CONVOLUTIONAL NEURAL NETWORKS

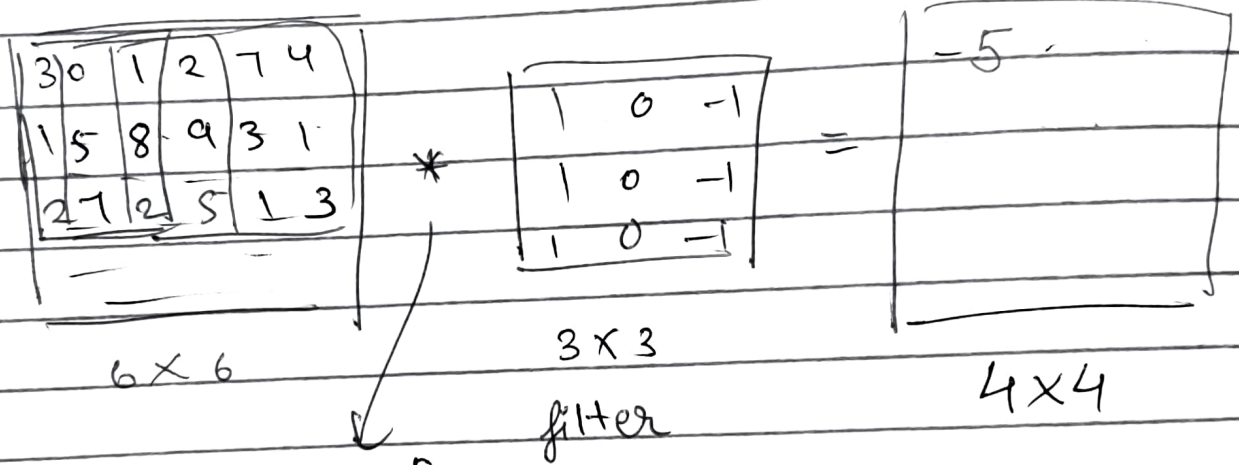
* Computer Vision

Problems: Image classification, object detection, Neural style transfer

* Edge detection

~~vertical~~ vertical edge detection:

Consider a 6×6 grayscale image



convolution operator

element-wise multiplication with the filter

ie the (1,1) th element of resultant will be

$$(3\times1)+(1\times1)+(2\times1)+(0\times0)+(0\times5)+(0\times7)+(1\times-1)+(8\times-1)+(2\times-1)$$

$$= -5$$

A =

$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix}$$

Consider image A it should look like 

filter $F = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$

$A * F = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}$ 

edge

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

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

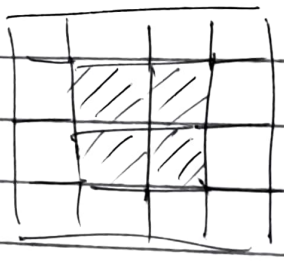
Schorr filter $\begin{matrix} 3 & 0 & 3 \\ 10 & 0 & 10 \\ 3 & 0 & 3 \end{matrix}$

Padding

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

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

to keep the dimensions of the resultant image the same as the original image, add a border to it



here padding $p = 1$

resultant will be

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

valid conv: no filter padding

same conv: pad so that output size is the same as the input size

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

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

by convention f is odd

Strided Conv

consider a 7×7 input, 3×3 filter
now instead of moving our grid window by 1, we move it by 2.

2 3 7 4 6 2 9
6 6 9 7 8 4 3
3 4 8 3 3 9 7
. . .
. . .
.

filter in 1, 1
2 3 7
6 6 9
3 4 8

in iteration 2:
7 4 6
9 7 8
8 3 3

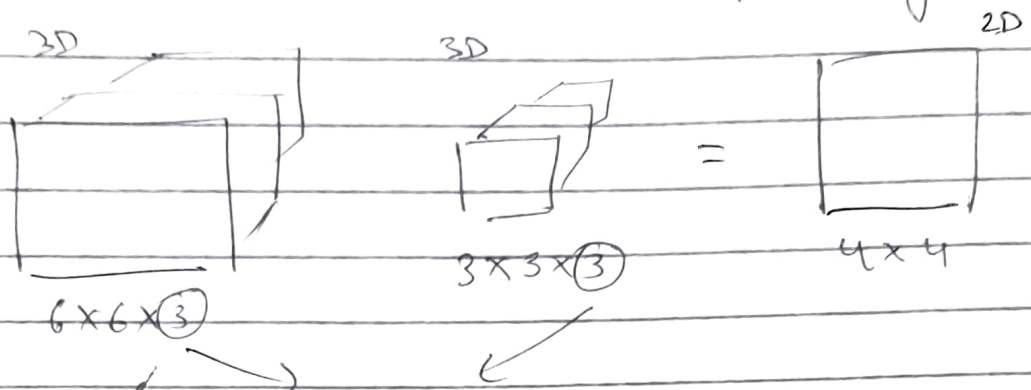
Input: 7×7 filter: 3×3 output: 3×3
Here stride = 2

input n×n with filter f×f ~~or~~
with padding p & stride s
output:

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

floor

Convolutions over volumes / RGB images



also called channel

should be equal

do element wise multiplication on the cube window

if you want vertical edge detection on only the red channel

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

if it ~~do~~ color doesnt matter

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

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

	0	-1
1	0	-1
1	0	-1

If you want to detect both vertical & horizontal edge

vertical

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

horizontal

$3 \times 3 \times 3 = 4 \times 4$

$\rightarrow 4 \times 4 \times 2$

One layer of CNN

$6\times6\times3 \xrightarrow{*\ 3\times3\times3} \xrightarrow{\text{ReLU}(4\times4 + b_1)} 4\times4$ (→ $\mathbb{R}$)

$\xrightarrow{*\ 3\times3\times3} \xrightarrow{\text{ReLU}(4\times4 + b_2)} 4\times4$

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

$p^{[l]}$ = padding

$s^{[l]}$ = stride

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

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

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

A simple ConvNet

$a^{[l]}$

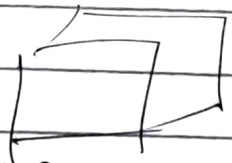
Consider:

$39\times39\times3$

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

$n_c = 3$

$f^{[1]} = 3$, $s^{[1]} = 1$, $p^{[1]} = 0$, 10 filters



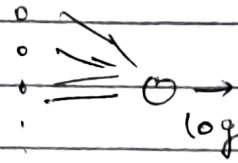
$37\times37\times10$

20 filters, $f^{[2]} = 5$, $s^{[2]} = 2$, $p^{[2]} = 0$

$17\times17\times20$

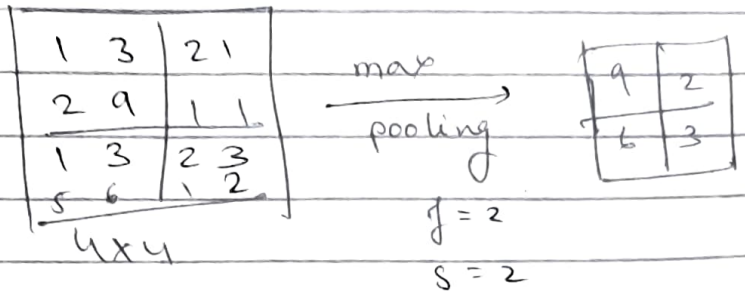
$f^{[3]} = 5$, $s^{[2]} = 2$, 40 filters

$7\times7\times40$



→ logistic / softmax

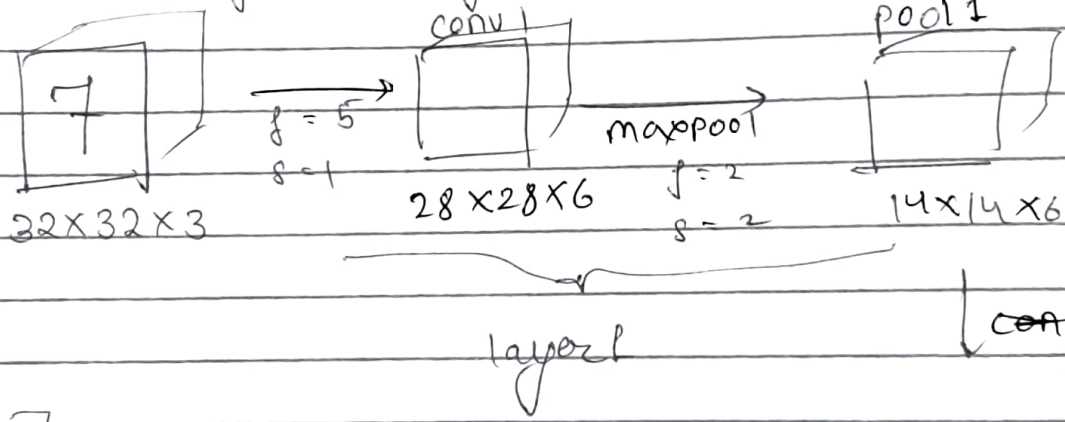
Pooling Layers



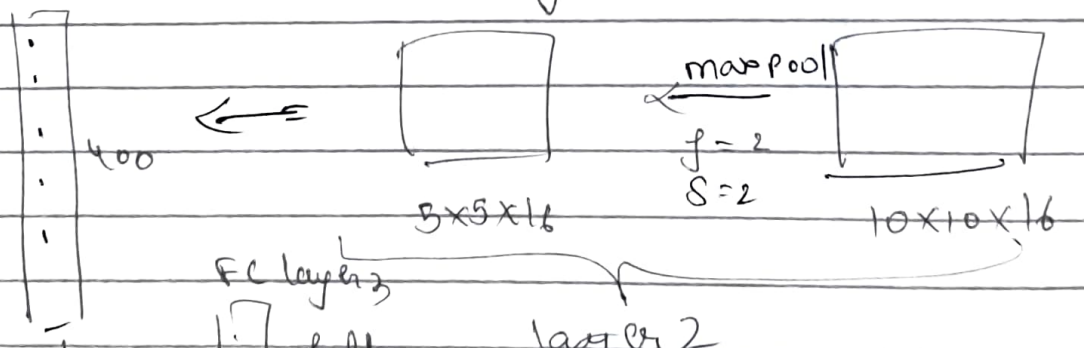
avg pooling, min pooling ...
padding is rarely used in pooling.

CNN Example

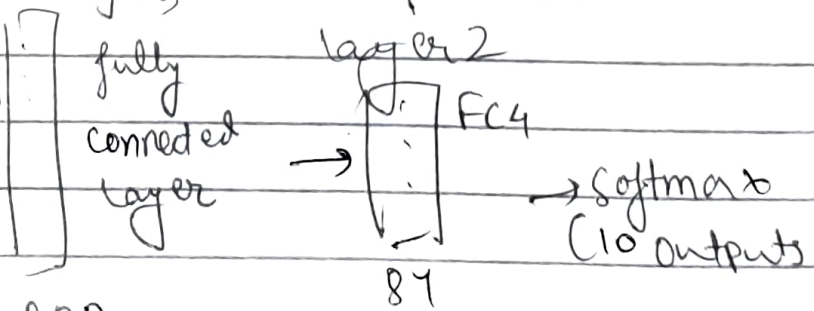
LeNet - 5
to recognize digits



layer1

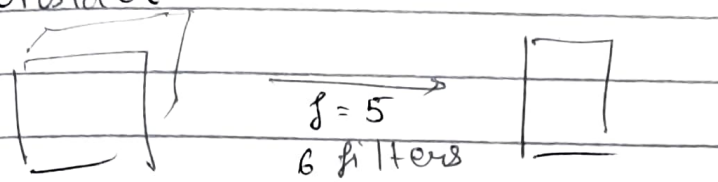


FC layer3



Why Convolutions

Consider



3072

4704

if it were a fully connected layer, we would need $3072 \times 4704 \approx 14$ million parameters

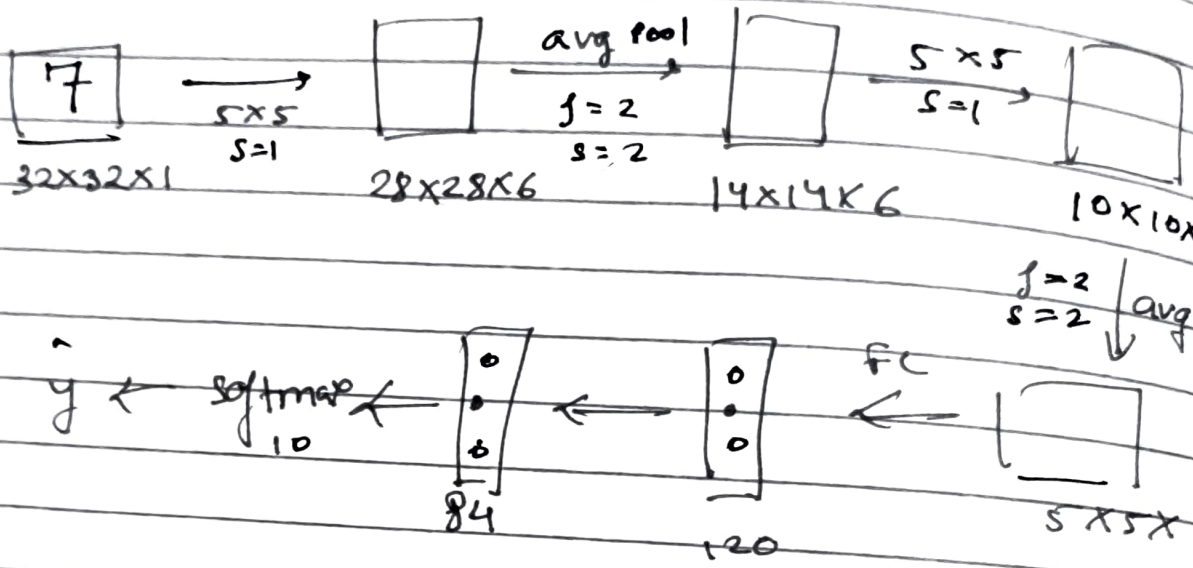
but from conv we have $(5 \times 5 + 1) \times 6 = 156$ params

Why convs get away w so few parameters?

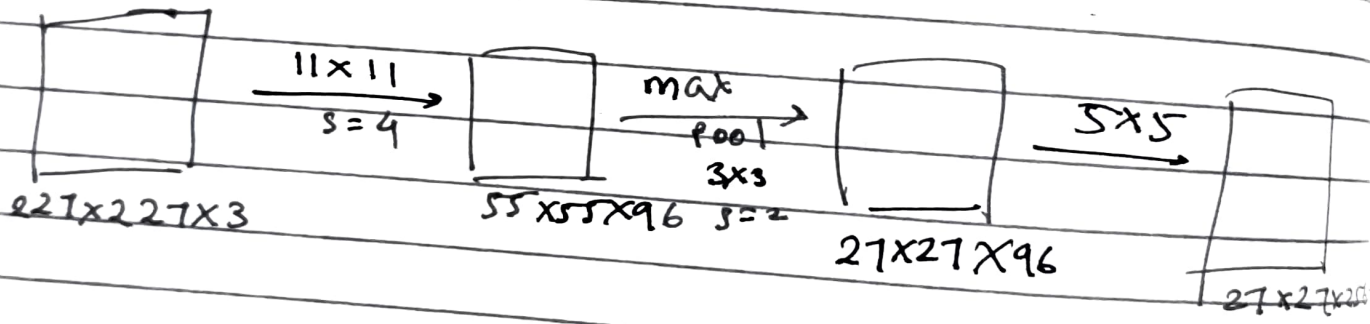
1) Parameter sharing: A feature detector (eg vertical edge detector) will mostly be usable in other part of the image
2) Sparsity of connections: In each layer, an output is dependent on a small no. of input features.

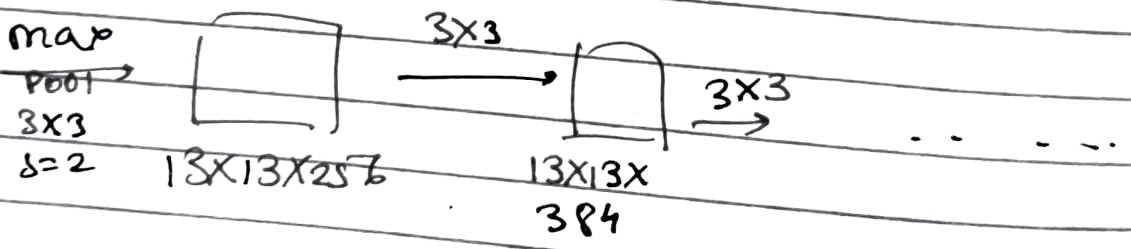
Classical Networks

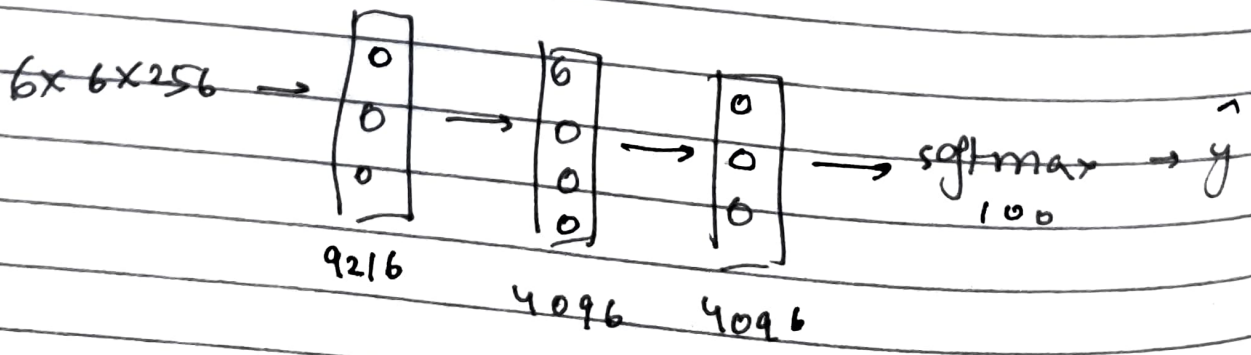
* LeNet 5



* AlexNet







* VGG 16

Conv: 3×3 filters $s=1$

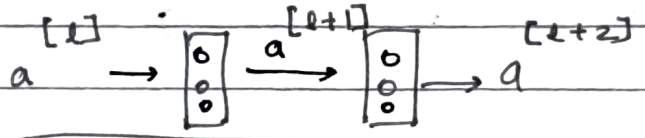
maxpool: 2×2, $s=2$

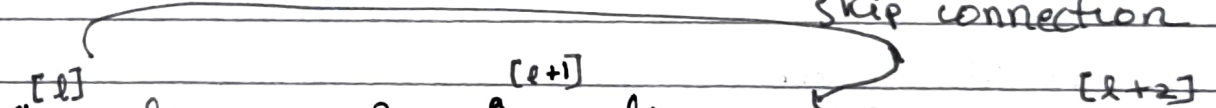
$224\times224\times3 \xrightarrow[\times2]{\text{conv } 64} 224\times224\times64 \xrightarrow{\text{pool}} 112\times112\times64$

$\xrightarrow[\times2]{\text{conv } 128} 112\times112\times128 \xrightarrow{\text{pool}} 56\times56\times128 \quad \cdots$

Residual Networks (ResNets)

* residual block





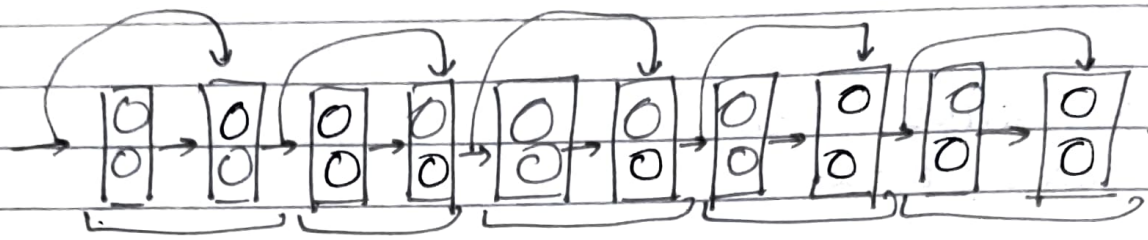
$$z^{[l+1]} = w^{[l+1]} a^{[l]} + b^{[l+1]}$$

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

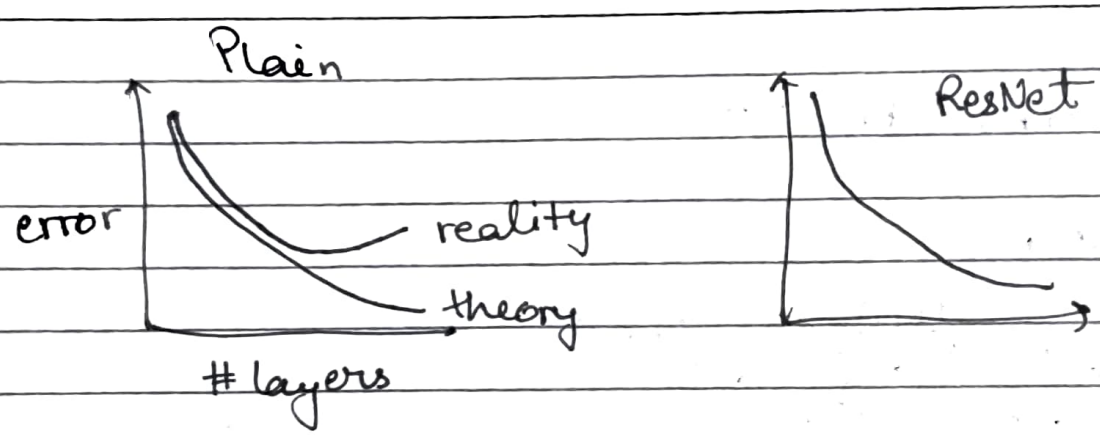
$$z^{[l+2]} = w^{[l+2]} a^{[l+1]} + b^{[l+2]}$$

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

$a^{[l]}$: residual block

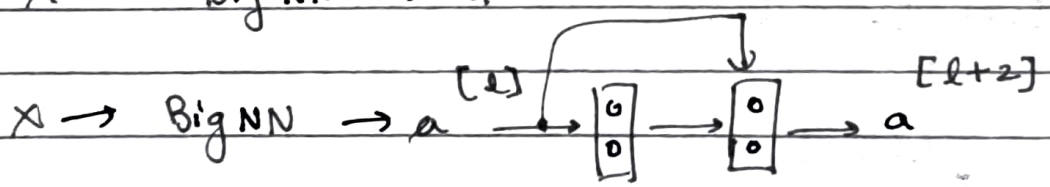


has 5 residual blocks
without the residual connections,
the network is called "plain"



Why ResNets work

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



activation : ReLU $a \geq 0$.

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

$$= g(\underbrace{w^{[l+2]} a^{[l+1]} + b^{[l+2]}}_{x \quad x} + a^{[l]}) = g(a^{[l]})$$

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

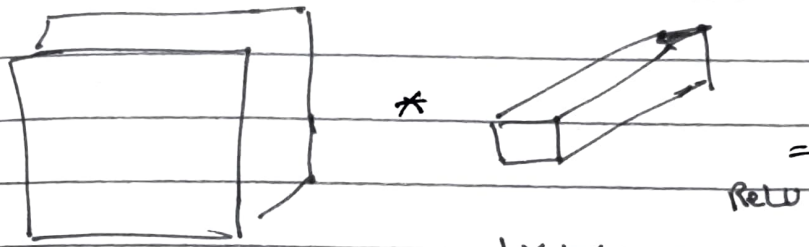
if $w^{[l+2]} = b^{[l+2]} = 0$

1x1 Convolutions

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} * \boxed{2} = \begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \\ 14 & 16 & 18 \end{bmatrix}$$

is basically element wise multiplication in this case

3x3x1



6x6x32 * 1x1x32 = 6x6x# filters

← inception network →

Practical Advice

* Use open source CNN implementations available on github
* Use weights trained by other people for your network. (transfer learning)
* Augment your data to increase the data set size and hence improve performance
  - mirroring
  - random cropping
  - rotating
  - color shifting (R+20, G+20, B+20)

Object Detection

Object localization

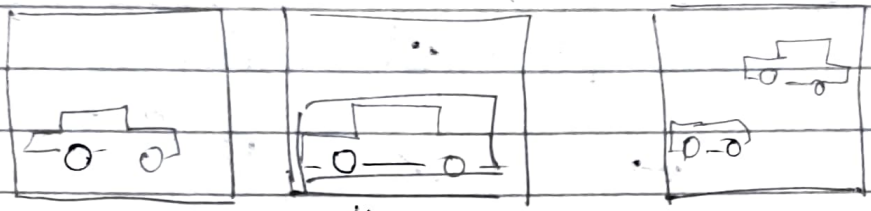


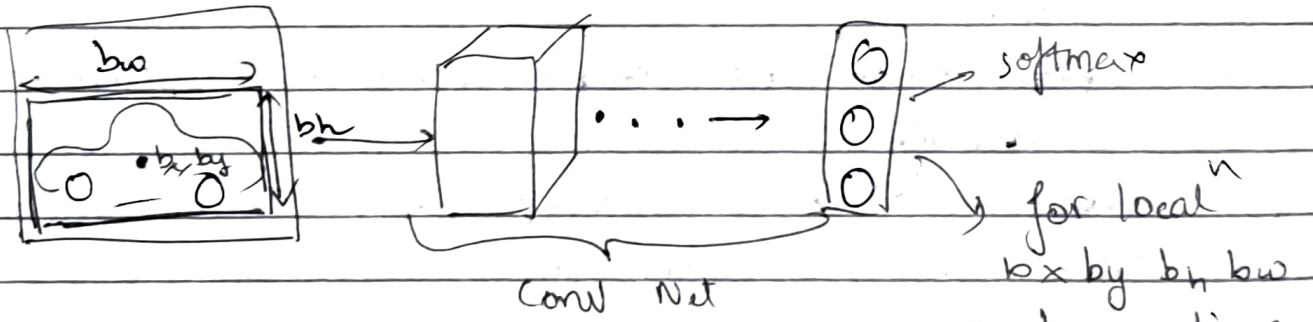
image classificat^{n}

image classificat^{n} & localizat^{n}

Detection

(object)

multiple objects



1. pedestrian
2. car
3. motorcycle
4. background

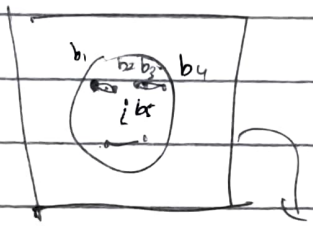
$y =$ p_c → is there any object?
b_x
b_y
b_h
b_w
c_1
c_2
c_3
[illegible]

$$f(\hat{y}, y) = (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 \cdots (\hat{y}_8 - y_8)^2 \quad \text{if } y = 1$$

$$= (\hat{y}_1 - y_1)^2 \quad \text{if } y_1 = 0$$

* Landmark Detection

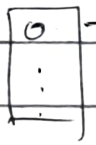
- face detect^n



has 64 landmarks (single points) corners of the eyes, nose locators, lips etc

used in AR

convnet



→ face - ? 1 : 0

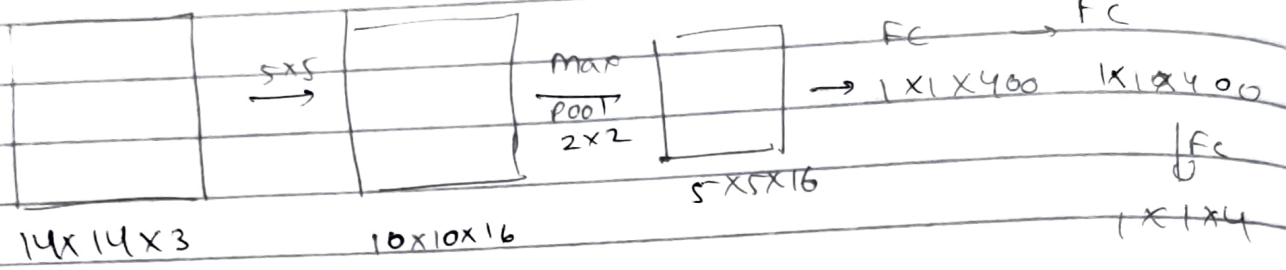
$b_1 x$
$b_1 y$
$b_2 x$
$b_2 y$
⋮

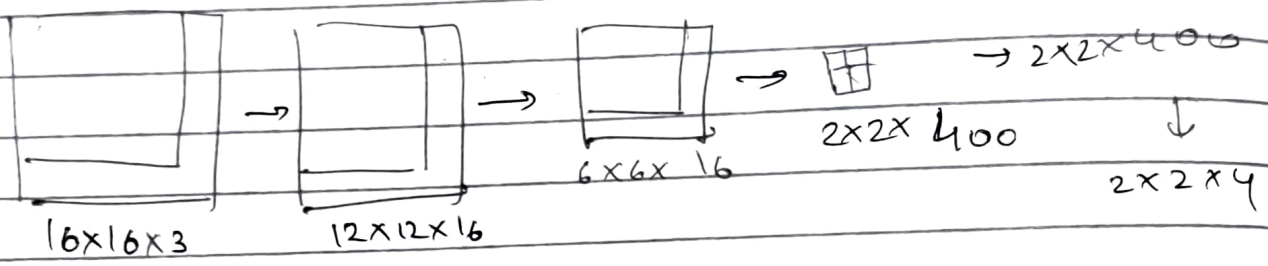
} 129 outputs

* Object Detect^n

- Train the model using a dataset of closely cropped object images / bgs
- Use a ~~sling~~ sliding window grid over the larger test image to localize the object.
- change the size of window & iterate again
- has high computational cost.

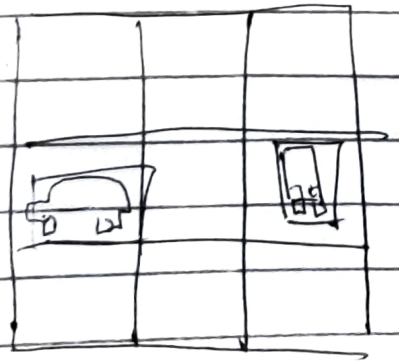
* Conv implementation of Sliding windows.





* Bounding box predictions
the ~~str~~ sliding windows from the convolution strides may not fit the objects accurately
- ~~U~~use You only Look Once algo (YOLO) algorithm
- Divide the test image to n×n grids.
apply conv to each grid

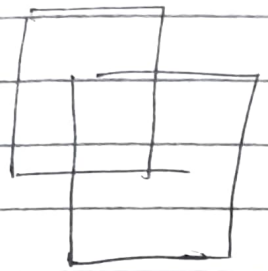
suppose
conv gives output : y: $\begin{bmatrix} p_c \\ b_x \\ \vdots \\ c_3 \end{bmatrix}$



→ now returns a 3×3×8 output

* Intersection Over Union

Lets say there are 2 boundary regions who which localized the object

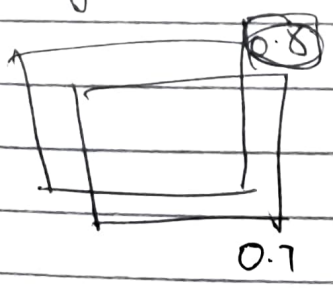


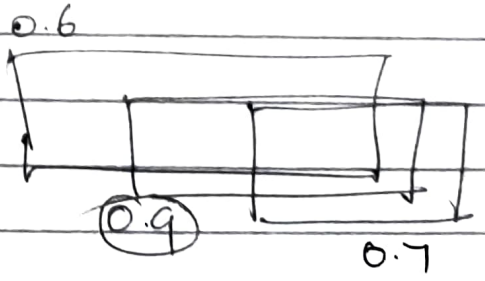
Intersection over Union (IoU)
= $\frac{\text{size (area) of interset region}}{\text{size(area) of union region}}$

correct if IoU ≥ 0.5 usually

more generally, it is the measure of overlap of two boxes

* Non Max Suppression:
What if the object appears on more than one grid. ~~There should be~~
The count of object should not be incremented for each appearance of the object





the higher pc grid is kept

Consider the image divided into 19x19 grids

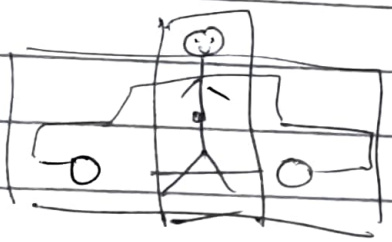
each output prediction gives

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \end{bmatrix}$$

- discard all boxes with $p_c \leq 0.6$
- pick the box with highest p_c output that as predict^n
- discard any box with ~~$p_c \leq 0.5$~~ $p_c \geq 0.5$ with the box from the previous step

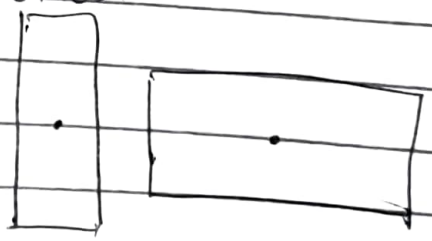
* Anchor box

What if we have multiple overlapping objects.



we have two anchor boxes:

box1



Output earlier: $3 \times 3 \times 8$

now: $3 \times 3 \times 16$

* Putting Together YOLO

Face Recognition

	Face recognition	vs	Face verification
input:	image		image, name / ID
output:	id of person in image / not recognized		whether the input image is of the claimed person

* One Shot learning

algo gets only 1 image per person

conv + softmax doesn't work very well in this situation

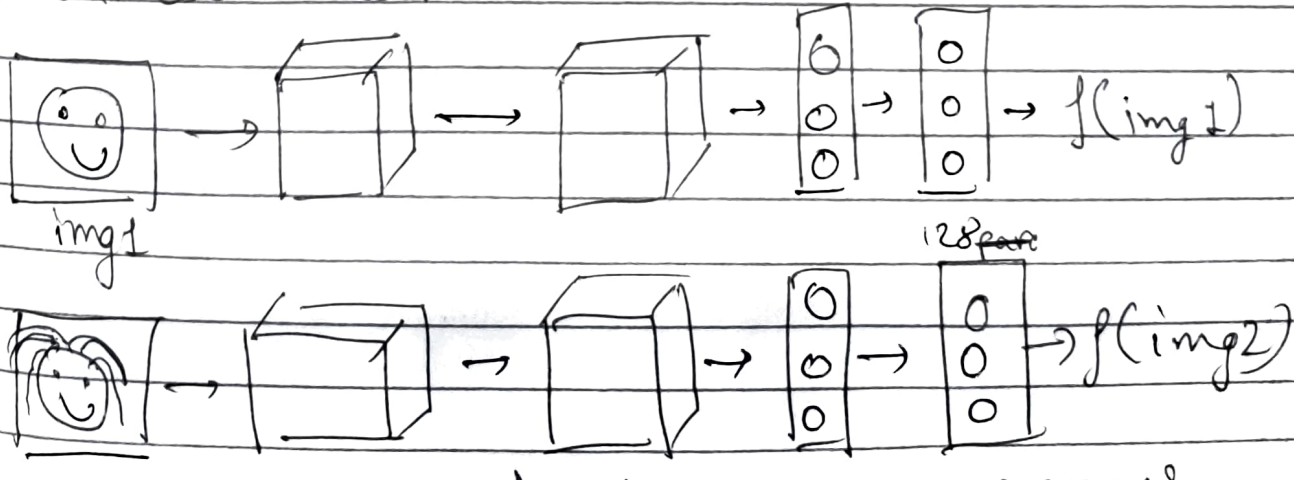
The model has to learn a similarity fn

d(img1, img2) = degree of diff b/w images

$d(img1, img2) \leq \tau$ "same"

$d(img1, img2) > \tau$ "not same"

* Siamese Network



same network so same params

$$d(img1, img2) = \|f(img1) - f(img2)\|_2^2$$

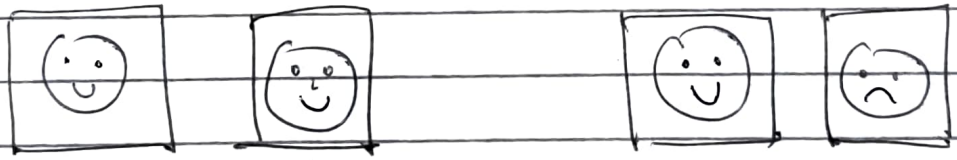
Learn parameters s.t.

if img1 & img2 are same person,
$\|f(img1) - f(img2)\|$ is small

if img1 & img2 are diff person,
$\|f(img1) - f(img2)\|$ is large

* Triplet loss fn

Learning objective:



anchor A — positive P — anchor A — negative N

same person — diff person

want $\|f(A) - f(P)\| \leq \|f(A) - f(N)\|$

Or often written

$$\|f(A) - f(P)\| - \|f(A) - f(N)\| + \underbrace{\alpha}_{\text{margin}} \leq 0$$

Given 3 images A, P, N

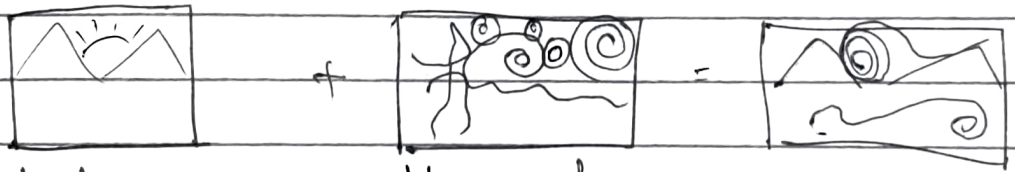
$$\mathcal{L}(A, P, N) = \max\left(\|f(A) - f(P)\| - \|f(A) - f(N)\| + \alpha, 0\right)$$

$$J = \sum_{i=1}^{m} \mathcal{L}(A^{(i)}, P^{(i)}, N^{(i)})$$

Choose A & corresponding P dont choose N randomly for training

Choose triplets that are hard to train on i.e N ~~shar~~ can be the image A closely resembles to.

Neural Style Transfer



landscape Content(C) | Van gogh Style(S) | generated image (G)

kinda like what prisma did back in the day

* what are deep convnets learning here
As we move towards deeper layers, they detect more complex structures

* Cost fn
define a cost fn J s.t.
$J(G) = \alpha J_{content}(C, G) + \beta J_{style}(S, G)$

To find generated image G

1. Initialize G randomly

   G: 100 × 100 × 3

2. Use gradient descent to minimize $J(G)$

$$G := G - \frac{\partial}{\partial G} J(G)$$

* The content cost fn

ie.

~~$J(G)$~~ $= \alpha J_{content}$

- Say you use hidden layer l to compute content cost
- Use pretrained Convnet (VGG network)
- Let $a^{[l](C)}$ & $a^{[l](G)}$ be the activation of layer l on the images
- If $a^{[l](C)}$ & $a^{[l](G)}$ are similar, both have similar content.

$$J_{content}(C, G) = \frac{1}{2} \| a^{[l](C)} - a^{[l](G)} \|^2$$

* Style Cost fn

~~Any~~ Take any intermediate conv ~~lay~~ volume from the net

It has multiple channels >3

They store more complex properties of the image like vertical, edges, circular patters etc

let

$a^{[l]}_{i,j,k}$ = activation at (i, j, k). $G^{[l]}$ is $n_c \times n_c$

$$G^{[l]}_{kk'} = \sum_i \sum_j a^{[l](s)}_{ijk} \, a^{[l](s)}_{ijk'}$$

$$G^{[l]}_{kk'} = \sum_i \sum_j a^{[l](q)}_{ijk} \, a^{[l](q)}_{ijk'}$$

$$J^{[l]}_{style}(s, q) = \| G^{[l](s)} - G^{[l](q)} \|^2_F$$

$$= \frac{1}{(2 n_H n_W n_C)^{[l]}} \sum_k \sum_{k'} \left(G^{[l](s)}_{kk'} - G^{[l](q)}_{kk'} \right)$$