Course 4 - Computer Vision

Mohammad Khalaji

September 2, 2020

1 Week 1

1.1 Computer Vision Problems

- Image Classification
- Object Detection
- Neural Style Transfer

1.2 Padding

One modification that needs to be done in order to implement convolution in neural networks. For example, convolving a 3×3 filter with a 6×6 image results in a 4×4 image.

Generally, convolving a $f \times f$ filter with a $n \times n$ image yields a $n - f + 1 \times n - f + 1$ image. Two downsides:

- Every time we convolve, the image shrinks.
- Corner and edge pixels are only used once in computing the convolution.

Padding helps resolve these issues: If the filter is $2k + 1 \times 2k + 1$, pad the original image with $p = k$ pixel over each edge. As a result, the resulting image, the one that is fed into the convolution with the filter, will be $n + 2p \times n + 2p$.

1.3 Valid and Same Convolutions

- Valid: No padding, which means that convolving a $f \times f$ filter with a $n \times n$ image yields a $n - f + 1 \times n - f + 1$ image.
- Same: The output size is the same as the input size (as explained above).

$$p = \frac{f-1}{2}$$

1.4 Strided Convolutions

Parameters of convolution:

- $n \times n$ image
- $f \times f$ filter
- p padding
- s stride

The size of the resulting image:

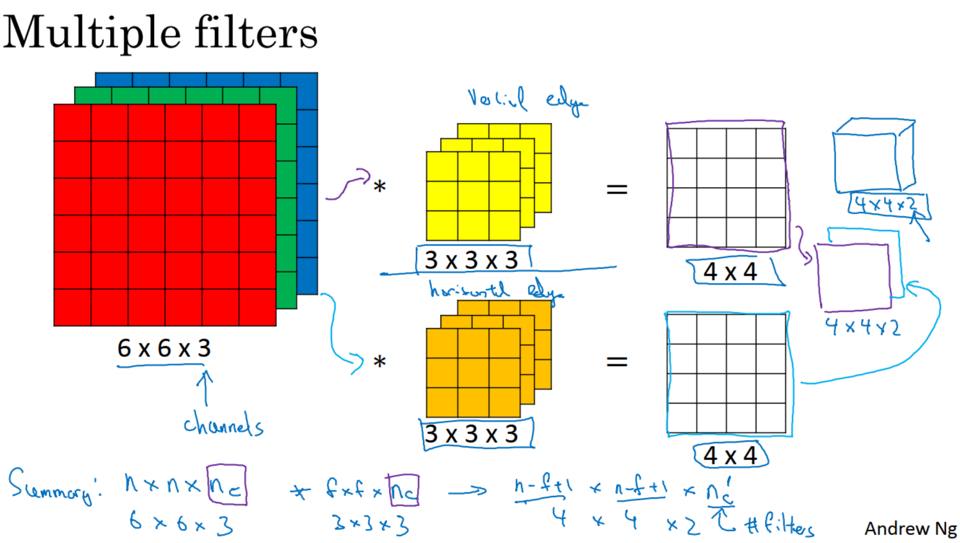
$$(\frac{n+2p-f}{s} + 1) \times (\frac{n+2p-f}{s} + 1)$$

1.5 Convolutions over Volumes

We must use filters that are volumes themselves. For example, the convolution of a $6 \times 6 \times 3$ image with a $3 \times 3 \times 3$ filter yields a $4 \times 4 \times 1$ image. Note that the number of channels in the filter must be equal to the number of channels in the image. However, if we use multiple filters, for example, a filter that detects vertical edges, and another filter that detects horizontal edges, the result will no longer be 2 dimensional. Generally:

$$(n \times n \times n_c) * (f \times f \times n_c) = ((n - f + 1) \times (n - f + 1) \times n_c^{'})$$

Where $n_c^{'}$ is the number of filters.



1.6 One Layer of a CNN

For each of convolution outputs, we are going to add a bias, and apply an activation function to the result of that addition. We have the same bias added to all pixels rather than having separate biases for each pixel.

Exercise: If you have 10 $3 \times 3 \times 3$ filters, how many parameters does that layer have?

- Filter Parameters: $10 \times 3 \times 3 \times 3 = 270$
- One real number bias for each filter: $10 \times 1 = 10$

So, a total of 280 parameters.

Notation Summary: If layer l is a convolution layer:

- $f^{[l]}$: filter size
- $p^{[l]}$: padding
- $s^{[l]}$: stride
- $n_c^{[l]}$: number of filters = number of output channels
- $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$: each filter
- $n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$: input
- $n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$: output, where:

$$n_H^{[l]} = 1 + \frac{n_H^{[l]} + 2p^{[l]} - f^{[l]}}{s^{[l]}}$$

$$n_W^{[l]} = 1 + \frac{n_W^{[l]} + 2p^{[l]} - f^{[l]}}{s^{[l]}}$$

- $a^{[l]} : n_H^{[l]} \times n_H^{[l]} \times n_c^{[l]}$: activations
- $A^{[l]} : m \times n_H^{[l]} \times n_H^{[l]} \times n_c^{[l]}$: vectorized activations
- $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$: weights
- $1 \times 1 \times 1 \times n_c^{[l]}$: biases

1.7 Simple CNN Example

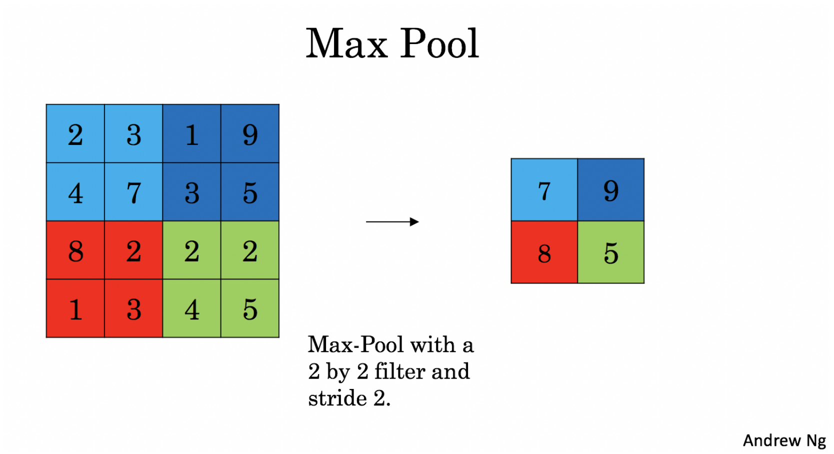
- Input: $39 \times 39 \times 3$
- Layer 1: 10 filters, each $3 \times 3 \times 3$ ($f = 3$), no padding, stride 1
- Output: $(1 + \frac{39+0-3}{1}) \times (1 + \frac{39+0-3}{1}) \times 10 = 37 \times 37 \times 10$
- Layer 2: 20 filters, each $5 \times 5 \times 10$ ($f = 5$), no padding, stride 2
- Output: $(1 + \frac{37+0-5}{2}) \times (1 + \frac{37+0-5}{2}) \times 20 = 17 \times 17 \times 20$
- Layer 3: 40 filters, each $5 \times 5 \times 40$ ($f = 5$), no padding, stride 2
- Output: $(1 + \frac{17+0-5}{2}) \times (1 + \frac{17+0-5}{2}) \times 40 = 7 \times 7 \times 40$
- Layer 4: Fully Connected...

Layer Types in Conv Nets:

- Convolution (CONV)
- Pooling (POOL)
- Fully Connected (FC)

1.8 Pooling Layers

Pooling layers have no learnable parameters. Only f and s determine how the pooling is done. Pooling is either max pooling or average pooling.



The resulting image size is calculated just like how it was for convolutional layers:

$$(\frac{n + 2p - f}{s} + 1) \times (\frac{n + 2p - f}{s} + 1)$$

2 Week 2

3 Week 3

4 Week 4