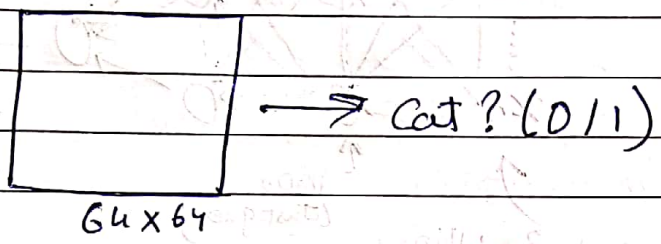
Convolutional Neural Networks.

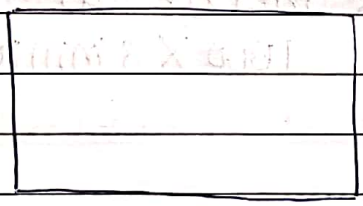
* Computer Vision.

Some Example of Computer Vision

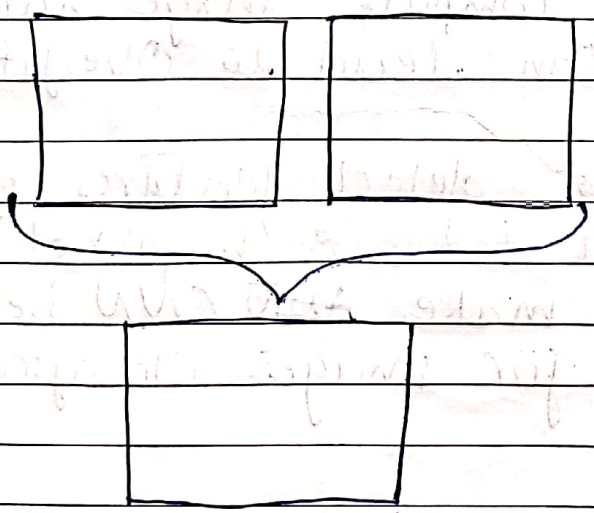
⇒ image classification.



⇒ object detection.

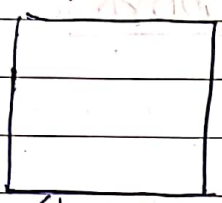


⇒ Neural Style Transfer



★ Deep learning on large images

Small image

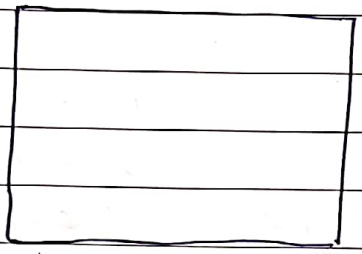


→ cat? (0/1)

64 × 64 × 3

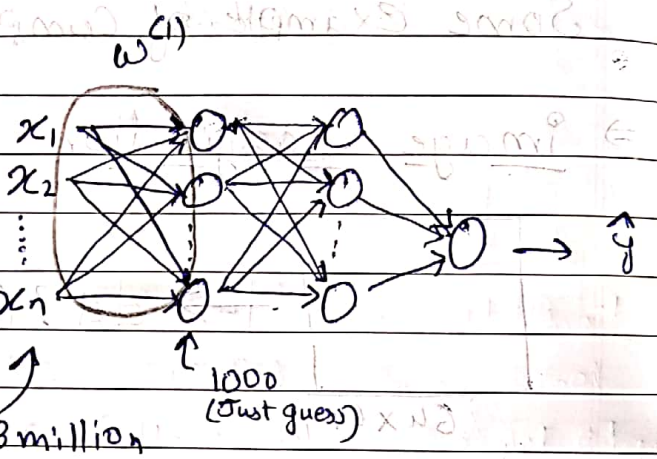
= 12288

Large image



1000 × 1000 × 3

= 3 million



Then Total Number of weight ~~on metx~~ metrix $w^{(1)}$ will be 1000 × 3 million

$x \in \mathbb{R}^{3m}$

∴ if we use Artificial Neural Network for solving image classification problem, model has have to compute large number of parameter will can lead to Overfitting

→ Also, ANN can not detect features of images, instead it takes each pixel as features, which make ~~ANN~~ CNN better than ANN ~~in~~ for images classification problems.

* Edge detection Examples.

→ Vertical edge detection

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

6x6 (grey image)

"Convolution"

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

3x3 filter (kernal)

=

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

4x4

4x4 metrix Solution:

→ (3 x1) + (1x1) + (2x1) + (0x0) + (5x0) + (7x0) + (1x-1) + (8x-1) + (2x-1) = -5

→ (0x1) + (5x1) + (7x1) + (1x0) + (8x0) + (2x0) + (2x-1) + (9x-1) + (5x-1) = -4

⋮

→ (1x1) + (6x1) + (2x1) + (7x0) + (2x0) + (3x0) + (8x-1) + (8x-1) + (9x-1) = -16

→ Horizontal Edge Detection

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

*

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

=

0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0

* Learning to detect edge.

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

Prewitt
Top edge filter

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

Prewitt
Left edge filter

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

Prewitt
bottom edge filter

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

Prewitt
Right edge filter

-1	-2	-1
0	0	0
1	2	1

Sobel
Top edge filter

-1	0	1
-2	0	2
-1	0	1

Sobel
Left edge filter

1	2	1
0	0	0
-1	-2	-1

Sobel
bottom edge filter

1	0	-1
2	0	-2
1	0	-1

Sobel
Right edge filter

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

Schass
Top edge filter

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

Schass
Left edge filter

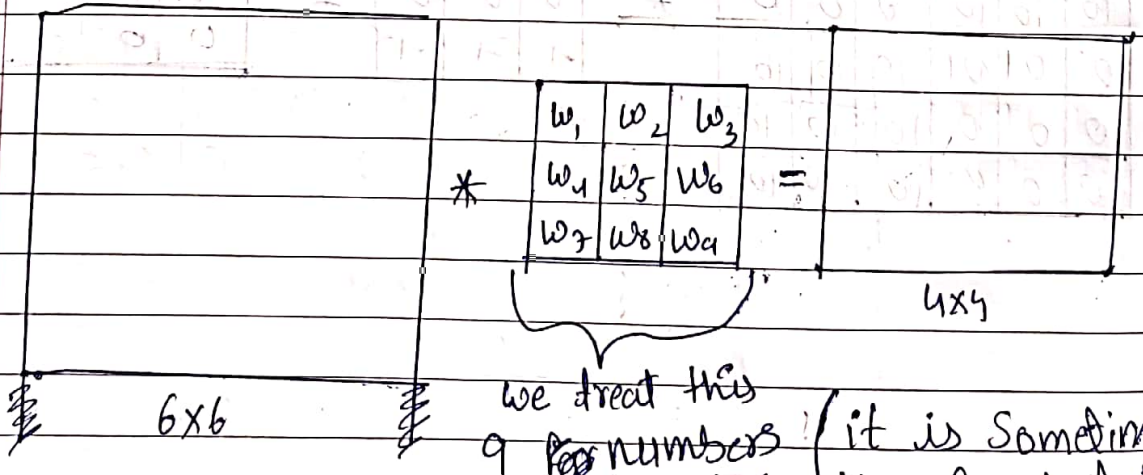
3	10	3
0	0	0
-3	-10	-3

Schass
bottom edge filter

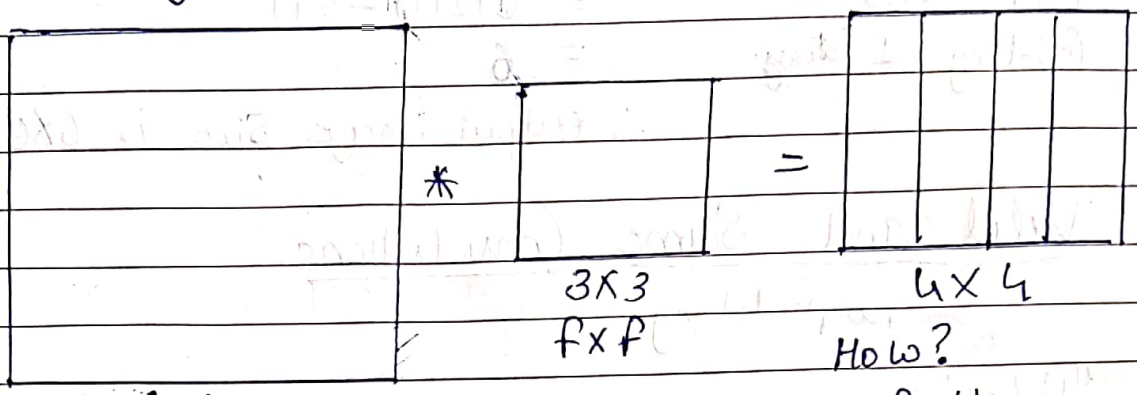
3	0	-3
10	0	-10
3	0	-3

Schass
Right edge filter

*



* Padding



How?

$n - f + 1$

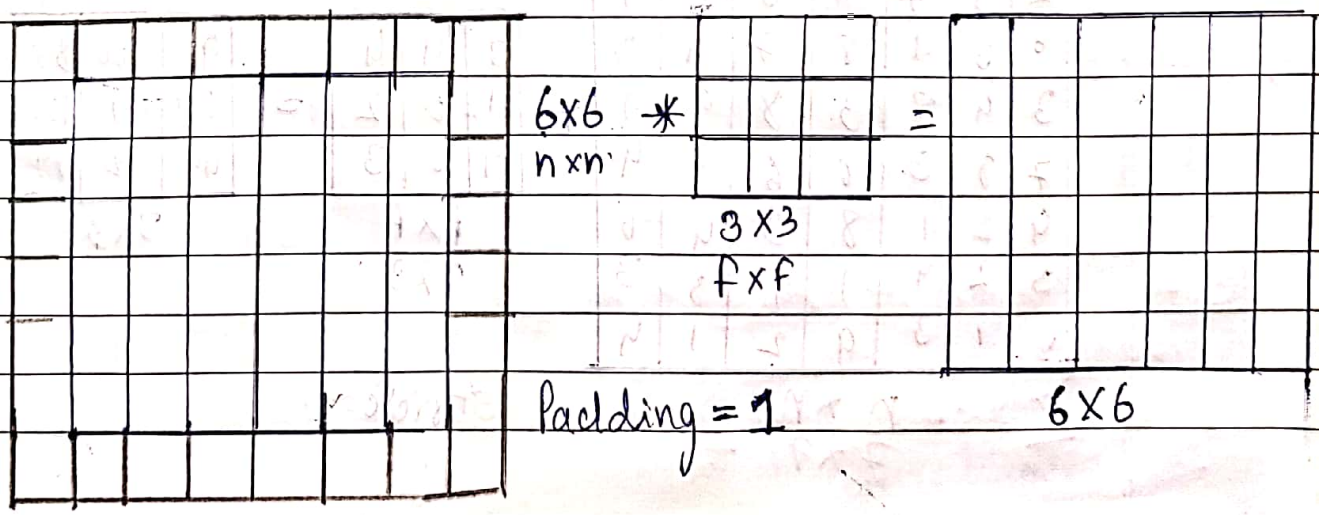
$= 6 - 3 + 1$

$= 4$

⇒ two down sides of this

1) Everytime we apply Convolution operator, image shrink, ~~we to~~ So we've gone from 6X6 to 4X4 then 2X2, which ~~as~~ we do not want every time, aspecially when image size is small.
2) Pixels on the corners around ~~the~~ edges are used much less in output, throwing away lots of information near the edge of the image.

* Two Solve both of this problems, ~~the~~ use padding → ~~add~~ addition border to the image.



$n \times n = 6 \times 6$
$f \times f = 3 \times 3$
Padding = 1

$\therefore \quad n+2p-f+1$
$= 6+2(1)-3+1$
$= 6$

$\therefore$ output image size is 6×6

✱ Valid and Same Convolutions.

no padding

"Valid":

$$n \times n \;*\; f \times f \longrightarrow n-f+1 \times n-f+1$$
$$6 \times 6 \quad 3 \times 3 \quad 4 \times 4$$

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

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

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

$\Rightarrow$ filter is usually odd number matrix ; $3 \times 3, 5 \times 5, 7 \times 7$.

✱ 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

$n \times n$
7×7

$*$

3	4	4
1	0	2
-1	0	3

$f \times f$
3×3

$=$

91	100	83
69	91	127
44	72	74

3×3

Stride = 2

Padding → p
Stride → s

$$= \left[\frac{n+2p-f}{s}+1\right] \times \left[\frac{n+2p-f}{s}+1\right]$$

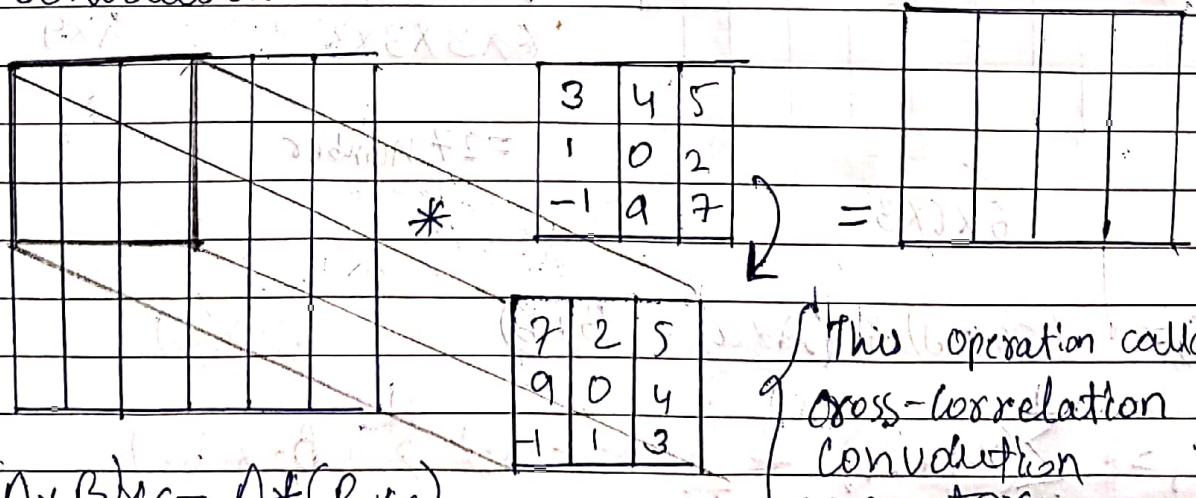
$$= \left[\frac{7+2(0)-3}{2}+1\right] \times \left[\frac{7+2(0)-3}{2}+1\right] = 3 \times 3$$

→ Stride is the number of pixels ~~no~~ shifts over the input matrix when the stride is 1 then we move the filter to 1 pixel at a time. when the stride is 2 then we move the filter to 2 pixel at a time and so on.

→ if $\left[\frac{n+2p-f}{s}+1\right]$ part is not integer about fraction, then, ~~some~~ round-off value will be counted

i.e $\lfloor z \rfloor$ = floor(z) ; here $z = \frac{n+2p-f}{s}+1$

☆ Technical note on Cross-correlation Vs. convolution.

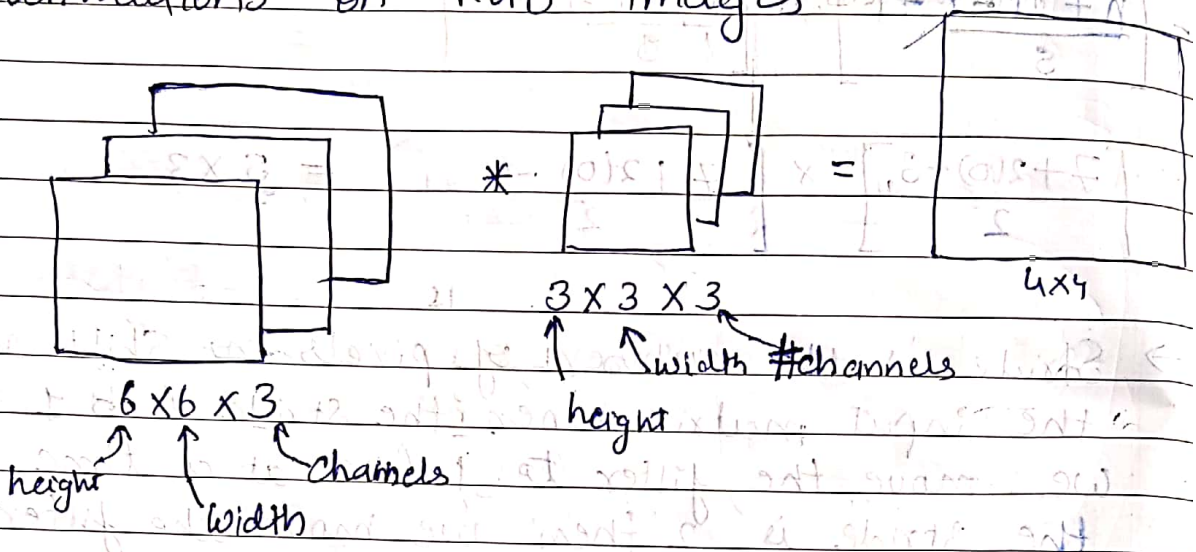


This operation called cross-correlation or convolution operator.

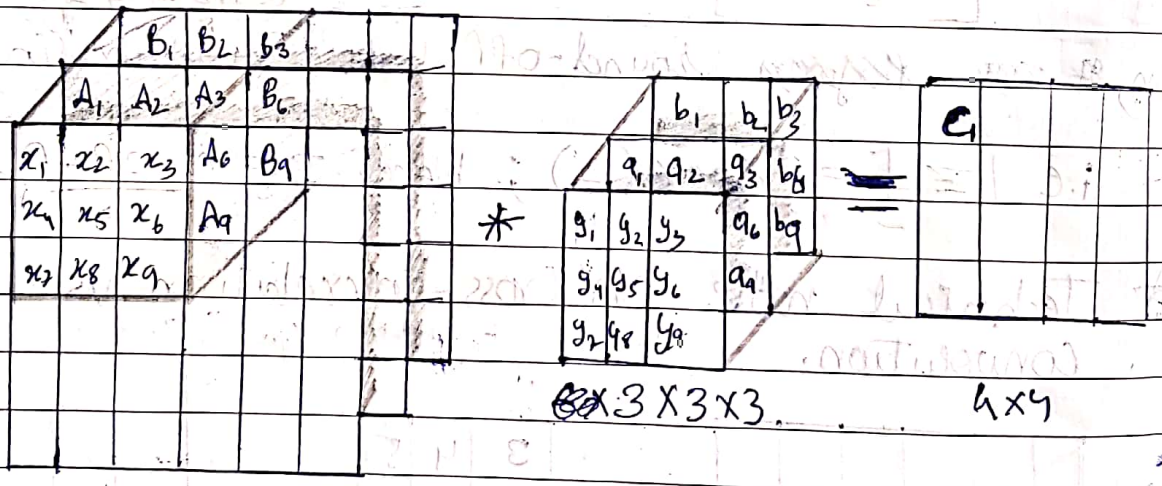
Ag: $(A * B) * C = A * (B * c)$

* Convolutions over volumes.

✱ Convolutions on RGB images



⇒ How it works?

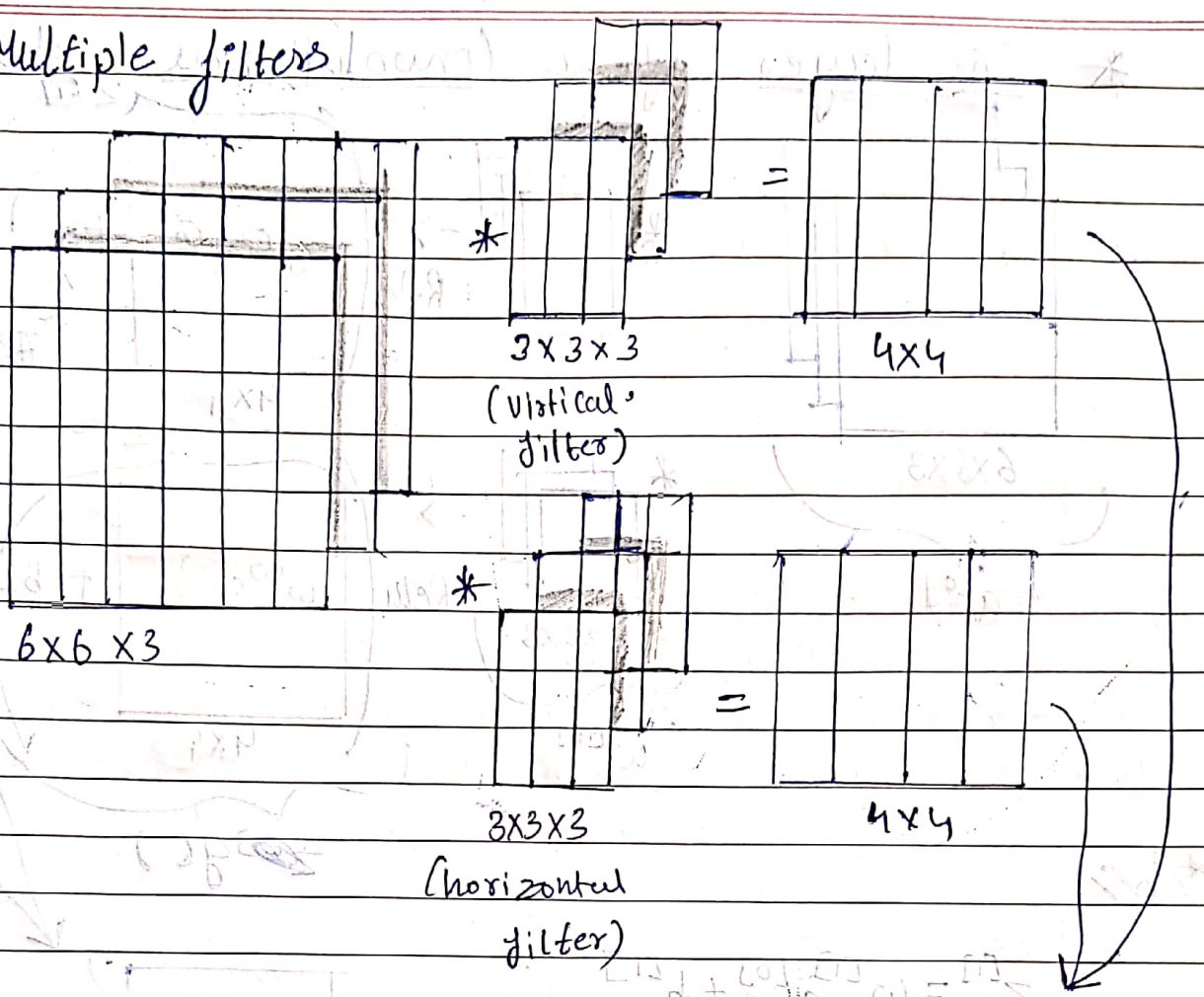


=27 number

~~(X*Y)+(A*a)+(B*b)~~

$$C_1 = \sum_{i=1}^{9}(x_i * y_i) + \sum_{i=1}^{9}(A_i * a_i) \sum_{i=1}^{9}(B_i * b_i)$$

⇒ Multiple filters



Summary:

$n \times n \times n_c * f \times f \times f_c$

$6 \times 6 \times 3 \quad 3 \times 3 \times 3$

have to be same

$\rightarrow n-f+1 \times n-f+1 \times n_f$

$4 \times 4 \times 2$

n_f → # filters

Example of a layer

* One layer of a Convolutional Network.

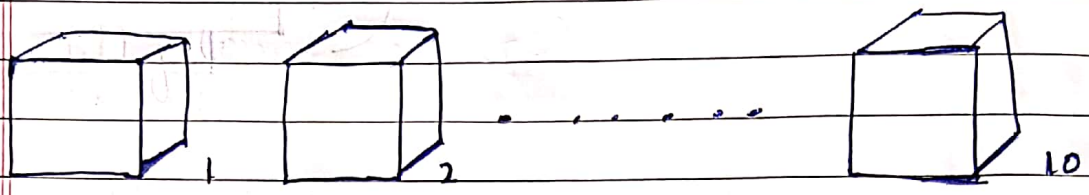
$$Z^{[1]} = w^{[1]} a^{[0]} + b^{[1]}$$

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

** b_1 and b_2 are the bias. b_1 will be added to each pixel of first (4x4) image, and b_2 will be added to each pixel of second (4x4) image.

✱ Number of parameter in one layer.

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



3x3x3 = 27 param + bias (b_1) = 28 parameters

3x3x3 = 27 param + bias (b_2) = 28 param

3x3x3 = 27 Param + bias (b_{10}) = 28 Param

Then, Total parameter in one layer will be,

$10 \times 28 = 280$

✱ Summary of Notation

if layer l is a convolution layer :

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

$p^{[l]}$ = Padding.

$s^{[l]}$ = Stride

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

Output : $n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$

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

Activation : $a^{[l]} \rightarrow n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$

Weights : $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$ ($n_c^{[l]}$ ↑ # filters in l layer)

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

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

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

A Simple Convolution network Example.

* Example ConvNet

✡ Type of layer in a Convolutional network:

- Convolution (Conv)
- Pooling (Pool)
- → Fully Connected (FC)

✳ Pooling Layer

⇒ Max pooling: Calculate largest value in each patch of feature map. filter size (2x2)

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

4x4

→

9	2
6	3

2x2

→ hyperparameters:

$f = 2$

$S = 2$

→ No parameters!

filter : 3×3

1	3	2	1	3
2	9	1	1	5
1	3	2	3	2
8	3	5	1	0
5	6	1	2	9

$5 \times 5 \times n_c$

$n = 5$
$f = 3$
$s = 1$

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

$= 3$

9	9	5
1	9	5
8	6	9

$3 \times 3 \times n_c$

⇒ Average pooling : Calculate Average value in each patch of feature map.

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

$f = 2$
$s = 2$

3.75	1.25
4	2

Summary of pooling

Hyperparameter:

f : filter size
s : stride
max or average pooling.
p : padding (Very Very rare)

No parameter to learn

$n_H \times n_w \times n_c$

$\frac{n_H - f + 1}{s} \times \frac{n_H - f + 1}{s} \times n_c$

Convolutional neural network Example

	Activation Shape	Activation Size	# Parameters
Input :	(32,32,3)	3,072	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
FC-3	(120,1)	120	48120
fC-4	(84,1)	84	10164
Softmax	(10,1)	10	924

Rule: (f * f * # Previous channels + bias(1)) * # filters

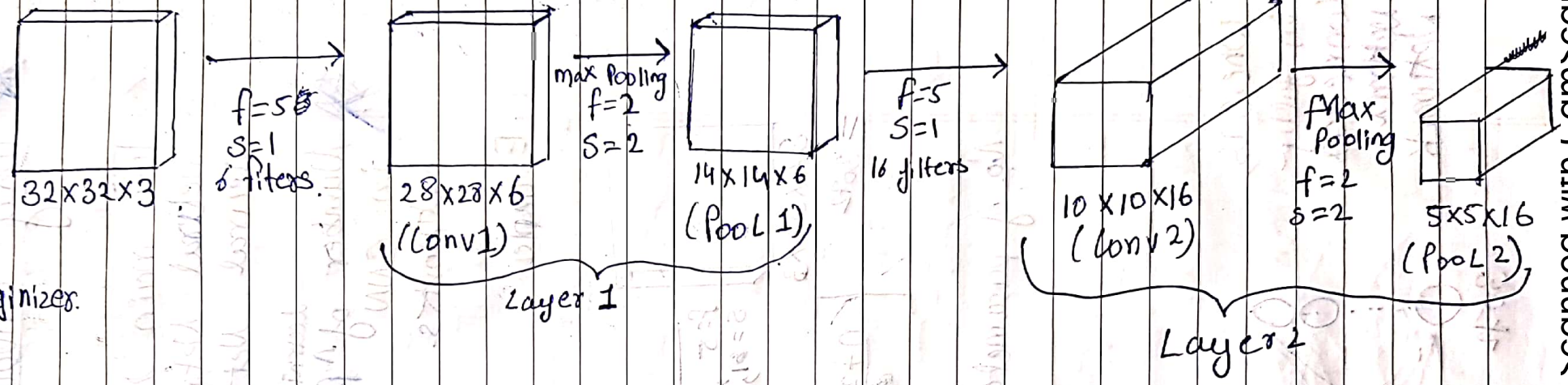
⇒ Pooling layers: do not have parameters

for fully connected layer

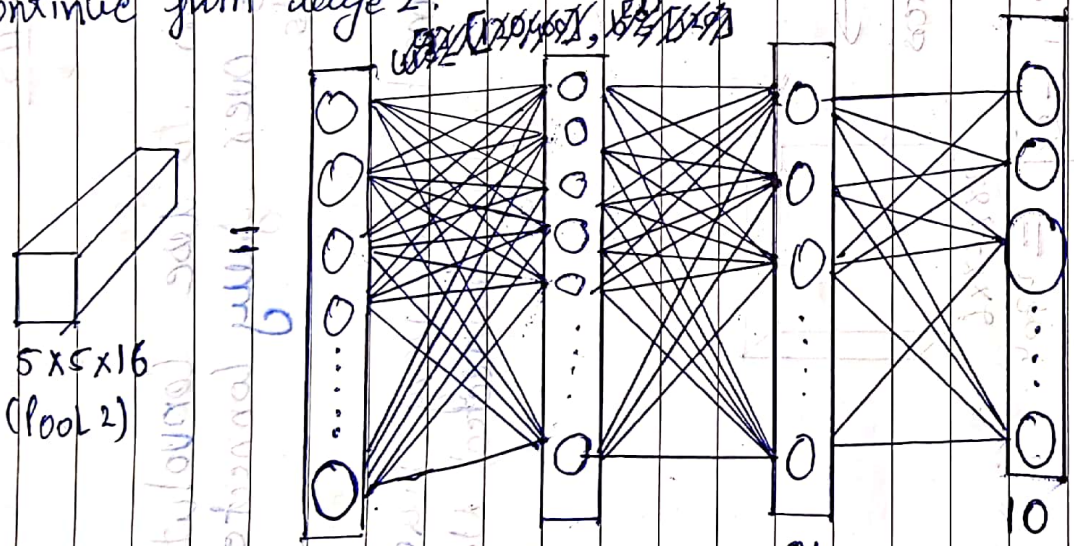
Rule:

(# Nodes in layer * # Nodes in previous layer) + # Nodes in layer (bias)

Consider.
digit-Recognizer.
Problem



Continue from above.



$\because 5 \times 5 \times 16 = 400$

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

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

(LeNet-5)

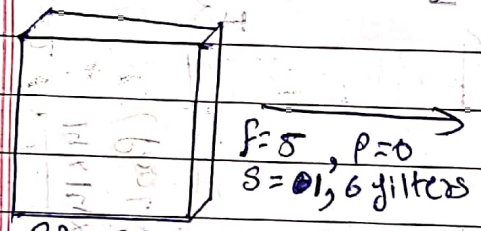
* As we go deeper in network n_H, n_W will decrease (↓) and n_c will increase (↑)
* common pattern of networks
  Conv - Pool - Conv - Pool - ... - Pool - FC - FC - Softmax
* $f = 5$, means $5 \times 5 \times n_c$
  n_c : number of channels.
  for example
  input $(32 \times 32 \times 3)$ $\xrightarrow{f=5}$ Conv $(28 \times 28 \times 6)$
  $5 \times 5 \times 3$
  6 filters

Why Convolutions

⇒ Two main reason to use Convolutional Neural Networks over fully connected neural Networks.

- Number of learning parameters are very much less in CNN than fully connected Neural Networks.

Example



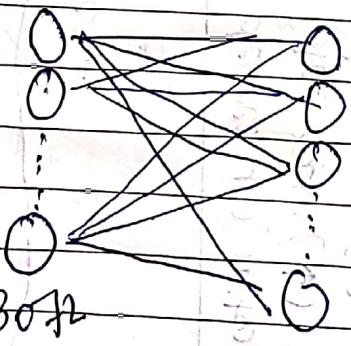
$= 3072$ $\quad \frac{n+2p-f+1}{s} = 28$ $\quad = 4704$

No. of parameter $= (f^{[l]} * f^{[l]} * n_c^{[l-1]} + 1) * \#filters$

$= (5 * 5 * 3 + 1) * 6$

$= 456$

Whereas, in fully-connected Neural Network



$= (3072 * 4704) + 4704$ (bias)

≈ 14 million

1) 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.

2) Sparisity of connections : In each layer, each output value depends only on a small number of input.

** Translation Invariance :
Using CNN methodology means that if image of cat is shifted by a few pixels, it'll still be identified as a cat.

* Putting it together.

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

x is image, y is label. (output)

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

$\hat{y}$ is predicted label (output)

use gradient descent to optimize parameters to reduce J

☆ Case Studies.

* Why look at case studies?

* Outline

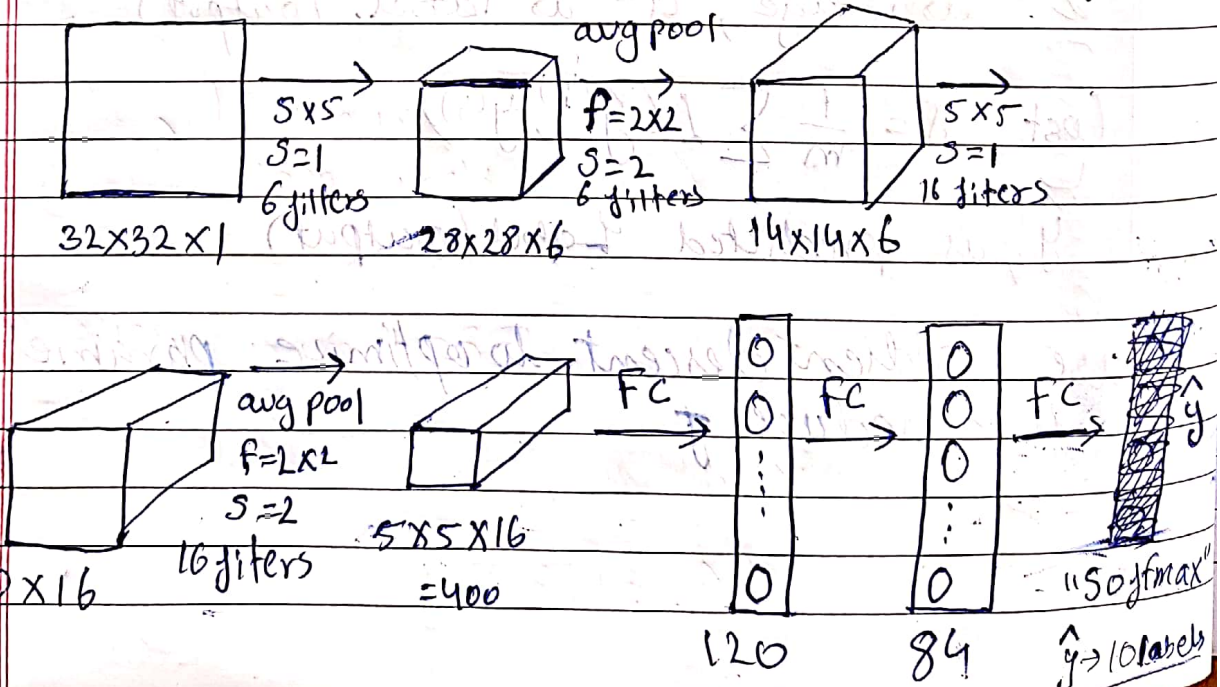
Classic networks:

- LeNet-5
- AlexNet
- VGG

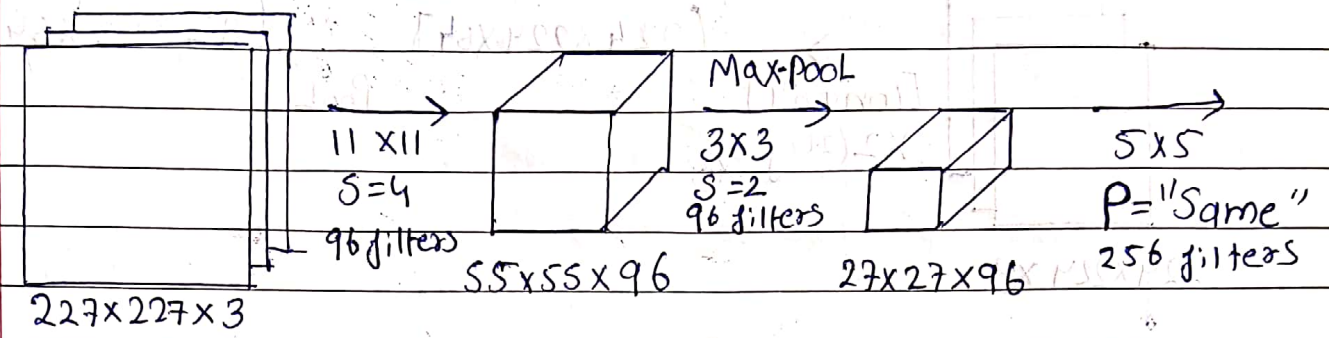
- ResNet (Residual Network, 152 layers)

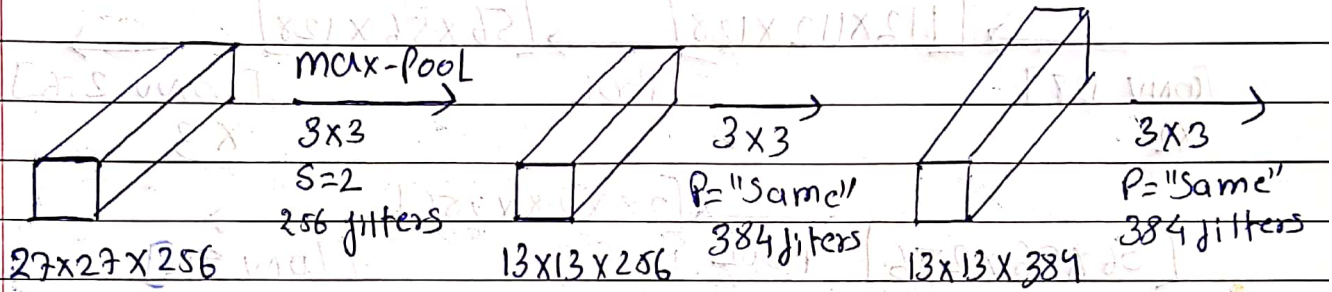
- Inception

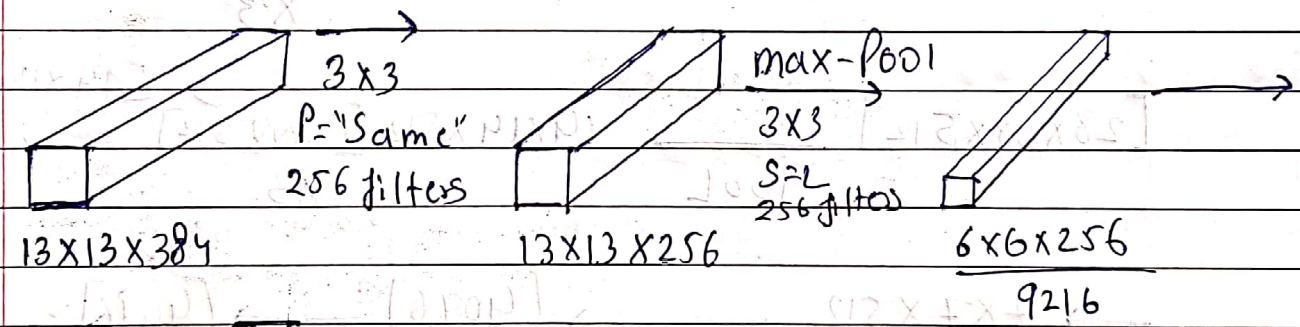
⇒ LeNet-5 (LeCun et al., 1998. Research paper)

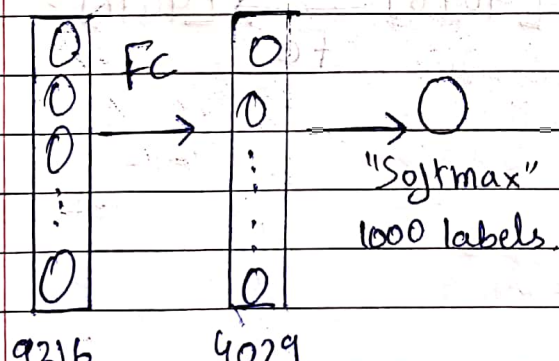


⇒ AlexNet [Krizhevsky et al., 2012. ImageNet classification with deep CNN]



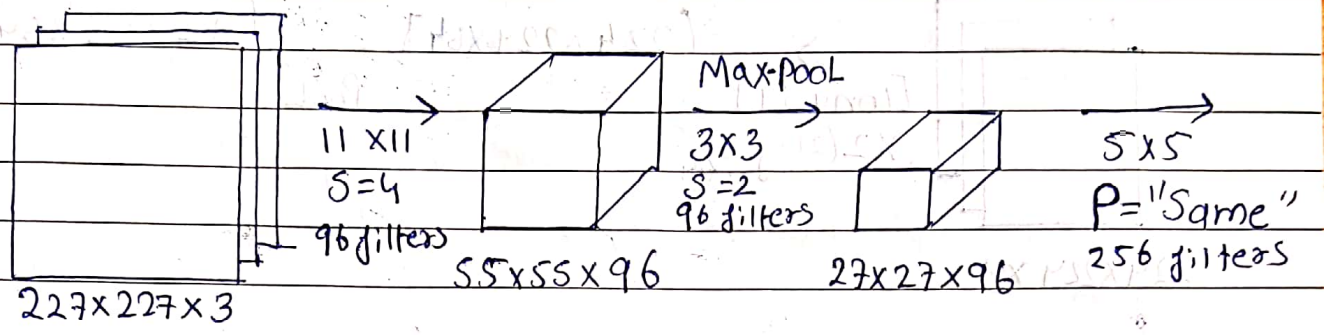


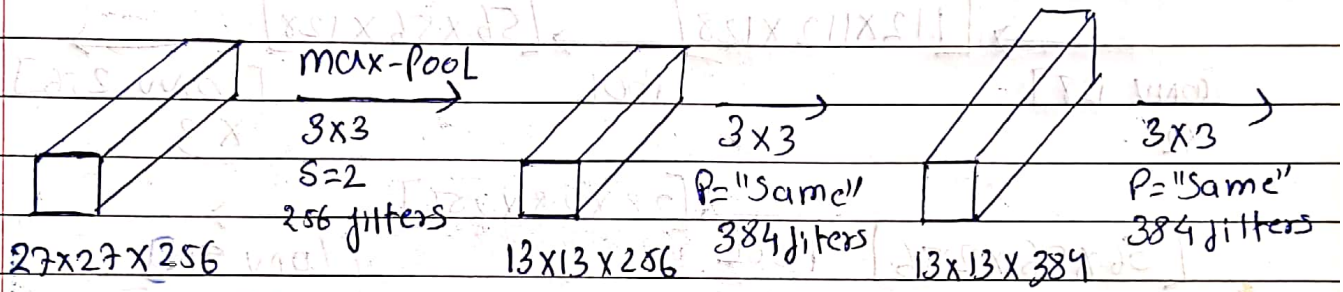


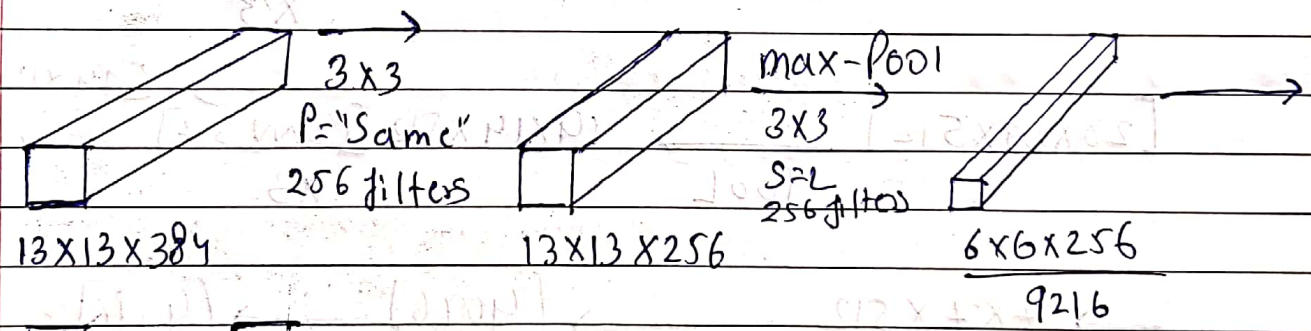


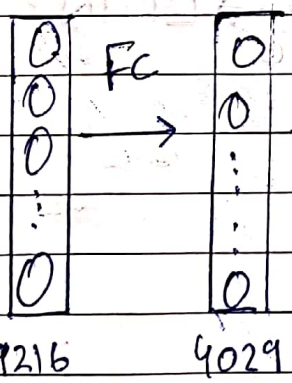
→ Similar to LeNet but much bigger (≈60M parameter)
→ ReLu as activation

⇒ AlexNet [Krizhevsky et al., 2012. ImageNet classification with deep CNN]







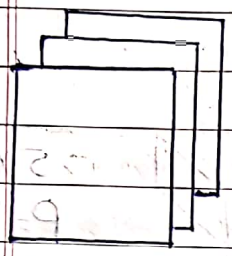


"Softmax"
1000 labels.

→ Similar to LeNet but much bigger (≈60M parameter)
→ ReLu as activation

⇒ VGG-16 [Simonyan & Zisserman 2015. Very deep CNN for large scale image Recognition]

(Conv = 3×3 filters, S=1, P="same") (Max-pool = 2×2, S=2)



224×224×3 → [Conv 64] ×2 (2 layers) → [224×224×64] → Pool → [112×112×64]

→ [Conv 128] ×2 → [112×112×128] → Pool → [56×56×128] → [Conv 256] ×3

[56×56×256] → Pool → [28×28×256] → [Conv 512] ×3

[28×28×512] → Pool → [14×14×512] → [Conv 512] ×3 → [14×14×512]

→ Pool → 7×7×512 → FC → [4096] → FC → [4096]

→ Softmax → 1000 labels

→ ≈138M parameters.

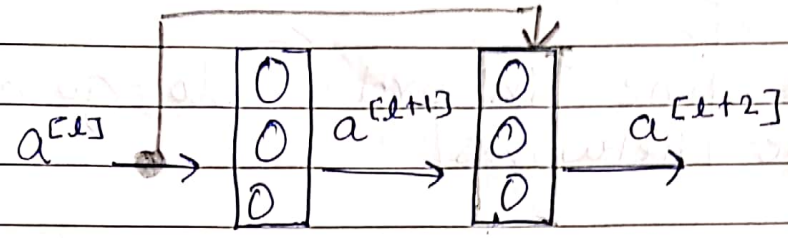
→ go deeper n_H, n_W (↓)

n_c (↑)

Case studies.

Residual Networks (ResNet)

Residual block.



$a^{[l]} \rightarrow$ linear $\rightarrow$ Relu $\xrightarrow{a^{[l+1]}}$ linear $\rightarrow$ Relu

$z^{[l+1]} = W^{[l+1]} a^{[l]} + b^{[l+1]}$ $\qquad$ $z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]}$

$a^{[l+1]} = g(z^{[l+1]})$ $\qquad$ $a^{[l+2]} = g(z^{[l+2]})$

$a^{[l]} \rightarrow$ linear $\rightarrow$ Relu $\rightarrow$ linear $\rightarrow$ Relu $\rightarrow a^{[l+2]}$

"main Path"

In ResNet

$a^{[l]} \rightarrow$ linear $\rightarrow$ Relu $\rightarrow$ linear $\rightarrow$ ReLu $\rightarrow a^{[l+2]}$

"Short cut"

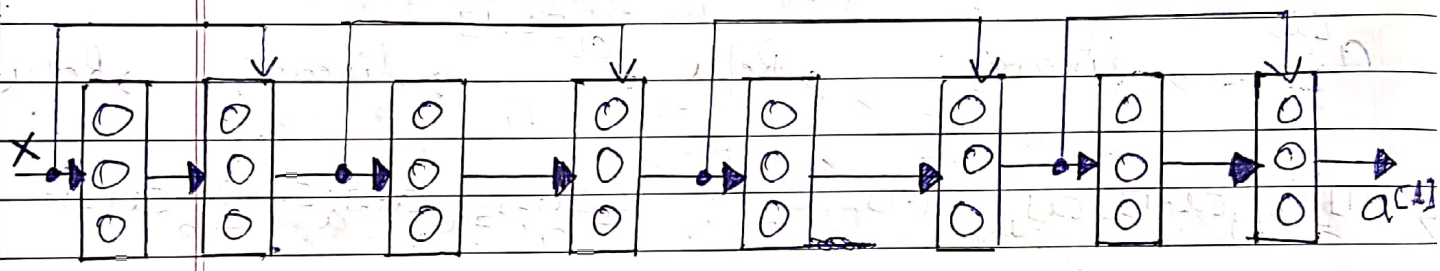
therefor Now equation

$$a^{[l+2]} = g\left(z^{[l+2]} + a^{[l]}\right)$$

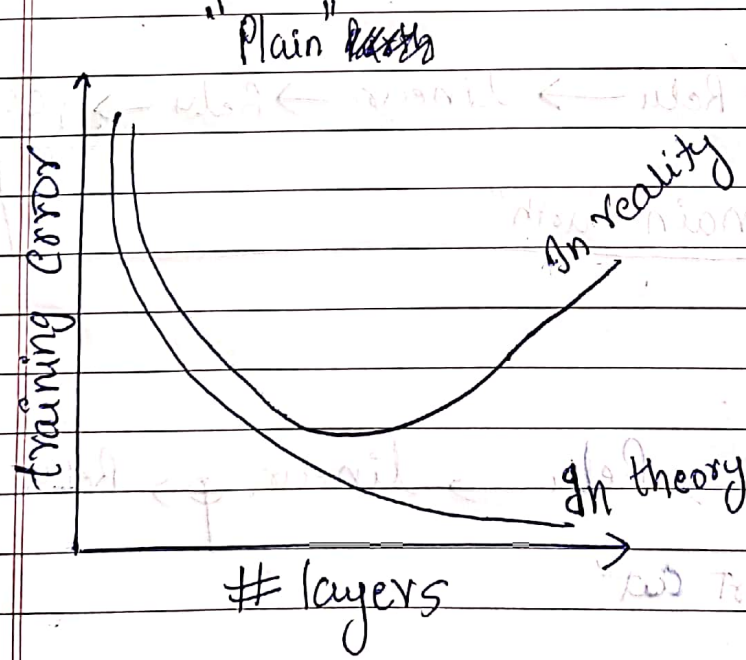
* This is called "Short cut" or "Skip Connection"

This is done in order to go deeper into the networks

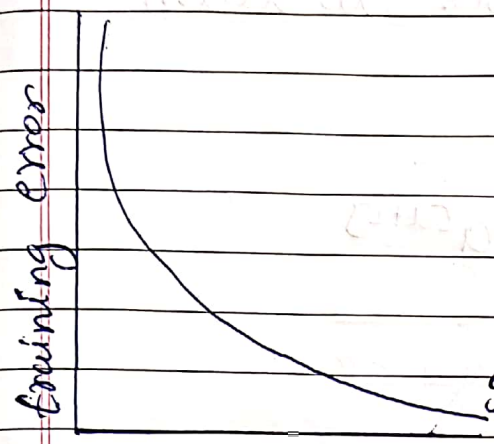
Residual Networks.



"Plain": Without Skipping layers.



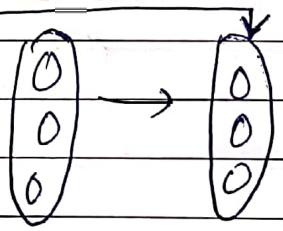
ResNet



ResNet allows to go deeper as it uses "short cut" or "skip connection" method without ↑ training error.

Why ResNets Work So well?

$X \rightarrow$ [Big NN] $\rightarrow a^{[l]}$

$X \rightarrow$ [Big NN] $\rightarrow a^{[l]} \rightarrow$  $\rightarrow a^{[l+2]}$

ReLu $a \geq 0$

$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

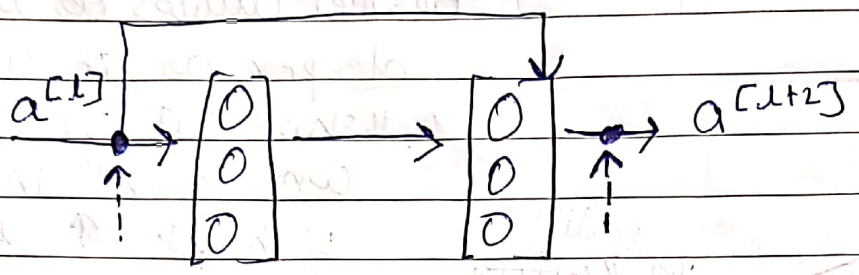
$$= g(w^{[l+2]} a^{[l+1]} + b^{[l+2]} + a^{[l]})$$

if $w^{[l+2]} = 0$, $b^{[l+2]} = 0$

then, $g = (a^{[l]})$

$$a^{[l+2]} = a^{[l]}$$

It shows that, identity function is easy for Residual block to learn.



"Same" Convolution means dimension is same at input point and output point.

In order to make "Same" Convolution multiple W_s to $a^{[l]}$

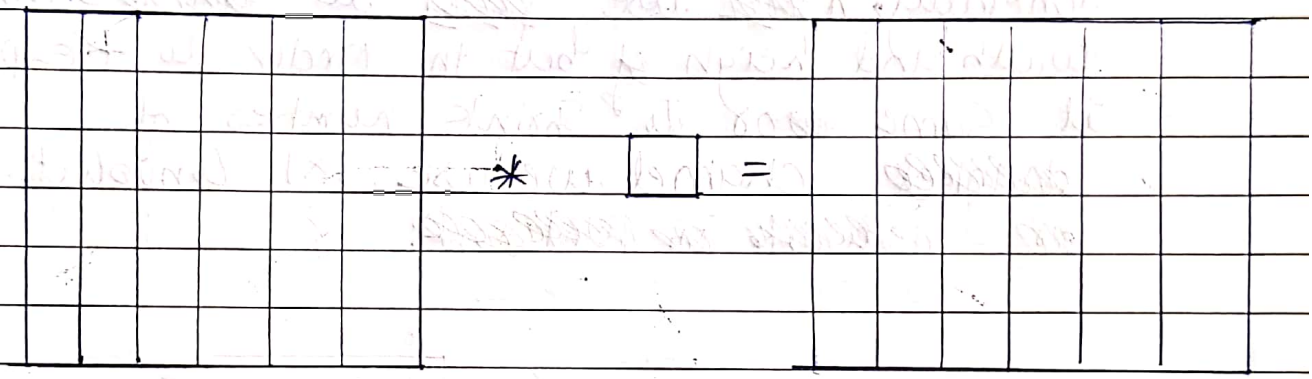
Example if $a^{[l+2]} \rightarrow \mathbb{R}^{256 \times 256}$

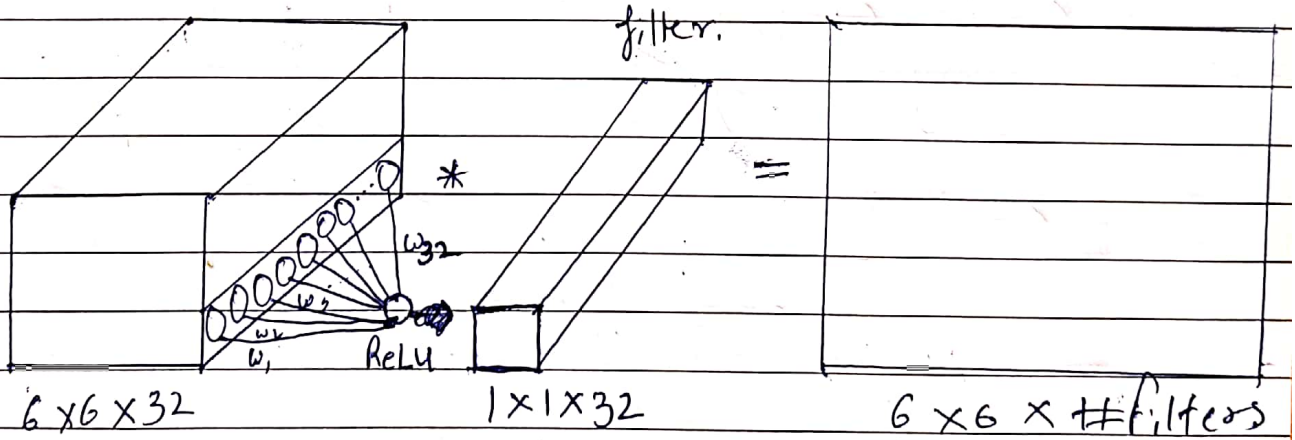
and $a^{[l]} \rightarrow \mathbb{R}^{128 \times 128}$.

then do, $W_s a^{[l]}$ where $W_s \rightarrow \mathbb{R}^{256 \times 12}$

Network in Network and 1X1 Convolutions.

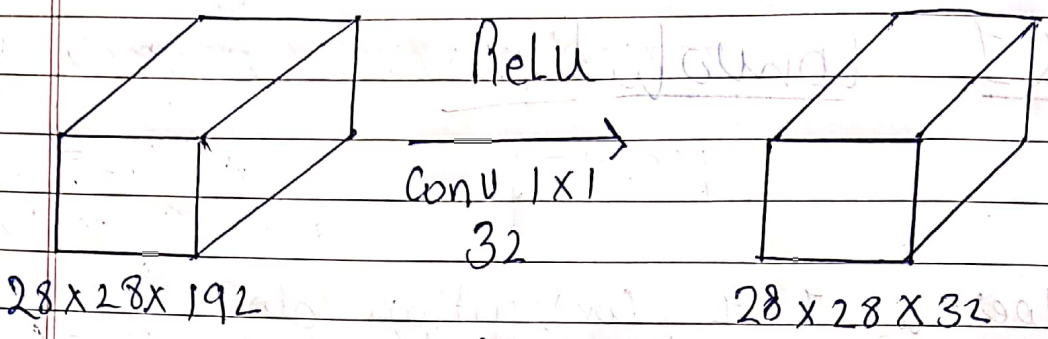
Why does a 1x1 convolution do?





here, w_1 = first channel of filter
w_2 = Second channel of filter
w_3 = Third channel of filter
w_{32} = 32th channel of filter.

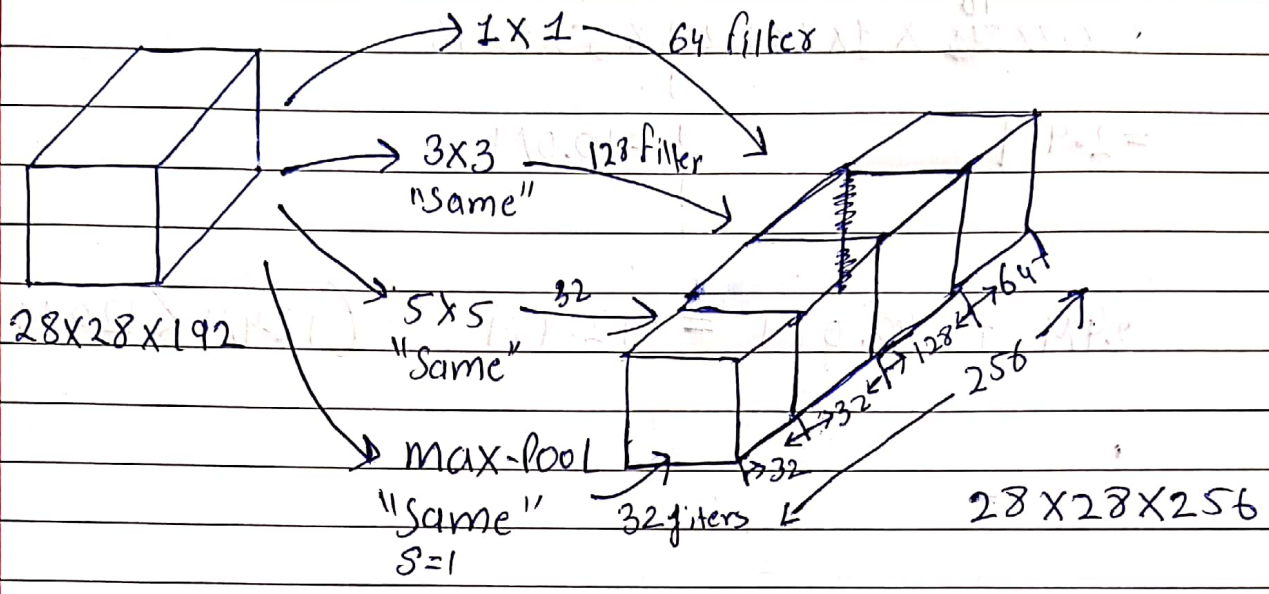
Example using 1x1 Convolutions



Conv or Pool layer

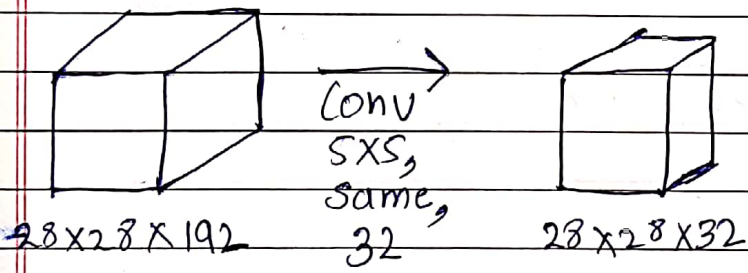
Generally, we use to Shrink width and heigh of but in order to keep it same or its Shrink numbes of channel we use 1x1 convolutions

Inception network motivation.



⇒ The problem of Computational Cost.

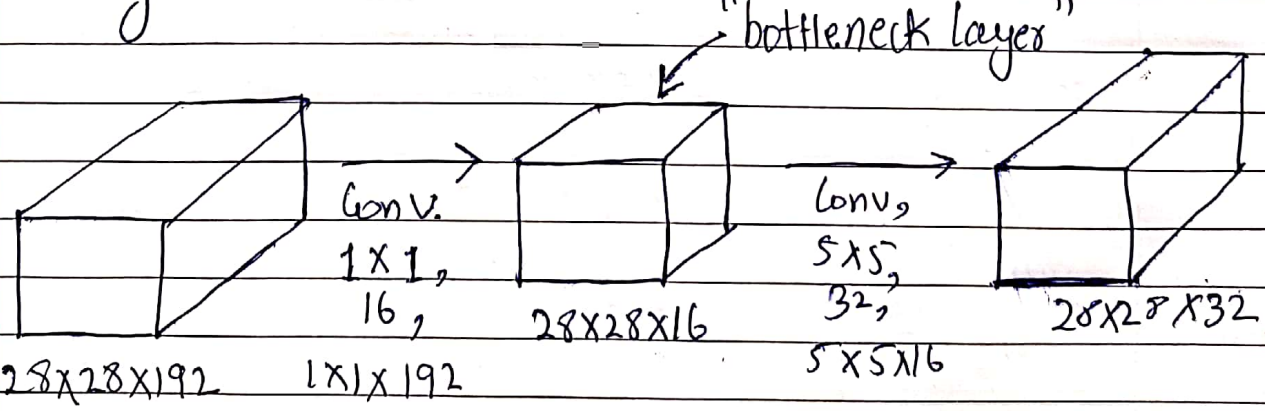
As it is 32 filter



5×5×192 ×
28×28×32

= 120M

- Using 1×1 convolution.



Let look at computational cost here,

$28 \times 28 \times 192 \times 1 \times 1 \times 16$ | $28 \times 28 \times 32 \times 5 \times 5 \times 16$

$= 2.4M$ | $= 10.0M$

$2.4M + 10.0M = 12.4M$ $(12.4M < 120M)$