Module 2:
Source coding

Lecturer: Shashi Raj Pandey

Email: srp@es.aau.dk

AALBORG UNIVERSITY
DENMARK

Connectivity

Today’s lecture

1. Information theory basics
2. Shannon-Fano and Huffman coding
3. LZ77 and LZW coding
4. Lossy coding: JPEG and video

Information theory basics

AALBORG UNIVERSITY
DENMARK

Connectivity

What is information?

Shannon (1947) decided that information was related to probability, and novelty:

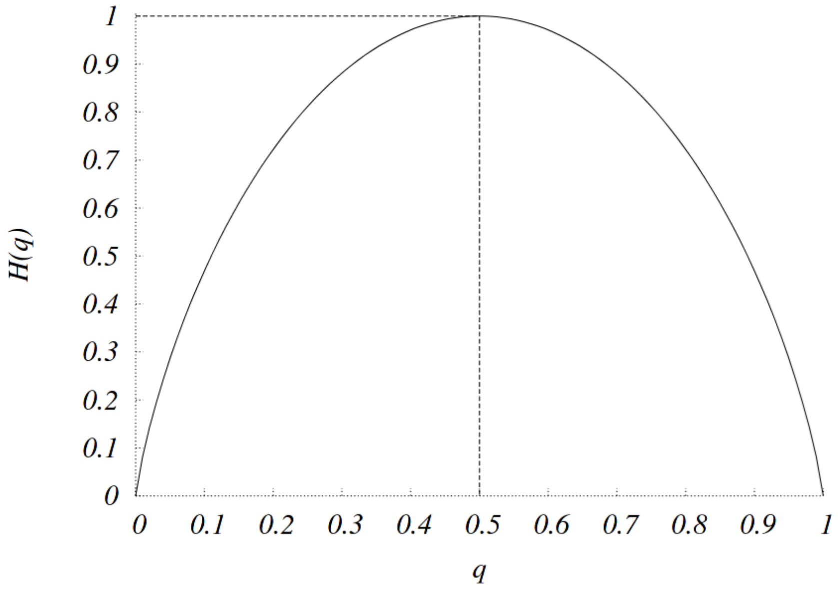
- The outcome of a match between Real Madrid and a kids' team is not very novel: the probability of Real losing is very low
- The outcome of a tight match between Real Madrid and Barcelona is very informative, as it could go either way

For a binary outcome, the closer the probability is to 0.5, the more informative it is

Information entropy

Shannon's definition of entropy uses the logarithm of probability (it is valid for any discrete distribution)

$$H(x) = \sum_{x \in X} -p(x) \log_2(p(x))$$



Properties of entropy

- H(x) is always positive
- H(x) is 0 only if x is already certain
- If X has M elements, the entropy is **maximum with a uniform distribution**[why is that? **show mathematically**], with entropy $log_2(M)$
- If X and Y are independent, H(x,y)=H(x)+H(y), but in general, H(x,y)≤H(x)+H(y) (show this! Hint: use definition)

Why is this useful?

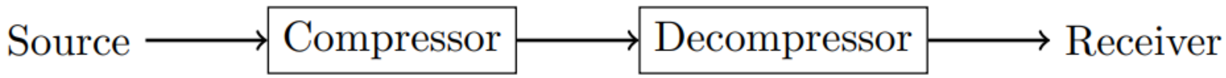
Entropy is measured in bits (per symbol)! The entropy of a random variable tells us the **minimum (average) number of bits that we need to transmit its value**

- ➔ Basic encoding idea: more frequent values are encoded as shorter **symbols**
- ➔ We need to ensure that sequences are interpreted correctly: we can do this by using prefix code (no codeword is a prefix of any other – **prefix free code**)

Noiseless Coding Theorem
(Source Coding)

AALBORG UNIVERSITY
DENMARK

Connectivity


Source
Compressor
Decompressor
Receiver

Classic Information Theory

Alice (A) wants to transmit information over a channel to Bob (B)

Source Coding Theorem: The cost of Alice transmitting *n* i.i.d. copies of discrete random variable *X* to Bob *over a noiseless channel* scales as Shannon's entropy *H(X)* as $n \to \infty$

$$H(X) = \sum_{x \in \mathrm{supp}(X)} \Pr[X = x] \log \frac{1}{\Pr[X = x]}$$

Interpretation

- Consider X^n the concatenation of n independent samples from X
- C(Y) the expected number of bits needed to transmit a sample of random variable Y to Bob
- **Source Coding Theorem** asserts:

$$\lim_{n \to \infty} \frac{C(X^n)}{n} = H(X)$$

$$H(X^n) = n \cdot H(X)$$

Additive property of entropy
(on independence)

Noisy Coding Theorem (Channel Coding)

AALBORG UNIVERSITY
DENMARK

Connectivity

any source of noise:
wireless link, error prone device, IP network w\ the potential of packet loss, ….


Source
Compressor
Channel
Decompressor
Receiver


Compressed Message
Encoder
Input X
Channel
Output Y
Decoder
To Decompressor

Information-theoretic coding

AALBORG UNIVERSITY
DENMARK

Connectivity

Encoding a random variable

Let us consider the random variable X, which can take 5 different values and has H(x)=2.167

x	p(x)
A	0.39
B	0.2
C	0.153
D	0.143
E	0.114

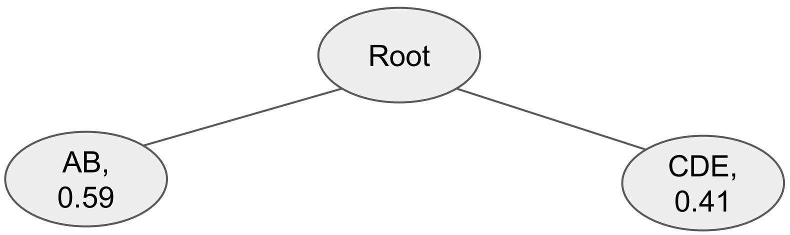
Shannon-Fano coding

We can encode any variable by using a tree splitting method (Shannon-Fano):

➔ At every step, divide the symbols so that the list is as close as possible

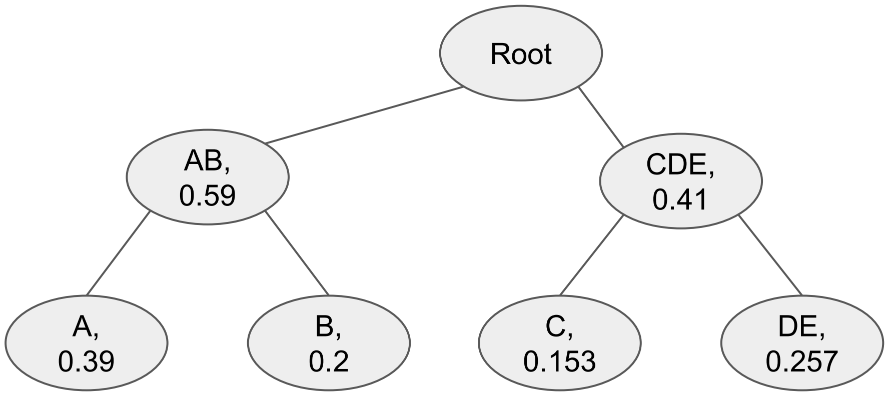
x	p(x)
A	0.39
B	0.2
C	0.153
D	0.143
E	0.114

Shannon-Fano coding



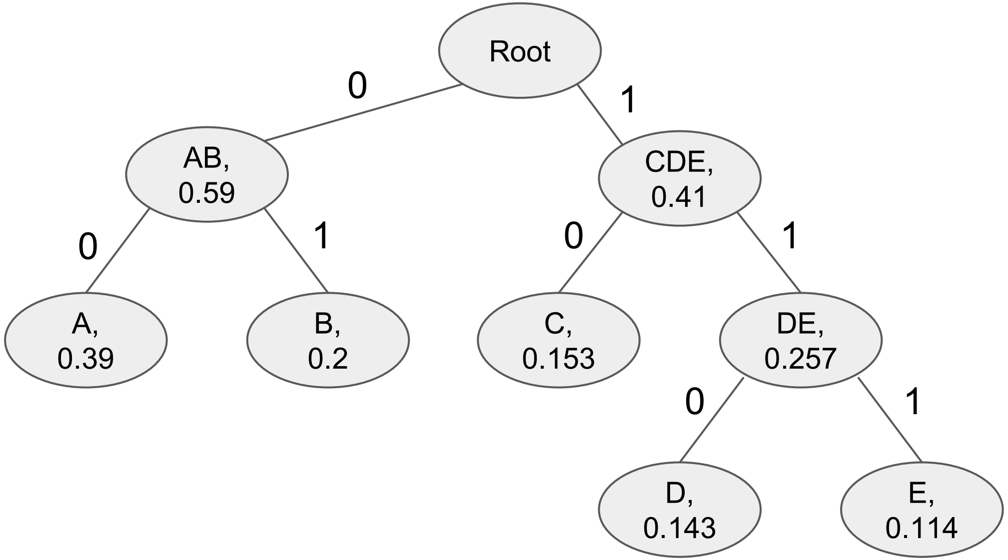
x	p(x)
A	0.39
B	0.2
C	0.153
D	0.143
E	0.114

Shannon-Fano coding



x	p(x)
A	0.39
B	0.2
C	0.153
D	0.143
E	0.114

Shannon-Fano coding



x	p(x)	C(x)	E[L]
A	0.39	00	0.78
B	0.2	01	0.4
C	0.153	10	0.306
D	0.143	110	0.429
E	0.114	111	0.342

$\boxed{E[L]=2.257}$

Huffman coding

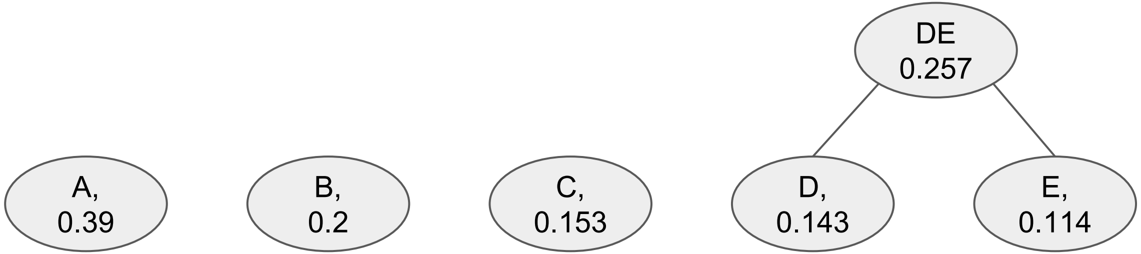
We can build the tree in reverse:

➔ Start from each symbol as a leaf, then join the two nodes with the lowest probability

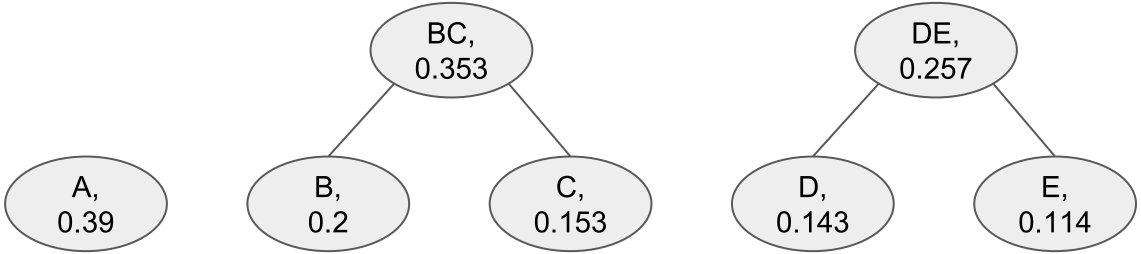
x	p(x)
A	2/5
B	1/5
C	1/7
D	1/7
E	4/35

Huffman coding

x	p(x)
A	2/5
B	1/5
C	1/7
D	1/7
E	4/35

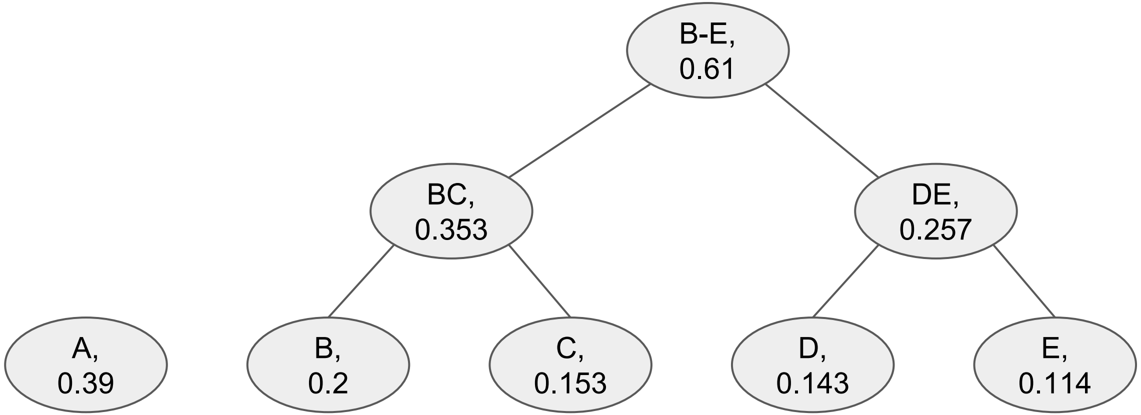


Huffman coding



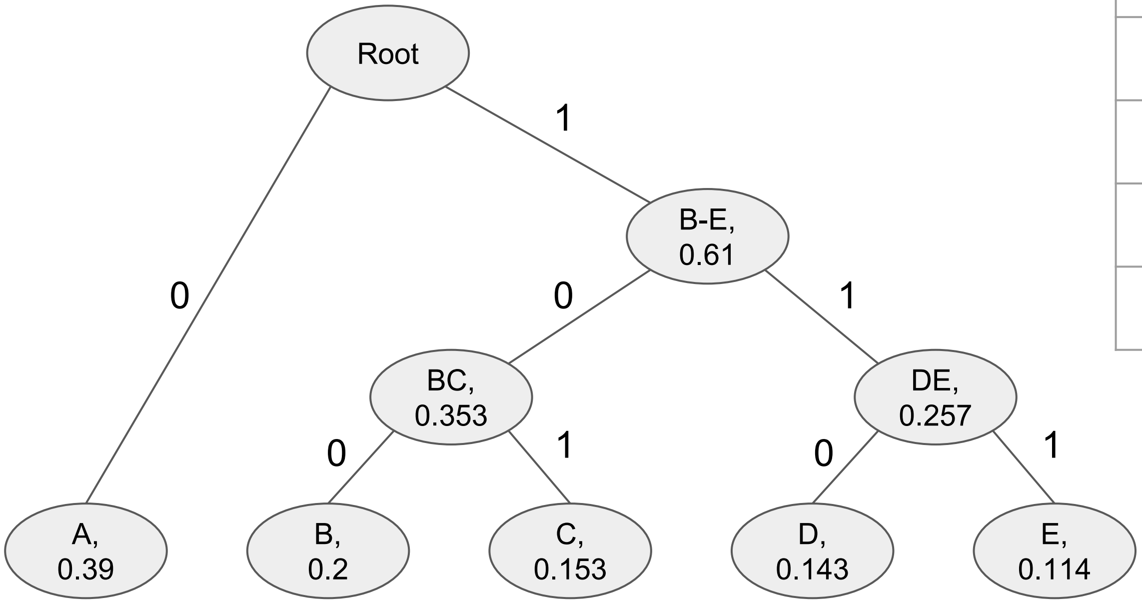
x	p(x)
A	2/5
B	1/5
C	1/7
D	1/7
E	4/35

Huffman coding



x	p(x)
A	2/5
B	1/5
C	1/7
D	1/7
E	4/35

Huffman coding



x	p(x)	C(x)	E[L]
A	0.39	0	0.39
B	0.2	100	0.6
C	0.153	101	0.459
D	0.143	110	0.429
E	0.114	111	0.342

E[L]=2.22

Comparison

- Entropy: 2.167 (theoretical minimum)
- Huffman: 2.22
- Shannon-Fano: 2.257

Huffman is (in some cases) slightly more efficient, and is generally used for compression

Huffman codes are close to the theoretical limit, and even Shannon-Fano codes have an expected length lower than H(x)+1 (loss of only 1 bit)

Dictionary-based coding

AALBORG UNIVERSITY
DENMARK

Connectivity

Data compression

lossless: eliminate statistical redundancy

- video, audio
- no better compression ratio than 2:1

compression ratio = uncompressed/compressed

lossy: eliminate less important information

- JPEG, MP3

Dictionary-based coding

What happens if we do not know the probability distribution of symbols, but just have a very large binary file?

Furthermore, symbols are not always IID (which is a basic assumption for Huffman and Shannon-Fano)

We can use the concept of a **dictionary**: we can encode sequences of symbols and reduce them

The Lempel-Ziv (1977) algorithm does just that!

LZ77

Lossless data compression

- compression ratio no better than 2:1

Sliding window compression (why so?- we see in the next slide)

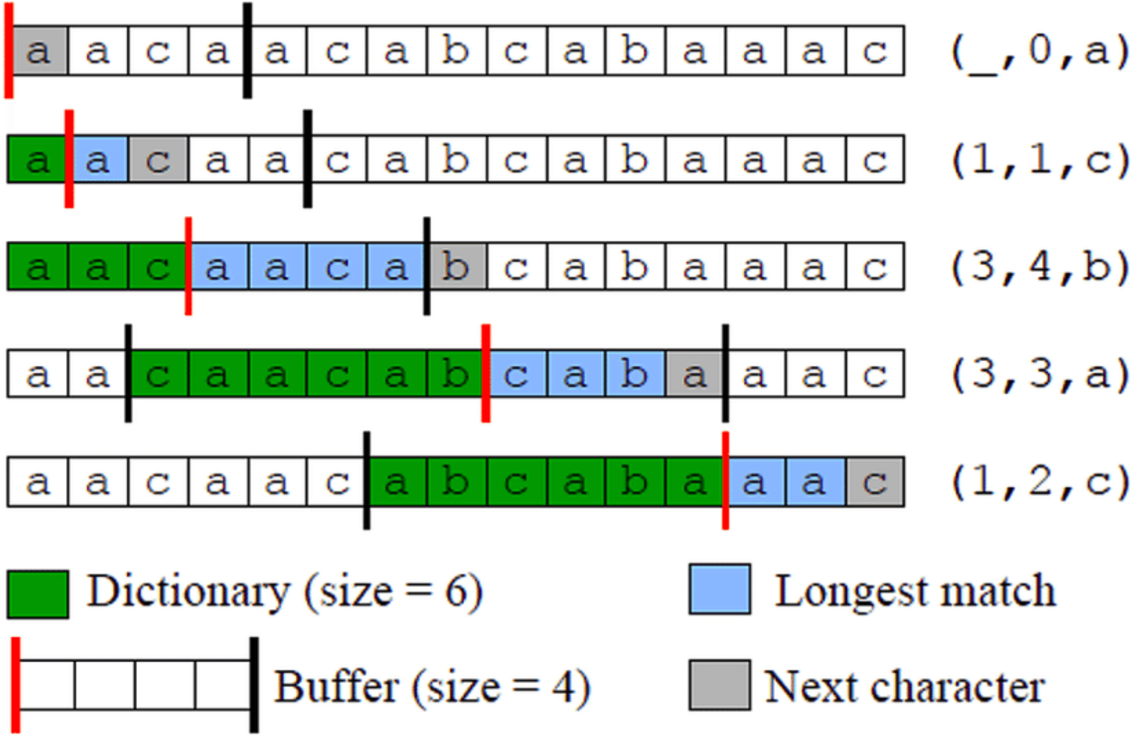
variants – LZW, LSS , ..used in PNG, ZIP, GIF…

LZ77 pseudocode

1. **Match**: longest repeated occurrence of input that begins in the window
2. If there is a match output the distance *d* (go-back) and length *l* of the match, along with the next character *c*: ***(d, l, c)***
3. Otherwise output (***0, 0, c)***
4. **Remove** *l+1* characters from the front of the window
5. **Move** the first *l+1* characters from the window to the input

(sliding..the window..)

LZ77 example



LZW coding

Lempel-Ziv-Welch (1984) is another option:

We encode a sequence of bytes (8 bits) into 12-bit indices

0-255: 1-character sequences

256-4095: dictionary words (longer sequences)

We encode as we go, finding longer and longer sequences and increasing compression as the file goes on (compression is negative for shorter files, but can be huge for longer files)

LZW pseudocode

1. Initialize the dictionary with all possible strings of length 1
2. Find the longest string W in the dictionary that matches the current input
3. Substitute W with the related index
4. Add W followed by the next symbol to the dictionary
5. Go to 2

How does LZW work?

TOBEORNOTTOBEORTOBEORNOT#

1. T: sub 20, add TO
2. O: sub 15, add OB
3. B: sub 2, add BE
4. E: sub 5, add EO
5. O: sub 15, add OR
6. R: sub 18, add RN
7. N: sub 14, add NO
8. O: sub 15, add OT
9. T: sub 20, add TT
10. TO: sub 27, add TOB
11. BE: sub 29, add BEO
12. OR: sub 31, add ORT
13. TOB: sub 36, add TOBE
14. EO: sub 30, add EOR
15. RN: sub 32, add RNO
16. OT: sub 34, stop (reached end)

Final string: 20-15-2-5-15-18-14-15-20-27-29-31-36-30-32-34-0

Symbol	Decimal	Symbol	Decimal	Symbol	Decimal	Symbol	Decimal
#	0	L	12	W	23		
A	1	M	13	X	24		
B	2	N	14	Y	25		
C	3	O	15	Z	26		
D	4	P	16				
E	5	Q	17				
F	6	R	18				
G	7	S	19				
H	8	T	20				
I	9	U	21				
J	10	V	22				
K	11	W	23				

How does LZW work?

TOBEORNOTTOBEORTOBEORNOT#

1. T: sub 20, add TO
2. O: sub 15, add OB
3. B: sub 2, add BE
4. E: sub 5, add EO
5. O: sub 15, add OR
6. R: sub 18, add RN
7. N: sub 14, add NO
8. O: sub 15, add OT
9. T: sub 20, add TT
10. TO: sub 27, add TOB
11. BE: sub 29, add BEO
12. OR: sub 31, add ORT
13. TOB: sub 36, add TOBE
14. EO: sub 30, add EOR
15. RN: sub 32, add RNO
16. OT: sub 34, stop (reached end)

Final string: 20-15-2-5-15-18-14-15-20-27-29-31-36-30-32-34-0

Symbol	Decimal	Symbol	Decimal	Symbol	Decimal	Symbol	Decimal
#	0	L	12	W	23	TT	35
A	1	M	13	X	24	TOB	36
B	2	N	14	Y	25	BEO	37
C	3	O	15	Z	26	ORT	38
D	4	P	16	TO	27	TOBE	39
E	5	Q	17	OB	28	EOR	40
F	6	R	18	BE	29	RNO	41
G	7	S	19	EO	30		
H	8	T	20	OR	31		
I	9	U	21	RN	32		
J	10	V	22	NO	33		
K	11	W	23	OT	34		

Where are LZ77 and LZW used?

The Unix *compress* utility uses LZW, as well as GIF images, pdfs

More recent versions use other versions of encoding based on the LZ77 idea (.tar.gz format), with the *deflate* tool (also used in .zip compression)

Other formats (.7z, .rar) are also based on LZ variants and combinations with other algorithms

Lossy encoding: JPEG and video

AALBORG UNIVERSITY
DENMARK

Connectivity

Lossless and lossy compression

Lossless compression: the file can be reconstructed perfectly (the message is the same)

Lossy compression: some information is lost, but acceptable quality is maintained

The Discrete Cosine Transform

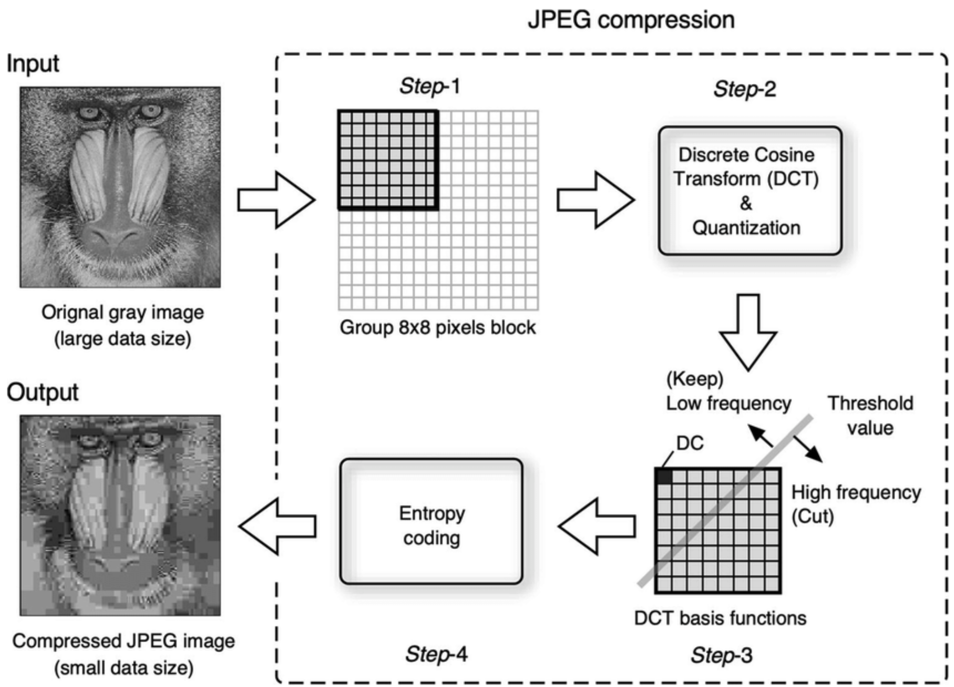
DCT is an alternative to the Fourier transform:

- Sharp edges are high-frequency
- Large blots of color are low-frequency

In general, humans notice low-frequency changes more (the loss of high-frequency components is seen as fuzziness).

JPEG compression

As high-frequency elements (sharp edges) are less noticeable, JPEG compression removes them by cutting them out and quantizing the DCT parameters. This can lead to artifacts in text and line drawings



Video encoding

Basic idea: each frame is a JPEG image

However, we have an additional correlation: over time!

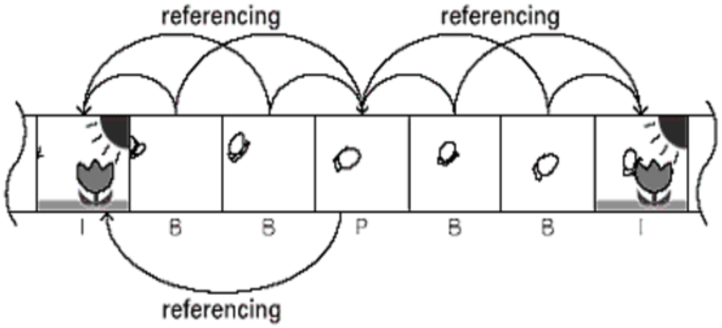
Subsequent frames are highly similar, and we can use *differential encoding*

MPEG Group of Pictures encoding

I frame: independently encoded (full .jpg image) – key frame

P frame: references previous I frame –predictive coded picture

B frame: references previous I frame and next P or I frame – Bipredictive coded picture



More I frames makes the video easier to edit and cut, but reduce compression efficiency: differential encoding works!