CNN

Vertical Edge detection

- Convolutional Operation

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

6×6 grayscale img

* convolution operation

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

3×3 kernel/ filter

=

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

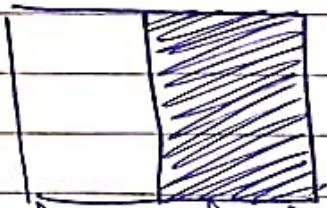
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

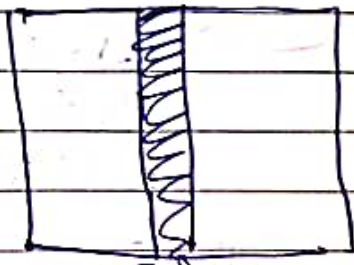
*

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





Vertical Edge is a n×n filter which has bright pixels on left and dark on right

positive edge

If transition is light to dark, edge is light and if " is dark to light, edge is dark

↳ neg edge

- Filter for Horizontal Edge

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

- Sobel filter

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

puts more emphasis on centre pixel hence more robust

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

→ Scharr filter

We can initialize the nine values of 3×3 filter as random values and learn their values by back propagation.

→ Image → n×n
Filter / kernel → f×f
Output → n-f+1 × n-f+1

To reduce shrinking of image and loss of data in convolutional operation, use padding
If p is padding amount then:
Output: n+2p-f+1 × n+2p-f+1
Generally by convention, padding is of zero and output size same as of image

- Valid convolution: $p = 0$
- Same convolution: ~~$p + k \in Z$~~ $p = k$ $[1, 2, 3 \ldots]$

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

$\therefore 2p - f + 1 = 0$

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

$\therefore$ If f is generally odd sized filter i.e. 3×3 or 5×5 etc so p is an integer and symmetric padding.
If f is even → asymmetric padding
In odd sized filter, central pixel can be used to ~~denote~~ denote location of filter

Input image → $n \times n$
kernel → $f \times f$
padding → p
stride → s

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

If this is not integer, round it down

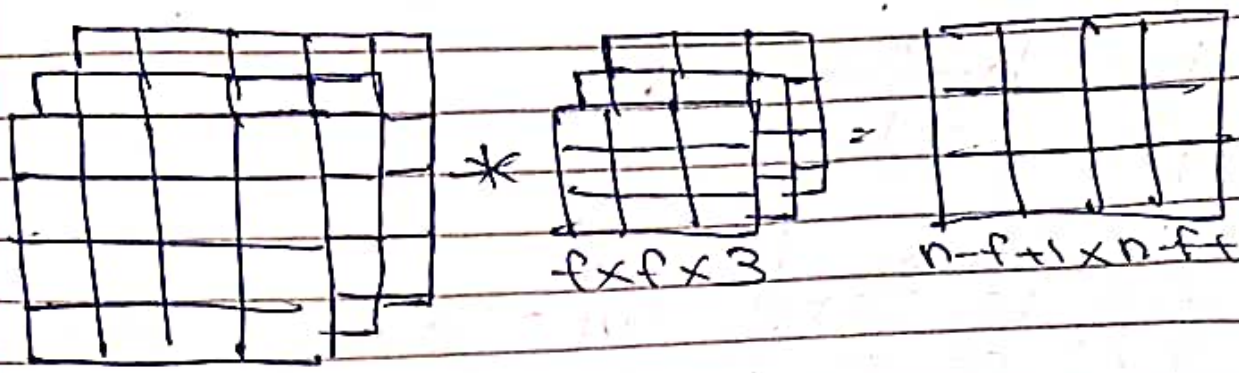
$\frac{n+2p-f}{s} + 1 = n$

$n + 2p - f + s = ns$

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

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

In RGB Image



$n \times n \times 3$

↳ no. of channels

Input → $n \times n \times n_c$ — same
Kernel → $f \times f \times n_c$
Output → $n-f+1 \times n-f+1 \times n_f$

↳ no. of filters

If padding = p stride = s then

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

Convolutional Neural net of l layers

$f^{[l]}$ → filter size of layer l
$s^{[l]}$ → stride of layer l
$p^{[l]}$ → padding of layer l
$n_c^{[l]}$ → no. of filters in layer l
Input → $n_h^{[l-1]} \times n_w^{[l-1]} \times n_c^{[l-1]}$
(Input of a layer is output of prev)

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

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

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

Size of each filter = $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$

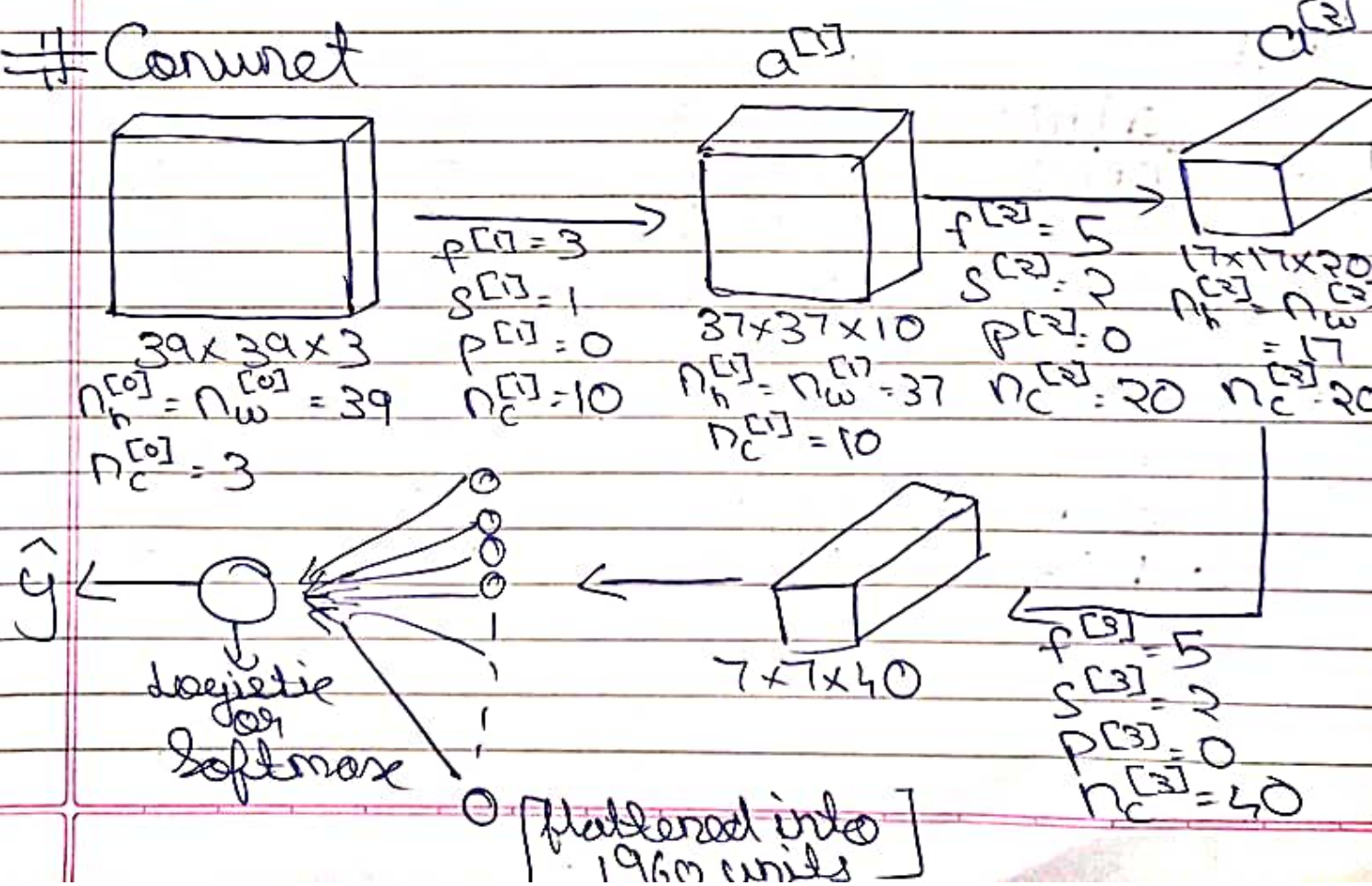
∴ After Act Activation,
$a^{[l]} \rightarrow n_h^{[l]} \times n_w^{[l]} \times n_c^{[l]}$

In Vectorized form,
$A^{[l]} \rightarrow m \times n_h^{[l]} \times n_w^{[l]} \times n_c^{[l]}$
↳ no. of training Examples

Weights $w^{[l]} = f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$

Bias → $b^{[l]} \rightarrow n_c^{[l]}$

Convnet



Max and Avg Pooling

Input → $n_h \times n_w \times n_c$
filter size → $f \times f \times n_c$
~~Output → $n_h - f + 1$~~
Stride → s
Padding → p

Output → $\left\lfloor \frac{n_h + 2p - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_w + 2p - f}{s} + 1 \right\rfloor \times n_c$

In Each channel, Pooling happens individually.
No. of Input channel = no. of output channel
No Parameters to learn, once set, they are constant

~~Example of Convnet~~

		$a^{[l]}$ shape	$a^{[l]}$ size	Params
	Input:	32,32,3	3072	0
f=5 s=1	Conv 1	28,28,6	4704	456
f=2 s=2	Pool 1	14,14,6	1176	0
f=5 s=1	Conv 2	10,10,16	1600	2416
f=2 s=2	Pool 2	5,5,16	400	0
	FC3	120,1	120	48120
	FC4	84,1	84	10,164
	Softmax	10,1	10	850
	FC1	400,1	400	400

Page No.
Date

Example of ConvNext

Object Detection

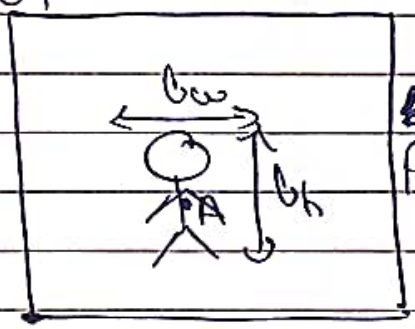
Page No.
Date

Input → Random image
Output → 1] Class 1 (Pedestrian)
2] Class 2 (Car)
3] Class 3 (Motorcycle)
4] Background (NOTA)

Output Vector →
- P_c → 1 if object is there, 0 otherwise
- b_x, b_y → midpoint of object
- b_h, b_w → h, w of object
- c_1, c_2, c_3 → which class object belongs

Logistic Regression loss
Squared error or something similar
Log loss error

Eg:



A = (b_x, b_y)
(0.5, 0.5)

1
0.5
0.5
0.3
0.6
1
0
0

If $P_c = 0$, other all elements are dont care cases

$$L(\hat{y}, y) = \begin{cases} (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 \\ \cdots (\hat{y}_8 - y_8)^2 & \text{If } y = 1 \\ (\hat{y}_1 - y_1)^2 & \text{If } y = 0 \end{cases}$$

$y_1 = P_c$

Face Recognition

- Verification 1 : 1
  Input img, name / ID
  Output whether input image is desired person

- Recognition 1 : K
  Has a database of K person
  Get an input image
  Output ID if the image is any of the K persons

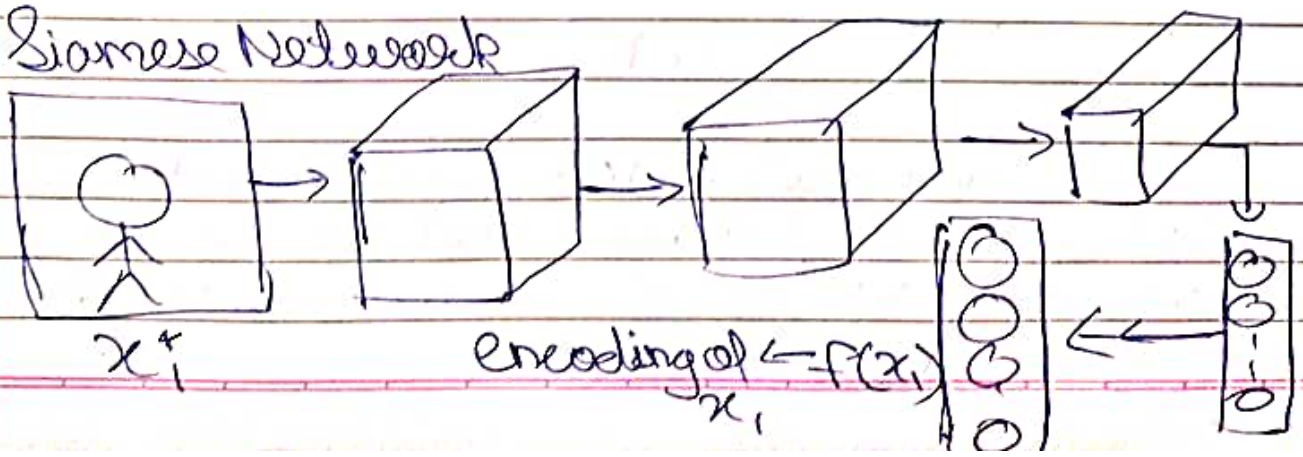
One Shot learning
Deep learning from one example to recognize the person again

- Similarity func
  d(img1, img2) → degree of difference between images

for Verify
If d(img1, img2) ≤ τ (same person)
> τ (Diff person)
τ → Hyper parameter

For face recognition
Compute d for input image with all img in database

Siamese Network



Page No.
Date

Calculate $f(x_1)$ and $f(x_2)$ for two input image x_1 and x_2 then
$d(x_1, x_2) = ||f(x_1) - f(x_2)||^2$
Train NN such that $||f(x_1) - f(x_2)||^2 \leq \tau$ if image are same else greater

NN has same parameters for all images

Triplet loss func

Img 1	Img 2	Img 1	Img 3
Anchor	Positive	Anchor	Negative
A	P	A	N

Want: $||f(A) - f(P)||^2 + \alpha \leq ||f(A) - f(N)||^2$

$\therefore ||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + \alpha \leq 0$

$\alpha \rightarrow$ Margin
α is used so that NN doesn't output zero or same value encoding for each img

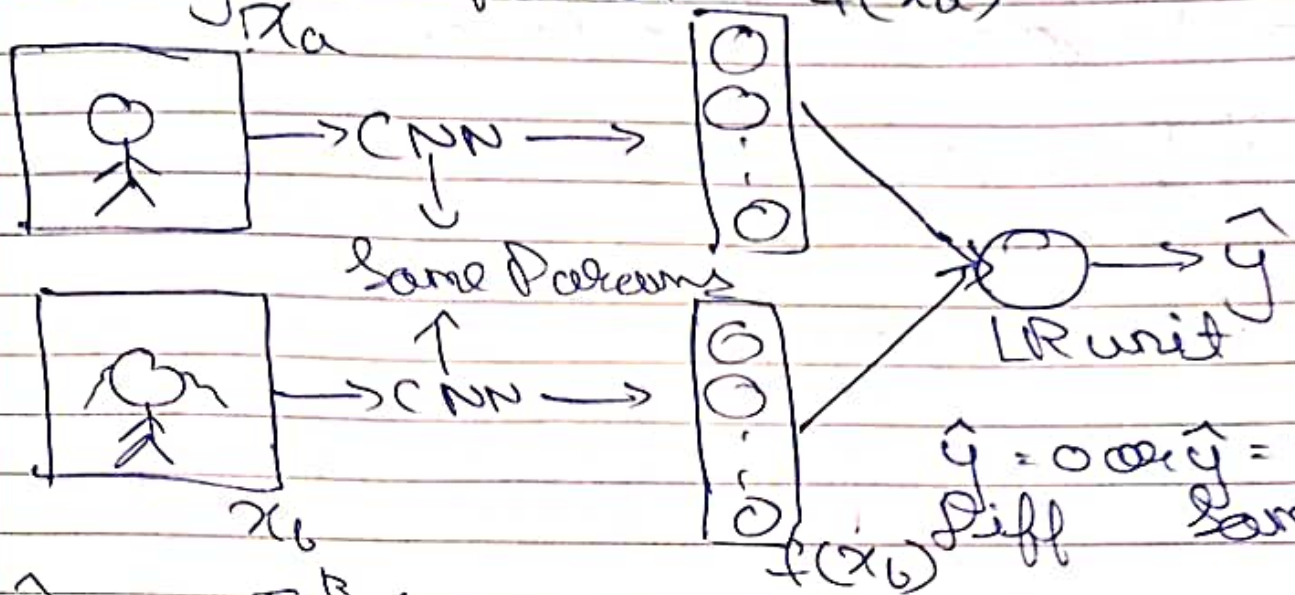
α Pushes (A, P) pair and (A, N) pair further away from each other

$L(A, P, N) = \max(||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + \alpha, 0)$

$J = \sum_{i=1}^{m} L(A_i, P_i, N_i)$

If training set is 10k pics of 1k diff persons - take 10k pics and generate triplets and train NN for gradient descent on cost func

Binary Classification



~~$\hat{y} = \sigma\left[\sum_{i=1}^{k} (w_i | f(x_i)_k = f($~~

$$\hat{y} = \sigma\left[\sum_{i=1}^{k} (w_i | f(x_a)_i - f(x_b)_i | + b)\right]$$

Then do gradient descent

Neural Style Transfer

Content (C) + Style (S)

↓

Generated img (G)

$J(G) = \alpha J_{content}(C, G) + \beta J_{style}(S, G)$

First randomly generate G then

$G = G - \frac{d J(G)}{dG}$

Object Detection

Performance Evaluation Metrics

1] How good is location (IOU)
2] How good is classification (mAP)

- IOU → how close is predicted bounding box to ground truth

$$IOU \rightarrow \frac{Area(A \cap B)}{Area(A \cup B)} \quad [0, 1]$$

- Average Precision

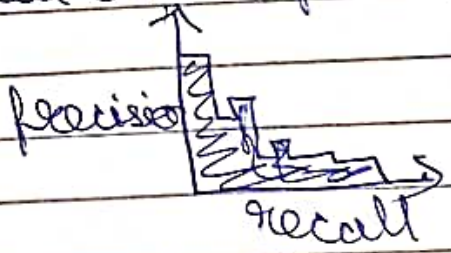
1] Confusion Matrix

predicted \ True	True +ve	False +ve
	False -ve	True neg

2] $Precision = \frac{TP}{TP + FP}$

3] $Recall = \frac{TP}{TP + FN}$

Average Precision → Area under precision recall curve



Average Precision → Single class
Mean Avg Precision → multiple class
mAP

Single Shot Multibox Detector

Page No.
Date

Img → Feature Extract → feature maps → Detection head → Bounding box → NMS → Prediction

1) for Feature Extraction

Img $\xrightarrow[3\text{channel}]{300\times300}$ VGG net $\xrightarrow[512\text{channels}]{38\times38}$ feature maps

VGG net:

Conv3-64 → Conv3-64 → Conv3-128 → Conv3-128 → Conv3-256 (×4) → Conv3-512 (×4) → Conv3-512 (×4) → max pool → FC-4096 → FC-4096 → FC 1000 → Softmax

Feature maps

Conv: 1×1 → 3×3 — 3×3

Conv: 1×1 → 3×3 — 5×5

Conv: 1×1: 256 → 3×3×512 — 10×10

Conv: 3×3×1024 → 1×1×1024 — 19×19

Img → VGG — 38×38 Conv4_3