DIP

Ch - 8 Image Compression

Image compression Model

→ Image compression is very imp table in img processing

→ Images & video requires lot of space & large transmission time

→ Data compression is the process of encoding data so that it takes less storage space & less transmission time than it was not compressed

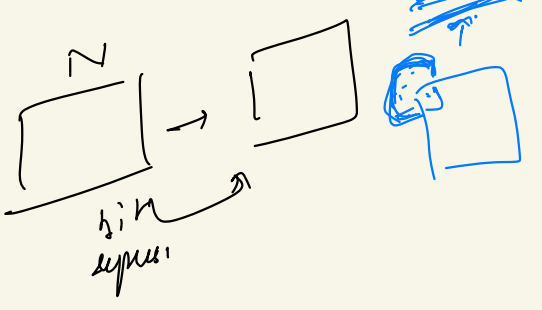
→ Mathematical process of transforming data to smaller representation

Original N_1 → Compressed N_2

$N_1 = = N_2$ no compression

$N_2 << N_1$ heavy compression

$N_2 >> N_1$ Data explosion ✗



perceptual quality
↓
content of image
↓
objects remain intact

Data & information

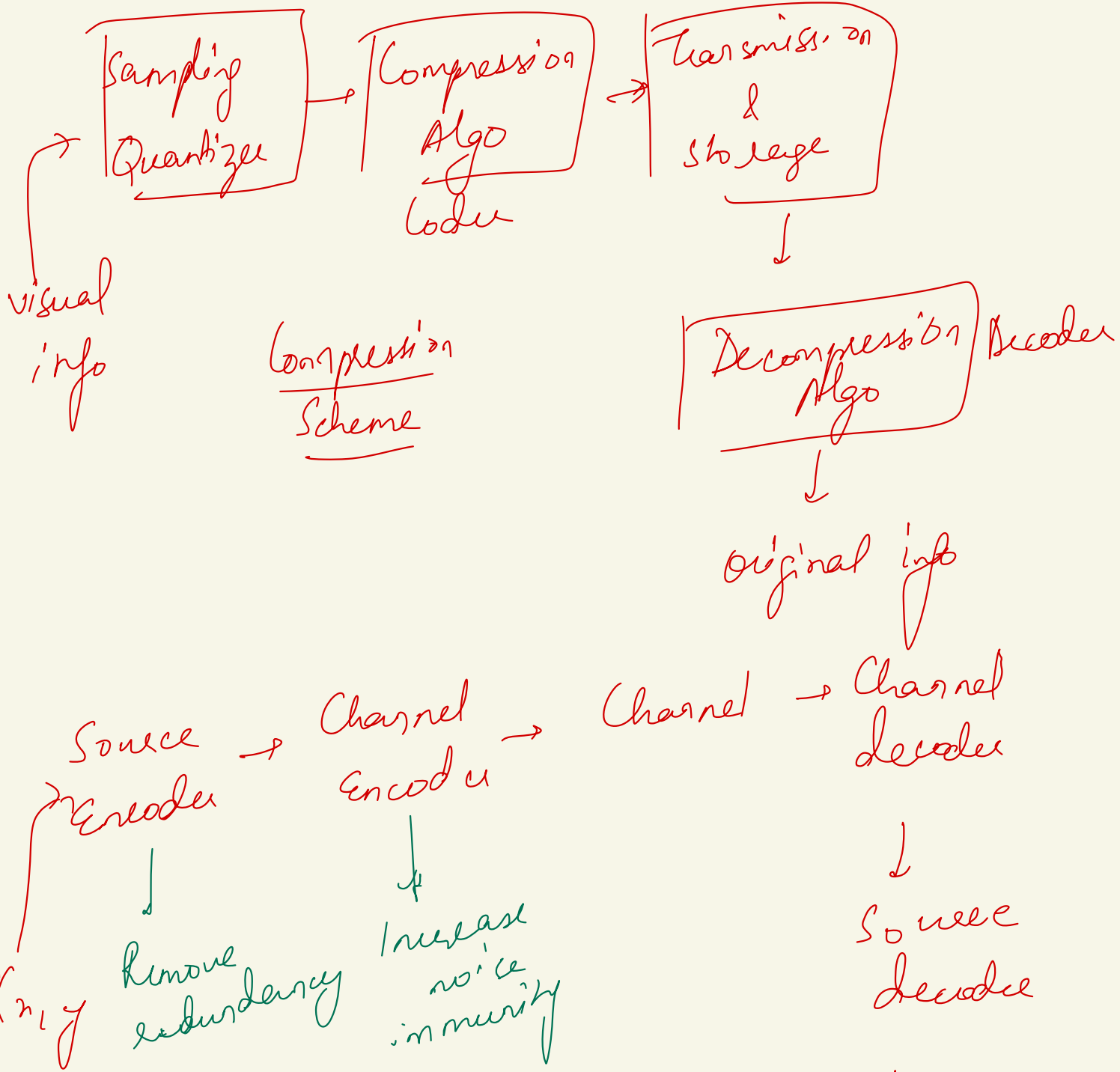
↓ raw facts

↓ interpretation of data in meaningful way

Types of Data
- → Text
- → Binary data
- → Images
- → Graphics - vector notation
- → audio
- → video

Why compression?
- → Storage
- → Transmission ease
- → faster computation

Applications - fax, voice mail, telephony
- → over Internet
- → multimedia
- → Image & signal processing
- → Video conferencing
- → Digital & Satellite TV


Sampling
Quantizer
Compression
Algo
Coder
Transmission
&
Storage
visual
info
Compression
Scheme
Decompression
Algo
Decoder
original info
Source
Encoder
Channel
encoder
Channel
Channel
decoder
f(x,y)
Remove
redundancy
Increase
noise
immunity
Source
decoder
$\hat{f}(x,y)$
If disk is noise free → channel encoder & decoder can be removed

Stages of encoding

$f(x,y)$ → Mapper → Quantizer → Symbol encoder → $\hat{f}(x,y)$ → channel

↓

* Reduces interpixel redundancy
* Reversible operation

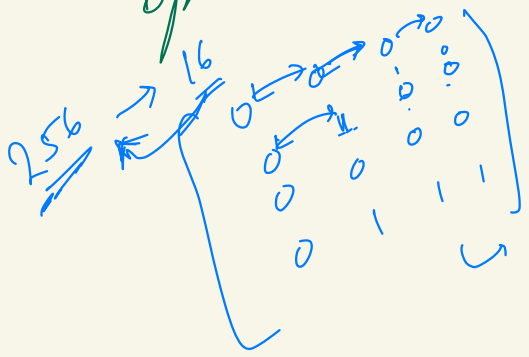
↓

* Psychovisual redundancy
* not reversible operation

↓

make a pixel or variable length code

* Reversible operation



Source decoder

channel → Source decode → Inverse Mapper → $f(x,y)$

Nobel & Lucid / Lossless principles

Compression Algos can be visualized as - the mapping of as set of message symbols to codes using some logical rules of conversation

Eg: JPEG
Joint Photographic Expert Group

→ logical compression
→ Physical compression

pixels or block → set of pixels
code → sequence of symbols / numbers
code word → string of codes

Original N_1 → Compressed N_2

Compression ratio (CR) $= \frac{N_1}{N_2}$

1) $N_1 = N_2$

$CR = N_1/N_2 = 1$

Relative Redundancy $R_D = 1 - \frac{1}{CR} = 0$

I/P = O/P

4) $CR = \frac{N_1}{N_2} = \infty$

$R_D = 1$

Compression is done

3) $N_1 << N_2$

Reverse Compression

$CR = \frac{N_1}{N_2} \to 0$

$R_D = 1 - \frac{1}{0} = \infty$

Data in transformed is much higher than original

Measures:

Metrics

1) Compression Ratio = $\frac{\text{Uncompressed file size}}{\text{compressed file size}} = N_1/N_2$

2) Saving Percentage

$SP = 1 - \frac{\text{compressed file size}}{\text{Uncompressed file size}} = 1 - \frac{N_2}{N_1}$

3) Bit rate = $\frac{\text{Size of compressed file}}{\text{No of pixels in image}}$

Compression Algos & its types

→ Objective of compression algo: reduce the source data to a compressed form & decompress it to get original data

2 components - i) Modeller ii) used to condition the data for compression using knowledge of data.

2) Present in both sender & receiver

3) can be static or dynamic

ii) Coder : 1) Sender side coder → encoder
2) Codes the symbols independently
er using the model
3) Reciever side coder → decoder
4) Decoder decodes the message from
compressed data

Static : do not change

Dynamic : models change depending upon
the change of data

→ Symmetric model scheme : models at
sender & reciever side are same

→ Asymetric : different models at
sender & reciever

Compression Algos

Lossless compression

→ Reconstructed data is identical to original data (Entropy coding)

→ Examples:
- Huffman coding
- Arithmatic coding
- Shannon Faro coding

→ used in legal & medical domains

Lossy compression

→ Reconstructed data is an approx. of original data (Source coding)

→ Ex:
- Linear Prediction
- Transform Coding

→ Broadcast TV, multimedia

Lossless compression

→ Reversible process
→ No info is lost
→ Compression ratio is less
→ Used for that data which human can handle directly like text data
→ Compression is independent of psychovisual system

Lossy Compression

→ Non reversible process
→ Some info is lost
→ Compression ratio high
→ Used for diffused data which human can't understand / interpret
→ compression is dependent on psychovisual system

Types → Entropy Coding → Transform Coding
→ Predictive Coding → Layered Encoding

→ Entropy Coding :- Average info in an image is known as entropy
- Coding is based on the entropy of the source & on the possibility of occurence of symbols
- An event is less likely to occur

is said to contain more info than event that is more likely to occur

Set of symbols (alphabet) $S = \{s_1, s_2, \ldots s_N\}$

N is no of symbols in the alphabet

Probability distribution of symbols

$$P = \{p_1, p_2, \ldots p_N\}$$

Acc. to Shannon, entropy H of an information source S is defined as

$$H = -\sum_{i=1}^{N} p_i \log_2 p_i$$

→ Predictive coding - Idea is to remove the mutual dependency b/w successive pixels & then perform the encoding

→ Normally samples would be large but the difference would be small

Eg: Pixels : 400 405 420 430

Difference : 5 15 10

→ Transform coding → objective is to exploit the information packing capability of the transform

→ Energy is packed into fewer components only these component are encoded and transmitted

→ Idea is to remove the redundant high freq. components to create compression

→ Removal of these freq. components leads to loss of information

→ This loss of information if tolerable can be used for image & video applications

Layered Coding

→ Used in case of layered images

→ Data structures like pyramids are useful to represent an image in multiresolution form

→ Sometimes these images are segmented as foreground & background & based on the application requirement encoding is done

→ It is also in the form of selected freq. coefficients or selected bits of pixels in an image

Redundancy : Repetitive data

Data may share common characteristics / overlapped info. Can be implicit / explicit

Eg: AAAA BB

{(A, 4), (B, 2)}

Eg. $\begin{pmatrix} 10 & 10 & 20 \\ 20 & 20 & 20 \\ 20 & 20 & 10 \end{pmatrix}$

Explain 10 → 3, 20 → 6

Implicit

$$\begin{pmatrix} 00 & 01 \\ 00 & 10 \end{pmatrix}$$

$$I = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \text{ and } \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

Redundancy Types

1) Coding Redundancy - Caused due to poor selection of coding technique
   - → unique code to all message symbols
   - → wrong choice of coding technique leads to unnecessary additional bits. These extra bits leads to redundancy

Amount of uncertainty → Self information $I = \log_2 \frac{1}{p}$ of event $= -\log_2 p$ bits

Coding redundancy = Avg bits used to code − Entropy

$$L_{avg} = \sum_{k=0}^{n} l(r_k) \cdot p(r_k)$$

$$H = -\sum_{i=1}^{m} p_i \log_2 p_i$$

$p(r_k)$ → Probability of pixels

r_k → grey level

l_k → Length of code

Total no of bits required to code image of size $M \times N = M \times N \times L_{avg}$

Interpixel redundancy → related with inter pixel correlations within an image

→ Much of the visual contribution of a single pixel is redundant & can be guessed from the values of its neighbors

Eg: Consider an image with constant background

→ Visual nature of image background is given by many pixels that are not actually necessary

→ known as spatial / Geometrical redundancy

↓

→ single frame (inter-frame)

→ or among multiple frames (interframe/temporal)

Inter pixel reduced by predictive, bit plane algo, run length coding & dictionary based algos

Psychovisual redundancy - Eye & brain do not respond to all visual info with same sensitivity

→ Some info is neglected during processing by brain.

Elimination of this info doesn't affect interpretation of image by brain

→ Edges & textural regions are important features & brain groups & correlates such grouping for object perception

Eg Quantization - When 256 levels are reduced to 16 levels but object is recognizable

Chromatic Redundancy → presence of unnecessary colors in image

→ Color channels of color images are highly correlated & human visual system cannot perceive millions of colors

→ Therefore colors not perceived by human visual system can be removed without affecting image quality

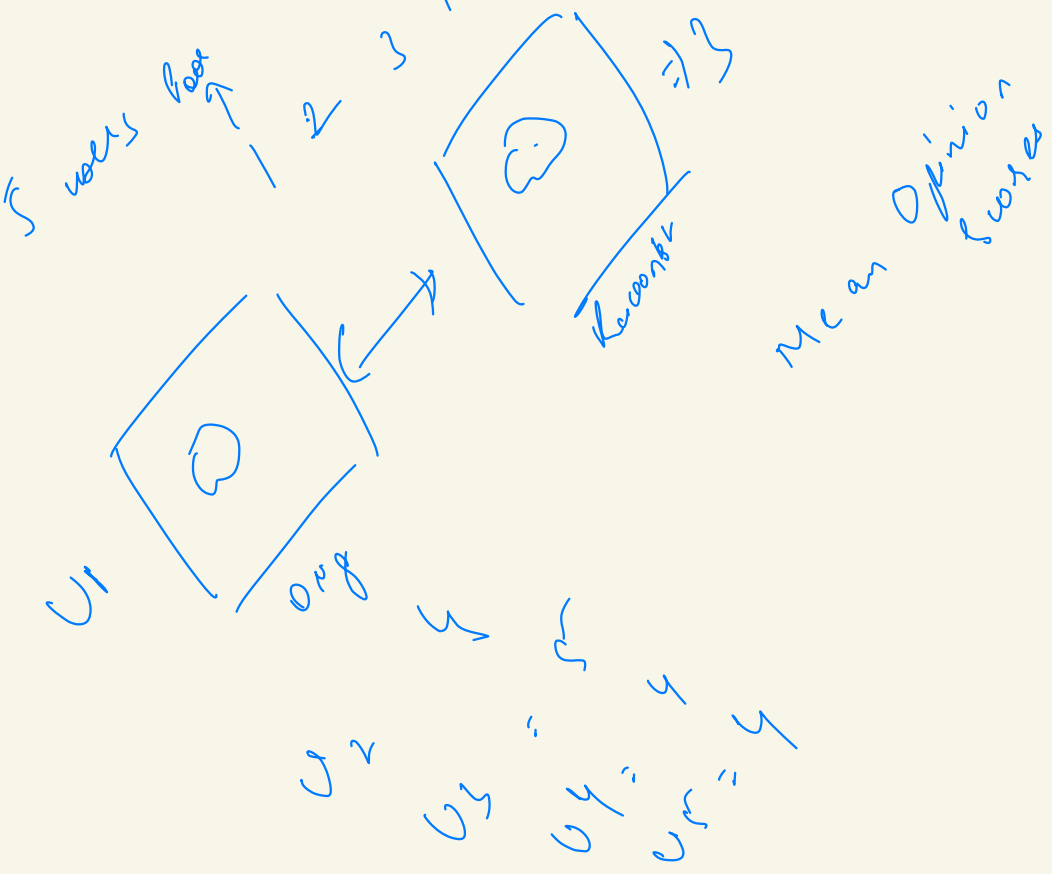
Fidelity - Accurate reproduction of data

Distortion → original - reconstructed image

Objective fidelity - Error, SNR, PSNR

Subjective fidelity → Rating 1-5

poor ↓ ↓ Excellent



Run length coding

→ Identify length of pixel values & encodes the image in the form of run

→ Each row of image is written as sequence

→ Then length is represented as run of black & white pixels known as run length coding

→ Very effective in compressing image

→ Further compression can be done using variable length coding to code run lengths

→ Used for binary & gray level images

Eg:

0	0	0	0	0
0	0	0	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

5×5

① Horizontal RLC

(0, 5)

(0,3) (1,2)

(1, 5)

(1, 5)

(1, 5)

* No of vectors = 6
* Max length = 5
  ↓
  3 bit in binary
* No of bits per pixel = 1 (color 1)
* Total no of pixel
  = 6 × (3+1)
  = 24

* No of pixels for original image
  = 5×5 = 25

Compression ratio = $\frac{25}{24}$ = 1.04

② Vertical RLC

Run length vectors

(0, 2), (1, 3)

(0, 2), (1, 3)

(0, 2), (1, 3)

(0, 1), (1, 4)

(0, 1), (1, 4)

No of vectors = 10

* 3 bits
* 1 bit / pixel
* No of pixels = 10 * (3+1) = 40
* Original image = 5 * 5 = 25

Compression ratio $= \frac{25}{40} = 0.625 : 1$

→ Compression ratio changes with scan line

→ Estimate of entropy of black run is H_0

" " " " white " " H_1

" " avg value " black " " L_0

" " " " " white " " L_1

Run length entropy

$$H_{\text{run length}} = \frac{H_0 + H_1}{L_0 + L_1}$$

Huffman Coding (Lossless)

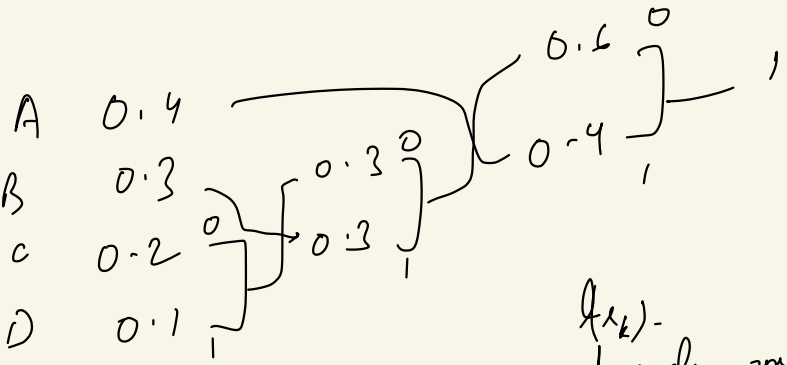
- It is a type of variable length coding
- Here coding redundancy can be eliminated by choosing a better way of assigning codes

Huffman Coding algo. is given as:

- List of the symbols & sort probabilities per symbol
- Combine the lowest 2 probabilities of symbols & label the new node with it
- Newly created item is given priority & placed at the highest position in the sorted list
- Repeat step 2 until only one node remain
- Assign code 0 to higher up symbol & 1 to lower down symbol
- Now trace the code symbols going backwards

Eg

Symbols	A	B	C	D
Prob.	0.4	0.3	0.2	0.1



$l(r_k)$ - length of code words

A	1
B	01
C	000
D	001

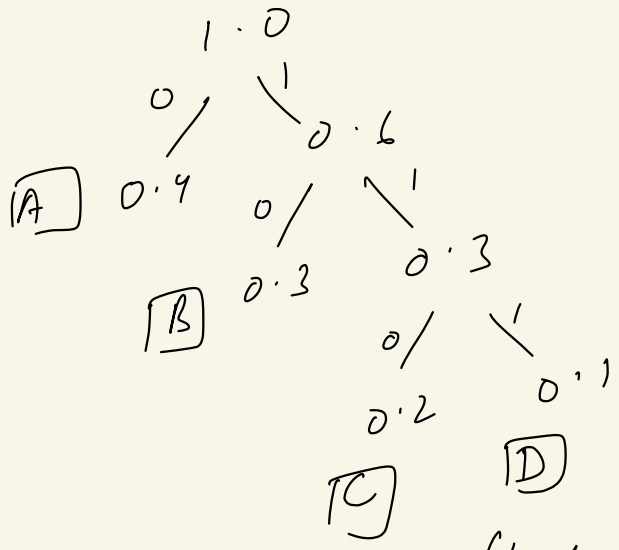
L_{avg} = avg no of bits used to represent memory

$= \sum l_{r_k} p_{r_k}$

$= 1 \times 0.4 + 2 \times 0.3 + 3 \times 0.2 + 3 \times 0.1$

$= 1.9$ bits/pixel

Q A B C D
0.4 0.3 0.2 0.1



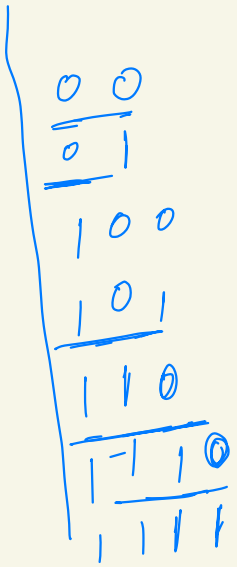
A - 0
B - 10
C - 110
D - 111

Shannon Fano coding (lossless)
- Shannon Fano algos was independently developed by Shannon at Bell labs & Robert Fano at MIT
- It is a type of variable length coding
- It assigns a code to each symbol based on their probability of occurence

Shannon Fano:
- Create a list of probabilities or freq. counts for the given set of symbols
- Sort the list of symbols in decreasing order of probability
- Split the list into 2 parts with total probability of both the parts being as close to each other as possible
- Assign value 0 for upper half & 1 to lower half
- Repeat steps 3 & 4, until all symbols are split into individual subgroups

A	B	C	D	E	F	G
0.3	0.4	0.2	0.1	0.1	0.05	0.05

Symbol	Probability				
A	0.3	00	A	00	(2)
B	0.2	01	B	01	(2)
C	0.2	100	C	100	(3)
D	0.1	101	D	101	(3)
E	0.1	110	E	110	(3)
F	0.05	1110	F	1110	(4)
G	0.05	1111	G	1111	(4)

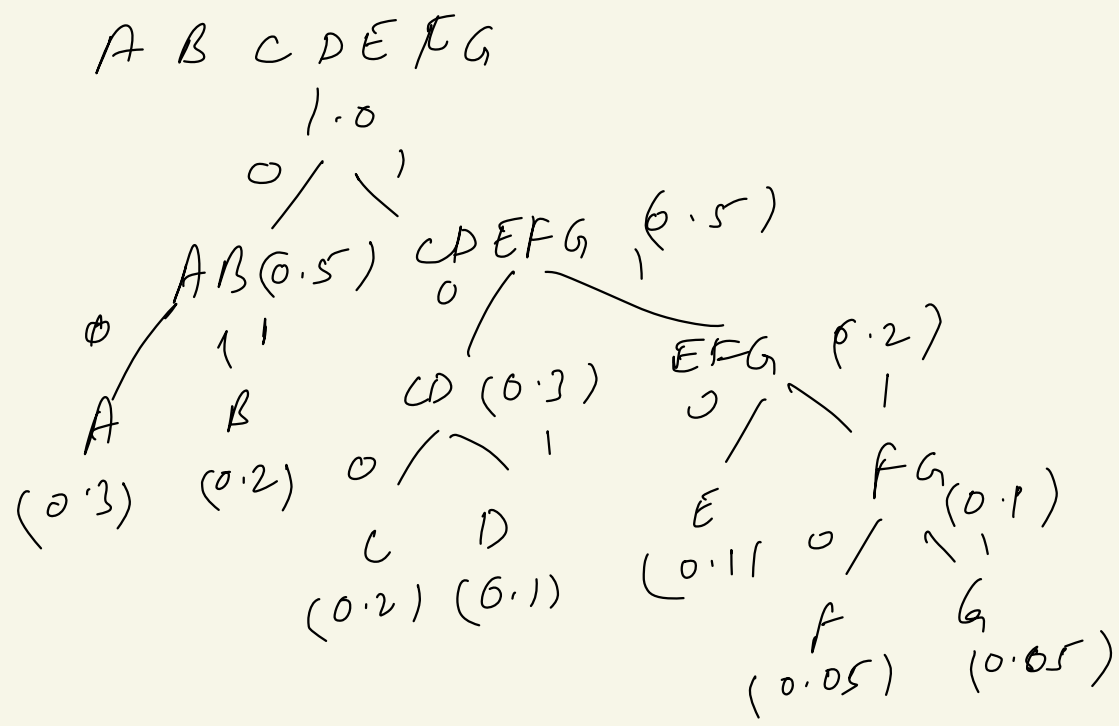


$$L_k = \sum l_{x_k} p_{x_k} = 2.6$$

$$H = -\sum p_i \log_2 p_i = 2.546$$

$$\text{Coding Redundancy} = L_k - H = 0.1$$

$$\text{Coding Efficiency} = \eta = \frac{H}{L_{avg}} = 0.979$$



Shannon Fano Tree Approach

A	0 0
B	0 1
C	1 0 0
D	1 0 1
E	1 1 0
F	1 1 1 0
G	1 1 1 1

Bit Plane Coding (Lossless)

- Idea of run length coding can be extended for multilevel images
- Bit plane coding technique splits a multilevel image into a series of 1 level images
- Effective approach to provide bit stream truncation ability

Any m-bit grey level image can be represented as

$$a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + \ldots + a_1 2^1 + a_0 2^0$$

- Zeroth order bit plane is generated by collecting the a_0 bits of each pixel
- The first order bit plane is generated by collecting all the first bits of each pixel
- Similarly $m-1$ order bit plane is generated by collecting all the a_{m-1} bits of each pixel

Q. $A = \begin{matrix} 2 & 6 & 6 \\ 6 & 7 & 7 \\ 1 & 2 & 4 \end{matrix}$ 3×3

individual plane of the image can be compressed using RLC technique

Binary equivalent of image A)

Three planes 1) MSB 2) MIDDLE 3) LSB

$$A = \begin{matrix} 010 & 110 & 110 \\ 110 & 111 & 111 \\ 001 & 010 & 100 \end{matrix}$$

$$A(MSB) = \begin{matrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{matrix}$$

$$A(Middle) = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$A(LSB) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

- Disadvantage — Neighborhoods in the spatial domain like 3 and 4 having binary coded 011 and 100 are not present together in any of the planes

- To avoid this problem, grey codes can be used instead of binary codes

- In grey codes, successive codes differ by only 1 bit

Algo for generating grey code can be written as

$$g_i = \begin{cases} a_i \oplus a_{i+1} & 0 \le i \le m-2 \\ a_i & i = m-1 \end{cases}$$

Another bit plane coding scheme (Constant Area Coding)

- Bit planes have uniform regions 1's & 0's. A constant portion of the plane can be uniquely coded using less no of bits

- CAC divided the image into set of blocks of size $m \times m$ like 8x8 or 16x16

3 types of blocks

- Block of all white pixels
- " " black "
- " " mixed "

Most probable blocks is assigned a single code of either 0 or 1, the remaining blocks are assigned a 2 bit code

- White block skipping is a scheme where a majority of blocks (white) are assigned a single bit. Rest of blocks including mixed pixel blocks are encoded together

Arithmetic coding (lossless)

- In this coding technique, a string of characters like the words "hello there" is represented using a fixed no. of bits per character as in ASCII code
- When a string is converted to arithmetic encoding, frequently used characters are stored with fewer bits & not so frequently occuring characters are stored with more bits resulting in relatively fewer bits used in total

Arithmetic	Huffman
- Complex technique for coding short messages - Gives optimum result - There is no. one to one correspondence b/w	- Simple technique - Does not give optimum result - There is 1-to-1 correspondence b/w source symbol & code word

source symbol & code coded

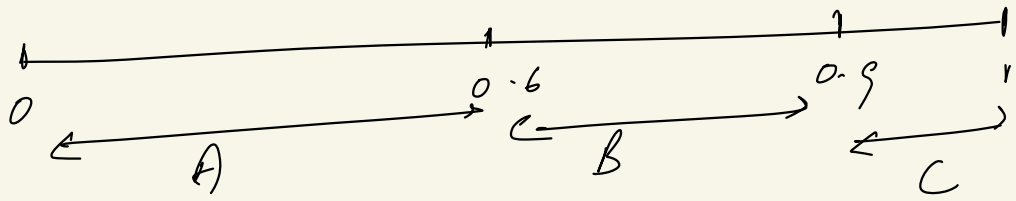
- Compression ratio is more
- Execution time is more

- Compression ratio is less
- Execution time is less

Q. Code string CAB using arithmetic coding

A	B	C
0.6	0.3	0.1

1) Divide the range into 0 - 1



First character C → Range 0.9 - 1

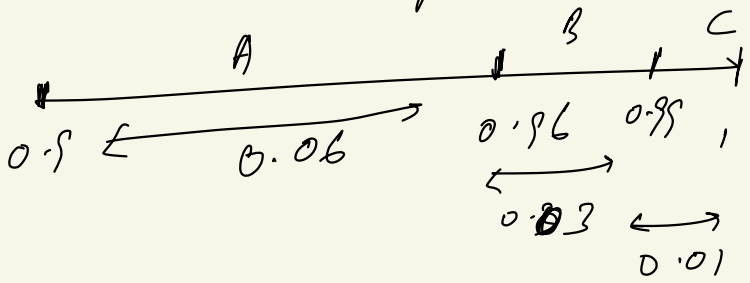
Code would start in this range only (0.1)

Divided among symbols

Cumulative character probability table

	Cumulative probability
A	$0.9 + 0.6 \times 0.1 = 0.96$
B	$0.96 + 0.3 \times 0.1 = 0.99$
C	$0.99 + 0.1 \times 0.1 = 1.0$

2) Resultant new range is :



* Next character A

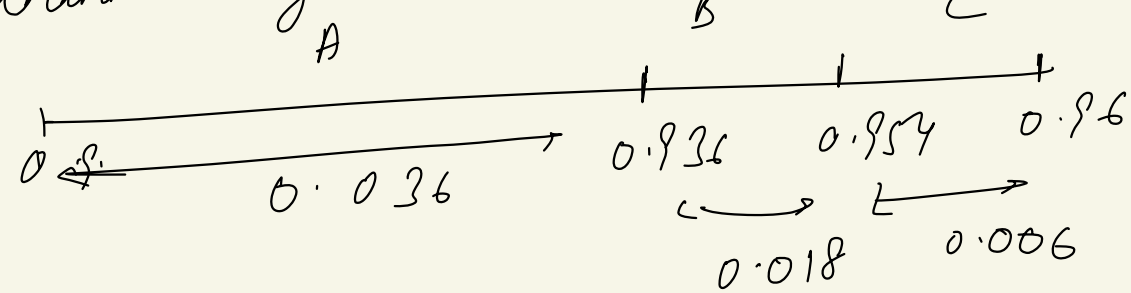
$0.9 - 0.96$

Range $0.96 - 0.9 = 0.06$

Cumulative character probability table

A	$0.9 + 0.6 \times 0.06 = 0.936$
B	$0.936 + 0.3 \times 0.06 = 0.954$
C	$0.954 + 0.1 \times 0.06 = 0.96$

3) Resultant range is



Next character B

Range = 0.936 – 0.954

Final code for string CAB

0.936 – 0.954

Dictionary Coding

- Dictionary based coding techniques replace input strings with a code to an entry in a dictionary
- Most well known dictionary based technique is Lempel-Ziv Algorithm
- Idea behind Lempel Ziv Welch (LZW) coding is to use a dictionary to store the string patterns that have already been encountered
- Indices are used to encode the repeated patterns
- Encoder reads input string & identifies the recurrent words & outputs their indices from the dictionary
- If a new word is encountered, the word is sent as output in the uncompressed form and is entered into dictionary as a new entry

Advantages - Faster
- Adaptive
- Not based on statistics, so no dependency on data distribution

Encoding - Objective is to identify longest pattern for each collected segment of the input string
- It is checked in the dictionary
- There is no match, segment becomes a new entry in the dictionary

Q String ADBB

Initial Dict.		Dict with new entry		Dict. with new entry		Dict with new entry	
A	1		1		1		1
		A	2		2		2
D	2			D	3		3
		D	3				
B	3		4	B	4		4
		B	5				5
					5	B	6

AD DB BB

1) Word = Null
2) Read string
K ← A
word + k = A → already in dictionary
word = word + A = Null + A = A
3) Read string again : DB B

Hence k = D

word + k = AD → not in dictionary

Send o/p = 1

New entry = A + D = AD → add to dictionary

New word = D

4) Read char — B

Next character B

Word + k = DB

Send O/P = 3

New entry - DB

5) Read string B

k = B

word + k = BB (enter in dictionary)

O/P = 2

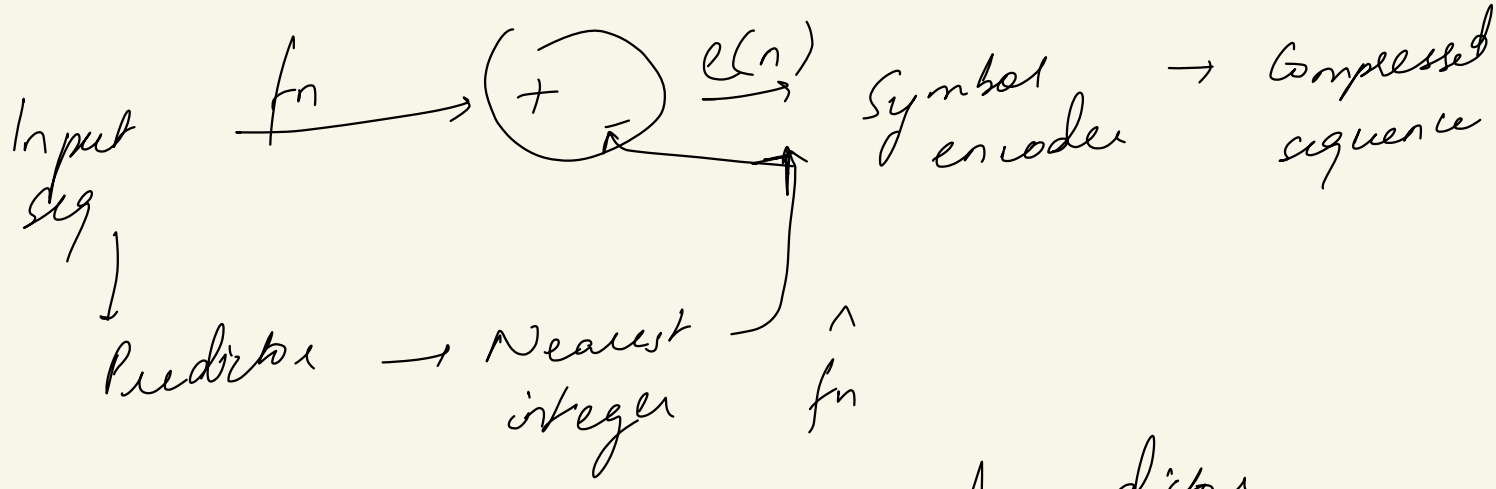
6) Read = Null

Stop

Lossless predictive coding

- Technique eliminates the inter pixel dependencies by predicting new information which is obtained by taking the difference b/w actual & predicted value of pixel
- Encoder takes a pixel of input image f_n
- Predictor predicts the anticipated value of that pixel using past inputs
- Predicted value is rounded to nearest integer value, which is known as predicted value $\hat{f}_n$
- Error is the difference b/w actual & predicted value $e_n = f_n - \hat{f}_n$
- Error is sent across the channel. The same predictor is used in the decoder side to predict the value
- Reconstructed image

$$f_n = e_n + \hat{f}_n$$



- Most crucial part ⇒ design of predictor
- Prediction by linear predictor taking a linear estimation of previous n sits i

$$\hat{f}_n = \text{round}\left[\sum_{i=1}^{m} \alpha_i f_{n-i}\right]$$

m - order of predictor as a fn.

- 1D linear predictive coding

$$\hat{f}_n(x,y) = \text{round}\left[\sum_{i=1}^{m} \alpha_i f(x,y-i)\right]$$

Quantized error

$$f_n = e_n + \hat{f}_n$$

Q 23, 34, 39, 47, 55, 63

Value	Predictive Coding
23	23
34	34-23 = 11
39	39-34 = 5
47	47-39 = 8
55	55-47 = 8
63	63-55 = 8

max value in original sequence = 63

requires 6 bits + 1 bit for sign

* No of bits required to code original message
= 6 × 7 = 42 b

* Predicted coded sequence
→ max value is 23

5 bits + 1 bit for sign

→ Total no of bits required =

6 × 6 = 36 b

(5+1)

It saves 6 bits

Lossy Predictive Coding
- more loss of info
- distortion

if data loss is acceptable - it is tolerable
- Compression ratio is very large

23, 64, 39, 47, 55, 63

23 — 23

64 — 64 − 23 = 41 → overloading → (31)

39 — 39 − 64 = −25

47 — 47 − 39 = 8

55 — 55 − 47 = 8

63 — 63 − 55 = 8

max 5 bits are required

+ 1 sign bit : 6

5 bit highest value → 31 values

↳ drastically reduces no of bits

↳ loss of information ↑

6 pixels × 6 bits = 36 bits

- This process can be extended further by using only 1 bit to encode difference b/w adjacent pixels
- Scheme is known as delta modulation

+ ve or - ve

- Predictor defined as

$$\hat{f} = \alpha \times f_{n-1}$$

α → prediction coefficient

- generally for first digit

$$\hat{f} = f_{n-1} \qquad \mathcal{E} \rightarrow \text{varies}$$

- Error is computed

$$e_n = f_n - \hat{f_n} = f_n - \hat{f}_{n-1}$$

- Error is quantized as

$$e_n = \begin{cases} +\mathcal{E} & \text{for } e_n > 0 \\ -\mathcal{E} & \text{otherwise} \end{cases}$$