

Multimedia System

Course Code: CACS457 | Year/Semester: IV / VIII

Unit 5: Data Compression

Earlier units repeatedly showed that raw audio, image, and video data is huge (Units 2-4 worked examples). Data compression is the set of techniques that makes multimedia storage and transmission practical by reducing this data size, while keeping perceived quality as high as possible. This unit covers why compression is needed, how it works in general, and then looks closely at the JPEG and MPEG compression standards.

1. Need for Data Compression

Uncompressed multimedia data is extremely large. Earlier units calculated this directly: a 30-second stereo CD-quality audio clip was about 5 MB (Unit 2), a single 1024×768 true-color image was about 2.25 MB (Unit 3), and just 10 seconds of modest-resolution video required over 200 Mbps (Unit 4). Multiplying these numbers across full songs, photo libraries, or full-length films makes the scale of the problem clear.

Reason Compression Is Needed	Plain Explanation
Limited storage capacity	Devices and servers have finite disk/SSD space; raw multimedia would fill it almost instantly
Limited network bandwidth	Internet and mobile connections can only transmit a limited amount of data per second
Streaming and real-time delivery	Video calls, live streams, and online gaming need data delivered fast enough to avoid lag/buffering
Cost	Storage and bandwidth both cost money — smaller files reduce hosting and data transfer costs
Practical sharing	Smaller files are faster to upload, download, email, and share between devices

Note: Compression is not a luxury feature — without it, none of today's common multimedia experiences (HD video streaming, music streaming, instant photo sharing) would be practically possible.

2. Compression Basics

Data compression is the process of re-encoding data using fewer bits than the original representation, by removing redundancy — information that is repeated, predictable, or not essential. Compression algorithms work by detecting and exploiting one or more of the following types of redundancy.

Four Types of Redundancy That Compression Exploits

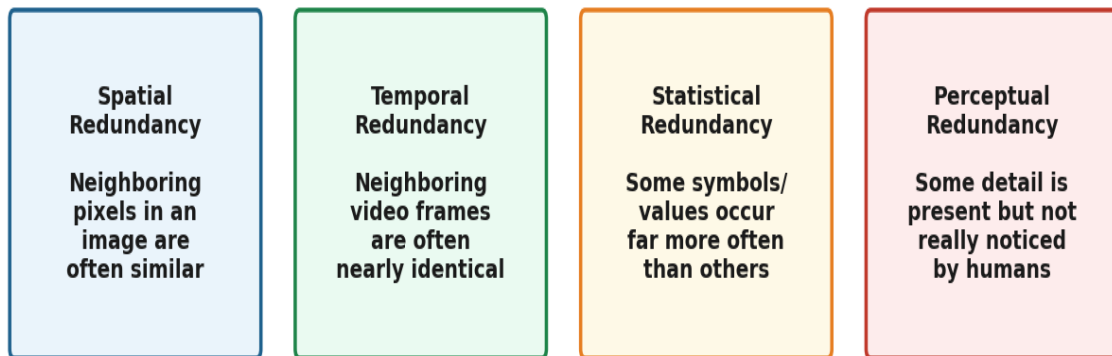


Figure 1: Compression techniques exploit spatial, temporal, statistical, and perceptual redundancy.

Compression Ratio

The effectiveness of a compression method is measured using the compression ratio: the size of the original (uncompressed) data divided by the size of the compressed data. A compression ratio of 10:1 means the compressed file is one-tenth the size of the original.

Example 1: Compression Ratio Calculation

Question: A 50 MB uncompressed video file is compressed down to 4 MB. Calculate the compression ratio and the percentage of space saved.

Solution / Explanation:

Compression ratio = Uncompressed size $\div$ Compressed size = $50 \div 4 = 12.5 : 1$.

Space saved (%) = $[(\text{Uncompressed} - \text{Compressed}) \div \text{Uncompressed}] \times 100$.

Space saved = $[(50 - 4) \div 50] \times 100 = (46 \div 50) \times 100 = 92\%$.

So the compressed file uses only 8% of the original storage space — a 92% reduction.

3. Storage Space

Understanding storage space requires understanding the units used to measure digital data, and how quickly multimedia content can consume them.

Unit	Size	Rough Multimedia Equivalent
1 Kilobyte (KB)	1,024 bytes	A few seconds of plain text
1 Megabyte (MB)	1,024 KB	About one minute of MP3 audio, or one photo
1 Gigabyte (GB)	1,024 MB	About 200 MP3 songs, or roughly 1 hour of compressed HD video
1 Terabyte (TB)	1,024 GB	Several hundred hours of compressed HD video, or thousands of photos

Without compression, even modest libraries of photos, songs, or videos would quickly exceed the capacity of typical consumer storage devices and would be impractical to back up, sync, or transfer over typical home or mobile internet connections.

Note: This is precisely why every common storage and streaming service (cloud photo backup, music/video streaming) relies on compressed formats as the default, not the exception.

4. Coding Requirements

A compression scheme assigns codes (sequences of bits) to represent the original data. For this encoding/decoding process to actually work correctly and efficiently, the coding scheme used must satisfy several requirements.

Requirement	Plain-English Meaning
Unique decodability	Given a stream of coded bits, there must be only one possible way to decode it back to the original symbols
Prefix property (instantaneous codes)	No codeword should be the prefix (beginning) of another codeword, so the decoder always knows exactly when one code ends and the next begins
Efficiency / minimal average length	Frequently occurring symbols should be assigned shorter codes, and rare symbols longer codes, to minimize the average number of bits used
Error resilience (where needed)	For transmission over unreliable channels, codes may need some structure to detect or recover from transmission errors
Reasonable computational cost	Encoding and decoding must be fast enough for the intended use (e.g. real-time video must encode/decode within strict time limits)

Note: The 'prefix property' is why Huffman coding (Section 6) always produces a valid, uniquely decodable binary tree-based code — no finished code is ever a prefix of another.

5. Lossless and Lossy Compression Techniques

Compression techniques fall into two broad categories, based on whether any original information is permanently discarded.

Lossless compression allows the original data to be perfectly reconstructed from the compressed version, with absolutely no loss of information — essential for text, program files, and any data where exact accuracy matters. Lossy compression permanently discards some information — usually detail that is least noticeable to human senses — to achieve much higher compression ratios, commonly used for audio, images, and video

where some quality loss is an acceptable trade-off for much smaller files.

Category	Common Techniques	Typical Use
Lossless	Run-Length Encoding (RLE), Huffman coding, Arithmetic coding, Lempel-Ziv (LZ77/LZ78, LZW)	Text, executable files, PNG images, FLAC audio, ZIP archives
Lossy	Transform coding (e.g. DCT), Quantization, Predictive coding with loss, Perceptual coding	JPEG images, MP3/AAC audio, MPEG/H.264 video

Run-Length Encoding (RLE) — A Simple Lossless Example

RLE replaces runs of repeated values with a single value plus a count. For example, the sequence 'AAAAABBBCCCCCCCC' (16 symbols) can be losslessly encoded as 'A5 B3 C8' — far shorter, yet perfectly reversible back to the original sequence.

6. Source, Entropy and Hybrid Coding

Source Coding

Source coding removes redundancy that is specific to the meaning/structure of the source data itself — for example predicting a pixel's value from its neighbors, or a video frame from a previous frame — before any general-purpose bit-level compression is applied.

Entropy Coding

Entropy coding is a lossless technique that assigns shorter codes to more frequent symbols and longer codes to rarer symbols, based on each symbol's probability of occurring. Huffman coding is the most well-known entropy coding method: it builds a binary tree from symbol frequencies, where more frequent symbols end up closer to the root (shorter codes).

Example Huffman Coding Tree (5 Symbols)

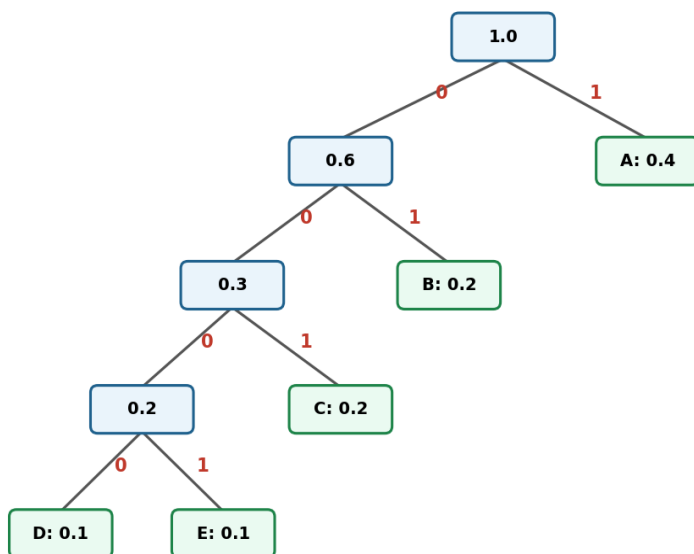


Figure 2: A Huffman coding tree — frequently occurring symbols (like A) get shorter codes than rare ones (like D, E).

Example 2: Reading Codes from a Huffman Tree

Question: Using the Huffman tree in Figure 2, write down the binary code for symbols A and D, and calculate the average code length given the symbol probabilities shown (A=0.4, B=0.2, C=0.2, D=0.1, E=0.1).

Solution / Explanation:

Reading from the root: A = 1 (1 bit). D = 0001 (going left, left, left, then left at the final split = 4 bits, since D and E share the deepest branch).

Code lengths: A=1 bit, B=2 bits, C=3 bits, D=4 bits, E=4 bits.

Average code length = $\sum (\text{probability} \times \text{code length}) = (0.4 \times 1) + (0.2 \times 2) + (0.2 \times 3) + (0.1 \times 4) + (0.1 \times 4)$.

= $0.4 + 0.4 + 0.6 + 0.4 + 0.4 = 2.2$ bits per symbol, compared to 3 bits per symbol needed for a simple fixed-length code over 5 symbols — a clear saving from entropy coding.

Hybrid Coding

Most practical multimedia compression standards (including JPEG and MPEG, Sections 7-9) are hybrid coding systems: they combine several different techniques in stages — for example a lossy transform/source-coding stage to reduce perceptual redundancy, followed by a lossless entropy coding stage to squeeze out any remaining statistical redundancy. This combination achieves far better compression than any single technique alone.

7. Lossy Sequential DCT-Based Mode

The lossy sequential DCT-based mode (often written 'DCT' for Discrete Cosine Transform — note: this is the same process some syllabi abbreviate as 'OCT-based mode') is the most common and widely supported JPEG compression mode, used as the default 'baseline' JPEG encoder in most software.

JPEG Baseline Sequential DCT-Based Encoding Pipeline

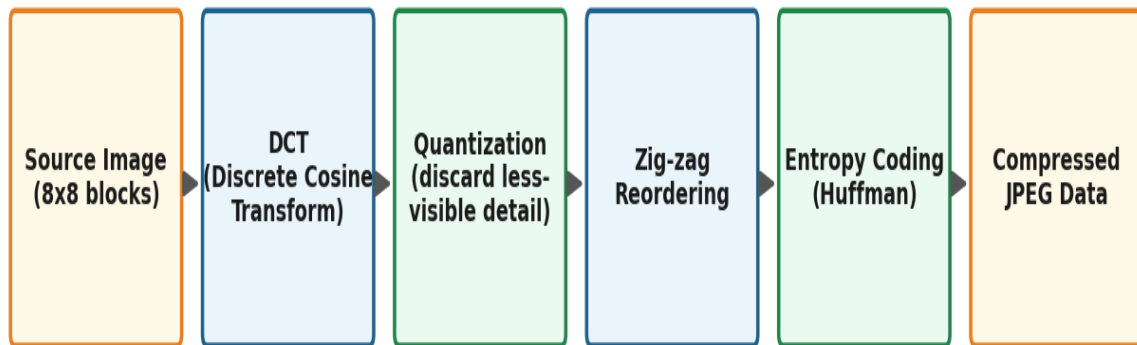


Figure 3: The baseline sequential DCT-based JPEG pipeline processes the image in 8x8 pixel blocks, left to right, top to bottom.

Steps of the Baseline Sequential DCT-Based Mode

Step	What Happens
1. Block splitting	The image is divided into non-overlapping 8×8 pixel blocks
2. Discrete Cosine Transform (DCT)	Each block is converted from the spatial (pixel) domain into the frequency domain, separating broad shape (low frequency) from fine detail (high frequency)
3. Quantization	High-frequency components (fine detail least noticed by the eye) are reduced/discarded — this is the actual lossy step
4. Zig-zag reordering	The quantized values are reordered to group similar/zero values together, improving the next step's efficiency
5. Entropy coding	The reordered values are losslessly compressed further using Huffman coding

The term 'sequential' means the image is encoded and can be decoded in a single top-to-bottom, left-to-right pass, block by block — as opposed to the progressive approach discussed next.

8. Expanded Lossy DCT-Based Mode

Beyond the basic baseline sequential mode, the JPEG standard defines an expanded (extended) lossy DCT-based mode that adds extra capabilities for higher precision and more flexible display, while still using the same core DCT-based approach.

Extended Mode Feature	Plain Description
Progressive DCT-based mode	The image is encoded in multiple scans, each adding more detail — a low-quality version appears almost immediately, sharpening progressively as more data arrives
Hierarchical mode	The image is encoded at multiple resolutions, so a lower-resolution version can be decoded quickly, and higher resolutions are built on top of it
12-bit sample precision	Supports higher color-depth source samples (beyond the standard 8 bits) for specialized/professional use
Arithmetic coding option	Allows arithmetic coding instead of Huffman coding for the entropy coding stage, which can compress slightly better

Note: Progressive JPEG is a great practical example of expanded DCT-based mode: it is why some images on slow web connections appear blurry at first and then sharpen gradually, rather than loading sharply from top to bottom.

9. JPEG and MPEG Compression

JPEG (Joint Photographic Experts Group) is the compression standard for still images discussed in Sections 7-8. MPEG (Moving Picture Experts Group) extends similar ideas to video, adding techniques to also exploit temporal redundancy between frames (Section 2).

How MPEG Builds on JPEG-Style Compression

Each individual video frame can be compressed using DCT-based techniques very similar to JPEG. However, MPEG adds motion compensation: instead of encoding every frame completely from scratch, it predicts each frame from nearby frames and only encodes the difference (motion and residual detail), since consecutive video frames are usually very similar (temporal redundancy).

MPEG Frame Types within a Group of Pictures (GOP)

Group of Pictures (GOP): I = key frame, P = predicted from earlier frame, B = predicted from both sides

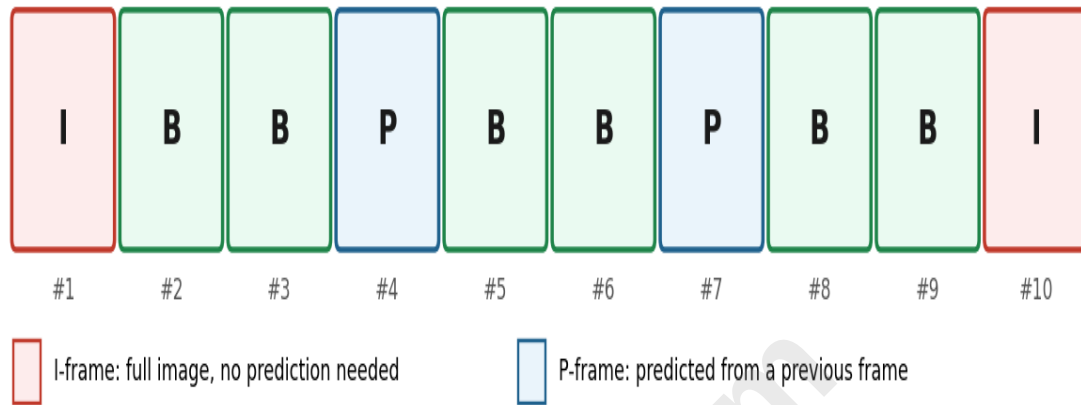


Figure 4: MPEG organizes frames into a Group of Pictures (GOP) using I, P, and B frame types.

Frame Type	Plain Description
I-frame (Intra-coded)	A complete, self-contained frame compressed like a JPEG image — no reference to other frames needed
P-frame (Predicted)	Encoded as the difference from an earlier I- or P-frame, exploiting temporal redundancy — much smaller than an I-frame
B-frame (Bidirectional)	Encoded as the difference from both an earlier and a later frame, usually achieving the smallest size of all three types

JPEG vs MPEG — Summary Comparison

Aspect	JPEG	MPEG
Content type	Still images	Video (sequence of moving frames)
Core technique	DCT, quantization, entropy coding (per image)	DCT-based frame coding plus motion compensation between frames
Redundancy exploited	Spatial and statistical (within one image)	Spatial, statistical, and temporal (across frames)
Typical compression ratio	10:1 to 20:1 for good visual quality	Can exceed 50:1 to 100:1+ due to added temporal compression

Example 3: Why MPEG Compresses Video So Much Better Than JPEG-Per-Frame

Question: A 2-second video clip has 60 frames (30 fps). If each frame were stored independently as a JPEG-style image at 100 KB per frame, what would the total size be? If MPEG instead uses 2 I-frames at 100 KB each and the remaining 58 frames as P/B-frames averaging only 8 KB each (thanks to motion compensation), what is the MPEG total size, and what compression benefit does this give versus storing every frame as a full JPEG-style image?

Solution / Explanation:

All-JPEG approach: $60 \text{ frames} \times 100 \text{ KB} = 6,000 \text{ KB}$ (about 5.86 MB).

MPEG approach: $(2 \times 100 \text{ KB}) + (58 \times 8 \text{ KB}) = 200 + 464 = 664 \text{ KB}$.

Reduction = $6,000 \div 664 \approx 9$ times smaller than encoding every frame independently.

This demonstrates exactly why MPEG's use of I/P/B frames and motion compensation (exploiting temporal redundancy) is so much more powerful for video than simply compressing each frame as a separate JPEG image.

Key Takeaways — Unit 5

- Data compression is essential because raw audio, image, and video data is far too large for practical storage, bandwidth, and streaming needs.
- Compression works by removing spatial, temporal, statistical, and perceptual redundancy; effectiveness is measured by the compression ratio.
- Storage units (KB, MB, GB, TB) scale by $1024\times$ at each level, and multimedia content fills them quickly without compression.
- Good coding schemes must be uniquely decodable, satisfy the prefix property, and minimize average code length.
- Lossless compression (RLE, Huffman, LZW) perfectly preserves original data; lossy compression (transform coding, quantization) discards less-noticeable detail for much smaller files.
- Source coding removes structure-specific redundancy; entropy coding (e.g. Huffman) assigns shorter codes to frequent symbols; hybrid coding combines multiple techniques in stages.
- JPEG's baseline sequential DCT-based mode splits images into 8×8 blocks and applies DCT, quantization, and entropy coding in a single pass.
- Expanded (extended) DCT-based modes add progressive and hierarchical encoding, higher precision, and alternative entropy coding.
- MPEG extends JPEG-style per-frame compression with motion compensation and I/P/B frame types, exploiting temporal redundancy for far higher video compression ratios.

— End of Unit 5 —

bcalink.com