In our last tutorial of image compression, we discuss some of the techniques used for compression. We are going to discuss JPEG compression which is lossy compression, as some data is loss in the end.

Let’s discuss first what image compression is.

**Image compression**

Image compression is the method of data compression on digital images. The main objective in the image compression is:

- Store data in an efficient form
- Transmit data in an efficient form

Image compression can be lossy or lossless.

**JPEG compression**

JPEG stands for Joint photographic experts group. It is the first international standard in image compression. It is widely used today. It could be lossy as well as lossless. But the technique we are going to discuss here today is lossy compression technique.

**How jpeg compression works**

First step is to divide an image into blocks with each having dimensions of 8 x 8.

Let’s for the record, say that this 8x8 image contains the following values.

```
52 55 61 66 70 61 64 73
63 59 55 90 109 85 69 72
62 59 68 113 144 104 66 73
63 58 71 122 154 106 70 69
67 61 68 104 126 88 68 70
79 65 60 70 77 68 58 75
85 71 64 50 55 61 65 83
```
The range of the pixel intensities now are from 0 to 255. We will change the range from -128 to 127.

Subtracting 128 from each pixel value yields pixel value from -128 to 127. After subtracting 128 from each of the pixel value, we got the following results.

Now we will compute using this formula.

\[
G_{u,v} = \alpha(u)\alpha(v) \sum_{x=0}^{7} \sum_{y=0}^{7} g_{x,y} \cos \left[ \frac{\pi}{8} \left( x + \frac{1}{2} \right) u \right] \cos \left[ \frac{\pi}{8} \left( y + \frac{1}{2} \right) v \right]
\]

\[
\alpha_p(n) = \begin{cases} 
\sqrt{\frac{1}{8}}, & \text{if } n = 0 \\
\sqrt{\frac{2}{8}}, & \text{otherwise}
\end{cases}
\]

The result comes from this is stored in let's say \(A_{j,k}\) matrix.

There is a standard matrix that is used for computing JPEG compression, which is given by a matrix called as Luminance matrix.

This matrix is given below.

\[
Q_{j,k} = \begin{bmatrix}
16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\
12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\
14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\
14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\
18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\
24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\
\end{bmatrix}
\]
Applying the following formula

\[ B_{j,k} = \text{round} \left( \frac{A_{j,k}}{Q_{j,k}} \right) \]

We got this result after applying.

\[
B_{j,k} = \begin{bmatrix}
-26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\
0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\
-3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\
-4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

Now we will perform the real trick which is done in JPEG compression which is ZIG-ZAG movement. The zig zag sequence for the above matrix is shown below. You have to perform zig zag until you find all zeroes ahead. Hence our image is now compressed.

\[
B_{j,k} = \begin{bmatrix}
-26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\
0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\
-3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\
-4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

Summarizing JPEG compression

The first step is to convert an image to Y’CbCr and just pick the Y’ channel and break into 8 x 8 blocks. Then starting from the first block, map the range from -128 to 127. After that you have to find the discrete Fourier transform of the matrix. The result of this should be quantized. The last
step is to apply encoding in the zig zag manner and do it till you find all zero.

Save this one dimensional array and you are done.

**Note. You have to repeat this procedure for all the block of 8 x 8.**