

Unit 5: Data Compression

Total Time: 8 Hours

1. Storage Space

Syllabus Focus:

- Basic knowledge about storage space requirement in uncompressed vs compressed data
- Importance of compression for transmission and storage

Related PYQs:

- Describe the major steps of data compression.
- Why is data compression necessary in multimedia systems?

2. Coding Requirements

Syllabus Focus:

- Storage and bandwidth constraints for images, audio, and video
- Delay requirements for dialogue and retrieval mode multimedia applications

Related PYQs:

- *(No direct PYQ found but useful as introductory or supporting theory)*

3. Source, Entropy, and Hybrid Coding

Syllabus Focus:

- **Entropy coding:** Huffman coding, Arithmetic coding (intro), Run Length Encoding (RLE)
- **Source coding:** DPCM, DM, DCT (intro only)
- **Hybrid coding:** Combination of source and entropy coding, used in JPEG/MPEG

Related PYQs:

- Explain the Huffman coding process with example.
- What do you mean by Huffman coding? Explain the Huffman coding process with an example.
- Why do we need Huffman coding? Explain it with a suitable example.
- Discuss how Huffman code tree is generated in Huffman encoding. Construct Huffman code for characters A, B, C, and D with probabilities 0.4, 0.3, 0.2, 0.1 respectively.
- How does Run Length Encoding (RLE) compress data? Given original data "AAAAACBBBACCCCCZZZZU", what is its compressed data using RLE?
- Compare between lossless compression and lossy compression.
- Differentiate between lossless and lossy compression.

4. Lossy Sequential DCT-Based Mode

Syllabus Focus:

- 8×8 block division
- DCT transformation
- Quantization and Entropy Encoding
- Used in JPEG

Related PYQs:

- *(Part of JPEG compression questions below)*

5. Expanded Lossy DCT-Based Mode

Syllabus Focus:

- Full detail steps: FDCT → Quantization → Zigzag → DPCM → RLE → Huffman
- Applies to JPEG

Related PYQs:

- Explain the JPEG compression process steps in detail with an example.

6. JPEG and MPEG

Syllabus Focus:

- Compression steps and techniques used in **JPEG** (image) and **MPEG** (video)
- Color model conversion, transformation, quantization, motion estimation, entropy coding

Related PYQs:

- Explain the JPEG compression process steps in detail with an example.
- Explain the steps of the MPEG compression process.
- Differentiate between JPEG and MPEG.

Storage Space

- **Uncompressed multimedia data** (like raw audio, image, video) requires **huge storage space**.
- Transmission of such uncompressed data over networks is **difficult and inefficient**, as it demands **very high bandwidth**.

Why Compression is Necessary:

- Compressed data needs **less storage**.
- Reduces **transmission time** and **cost**.
- Allows smooth **streaming and communication** over networks.

Multimedia Systems Use Compression:

To efficiently store and transmit multimedia content (audio, image, video), various **compression standards** are used:

Media Type Common Compression Standard

Image	JPEG (Joint Photographic Experts Group)
Video	H.263, MPEG (Moving Picture Experts Group)
Audio	MPEG Audio Layer 3 (MP3), AAC

Example:

- A **raw image** of 1024×768 resolution with 24-bit color depth requires:
 $1024 \times 768 \times 24 = 18,874,368$ bits = ~ 2.26 MB
- The **same image compressed in JPEG** format may only take **200 KB – 500 KB** depending on quality.

Thus, compression helps save **more than 80–90% storage** without noticeable quality loss.

Coding Requirements

Multimedia data (image, audio, video) requires **higher data rates and larger storage** compared to simple text. Therefore, compression becomes essential—but it must also meet **real-time system constraints**.

i) Dialogue Mode Applications

Definition:

Interactive communication systems where **humans interact in real time** (e.g., video calls, live audio chat).

Requirements:

Parameter	Value
End-to-end delay (acceptable)	< 150 ms
Ideal for face-to-face	< 50 ms

Reason:

- Delays beyond 150 ms can disrupt natural conversation flow.
- Face-to-face apps (e.g., Zoom, Skype) require very **low latency** to maintain smooth interaction.

ii) Retrieval Mode Applications

Definition:

Applications where users retrieve stored multimedia content from databases.

Examples: **YouTube, Netflix, Digital Libraries**

Requirements:

Requirement	Details
Fast forward/backward	User must quickly navigate video/audio
Simultaneous display	Retrieval should happen without buffering
Random access	Single frame or image access in < 0.5 sec
Fast search	Efficient indexing for text/audio/video

Key Coding Constraints in Multimedia Systems:

Constraint Type	Description
1. Low Latency	Delay between input and output must be minimal. Especially crucial in video conferencing or live streams .
2. Low Bitrate	The compression should use minimum bits per second to save bandwidth and storage .
3. Acceptable Quality	Even after compression, the output should be visually/audibly acceptable to human users.
4. Fast Encoding/Decoding	Algorithms must work quickly for real-time use, especially on low-power devices.
5. Scalability	Compression schemes should allow adjusting quality vs. bitrate based on network conditions or device capability.
6. Error Resilience	Compressed data should tolerate small errors (due to network loss) without major degradation in quality.

Source, Entropy, and Hybrid Coding

In multimedia compression, **different coding techniques** are used to reduce file size while maintaining acceptable quality. These include:

1. Entropy Coding (Lossless)

Definition:

Entropy coding is a **lossless compression technique** that compresses data **without losing any information**. It uses **statistical properties** of the data.

Key Features:

- No information is lost.
- Works on **symbol frequencies**.
- Regenerates the exact original data after decompression.
- Used in the final stages of compression pipelines.

Common Types:

Technique	Description
Run-Length Encoding (RLE)	Replaces repeating values with a single value and count.
Huffman Coding	Assigns shorter binary codes to more frequent symbols.
Arithmetic Coding	Represents a whole message as a single floating-point number.

Example – Run Length Encoding (RLE):

Original: AAAAACBBBACCCCCZZZZU

Compressed: A5C1B3A1C5Z5U1

2. Source Coding (Lossy)

Definition:

Source coding is a **lossy compression** technique that reduces data by **removing redundancies or less important information** based on media characteristics.

Key Features:

- Some data is **permanently removed**.
- Uses knowledge of **human perception** (e.g., less sensitivity to color than brightness).
- Focused on the **semantics and structure** of the source (e.g., audio, image, video).
- **High compression ratio**, but some quality loss.

Common Types:

Technique	Description
DPCM (Differential PCM)	Encodes differences between consecutive samples.
DM (Delta Modulation)	Encodes signal as steps (simplified DPCM).
DCT (Discrete Cosine Transform)	Converts data into frequency domain and removes high-frequency (less important) parts.

3. Hybrid Coding (Entropy + Source Coding)

Definition:

Hybrid coding combines both **source** and **entropy** coding to achieve **better compression efficiency**.

Key Features:

- First, **source coding** removes perceptually unimportant data.
- Then, **entropy coding** compresses the remaining data losslessly.
- Most **modern multimedia systems** use hybrid coding.

Common Standards (Examples):

Standard	Usage
JPEG	Image compression (uses DCT + Huffman)
MPEG	Video compression (uses DCT + motion estimation + entropy coding)

Huffman Coding (Lossless Compression)

Definition:

Huffman coding is a **lossless data compression algorithm** that assigns **variable-length binary codes** to characters based on their **frequency** of occurrence.

- Characters with higher frequency are assigned **shorter codes**.
- Characters with lower frequency are assigned **longer codes**.
- It uses **prefix codes**, meaning no code is a prefix of another. This ensures **uniqueness and no ambiguity** in decoding.

Purpose:

- To minimize the total number of bits used for representation.
- To ensure exact and complete reconstruction of the original data.
- Used in various compression formats like **JPEG**, **MP3**, and **ZIP**.

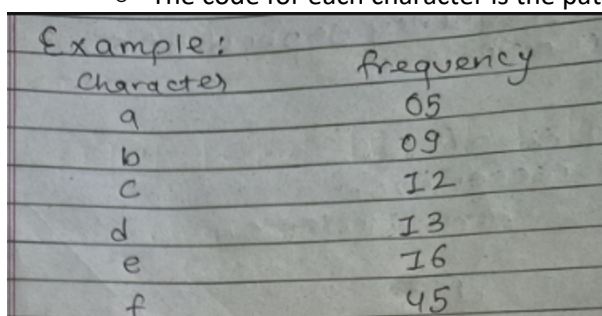
Steps of Huffman Coding:

Input:

A set of characters along with their frequencies of occurrence.

Step-by-Step Procedure:

- 1. Create Leaf Nodes**
 - Create a node for each character and its frequency.
- 2. Build a Min-Heap (Priority Queue)**
 - Insert all nodes.
 - The node with the lowest frequency is at the root.
- 3. Combine the Two Lowest-Frequency Nodes**
 - Remove the two nodes with the lowest frequency.
 - Create a new internal node:
 - Frequency = sum of both nodes.
 - Left child = one node, Right child = other.
 - Insert the new node back into the heap.
- 4. Repeat the Process**
 - Continue combining nodes until only one node remains.
 - This final node becomes the **root** of the **Huffman Tree**.
- 5. Assign Binary Codes**
 - Traverse the tree from root:
 - Assign 0 for left edges and 1 for right edges.
 - The code for each character is the path from the root to its leaf node.



Example:

Character	Frequency
a	05
b	09
c	12
d	13
e	16
f	45

Q. Construct Huffman code for characters A, B, C and D with probabilities 0.4, 0.3, 0.2, 0.1 respectively.

Solⁿ: Sorting:-

D 0.1 C 0.2 B 0.3 A 0.4

Combining smallest two:-

0.3

D 0.1 C 0.2 B 0.3 A 0.4

Sorting & combining smallest two:-

0.6

0.3 B 0.3 A 0.4

D 0.1 C 0.2 0

Again com-sorting and combining smallest two:-

1.0

0 1

A 0.4 0.6

0 1

0.3 B 0.3

0 1

D 0.1 C 0.2

So, huffman codes for,

A \Rightarrow 0

B \Rightarrow 11

C \Rightarrow 100

D \Rightarrow 101

Arithmetic Coding

Arithmetic coding is a lossless data compression technique that encodes data by representing a sequence of symbols as a fraction (a real number) between 0 and 1. Unlike Huffman coding, which assigns fixed-length codes to symbols, arithmetic coding encodes the entire message as a single number, making it more efficient for certain types of data.

Key Points:

- **Representation of symbols:** Each symbol (e.g., character) in the input message is represented by a fractional range on the number line between 0 and 1.
- **Compression efficiency:** Symbols that appear more frequently in the input string will have smaller fractional ranges, using fewer bits, while less frequent symbols will take up larger ranges.
- **Lossless:** The original data can be fully recovered during the decompression process.

Process:

1. **Assign probabilities** to each symbol based on their frequencies.

2. **Divide the number line** from 0 to 1 according to these probabilities.
3. For each symbol in the input string, narrow the range of the number line to represent the symbol.
4. After processing the entire message, the resulting number represents the entire string in a fractional format.

Example:

- Input: ABABA
- Probabilities: A (0.6), B (0.4)
- First symbol: A → range from 0 to 0.6.
- Second symbol: B → range from 0.6 to 1.
- Continue this process until the entire string is represented by a single number between 0 and 1.

Run Length Encoding (RLE)

Run Length Encoding (RLE) is a simple form of lossless compression where consecutive repeated characters (or bytes) are stored as a single data value and a count of how many times the value is repeated.

Key Points:

- **Simplicity:** RLE is easy to implement and efficient when there are long runs of repeated characters.
- **Compression ratio:** Compression can be as high as 25% for data with many repeated characters, but it may increase the size when the data contains few or no repeated characters.

Steps of RLE Compression

1. **Scan the input data** from left to right.
2. **Count consecutive repeating characters (run).**
3. **If count ≥ 4 ,** replace the run with:
character ! count

(e.g., A!5 for five A's)
4. **If count < 4 ,** write characters as they are (uncompressed).
5. **Continue until the end** of the input.

Example 1: Compression

Input:

ABCCCCDDDF

Step-by-step:

- A → 1 → Keep as A
- B → 1 → Keep as B
- C → 5 → Compress as C!5
- D → 3 → Keep as DDD
- F → 1 → Keep as F

Compressed Output:

AB!5DDDF

Example 2: Better Compression

Input:

ABBBBBBBBBBCCDDDDDDDDDE

Step-by-step:

- $A \rightarrow 1 \rightarrow A$
- $B \rightarrow 11 \rightarrow B!11$
- $C \rightarrow 2 \rightarrow CC$
- $D \rightarrow 8 \rightarrow D!8$
- $E \rightarrow 1 \rightarrow E$

Compressed Output:

AB!11CCD!8E

Example 3: When RLE Fails

Input:

AABBCCD

Step-by-step:

All characters repeat < 4 times → Not suitable for compression

Compressed Output (unnecessary):

A02B02C02D01

Original size: 7 bytes

Compressed size: 12 bytes → **Compression failed**

Lossy Sequential DCT-Based Mode

Lossy sequential DCT is the standard mode used in **JPEG compression** for still images. It compresses data by transforming pixel values into frequency components and discarding less important ones.

Processing Steps

Uncompressed Image → 8×8 Blocking → Forward DCT → Quantization → Entropy Encoding → Compressed Output

1. Image Preparation and Blocking

- The uncompressed image is divided into **8×8 pixel blocks**.
- Each pixel is represented by **8 bits** (values from 0 to 255).
- This division helps apply frequency transformation block-wise.

2. Forward Discrete Cosine Transform (FDCT)

Each 8×8 block undergoes **DCT** to convert spatial domain data into frequency domain.

DCT Formula:

$$T(u, v) = \frac{1}{4} C_u C_v \sum_{x=0}^7 \sum_{y=0}^7 Z(x, y) \cdot \cos\left(\frac{(2x+1)u\pi}{16}\right) \cdot \cos\left(\frac{(2y+1)v\pi}{16}\right)$$

Where:

- $Z(x, y)$: pixel value at position (x, y)
- $T(u, v)$: transformed DCT coefficient
- $C_u = C_v = \frac{1}{\sqrt{2}}$ for $u, v = 0$, otherwise 1

3. Quantization

- This is the **lossy step** of the process.
- Each DCT coefficient is divided by a **quantization value** and rounded.

Quantization Formula:

$$Q(u, v) = \text{round}(T(u, v) / Q_{\text{table}}(u, v))$$

Where:

- $Q_{\text{table}}(u, v)$ is a standard quantization matrix
- Lower frequencies are preserved better, higher ones are reduced or zeroed

Effects of Quantization:

- Reduces precision, especially of higher frequencies
- Exploits the fact that human vision is less sensitive to high frequencies
- Allows significant compression with minimal perceived loss

4. Entropy Encoding

This step compresses the quantized coefficients **losslessly**.

Main Techniques Used:

- **DC Coefficient:**
Difference between current DC and previous DC is encoded.
- **AC Coefficients:**
 - Scanned in **zig-zag order** to group zeros
 - Encoded using **Run-Length Encoding (RLE)** followed by **Huffman Coding**

Expanded Lossy DCT-Based Mode

1. Extension of Sequential DCT:

It enhances the basic sequential DCT mode used in JPEG by supporting more features like color components and progressive scans.

2. Color Subsampling:

Supports subsampling of chrominance (color) components (e.g., 4:2:0 format) to reduce redundancy in color data.

3. Multiple Scan Capability:

Image data can be sent in multiple scans (progressive JPEG), improving perceived loading time

and compression flexibility.

4. Uses Same Core Steps:

- Block-wise DCT on 8×8 pixel blocks
- Quantization using standard quantization tables
- Entropy encoding (typically Huffman coding)

5. Improved Compression:

By spreading data over multiple scans and compressing chroma channels more aggressively, it achieves better compression with acceptable quality loss.

Comparison: Lossless vs Lossy Compression

Aspect	Lossless Compression	Lossy Compression
Definition	Compresses data without losing any information	Compresses data by removing some information
Original Data	Can be perfectly reconstructed from the compressed data	Cannot be perfectly reconstructed (some data lost)
Data Quality	No loss in quality	Slight to significant quality loss depending on ratio
Compression Ratio	Lower compression ratio	Higher compression ratio
Use Cases	Text, documents, executable files, PNG images	Images (JPEG), audio (MP3), videos (MPEG)
Techniques Used	Huffman coding, RLE, Arithmetic coding	DCT, Quantization, Predictive coding
Example Formats	PNG, ZIP, FLAC	JPEG, MP3, MPEG

JPEG Compression Process

Full Form: Joint Photographic Experts Group

Used for: Lossy compression of still images

Goal: Reduce image size with minimal loss in quality, for storage and transmission efficiency.

Overview of Steps:

Uncompressed Image



Split into 8×8 Blocks (MCU)



Color Conversion (RGB → YCbCr)



DCT (Discrete Cosine Transform)



Quantization



Zigzag Scanning



DPCM (for DC) + RLE (for AC)

↓
Huffman Encoding
↓
Compressed Image

Step-by-Step Explanation

Step 1: Split Image into 8×8 Pixel Blocks

- The image is divided into **blocks of 8×8 = 64 pixels**.
- Each pixel has a value between 0 and 255.
- These blocks are called **Minimum Coded Units (MCUs)**.

Step 2: Color Space Conversion (RGB to YCbCr)

- JPEG uses **YCbCr** color model instead of RGB.
 - Y = brightness (luminance)
 - Cb = blue chrominance
 - Cr = red chrominance
- Reason: Human eyes are more sensitive to brightness than color.

Step 3: Discrete Cosine Transform (DCT)

- Converts image data from **spatial domain** → **frequency domain**.
- Low-frequency parts (important features) stay at top-left.
- High-frequency (noise) goes to bottom-right.

Step 4: Quantization (Lossy Step)

- Divides each DCT coefficient by a **quantization table** value and rounds off.
- Removes less important high-frequency components.
- This is where **data loss** happens, reducing file size.

Step 5: Zigzag Scanning

- The 8×8 matrix is converted into a **1×64** vector using **zigzag** order.
- It groups low-frequency (important) values together at the start.

Example:

Zigzag order:

0 1 5 6 14 ...

Step 6: Differential Coding (DPCM) for DC Coefficient

- DC value of a block is **predicted** from the DC of previous block.
- Only the **difference** is stored.

DPCM = D_{current} – D_{previous}

Step 7: Run-Length Encoding (RLE) for AC Coefficients

- AC values often contain **many zeros**.
- Encoded as (skip, value) pairs, where skip is the number of zeros before the next non-zero.

Example:

AC values: [12, 0, 0, 0, 5, 0, 0, 3]

→ RLE: (0,12), (3,5), (2,3)

Step 8: Huffman Encoding

- Final compression step.
- DC and AC symbols are encoded using **Huffman coding**, which assigns shorter codes to more

frequent values.

Advantages of JPEG Compression

- High compression ratio
- Easy to implement
- Widely supported (used in phones, cameras, web, etc.)

MPEG Compression Process (Moving Picture Experts Group)

MPEG is a popular method used to compress video and audio files. Unlike JPEG, which compresses a single image, MPEG compresses a sequence of images (video). Instead of storing each frame fully, it stores only the **differences** between consecutive frames, resulting in a **high compression ratio**.

Common MPEG Versions:

- MPEG-1: Used in Video CDs (VCD)
- MPEG-2: Used in DVDs and TV broadcasting
- MPEG-4: Used in web video formats like .mp4

Steps of MPEG Compression:

1. Resolution Reduction (Color Space Conversion)

- Converts video frames from RGB to YUV color space.
- Y (luminance/brightness) is more important for human vision, so it is retained with higher accuracy.
- U and V (color components) are compressed more because they are less sensitive to human eyes.

2. Motion Estimation

- Compares the current frame with the previous one.
- Identifies blocks of pixels that have moved and calculates motion vectors.
- Detects and removes **temporal redundancy** (repetition over time).

3. Motion Compensation and Image Substitution

- Uses motion vectors to reconstruct frames.
- Instead of storing the entire frame, stores only the **difference** from the reference frame.
- Reduces file size and bandwidth requirements.

4. DCT (Discrete Cosine Transform)

- Same as in JPEG.
- Converts spatial domain pixel data into frequency domain.

5. Quantization

- Same as in JPEG.
- Reduces precision of high-frequency data, which is less visible to the human eye.

6. Run Length Encoding (RLE)

- Same as in JPEG.
- Compresses long sequences of zeros in AC coefficients by storing them as (skip, value) pairs.

7. Entropy Encoding

- Same as in JPEG.
- Applies Huffman coding to further compress data by reducing redundancy.

MPEG vs JPEG

1. Purpose:

- **JPEG** compresses still images.
- **MPEG** compresses video and audio.

2. Compression Type:

- **JPEG** uses **intra-frame compression** (within a single image).
- **MPEG** uses **inter-frame compression** (between multiple frames).

3. Redundancy Removed:

- **JPEG** removes **spatial redundancy**.
- **MPEG** removes both **spatial and temporal redundancy**.

4. Techniques Used:

- **JPEG** uses DCT, Quantization, Zigzag, RLE, and Huffman coding.
- **MPEG** includes **motion estimation and compensation** along with DCT and entropy coding.

5. Output Formats:

- **JPEG** produces image files like .jpg or .jpeg.
- **MPEG** produces video files like .mpg, .mpeg.