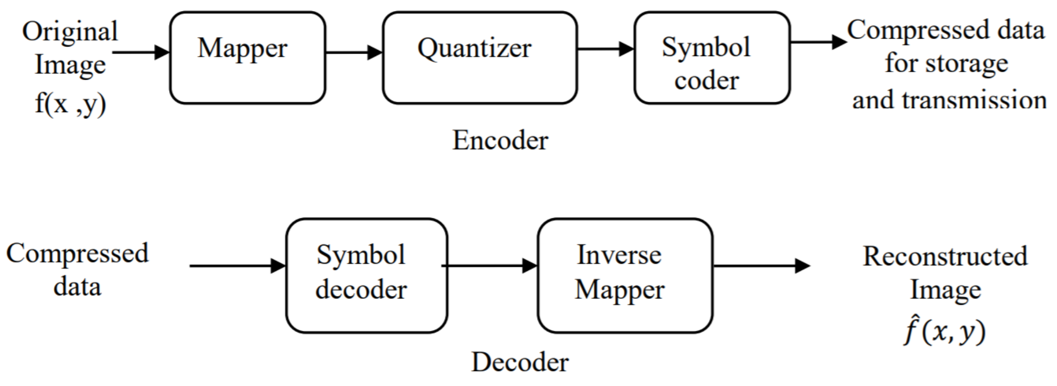
Introduction of Image Compression

The digital representation of images usually require a very large number of bits. In many applications, it is important to consider techniques for representing an image, or information contained in the image, with fewer bits while maintaining an acceptable fidelity of image quality.

The purpose of compression system is to shrink the number of bits to the possible extent, while keeping the visual quality of the reconstructed image as close to the original image. The following figure shows the basic block diagram of a general image compression system. It consists of an encoder block and a decoder block.



General image compression system

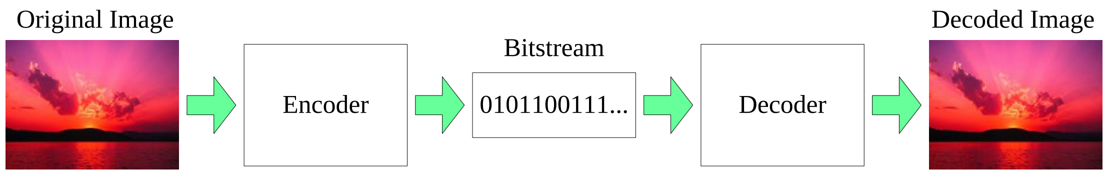


Fig. show the basic flow of image compression coding

The benefit of image compression are as follows:

1. Reducing the storage requirement or saving the storage space.
2. Potential cost saving associated with sending less data over the communication lines where the cost of call is usually based upon its duration.
3. Compression can reduce the probability of transmission error occurring since fewer characters are transmitted when data is compressed.
4. By converting the original data that is represented by conventional code into a different code, compression algorithm may provide a level of security illicit monitoring.
5. Reducing the time required for transmission of the total original image by transmitting its compressed version.

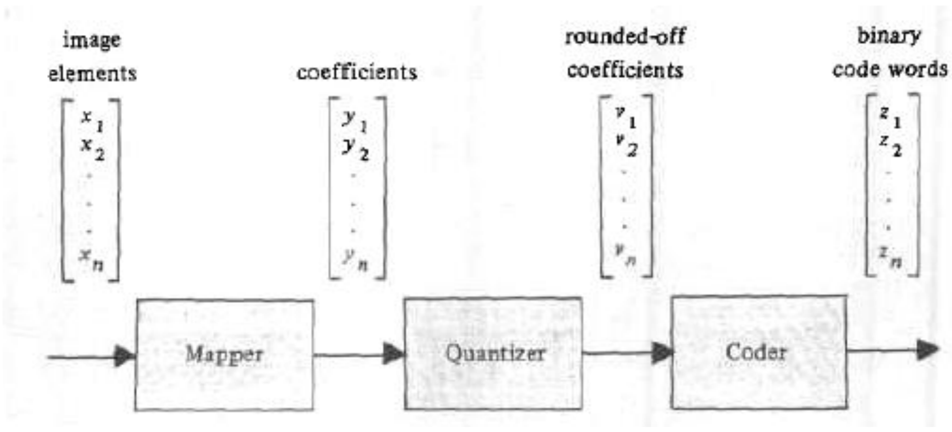
Lossless and Lossy Compression

There are two kinds of compression method; lossless and lossy compression. Both methods save storage space but have different results.
The term “lossless” means that the compressed file can be reconstructed to match the original, while the latter “lossy” means that there is some loss of information in the reconstructed image, where the loss of small amounts of data does not reduce the perceived over all quality of the output naturally.

There are many kinds of lossless compression technique such as Huffman coding , Arithmetic coding, Run Length Encoding (RLE), and Welsh (LZW) coding. Each one of these techniques is useful for both image and data compression. Also, There are many kinds of lossy compression technique, such as Vector Quantization, JPEG, MPEG. Each one of these techniques is useful for image compression, and is often based on the special transform such as Fourier transform, Fast Fourier Transform (FFT), Discrete Cosine transform (DCT), and Wavelet transform.

The Encoding Process

Encoders can be modeled as a sequence of three operations, as illustrated in the following figure. Where images are expressed in vector form. The mapping operation maps the input data from the pixel domain in to another domain where the quantizer and coder can be used more efficiently in the sense that fewer bits are required to code the mapped data than would be required to code the original input data.



An Encoder model

Huffman Coding Algorithm

A Huffman code can be constructed by first ordering the input probabilities, the two smallest probabilities are combined by addition to form a new set of probabilities. The new set of probabilities which has one fewer probability than the original set is again ordered and when we get down to two probabilities we stop. Code words are generated by starting at the last step and working backwards. We start by assigning a *0* to one of the last two combined probabilities and a *1* to the other.

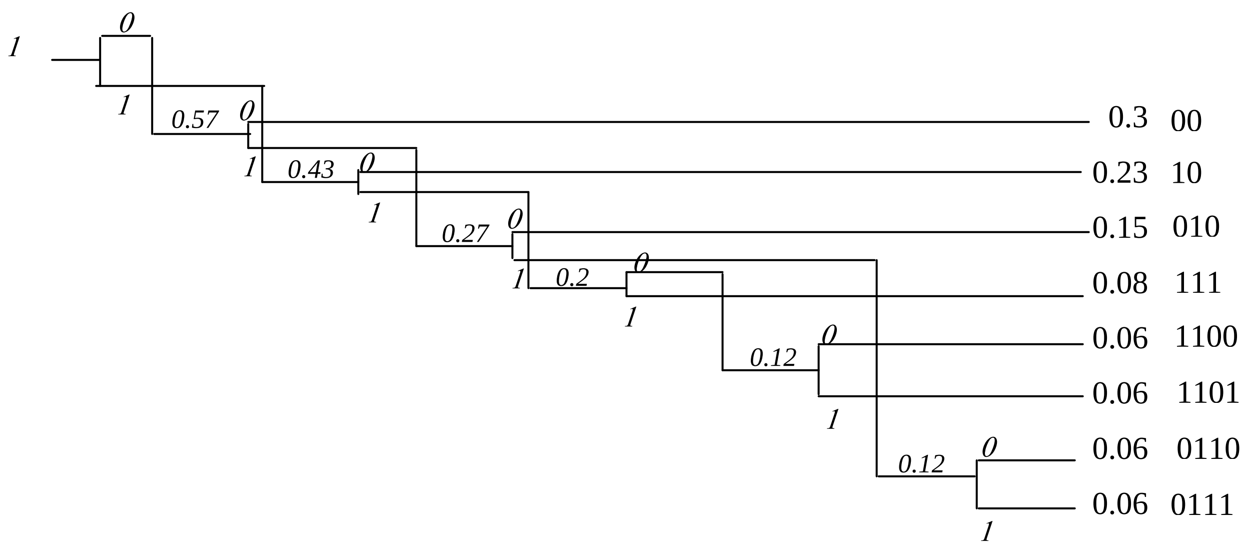
The more frequently occurring symbols can be allocated with shorter codewords than the less frequently occurring symbols.

The two least frequently occurring symbols will have code words of the same length and they differ only in the least significant bit.

Example: *Use Huffman encoding algorithm to encode (compress) an image with 8gray levels, when the probabilities of graylevels are as follows:*

Inputs

Graylevel	*probability*
0	*0.3*
1	*0.23*
2	*0.15*
3	*0.08*
4	*0.06*
5	*0.06*
6	*0.06*
7	*0.06*



Inputs	*Probabilities*	*Huffman code*
0	*0.3*	*00*
1	*0.23*	*10*
2	*0.15*	*010*
3	*0.08*	*111*
4	*0.06*	*1100*
5	*0.06*	*1101*
6	*0.06*	*0110*
7	*0.06*	*0111*

Advantages of Huffman Encoding:

- This encoding scheme results in saving lot of storage space, since the binary codes generated are variable in length.
- It generates shorter binary codes for encoding symbols/characters that appear more frequently in the input string.
- The binary codes generated are prefix-free.

Disadvantages of Huffman Encoding:

- Lossless data encoding schemes, like Huffman encoding, achieve a lower compression ratio compared to lossy encoding techniques. Thus, lossless techniques like Huffman encoding are suitable only for encoding text and program files and are unsuitable for encoding digital images.
- Huffman encoding is a relatively slower process since it uses two passes- one for building the statistical model and another for encoding. Thus, the lossless techniques that use Huffman encoding are considerably slower than others.
- Since length of all the binary codes is different, it becomes difficult for the decoding software to detect whether the encoded data is corrupt. This can result in an incorrect decoding and subsequently, a wrong output.

Real-life applications of Huffman Encoding:

- Huffman encoding is widely used in compression formats like `GZIP, PKZIP (winzip) and BZIP2`.
- Multimedia codecs like `JPEG, PNG and MP3` uses Huffman encoding.
- Huffman encoding still dominates the compression industry since newer arithmetic and range coding schemes are avoided due to their patent issues.