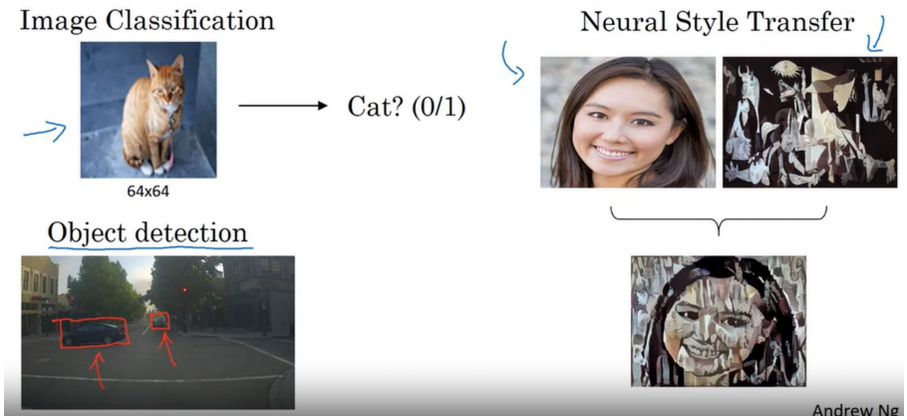
Week 1 - Convolutional Neural Networks

Wednesday, September 9, 2020 11:44 AM

① Computer Vision

→ Computer vision problem

- Image classification
- Object detection
- Neural style transfer



→ Regular neural networks become too big for images which are high dimensional data

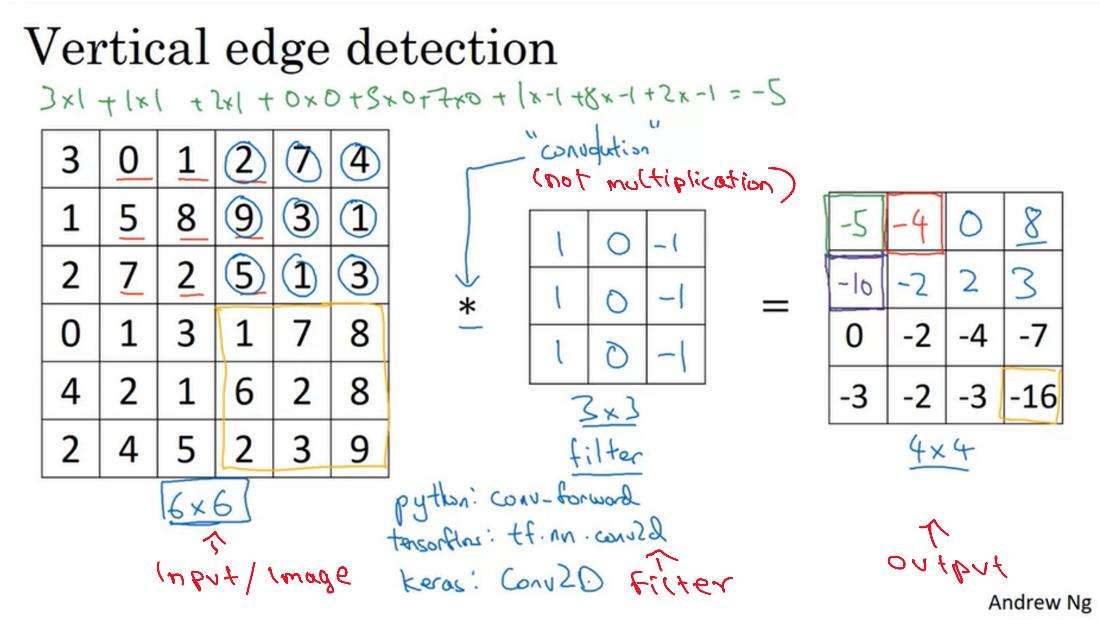
→ Hence we need CNNs

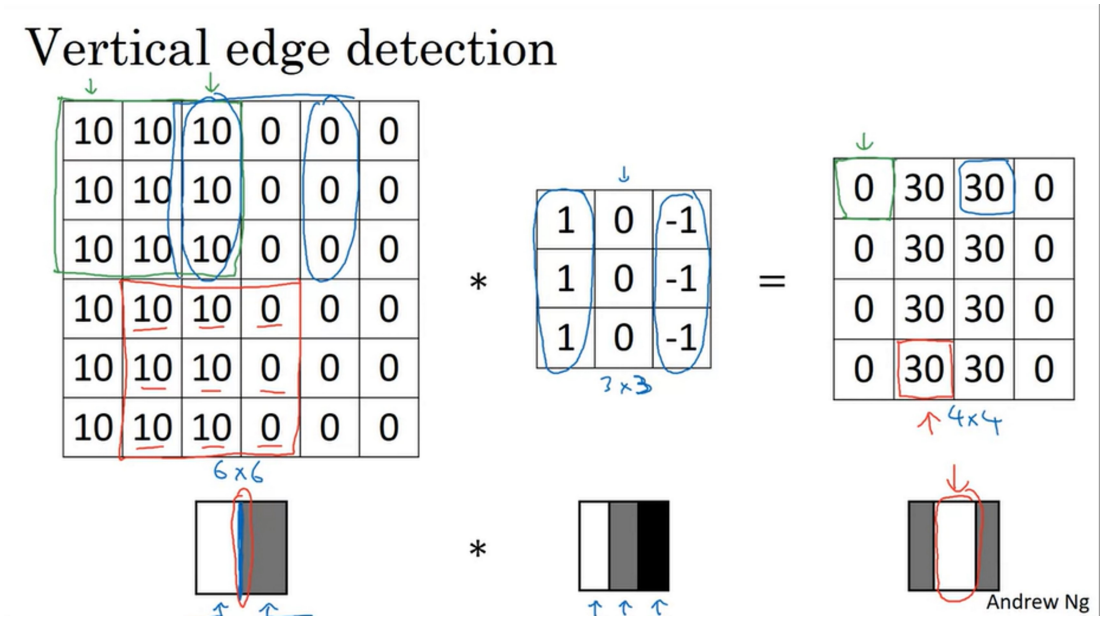
② Edge detection example

→ We try to detect horizontal & vertical edges separately

→ We perform a convolution operation to do this

→ Vertical edge detection



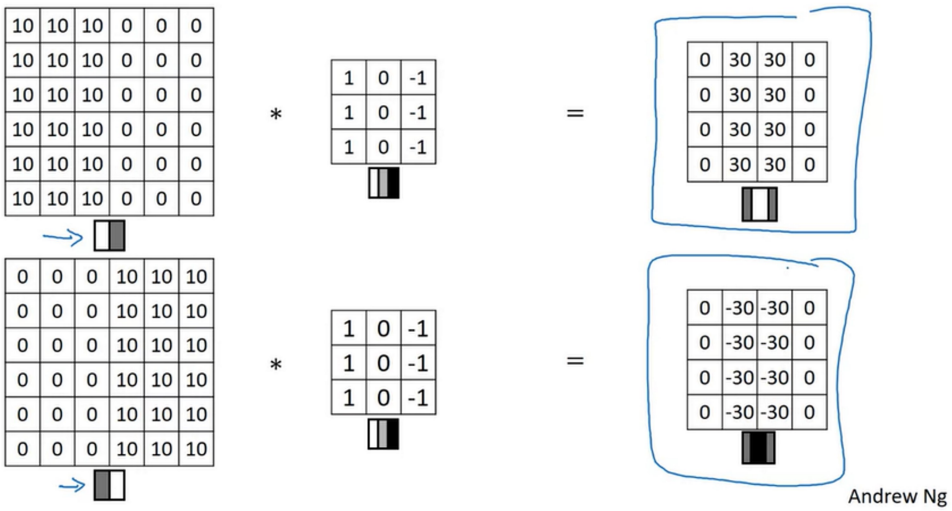


③ More edge detection

-> Differentiating dark-to-light and light-to-dark edges

- The values are negative in light to dark edges
- We can use the absolute value if we do not care about light or dark

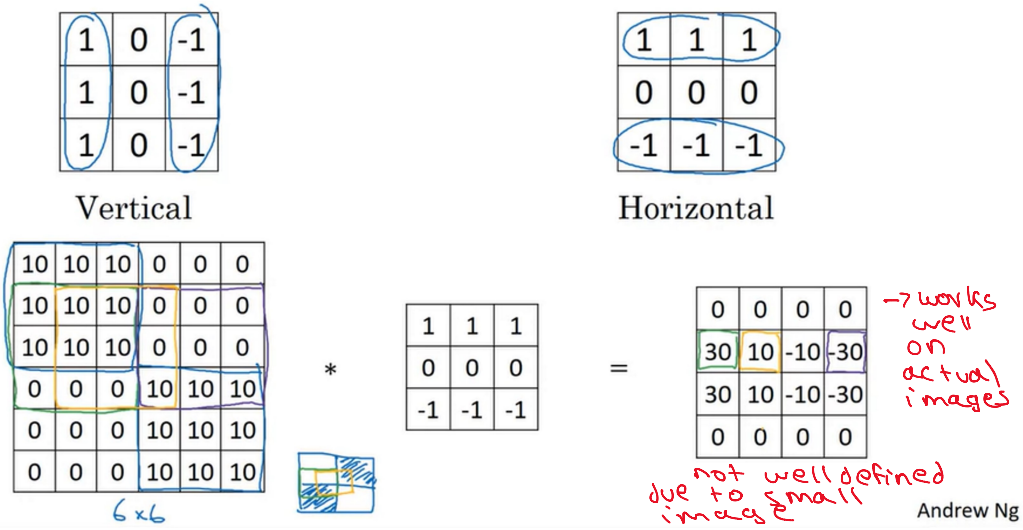
Vertical edge detection examples



Andrew Ng

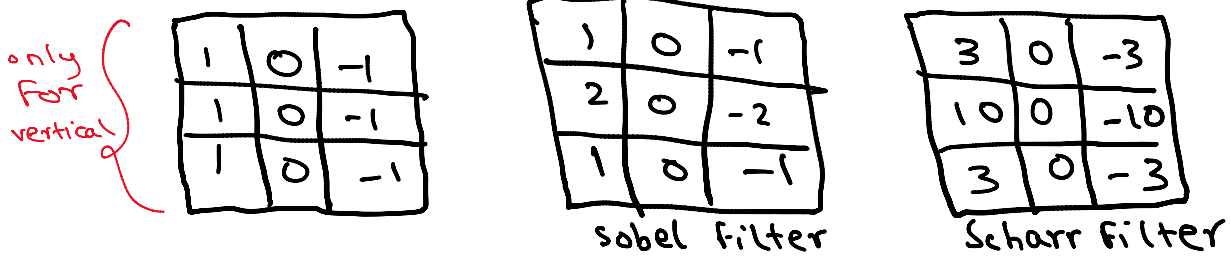
→ Vertical and Horizontal Edge detection

Vertical and Horizontal Edge Detection



Andrew Ng

→ Learning to detect edges



- We can make our filters as learnable parameters to make better filters and detect edges that are not horizontal or vertical too

④ Padding

→ when we use filters,

- $n \times n$ → image
  $f \times f$ → filter

  the output is $(n-f+1) \times (n-f+1)$

  This results in shrinking output
- The edges dont get weightage as much as center

→ we use padding to avoid both these problems

- padding is generally done with zeros
- p → amount of padding (pixels)

  $(n+p) \times (n+p)$ → image

  $(n+2p-f+1) \times (n+2p-f+1)$ → output

→ valid and same convolutions

"valid": no padding

"same": Pad so that output size is same as input size

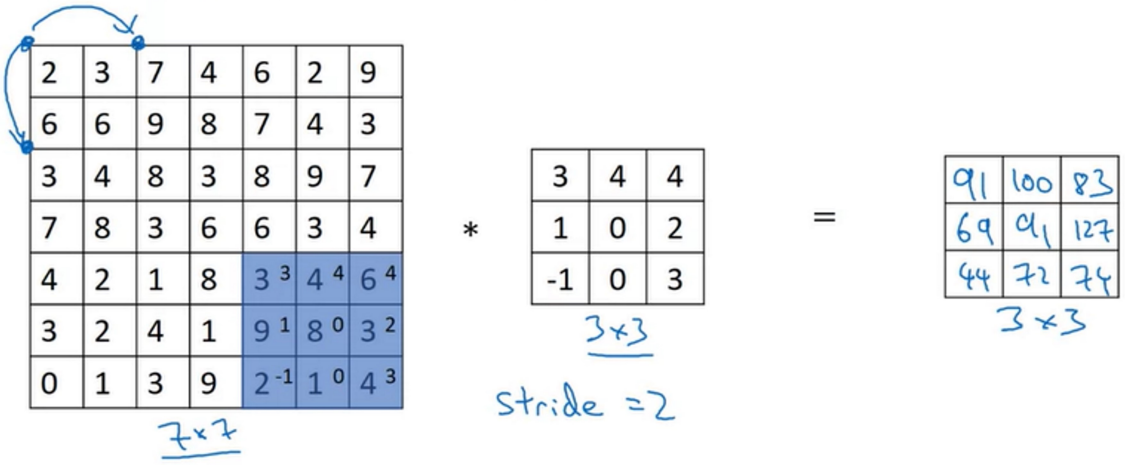
$$n+2p-f+1 = n$$

$$\Rightarrow 2p = f-1$$

$$\Rightarrow p = \frac{f-1}{2}$$ (f is almost always odd)

⑤ Strided Convolutions

Strided convolution



→ s → stride

∴ output dimension $= \left\lfloor \frac{n+2p-f}{s} \right\rfloor + 1$ ($\lfloor x \rfloor$ = floor(x))

We do not do computation if the filter goes outside the image and padding

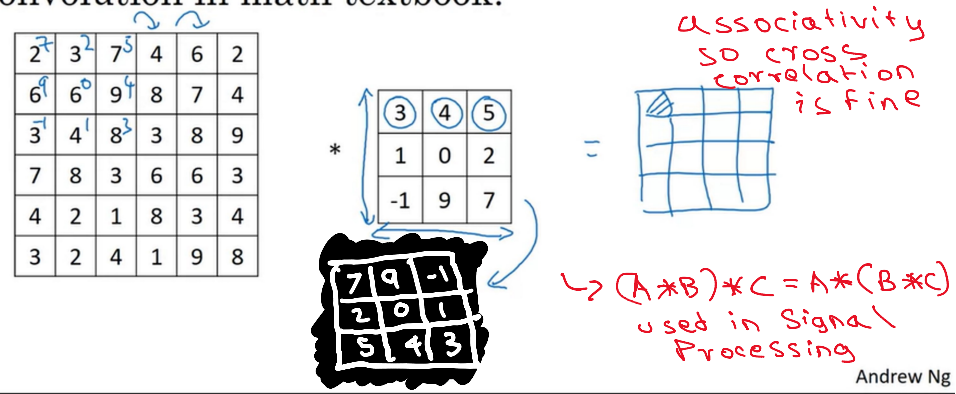
→ In math, the filter is mirrored when applying convolutions.

The operation we perform is generally called cross-correlation.

But in DL, it is referred to as just convolution.

Technical note on cross-correlation vs. convolution

Convolution in math textbook:



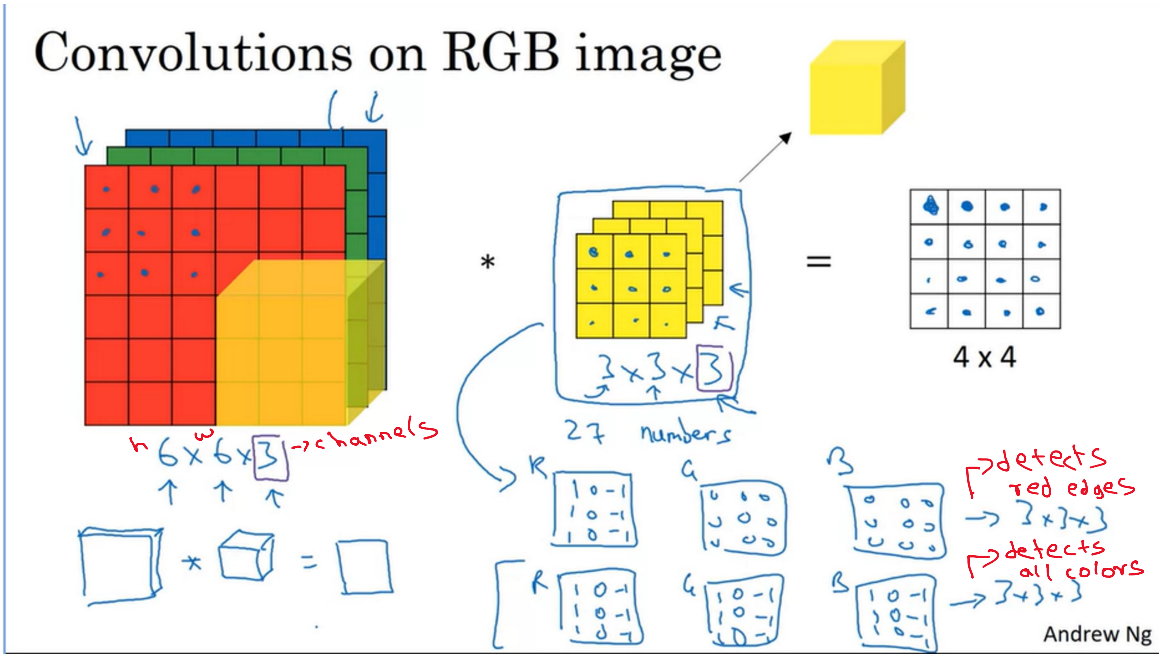
DL doesn't care about associativity so cross correlation is fine

↳ (A*B)*C = A*(B*C) used in Signal Processing

Andrew Ng

---

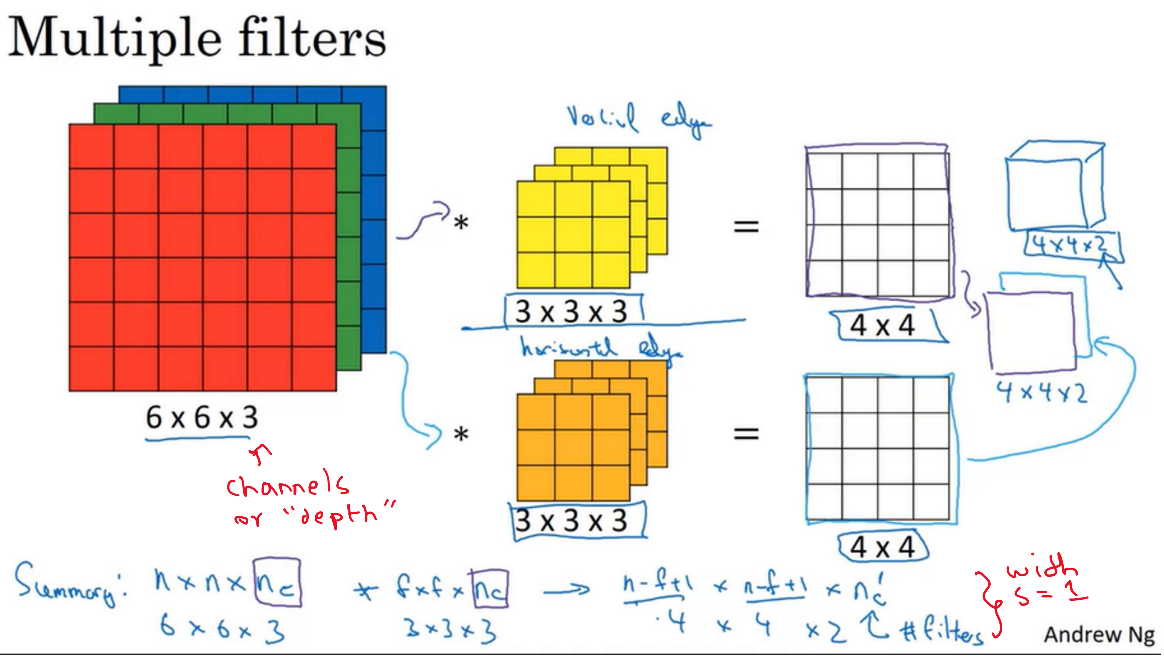
⑥ Convolutions over Volume



→ The number of channels must be same for image and filter

→ Multiple filters

- Multiple filters are used to detect horizontal and vertical edges.
- They are applied separately



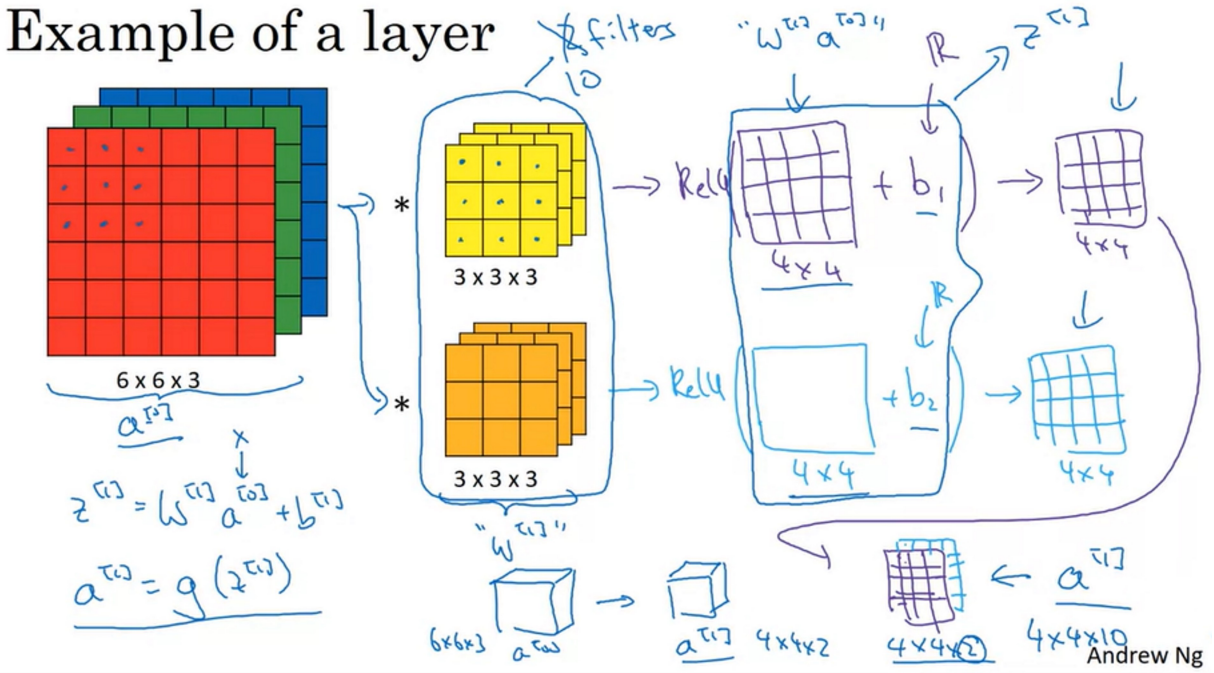
(7) One layer of a Convolutional Network

→ In a Convolutional Network, the image is the $a^{[0]}$, the filters are $w^{[1]}$.

→ We add a constant b to the output of the filter before applying the activation function (eg. ReLU)

output of the filter before applying the activation function (eg. ReLU)

→ we combine the outputs of all filters to give $a^{[l]}$



→ Notation

- $f^{[l]}$ = Filter size in layer l
- $p^{[l]}$ = padding
- $s^{[l]}$ = stride
- $n_c^{[l]}$ = number of filters
- Each filter is: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$
- Activations $a^{[l]} \to n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$
- $A^{[l]} \to m \times n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$
  ↳ mini-batches
- weights: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$
  no. of filters in layer l
- bias: $n_c^{[l]} - (1, 1, 1, n_c^{[l]})$

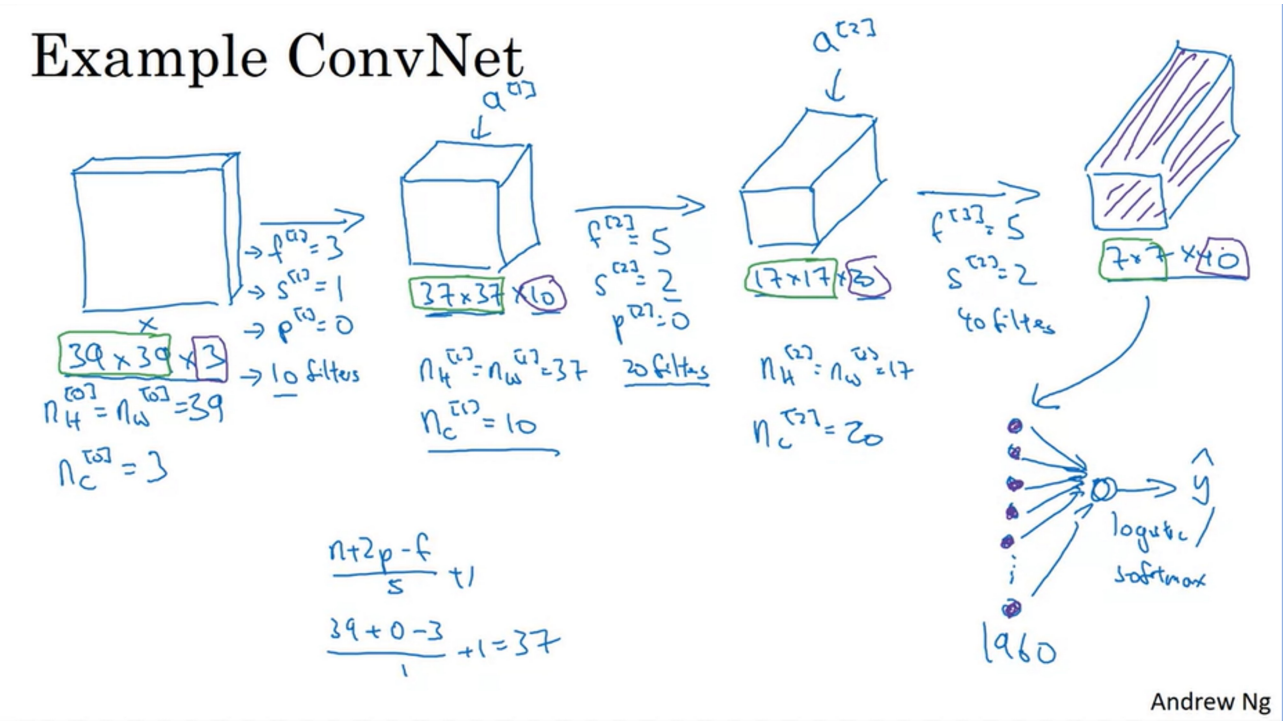
Input: $n_H^{[l-1]} \times n_w^{[l-1]} \times n_c^{[l-1]}$

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

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

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

(8) A simple Convolutional Network example



→ As we go deeper, image size reduces whereas the number of channels increases

→ Types of layer in a convolutional Network

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

⑨ Pooling Layers

→ Types of pooling

- Max pooling (Mostly used)
- Average pooling

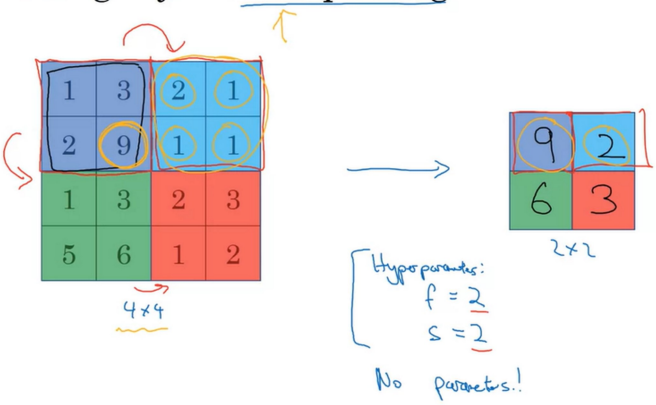
→ Pooling chooses the maximum value in the input (within every filter size) and outputs it

→ It has no parameters but only Hyperparameters

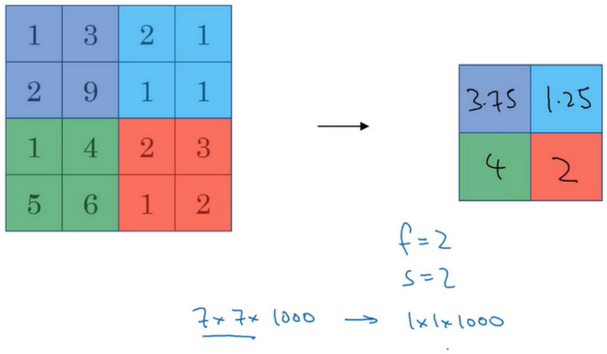
f: filter size (2, 3 ...)
s: stride (2 ...)

very rarely padding is applied

Pooling layer: Max pooling

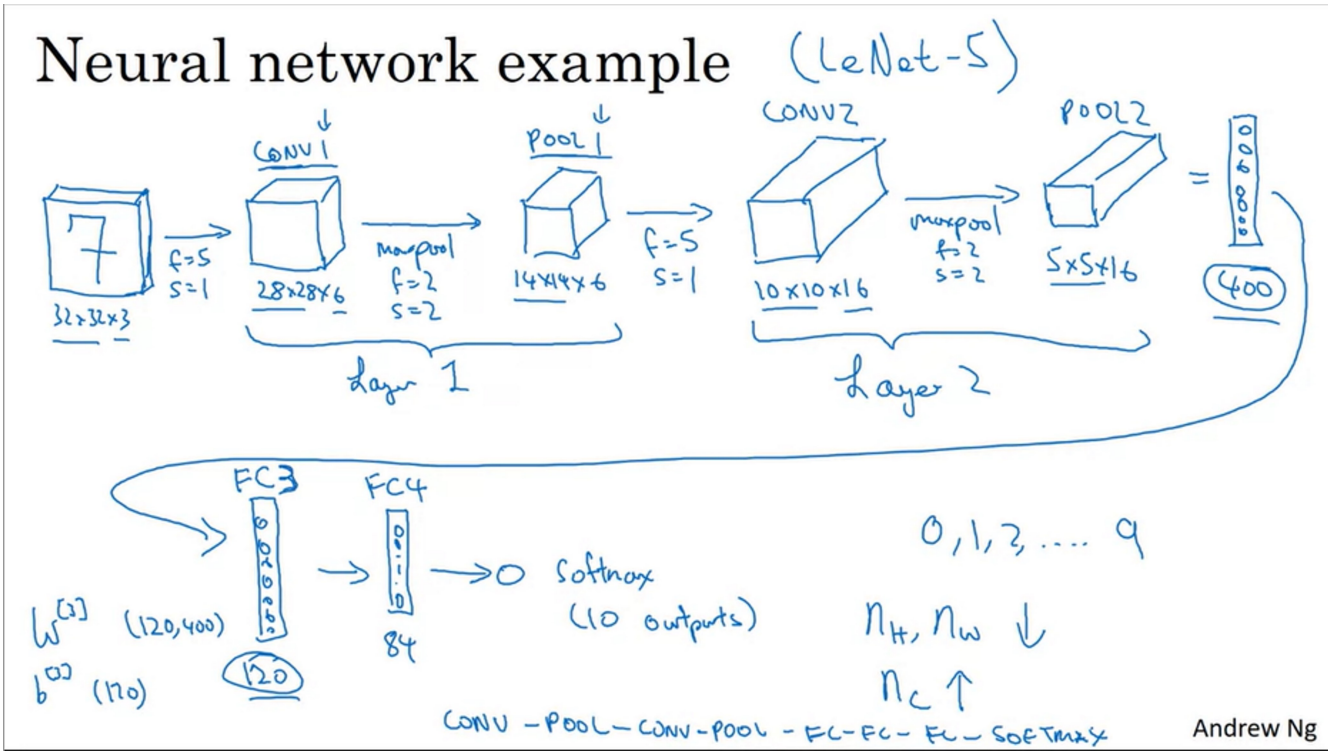


Pooling layer: Average pooling



7×7×1000 → 1×1×1000

(10) CNN Example



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	608
POOL1	(14,14,8)	1,568	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	3216
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48120
FC4	(84,1)	84	10164
Softmax	(10,1)	10	850

Andrew Ng

---

(11) 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
- Reduces no. of parameters greatly

→ Sparsity of connections

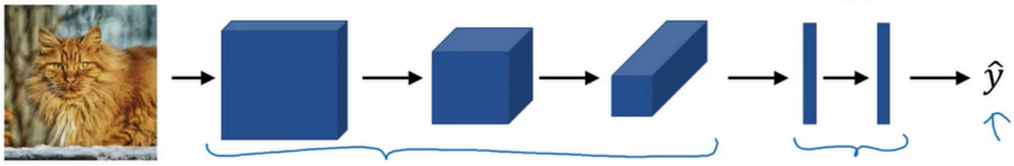
- In each layer, each output value depends only on a small number of inputs
- reduces overfitting

→ Translation invariance

- Changing few features would still result in a good detection

Putting it together

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



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