Edge Detection

"convolution" = "cross-correlation"

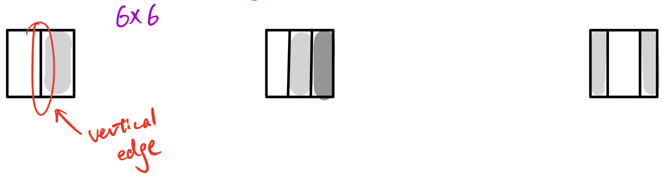
$$\begin{matrix} 3 & 0 & \cdots & 4 \\ \vdots & \ddots & & \vdots \\ 2 & - & \cdots & 9 \end{matrix} * \begin{matrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{matrix} = \begin{matrix} -5 & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & 2 \end{matrix}$$

6 x 6   3 x 3 filter   4 x 4

tf.nn.conv2d

$$\begin{matrix} 10 & \cdots & 10 & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 10 & \cdots & 10 & 0 & \cdots & 0 \end{matrix} * \begin{matrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{matrix} = \begin{matrix} 0 & 30 & 30 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 30 & 30 & 0 \end{matrix}$$

6x6   bright   dark   4x4



vertical edge detector $\Rightarrow$ Sobel Filter

$$\begin{matrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{matrix}$$

Scharr Filter

$$\begin{matrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{matrix}$$

Learning to detect edge

$$\begin{matrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{matrix}$$

Padding & Strided Convolutions

6 x 6 * 3x3 filter = 4 x 4

$n \times n$ * $f \times f$ filter = $(n-f+1) * (n-f+1)$

Cons : 1) image shrinking

2) lose edge info

$\Rightarrow$ Padding

* 3x3 filter = 6 x 6

8x8

p = padding = 1

Valid and Same convolution

"valid" : $n \times n \ * \ f \times f \ = \ (n-f+1) \times (n-f+1)$

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

$\Rightarrow \ \underline{p = \frac{f-1}{2}}$ ← f is usually odd

$\Rightarrow$ strided

$7 \times 7 \ * \ 3 \times 3 \ = \ 3 \times 3$

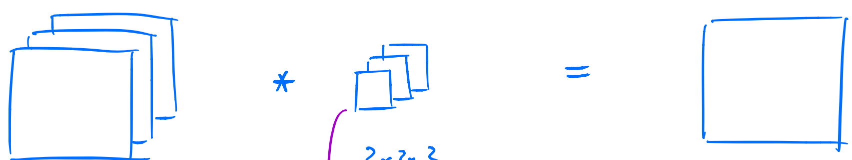
jump by 2

(stride = 2)

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

padding p stride s

Convolution over Volume



$6 \times 6 \times 3 \ * \ 3 \times 3 \times 3 \ = \ 4 \times 4$

height width depth

e.g. R G, B

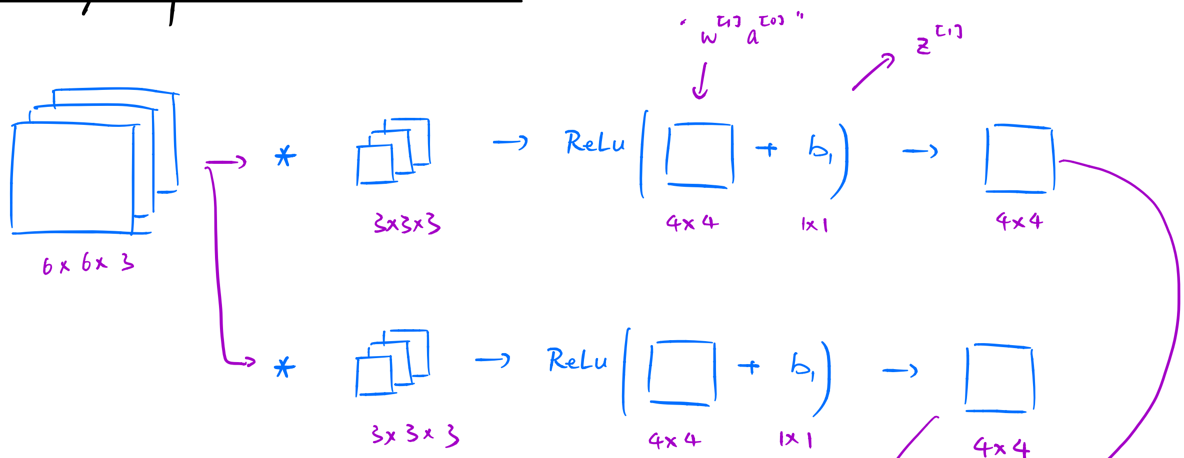
$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$ $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ $\rightarrow 3 \times 3 \times 3$

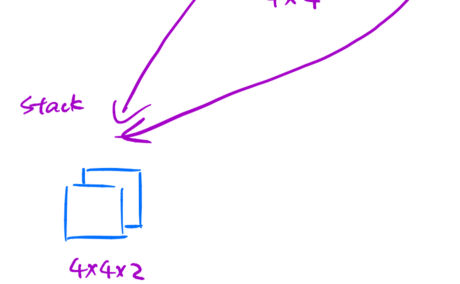
Summary $\Rightarrow$ $n \times n \times n_c \ * \ f \times f \times n_c \ \rightarrow \ (n-f+1) \times (n-f+1) \times n_c'$

e.g. $6 \times 6 \times 3 \ * \ 3 \times 3 \times 3 \qquad 4 \times 4 \times 2$

filters

One Layer of Convolutional Network





⇒ Notations:

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

$p^{[l]}$ = padding

$s^{[l]}$ = stride

$n_c^{[l]}$ = # filters

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

Output : $n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$, $n_{H,W}^{[l]} = \left\lfloor \dfrac{n_{H,W}^{[l]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$

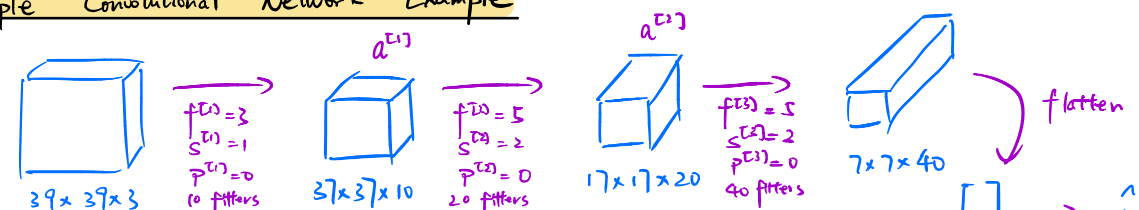
⇒ Each filter is : $f^{[l]} \times f^{[l]} \times \boxed{n_c^{[l-1]}}$

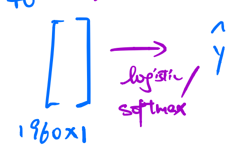
Activations : $a^{[l]} \rightarrow n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

weights : $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$ ← # filters

bias : $n_c^{[l]}$ — $(1,1,1, n_c^{[l]})$

Simple Convolutional Network Example





$n_H^{[0]} = n_W^{[0]} = 39$ $n_H^{[1]} = n_W^{[1]} = 37$ $n_H^{[2]} = n_W^{[2]} = 17$

$n_c^{[0]} = 3$ $n_c^{[1]} = 10$ $n_c^{[2]} = 20$

Types of layer in CNN:

- Convolutional (Conv)
- Pooling (Pool)
- Fully Connected (FC)

Pooling Layers

Max Pooling

e.g.

$$\begin{bmatrix} 1 & 3 & 2 & 1 \\ 2 & 9 & 1 & 1 \\ 1 & 3 & 2 & 3 \\ 5 & 6 & 1 & 2 \end{bmatrix} \longrightarrow \begin{bmatrix} 9 & 2 \\ 6 & 3 \end{bmatrix}$$

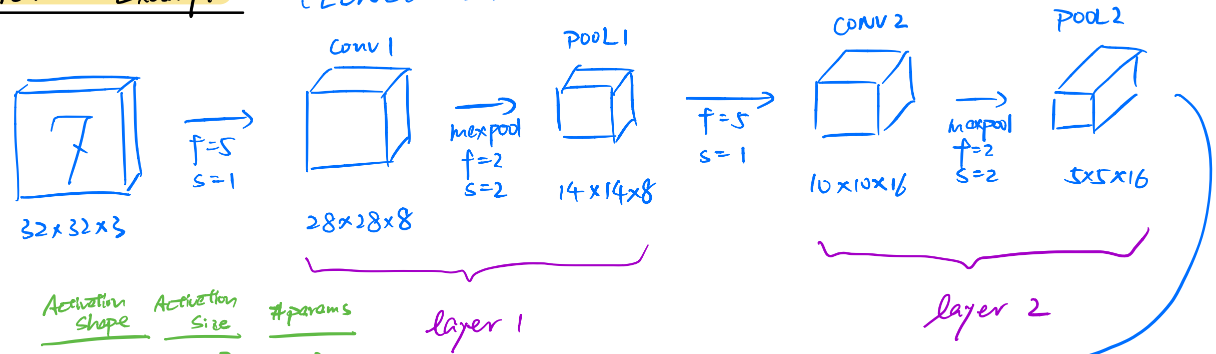
hyperparam
$f = 2$
$s = 2$

Nothing to learn!

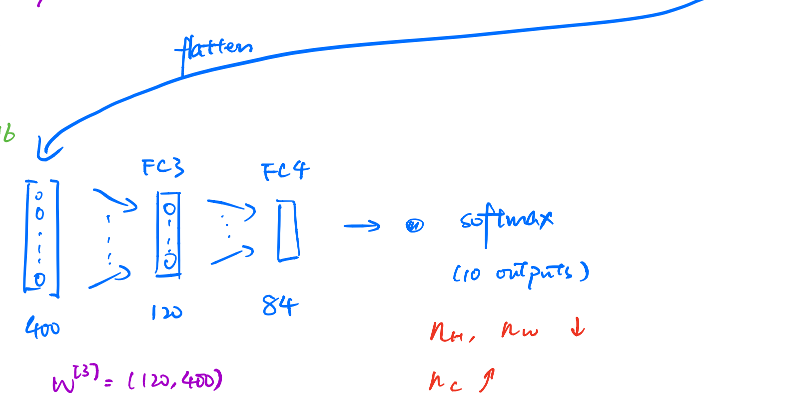
Average Pooling

CNN Example

(LeNet-5)



	Activation shape	Activation Size	#params
Input	(32,32,3)	3072	0
CONV1	(28,28,8)	6272	(5×5×3+1)×8
POOL1	(14,14,8)	1568	0
CONV2	(10,10,16)	1600	(5×5×8+1)×16
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	400×120+120
FC4	(84,1)	84	120×84+84
Softmax	(10,1)	10	84×10+10

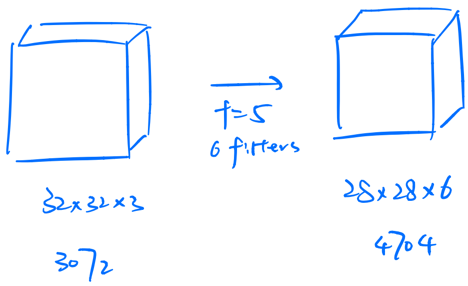


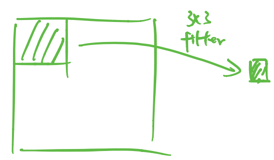
n_H, n_W ↓

n_C ↑

Pattern: CONV – POOL – CONV – POOL – FC – FC – Softmax

Why Convolution ?





With FC, $3072 \times 4704 \approx 14m$

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

⇒ Parameter Sharing

- A feature detector that's useful in one part of 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