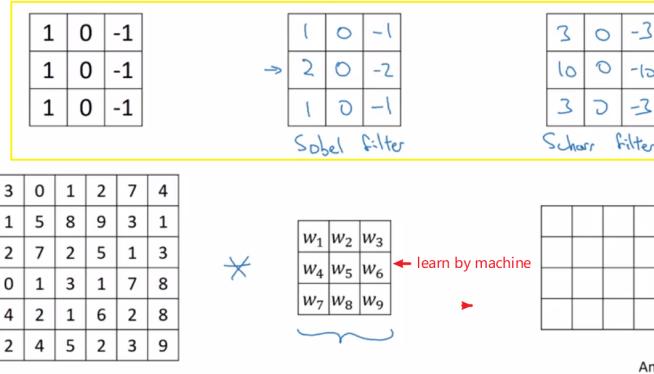


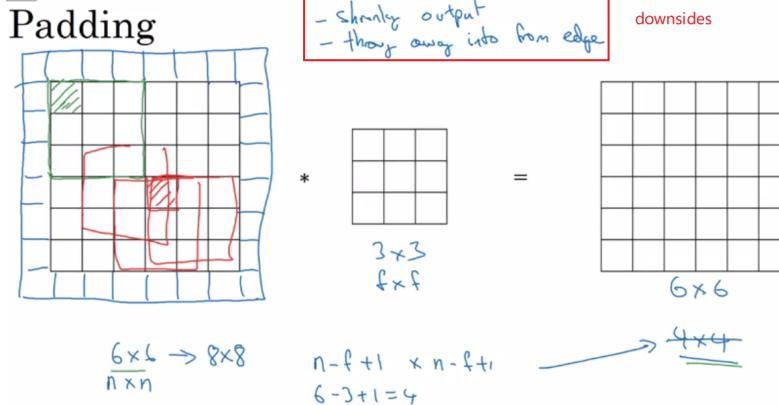
- we don't need to hand craft but instead using DL to learn them.
- Learning to detect edges**



Andrew Ng

padding

- in previous slide, we see two **downsides** of convolution
 - the matrix keep **shrinking**.
 - throwing away a lot of **information** that are in the edges. (central units would be counted more times than the edges units.)
- to solve this, we use **padding**



- formula here:
 - padding amount P**
 - two kinds of convolutions
 - valid (shrinking)
 - same(not shrinking)

Valid and Same convolutions

$$\text{"Valid": } \begin{matrix} \nearrow n \text{ padding} \\ n \times n \\ 6 \times 6 \end{matrix} * \begin{matrix} f \times f \\ 3 \times 3 \end{matrix} \rightarrow \begin{matrix} n-f+1 \\ 6-3+1=4 \end{matrix} \times \begin{matrix} n-f+1 \\ 4 \times 4 \end{matrix}$$

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

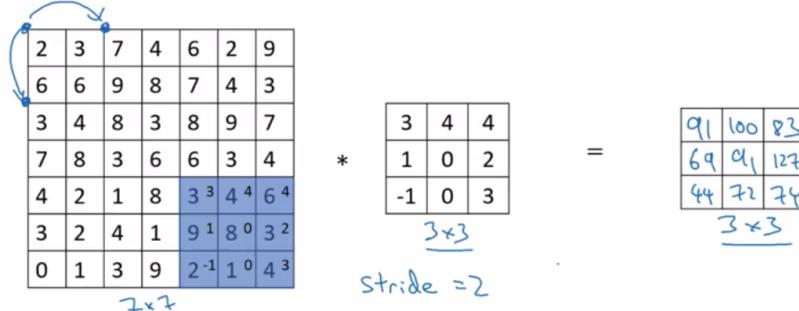
$$\begin{matrix} n+2p-f+1 & \times n+2p-f+1 \\ n+2p-f+1 = n & \Rightarrow p = \frac{f-1}{2} \end{matrix}$$

*f is usually odd
because the matrix can have a center.*

strided convolution

- stride s

Strided convolution



$$n \times n \text{ image} * f \times f \text{ filter}$$

padding p stride s

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

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

Andrew Ng

Summary of convolutions

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

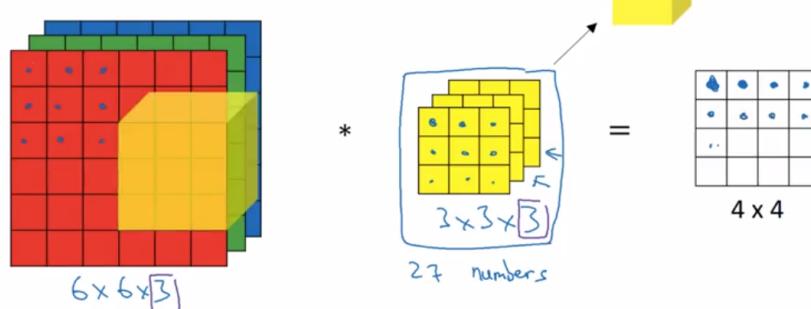
padding p stride s

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

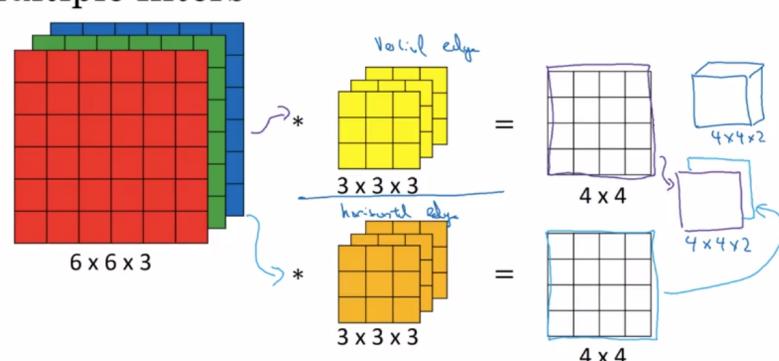
- over volume

- we call each layer in the third D as **channel**

Convolutions on RGB image



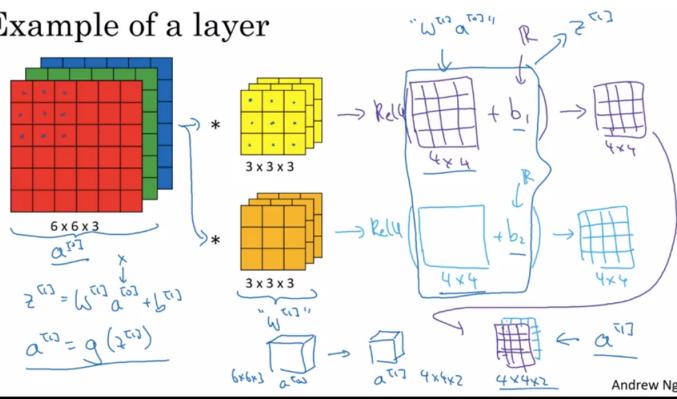
- Multiple filters



- one layer of convolutional NN

- o intuition

Example of a layer



Andrew Ng

- o notation

Summary of notation

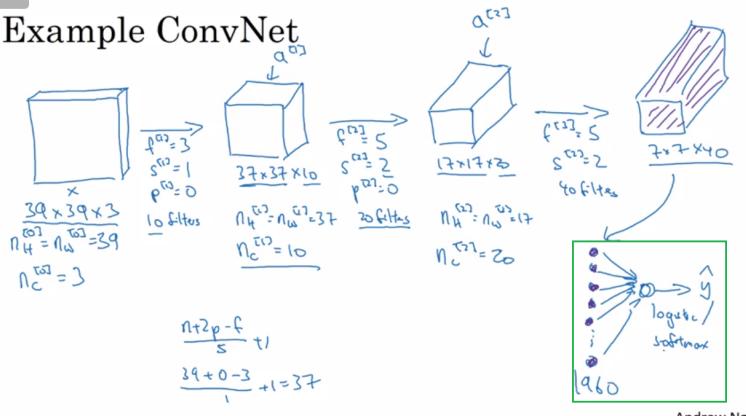
If layer l is a convolution layer:

$f^{[l]}$ = filter size	Input: $\frac{n_H^{[l-1]} \times n_W^{[l-1]}}{n_C^{[l-1]}}$
$p^{[l]}$ = padding	Output: $\frac{n_H^{[l]} \times n_W^{[l]}}{n_C^{[l]}}$
$s^{[l]}$ = stride	$n_H^{[l]} = \left\lfloor \frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$
$n_c^{[l]}$ = number of filters	Activations: $a^{[l]} \rightarrow m \times n_H^{[l]} \times n_W^{[l]} \times n_C^{[l]}$
→ Each filter is: $f^{[l]} \times f^{[l]} \times n_C^{[l]}$	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: bias in layer l .

Andrew Ng

- o example

Example ConvNet



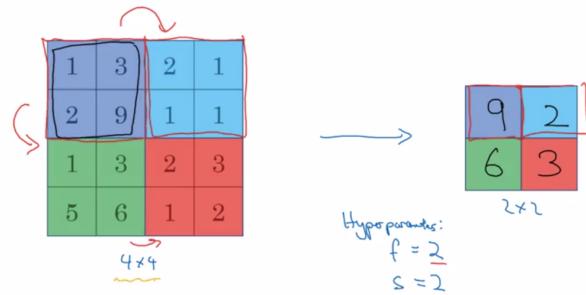
Andrew Ng

pooling

- o intuition of **Max pooling**

- select the biggest values in each region

Pooling layer: Max pooling



Hypoparameters:
 $f = 2$
 $s = 2$

- o **average pooling**

- taking the averages of the values instead of taking the max values.

- o summary: fixed layer. No parameter to learn.

Summary of pooling

Hyperparameters:

$$f : \text{filter size} \quad f=2, s=2$$
$$s : \text{stride} \quad f=3, s=2$$

conventional values

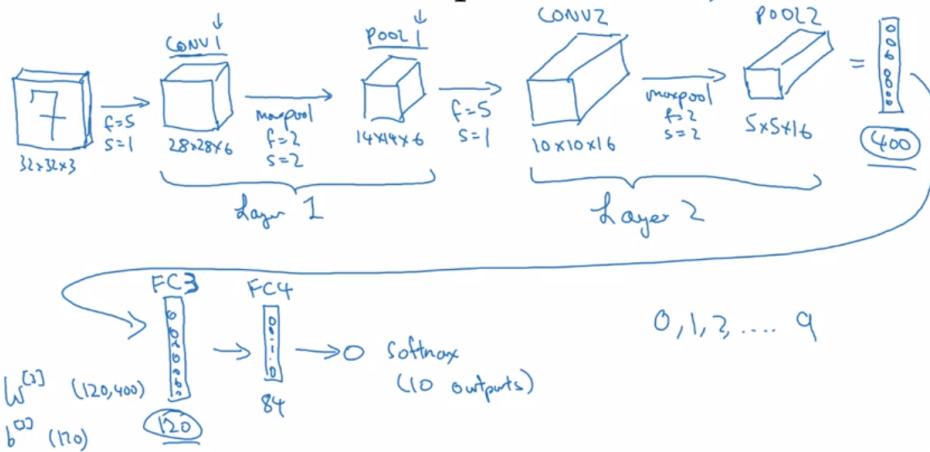
Max or average pooling

$\rightarrow p: \text{padding}$

No parameters to learn!

- Ex. a integrated CNN

Neural network example (LeNet-5)



Andrew Ng

- # **FC** here means fully connected layer

why using convolutions

- if using fully ANN, there are too many parameters to learn for image
- convolutions

- Sharing
- Sparsity

Why convolutions

A diagram showing the application of a 3x3 kernel to a 5x5 input matrix. The kernel is labeled 3×3 . The result is a 3x3 output matrix. Arrows indicate that the same kernel value (1) is reused across different spatial locations in the input, demonstrating shared weights.

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.

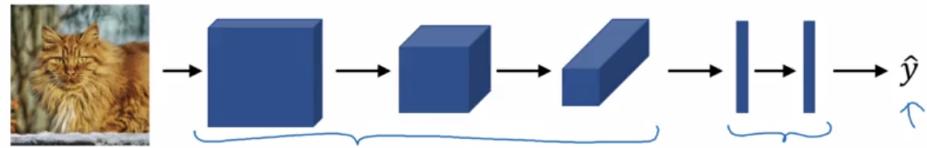
\rightarrow **Sparsity of connections:** In each layer, each output value depends only on a small number of inputs.

- Andrew Ng

Putting it together

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

$\underline{\omega}, \underline{b}$



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

Use gradient descent to optimize parameters to reduce J

o