CNN course

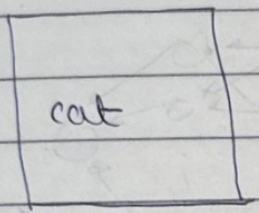
Date
Page

WEEK 1:-

COMPUTER VISION:-

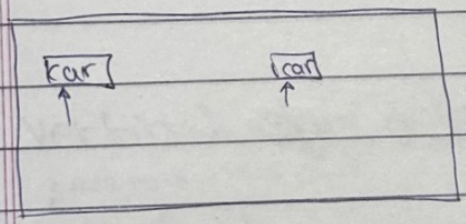
CV problems

1] Image classification



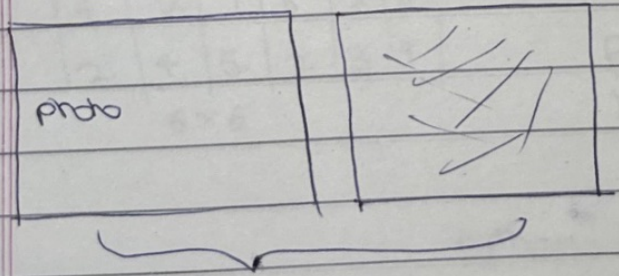
→ cat (0/1)

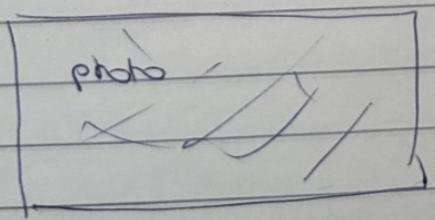
2] Object detection



position of car?

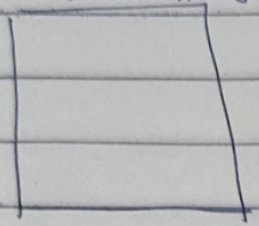
3] Nueral style transfer





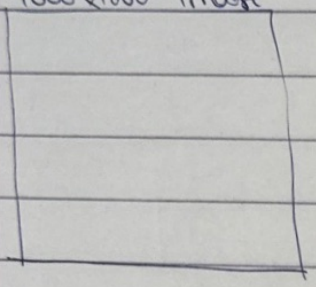
Date
Page

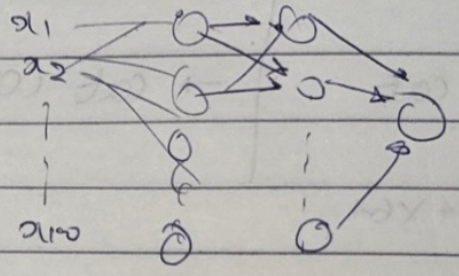
64 x 64 image



64 x 64 x 3 = 12288

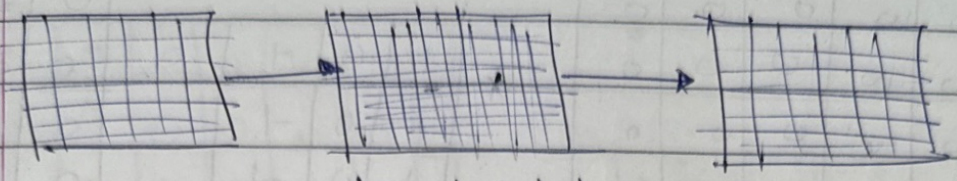
1000 x 1000 image



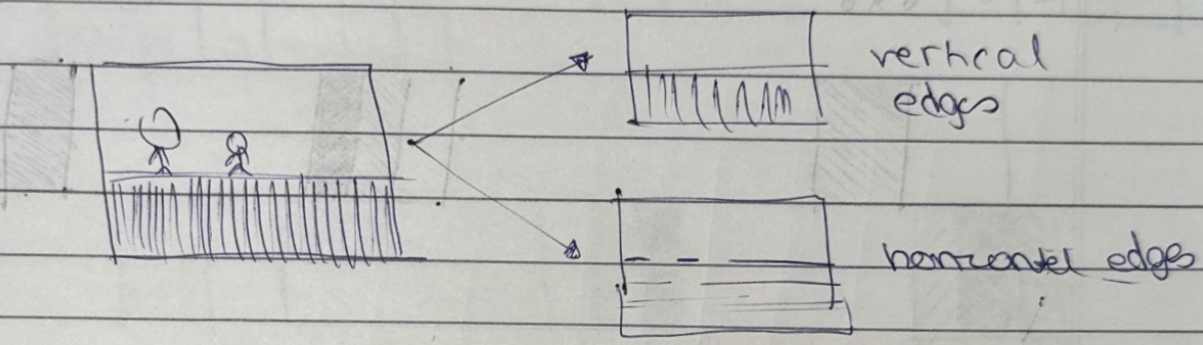


1000 x 1000 x 3 = 3 million

EDGE-DETECTION EXAMPLE:-



random object detection complete face detection



⇒ Vertical edge detection:-

3+1+2+0+0+0 -1-8-2 = -5

3 (1)	0 (0)	1 (-1)	2	7	4
1 (1)	5 (0)	8 (-1)	9	3	1
2 (1)	7 (0)	2 (-1)	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6×6

"convolutional operator"

*

1	0	-1
1	0	-1
1	0	-1

3×3

filter kernel

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

4×4

Python: conv-forward

tensorflow: tf.nn.conv2d

Date
Page

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

6×6

*

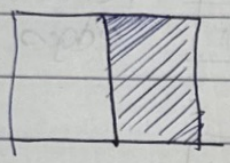
1	0	-1
1	0	-1
1	0	-1

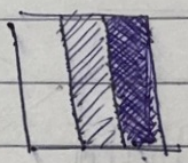
3×3

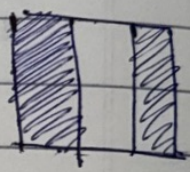
=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

4×4





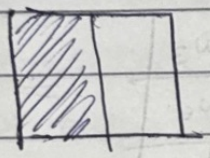


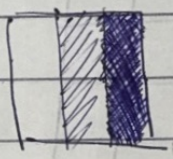
MORE EDGE DETECTION:-

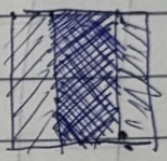
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

1	0	-1
1	0	-1
1	0	-1

= | 0 | -30 | -30 | 0 |
|---|---|---|---|
| 0 | -30 | -30 | 0 |
| 0 | -30 | -30 | 0 |
| 0 | -30 | -30 | 0 |







1	0	-1
1	0	-1
1	0	-1

Vertical

1	1	1
0	0	0
-1	-1	-1

Horizontal

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

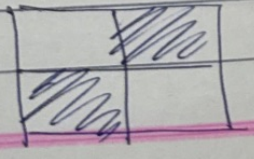
6×6

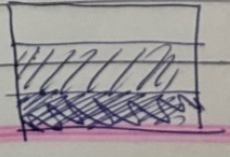
1	1	1
0	0	0
-1	-1	-1

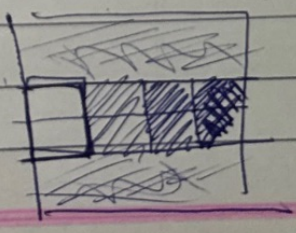
3×3

= | 0 | 0 | 0 | 0 |
|---|---|---|---|
| 30 | 10 | -10 | -30 |
| 30 | 10 | -10 | -30 |
| 0 | 0 | 0 | 0 |

4×4







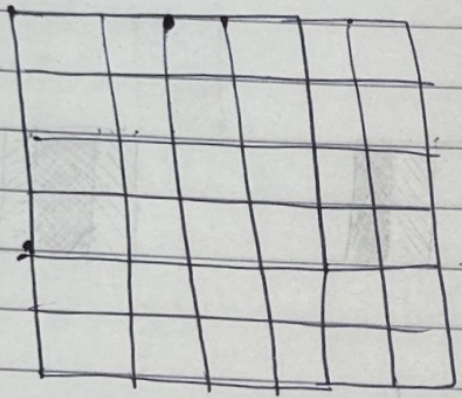
LEARNING TO DETECT EDGES

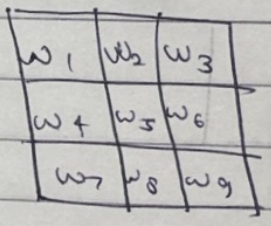
1	0	-1
1	0	-1
1	0	-1

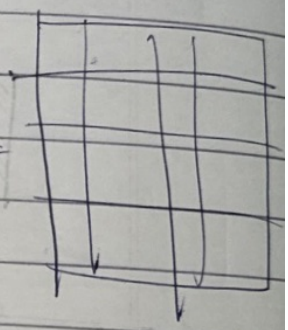
1	0	1
2	0	-2
1	0	-1

Sobel filter

3	0	-3
10	0	-10
-3	0	-3

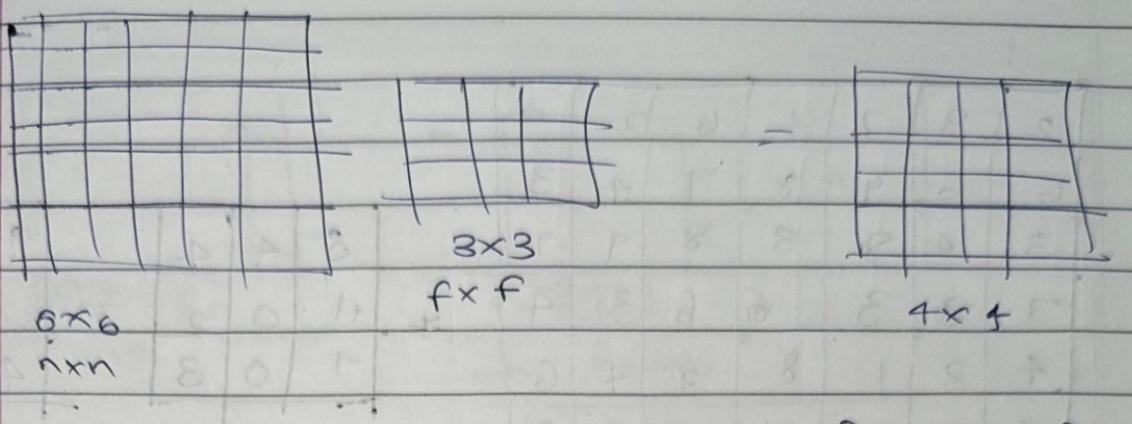






letting numbers be parameters

PADDING:-



two downsides

- image shrinks
- corner edges are used in only one output

So you padd the 6×6 image
it becomes 8×8 and after convolving it with 3×3
we get 6×6 image
p = padding amount
we pad it with 0

Valid and Same convolutions

"Valid" = $n \times n * f \times f \rightarrow n-f+1 * n-f+1$

$6 \times 6 * 3 \times 3 \rightarrow 4 \times 4$

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

STRIDED CONVOLUTIONS:-

2	3	7	4	6	2	9
6	6	9	8	7	4	3
3	4	8	3	8	9	7
7	8	3	6	6	3	4
4	2	1	8	3	4	6
3	2	4	1	9	8	3
0	1	3	9	2	1	4

7x7

*

3	4	4
1	0	2
-1	0	3

3x3

stride = 2

=

91	100	83
69	91	127
44	72	74

$n \times n$ * $f \times f$

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

Technical note on cross-correlation vs. convolution

Convolution in math textbook

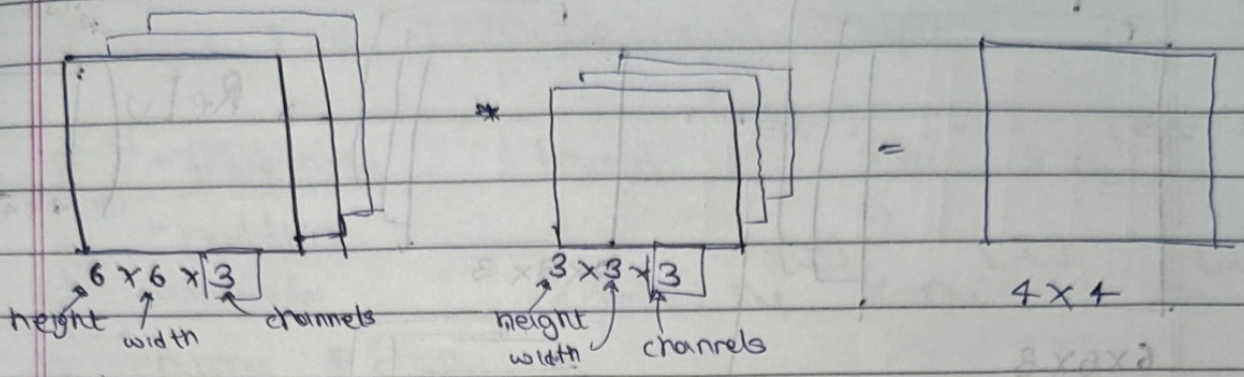
*

3	4	5
1	0	2
-1	9	7

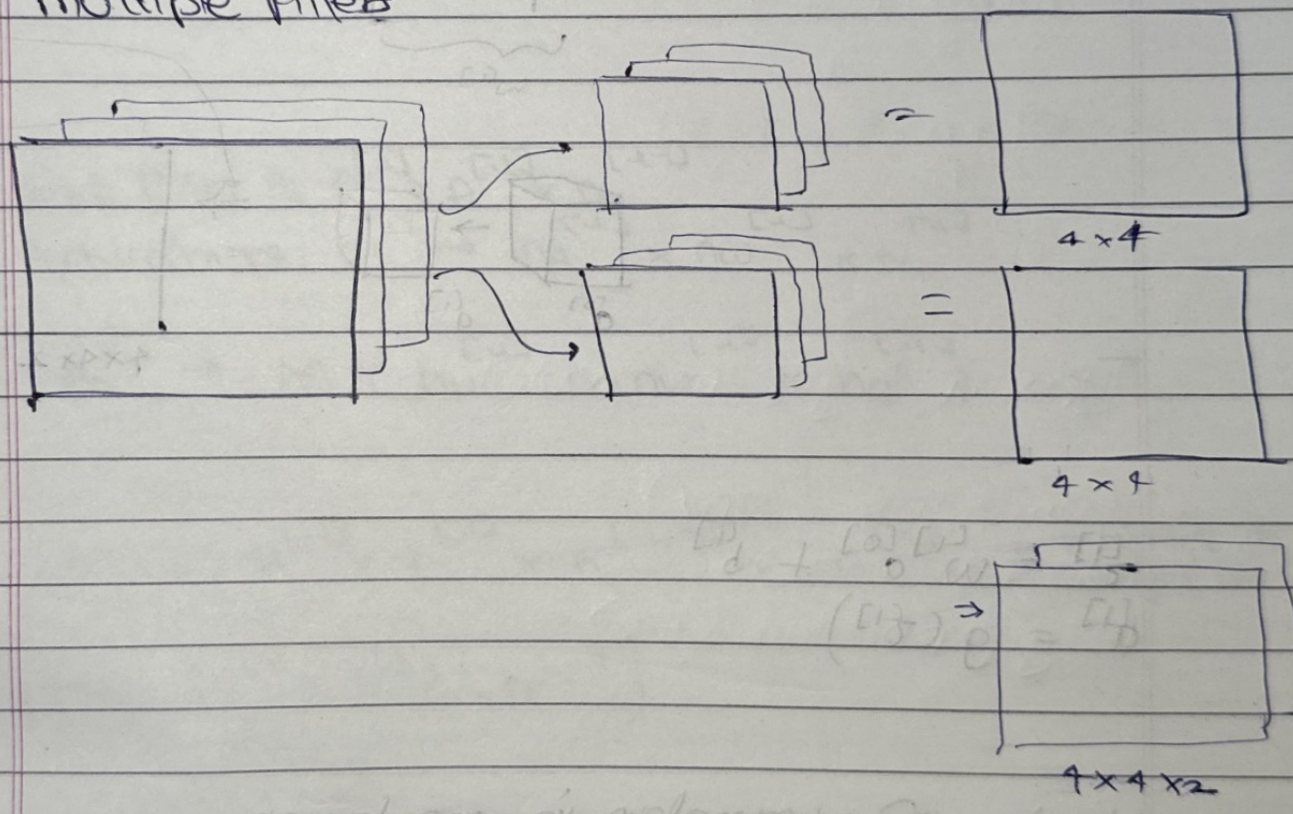
7	9	-1
2	0	1
5	4	3

$(A * B) * C = A * (B * C)$

CONVOLUTIONS OVER VOLUME



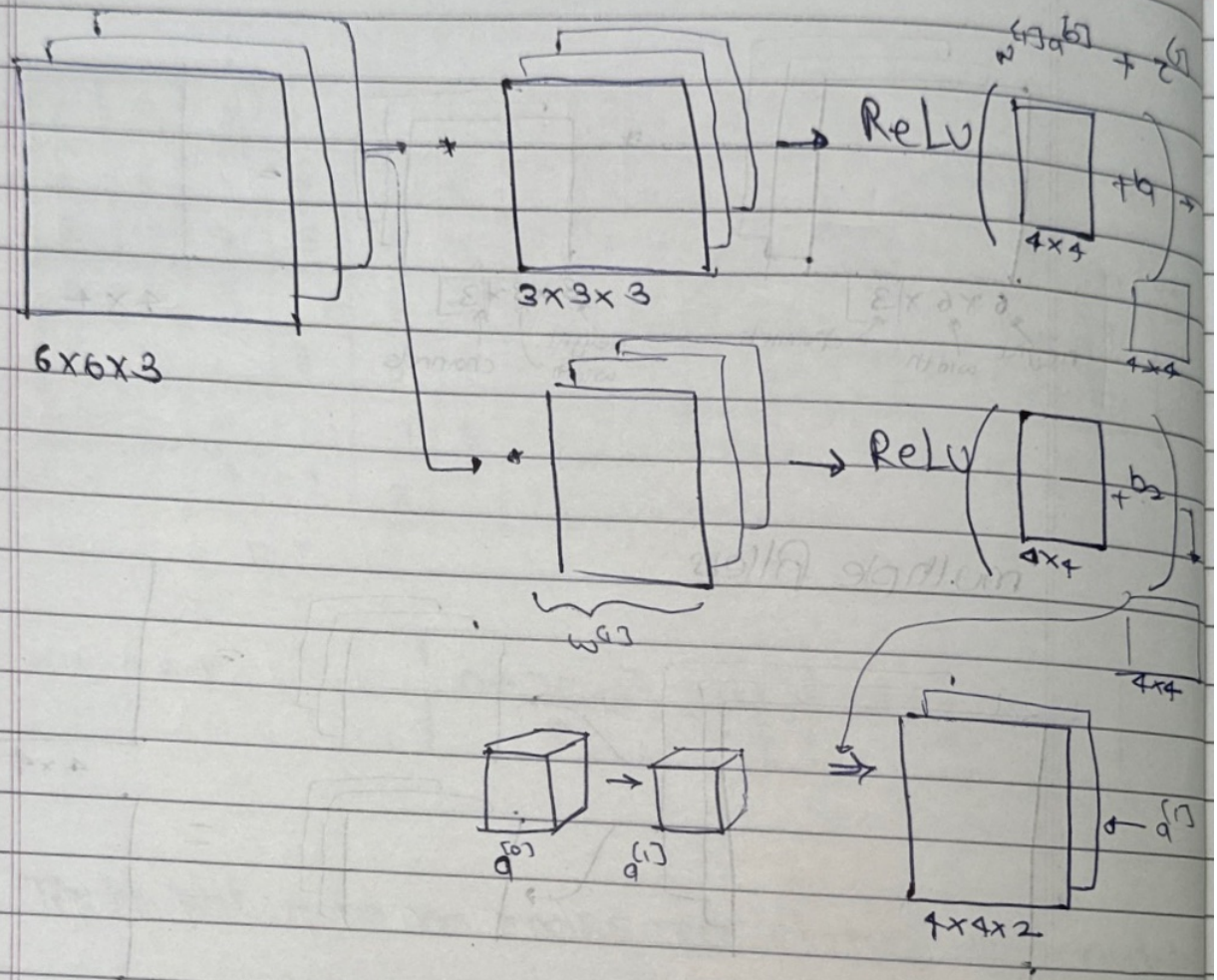
multiple filters



Summary

$n \times n \times n_c \quad * \quad f \times f \times n_c \Rightarrow n-f+1 \times n-f+1 \times n_c'$

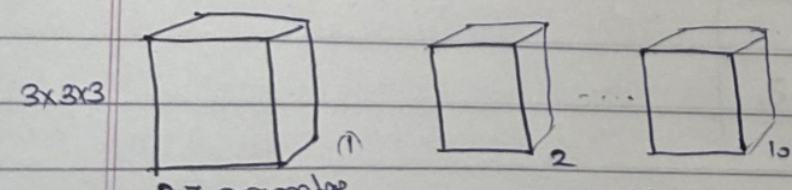
ONE LAYER OF CONVOLUTIONAL NN :



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

Number of parameters in one layer

if you have 10 filters that are 3x3x3 in one layer of a neural network, how many parameters does that layer have?



27 parameters + bias = 28 parameters

280 parameters

Date
Page

Summary of notation

If layer l is a convolutional layer:

$f^{[l]}$ = filter size Input: $n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$

$p^{[l]}$ = padding

$s^{[l]}$ = stride Output: $n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

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

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

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

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

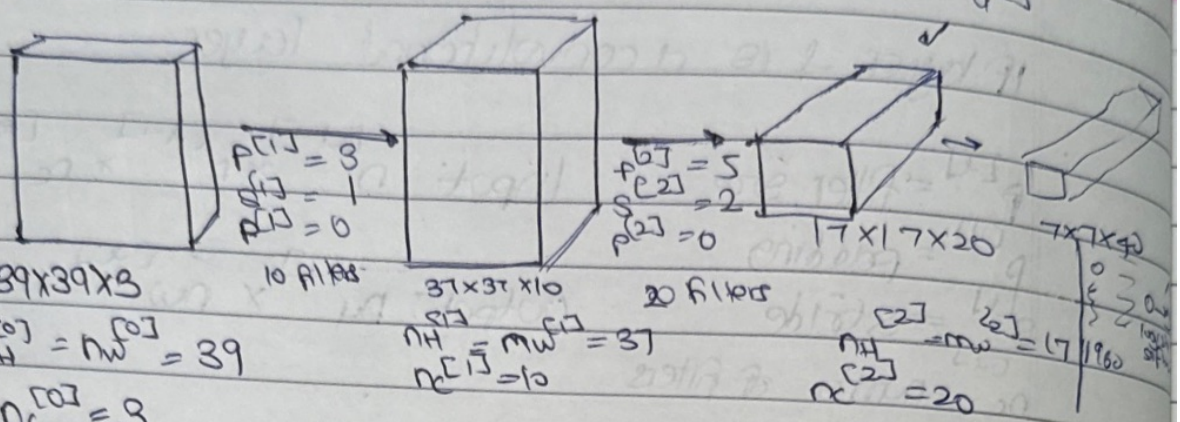
$A^{[l]} \rightarrow m \times n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

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

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

SIMPLE CONVOLUTIONAL NETWORK EXAMPLE:

ConvNet



$n_H^{[0]} = n_W^{[0]} = 39$

$n_C^{[0]} = 3$

$n_H^{[1]} = n_W^{[1]} = 37$

$n_C^{[1]} = 10$

$n_H^{[2]} = n_W^{[2]} = 17$

$n_C^{[2]} = 20$

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

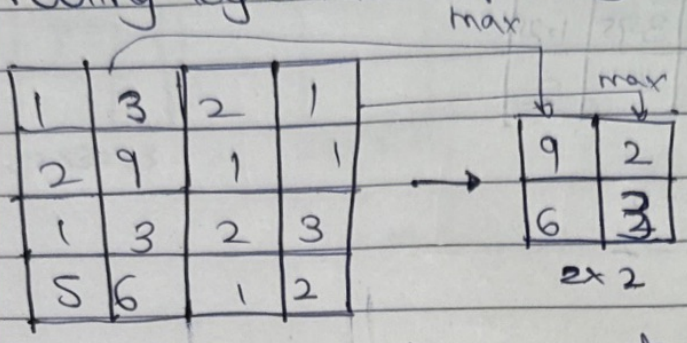
$$\frac{39+0-3}{1} + 1 = 37$$

Types of layer in convolutional network:

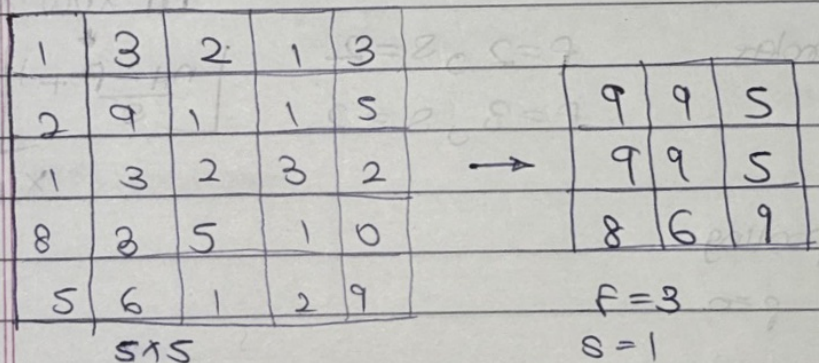
- Convolution (CONV)
- Pooling (POOL)
- Fully connected (FC)

POOLING LAYERS

Pooling layer: Max pooling



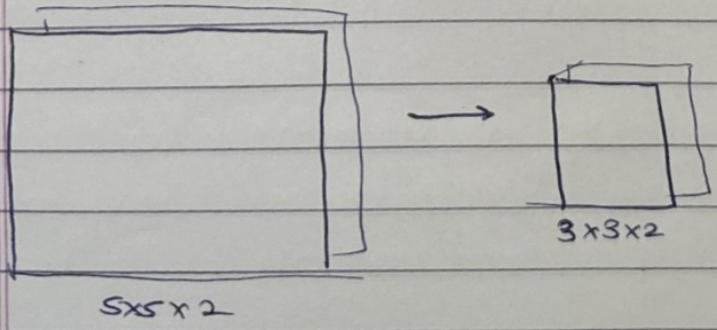
Hyperparameters
$f = 2$
$s = 2$



$f = 3$
$s = 1$

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

multiple layer



Date
Page

Average pooling

1	3	2	1
2	9	1	1
1	4	2	3
5	6	1	2

→

3.75	1.25
4	2

f=2
s=2

7x7x1000 → 1x1x1000

max pooling uses more NN than avg pooling

Hyperparameters
f: filter size
s: stride
Max or avg pooling
~~p: padding~~ p=0

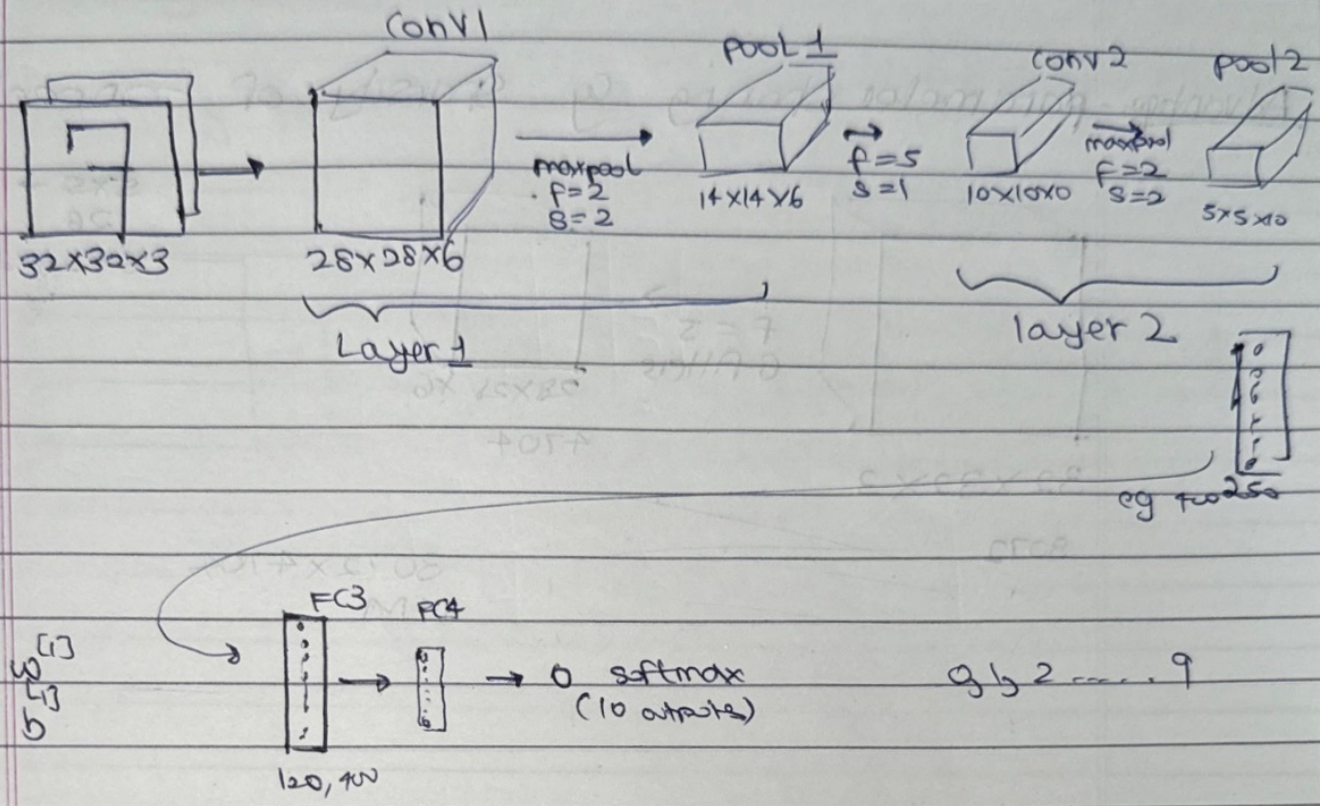
f=2, s=2
f=3, s=2

$$\left\lfloor \frac{n_H - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_W - f}{s} + 1 \right\rfloor \times n_C$$

($n_H \times n_W \times n_C$)

no parameters to learn!

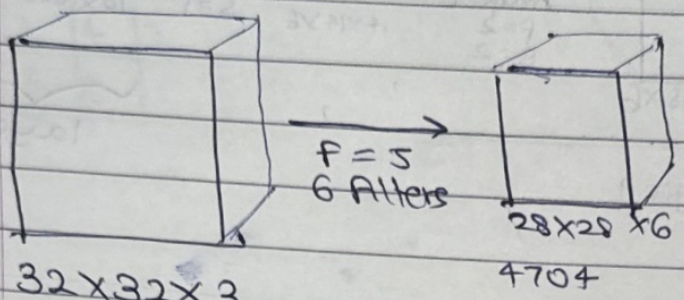
CNN EXAMPLE (LeNet-5)



NN example

WHY CONVOLUTIONS?

Advantage:- parameter sharing & sparsity of connections



5x5 → 25
26
6x26 → 156 parameters

4704

3072

3072x4704
14M

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

*

1	0	-1
1	0	-1
1	0	-1

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

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 image

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

Date .....
Page .....

Putting it together

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