

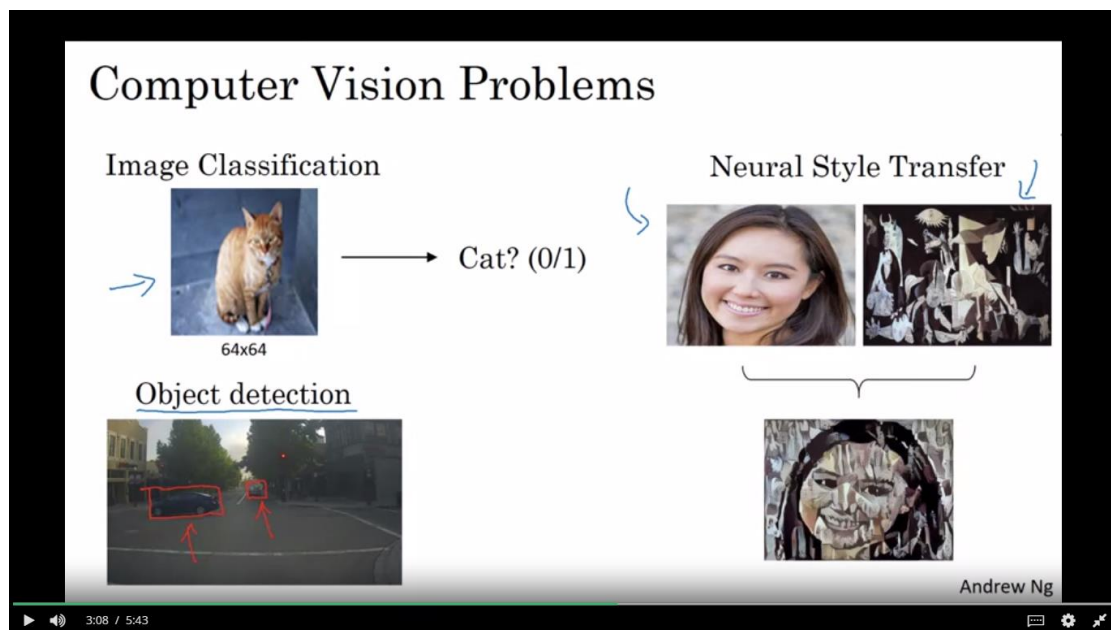
Ahmad Hussameldin Hamed Hassan

Shared Git-hub link: <https://github.com/ahmadhassan1993/sharing-github>

Convolutional Neural Network

Week 1 Summary

It is used in computer vision as in multi-image processing tasks like, image classification, object detection and image style:

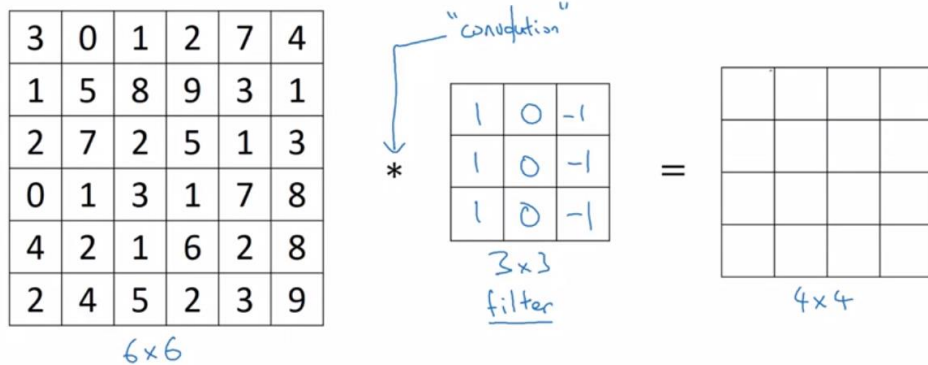


It is very suitable for large amount of data, as it implied the convolutional operation.

Edge Detection:

We do a convolutional filter (kernel) on an image to detect the places of the edges:

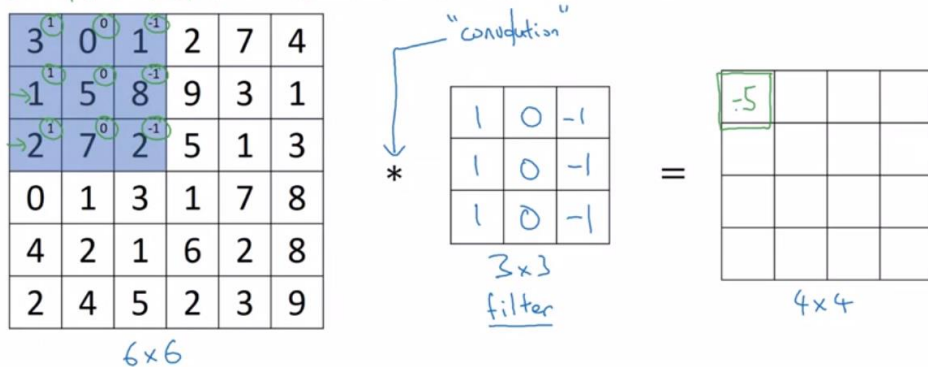
Vertical edge detection



Andrew Ng

Vertical edge detection

$$3 \times 1 + 1 \times 1 + 2 \times 1 + 0 \times 0 + 5 \times 0 + 7 \times 0 + 1 \times -1 + 8 \times -1 + 2 \times -1 = -5$$



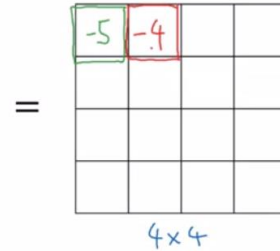
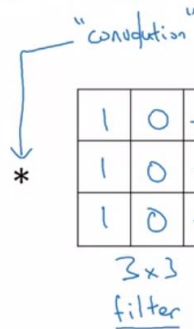
Andrew Ng

Vertical edge detection

$$3 \times 1 + 1 \times 1 + 2 \times 1 + 0 \times 0 + 5 \times 0 + 7 \times 0 + 1 \times -1 + 8 \times -1 + 2 \times -1 = -5$$

3	<u>0</u> ¹	<u>1</u> ⁰	<u>2</u> ⁻¹	7	4
1	<u>5</u> ¹	<u>8</u> ⁰	<u>9</u> ⁻¹	3	1
2	<u>7</u> ¹	<u>2</u> ⁰	<u>5</u> ⁻¹	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6x6



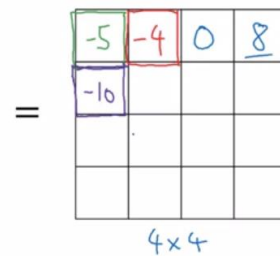
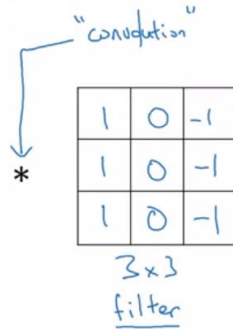
Andrew Ng

Vertical edge detection

$$3 \times 1 + 1 \times 1 + 2 \times 1 + 0 \times 0 + 5 \times 0 + 7 \times 0 + 1 \times -1 + 8 \times -1 + 2 \times -1 = -5$$

3	<u>0</u> ¹	<u>1</u> ⁰	<u>2</u> ⁻¹	<u>7</u> ¹	<u>4</u> ⁰
<u>1</u> ¹	<u>5</u> ⁰	<u>8</u> ⁻¹	<u>9</u> ¹	<u>3</u> ⁰	<u>1</u> ⁻¹
<u>2</u> ¹	<u>7</u> ⁰	<u>2</u> ⁻¹	<u>5</u> ¹	<u>1</u> ⁰	<u>3</u> ⁻¹
<u>0</u> ¹	<u>1</u> ⁰	<u>3</u> ⁻¹	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6x6



Andrew Ng

Different commands of convolution in different programming tools:

Vertical edge detection

$$3 \times 1 + 1 \times 1 + 2 \times 1 + 0 \times 0 + 5 \times 0 + 7 \times 0 + 1 \times -1 + 8 \times -1 + 2 \times -1 = -5$$

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

6x6

convolution

1	0	-1
1	0	-1
1	0	-1

3x3 filter

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

4x4

python: conv-forward
tensorflow: tf.nn.conv2d
keras: Conv2D

Andrew Ng

Example of picture with vertical edge in middle:

Vertical edge detection

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

6x6

*

1	0	-1
1	0	-1
1	0	-1

3x3

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

4x4

Andrew Ng

The convolution output gives high value in the middle, where the edge (between white and gray) is found in original image.

The convolution output is wide in the out matrix because the image is small, and the edge is strong. It could be more accurate (in size) in large image size.

In the convolutional filter, we take three consecutive values to represent the two colors in the image and the edge between them in descending order.

If we flipped horizontally the previous image, we get negative values for the edge. However, the place of the edge is the same:

Vertical edge detection examples

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

→

*

1	0	-1
1	0	-1
1	0	-1

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

→

*

1	0	-1
1	0	-1
1	0	-1

=

0	-30	-30	0
0	-30	-30	0
0	-30	-30	0
0	-30	-30	0
0	-30	-30	0
0	-30	-30	0

Andrew Ng

To overcome this, we can take the absolute value of the output matrix.

If we flipped the previous filter, we will detect the horizontal edges.

Here is a complex example:

Vertical and Horizontal Edge Detection

1	0	-1
1	0	-1
1	0	-1

Vertical

1	1	1
0	0	0
-1	-1	-1

Horizontal

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

→

*

1	1	1
0	0	0
-1	-1	-1

=

0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0

Andrew Ng

Here we have 30 and -30 represents the positive and negative edges respectively. Whereas for 10 and -10 represents the transition in the middle (as not all the white side is pure white, it has also dark regions). In large image size, this 10 will be a smaller value.

Two famous filters used are: Sobel and Scharr filters. The following screen is for their vertical edge detection usage. For horizontal detection, just flip 90 degrees.

Learning to detect edges

1	0	-1
1	0	-1
1	0	-1

\rightarrow

1	0	-1
2	0	-2
1	0	-1

Sobel filter

3	0	-3
10	0	-10
3	0	-3

Scharr filter

Andrew Ng

In NNs, we use the weights W as the filter parameters and simply update their values by learning and backpropagation. In this way, we can detect even any degree of edges, not only vertical and horizontal edges:

Learning to detect edges

1	0	-1
1	0	-1
1	0	-1

\uparrow

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

\rightarrow

1	0	-1
2	0	-2
1	0	-1

Sobel filter

3	0	-3
10	0	-10
3	0	-3

Scharr filter

\times

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

3x3

=

45°
70°
73°

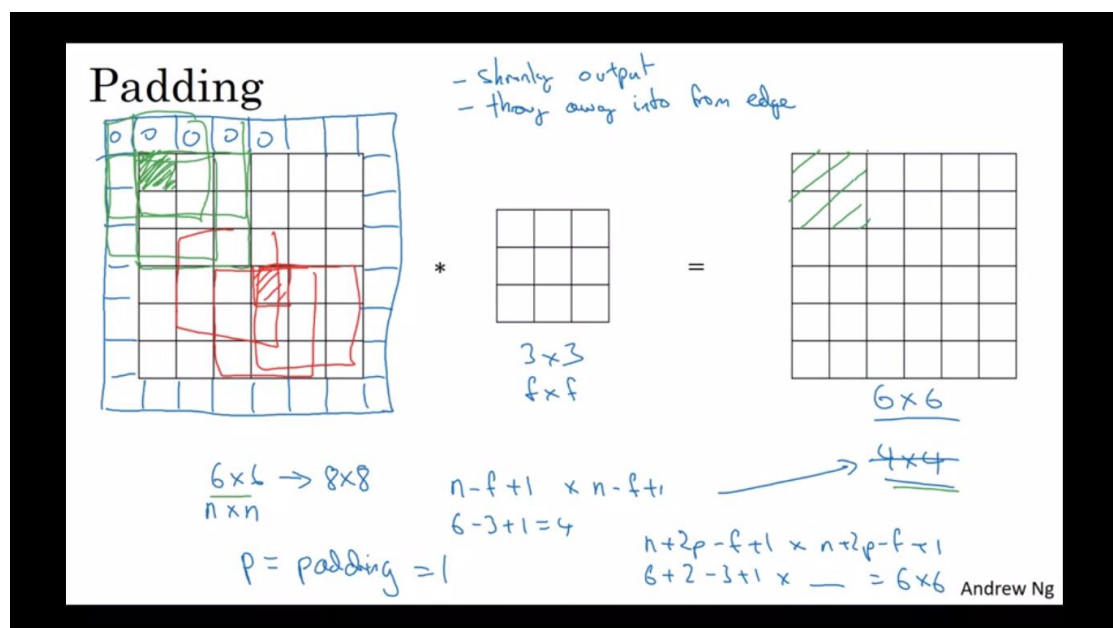
Andrew Ng

Padding:

There are two problems with the output of the convolution of an image:

- 1- Shrinky output
- 2- Through away info from edge of the image

To solve those two problems especially the second one, we do Padding. Padding means to add pixels at the original image size, this allow the effect of the original edge pixel to be represented more at the convolution output. The final dimension of the output is $(n+2p-f+1, n+2p-f+1)$ instead of $(n-f+1, n-f+1)$.



We can also enhance more by adding more pads.

There are two types of convolution, depending on padding:


- 1) Valid: this means without padding
- 2) Same: this means add pads such that the output is same dimension as input image. This can be achieved by the relation $p=(f-1)/2$ and f is usually odd.

Valid and Same convolutions

\rightarrow no padding
 "Valid": $n \times n$ * $f \times f \rightarrow \frac{n-f+1}{1} \times \frac{n-f+1}{1}$
 6×6 * $3 \times 3 \rightarrow 4 \times 4$

"Same": Pad so that output size is the same as the input size.

$$\begin{aligned}
 n+2p-f+1 &\times n+2p-f+1 \\
 n+2p-f+1 &= n \Rightarrow p = \frac{f-1}{2} \\
 3 \times 3 \quad p &= \frac{3-1}{2} = 1 \quad \left| \begin{array}{l} 5 \times 5 \\ f=5 \end{array} \right. \quad p=2
 \end{aligned}$$

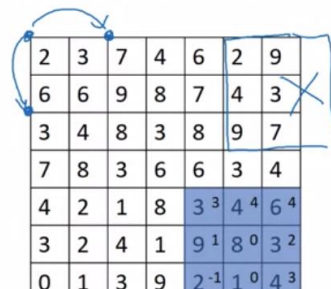
f is usually odd


Andrew Ng

Stride convolution:

Another type of convolution is stride convolution. This is a convolution where we move from convolution box to another by step s :

Strided convolution



$$\begin{array}{|c|c|c|} \hline 3 & 4 & 4 \\ \hline 1 & 0 & 2 \\ \hline -1 & 0 & 3 \\ \hline \end{array}$$

3×3
 stride = 2

$$\begin{array}{|c|c|c|} \hline 91 & 100 & 83 \\ \hline 69 & 91 & 127 \\ \hline 44 & 72 & 74 \\ \hline \end{array}$$

3×3

$n \times n$ * $f \times f$
 padding p stride s
 $s=2$

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

$$\frac{7+0-3}{2} + 1 = \frac{4}{2} + 1 = 3$$

Andrew Ng

Summary of convolutions

$n \times n$ image $f \times f$ filter

padding p stride s

Output Size:

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

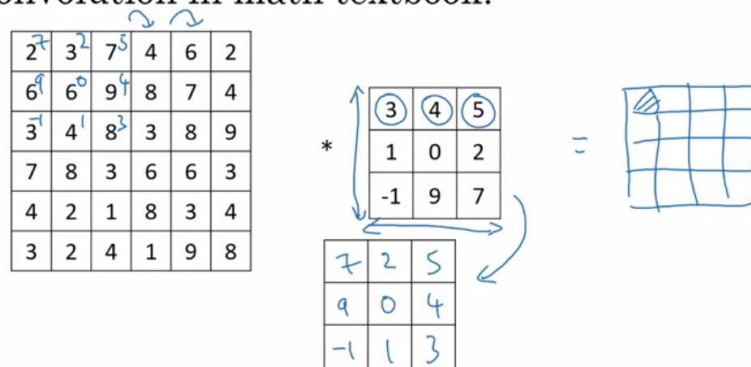
Andrew Ng

We must be sure to choose this fraction to be integer, so we take the floor of it and choose well values for both s and p to make Same Convolution as mentioned earlier.

Note (not important):

Technical note on cross-correlation vs. convolution

Convolution in math textbook:



Andrew Ng

The correct filter after flipping vertically and horizontally would be:

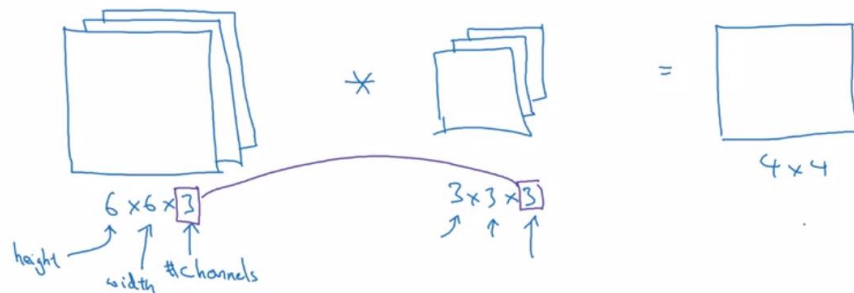
7	9	-1
2	0	1
5	4	3

What Andrew did was more a flip over the diagonal.

Convolution over volume:

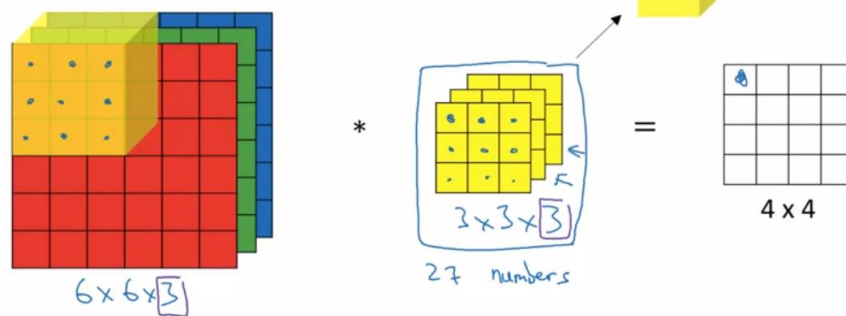
In RGB (Red-Green-Blue) images, there is filter for each color (channel). We can choose the values of each filter depending on our target. For example, if we want only red, then red filter have values while other two colors filters are zeros. The output of this volume convolution is 2D rather than its 3D inputs:

Convolutions on RGB images



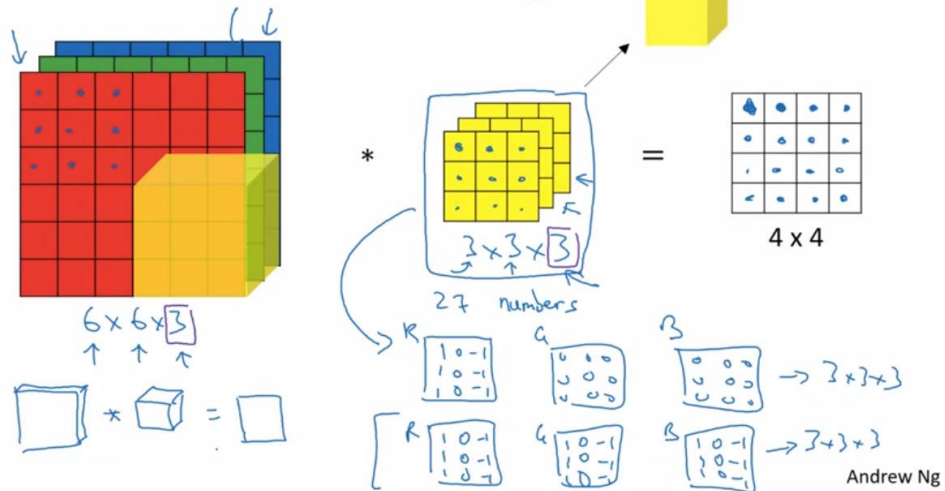
Andrew Ng

Convolutions on RGB image



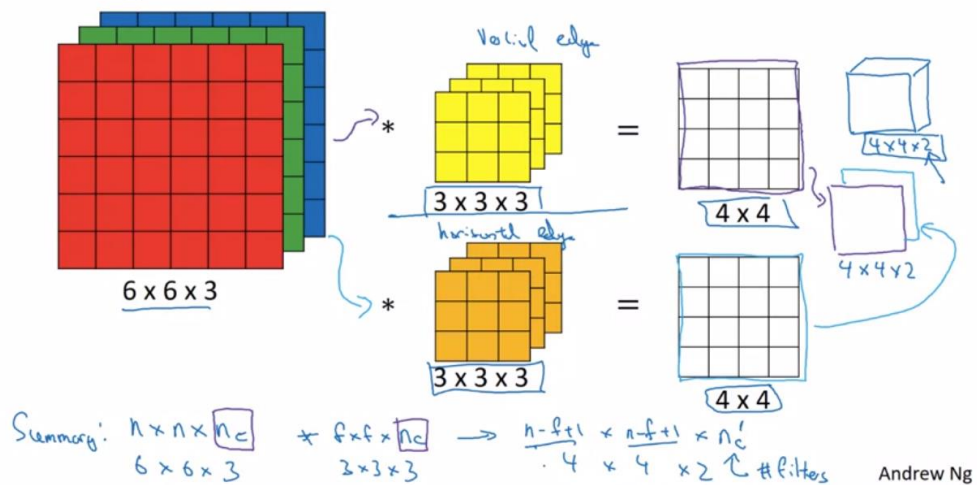
Andrew Ng

Convolutions on RGB image



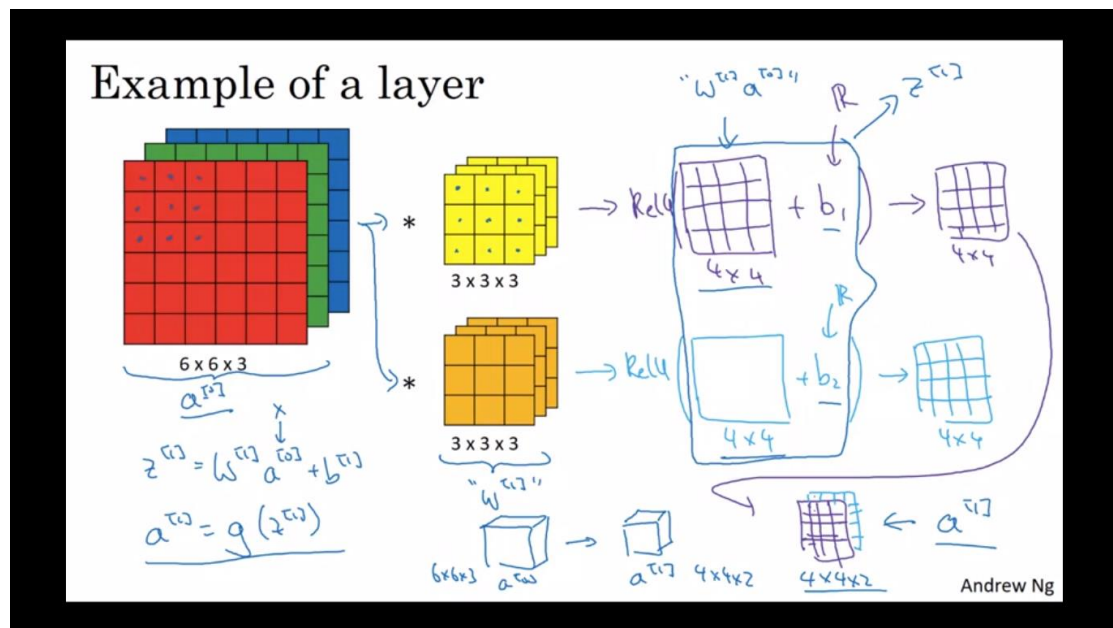
If more than one filter type is used, then the output is volume not 2D.
for example, if we want to detect both vertical and horizontal edges:

Multiple filters



Where n_c is the number of channels (colors) and n_c' is the number of filter types used.

One-layer CNN:



The input is $n \times n \times n_c$ while output is $(n-f+1) \times (n-f+1) \times n_c'$.

No matter the original image size, the parameters needed for one-layer CNN are the same. This is because it depends on the filters and biases only.

Example:

Number of parameters in one layer

If you have 10 filters that are $3 \times 3 \times 3$ in one layer of a neural network, how many parameters does that layer have?

Handwritten calculations:

27 parameters.
+ bias
→ 28 parameters.

280 parameters.

Andrew Ng

Summary of notations for layer l NN:

Summary of notation

If layer l is a convolution layer:

$f^{[l]}$ = filter size

$p^{[l]}$ = padding

$s^{[l]}$ = stride

$n_c^{[l]}$ = number of filters

→ Each filter is: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$

Activations: $a^{[l]} \rightarrow n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

Weights: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$

bias: $n_c^{[l]} - (1, 1, 1, n_c^{[l]})$ ← #f: (f: in layer l.

Input: $n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$

Output: $n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

$$n_{HW}^{[l]} = \left\lfloor \frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

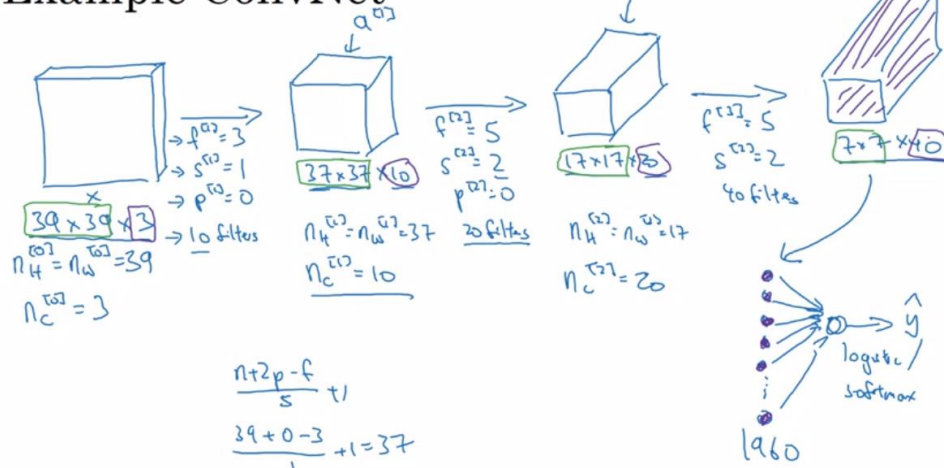
$$A^{[l]} \rightarrow m \times n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$$

Andrew Ng

Where $A^{[l]}$ is the vectorization symbol for m trials.

Example of 3-layers CNN:

Example ConvNet



Andrew Ng

Note that n_H and n_W decreases while n_c increases.

Types of layers:

Types of layer in a convolutional network:

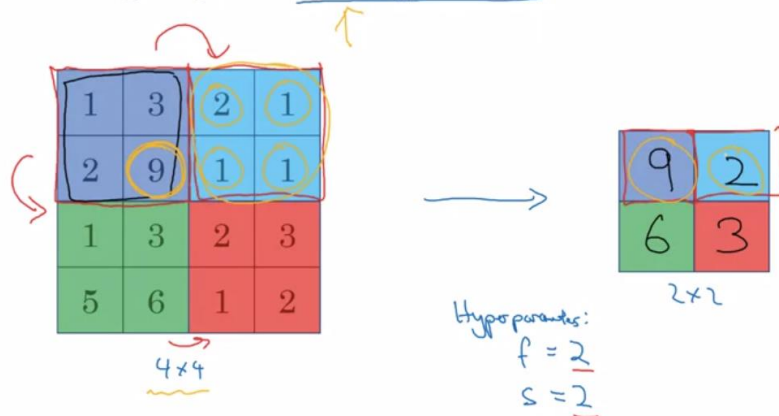
- Convolution (CONV) ←
- Pooling (POOL) ←
- Fully connected (FC) ←

Andrew Ng

Pooling:

1) Max Pooling, where we take the highest value:

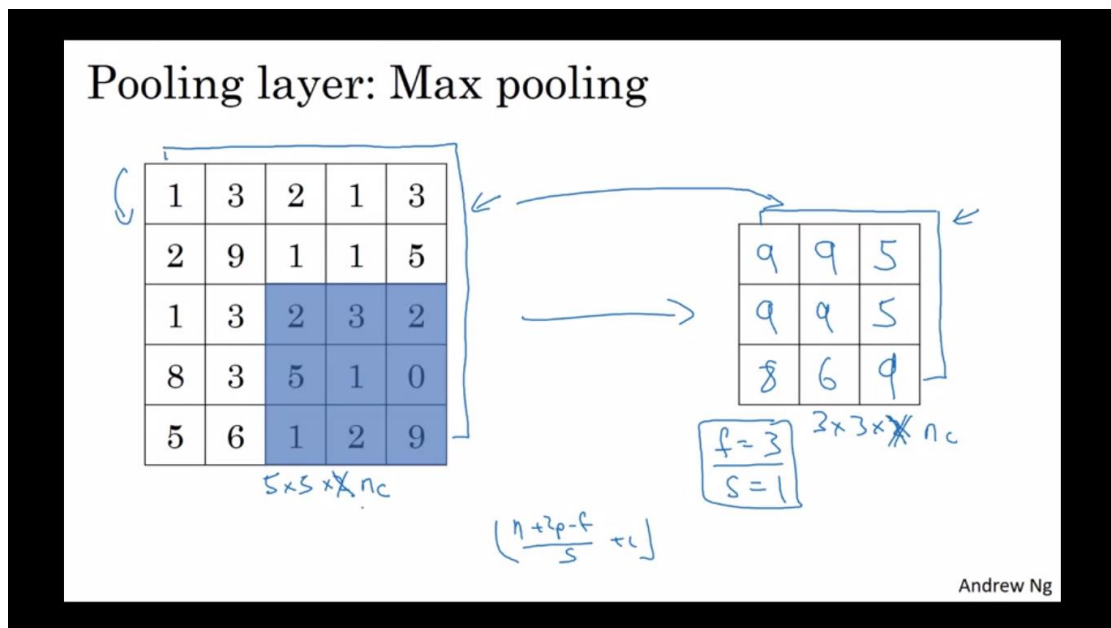
Pooling layer: Max pooling



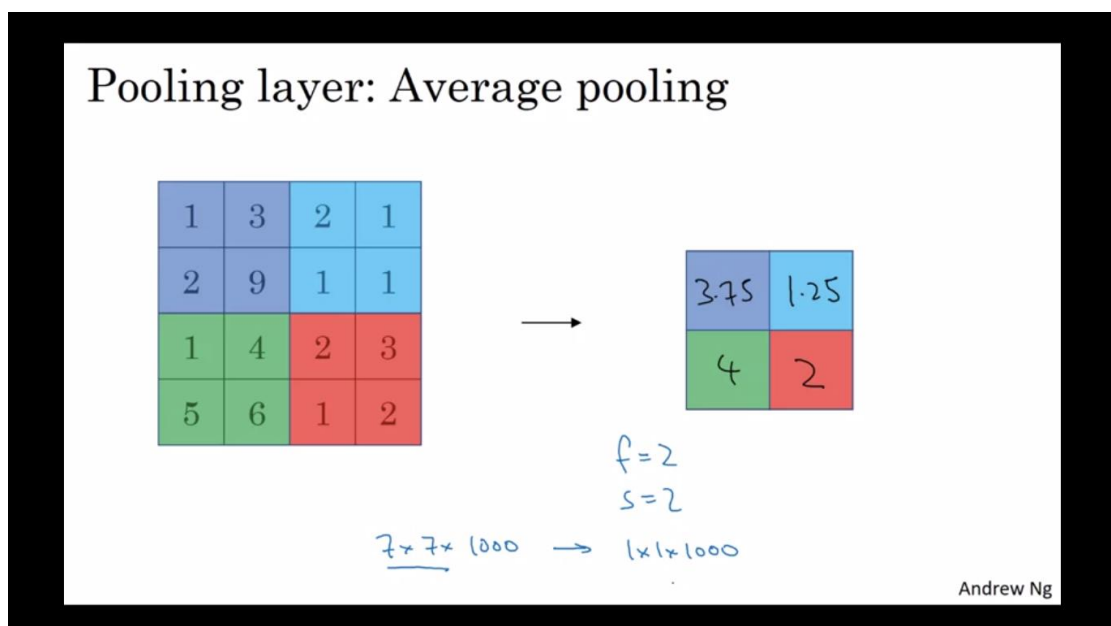
Andrew Ng

The gradient descent doesn't work on this pooling, as there are no parameters to change or to be learned.

The output and input have the same number of channels:



2) Average Pooling:



Used to collapse the dimension in Deep NN.

The Max Pooling is more famous than Average Pooling.

It is preferred not to use padding:

Summary of pooling

Hyperparameters:

- f : filter size $f=2, s=2$
- s : stride $f=3, s=2$
- Max or average pooling
- \rightarrow ~~p : padding.~~
- No parameters to learn!

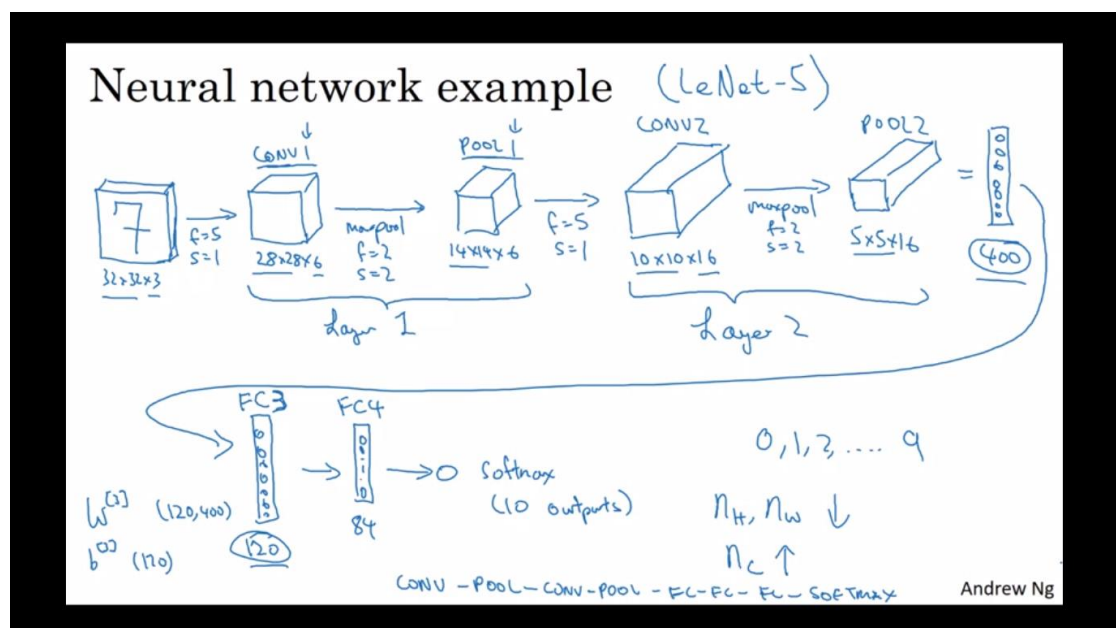
$$n_H \times n_W \times n_C$$

$$\downarrow$$

$$\left\lfloor \frac{n_H - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_W - f}{s} + 1 \right\rfloor \times n_C$$

Andrew Ng

CNN example:



Note that pool is not considered as layer, as there are no parameters in it. So, we include it with the CONV stage as a single layer.

FC means fully connected layer, like Deep NN taken in previous weeks.

Neural network example

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072 $a^{(0)}$	0
CONV1 (f=5, s=1)	(28,28,8)	6,272	208 ←
POOL1	(14,14,8)	1,568	0 ←
CONV2 (f=5, s=1)	(10,10,16)	1,600	416 ←
POOL2	(5,5,16)	400	0 ←
FC3	(120,1)	120	48,001 } 10,081 }
FC4	(84,1)	84	
Softmax	(10,1)	10	841

Andrew Ng

Here are the 5 typos:

1. 208 should be $(5*5*3 + 1) * 8 = 608$

2. 416 should be $(5*5*8 + 1) * 16 = 3216$

3. In the FC3, 48001 should be $400*120 + 120 = 48120$, since the bias should have 120 parameters, not 1

4. Similarly, in the FC4, 10081 should be $120*84 + 84$ (not 1) = 10164

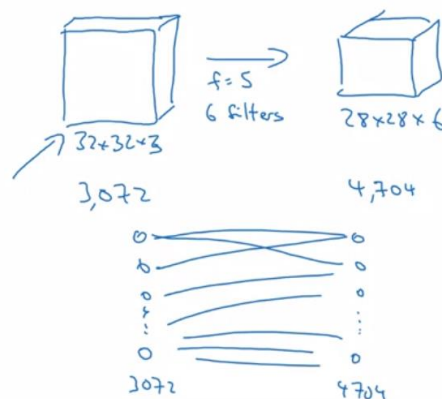
(Here, the bias is for the fully connected layer. In fully connected layers, there will be one bias for each neuron, so the bias become In FC3 there were 120 neurons so 120 biases.)

5. Finally, in the softmax, 841 should be $84*10 + 10 = 850$

Advantages of convolution layers over fully connected layers:

- 1- Parameter sharing
- 2- sparsity of connections

Why convolutions



$$5 \times 5 = 25$$

$$26$$

$$6 \times 26 = 156 \text{ parameters}$$

$$3,072 \times 4,704 \approx 14M$$

Andrew Ng

Starting around 2:15 minute, the number of parameters should have been:

$$(5 \times 5 \times 3 + 1) \times 6 = 456$$

This is based on the equation:

$$(f^{[l]} \times f^{[l]} \times n_c^{[l-1]} + 1) \times n_c^{[l]}$$

$f^{[l]}$ is the filter height (and width).

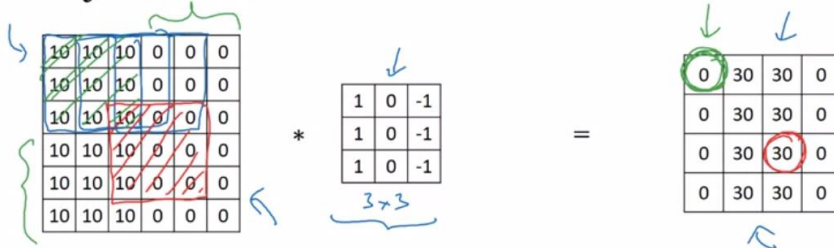
$n_c^{[l-1]}$ is the number of channels in the previous layer.

$n_c^{[l]}$ is the number of channels in the current layer.

The "1" is the bias term.

(It was $(5 \times 5 + 1) \times 6 = 156$ in the video.)

Why convolutions



Parameter sharing: A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

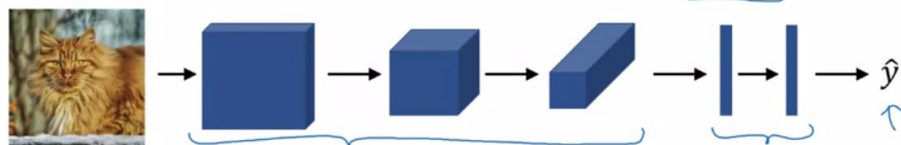
→ **Sparsity of connections:** In each layer, each output value depends only on a small number of inputs.

Andrew Ng

Training CNN:

Putting it together

Training set $(x^{(1)}, y^{(1)}) \dots (x^{(m)}, y^{(m)})$.



$$\text{Cost } J = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

Use gradient descent to optimize parameters to reduce J

Andrew Ng