dec1: computer vision

example:

① image classification

$$(X \rightarrow y(0,1)\ \text{cat})$$

② object detection

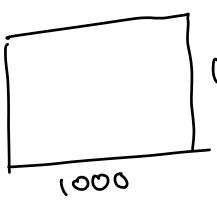
$$(X \rightarrow \text{bounding box, labels})$$

③ Neural style Transfer

$$(X + \text{style} \Rightarrow X\text{style image})$$

Deep learning on large images:

$\Rightarrow$ input features $(64 \times 64 \times 3) = 12288$



$\Rightarrow$ input features $(1000 \times 1000 \times 3) =$ 3million.

x_1 1000

x_n

$X \in \mathbb{R}^{3M}$

$\rightarrow$

$w^{[1]}$

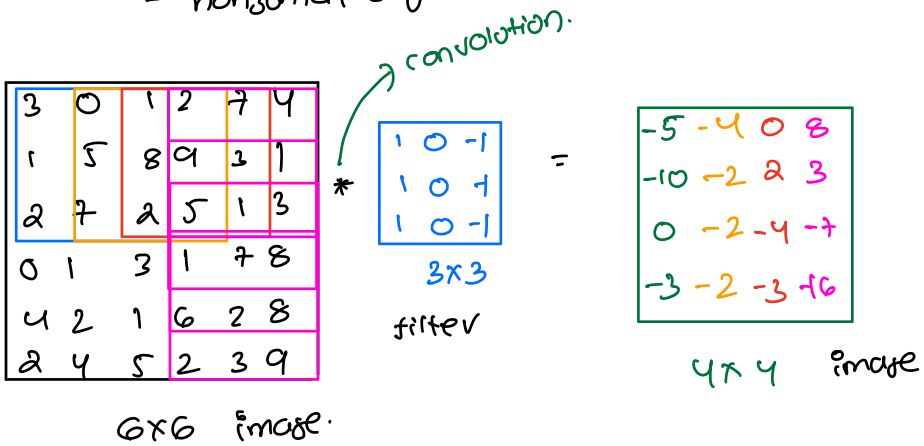
$w^{[1]} = (1000, 3M) \Rightarrow$ Very huge matrix

$\Rightarrow$ Memory issue.

$\Rightarrow$ need very large data.

deca: Edge Detection Example:-

- vertical edges
- horizontal edges

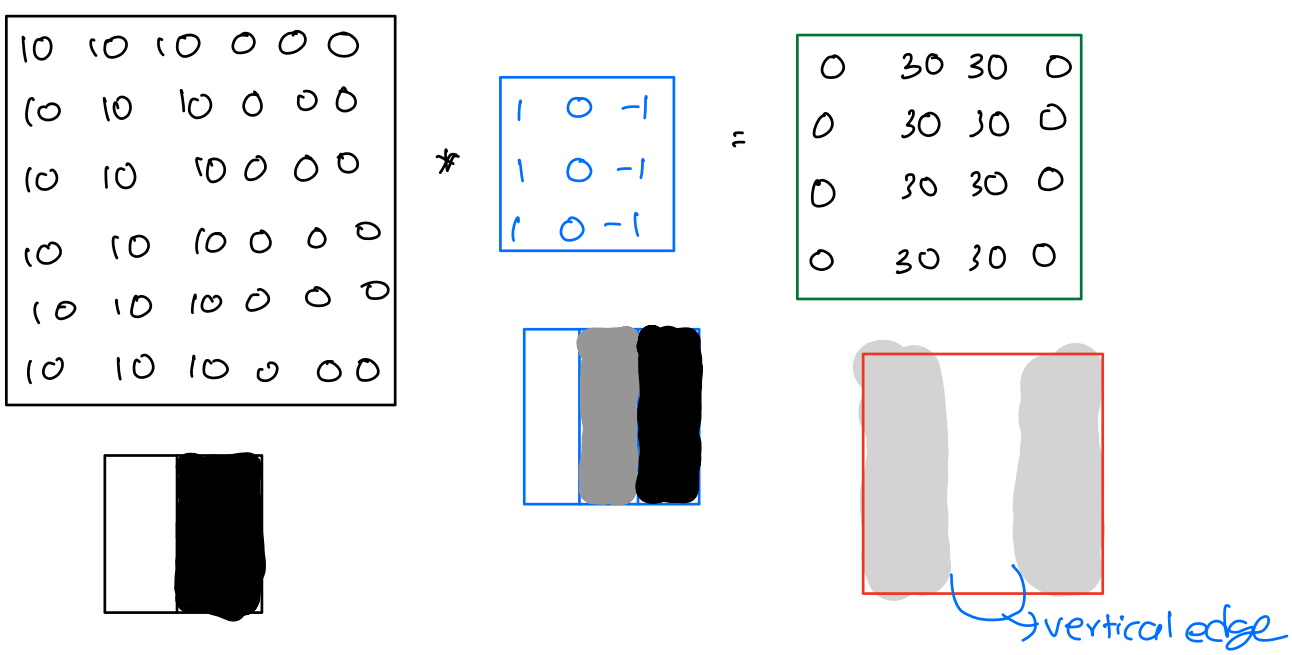


$(0,0) = 3\times1 + 0\times1 + 1\times1 + 2\times0 + 7\times0 + 4\times0 + 1\times-1 + 8\times-1 + 2\times-1 = -5$

$(0,1) = 0\times1 + 5\times1 + 7\times1 + 1\times0 + 8\times0 + 2\times0 + 2\times-1 + 9\times-1 + 5\times-1 = -4$

python: conv-forward

tensorflow: tf.nn.conv2d

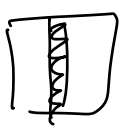


Lec 3: More edge detection:-

$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}$$

(dark to light)

$$\begin{bmatrix} 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \end{bmatrix}$$



(light to dark)

Vertical edge filter $\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$

Horizontal edge filter $\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

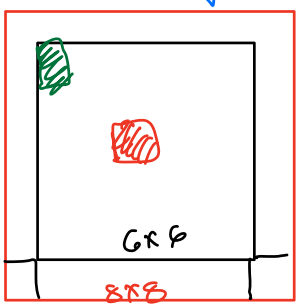
Sobel filter $\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$

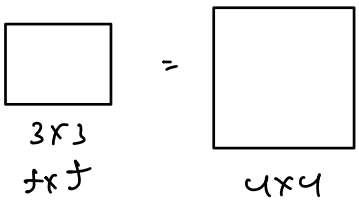
Schar filter $\begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix}$

Goal is learn this matrix, by treating is parameters.

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

dec4: padding:-





- used very less
- used very often

output dim without pad $= (n-f+1) \times (n-f+1)$

$= (6-3+1) \times (6-3+1)$

$= (4 \times 4)$

Output dim with pad $=$ p = 1 pixel

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

$= (6+2\times1-3+1) \times (6+2\times1-3+1)$

$= (6 \times 6)$ (original dim is preserved)

- used bit more with padding

Valid and Same Convolutions:-

valid: No padding $(n, n) * (f, f) = (n-f+1, n-f+1)$

Same: padding input size = output size

(n,n) with p padding $* (f,f) = (n,n)$

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

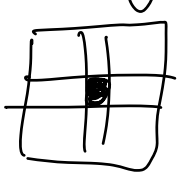
$$p = \frac{f-1}{2}$$

(1,1)
(3,3)
(5,5)
(7,7)

f is usually odd.

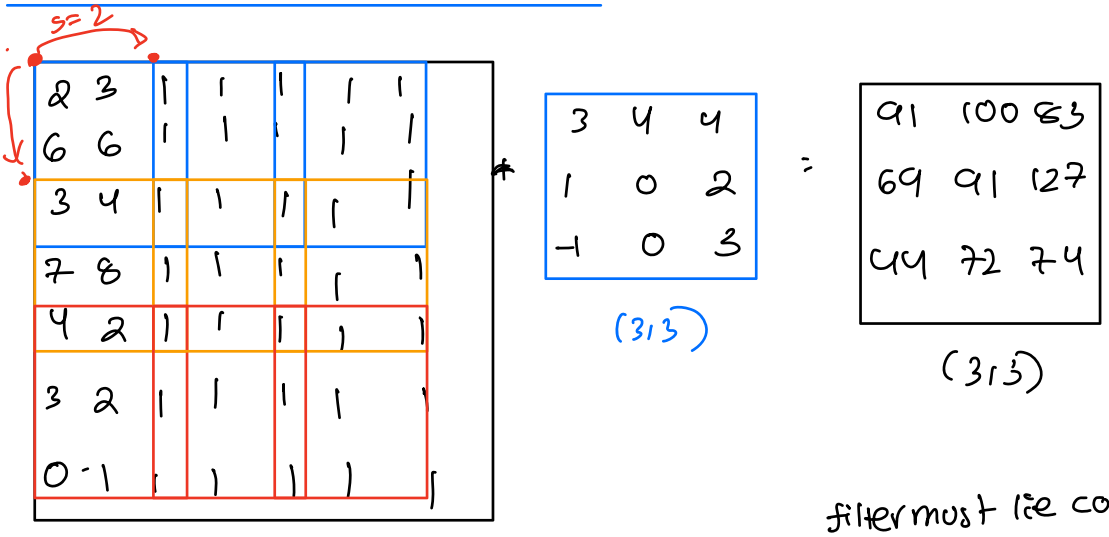
if $f = (3,3)$ then $p = 1$

if $f = (5,5)$ then $p = 2$



• position of filter.

Dec 5: Strided Convolutions:-



7×7

n×n

(3×3)

f×f

(3×3)

filter must lie completely inside the padded image.

$(n \times n) * (f \times f)$ with pad p & stride s

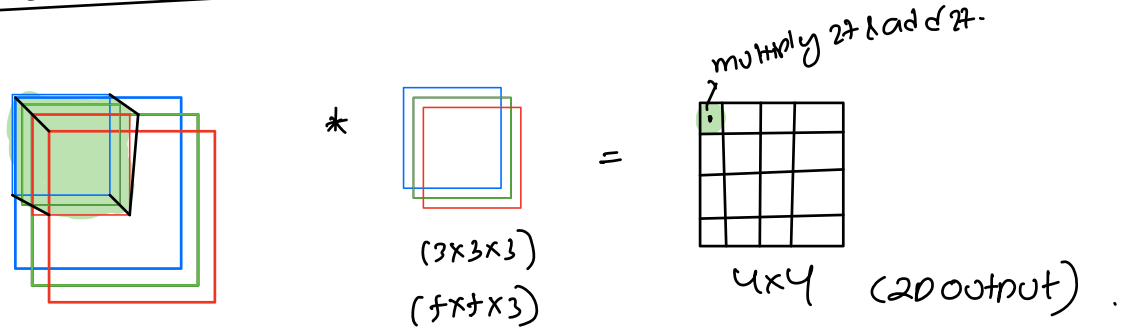
$$= \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-1}{s} + 1 \right\rfloor \quad \text{floor } [z]$$

$$= \left[\frac{7+0-3}{2} + 1 \right] \times \left[\frac{7-3}{2} + 1 \right]$$

$$= (3 \times 3)$$

dec6: convolution over volume :-

convolution over RGB image :-



#ch-image = #ch-filters

R
1 0 -1
1 0 -1
1 0 -1

G
0 0 0
0 0 0
0 0 0

B
0 0 0
0 0 0
0 0 0

→ edges in red channel.

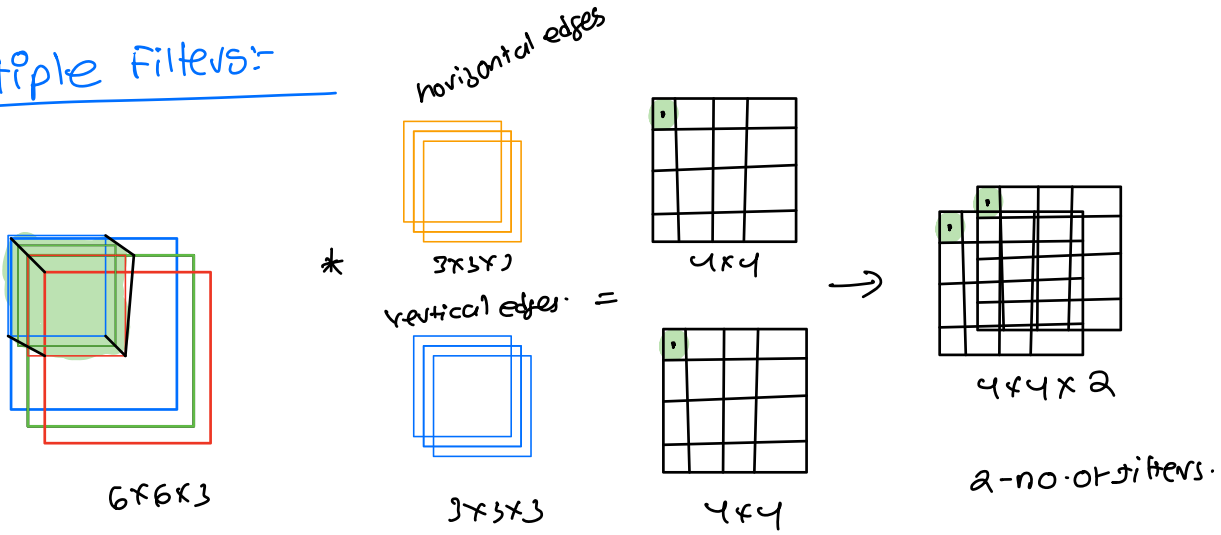
R
1 0 -1
1 0 -1
1 0 -1

G
1 0 -1
1 0 -1
1 0 -1

B
1 0 -1
1 0 -1
1 0 -1

→ edges in any colors.

Multiple Filters:-



summary:-

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

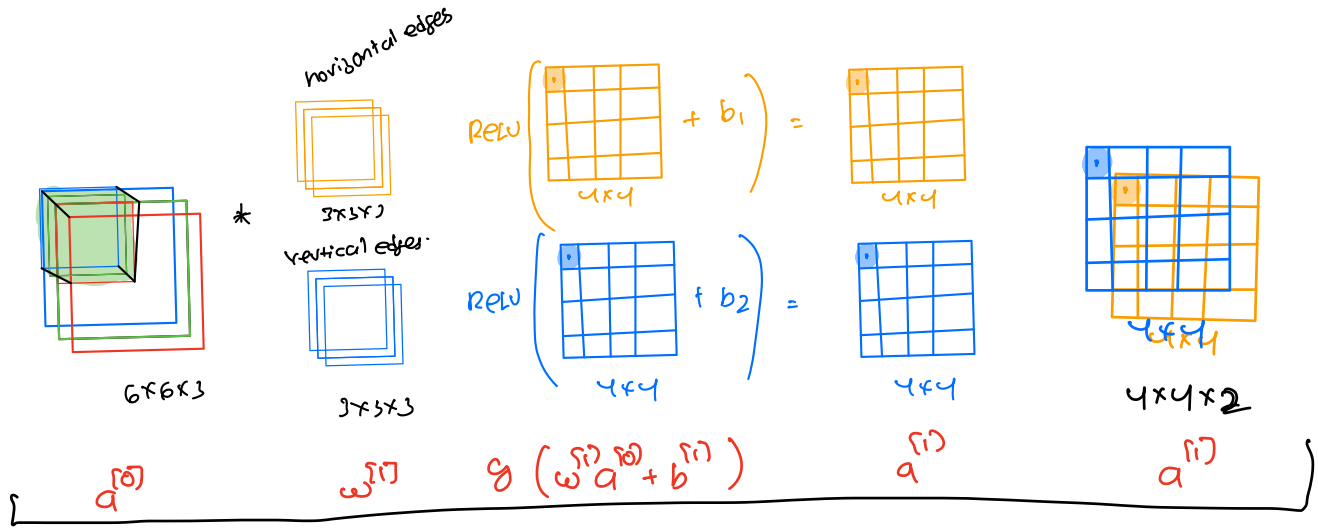
$6 \times 6 \times 3 \quad (3 \times 3 \times 3) \quad 4 \times 4 \times 2$

$S = 1$

$P = 0$

n_c' = no.of filters

dec 7: One layer of convolution Network:



one layer of convolution layer.

non convo

$$z^{[1]} = W^{[1]} a^{[0]} + b^{[1]}$$

$$a^{[1]} = g^{[1]}(z^{[1]})$$

$a^{[0]} \longrightarrow a^{[1]}$

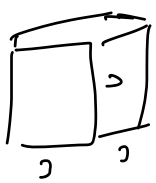
6x6x3 → 4x4x2

2 - no. of filters.

if we have 10 filters, then

$a^{[1]} = 4 \times 4 \times 10$.

Question) if you have 10 filters that are 3x3x3 in one layer of NN, how many parameters does the layer have?

each filter has  + bias = $(3 \times 3 \times 3 + 1) = 28$

∴ no. of parameter of NN = 280

which is independent to no. of input image, which can very large.

Summary of notation:-

If layer l is a convolution layer:

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

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

$f^{[l]}$ = filter size ($f \times f$)

$p^{[l]}$ = padding (valid or same)

$s^{[l]}$ = stride

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

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

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

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

Activation: $a^{[l]} \rightarrow \left(n_H^{[l]} \times n_W^{[l]} \times n_C^{[l]}\right)$

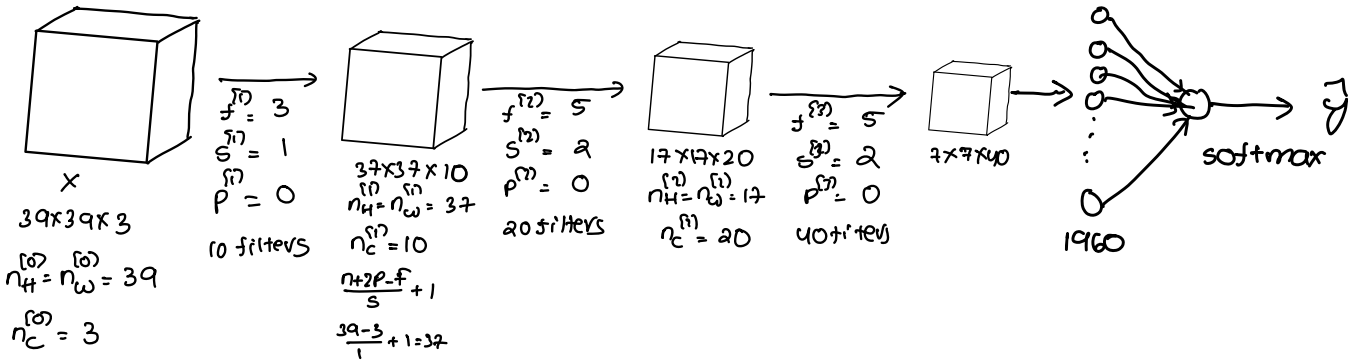
$A^{[l]} \rightarrow \left(m \times n_H^{[l]} \times n_W^{[l]} \times n_C^{[l]}\right)$

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

bias: $n_C^{[l]} - (1, 1, 1, n_C^{[l]})$

Lec 8: A simple convolution network:

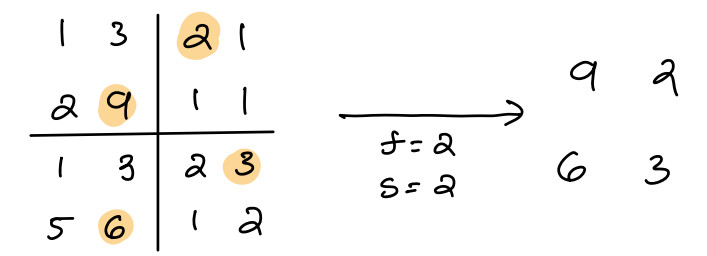
ConNet:



Types of layers in convnet:

- convolution (conv)
- pooling (pool)
- Fully connected (FC)

Lec 9: Pooling layer: Max Pooling:



No parameters to learn.

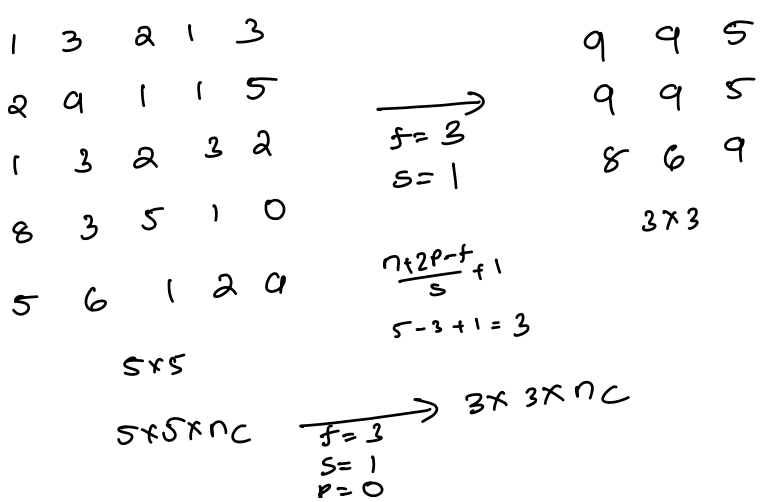
Hyperparameters:

f: filter size

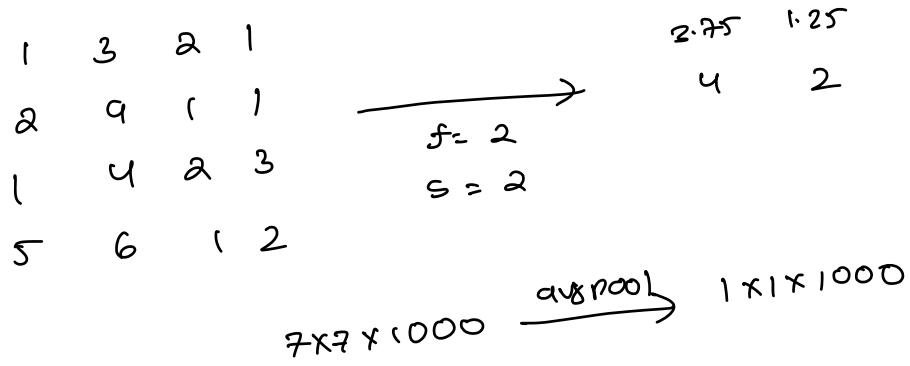
s: stride

p: padding

Max, avg pooling

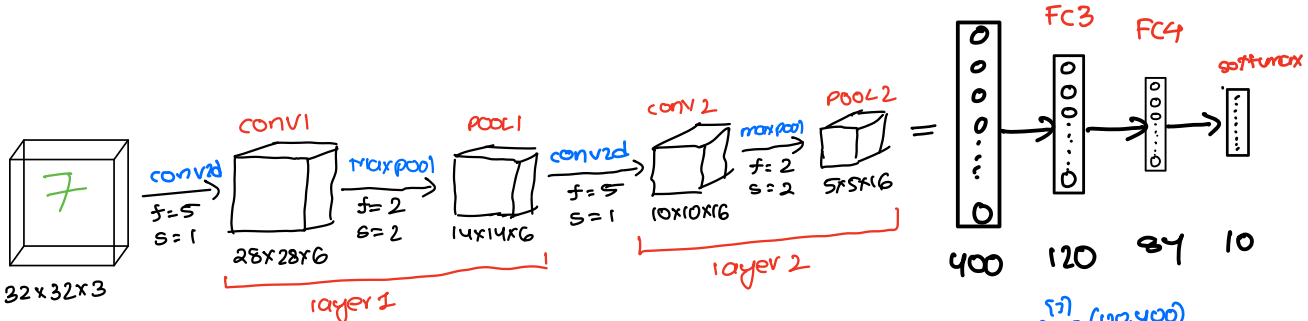


Average pooling:-



7x7x1000 → avg pool → 1x1x1000

Lec10: Neural Network Example: (LeNet-5)



$w^{[3]} = (120, 400)$

$w^{[4]} = (84, 120)$

conv1 filter = (5x5x3)

para = (5x5x3+1) = 76

6 filters = 456

conv2 filter = (5x5x6)

params = (5x5x6+1) = 151

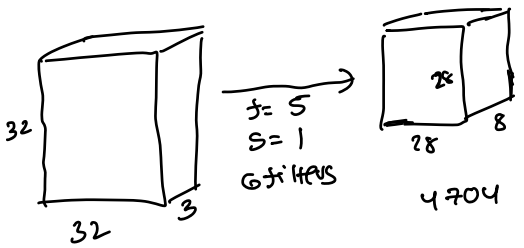
16 filters = 2416

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072 $a^{[0]}$	0
conv1 (f=5, s=1)	(28, 28, 6)	4704	456
POOL1 (f=2, s=2)	(14, 14, 6)	1176	0
conv2 (f=5, s=1)	(10, 10, 16)	1600	2416
POOL2 (f=2, s=2)	(5, 5, 16)	400	0
FC3	(120, 1)	120	(400x120+120) = 48,120
FC4	(84, 1)	84	(120x84+84) = 10164
Softmax	(10, 1)	10	(84x10+10) = 850

Lec11: Why convolution layers:

advantages:

- parameters sharing
- sparsity of connections



parameters = (5x5x3+1)x6 = 456 parameters.
of convl (row parameters).

if you use FC layer

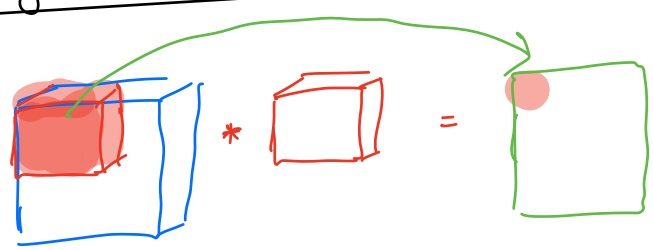
W = (4704, 3072)

parameters = 14M >> large.

parameter sharing :-

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

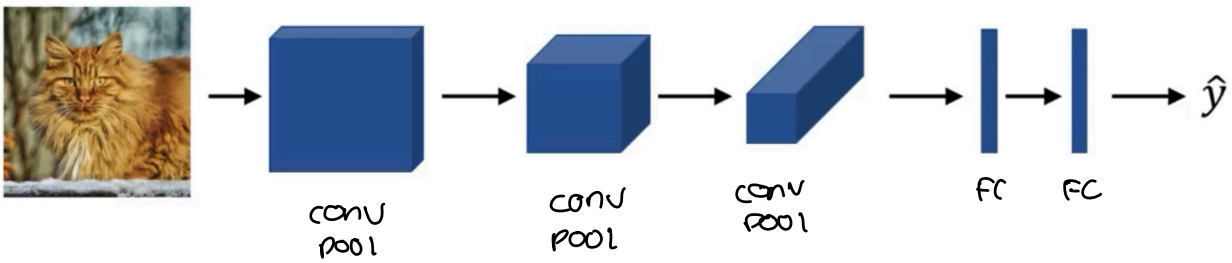
Sparsity of connection :



In each layer, each output value depends only on a small number of inputs. (less overfitting)

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

w, b



$x^{(i)}$ - image

$y^{(i)}$ - binary output or k classes

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

use gradient descent to optimize parameters to reduce J

② $w = [256, 16384) = 4194304$

$b = [256)$

$+$

4194560

⑦ $f = (7 \times 7 \times 3)$

$\text{para} = (7 \times 7 \times 3 + 1) \times 128 = 18944$

④ $63, 63, 16 \xrightarrow[s=2,\ p=0,\ 32 \text{ filters}]{f=7} \frac{n+2p-f}{s}+1 = \frac{63-7}{2}+1 = 29$

$\Rightarrow 29 \times 29 \times 32$

⑤ $61 \times 61 \times 32 \quad p=3$

$61+6 = (67 \times 67 \times 32)$

⑥ $(121, 121, 32) \xrightarrow[p=?,\ \#\text{filters}=32]{f=5,\ s=1} (121, 121, 32)$

$\frac{n+2p-f}{s}+1 = 121$

$\frac{121+2p-5}{1}+1 = 121$

$121+2p = 125$

$2p = 4$

$p = 2$