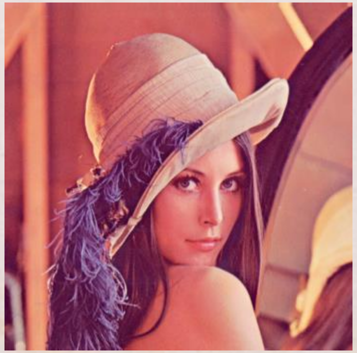
How does work JPEG?

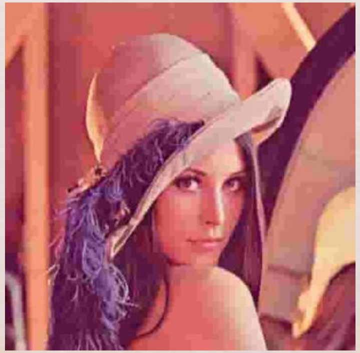
JPEG compression

- JPEG Compression is lossy, such that the original image cannot be exactly reconstructed (although a "Lossless JPEG" specification exits as well).

- JPEG exploits the caracteristics od human vision, eliminating or reducing data to which the eye is less sensitive. JPEG works well on grayscale and color images, especially on photographs, but is not intended for two-tone images.


original: 104 KB

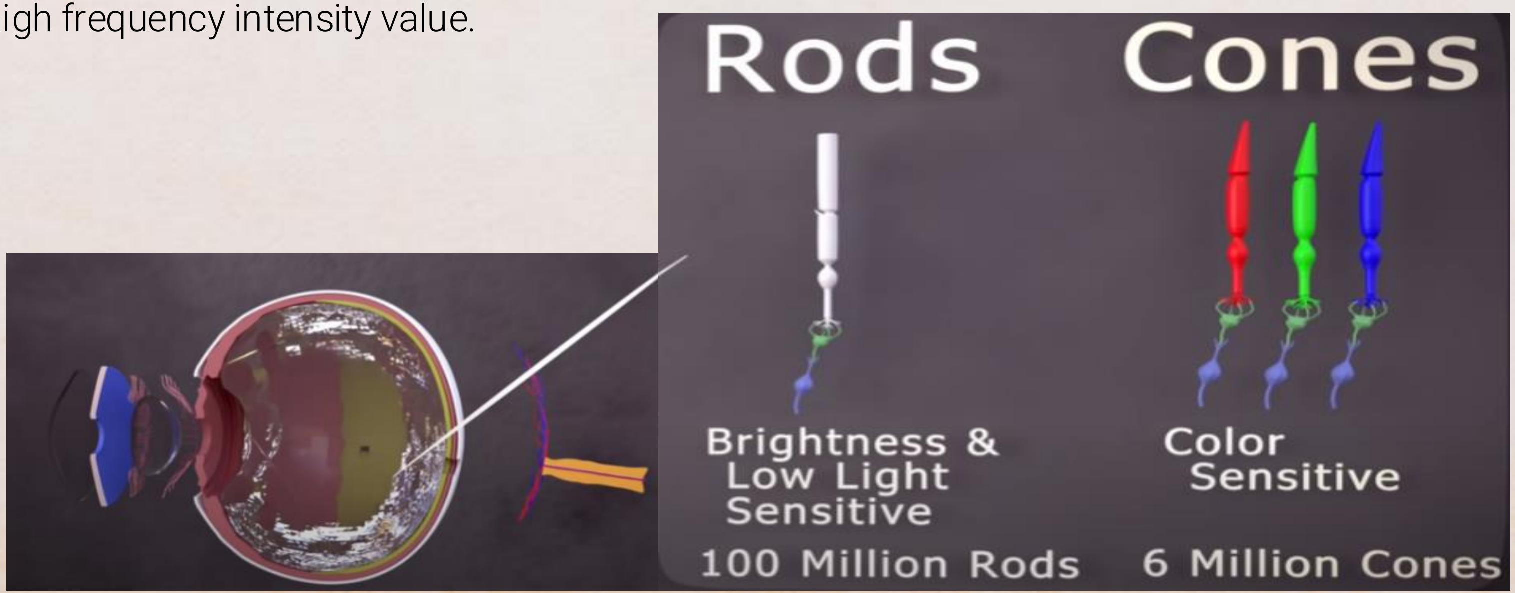
compressed: 92 KB(-11%)


crompressed: 5 KB(-95%)

Idea behind compression (JPEG)

- Our eyes is highly sensitive to intensity(grayscale) information and less sensitive to color information.

- Again we are more sensitive to low frequency intensity value and less sensitive to high frequency intensity value.



Steps of compression

01 color Space conversion

02 chrominance Downsampling

03 Discrete cosine Transformation

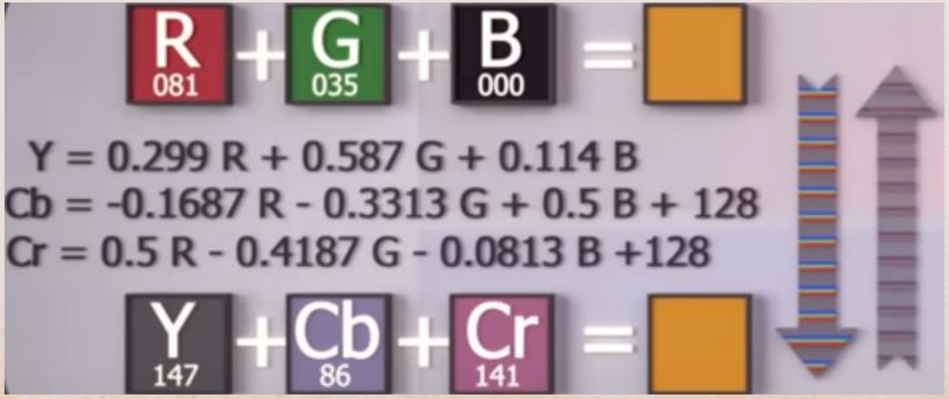
04 Quantization

05 Run Length ,Huffman Encoding and Aritmetic encoding

color Space conversion

- Calculates 3 new luminance values:
  - Blue chrominance
  - Red chrominance
  - Y luminance

- This process is reversible and there is no loss of date

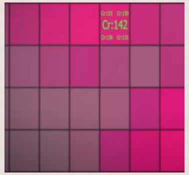


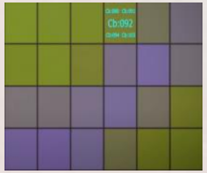
$$Y = 0.299\,R + 0.587\,G + 0.114\,B$$
$$Cb = -0.1687\,R - 0.3313\,G + 0.5\,B + 128$$
$$Cr = 0.5\,R - 0.4187\,G - 0.0813\,B + 128$$

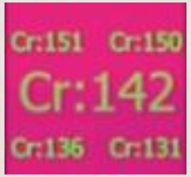
Chrominance Downsampling

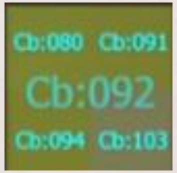
- In the next step we remove a considerable amount of data, remember that our eyes are bad at detecting color or prominance versus brightness or luminance.

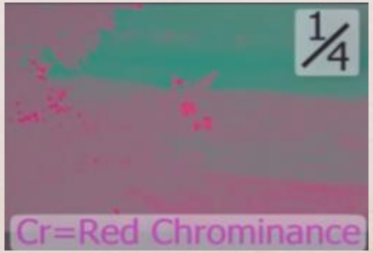
- In general, jpeg down sample by 2 in each direction. Means that the color information is down sampled by 4 (4 time less color than the original one).

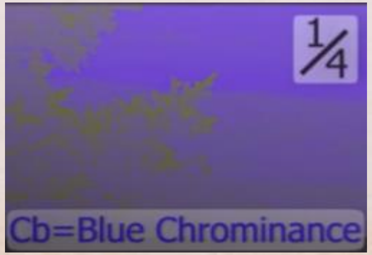


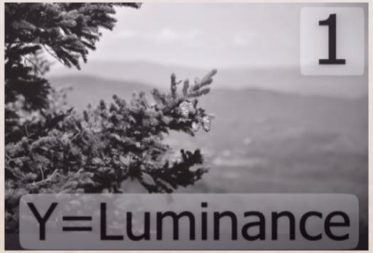










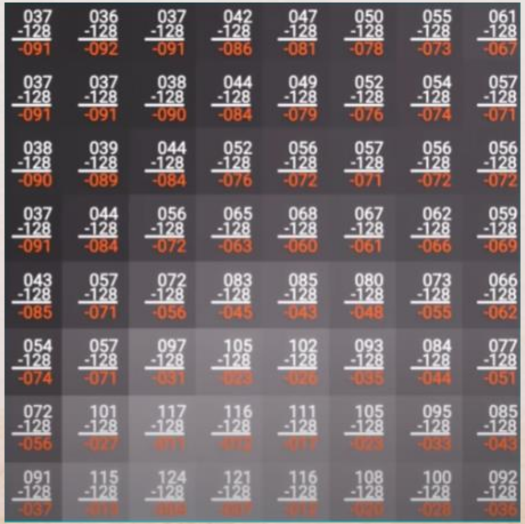


Before: 1+1+1=3.0

After: 1/4+1/4+1=1.5

Preprocessing Discrete cosine Transformation

- First the image is separated into non-overlapping blocks where each block is 8 X 8 pixels(64 pixels per block).
- Then normalize the pixels value to -128 to 127 from 0 to 255, just by subtracting 128 from each pixel value.



037 -128 -091	036 -128 -092	037 -128 -091	042 -128 -086	047 -128 -081	050 -128 -078	055 -128 -073	061 -128 -067
037 -128 -091	037 -128 -091	038 -128 -090	044 -128 -084	049 -128 -079	052 -128 -076	054 -128 -074	057 -128 -071
038 -128 -090	039 -128 -089	044 -128 -084	052 -128 -076	056 -128 -072	057 -128 -071	056 -128 -072	056 -128 -072
037 -128 -091	044 -128 -084	056 -128 -072	065 -128 -063	068 -128 -060	067 -128 -061	062 -128 -066	059 -128 -069
043 -128 -085	057 -128 -071	072 -128 -056	083 -128 -045	085 -128 -043	080 -128 -048	073 -128 -055	066 -128 -062
054 -128 -074	057 -128 -071	097 -128 -031	105 -128 -023	102 -128 -026	093 -128 -035	084 -128 -044	077 -128 -051
072 -128 -056	101 -128 -027	117 -128 -011	116 -128 -012	111 -128 -017	105 -128 -023	095 -128 -033	085 -128 -043
091 -128 -037	115 -128 -013	124 -128 -004	121 -128 -007	116 -128 -012	108 -128 -020	100 -128 -028	092 -128 -036

Discrete cosine Transformation

- DCT creates a new representation of image where it is easy to separate important and less important pixels of image.

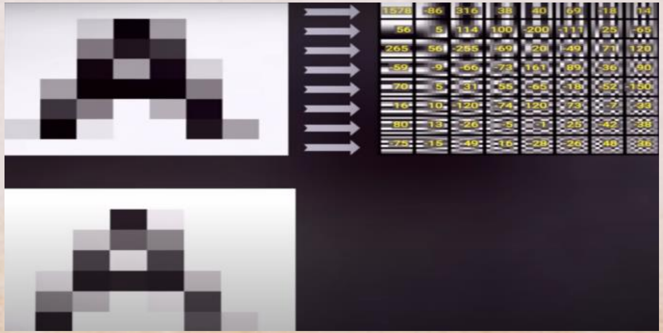
- Each "grid" or pattern corresponds to a specific type of variation:

  The first grid (in the top-left corner) represents uniform areas, meaning very low frequencies.
  Grids further down and to the right represent faster and more complex variations, meaning higher frequencies.

- Multiplication by coefficients:

  Each of these grids has a weight or coefficient that indicates how important that frequency pattern is to the original block.

  What happens is that each pattern (grid) is multiplied by its coefficient and then summed up to reconstruct the image.

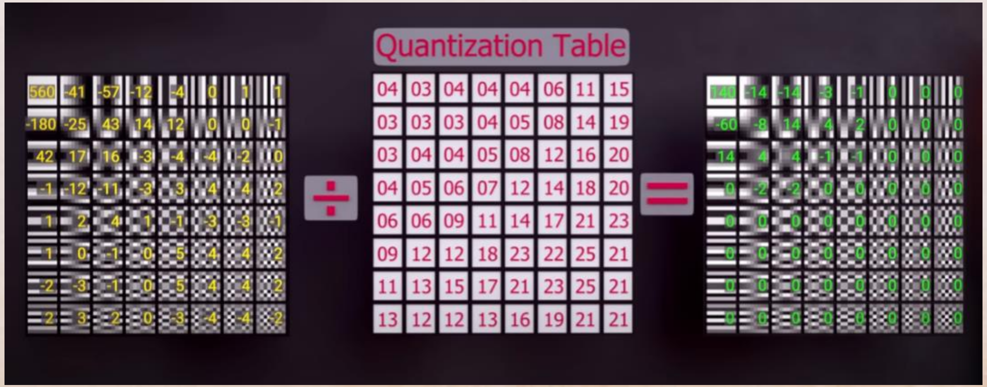


Quantization

- This is where the data is compressed in great .After performing DCT, we apply quantization on it to remove the high frequency components. This is the real data comprehension part of jpeg.

- For quantizing the DCT transformed data, jpeg use a predefined quantization table. The amount of comprehension is dependent on the choice of quantization table.

- This quantization is non reversible meaning that we can't recover the original data without loss.



560	-41	-57	-12	-4	0	1	1
-180	-25	43	14	12	0	0	-1
42	17	16	-3	-4	-4	-2	0
-1	-12	-11	-3	3	4	4	2
1	2	4	1	-1	-3	-3	-1
1	0	-1	0	5	4	4	2
-2	-3	-1	0	5	4	4	2
2	3	-2	0	-3	-4	-4	-2

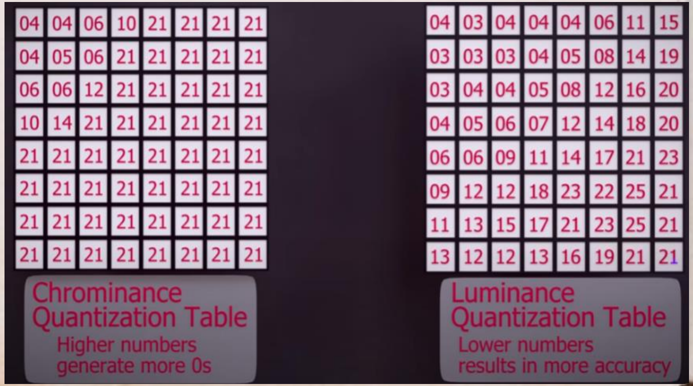
÷

Quantization Table

04	03	04	04	04	06	11	15
03	03	03	04	05	08	14	19
03	04	04	05	08	12	16	20
04	05	06	07	12	14	18	20
06	06	09	11	14	17	21	23
09	12	12	18	23	22	25	21
11	13	15	17	21	23	25	21
13	12	12	13	16	19	21	21

=

140	-14	-14	-3	-1	0	0	0
-60	-8	14	4	2	0	0	0
14	4	4	-1	-1	0	0	0
0	-2	-2	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



04	04	06	10	21	21	21	21
04	05	06	21	21	21	21	21
06	06	12	21	21	21	21	21
10	14	21	21	21	21	21	21
21	21	21	21	21	21	21	21
21	21	21	21	21	21	21	21
21	21	21	21	21	21	21	21
21	21	21	21	21	21	21	21

Chrominance Quantization Table
Higher numbers generate more 0s

04	03	04	04	04	06	11	15
03	03	03	04	05	08	14	19
03	04	04	05	08	12	16	20
04	05	06	07	12	14	18	20
06	06	09	11	14	17	21	23
09	12	12	18	23	22	25	21
11	13	15	17	21	23	25	21
13	12	12	13	16	19	21	21

Luminance Quantization Table
Lower numbers results in more accuracy

Zig Zag Scanning

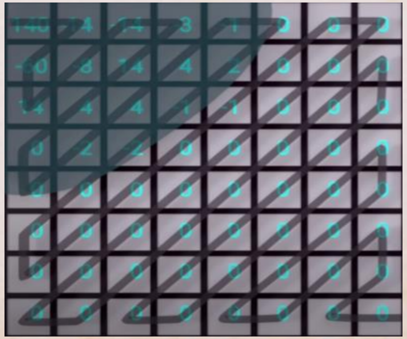
- The zigzag ordering groups non-zero coefficients (important information) at the beginning of the sequence and pushes all the zeros (less important information) to the end.

- This makes it easier to apply further compression methods, like Run-Length Encoding (RLE), which can represent long sequences of zeros efficiently.

- The zigzag path (highlighted in the image) starts at the top-left (low frequencies) and gradually moves toward the bottom-right (high frequencies).

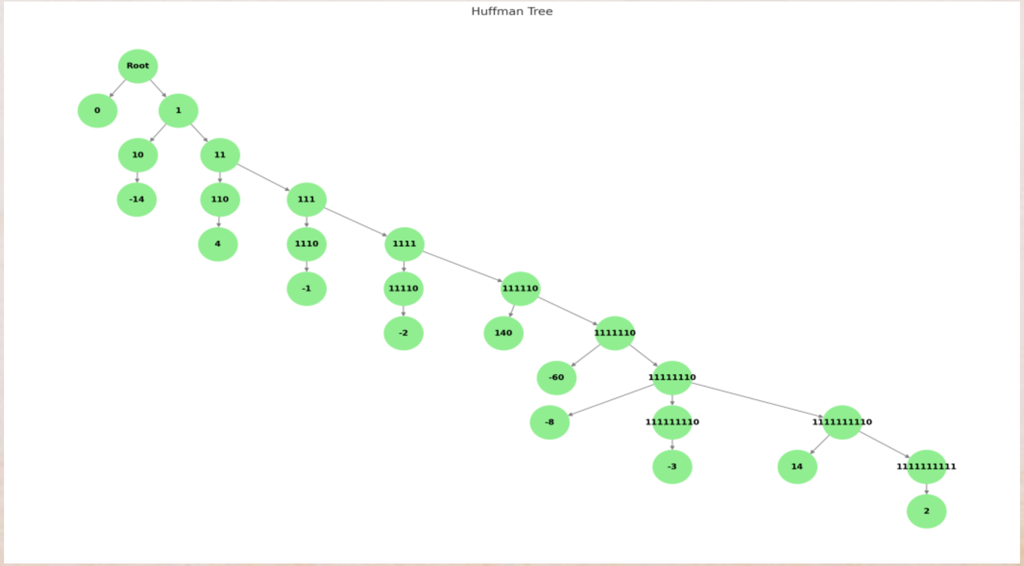
- Run Length Encoding

140,-14,-60.-14,-8,-14,-3,14,4,0[x2],-2,4[x2],-1,0,2,-1,
-2,0[x6],-1,0[x38]



Huffman coding

140, -14, -60, -14, -8, -14, -3, 14, 4, 0, 0, -2, 4, 4,
-1, 0, 2, -1, -2, 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



Value	Frequency
0	40
-14	3
4	3
-1	2
-2	2
140	1
-60	1
-8	1
-3	1
14	1
2	1

Huffman coding

140, -14, -60, -14, -8, -14, -3, 14, 4, 0, 0, -2, 4, 4, -1, 0,
2, -1, -2, 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

111110 10 1111110 10 11111110 10 111111110 1111111110
110 0 0 11110 110 110 1110 0 1111111111 1110 11110
0 0 0 0 0 0 1110 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

Original: 50 values * 8 bits = 400 bits

compressed: (1 x 6) + (3 x 2) + (1 x 7) + (1 x 8) + (1 x 9) + (1 x 10) +
(3 x 3) + (40 x 1) + (2 x 5) + (2 x 4) + (1 x 10) = 123 bits

Value	Huffman Code
0	0
-14	10
4	110
-1	1110
-2	11110
140	111110
-60	1111110
-8	11111110
-3	111111110
14	1111111110
2	1111111111

Arithmetic coding

Value	Frequency	Probability
0	40	40/50 =0.8
-14	3	3/50=0.06
4	3	3/50=0.06
-1	2	2/50=0.04
-2	2	2/50=0.04
140	1	1/50=0.02
-60	1	1/50=0.02
-8	1	1/50=0.02
-3	1	1/50=0.02
14	1	1/50=0.02
2	1	1/50=0.02

Value	Cumulative range
0	[0.00, 0.80)
-14	[0.80, 0.86)
4	[0.86, 0.92)
-1	[0.92, 0.96)
-2	[0.96, 1.00)
140	[0.00, 0.02)
-60	[0.02, 0.04)
-8	[0.04, 0.06)
-3	[0.06, 0.08)
14	[0.08, 0.10)
2	[0.10, 0.12)

Arithmetic coding

- Instead of assigning fixed-length codes to symbols, arithmetic coding represents an entire message as a single number between 0 and 1. It continuously subdivides the range [0, 1) based on symbol probabilities.

First Few Symbols: 140, -14, -60,

Step 1: Start with [0.00,1.00)

Step 2: Encode 140:Range of 140: [0.00, 0.02)

New range =[0.00+0.00x(1.00—0.00),0.00+0.02x(1.00—0.00)]=[0.00,0.02)

Step 3: Encode -14:Range of -14: [0.80, 0.86)

New range = [0.00+0.80x(0.02—0.00),0.00+0.86x(0.02—0.00)]=[0.016,0.0172)

Step 4: Encode -60:Range of -60: [0.02, 0.04)

New range = [0.016+0.02x(0.0172—0.016),0.016+0.04x(0.0172—0.016)]=[0.01604,0.01608)

JPEG 2000 VS JPEG

Feature	JPEG 2000	JPEG
Compression Quality	Higher, fewer artifacts	Lower, with noticeable blockiness
Compression Modes	Lossless & Lossy	Lossy only
Progressive Decoding	Yes	Yes (but less efficient)
Error Resilience	Better	Poor
Bit Depth Support	Up to 16 bits	8 bits
File Size	Smaller for same quality	Larger for same quality

Applications of JPEG in Digital Photography and Web Images

Digital Photography:

- High compression Efficiency;
- compatibility;
- Storage on Memory cards;
- Quick Sharing;
- common use case;

Web images:

- Efficient Web Page Loading;
- High-Quality Visuals;
- Universal Browser Support;
- Responsive and Scalable Design;
- Typical use cases;