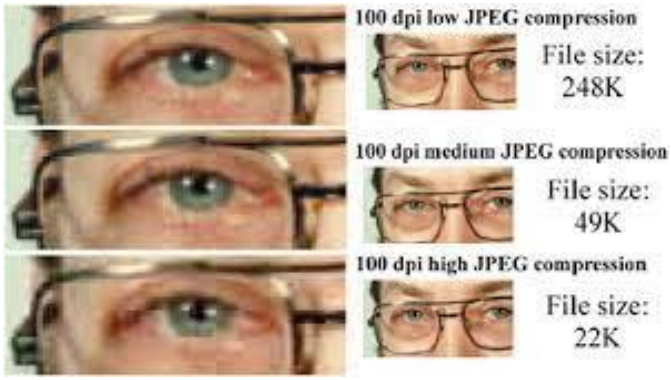
Unit V:IMAGE COMPRESSION

Image compression is a technique used to reduce the size of image files while maintaining an acceptable level of visual quality. It is an important process in various applications, including digital photography, web design, video streaming, and more. Image compression is especially crucial when dealing with large numbers of images, as it helps save storage space and accelerates data transmission.



In the digital age, where images are an integral part of communication, storage, and entertainment, the concept of image compression has emerged as a fundamental technology. Image compression is the art and science of reducing the size of image files while retaining their visual quality to a reasonable extent. This technological feat holds immense significance in a world where data is generated and exchanged at an unprecedented rate.

- **The need** for image compression arises from the inherent data-rich nature of images. High-resolution images contain an abundance of pixels, each storing color and intensity information. Transmitting or storing such images without optimization can be impractical due to the substantial storage space and bandwidth requirements. Here steps in image compression, offering innovative algorithms and techniques that strike a balance between efficient data representation and faithful image reproduction.

There are two main types of image compression: lossless compression and lossy compression.

- Lossless Compression: In lossless compression, the image is compressed without any loss of information. This means that when the compressed image is decompressed, it is identical to the original image. Common lossless compression formats include PNG (Portable Network Graphics) and GIF (Graphics Interchange Format). These formats use techniques such as entropy coding and predictive coding to reduce the file size.
- Lossy Compression: Lossy compression achieves higher levels of compression by discarding some of the image data. This can result in a reduction in image quality, especially when using aggressive compression settings. However, modern lossy compression algorithms are designed to minimize perceptual loss, meaning that the loss of quality is often not very noticeable to the human eye. Popular lossy compression formats include JPEG (Joint Photographic Experts Group) and WebP.
- ✧ Lossless compression preserves every detail of the original image while reducing file size through sophisticated coding schemes. This makes it ideal for applications where pixel-perfect accuracy is critical, such as medical imaging and archival purposes.
- ✧ On the other hand, lossy compression achieves higher levels of compression by discarding certain image information that may not be as noticeable to the human eye. This method is widely used in scenarios like web imagery, where the focus is on maintaining an acceptable visual quality while significantly reducing file sizes.

Here are some key points about image compression:

- Compression Algorithms: Various algorithms are used to compress images, including run-length encoding, Huffman coding, and discrete cosine transform (DCT). These algorithms exploit redundancies in the image data to achieve compression.
- Applications: Image compression is used in a wide range of applications, from sharing images online and optimizing website performance to reducing the storage requirements for images on devices.
- Trade-off: There is often a trade-off between file size and image quality. Higher compression ratios lead to smaller file sizes but may result in noticeable artifacts or degradation in image quality.
- Encoding and Decoding: Image compression involves two main processes: encoding (compression) and decoding (decompression). The encoding process reduces the file size, while the decoding process reconstructs the original image from the compressed data.
- Codecs: A codec (compression-decompression) is a software or hardware tool that implements compression and decompression algorithms. Codecs are used to encode images into compressed files and decode them back to their original format.
- Chroma Subsampling: In some compression algorithms, like JPEG, chroma subsampling is used to reduce the amount of color information in the image, focusing on human visual perception to preserve image quality.

Need for image compression

The need for image compression arises from several practical considerations and challenges in various domains. Here are some key reasons why image compression is essential:

- **Storage Efficiency:** High-resolution images can occupy significant amounts of storage space. Image compression reduces the file size, allowing for more images to be stored on devices, servers, or cloud platforms without rapidly depleting available storage resources.
- **Bandwidth Conservation:** When images are transmitted over networks, whether for web browsing, social media, or video streaming, large file sizes can lead to slow loading times and congestion. Image compression minimizes the data transferred, leading to faster content delivery and improved user experiences.
- **Faster Loading Times:** In web design and online applications, large image files can lead to sluggish loading times, which can deter users and impact website engagement. Compressed images load more quickly, enhancing website performance and user satisfaction.
- **Mobile Devices:** With the proliferation of smartphones and tablets, efficient image compression becomes crucial. Mobile devices often have limited storage capacity and rely on cellular data connections. Compressed images help conserve both storage space and data usage.
- **Remote Communication:** In scenarios where images need to be transmitted over long distances, such as satellite imagery or telemedicine applications, efficient compression is vital to minimize transmission time and ensure timely data delivery.
- **Real-Time Applications:** Applications like video conferencing, online gaming, and virtual reality rely on fast data transmission. Compressed images contribute to reduced latency and smoother real-time experiences.
- **Content Sharing:** Social media platforms, messaging apps, and email services are popular avenues for image sharing. Compressed images are quicker to upload and download, making content sharing more convenient.
- **Archiving and Backup**: In archival contexts where preserving image quality is essential, lossless compression can reduce storage requirements while maintaining the original image fidelity.
- **Energy Efficiency:** In portable devices with limited battery life, transmitting or processing smaller image files consumes less energy, extending device usage time.
- **Cost Savings:** Data storage and transmission costs can be significant, especially for large-scale applications. Image compression directly translates to cost savings by reducing the need for additional storage resources and minimizing data transfer fees.
- **Environmental Impact:** As data centers and networks grow, the energy consumption associated with storing and transmitting data increases. Image compression can contribute to a more energy-efficient digital ecosystem.
- **Data Analysis:** In fields like scientific research, where large volumes of images are collected and analyzed, compression can accelerate data processing and analysis workflows.
- **Accessibility**: In regions with limited internet connectivity, compressed images enable more people to access visual content despite bandwidth constraints.

Redundancy in images

Redundancy in images refers to the presence of patterns, correlations, or repeated information within the image data. This redundancy is a fundamental characteristic of most images and is a key factor that allows for effective image compression. By identifying and exploiting this redundancy, compression algorithms can significantly reduce the amount of data required to represent the image without sacrificing its perceived quality.

There are several types of redundancy present in images:

- **Spatial Redundancy:** This type of redundancy is a result of neighboring pixels in an image often having similar or related values. In natural images, areas with smooth color gradients or uniform textures exhibit high spatial redundancy. Compression techniques like run-length encoding and predictive coding take advantage of spatial redundancy by representing repetitive patterns more efficiently.
- **Spectral Redundancy:** In color images, spectral redundancy refers to correlations between different color channels. Since neighboring pixels typically have similar color values, the information across color channels can be correlated. Color space transformations, such as converting RGB images to YUV or YCbCr color spaces, can help separate luminance (Y) and chrominance (UV) information, allowing more effective compression.
- **Temporal Redundancy:** In sequences of images or videos, temporal redundancy exists when consecutive frames are similar. Video compression exploits this redundancy by encoding only the changes between frames, reducing data duplication.
- **Statistical Redundancy:** Images often exhibit statistical patterns that can be leveraged for compression. For instance, some pixel values or combinations of values might be more common than others. Huffman coding and arithmetic coding are entropy coding techniques that take advantage of statistical redundancy to assign shorter codes to frequently occurring symbols.
- **Psychovisual Redundancy:** Human visual perception has limitations, and some details in an image might not be noticeable to the human eye. Compression algorithms use this psychovisual redundancy to discard information that is less likely to be detected by viewers.
- **Transform Redundancy:** Transform-based compression techniques, like the Discrete Cosine Transform (DCT) used in JPEG compression, exploit the fact that image data can be represented more compactly in frequency domains. The DCT transforms the image data from the spatial domain to the frequency domain, where the energy of the image is concentrated in a few coefficients, allowing for effective compression.
- **Bit Redundancy:** Bit redundancy occurs when the code used to represent data contains more bits than necessary. For example, if an image has only two possible pixel values (black and white), but the encoding uses 8 bits per pixel, there is significant bit redundancy. Bit-level compression techniques, like variable-length coding, aim to minimize this redundancy.

image compression scheme

Image compression schemes are methodologies or techniques used to reduce the size of image files while preserving the essential visual information. These schemes leverage various forms of redundancy present in images to achieve efficient compression. Here are some commonly used image compression schemes:

Lossless Compression:

- Run-Length Encoding (RLE): This scheme replaces sequences of identical pixels with a single value and a count.
- Huffman Coding: A variable-length coding technique that assigns shorter codes to frequently occurring pixel values or patterns.
- Arithmetic Coding: Similar to Huffman coding but allows for more precise representation of probabilities.
- Lempel-Ziv-Welch (LZW) Coding: A dictionary-based compression method that replaces repeated patterns with shorter codes.
- Predictive Coding: Predicts pixel values based on neighboring pixels and encodes only the prediction error.

Lossy Compression:

- Discrete Cosine Transform (DCT): Converts the image into frequency components, enabling more efficient representation of image data. Used in JPEG compression.
- Wavelet Transform: Decomposes the image into different frequency bands, allowing for localized compression. Used in JPEG2000.
- Vector Quantization: Groups similar pixel values into clusters and represents each cluster by its centroid.
- Color Subsampling: Reduces color resolution in certain channels (chrominance) while preserving luminance, often used in JPEG and MPEG compression.
- Quantization: Reduces the precision of pixel values, effectively reducing the number of possible values and saving space.
- Transform Coding: Applies a mathematical transformation to the image data, allowing for more compact representation in the transformed domain.

Hybrid Compression:

- JPEG: A widely used lossy compression scheme for images. It employs a combination of DCT, quantization, and entropy coding.
- JPEG2000: Builds on JPEG with improvements like wavelet-based compression, better quality scalability, and improved compression efficiency.
- WebP: A modern image format that uses a combination of techniques including VP8 video compression and a variant of the DCT used in JPEG.

Fractal Compression:

- Uses mathematical fractal representations to encode images as a set of self-replicating patterns.
- Well-suited for compressing certain types of images with intricate patterns and structures.

Context-Based Compression:

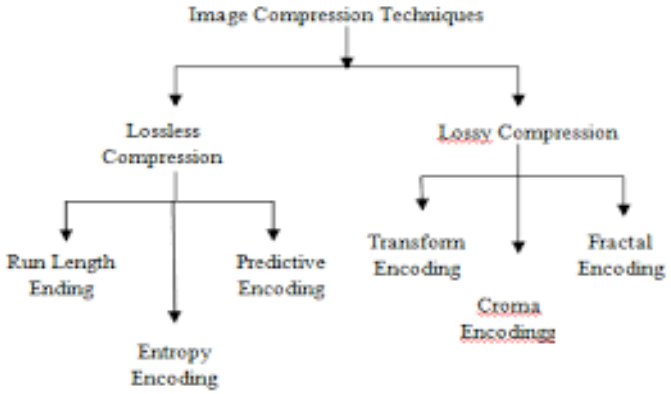
- Uses context modeling and prediction to encode image data efficiently.
- Example: Context Adaptive Binary Arithmetic Coding (CABAC) used in H.264 video compression.

Sparse Representations:

- Represents images using a sparse set of coefficients, allowing efficient storage of only the most significant information.
- Used in image compression algorithms based on compressive sensing principles.

Deep Learning-Based Compression:

- Utilizes neural networks to learn efficient image representations for compression.
- Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) are examples of architectures used in this context.



Classification of image compression schemes

Image compression schemes can be classified into two main categories based on their approach and the level of data loss involved: Lossless Compression and Lossy Compression.

Lossless Compression:

Lossless compression methods aim to reduce the file size of an image without losing any information. The decompressed image is an exact replica of the original. Common lossless compression schemes include:

- Run-Length Encoding (RLE)
- Huffman Coding
- Arithmetic Coding
- Burrows-Wheeler Transform (BWT)
- Lempel-Ziv-Markov Chain Algorithm (LZMA)
- Predictive Coding
- Bitplane Coding

Lossy Compression:

Lossy compression methods achieve higher levels of compression by removing some image information that is less noticeable to the human eye. While the decompressed image is not an exact replica of the original, the loss of quality is often controlled to minimize perceptual differences. Common lossy compression schemes include:

- Discrete Cosine Transform (DCT) used in JPEG
- Wavelet Transform used in JPEG2000
- Fractal Compression
- Vector Quantization
- Transform Coding
- Color Subsampling
- Quantization
- Delta Modulation

Additionally, image compression schemes can be further classified based on the underlying techniques they utilize:

Transform-Based Compression:

These schemes apply mathematical transforms to convert image data from the spatial domain to another domain (e.g., frequency domain), where data can be more efficiently represented.

- Discrete Cosine Transform (DCT)
- Wavelet Transform

Predictive Coding:

Predictive coding schemes predict pixel values based on neighboring pixels and encode only the prediction errors.

- Differential Pulse Code Modulation (DPCM)
- Adaptive Differential Pulse Code Modulation (ADPCM)
- Delta Modulation

Entropy Coding:

Entropy coding methods assign shorter codes to frequently occurring symbols, reducing the overall bitstream length.

- Huffman Coding
- Arithmetic Coding
- Golomb Coding
- Shannon-Fano Coding

Dictionary-Based Compression:

These schemes build and use dictionaries to replace repeated patterns with shorter codes.

- Lempel-Ziv-Welch (LZW) Coding
- Burrows-Wheeler Transform (BWT)

Hybrid Compression:

Hybrid compression schemes combine multiple techniques to achieve higher compression efficiency and better quality.

- JPEG (DCT + Huffman Coding)
- JPEG2000 (Wavelet Transform + Arithmetic Coding)
- WebP (Lossless: Predictive Coding + Entropy Coding, Lossy: VP8 Video Compression + DCT)

Neural Network-Based Compression:

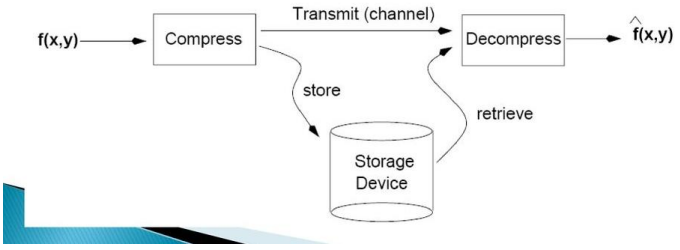
Recently, deep learning techniques like autoencoders and generative adversarial networks (GANs) have been explored for image compression.

- Variational Autoencoders (VAEs)
- Generative Adversarial Networks (GANs)

Fundamentals of information theory of IMAGE COMPRESSION

Information theory is a fundamental concept in the field of image compression. It provides a theoretical framework for understanding the fundamental limits of data compression and the relationship between data, entropy, and efficiency. Claude Shannon's groundbreaking work in the mid-20th century laid the foundation for information theory, which has since become a cornerstone of various compression techniques, including image compression.

- Data compression aims to reduce the amount of data while preserving as much information as possible.
- The goal of image compression is to reduce the amount of data required to represent an image.



Key concepts and fundamentals of information theory in the context of image compression include:

1. **Entropy:** Entropy measures the average amount of uncertainty or randomness in a set of symbols. In the context of images, symbols could be pixel values. The entropy of an image represents the inherent information content or randomness within it. Images with more uniform pixel values have lower entropy, while those with diverse and complex patterns have higher entropy.**Entropy** measures the amount of uncertainty or randomness in a dataset. In the context of images, it represents

the average "surprise" associated with observing a pixel value. Images with uniform pixel values or simple patterns have lower entropy because their pixel values are more predictable. Conversely, images with intricate textures and diverse pixel values have higher entropy due to their increased unpredictability.

2. **Information Content**: Information content quantifies the amount of information carried by a message. In the case of images, higher information content is associated with less predictable pixel values and, consequently, higher entropy. Redundancy reduction, a key goal in compression, involves minimizing the information content while maintaining acceptable quality.**Informatio**n content refers to the quantity of information carried by a message. In an image, pixels represent information. Images containing pixel values that are less predictable (higher entropy) carry more information. For example, in a grayscale image, if all pixels are of the same shade, the information content is low. But if the image contains a wide range of pixel values, the information content is higher.

3. **Entropy Coding :** Entropy coding techniques, such as Huffman coding and arithmetic coding, exploit the statistical properties of symbols to represent frequent symbols with shorter codes and less frequent symbols with longer codes. These techniques are used to represent data more efficiently, reducing the number of bits required for encoding**.Entropy** coding techniques, like Huffman coding, exploit the statistical distribution of symbols in a dataset to assign shorter codes to frequently occurring symbols and longer codes to less frequent ones. This reduces the overall length of the encoded message. Arithmetic coding is a more flexible alternative that can achieve even better compression efficiency.
4. **Shannon's Source Coding Theorem :** Shannon's theorem establishes the theoretical limits of lossless data compression. It states that for a given data source with a certain entropy, the average code length produced by an efficient code (like Huffman or arithmetic coding) cannot be shorter than the entropy of the source.**Shannon's** theorem sets a fundamental limit for lossless compression. It states that no lossless compression algorithm can achieve an average code length shorter than the entropy of the source. In image compression, this means that efficient codes can't compress data beyond the inherent randomness or information content present in the image.

5. **Rate-Distortion Theory :** Rate-distortion theory explores the trade-off between compression rate (number of bits) and distortion (quality loss) in lossy compression. It provides a framework to determine the optimal balance between compression and quality based on a specified rate or a desired level of distortion.**Rate-distortio**n theory addresses the balance between compression rate and distortion in lossy compression. It helps determine the optimal trade-off between the amount of compression (bit rate) and the quality of the reconstructed image. Achieving higher compression ratios often involves allowing controlled distortion in the reconstructed image.
6. **Lossless vs. Lossy Compression :** Information theory helps explain why lossy compression is possible. By focusing on removing perceptually less significant information while preserving the most critical information, lossy compression achieves high compression ratios with acceptable visual quality. This is achieved by exploiting the fact that some information can be discarded without significantly affecting the viewer's perception**.Information** theory explains why lossy compression is possible. It highlights that images contain perceptually less significant information that can be discarded without significantly affecting human perception. Lossy compression methods exploit this by quantizing, subsampling, or otherwise reducing less noticeable details, leading to high compression ratios while maintaining acceptable visual quality.

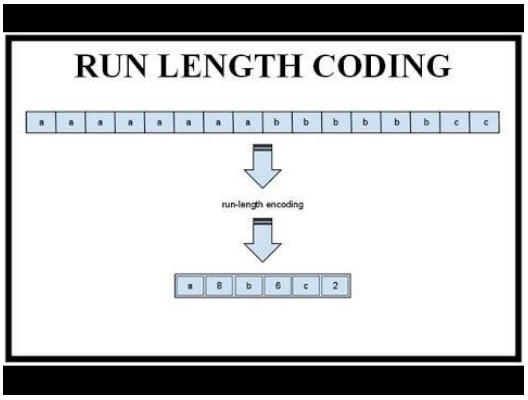
7. **Coding Efficiency and Redundancy :** Information theory underscores the concept of redundancy in data. Redundancy refers to the presence of patterns or correlations that can be leveraged for compression. Compression algorithms aim to reduce redundancy, thereby increasing coding efficiency**.Redundancy** exists in images due to predictable patterns and correlations among neighboring pixels. Compression algorithms identify and exploit this redundancy to represent data more efficiently. This involves encoding frequent patterns using shorter codes, thereby reducing the total number of bits needed for representation.

8. **Data Compression Efficiency :** Information theory provides insights into how much compression can potentially be achieved for a given dataset. The difference between the entropy of the original data and the entropy of the compressed data indicates the degree of compression achieved**.Information** theory helps quantify the potential compression efficiency of a dataset. The difference between the entropy of the original data and the entropy achieved by the compressed data provides insight into how well redundancy has been eliminated and how efficiently data has been encoded.
9. **Bits per Pixel (BPP) :** In image compression, the BPP metric quantifies the average number of bits required to represent each pixel. Information theory helps analyze how different compression methods affect the BPP and the resulting image quality**.Bits per Pixel** (BPP) is a metric indicating the average number of bits needed to represent each pixel in an image. Information theory guides the analysis of how different compression techniques impact the BPP value, influencing both compression ratio and resulting image quality.

Run length coding

Run-Length Coding (RLE) is a simple and efficient lossless compression technique used to reduce the size of data, including images. It works particularly well on data with long runs of repeated values or patterns. RLE exploits spatial redundancy in images where adjacent pixels often have the same color or intensity.

The basic idea of RLE is to replace consecutive repeated values with a single value followed by a count of how many times that value repeats. This allows for efficient encoding of repetitive sequences, reducing the amount of data needed to represent them.



Here's how Run-Length Coding works:

Encoding:

- Scan the image data sequentially, pixel by pixel or row by row.
- Whenever a run of identical pixels is encountered, count the number of consecutive occurrences (run length).
- Replace the run with the pixel value and its run length. For example, if you have a run of 10 white pixels, you'd replace it with "10W."

Decoding:

- While decoding, when a number followed by a pixel value is encountered (e.g., "10W"), replicate that pixel value the specified number of times to reconstruct the original run.
- RLE is most effective when there are long sequences of repeated values. For example, in binary images (black and white), RLE can significantly reduce the size because runs of the same pixel value are common. In grayscale or color images, it can still provide reasonable compression if there are regions with uniform colors or textures.

However, RLE may not be very effective on images with high variability or random patterns, as it won't find many repeated runs to compress.

Here's a simple example to illustrate RLE compression:

Original Image: BBBBBWWWWWBBBBB

Encoded using RLE: 5B5W5B

Decoding the RLE-encoded data would reproduce the original image.

Shannon – Fano coding

Shannon-Fano coding is a variable-length entropy coding technique used in data compression. It was developed by Claude Shannon and Robert Fano in the late 1940s as one of the earliest methods to achieve lossless data compression. Shannon-Fano coding assigns shorter codes to more frequently occurring symbols, helping to reduce the overall average code length and achieve compression.

The key idea behind Shannon-Fano coding is to divide the symbols into subsets in a way that the subsets represent different probability ranges. Symbols with higher probabilities are assigned shorter codes, while symbols with lower probabilities are assigned longer codes.

Here's how the Shannon-Fano coding process works:

1. **Sorting and Partitioning:**
   - Begin with a list of symbols and their associated probabilities (or frequencies) in descending order.
   - Divide the list into two subsets, attempting to balance the probabilities as closely as possible.
   - Recursively apply the partitioning process to each subset, creating smaller subsets.
2. **Assigning Codewords:**
   - For each subset, add '0' to the codeword of the symbols in the first subset and '1' to the codeword of the symbols in the second subset.
3. **Repetition:**
   - If a subset contains only one symbol, assign the same codeword to that symbol.
4. **Decoding:**
   - To decode, start from the beginning of the encoded data and traverse the code tree according to the encountered bits until a leaf node (symbol) is reached. Output the symbol and continue decoding.

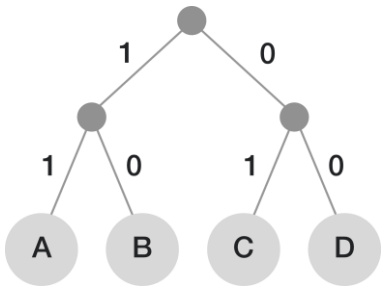
Shannon-Fano coding achieves entropy encoding by assigning shorter codes to symbols with higher probabilities. This means that more common symbols are represented by shorter bit sequences, reducing the overall average bit length required to represent the data.

However, one limitation of Shannon-Fano coding is that it might not always produce the optimal prefix-free codes (codes with no prefix being the prefix of another code) as guaranteed by Huffman coding. In some cases, the partitioning process can lead to imbalanced subsets and suboptimal compression efficiency.

Compared to Huffman coding, which uses a bottom-up approach and guarantees optimal prefix-free codes, Shannon-Fano coding uses a top-down approach that may result in slightly longer average code lengths. Due to its suboptimal nature, Shannon-Fano coding is less commonly used in modern compression algorithms. However, it provides valuable insights into the principles of entropy coding and variable-length codes, which are fundamental concepts in data compression.

Huffman coding

Huffman coding is a widely used variable-length entropy coding technique used in data compression to achieve lossless compression of information. Developed by David A. Huffman in 1952, it is particularly effective in reducing the size of data with variable probabilities or frequencies of occurrence for different symbols.



The main idea behind Huffman coding is to assign shorter codes to more frequent symbols and longer codes to less frequent symbols. This approach optimally utilizes the variable-length property of the codes to achieve compression by minimizing the average code length. Huffman coding produces prefix-free codes, meaning no code is a prefix of another code, ensuring unambiguous decoding.

Here's how the Huffman coding process works:

1. **Symbol Frequencies:**
- Count the frequency of each symbol in the input data (e.g., characters in a text document or pixel values in an image).
- Create a list of symbols along with their frequencies.

2. **Creating a Huffman Tree:**
- Build a priority queue (min-heap) of nodes, each representing a symbol along with its frequency.
- While there is more than one node in the queue, extract the two nodes with the lowest frequencies.
- Create a new internal node with a frequency equal to the sum of the frequencies of the two nodes. Make the two extracted nodes the left and right children of the new node.
- Insert the new node back into the queue.
- The process continues until only one node remains in the queue. This node becomes the root of the Huffman tree.

3. **Assigning Codewords:**
- Traverse the Huffman tree from the root down to each leaf node.
- Assign '0' to the left branch and '1' to the right branch.
- The path from the root to each leaf node corresponds to the binary codeword for that symbol.

4. **Decoding:**
- To decode, start from the root of the Huffman tree.
- Traverse down the tree based on the encountered bits (0 or 1).
- When a leaf node is reached, output the corresponding symbol and start decoding from the root again.

Huffman coding is very efficient when symbols have significantly different probabilities, as it assigns shorter codes to more frequent symbols. This results in reduced average code length and improved compression efficiency. It is widely used in various applications, including text compression (e.g., in ZIP files) and image compression (e.g., in JPEG).

Arithmetic coding

Arithmetic coding is a sophisticated and powerful entropy coding technique used in data compression to achieve efficient lossless compression. Unlike Huffman coding, which assigns codes to individual symbols, arithmetic coding encodes entire sequences of symbols into a single fractional value, allowing for more precise compression.

Developed by Peter Elias in the 1970s, arithmetic coding is known for its ability to approach the entropy limit more closely than many other compression methods. It achieves this by dividing the range of possible values into subintervals that correspond to the probabilities of the symbols being encoded.

Here's how the arithmetic coding process works:

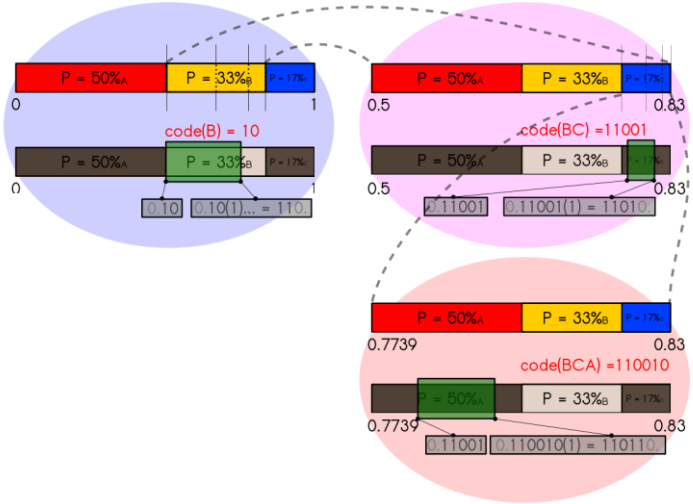
Symbol Probabilities:
- Calculate the cumulative probabilities of the symbols based on their frequencies or probabilities in the input data.
- Construct a cumulative probability distribution (also known as a cumulative frequency table).

Interval Division:
- Create an initial interval [0, 1) that spans the entire range of possible values.
- For each symbol in the input sequence, divide the current interval proportionally based on the cumulative probabilities of the symbols.

Encoding:
- Repeat the interval division for each symbol in the sequence.
- The final interval [low, high) represents the range of possible values for the encoded sequence.

Decoding:
- To decode, reverse the process: start with the encoded fractional value and the cumulative probability distribution.
- Iterate through the symbols, narrowing down the interval until the original sequence is reconstructed.

Arithmetic coding achieves higher compression efficiency compared to Huffman coding by allowing more precise representation of symbol probabilities. It's particularly effective for sequences of symbols with non-uniform or highly variable probabilities.



However, its efficiency comes at the cost of increased complexity in both encoding and decoding processes.

Key advantages of arithmetic coding include its ability to approach the entropy limit more closely and its adaptability to various symbol distributions without requiring predefined code lengths. However, there are some practical challenges, such as the need for precision during calculations, handling of boundary cases, and the requirement to transmit the codebook (probability distribution) along with the compressed data.

Predictive coding

Predictive coding is a data compression technique that takes advantage of the correlation or redundancy between successive data samples to achieve efficient compression. It's particularly effective in scenarios where adjacent data points are highly correlated, such as in images, audio signals, and video sequences. Predictive coding exploits the fact that the value of a data point can often be predicted based on the values of previous or neighboring data points.

The basic concept of predictive coding involves predicting the value of a data point using a mathematical model and then encoding the difference between the actual value and the predicted value. This difference, often called the prediction error or residual, is typically smaller than the original data value, and encoding it requires fewer bits.

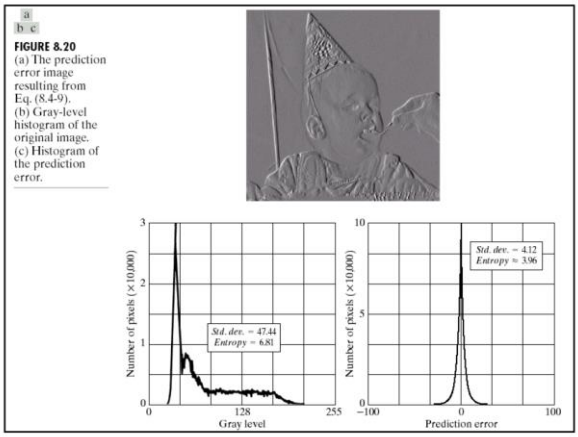
Here's how predictive coding works:

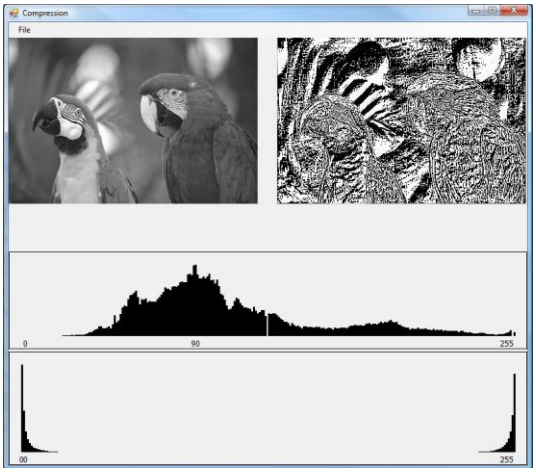
1. **Prediction Step:**
- Choose a prediction model that estimates the value of the current data point based on previous data points or a local neighborhood.
- Predict the value of the current data point using the chosen model.

2. **Residual Calculation:**
- Calculate the difference between the predicted value and the actual value of the current data point. This difference is the prediction error or residual.

3. **Encoding:**
- Encode the prediction error using an entropy coding technique, such as Huffman coding or arithmetic coding.
- The prediction error is often smaller in magnitude than the original data point, which results in more efficient compression.

4. **Decoding:**
- To decode, apply the reverse process: start with the predicted value, decode the encoded prediction error, and add it to the predicted value to reconstruct the original data point.





Predictive coding is used in various compression algorithms and formats:

- **DPCM (Differential Pulse Code Modulation):** This is a basic form of predictive coding where the prediction is based on the previous data point. The prediction error is quantized and then encoded.
- **ADPCM (Adaptive Differential Pulse Code Modulation):** ADPCM improves DPCM by adapting the prediction model based on the characteristics of the data.
- **Intra Prediction in Video Compression:** In video compression standards like H.264 and H.265 (HEVC), intra prediction uses neighboring pixels to predict the values of current pixels within a block, and the prediction errors are encoded.
- **JPEG Image Compression:** JPEG compression includes a predictive coding step using the Discrete Cosine Transform (DCT) to transform the image data into a frequency domain, followed by quantization and entropy coding.
- **Lossless Audio Compression:** Some lossless audio codecs, like FLAC, utilize predictive coding to compress audio samples.

Predictive coding is effective in reducing redundancy in data with strong correlations between adjacent values. It is often used in combination with other compression techniques, such as entropy coding, to achieve higher compression ratios while maintaining the ability to reconstruct the original data accurately.