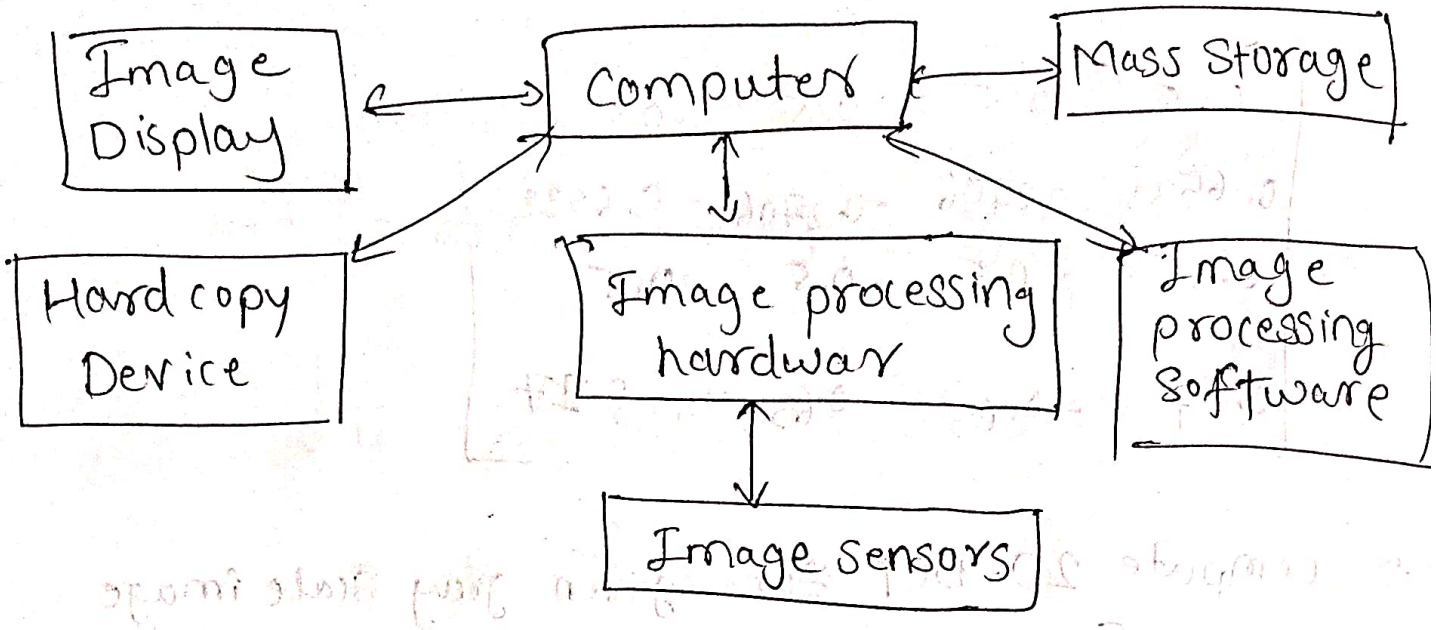
Components of DIP System)



* Hadamard Transform

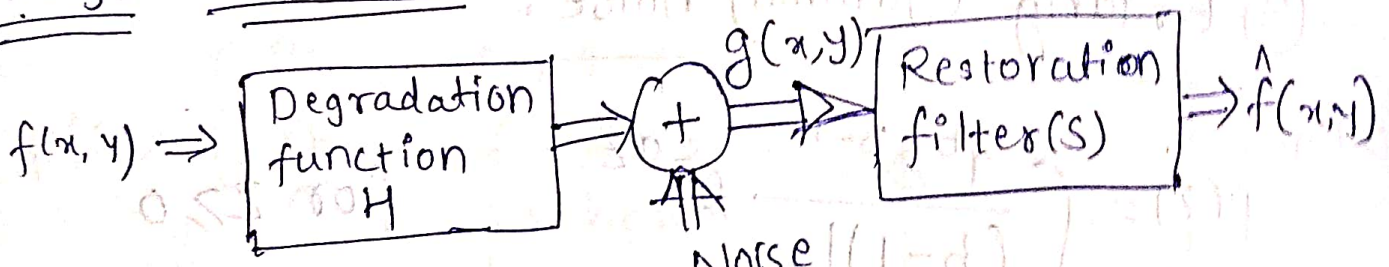
order N=2 $H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

~~$H_{2N} =$~~

$$H_{2N} = \begin{bmatrix} H_N & H_N \\ H_N & -H_N \end{bmatrix}$$

$N = 2$ $H_4 = \begin{bmatrix} H_2 & H_2 \\ +H_2 & -H_2 \end{bmatrix} = \left[\begin{array}{cc:cc} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ \hdashline 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{array}\right]$

Image Restoration



Degradation Restoration

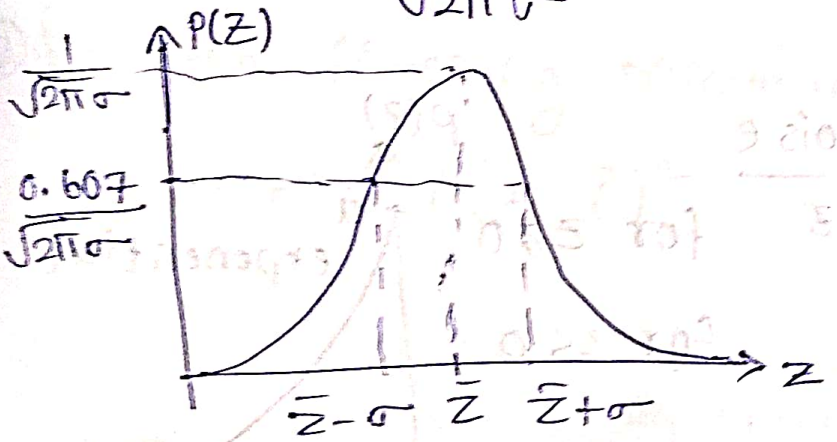
$$g(x,y) = h(x,y) * f(x,y) + \eta(x,y)$$

↓ fourier Transform

$$G(u,v) = H(u,v)\,F(u,v) + N(u,v)$$

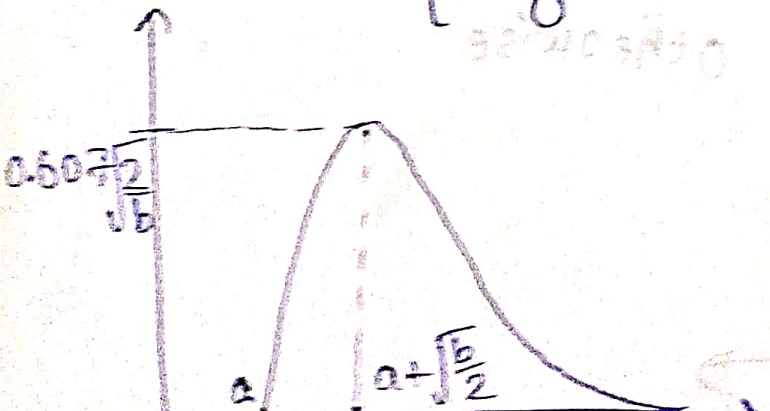
① Gaussian Noise

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(z-\bar{z})^2/2\sigma^2}$$



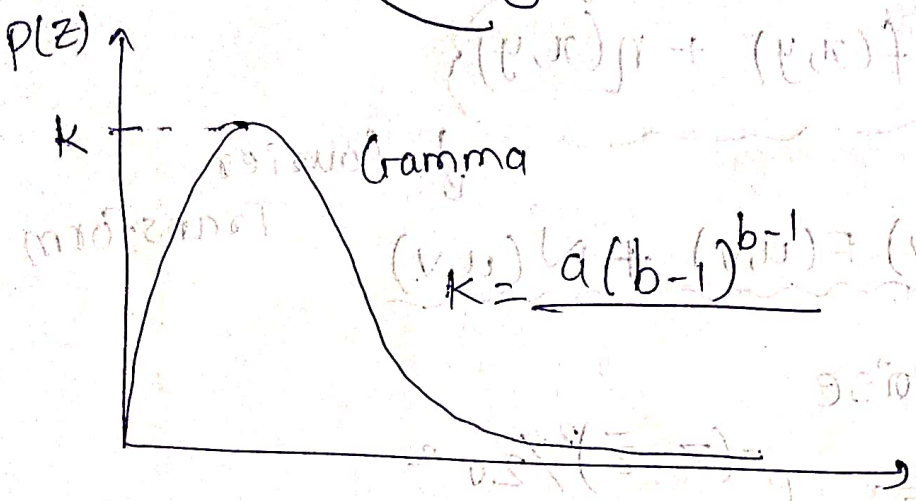
② Rayleigh noise

$$p(z) = \begin{cases} \frac{2}{b}(z-a)\,e^{-(z-a)^2/b} & \text{for } z \geq a \\ 0 & \text{for } z < a \end{cases}$$



③ Erlang (gamma) noise:

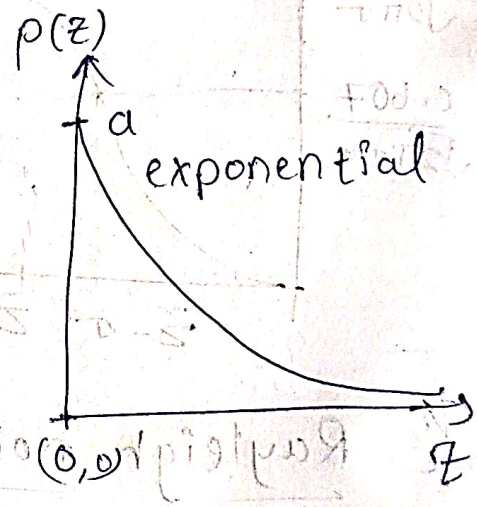
$$p(z) = \begin{cases} \dfrac{a^b z^{b-1}}{(b-1)!} e^{-az} & \text{for } z \geq 0 \\ 0 & \text{for } z < 0 \end{cases}$$



$$k = \frac{a(b-1)^{b-1}}{}$$

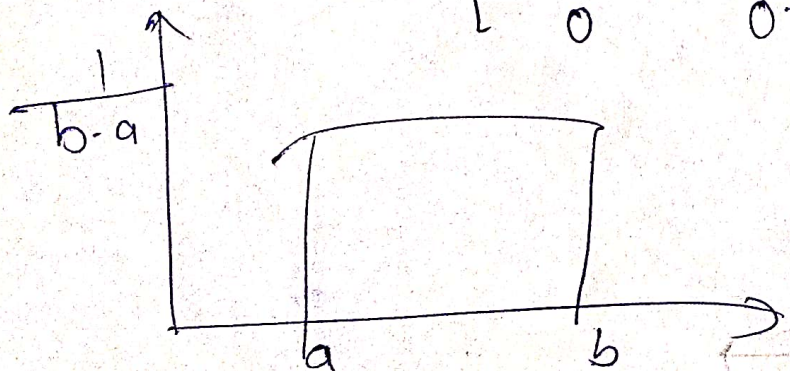
④ Exponential noise

$$p(z) = \begin{cases} ae^{-az} & \text{for } z \geq 0 \\ 0 & \text{for } z < 0 \end{cases}$$



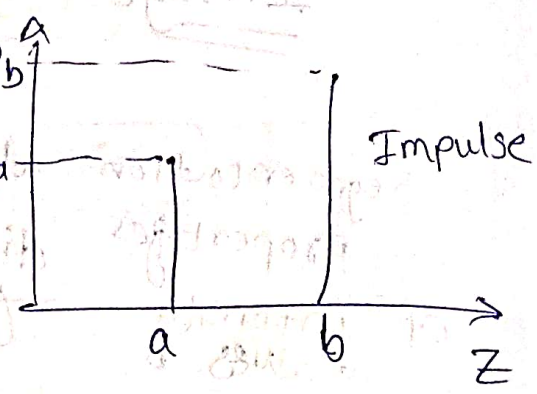
⑤ Uniform Noise

$$p(z) = \begin{cases} \dfrac{1}{b-a} & \text{if } a \leq z \leq b \\ 0 & \text{otherwise} \end{cases}$$



⑥ Impulse (Salt & pepper) noise

$$p(z) \begin{cases} P_a & \text{for } z=a \\ P_b & \text{for } z=b \\ 0 & \text{otherwise} \end{cases}$$



* To Remove Noise

① Inverse filtering

$$\hat{F}(u,v) = \frac{G(u,v)}{H(u,v)}$$

$$\hat{F}(u,v) = F(u,v) + \frac{N(u,v)}{H(u,v)}$$

② (Wiener) filtering Minimum Mean Square error

$$e^2 = E(f-\hat{f})^2$$

Image Segmentation

Sobel operator

Segmentation algorithms based on on two basic properties of intensity values

discontinuity & Similarity

↳ sudden colour change

e.g. car, car

* ~~Discon~~ Detection of Discontinuties

Point detection

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Mask

×

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

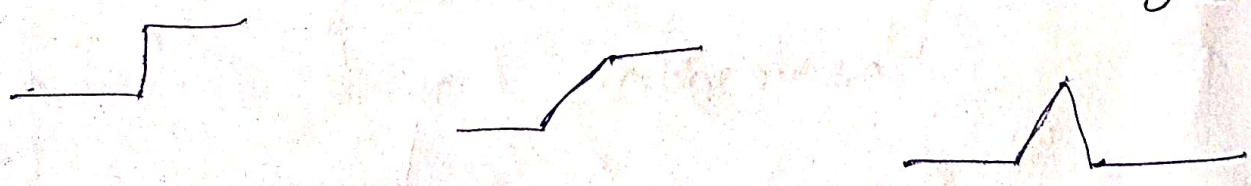
$$R_K = \sum_{i=1}^{9} w_i z_i$$

Response of the mask.

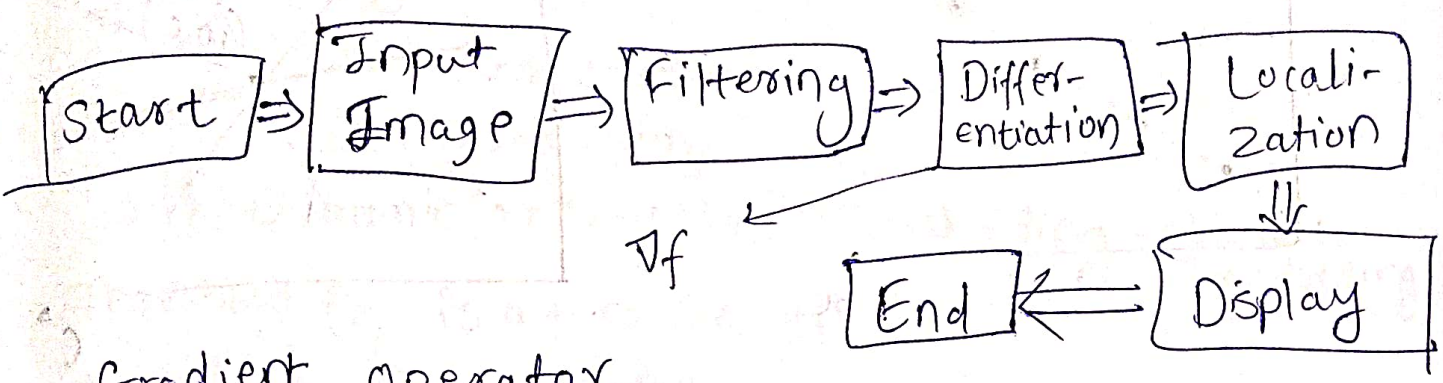
Edge Detection

Edge models:

① Step edge ② ramp edge ③ roof edge



Stages in Edge Detection



Gradient operator.

~~$\nabla f = f$~~

* Laplacian of Gaussian (LoG) Operator.

$h(r) = e$

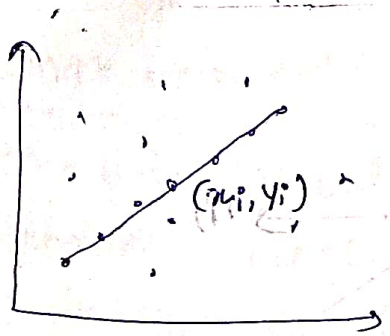
$\nabla h(r)$

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

(a)

* Edge linking &
  Boundary Detection

* HOUGH Transform. → solution to detect simple shapes as circles, lines, etc.



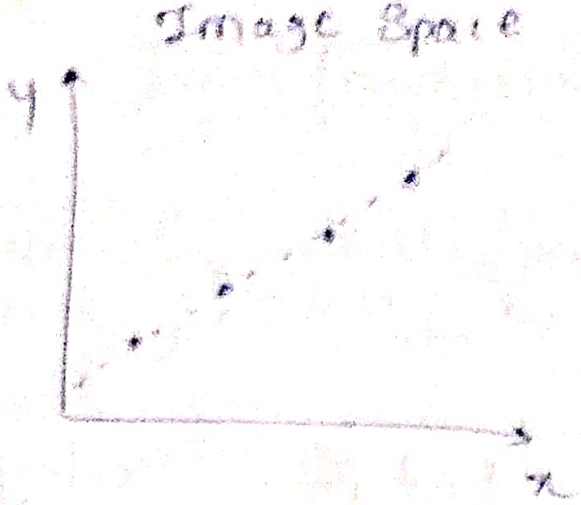
$y_i = mx_i + c$

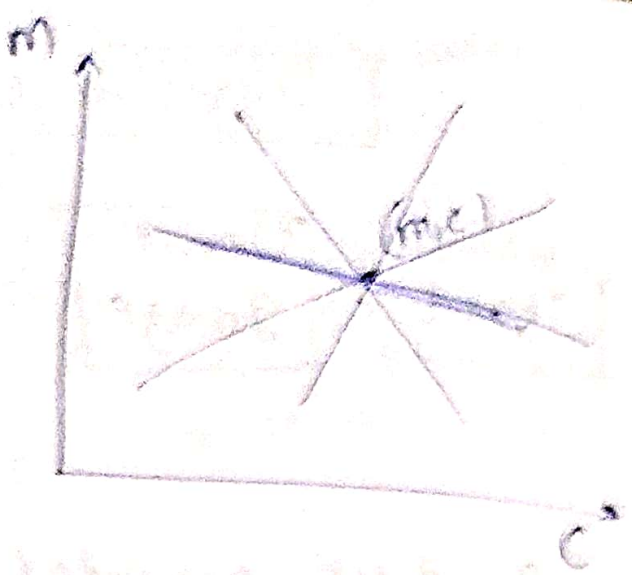
$\Rightarrow c = -mx_i + y_i$

Given: Edge point (x_i, y_i)

Task: Detect line

$y = mx + c$





Ex 1 Given five points use Hough transform to draw a line joining these points.

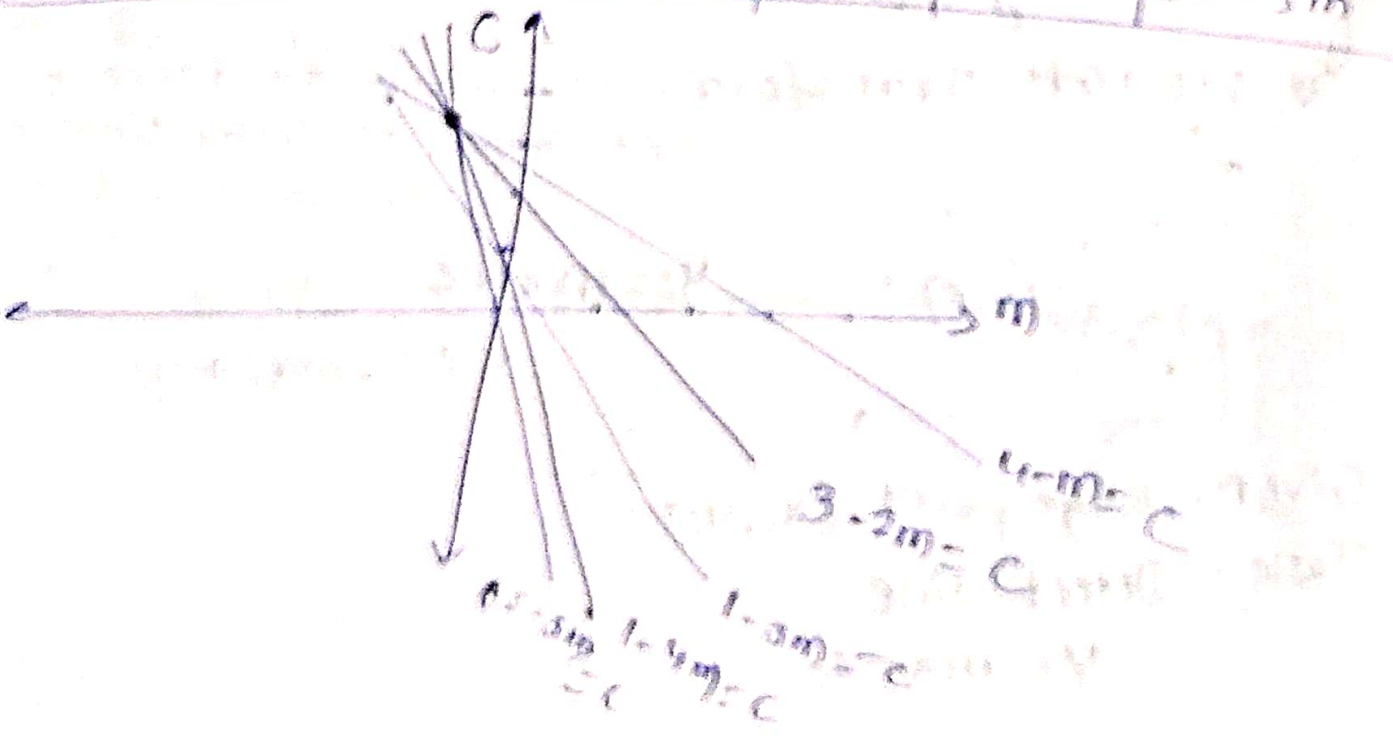
(1,4) (2,3) (3,1) (4,1) (5,0)

X	1	2	3	4	5
Y	4	3	1	1	0

Equation of straight line → $y = mx + c$

$\therefore c = y - mx$

X	1	2	3	4	5
Y	4	3	1	1	0
Eq of lines	$c = 4 - m$	$c = 3 - 2m$	$c = 1 - 3m$	$c = 1 - 4m$	$c = -5m$



% HW Circle detection

Thresholding

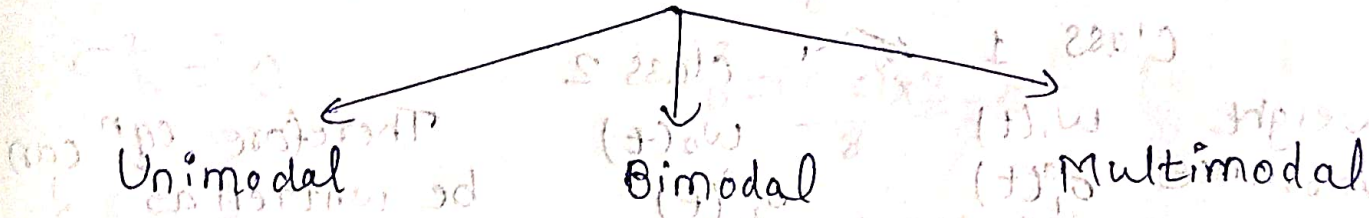
(1) Global Thresholding
(2) Local Thresholding
(3) Dynamic (or Adaptive) Thresholding → Variable Thresholding

Thresholding is a function of:
> Spatial Co-ordinates
> Grey level of the pixel
> Some local property of the image i.e A(x,y)

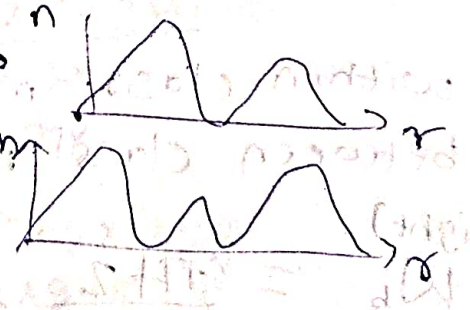
$$Th = T[x, y, f(x,y), A(x,y)]$$

Histogram & Thresholding

Types of Histograms



Single Level Thresholding →
Multi level Thresholding →



Which threshold to select

The basic global Threshold (T) is calculated as;

1) Random select T.
2) Two segment. G_1, G_2
3) Determine mean (m_1) in (G_1) lie below T in histogram
4) Determine mean (m_2) of the pixels (G_2) lie below T
5) New threshold is: $T_{new} = (m_1 + m_2)/2$
6) Select min T

* Mutiple thresholding

$$\text{output image} = \begin{cases} g_1 & \text{if } 0 \le f_i \le t_1 \\ g_2 & \text{if } t_1 \le f_i < t_2 \\ \vdots & \\ g_n & \text{if } t_{n-1} \le f_i \le 255 \end{cases}$$

* Adaptive Thresholding

Techniques to apply Thresholding algorithms

* Optimal Otsu Thresholding Algorithm:

usea merit function either maximization or minimization to determine the optimal threshold.

Class 1 ↔ Class 2

weight $w_1(t)$ $w_2(t)$

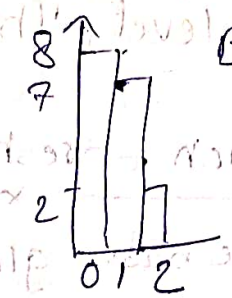
Variance $\sigma_1^2(t)$ $\sigma_2^2(t)$

Therefore eqn can be written as

$$\sigma_b^2(t) = \sigma^2 - \sigma_w^2(t)$$

within class minimization

between class maximization.



(weight)

$$W_b = \frac{8+7+2}{36} = 0.4722$$

$$\text{Mean } (\mu_b) = \frac{(0\times8)+(1\times7)+(2\times2)}{17} = 0.6471$$

$$\text{Variane } (\sigma_b^2) = \frac{(0-0.6471)^2\times8 + (1-0.6471)^2\times7 + (2-0.647)^2\times2}{17}$$

for Background

Within class variance

$$\sigma_w^2 = w_b \sigma_b^2 + w_f \sigma_f^2$$

Verify

T=0

$$w_b = \frac{0}{36} = 0$$

Mean → $M_b = 0$

$$\sigma_b^2 = 0$$

$$w_f = 1$$

$$M_f = \frac{0\times8 + 1\times7 + 2\times2 + 3\times6 + 4\times9 + 5\times4}{36}$$

$$M_f = \frac{11+18+45+20}{36}$$

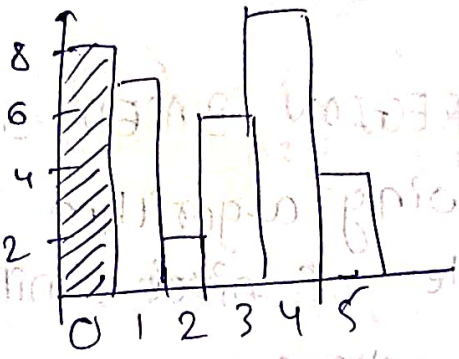
$$M_f = 2.36$$

$$\sigma_f^2 = \frac{(0-2.36)^2\times8 + (1-2.36)^2\times7 + \cdots + (5-2.36)^2\times4}{36}$$

$$\sigma_f^2 = 3.1196$$

$$\sigma_w^2 = w_b \sigma_b^2 + w_f \sigma_f^2 = 3.1196$$

T=1



$$w_b = \frac{8}{36} = 0.222$$

$$M_b = \frac{0\times8}{8} = 0$$

$$\sigma_b^2 = \frac{(0-0)^2\times8}{8} = 0$$

$$w_f = \frac{7+2+6+9+4}{36}$$

$$w_f = 0.778$$

$$M_f = \frac{1\times7 + 2\times2 + 3\times6 + 4\times9 + 5\times4}{28}$$

$$M_f = 3.036$$

$$\sigma_f^2 = \frac{(1-3.036)^2\times7 + (2-3.036)^2\times2 + \cdots + (5-3.036)^2\times4}{28}$$

$$\sigma_f^2 = 1.9639$$

$$\sigma_w^2 = 1.5268$$

Between class Variance.

$$= W_b W_f (\mu_b - \mu_f)^2$$

Without using built in function Sun, Night

Write program for Otsu threshold method for any image

REGION BASED SEGMENTATION.

* Region- Growing algorithm
  - (i) principle of ~~pixel~~ similarty.
  - (ii) region is coherent → pixels of group homogeneous.

3 - steps:

(i) Select the intial seed
(ii) How seed growing
(iii) Termination of segmentation process.

$S_1 = 9$ $S_2 = 1$, $T = 4$ → Threshold

* $S_1 = 9$

$$|f(x,y) - f(x',y')| \leq 4$$

$$f(x,y) = \{5,6,7,8,9,10,11,12,13\} \rightarrow A$$

* $S_2 = 1$ $|f(x,y) - f(x',y')| \leq 4$

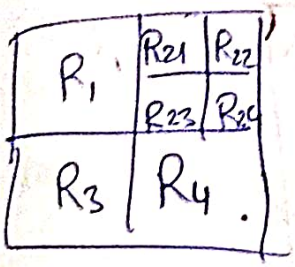
$$f(x,y) = \{0,1,2,3,4,5\} \rightarrow B$$

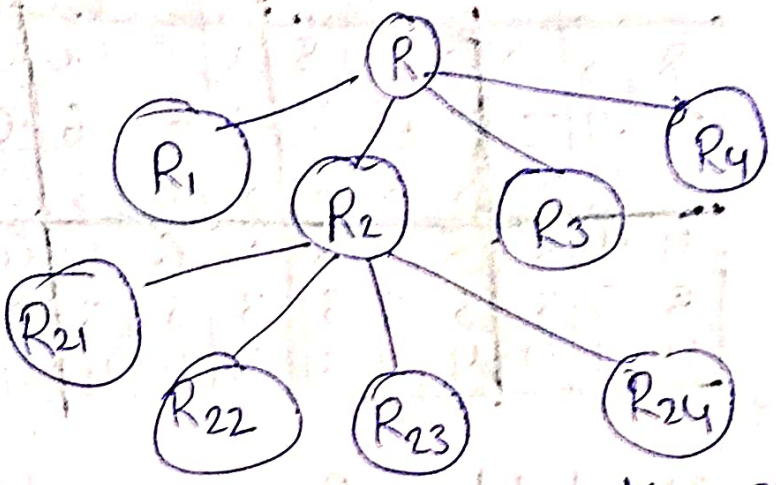
1	0	7	8	7
0	1	8	9	8
0	0	7	9	8
0	1	8	8	9
1	2	8	8	9

→

* Split & Merge Algorithm

phase-I → Splitting , Phase II → Merging
alternative method of image segmentation





Quadtree Splitting

If adjacent regions are similar merge

Note: Inconsisten sizes will lead to inaccurate segmentation.

Ex

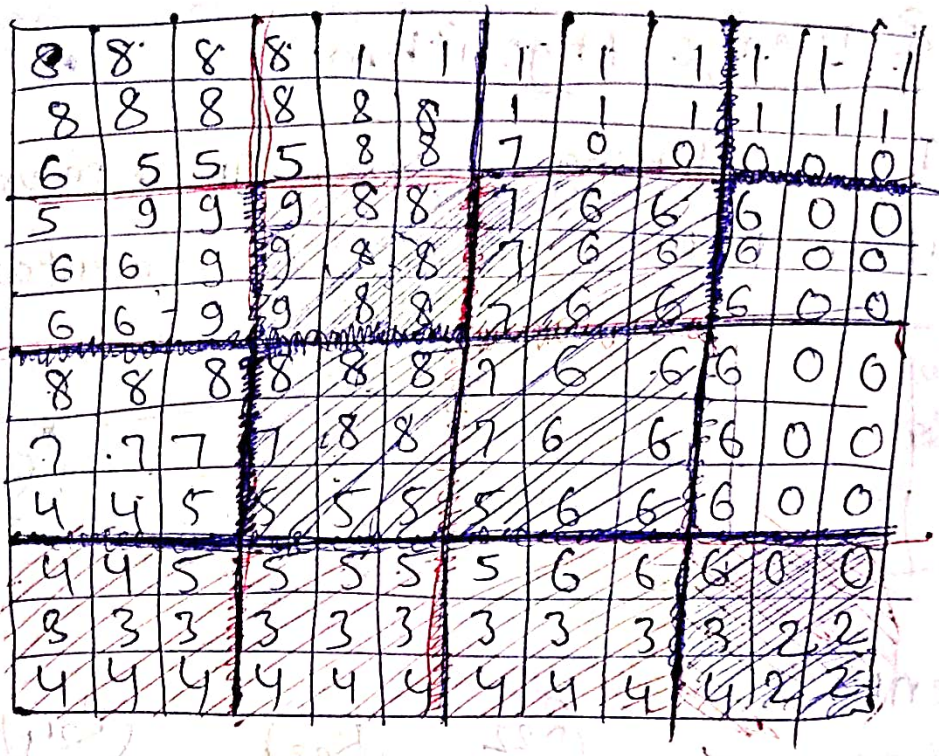
6	5	6	6	7	7	6	6
6	7	6	7	5	5	4	7
6	6	4	4	3	2	5	6
5	4	5	4	2	3	4	6
0	3	2	3	3	2	4	7
0	0	0	0	2	2	5	6
1	1	0	1	0	3	4	4
1	0	1	0	2	3	5	4

Threshold = 3

Max - Min = 7 - 0 = 7

~~skip~~ split.

~~Split~~



Let Threshold = 3

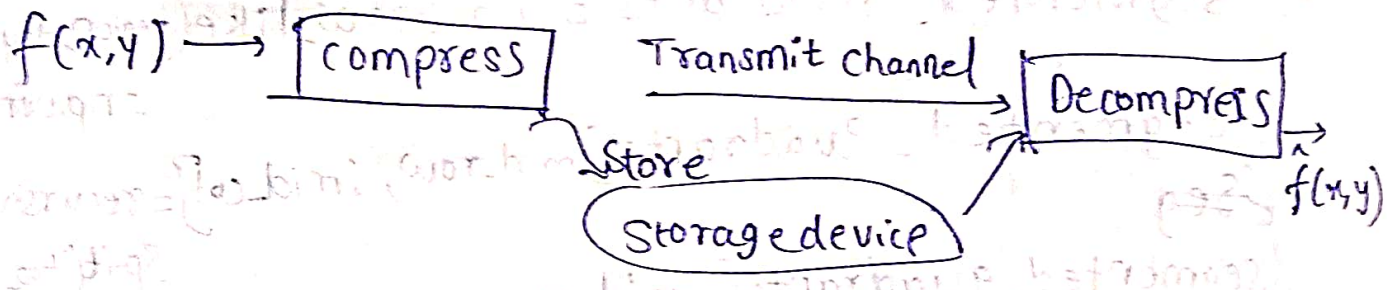
Max - min = 9 - 0 = 9 ⇒ 9 > 3 ∴ Split

```
import cv2
import numpy as np
import requests.

def is_homogeneous(region, threshold):
    min_val, max_val = np.min(region), np.max(region)
    return (max_val - min_val) <= threshold.

def ~~is~~split_and_merge(image, threshold):
    def recursive_split(region):
        rows, cols = region.shape
        if row <= 1 or cols <= 1:
            return np.zeros_like(region, dtype=np.uint8)
        if is_homogeneous(region, threshold):
            return np.ones_like(region, dtype=np.uint8)
```

```
mid_row, mid_col = rows//2 , cols//2
top_left = region[:mid_row , :mid_col]
top_right = region [: mid_row , mid_col: ]
bottom_left = region[mid_row: , :mid_col ]
bottom_right = region [mid_row: , mid_col: ]

segmented_quadrants = np.zeros_like(region, dtype
                                     =np.uint8)
segmented_quadrants[:mid_row, :mid_col] = recursive_
                                     split(top_left)
segmented_quadrants[:mid_row, mid_col:] = recursive_split
  "          "      [mid_row: , :mid_col] = "   (top_right)
  "          "      [  "      , mid_col:] = "   (bottom_left)
                                                (bottom_right)
def merge_regions(segmented):
    return segmented

if len(image.shape) == 3:
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

segmented_image = recursive_split(image)
segmented_image = merge_regions(segmented_image)
return segmented image
def main():
    image = cv2.imread('E:NITW
    threshold = 20
    result = split_and_merge(image, threshold)
    cv2.imshow('Segmented Image', result*255)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

```
__name__ = "__main__";
main()
```

Compression — The goal of image compression is to reduce the amount of data required to represent a digital Image



- Lossless
- Lossy

⇒ Interlaced Scanning (1080i) ⇒ Splits frames into odd & even fields

⇒ Progressive Scanning (1080p) ⇒ Scans the entire picture line by line in sequential order

Standard Definition — High Definition.

⇒ DATA ≠ Information

Compression Ratio: $C_R = \frac{n_1}{n_2}$

n_1 bits $\xrightarrow{\text{compression}}$ n_2 ;

For ex — $C_R = \frac{10}{1} = 10:1$

Data	Info	Data
00000000	0	00
00000001	1	01
00000010	2	10
00000011	3	11

Relevant Data Redundancy

$R_D = 1 - \frac{1}{C_R}$

If $C_R = \frac{10}{1}$, then $R_D = 1 - \frac{1}{10}$

$= 0.9$

Data

00000000
00000001
00000010
00000011

Redundant.

Type of DATA Redundancy.

1. coding redundancy (Symbol coder)
2. Spatial & Temporal redundancy (Mapper)
3. Irrelevant Information (Quantizer)

1. Coding Redundancy.

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k) P_r(r_k)$$

I/P intensity values r_k

*(Huffman coding, Golomb coding, etc.)

2. Spatial & Temporal Redundancy:

100 × 8 + 100 × 8 = 1600 bits for 2 lines

(5, 100) (255, 100)

8 8 8 8 = 32 bits required now.

→ Spatial

⇒ Temporal redundancy

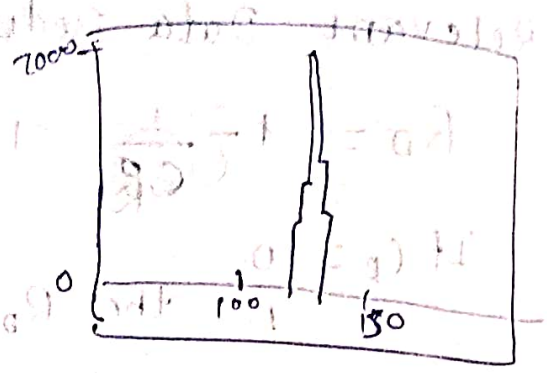
:- Similarity blw neighbouring frames in a video.

3. Irrelevant Information:

129, (130) 131, 132

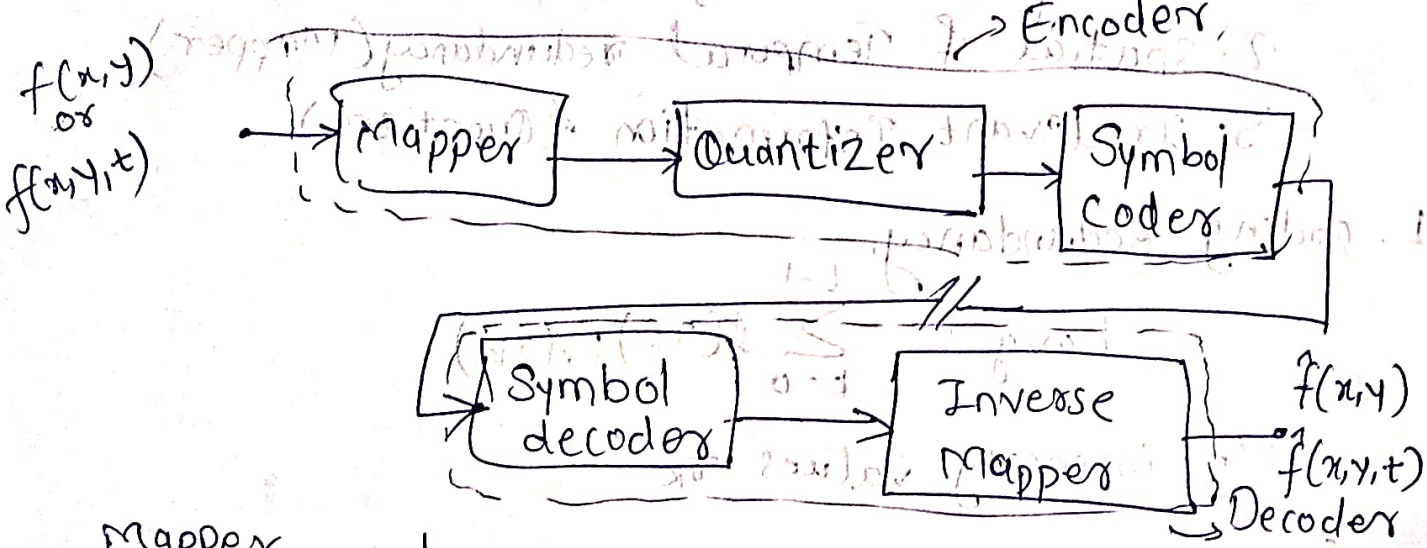
↓

representing using 130



Histogram

Image Compression Model



Mapper reduce Spatial & Temporal redundancy.

⇒ A "codec" is a device or program that is capable of both encoding & decoding.

Taxonomy of lossless methods

Lossless coding Techniques

Repetitive Sequence Encoding

Statistical Encoding

* Huffman Coding

Same pixel-values

1) Sort probabilities per symbol
2) Combine the lowest two probabilities
3. Repeat Step2 until only two probabilities remain.

Problem:

10x10 (3 bit image)

Frequency:

$a_2=40$ $a_6=30$ $a_1=10$ $a_4=10$ $a_3=6$ $a_5=4$

$P(a_2) = 40/100 = 0.4$

Symbol like intensity levels	Probabilities (Sorted)		Source reduction 1	2	3	4
a_2	0.4	1	0.4	0.4	0.4	0.6 0
a_6	0.3	00	0.3	0.3	0.3	0.4 1
a_1	0.1	011	0.1	0.2 010	0.3 01	
a_4	0.1	0100	0.1	0.1 011		
a_3	0.06	01010	0.1			
a_5	0.04	01011				

Entropy $H(x) = -\sum_x P(x).\log_2 P(x) = \sum_x p(x).\log_2 \frac{1}{P(x)}$

$$\text{Your Saved} = \frac{10\times10\times5 - 10\times10\times2.2}{10\times10\times5} = 0.56$$

7	7	3	1	1
6	2	3	1	1
4	3	0	0	7
3	4	3	3	4
5	5	6	2	2

Symbol	Frequency	Probability (Frequency/25)
0	2	0.08
1	4	0.16
2	3	0.12
3	6	0.24
4	3	0.12
5	2	0.08
6	2	0.08
7	3	0.12

Code	Symbol (like intensity levels)	Code	Probabilities (Sorted)	Source reduction					
001	3		0.24	0.24	0.24	0.24	0.32	0.44	0.56
000	1	000	0.16	0.16	0.20	0.24	0.24	0.32	0.44
100	2	100	0.12	0.16	0.16	0.20	0.24	0.24	
101	4	101	0.12	0.12	0.16	0.16	0.20		
110	7	110	0.12	0.12	0.12	0.16			
111	0	111	0.08	0.12	0.12				
0010	5	0010	0.08	0.08					
0011	6	0011	0.08						

Codes written beside the reductions: 01, 10, 11, 000, 001, 100, 101, 110, 111, 0010, 0001

3	0.24
1	
2	
4	
7	
0	
5	
6	

123BBBBCAADDDCCAADEFGGZZACACAC

A → 7 B → 4 C → 6 D → 4 E → 1 F → 1

G → 2 Z → 2 1 → 1, 2 → 1, 3 → 1

4

6

(Symbol like intensity levels) | Probabilities (Sorted)

01	A	7	0.233	0.233	0.233	0.233	0.233	0.233	0.266
00	C	6	0.20	0.2	0.2	0.2	0.2	0.2	0.233
110	B	4	0.133	0.133	0.133	0.133	0.133	0.167	0.2
101	D	4	0.133	0.133	0.133	0.133	0.133	0.133	0.167
1110	G	2	0.067	0.067	0.067	0.1	0.133	0.133	0.133
1001	Z	2	0.067	0.067	0067	0.067	0.067	0.133	010
~~1111~~	E (10001)	1	0.033	0.067	0.067	0.067	0.067	0000	011
~~11111~~	F (10000)	1	0.033	0.033	0.067	0.067	0110	0001	
~~111110~~	1 (10010)	1	0.033	0.033	0.033	00000	0111		
~~100001~~	2 (111111)	1	0.033	0.033	00001				
111110	3 (111110)	1	0.033	000000					
			0.033	000001					
			01110						
			01111						

```
class NodeTree(object):
    def __init__(self, left=None, right=None):
        self.left = left
        self.right = right
    def children(self):
        return (self.left, self.right)
```

A 10
C
B
D
G
Z
E
F

1 000001
2 01110
3 01111

0.3
0.266
0.233
0.2
000
001

0.433 0.566 0
0.3 0.433 1
0.266 01
10
11

ARITHMET

Main Idea → Seq

Arithmethic coding

Encode the

Symbols	e	
Probability	0.3	0.

Find the code wo

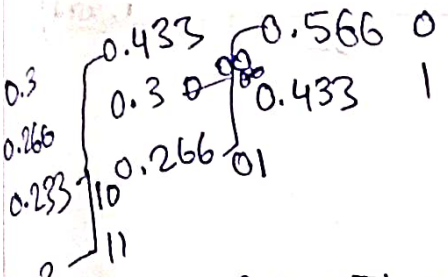
Sol:

Step 1

1
0.9 w
0.8 t
0.6 n
0.3 e

d = UB
Range of
lower limit = lower

Range of
Range of
Range of t
Range of
Range o



ARITHMETIC (OR RANGE) coding

Main Idea → sequence mapped to → 1, 0.6457, 0

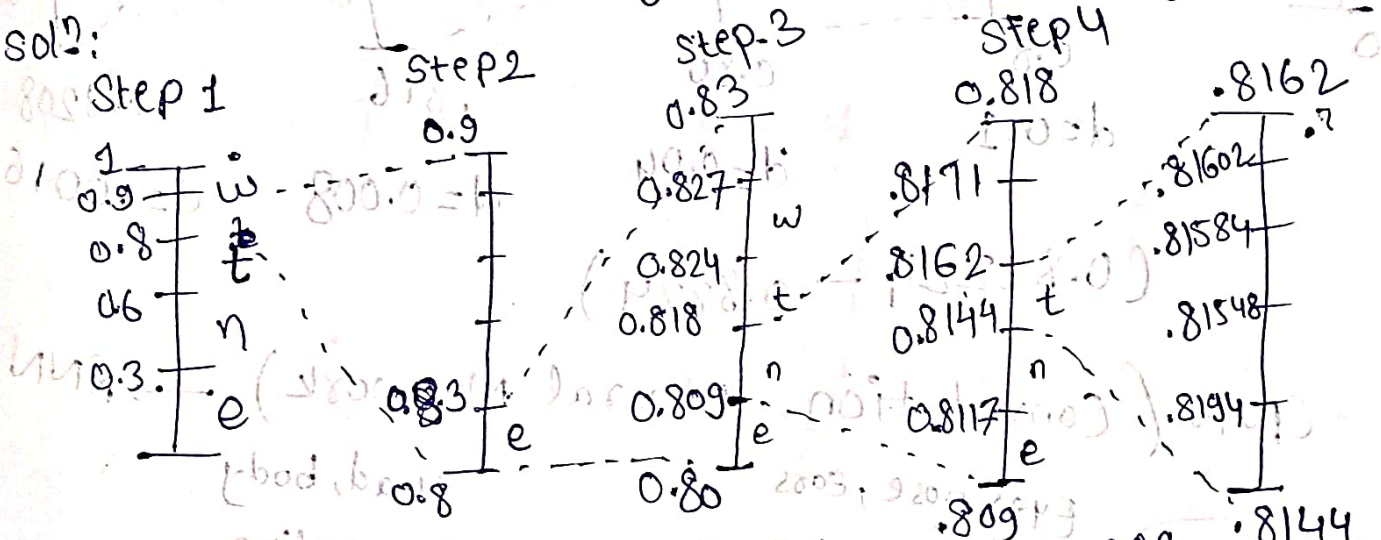
Arithmethic coding Problem:

Encode the message "went."

Symbols	e	n	t	w	.
Probability	0.3	0.3	0.2	0.1	0.1

Find the code words & generation of the Tag.

Sol?:



d = UB − LB = 0.9 − 0.8 = 0.1, d = .009, d = .0018

Range of symbol

lower limit = lower limit + d(probability of symbol)

Range of e = 0.83 : 0.83 = 0.8 : [0.8 + 0.1(0.3)]

Range of n = 0.83 : 0.86

Range of t = 0.86 : 0.88

Range of w = 0.88 : 0.89

Range of . = 0.89 : 0.90

62
−44
.0018

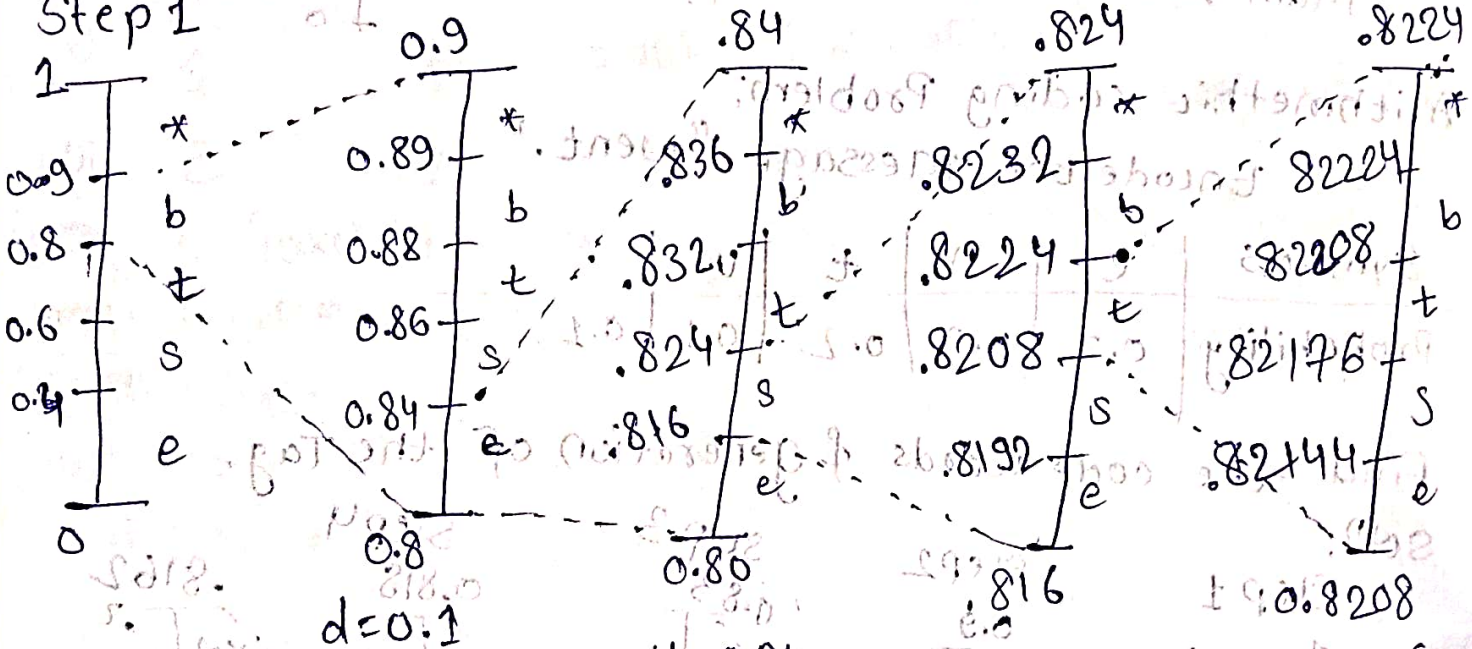
0.81602 < codeword < 0.8162

Generation of Tag : Tag = $\frac{(\text{UL of code word} + \text{LL of code word})}{2}$

Find codeword for best*

e	S	t	b	*
0.4	0.2	0.2	0.1	0.1

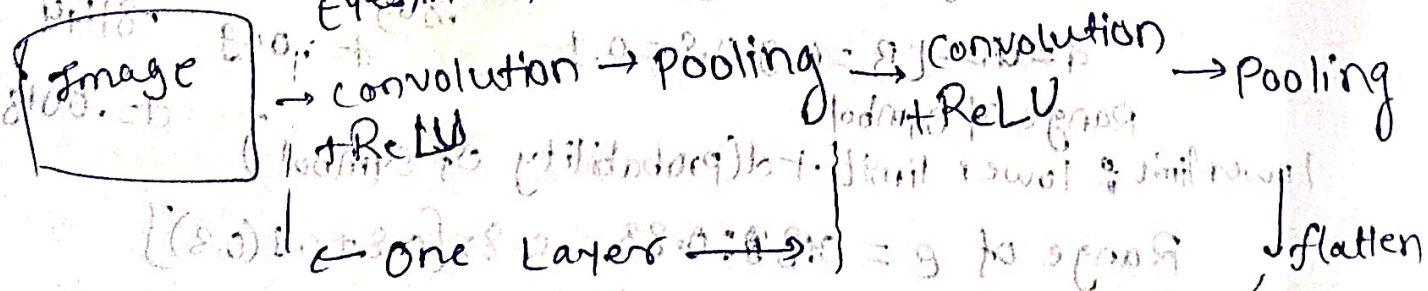
Step 1

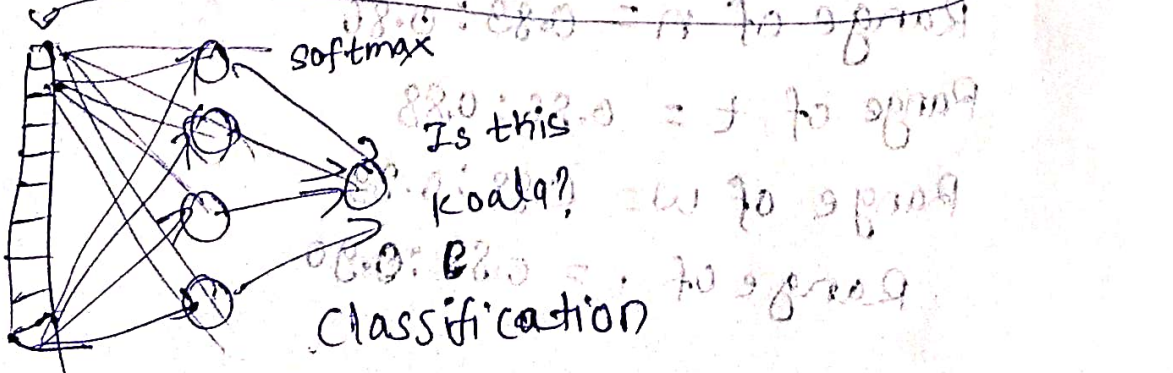


(0.82224 — 0.8224)

CNN (Convolution Neural Network) → RNN

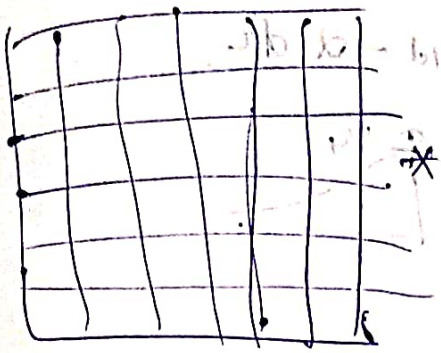
Eyes, nose, ears — Head, body





convolution

Input



Filter

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

Result

2	6	-4	5
0	4	0	-1
-5	0	3	6
-2	5	0	3

parameters:

6x6 il size, f=3

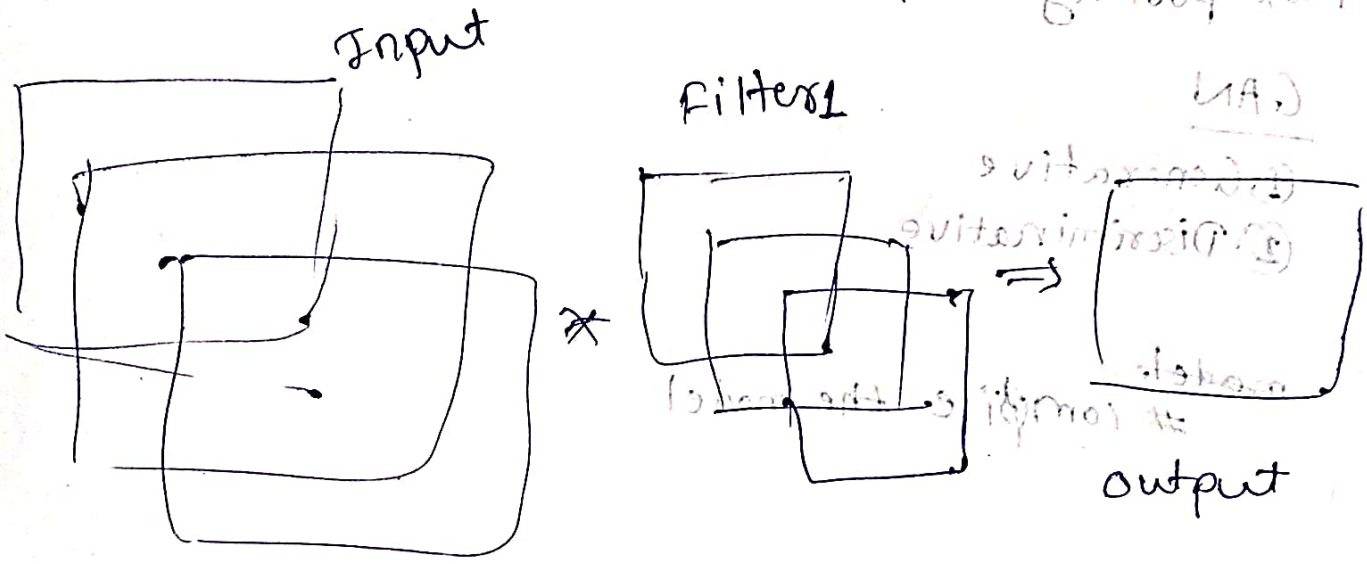
stride S=1

padding P=0 $p=(f-1)/2$

$$O = \left\lfloor \frac{n-f+2P}{S} + 1 \right\rfloor$$

$\lfloor\ \rfloor$: floor()

* convolution Operation on Volume.



Multiple filters

* ONE CONVOLUTION LAYER

ReLU{output + b}

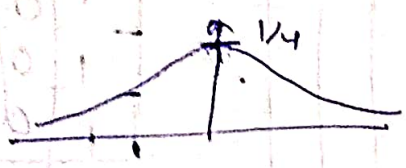
ReLU ⇒ max(0, x)

Sample Complete Network

* Vanishing gradient

1 layer → .1/4
4 " → $(\frac{1}{4})^4$
1 " → $(\frac{1}{4})^L$

$W_{new} = W_{old} - \alpha \cdot \frac{dL}{}$



3. ReLU ~~(Rectilinear Unit)~~ (rectified linear)

$g(x) = Relu(x) = max(0, x)$

4. ~~P~~ Leaky Relu

Pooling Layer :-

> It has hyper-parameters : mask size(f), stride(s)

Max pooling.

GAN

① Generative
② Discriminative.

- ~~model~~
  # compile the model