Digital Image Processing

Huffman Coding

DR TANIA STATHAKI
READER (ASSOCIATE PROFESSOR) IN SIGNAL PROCESSING
IMPERIAL COLLEGE LONDON

Imperial College
London

Elements from Information Theory

- Any information generating process can be viewed as a source that emits a sequence of symbols chosen from a finite alphabet.
  - ➢ ASCII symbols (text)
  - ➢ n –bit image values (2^n symbols)

- The simplest form of an information source is the so called Discrete Memoryless Source (DMS). Successive symbols produced by such a source are statistically independent.

- A DMS is completely specified by the source alphabet $S = \{s_1, s_2, \dots, s_n\}$ and the associated probabilities $P = \{p_1, p_2, \dots, p_n\}$

Imperial College London

Elements from Information Theory

- The **Self-Information** of a symbol s_i with probability p_i is defined as:

$$I(s_i) = \log_2 \frac{1}{p_i} = -\log_2 p_i$$

  - ➢ The occurrence of a less probable event provides more information.
  - ➢ The information of a sequence of independent events taken as a single event equals the sum of their individual information.
  - ➢ An event can be the occurence of a symbol.

- The **Average Information per Symbol** or **Entropy** of a DMS is:

$$H(S) = \sum_{i=1}^{n} p_i I(s_i) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

- The Entropy is measured in bits/symbol.

What is actually entropy?

- Interpretation of entropy:
  - ➢ By definition it is the average amount of information that symbols of a specific source carry.
  - ➢ Entropy is also a measure of the "disorder" (uncertainty) of a system.

- When we design a coding scheme the average number of bits per symbol we can achieve is always greater that the entropy. **Therefore, the entropy is the best we can do in term of bits/symbol!**

Imperial College London

Extension of a DMS

- Given a DMS of size n, it might be beneficial to group the original symbols of the source into blocks of N symbols. Each block can now be considered as a single source symbol generated by a source S^N which has n^N symbols.

- In this case the entropy of the new source is
$$H\left(S^N\right) = N \times H(s)$$

- We observe that when the source is extended, the entropy increases, however, the symbols increase in length. The entropy per original symbol remains the same.

Imperial College London

Noiseless source coding theorem

- Let S be a source with alphabet size n and entropy $H(s)$.

- Consider coding blocks of N source symbols into binary codewords. For any $\delta > 0$ (with δ a small number), it is possible by choosing N large enough to construct a code in such a way that the average number of bits per original source symbol l_{avg} satisfies the following:

$$H(S) \le l_{avg} < H(s) + \delta$$

- We observe that for δ small enough, the average number of bits per symbols converges to the entropy of the source. This is the best coding we can achieve.

- The above is not realistic since the alphabet size increases too much with N.

Imperial College London

Examples of possible codes for a 4-symbol source

- In the table below we see four different codes for a four-symbol alphabet. The entropy of the source is 1.75bits/symbol.

Symbols	Probability	Code 1	Code 2	Code 3	Code 4
s_1	1/2	0	0	0	0
s_2	1/4	0	1	10	01
s_3	1/8	1	00	110	011
s_4	1/8	10	11	111	0111
Average length		1.125	1.25	1.75	1.875

- The **Average Length** of a code is

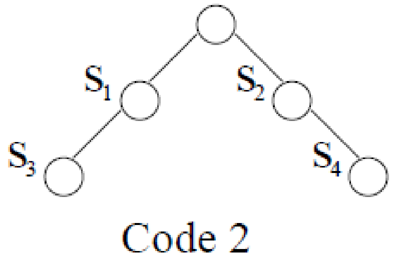
$$l_{avg} = \sum_i l_i p_i$$

- l_i is the length of the codeword in bits which corresponds the symbol s_i.

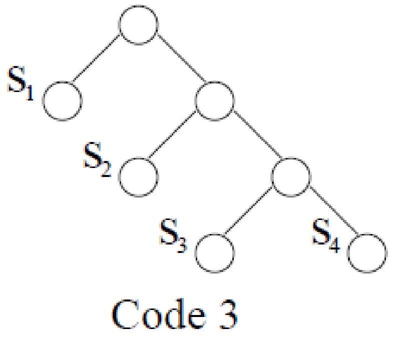
Imperial College London

Code characteristics

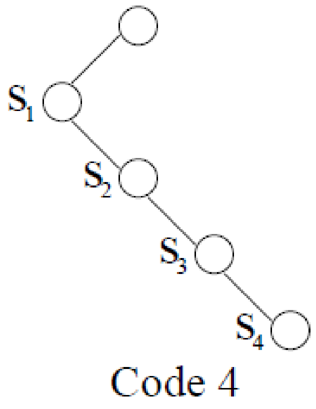
- It is desirable for a code to exhibit Unique Decodability.
- **Prefix Codes**: no codeword is a prefix of another codeword.
- A prefix code is always uniquely decodable. The reverse is not true.
- A code can be possible depicted as a binary tree where the symbols are the nodes and the branches are 0 or 1. The codeword of a symbol can be found if we concatenate the 0s and 1s that we have to scan until we reach that symbol, starting from the "root " of the tree.
- In a prefix code the codewords are associated only with the external nodes.



Code 2



Code 3



Code 4

Imperial College
London

Huffman coding (1952)

- Huffman coding is a very popular coding method.

- Huffman codes are:
  - Prefix codes.
  - Optimal for a given model (set of probabilities).

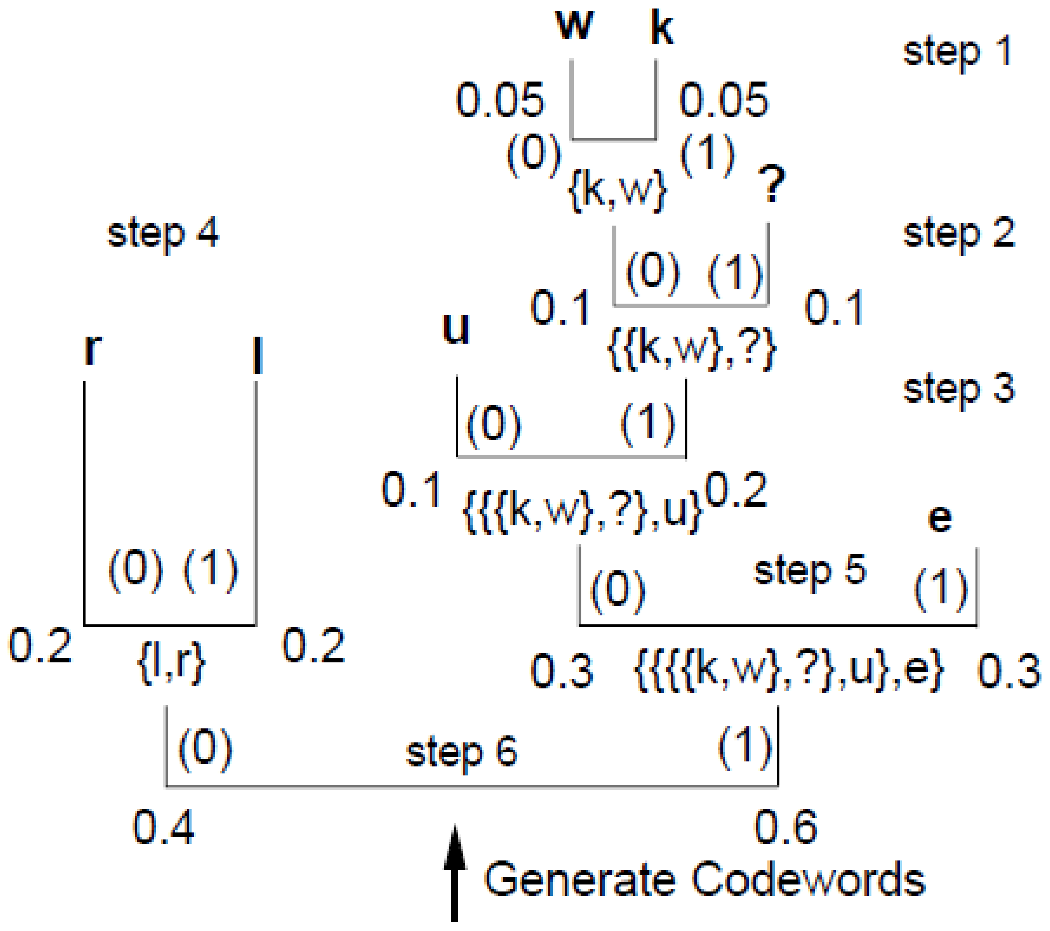
- Huffman coding is based on the following two observations:
  - Symbols that occur more frequently will have shorter codewords than symbols that occur less frequently.
  - The two symbols that occur less frequently will have the same length.

Imperial College London

Huffman coding

Symbol	Probability	Codeword
k	0.05	10101
l	0.2	01
u	0.1	100
w	0.05	10100
e	0.3	11
r	0.2	00
?	0.1	1011

↓ Merge Symbols



Huffman coding

	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
k 1/20	e 0.3	e 0.3	e 0.3	e 0.3	{l,r} 0.4	{{{{k,w},?},u},e} 0.6
l 0.2	l 0.2	l 0.2	l 0.2	{{{k,w},?},u} 0.3	e 0.3	{l,r} 0.4
u 0.1	r 0.2	r 0.2	r 0.2	l 0.2	{{{k,w},?},u} 0.3	
w 1/20	u 0.1	u 0.1	{{k,w},?} 0.2	r 0.2		
e 0.3	? 0.1	? 0.1	u 0.1			
r 0.2	k 0.05	{k,w} 0.1				
? 0.1	w 0.05					

Imperial College London

Properties of Huffman Codes

- $H(S) \leq l_{avg} \leq H(S) + 1$
- If $p_{max} < 0.5$ then $l_{avg} \leq H(S) + p_{max}$
- If $p_{max} \geq 0.5$ then $l_{avg} \leq H(S) + p_{max} + 0.086$
- $H(S) = l_{avg}$ if the probabilities of the symbols are of the form 2^k, with k a negative integer.
- For an N –th extension of a DMS we have $H(S) \leq l_{avg} \leq H(S) + \frac{1}{N}$
- The complement of a Huffman code is also a valid Huffman code.
- A minimum variance Huffman code is obtained by placing the combined letter in the sorted list as high as possible.
- The code efficiency is defined as $H(S)/l_{avg}$
- The code redundancy is defined as $l_{avg} - H(S)$

Imperial College London

Huffman Decoding: Bit-Serial Decoding

- This method is a fixed-input bit rate but variable-output symbol rate scheme. It consists of the following steps:
  1. Read the input compressed stream bit by bit and traverse the tree until a leaf node is reached.
  2. As each bit in the input stream is used, it is discarded. When the leaf node is reached, the Huffman decoder outputs the symbol at the leaf node. This completes the decoding for this symbol.

- We repeat these steps until all of the input is consumed. Since the codewords are not of the same length, the decoding bit rate is not the same for all symbols. Hence, this scheme has a fixed input bit rate but a variable output symbol rate.

https://dzone.com/articles/binary-trees-part-1

Imperial College London

Huffman Decoding: Lookup-Table-Based Decoding

- Lookup-table-based methods have a variable input bit rate and constant decoding symbol rate.
- We have to construct the so-called **Lookup Table** using the symbol-to-codeword mapping table (Huffman code). If the longest codeword in this table is L bits, then the lookup table will have 2^L rows.
- Let c_i be the codeword that corresponds to symbol s_i. Assume that c_i has l_i bits. In this method we associate c_i not with a single codeword but with 2^{L-l_i} codewords. These are all the codewords where the first l_i bits are the codeword c_i and the last $L - l_i$ bits can be all possible binary numbers with $L - l_i$ bits. These are 2^{L-l_i} on total. Therefore, each symbol s_i is associated with 2^{L-l_i} codewords of fixed length L.
- The 2^{L-l_i} pairs $\left(s_i, \mathrm{codeword}_{ij}\right), j = 1, \ldots, 2^{L-l_i}$ are stored in into the Lookup Table.

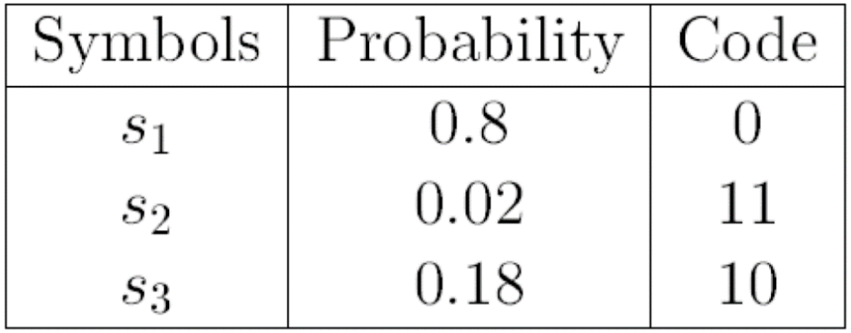
Huffman Decoding: Lookup-Table-Based Decoding (cont.)

- When we receive the bit stream for decoding we read the first L bits.
- By checking the Lookup Table, we find the symbol s_i which has the read L-bit word as one if its possible codewords.
- When we find this symbol, we know that the “true” codeword for that symbol is formed by the first l_i bits only of the read L-bit word.
- The first l_i bits are discarded from the buffer.
- The next l_i bits are appended to the buffer so that the next L-bit word for investigation is formed.
- We carry on this procedure until the entire bit stream is examined.

Imperial College London

Example of Huffman coding with poor performance

- Consider the following example of the Huffman code of a 3-symbol alphabet

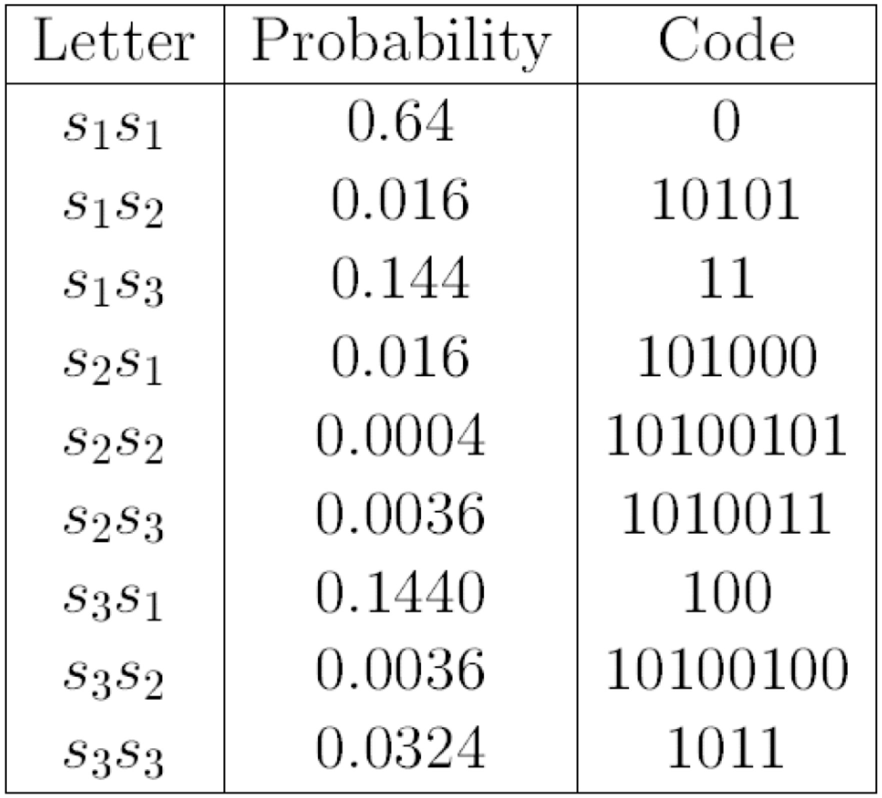


Symbols	Probability	Code
s_1	0.8	0
s_2	0.02	11
s_3	0.18	10

- In that case $H=0.816$ bits/symbol.
- Average number of bits per symbol is $lavg=1.2$ bits/symbol.
- Redundancy $lavg-H=1.2-0.816=0.384$ bits/symbol, which is 47% of entropy.

Example of Huffman coding with poor performance

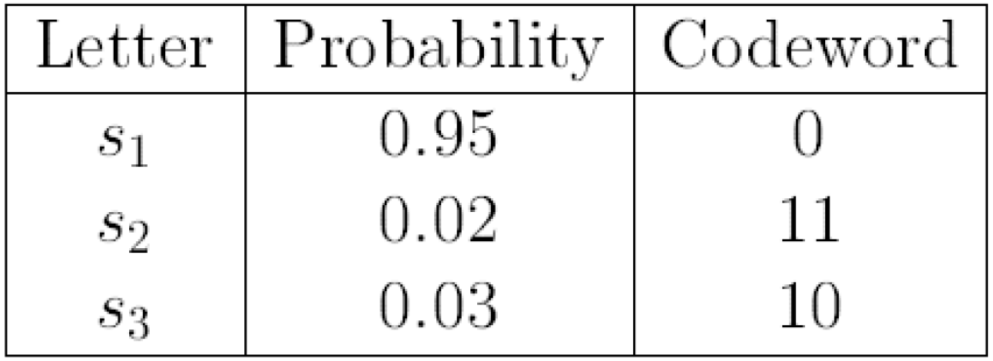
- Consider the previous example where the source is extended by 2.
- Average number of bits per symbol is $lavg$=1.7228 bits/(new symbol).
- Average number of bits per original symbol is $lavg$=0.8614 bits/(original symbol).
- Redundancy is $lavg-H=0.8758-0.816=0.06$ bits/symbol.
- This is 7% of entropy.



Letter	Probability	Code
s_1s_1	0.64	0
s_1s_2	0.016	10101
s_1s_3	0.144	11
s_2s_1	0.016	101000
s_2s_2	0.0004	10100101
s_2s_3	0.0036	1010011
s_3s_1	0.1440	100
s_3s_2	0.0036	10100100
s_3s_3	0.0324	1011

Example of Huffman coding with poor performance

- Consider the following example of the Huffman code of a 3-symbol alphabet



Letter	Probability	Codeword
s_1	0.95	0
s_2	0.02	11
s_3	0.03	10

- In that case $H = 0.335$ bits/symbol.
- Average number of bits per symbol is $lavg = 1.05$ bits/symbol.
- Redundancy $lavg - H = (1.05 - 0.335)$ bits/symbol, which is 213% of entropy.

Imperial College London

Example of Huffman coding with poor performance

- Consider the previous example where the source is extended by 2.
- Average number of bits per symbol is $lavg=1.222$ bits/(new symbol).
- Average number of bits per original symbol is $lavg=0.611$ bits/(original symbol).
- Redundancy is $lavg-H=(0.611-0.335)$ bits/symbol.
- This is 72% of entropy.
- Redundancy drops to acceptable values for $N=8$. The alphabet size is then 6561 symbols.

Letter	Probability	Code
s_1s_1	0.9025	0
s_1s_2	0.0190	111
s_1s_3	0.0285	100
s_2s_1	0.0190	1101
s_2s_2	0.0004	110011
s_2s_3	0.0006	110001
s_3s_1	0.0285	101
s_3s_2	0.0006	110010
s_3s_3	0.0009	110000

Differential coding

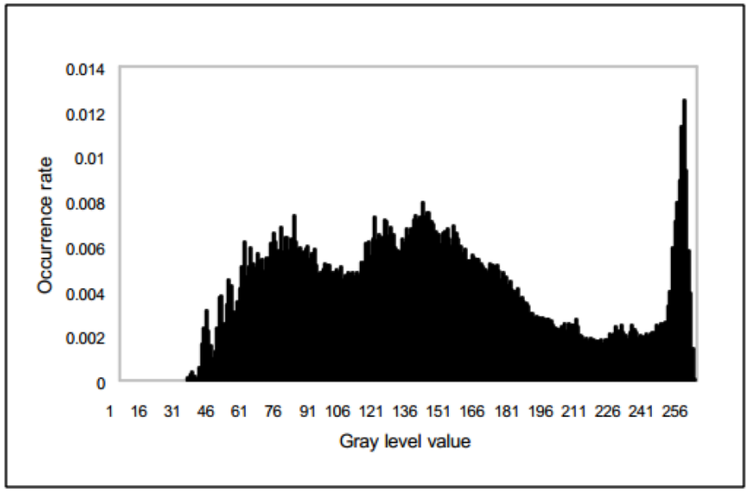
- Huffman coding works well when the distribution of probabilities of the symbols deviate from uniform.
- In case where the symbols are image intensities we can change the distribution of probabilities by replacing each pixel intensity with its differential.
- The differential is the difference between the intensity of the pixel of interest and a function of the neighbouring intensities. **For real life images this is very small for most pixels!**
- The function of the neighbouring intensities can be considered as an approximation of the prediction of the pixel of interest.
  - ➢ This method falls within the so called **predictive coding**.
- For example $f(x,y)$ can be replaced by $g(x,y) = f(x,y) - f(x,y-1)$.
- Another alternative is $g(x,y) = f(x,y) - \frac{1}{3}(f(x,y-1) + f(x-1,y) + f(x-1,y-1))$.

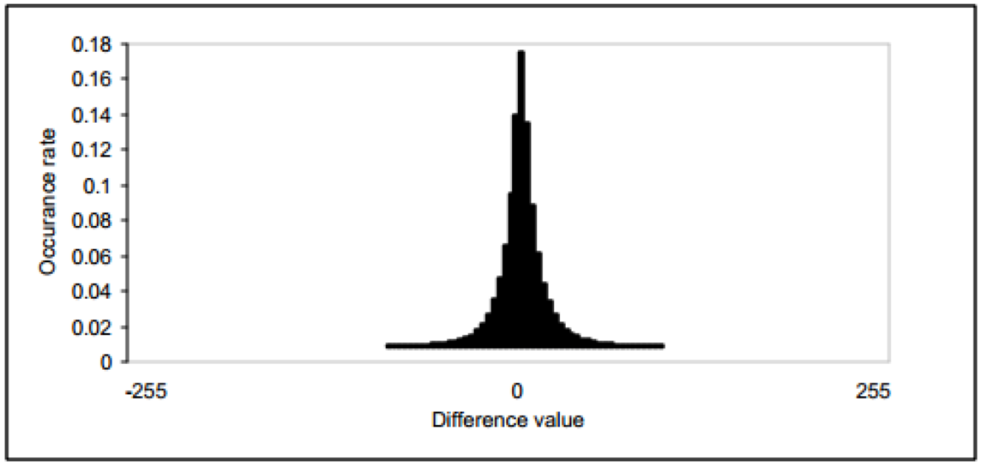
Imperial College London

Differential coding

- The histogram of the original image is shown in the top figure.
- The histogram of the difference image obtained by using horizontal pixel-to-pixel differencing is shown in the bottom figure.

  Seems more appropriate for Huffman coding.
- The dynamic range increases from 256 to 511.

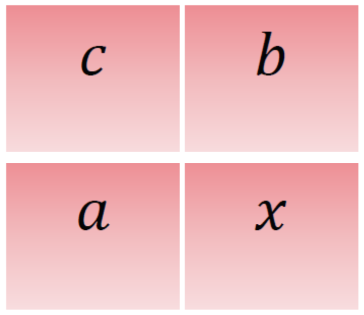




Imperial College London

The Lossless JPEG Standard

- We use **Differential Coding** to form prediction residuals.
- Residuals then coded with either a Huffman coder or an arithmetic coder.
- We will focus on Huffman coding.
- In lossless JPEG, one forms a prediction residual using "previous" pixels in the current line and/or the previous line.
- If x is the pixel of interest, the **prediction residual** is $r=y-x$ with $y=f(a,b,c)$, a,b,c the “previous” pixels (we can define previous as the top and left pixels and furthermore, the top - left diagonal pixel.



Imperial College London

The Lossless JPEG Standard

- The residual is expressed as a pair of symbols: the **category** and the **actual value (magnitude)**.
- The magnitude is expressed in binary form with the Most Significant Bit (MSB) always 1 if it is positive.
- The category represents the number of bits needed to encode the magnitude. This value ONLY is Huffman coded.

  Example: Assume that the residual has magnitude 42. $(42)_{10}=(101010)_2$ belongs to Category 6.

  Codeword: (Huffman-code-for-6)∪(Binary-number-for-42-with-MSB-1)
- If the residual is negative, then the code for the magnitude is the one's complement of its absolute value.
- Codewords for negative residual always start wish a zero bit.

Imperial College
London

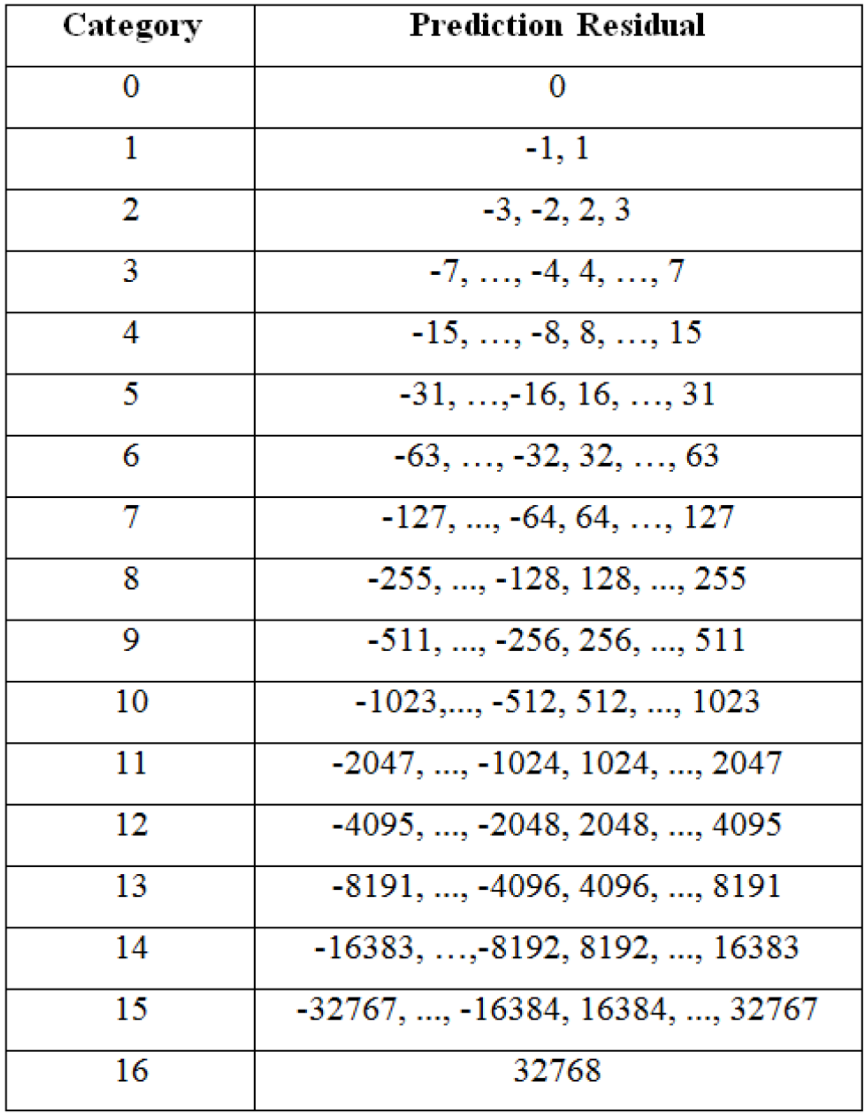
Categories-Range of prediction residuals

- The range of residuals is: $-32767, \dots, 32768 = -(2^{15}-1). \dots,\ 2^{15}$
- On total we have $2 \cdot 2^{15} = 2^{16}$residuals
- In a fixed-length representation we would need 16 bits to represent the residual.

- Category is Huffman coded

- A prediction residual is represented by a binary word with the same length as its category.

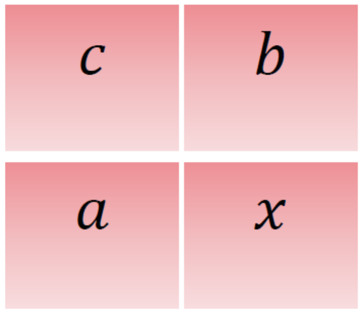
- For 0 residual we don't use a binary word. The code of Category 0 is only stored.



Category	Prediction Residual
0	0
1	-1, 1
2	-3, -2, 2, 3
3	-7, …, -4, 4, …, 7
4	-15, …, -8, 8, …, 15
5	-31, …,-16, 16, …, 31
6	-63, …, -32, 32, …, 63
7	-127, ..., -64, 64, …, 127
8	-255, ..., -128, 128, ..., 255
9	-511, ..., -256, 256, ..., 511
10	-1023,..., -512, 512, ..., 1023
11	-2047, ..., -1024, 1024, ..., 2047
12	-4095, ..., -2048, 2048, ..., 4095
13	-8191, ..., -4096, 4096, ..., 8191
14	-16383, …,-8192, 8192, ..., 16383
15	-32767, ..., -16384, 16384, ..., 32767
16	32768

Encoding of the Prediction Residual

$a = 100$
$b = 191$
$c = 100$
$x = 180$
$y = \frac{a+b}{2} \cong 145$
$r = y - x = 145 - 180 = -35$
$35_{10} = (32 + 2 + 1)_{10} = (2^5 + 2^1 + 2^0)_{10} = (100011)_2$
$-35_{10} = (\mathbf{011100})_2$



We see that -35 requires 6 bits and therefore, it belongs to **Category 6**.

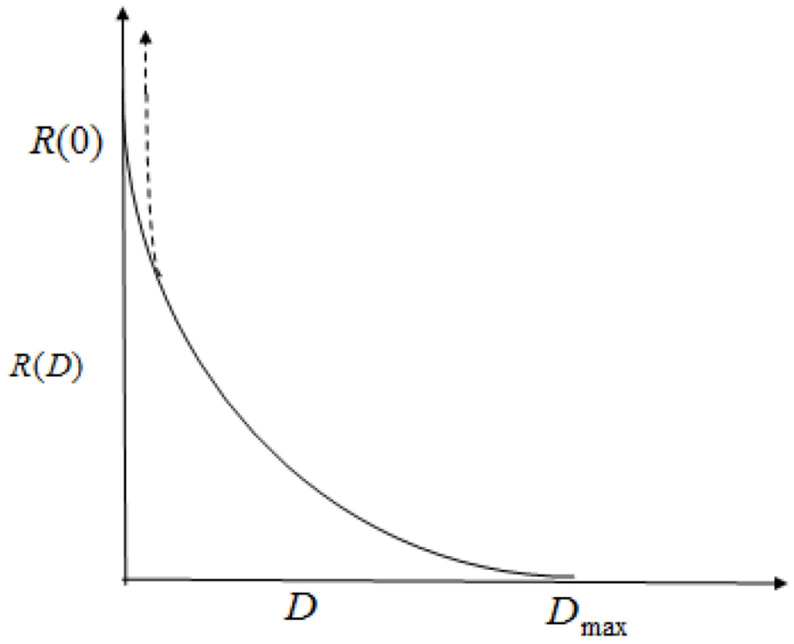
- Suppose Huffman code for **Category 6** is **1110**. Therefore, -35 is coded by the $10-$bit codeword **1110** **011100**.
- Without entropy coding, -35 would require 16 bits.

Lossy compression

- Lossy compression of images deals with compression processes where decompression yields an **imperfect reconstruction of the original image data**.
- Regardless of the compression method that has being used, **given the level of image loss (or distortion), there is always a bound on the minimum bit rate of the compressed bit stream**.
- The analysis that relates signal distortion and minimum bit rate falls within the so called **Rate-Distortion Theory**.

Lossy compression

- The figure below demonstrates the rate-distortion $R(D)$ relationship.
- The figure below exists for lossy schemes only, since in lossless schemes we don’t allow distortion.
- For a discrete signal zero distortion coding is achieved when $\boldsymbol{R(0) = \text{the source entropy}}$. For a continuous source the rate rises without limit (observe the dashed line ---).



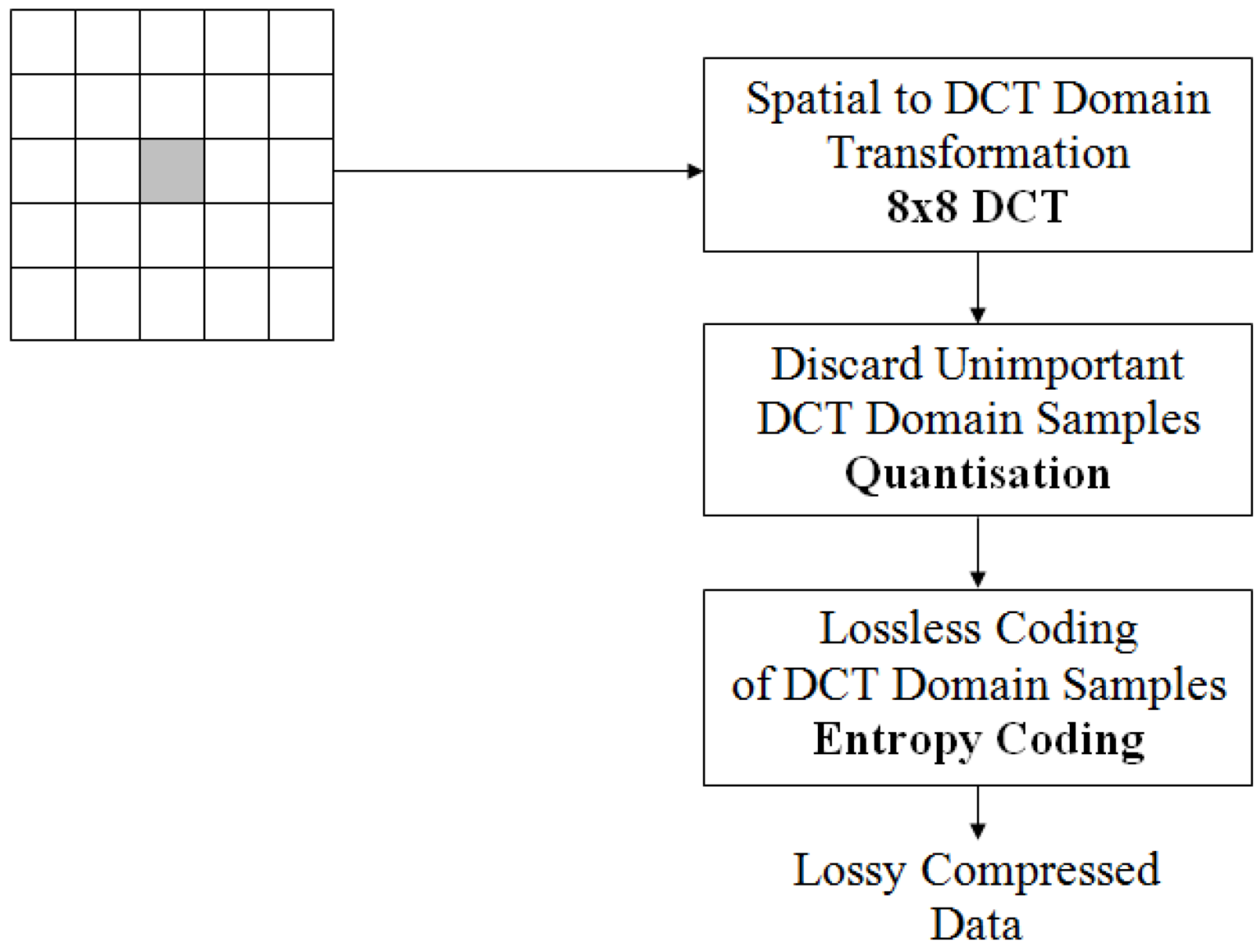
Block-based coding

- Spatial-domain block coding
  The pixels are grouped into blocks and the blocks are then compressed in the spatial domain.
  Example: Vector quantization

- Transform-domain block coding
  The pixels are grouped into blocks and the blocks are then transformed to another domain, such as the frequency domain.
  Example: DCT DFT DHT KL

Imperial College London

A Generic DCT-Based Image Coding System


Spatial to DCT Domain Transformation
8x8 DCT
Discard Unimportant DCT Domain Samples
Quantisation
Lossless Coding of DCT Domain Samples
Entropy Coding
Lossy Compressed Data

Imperial College
London

DCT Based Coding Example – Low activity region

The input block (labelled **original**) is taken from a low activity region; that is, there are very small differences among pixel values in that area.

$$X = \begin{bmatrix} 168 & 161 & 161 & 150 & 154 & 168 & 164 & 154 \\ 171 & 154 & 161 & 150 & 157 & 171 & 150 & 164 \\ 171 & 168 & 147 & 164 & 164 & 161 & 143 & 154 \\ 164 & 171 & 154 & 161 & 157 & 157 & 147 & 132 \\ 161 & 161 & 157 & 154 & 143 & 161 & 154 & 132 \\ 164 & 161 & 161 & 154 & 150 & 157 & 154 & 140 \\ 161 & 168 & 157 & 154 & 161 & 140 & 140 & 132 \\ 154 & 161 & 157 & 150 & 140 & 132 & 136 & 128 \end{bmatrix}$$

Imperial College London

DCT Based Coding Examples

- In order to provide for uniform processing, most standard DCT coders require that image pixels are pre-processed so that their expected mean value is zero.
- After subtracting 128 from each element of the block, the DCT output block is given by

$$Y = \begin{bmatrix} 214 & 49 & -3 & 20 & -10 & -1 & 1 & -6 \\ 34 & -25 & 11 & 13 & 5 & -1 & 15 & -6 \\ -6 & -4 & 8 & -9 & 3 & -3 & 5 & 10 \\ 8 & -10 & 4 & 4 & -15 & 10 & 6 & 6 \\ -12 & 5 & -1 & -2 & -15 & 9 & -5 & -1 \\ 5 & 9 & -8 & 3 & 4 & -7 & -14 & 2 \\ 2 & -2 & 3 & -1 & 1 & 2 & -3 & -4 \\ -1 & 1 & 0 & 2 & 3 & -2 & -4 & -2 \end{bmatrix}$$

DCT Based Coding Examples

- It is the process of quantization which leads to compression in DCT domain coding. Quantization of pixel $y[k,l]$ is expressed as:

$$z[k,l] = \text{round}\left[\frac{y[k,l]}{q[k,l]}\right]$$

- $\text{Round}[\cdot]$ replaces the argument of the function with the nearest integer.
- $q[k,l]$ are the elements of a quantisation matrix Q.
- The choice of Q depends on:
  - ➢ Psychovisual characteristics
  - ➢ Compression ratio considerations

Imperial College London

Quantization matrix

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Imperial College London

Quantized DCT

The quantized DCT matrix is given by:

$$Z = \begin{bmatrix} 13 & 4 & 0 & 1 & 0 & 0 & 0 & 0 \\ 3 & -2 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Only 11 values are needed to represent Z.

- Compression ratio of $\frac{64}{11} = 5.8$ is achieved.
- Z is entropy coded.

Decompression

- We do entropy decoding of the coded bit stream to get back Z.
- Inverse quantization on Z gives $\hat{z}[k,l] = z[k,l]q[k,l]$

$$\hat{Z} = \begin{bmatrix} 208 & 44 & 0 & 16 & 0 & 0 & 0 & 0 \\ 36 & -24 & 14 & 19 & 0 & 0 & 0 & 0 \\ 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 \\ 14 & -17 & 0 & 0 & 0 & 0 & 0 & 0 \\ -18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Imperial College London

8x8 IDCT

- IDCT on $\hat{Z}$ gives:

$$\hat{X} = \begin{bmatrix} 171 & 160 & 149 & 149 & 158 & 166 & 166 & 162 \\ 174 & 164 & 155 & 154 & 160 & 164 & 161 & 156 \\ 171 & 164 & 157 & 156 & 158 & 158 & 151 & 145 \\ 161 & 157 & 154 & 154 & 155 & 151 & 144 & 137 \\ 156 & 155 & 155 & 156 & 156 & 152 & 145 & 140 \\ 159 & 160 & 160 & 160 & 157 & 153 & 148 & 145 \\ 161 & 161 & 160 & 156 & 150 & 144 & 141 & 139 \\ 159 & 158 & 155 & 148 & 139 & 132 & 129 & 128 \end{bmatrix}$$

- Observe that $\hat{X} \neq X$.

Imperial College London

DCT Based Coding Example – High activity region

The input block (labelled **original**) is taken from a high activity region; that is, there are essential differences among pixel values in that area.

$$X = \begin{bmatrix} 197 & 184 & 144 & 103 & 130 & 133 & 70 & 51 \\ 200 & 158 & 111 & 141 & 179 & 151 & 70 & 73 \\ 172 & 110 & 111 & 179 & 192 & 135 & 95 & 144 \\ 118 & 77 & 139 & 193 & 156 & 102 & 128 & 193 \\ 73 & 75 & 151 & 163 & 110 & 84 & 154 & 197 \\ 54 & 84 & 142 & 122 & 73 & 90 & 160 & 162 \\ 50 & 95 & 130 & 71 & 52 & 101 & 146 & 117 \\ 68 & 115 & 106 & 55 & 63 & 116 & 118 & 72 \end{bmatrix}$$

Imperial College London

DCT Based Coding Example – High activity region

$$Y = \begin{bmatrix} -60 & -5 & -14 & -38 & 17 & 15 & 15 & 7 \\ 127 & 139 & -40 & 103 & 102 & -41 & 12 & -13 \\ -76 & 123 & 22 & 110 & -105 & -46 & 1 & -8 \\ -20 & -5 & 29 & -53 & -54 & 18 & -1 & 11 \\ -4 & 4 & 5 & -25 & -6 & 4 & 2 & 5 \\ -3 & -10 & 9 & -19 & -5 & 8 & 7 & 6 \\ 3 & 0 & 1 & 1 & 4 & 0 & -1 & -2 \\ -1 & -4 & 5 & -4 & -2 & -2 & 1 & 4 \end{bmatrix}$$

$$\hat{X} = \begin{bmatrix} 198 & 182 & 153 & 136 & 145 & 145 & 95 & 32 \\ 182 & 159 & 146 & 153 & 152 & 129 & 98 & 81 \\ 153 & 124 & 135 & 174 & 159 & 105 & 104 & 150 \\ 120 & 95 & 125 & 180 & 153 & 86 & 112 & 203 \\ 88 & 84 & 120 & 159 & 130 & 81 & 121 & 211 \\ 62 & 93 & 120 & 114 & 92 & 92 & 131 & 173 \\ 45 & 112 & 123 & 64 & 52 & 110 & 139 & 114 \\ 37 & 127 & 126 & 31 & 27 & 123 & 143 & 72 \end{bmatrix}$$

Imperial College London

Huffman Coding of DC Coefficients

- Let DC_i and DC_{i-1} denote the DC coefficients of blocks i and $i-1$.
- Due to the high correlation of DC values among adjacent blocks, JPEG uses differential coding for the DC coefficients.
- $(DC_i - DC_{i-1}) \in [-2047, 2047]$; this range is divided into 12 size categories. Go back to Slide 24: this range is Category 11.
- Each DC differential can be described by the pair (**size**, **amplitude**).
- From this pair of values, only the first (**size**) is Huffman coded.

-- This process is identical to residual coding mentioned before, where residuals are DC differences among adjacent blocks.

Imperial College London

Example

The DC differential has an amplitude of 195.

- $195 = 128 + 64 + 2 + 1 =$
- $195_{10} = (128 + 64 + 2 + 1)_{10} = (2^7 + 2^6 + 2^1 + 2^0)_{10} = (11000011)_2$
- $\text{size} = 8$.
- Thus, 195 is described by the pair $(8, \mathbf{11000011})$.
- If the Huffman codeword for $\text{size} = 8$ is $\mathbf{111110}$, then 195 is coded as $\mathbf{111110\ 11000011}$.
- Similarly, −195 would be coded as $\mathbf{111110\ 00111100}$.
- Huffman decoding is obvious.

Imperial College London

Huffman Coding of AC Coefficients

After quantization, most of the AC coefficients will be zero; thus, only the nonzero AC coefficients need to be coded.

- AC $\in [-1023, 1023]$; this range is divided into 10 size categories.
- Each AC differential can be described by the pair (**run/size**, **amplitude**).
- From this pair of values, only the first (**run/size**) is Huffman coded.
- **Size** is what we called before **category**.

Imperial College London

Example

Assume an AC coefficient is preceded by **six zeros** and has a value of -18.

- $18 = 16 + 2 =$
- $18_{10} = (16 + 2)_{10} = (2^4 + 2^1)_{10} = (10010)_2$
- $-18_{10} = (01101)_2$
- -18 falls into category 5.
- Hence, this coefficient is represented by $(6/5, \mathbf{01101})$.
- The pair $(6/5)$ is Huffman coded, and the bit value of -18 is appended to that code.
- If the Huffman codeword for $(6/5)$ is $\mathbf{1101}$, then the codeword for 6 zeros followed by -18 is $\mathbf{1101}\ \mathbf{01101}$.

Imperial College London

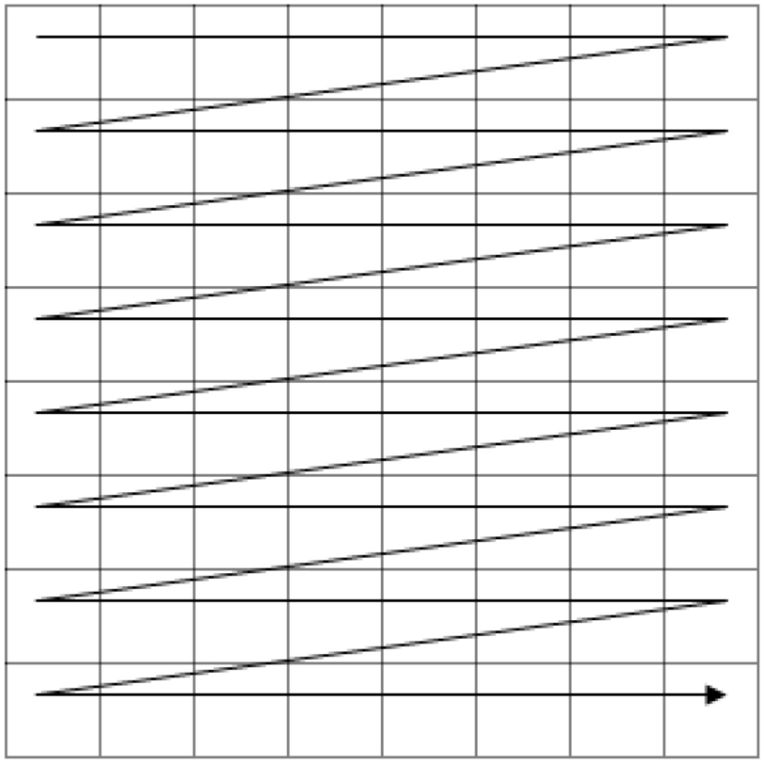
Special cases

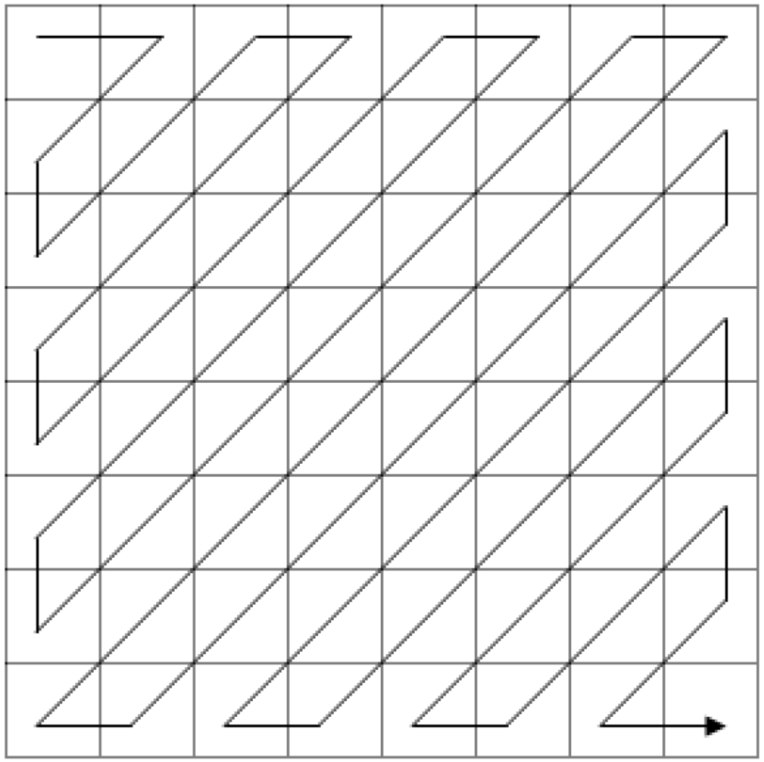
The run-length value cannot be larger than 15.

- In that case, JPEG uses the symbol $(15/0)$ to denote a run-length of 15 zeros followed by a zero.
- Such symbols can be cascaded as needed: **however, the codeword for the last AC coefficient must have a non zero amplitude.**
- If after a nonzero AC value all the remaining coefficients are zero, then the special symbol $0/0$ denotes an end of block (EOB).

Imperial College London

Conventional and zig-zag ordering


conventional order


zig-zag order

Imperial College London

Conventional and zig-zag ordering

Assume that the values of a quantized DCT matrix are given by:

$$\begin{matrix} 42 & 16 & -21 & 10 & -15 & 0 & 0 & 0 \\ 3 & -2 & 0 & 2 & -3 & 0 & 0 & 0 \\ 0 & 0 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

- If the DC value of the previous block is 40, then $DC_i - DC_{i-1} = 2$.
- This can be expressed as the (**size**, **amplitude**) pair $(2,2)$.
- If the Huffman codeword for size 2 is 011, then the codeword for the DC value is 01110.

Imperial College London

A coding example (cont.): AC

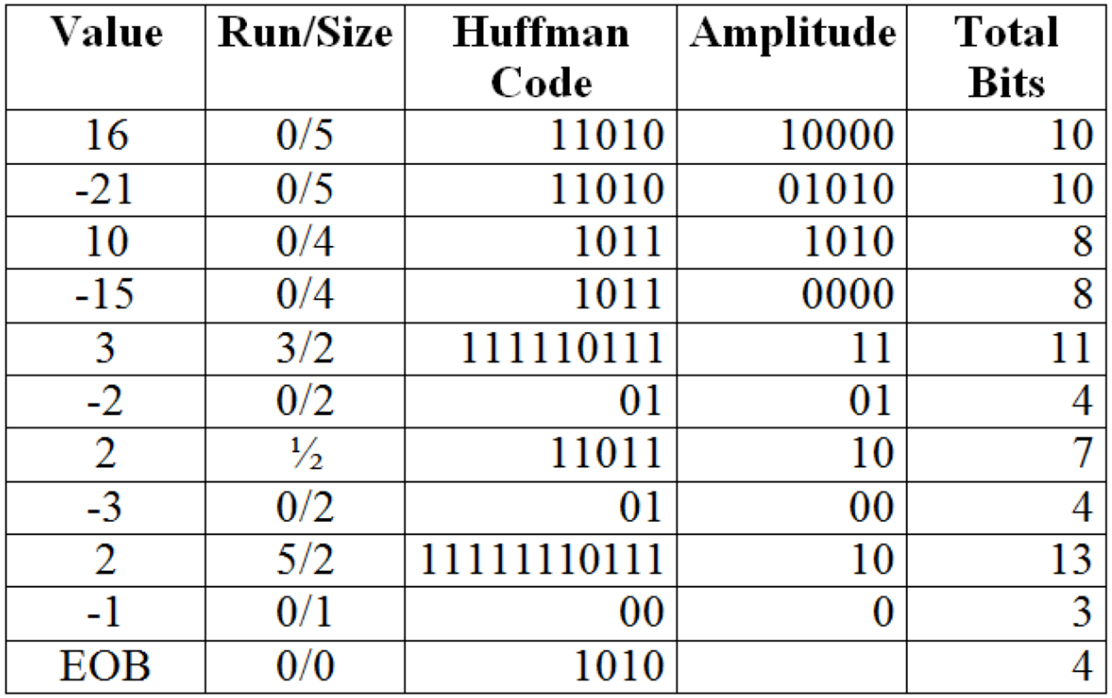
$$\begin{matrix} 42 & 16 & -21 & 10 & -15 & 0 & 0 & 0 \\ 3 & -2 & 0 & 2 & -3 & 0 & 0 & 0 \\ 0 & 0 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Value	Run/Size	Huffman Code	Amplitude	Total Bits
16	0/5	11010	10000	10
-21	0/5	11010	01010	10
10	0/4	1011	1010	8
-15	0/4	1011	0000	8
3	3/2	111110111	11	11
-2	0/2	01	01	4
2	½	11011	10	7
-3	0/2	01	00	4
2	5/2	11111110111	10	13
-1	0/1	00	0	3
EOB	0/0	1010		4

A coding example (cont.): AC

We require on total 82 bits to encode the AC coefficients.

- We require 5 bit to encode the DC coefficients.
- Average bit rate is $\frac{87}{64} = 1.36$ bits per pixel.
- Compression ratio is $\frac{8}{1.36} = 5.88$.



Value	Run/Size	Huffman Code	Amplitude	Total Bits
16	0/5	11010	10000	10
-21	0/5	11010	01010	10
10	0/4	1011	1010	8
-15	0/4	1011	0000	8
3	3/2	111110111	11	11
-2	0/2	01	01	4
2	½	11011	10	7
-3	0/2	01	00	4
2	5/2	11111110111	10	13
-1	0/1	00	0	3
EOB	0/0	1010		4

Imperial College London

JPEG Summary

The DC and AC coefficients are treated separately. This is motivated by the fact that the statistics for the DC and AC coefficients are quite dissimilar.

- Many of the AC coefficients within a block will be zero-valued.
- Values for the DC differentials range between -2047 and 2048, and for the AC coefficients range between -1023 and 1024.
- **Direct Huffman coding of these values would require code tables with $\mathbf{2048 \cdot 2 = 4096}$ and $\mathbf{1024 \cdot 2 = 2048}$ entries.**
- By Huffman coding only the **size** or the (**run/size**) information, the size of these tables is reduced to 12 and 162 entries, respectively.