Convolutional Neural Network

W1-L1 → Done

W1-L2 :

An Example of Convolution

A) Vertical EDGE detector

$$\underbrace{\begin{bmatrix} 3 & 0 & 1 & 2 & 7 & 4 \\ 1 & 5 & 8 & 9 & 3 & 1 \\ 2 & 7 & 2 & 5 & 1 & 3 \\ 0 & 1 & 3 & 1 & 7 & 8 \\ 4 & 2 & 1 & 6 & 2 & 8 \\ 2 & 4 & 5 & 2 & 3 & 9 \end{bmatrix}}_{\substack{6\times 6 \\ \text{Grayscale image} \\ (n\times n)}} * \underbrace{\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}}_{\substack{3\times 3 \text{ filter} \\ \text{or} \\ 3\times 3 \text{ Kernel} \\ (f\times f)}} = \underbrace{\begin{bmatrix} -5 & -4 & 0 & 8 \\ -10 & -2 & 2 & 3 \\ 0 & -2 & -4 & -7 \\ -3 & -2 & -3 & -16 \end{bmatrix}}_{\substack{4\times 4 \\ (n-f+1)\times(n-f+1)}}$$

(vertical edge detector kernel)

In 4x4: $-16(4,4) = 1\times1 + 0\times7 + (-1\times8) + 1\times6 + 0\times2 + (-1\times8) + 2\times1 + 0\times3 + (-1\times9)$

$= (1-8) + (6-8) + (2-9)$

$= -16$

B) Intuitive Example of why $\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$ is vertical edge detector.

bright pixel dark pixel

W1-L3

PADDING

* Solves shrinking problem of convolution: $[n\times n][f\times f] = [(n-f+1)\times(n-f+1)]$

PROBLEMS Introduced by Convolution:

1) Shrinks the image (See above)
2) It throws away information from edges (edge pixels are touched only one time by kernel)

Pad by 1 pixel:

$p=1$

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & & & & & & & 0 \\ 0 & & 6\times6 & & & & & 0 \\ & & n\times n & & & & & \\ 0 & & & & & & & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\substack{8\times 8 \\ (n+2p)\times(n+2p)}} * \underbrace{\boxed{3\times 3}}_{f\times f} \longrightarrow \underbrace{\boxed{6\times 6}}_{\substack{(n-2p-f+1)\times(n+2p-f+1) \\ \text{Image size preserved}}}$$

VALID Convolution (No padding):

$$n \times n \;*\; f \times f \longrightarrow (n-f+1) \times (n-f+1)$$

Same Convolution:

Pad so that output size is equal to input size.

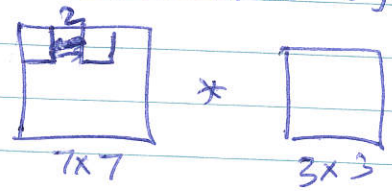
$$(n+2p-f+1) \times (n+2p-f+1)$$

for same convolution: $n+2p-f+1 = n \Rightarrow \boxed{p = \frac{f-1}{2}}$ i.e pad by $\left(\frac{f-1}{2}\right)$ pixels to get same convolution

In CV, f is usually odd

Strided Convolution:

Stride = 2 (jump the kernel "2" steps)



$= 3 \times 3$

$$\Rightarrow \underset{\text{padding "p" \& stride = s}}{(n \times n) * (f \times f)} = \left\lfloor \left(\frac{n+2p-f}{s} + 1\right) \right\rfloor \times \left\lfloor \left(\frac{n+2p-f}{s} + 1\right) \right\rfloor$$

$$\lfloor z \rfloor = \text{floor}(z)$$

- Convolution here is called cross-correlation in mathematics (W1-L5). In maths, convolution involves ~~flipping~~ & mirroring of kernel which gives this property

$$(A * B) * C = A * (B * C)$$ ~~Assocd~~ (Associative)

(Above property is useful in Signal Processing)

(W1-L6) *

Convolution on RGB

Formula: $(n \times n \times n_c) * (f \times f \times n_c) = \frac{n-f+1}{4} \times \frac{n-f+1}{4} \times n_c'$

$n \times n \rightarrow$ image

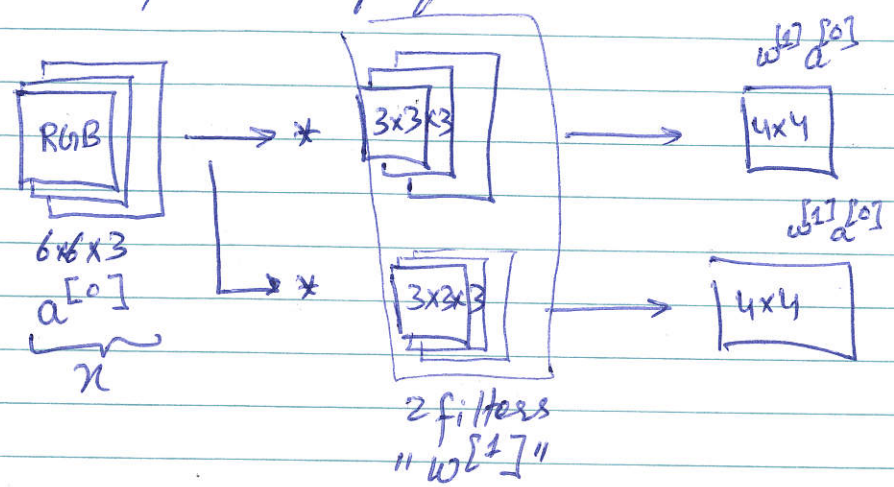
$n_c \rightarrow$ # of channel in image

$n_c' \rightarrow$ # of filters

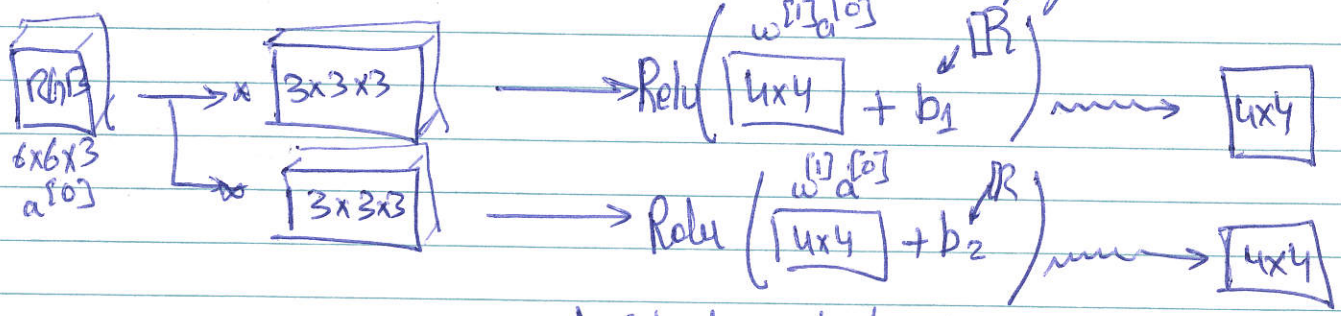
$f \times f$ = filter dim

W1-L7 ONE LAYER of CNN

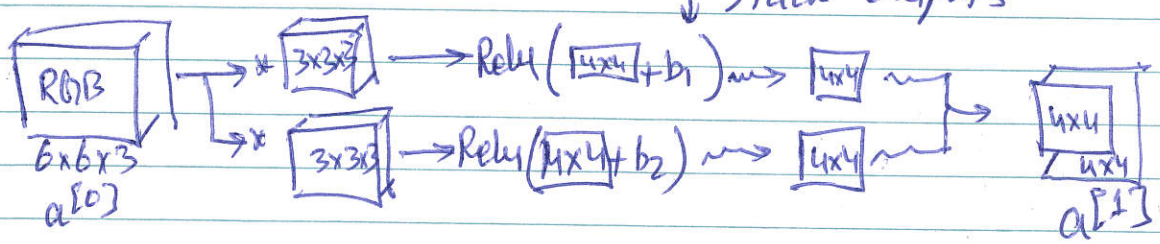
ONE feed-forward Step of CNN :



↓ Add bias & then apply Relu



↓ Stack outputs



=> one FF step of CNN:

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

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

here $a^{[0]} = x$ (input image)

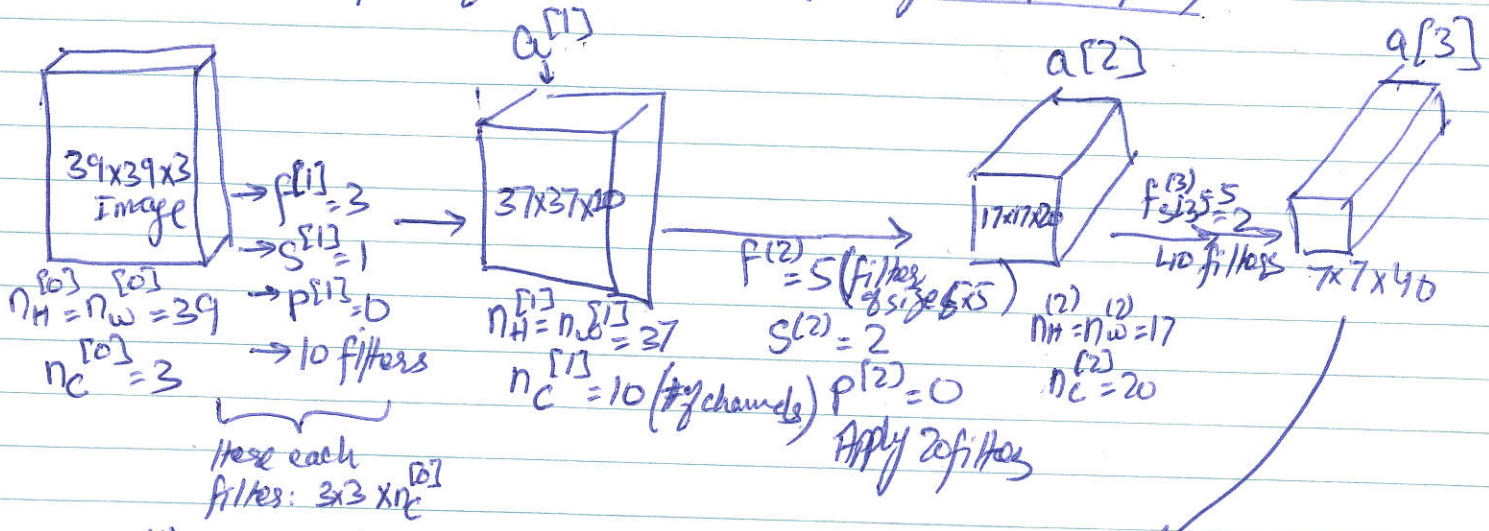
$w^{[1]}$ = 2 filters of each size (3x3x3)

& we will learn 27+27 total weights as we have 2 filters ~~of~~ here.

* See Summary of Notations at last ~~side~~ slide of W1-L7.

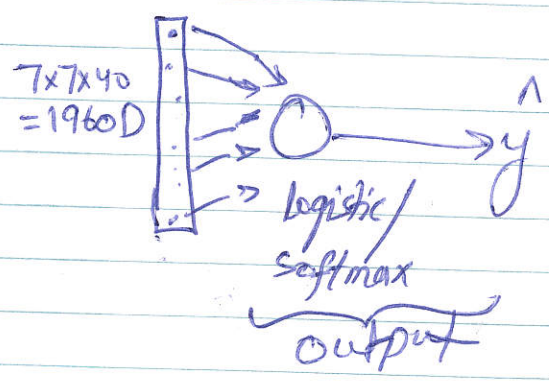
W1-L8

Example of Conv NET (3 layers & output)



$n_H^{(1)}: \frac{n+2p-f}{s} = \frac{39+0-3}{1}+1 = 37$

Roll it into 1960D vector



Types of LAYERS

1) Convolutional LAYER (CONV)
2) Pooling (POOL)
3) Fully Connected (FC)

W1-L9

Pooling Layers

1) MAX Pooling (used more often)
2) Avg Pooling

Summary:

Hyperparameters:

f: filter size
s: stride
Max or Avg. pooling

* Gradient descent does not learn hyperparameters of pooling