 256 pixels with 3 channels.
    - A fully connected (dense) layer has $256^2 \cdot 3 = 196608$ weight parameters **per activation unit**. With a single hidden layer with 10 units in it, you're already at almost 2 million parameters. Including many hidden layers/units is infeasible.
    - A single $3 \times 3 (\times 3)$ convolutional filter has 27 parameters that are re-used over different patches of the input, so is more efficient in terms of number of parameters used.

Overview of model structure

The vast majority of CNNs have the following general architecture:

- Input layer
- One or more convolutional layers with ReLU activations
- Max pooling layer
- One or more convolutional layers with ReLU activations
- Max pooling layer
- ...
- One or more dense/fully connected layers with ReLU activations leading to the output

The shapes of activations generally change as we go deeper in the network:

- First layers have relatively high values of n_H and n_W, but small values of n_C
- Later laters have relatively small values of n_H and n_W, but higher values of n_C

The filter activations are doing different things as we go deeper in the network:

- First layer activations detect simple details (edges, lines) in small patches of the image (3×3 or 5×5 pixels)
- Later layer activations detect more complicated information ("cat", "dog", "labrador retriever") across larger patches of the image