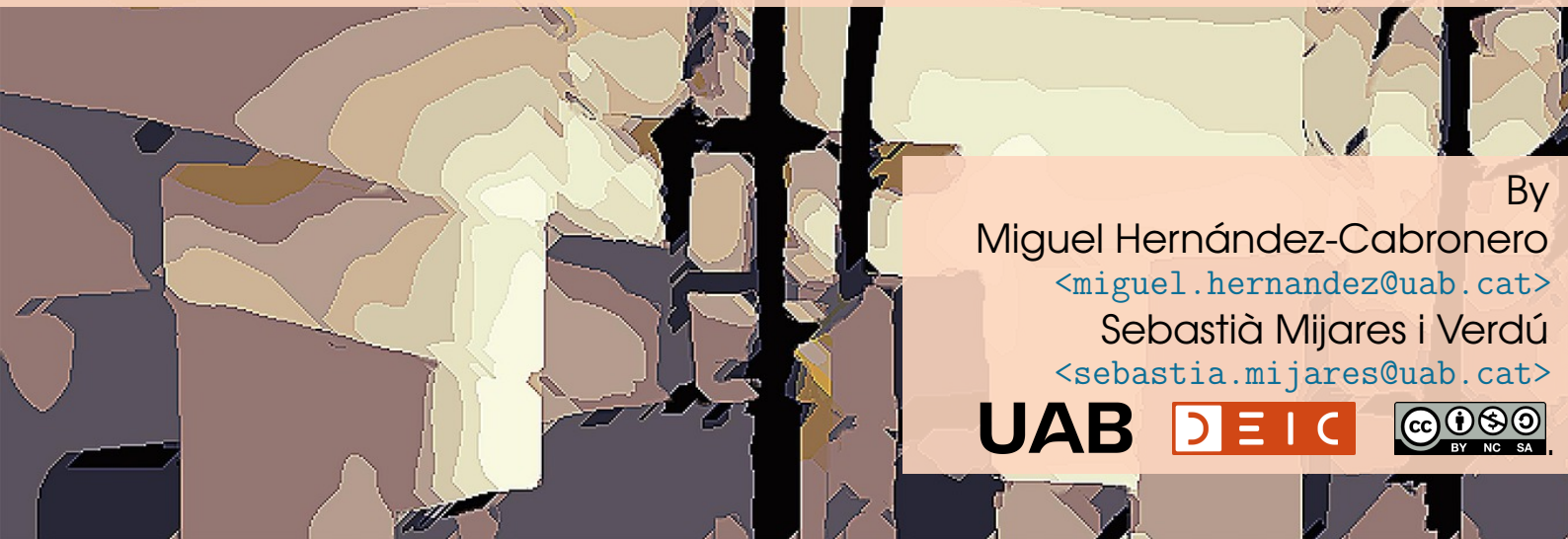
Data Transmission and Security

44731 — RICSE Master

Universitat Autònoma de Barcelona

Study Guide



By

Miguel Hernández-Cabronero

<miguel.hernandez@uab.cat>

Sebastià Mijares i Verdú

<sebastia.mijares@uab.cat>

UAB DEIC

This is document was compiled on July 31, 2025.

License

Credits

Many of the visual contents were created by third-party artists and released under compatible licenses:

- Mathias Legrand and Vel: base LaTeX template.
- WaveGenerics: Front cover.
- victorsteep: Chapter 1 header.
- TheDigitalArtist: Chapter 2 header.
- wynpnt : Chapter 3 header.
- Perchance : Chapter 4 header.
- LenaDu : Chapter 5 header.
- wal_172619_ll : Chapter 6 header.
- Schwarzwaldandi : Chapter7 header.
- M.C. Escher : Chapter 8 header.
- BrianPenny : Chapter 9 header.
- jarmoluk : Index of Concepts header.

Contents

1 Intro: Data Transmission and Security 4

2 Data 5
2.1 Digital media 5
2.2 Data representation, raw formats and metadata 5

3 Information 7
3.1 Elements of transmission 7
3.2 Surprise and entropy 7

4 Entropy coding 10
4.1 Codes and compressibility 10
4.2 Practical trade-offs 11
4.3 Universality and Adaptativity 11

5 Quantization 13
5.1 Symbol grouping 13
5.2 Popular quantizers 13

6 Performance analysis 16
6.1 Spacetime 16
6.2 Distortion 17

7 Prediction 19
7.1 Residuals 19
7.2 Linear and nonlinear 20

8 Transforms 22
8.1 Transform coding 22

9 Machine Learning data compression 24
9.1 Introduction to neural networks 24
9.2 Autoencoders for image compression 24

1. Intro: Data Transmission and Security

The subject

The goal of this subject is to help you acquire engineering and research competences in the area of data transmission. The course focuses on the efficiency and security of this process, with special attention to data compression.

Successful students will be able to design, analyze and evaluate data transmission solutions in realistic scenarios. To do so, they will be expected to understand the theoretical principles behind those systems (particularly surrounding the area of Information Theory), and to develop their own software toolboxes to perform quantitative analysis and evaluation.

The guide

This guide is designed to help students find and organize new information related to the course. This information should be actively and individually gathered by each student. The guide is **not** a book, a reference material or even a complete set of class notes, and **it is not intended to substitute your own notes**. You are expected to complete them with the proposed materials, including but not limited to (1, 2, 3).

> Full bibliographic entries are provided at the end of the document. Most of these are all freely accessible using your UAB credentials (see the Campus Virtual).

Sessions

The course is structured in 7 incremental units. Each one contains one or more of the following session types:

- **Discover** sessions: students will be exposed to new concepts via oral discussions and selected materials. Short, individual exercises will be proposed as homework until the next session.
- **Deepen** sessions: after a Discover session, students will discuss their solution to the proposed exercises and explore further, progressively more complex scenarios in one or more Deepen sessions.
- **Develop** sessions: at the end of each Discover-Deepen-Develop unit, one session will be devoted to autonomous, semi-supervised practice of the concepts pertaining to that (or previous) units.

8×	2h **Discover sessions**
9×	2h **Deepen sessions**
7×	2h **Develop (challenge) sessions**
Unlimited	**Office hour sessions** (*tutories*), individual or group: send email
92h	**Autonomous work** (total, expected)

A temporal plan describing the units and the session types is available in the Campus Virtual of the subject.

Evaluation

The final course grade is given by the arithmetic mean of the 7 tests taken during these many *Develop* sessions, *i.e.*, $\frac{1}{7}\sum_{i=1}^{7} D_i$, where D_i is the i-th test score.

2. Data

This unit introduces the notion of digital data and discusses multiple non-trivial aspects that need to be considered to read/receive, manipulate and write/send them.

2.1 Digital media

Digital images typically consist of one or more matrices of integer data (4). Each of these matrices is a band (also known as component). Each element (cell) of the matrix is a pixel. Indices are often referred to using their spatial position (x, y) and the index of their band, *e.g.*, $I_{x,y,z}$. If only one band is used, pixels can be referred to using only the spatial indices, *e.g.*, $I_{x,y}$.

One-band images are grayscale (sometimes loosely referred to as monochrome). In them, pixels with low values are often dark (black being the lowest possible value). Conversely, pixels with high values are bright (white being the highest possible value).

When more than one band is present, each band typically contains information about a limited range of light wavelengths. For instance, 3-band images are often RGB, meaning that bands describe the intensity of red, green and blue light, respectively. This means that the maximum pixel value in these components represent the maximum intensity of red, green and blue light, respectively.

There also exist images with more than 3 bands. Tens, hundreds and even thousands of bands can be present, *e.g.*, when produced by multispectral and hyperspectral sensors onboard satellites, or by special types of microscope. In this case, each band represents a narrow band of wavelengths, and the maximum pixel value indicates the maximum intensity of photons of those wavelengths.

Many other types of digital media exist, including video, audio, electrocardiograms (ECGs), seismographic and other time series, *etc.* It is often useful to consider these data as samples of 1D functions (*e.g.*, audio channels, ECGs), 2D functions (*e.g.*, monochrome images), 3D functions (*e.g.*, color images and monochrome video), 4D (*e.g.*, color video) and even higher-dimensionality data (*e.g.*, multispectral video and database records). In almost all cases, these digital data are also integer values although they can also be floating point/real.

2.2 Data representation, raw formats and metadata

When storing data in a file or transmitting them over the network, samples need to be arranged in a 1D pattern, *i.e.*, they need to be serialized. Raster, band-sequential BSQ, band-interleaved by line BIL and by pixel BIP data orders (5, 6, 7, 8) are the most prevalent.

Before applying compression, samples are often (temporarily) stored using fixed-length representations, hereafter referred to as raw representations or raw data. Longer (higher bitdepth) representations permit larger dynamic ranges, *i.e.*, a larger range of possible values for the samples. For instance, an image in which pixels can take one of 256 possible values may be represented using 8-bit samples. Another image in which pixels can take 1024 possible values (thus allowing many more colors or shades of gray and a better quality) would require at least 10 bits per sample when stored in raw format.

Typically, samples are stored using full bytes (8, 16, 32 and 64 bits) for efficiency reasons, so that programs can read them directly without having to process individual bits, *e.g.*, from the most significant bit (MSB) to the least significant one (LSB). Regardless of their bitdepth, samples

can be signed (*e.g.*, for audio) or unsigned (*e.g.*, most image data). For multi-byte samples, it is necessary to specify the byte ordering: big endian (*a.k.a.* network order), little endian (used by many CPU architectures).

Raw formats are not compressed and require custom software tools (*e.g.*, Imagej (9)). Thus, they are intermediate formats that simplify the input/output of other tools, particularly for data compression. Since raw formats contain no headers, metainformation about the sample geometry and data type must be stored separately as side information, *e.g.*, as a part of the file name or in a separate file.

Further reading and practice

Check the "References" section of the Campus Virtual if you have trouble finding any reference.

- (4): how sample values are stored in disk/memory.
- (10): how digital data is structured in bits and bytes, and their meaning.
- (5): basics of digital images ("rasters") and their properties. See in particular the "General properties of rasters" and "Raster Graphics" sections.
- (6, 7, 8): further info on BSQ, BIL and BIP data orders.

Exercise 2.1 Complete the "Quiz: Data I/O" activity on the Campus Virtual.

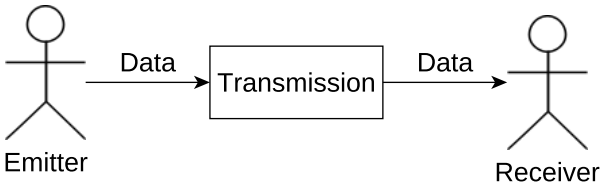
Exercise 2.2 Grab the mandrill image sample (`mandrill-u8be-3x512x512.raw`).

- Open it, *e.g.*, with ImageJ, and visualize its colors.
- Read the red, green and blue pixel values at position x=100, y=200 of the mandrill image and print the values.
- Count the number of bits set to 1 in the image. Print that number.
- Save a version of the mandrill image after setting all the most significant bits (MSBs) of the green component to 1 (name it `mandrill_t93_u8be-3x512x512.raw`).
- Save a version of the mandrill image after setting all the least significant bits (LSBs) of the blue component to 0 (name it `mandrill_t94_u8be-3x512x512.raw`).
- Save a version of the original mandrill image using signed integers, 16 bits per sample. Also, set the pixel at position $(0, 0, 0)$ to `0x0000` and the pixel at $(1, 0, 0)$ to `0xFFFF`. Save it as `mandrill_t95_s16be-3x512x512.raw` and open it with ImageJ.

3. Information

The main goal of this chapter is to formalize the concept of information (distinguishing it from plain data), and link it to the notion of compressibility.

3.1 Elements of transmission



The most simple transmission scheme involves an emitter, a receiver, and the data that is is produced by a source and transmitted through a channel (11, 12). Three key aspects of this transmission are its efficiency, its exactness and its privacy/authenticity. These are typically achieved thanks to the tools provided by, respectively, data compression, error correcting codes and cryptography/computer security. This course deals mainly with the first class of tools and their links to the third class.

The source $\mathcal{S}$ is the element that produces a sequence of symbols $s_1, s_2, \ldots, s_N$ from a finite alphabet $\mathcal{A} = \{\sigma_1, \sigma_2, \ldots, \sigma_{|\mathcal{A}|}\}$, *i.e.*, $s_i \in \mathcal{A}\ \forall i$. To characterize the source, we need to know the probability distribution of all possible symbols, *e.g.*, $P(\sigma_i)\ \forall i$. Often, symbol independence and a constant or stationary (1, §4.1) distribution during symbol production is assumed for simplicity, although it is not always the case (1, §2.2).

Symbols can be expressed using a fixed-length, uncompressed representation, as described in Chapter 2. While simple, this approach is generally not as efficient as it could. Data compression is the process of expressing a sequence of symbols in a compact representation more suitable for transmission. There are several strong motivations to use data compression (1, §1). These include limited bandwidth, transmission time, storage capacity and overall operational cost. In each scenario, we need to decide whether to use lossless compression (1, §1.1.1) or lossy compression (1, §1.1.2), weighing the pros and cons of each one.

3.2 Surprise and entropy

When a symbol σ_i with probability $P(\sigma_i)$ is produced by a source, the amount of information (surprisal, *a.k.a.* self-information), conveyed by that symbol is $I(\sigma_i) = \log_2(1/P(\sigma_i)) = -\log_2(P(\sigma_i))$ bits. The more surprising that symbol is (*i.e.*, the lower its probability), the more informative the appearance of that symbol is.

The logarithm in this expression is intuitively motivated by the fact that B bits can be used to distinguish between 2^B equiprobable (*i.e.*, $P(\sigma_i) = 1/2^B$ symbols), and the self-information of any of those symbols would be $I(\sigma_i) = B$ bits (1, §2.2),(11, §VI). The use of the logarithm also guarantees that the occurrence of two independent symbols σ_i and σ_i conveys an amount of information equal to the sum of their self-informations, *i.e.*, $I(\sigma_i, \sigma_j) = -\log_2(P(\sigma_i, \sigma_j)) = -\log_2(P(\sigma_i)P(\sigma_j)) = -\log_2(P(\sigma_i)) - \log_2(P(\sigma_j)) = I(\sigma_i) + I(\sigma_j)$.

The information produced by a stationary source is given by the weighted average of its symbols' self-information (11), *i.e.*, its first-order entropy: $\mathcal{H}(\mathcal{S}) = \sum_i P(\sigma_i)I(\sigma_i) = -\sum_i P(\sigma_i)\log_2(P(\sigma_i))$.

The minimum entropy possible is 0, which corresponds to the case of a single, deterministic outcome and no information conveyed. The maximum entropy given $\mathcal{A}$ is obtained for the uniform distribution case. Based on this, the upper bound is $\log_2(|\mathcal{A}|)$ bits (13, Theorem 2.6.4),(2, §2.1.2). One way to obtain this result is using Lagrangian optimization

$$\begin{aligned} J &= -\sum_{i=1}^{|\mathcal{A}|} P(\sigma_i)\log_2(P(\sigma_i)) + \lambda(\sum_{i=1}^{|\mathcal{A}|} P(\sigma_i) - 1), \\ 0 = \frac{\partial J}{\partial P(\sigma_i)} &= -(\log_2(P(\sigma_i)) + 1) + \lambda \Rightarrow P(\sigma_i) = e^{\lambda - 1}, \\ 1 = \sum_{i=1}^{|\mathcal{A}|} P(\sigma_i) &= |\mathcal{A}|e^{\lambda-1} = |\mathcal{A}|P(\sigma_i) \Rightarrow P(\sigma_i) = 1/|\mathcal{A}|. \end{aligned}$$

The maximum entropy case is particularly important when producing cryptographical keys to prevent an attacker from making educated guesses about the chosen key (14, Theorem 2.4).

Real sources don't typically produce symbols independently. Instead, the probability of the next symbol depends on what symbols have been already produced (11, §3) (think text or natural images). In this case, a higher-order entropy definition that considers a context of previously observed symbols is needed to assess the amount of information conveyed. For instance, one can use a context comprising the previously emitted symbol and calculate $P(\sigma_i) = \sum_j P(\sigma_j)P(\sigma_i|\sigma_j)$.

Further reading and practice

- (1): All things data compression, very useful to the course. It presents things in rigorous but not too mathy way. Freely available (online and offline) for all enrolled students. Chapters 2.1 and 2.2 are of special interest for this Unit.
- (3, §3): Section 3 provides a mathematically casual description of entropy, with minimal math "interference". Highly recommended as well.
- (13): A very complete manual on Information Theory, which deals with many of the concepts of the course from a mathematically rigorous point of view. It is also freely available online. The following chapters are of special interest for this Unit:
  - Chapter 2.1: notion of entropy. Read the rest of Chapter 2 for properties and related definitions.
  - Chapter 3.1: the Asymptotic Equipartition Property Theorem (Theorem 3.1.1), which justifies how we calculate entropy in practice
  - Chapter 4.2: definition and examples of the entropy rate of a sequence of n variables. Read the introduction to Chapter 4 and Chapter 4.1 for more information on sequences of n variables (Markov chains).
- (11): seminal paper by Claude Shannon first describing a theory of communication. It introduces the main elements of transmission (source, transmitter, channel, uncertainty, entropy, etc. It is a bit dry, math-oriented and describes many concepts beyond this course, but it is extremely relevant historically.
- (12): Relatively gentle description of the contents of Claude Shannon's paper ((11), see above). Chapters 1 to 4 deal with concepts most relevant to this course.

Exercise 3.1 Describe the sources and alphabets that best model the transmission of:

1. an 8-bit grayscale image, uncompressed
2. an 8-bit RGB image, compressed
3. a 12-bit multispectral image, stored using 2 bytes per sample
4. a 12-bit multispectral image, stored using 4 bytes per sample

Exercise 3.2 Model the source that produced the `mandrill-u8be-3x512x512.raw` image. Assume symbol independence.

Exercise 3.3 Discuss the preferability of lossless vs. lossy compression in the following scenarios:

1. Natural images
2. Digital pathology images
3. Remote sensing data
4. Text (*e.g.*, a book)
5. Program code
6. Audio

Exercise 3.4 How much information is conveyed by the following experiments:

1. Flipping 1 fair coin.
2. Flipping 1 fair coin, then another.
3. Flipping 2 fair coins simultaneously.
4. Flipping 1 fair coin and 1 fair die.
5. Checking the status of an alarm that is broken (and can never go off).
6. Checking the status of an alarm that is broken (and is constantly going off).

Exercise 3.5 How many bits do we need on average to express an outcome of a binary source ($|\mathcal{A}| = 2$) that produces one symbol three times as often as the other? If the number is not an integer, explain why.

Exercise 3.6 Model the source that produced the `mandrill-u8be-3x512x512.raw` image assuming that the probability of each pixel depends on the immediately previous emitted pixel (and no other pixel).

4. Entropy coding

The main goals of this chapter is to formalize the most important concepts about codes, to link the notions of entropy and compressibility, and understand the role of entropy coding in data compression pipelines.

4.1 Codes and compressibility

Source coding is the process of transforming a message, *i.e.*, a sequence of input symbols $s_1, \ldots, s_N$ into a compact, generally binary, representation. To be useful, this process must be invertible, *i.e.*, it must be possible to (uniquely) retrieve the original symbols from the compact representation.

Variable-length codes transform each input symbol σ_i into a codeword $C(\sigma_i)$, whose length L_i depends on that symbol's probability. The code can be expressed as a function, *e.g.*, $C : \mathcal{A} \longrightarrow \{0,1\}^*$.

To reduce the compressed data size, the most probable symbols are assigned codewords of smaller lengths, and vice-versa (1, §2.4, §2.4.1, §2.4.2). Given the length L_i of each symbol σ_i, the code's expected length of a code is obtained as $L_C = \sum_i P(\sigma_i)L_i$ (13, §5.1).

Prefix codes are those for which no $C(\sigma_i)$ is the prefix of another $C(\sigma_j)$. This property allows both instantaneous and unique decoding, making it highly desirable for practical data compression.

Shannon proved that a code C's expected length L_C must be at least $\mathcal{H}(\mathcal{S})$ (13, §5.4). One way to reach this result is by using Kraft-McMillan's inequality ($\sum_i D^{-L_i} \leq 1$ (1, §2.4.3)) and Lagrangian optimization:

$$
\begin{aligned}
J &= \sum_i P(\sigma_i)L_i + \lambda \sum_i D^{-L_i}, \\
0 = \frac{\partial J}{\partial L_i} &= P(\sigma_i) + \lambda D^{-L_i} \cdot (-1) \cdot \log D \Rightarrow D^{-L_i} = \frac{P(\sigma_i)}{\lambda \log D}, \\
1 \geq \sum_i D^{-L_i} &= \frac{1}{\lambda \log D} \sum_i P(\sigma_i) = \frac{1}{\lambda \log D} \Rightarrow \lambda \geq \frac{1}{\log D}, \\
D^{-L_i} &= \frac{P(\sigma_i)}{\lambda \log D} \leq P(\sigma_i) \Rightarrow L_i \geq -\log P(\sigma_i) \Rightarrow L_C = \sum P(\sigma_i)L_i \geq \mathcal{H}(\mathcal{S}).
\end{aligned}
$$

It is always possible to find a prefix code that satisfies $L_i = \lceil -\log_2(P(\sigma_i)) \rceil$ (1, §2.4.2, §2.4.3), *e.g.*, using Huffman's algorithm (see below). Thus, given a source with entropy $\mathcal{H}(\mathcal{S})$, it is always possible to find a prefix code that satisfies $\mathcal{H}(\mathcal{S}) \leq L_C \leq \mathcal{H}(\mathcal{S}) + 1$. Furthermore, by coding M symbols at once, the bound can be tightened as much as desired: $\mathcal{H}(\mathcal{S}) \leq L_C \leq \mathcal{H}(\mathcal{S}) + 1/M$ (and $\lim_{M \to \infty} L_C = \mathcal{H}(\mathcal{S})$). Arithmetic coders (also see below) are a prime example of this.

A poorly designed code will have a larger expected length than the data's actual entropy. In particular, this will happen if we optimize a code for a probability distribution P, but the source $\mathcal{S}$ has a different distribution Q. The Kullback-Leibler (KL) "distance", *a.k.a.* relative entropy or cross entropy, provides the expected overhead, *i.e.*, the extra bits needed per symbol on average (13, §2.3, Theorem 5.4.3). This distance is defined as

$$
D(P \parallel Q) = \sum_i P(\sigma_i) \log_2(P(\sigma_i)/Q(\sigma_i)),
$$

using the convention $0 \log_2(0/0) = 0$, $0 \log_2(0/q) = 0 \; \forall q$, and $p \log_2(p/0) = \infty \; \forall p > 0$.

4.2 Practical trade-offs

Entropy coding is a key stage of most compression pipelines. This stage is lossless, although it can be preceded by other lossy stages to enhance compression at the cost of introducing some distortion in the data (see Chapters 5 and 6).

In order for those pipelines to be practical, the algorithm selected for entropy coding must strike an acceptable trade-off between efficiency (defined as $\eta = \frac{\mathcal{H}(\mathcal{S})}{\text{compressed bitrate}}$) and computational complexity.

On one end of the spectrum, there are relatively simple algorithms designed to run fast requiring few resources (including energy). Often, these algorithms code each input sample independently, and work best when these samples have been previously decorrelated (*e.g.*, as described in Chapter 7). Their trade-off is that they attain $\eta = 1$ only for limited classes of probability distributions. Prime examples of this approach are:

- Huffman's algorithm (1, §3.2–3.3), (2, §2.2.1), (13, §5.6–5.7).
- Golomb-Rice codes (1, §3.5–3.6), employed for instance in the CCSDS 121.0-B-3 standard (15, §3.2–3.3).
- Run-Length Encoding (RLE) (16).

On the other side of the spectrum, there are algorithms designed to consistently achieve high efficiencies approaching $\eta = 1$ for most distributions, generally at the cost of higher computational complexities. To do so, instead of coding individual symbols, they process a sequence of symbols and produce a single output codeword. Traditionally, the main paradigm used for this purpose is Arithmetic Coding (1, §4),(2, §2.3), sometimes referred to as range coding. Some prime examples of this paradigm are:

- the CABAC entropy coders of the popular video codecs H.264 (AVC, 2003), H.265 (HEVC, 2013) and H.266 (VVC, 2017),
- the MQ entropy coder of JPEG 2000,
- the PAQ family of entropy coders, among others.

More recently, thanks to Jarosław Duda's contributions (17), entropy coding has been shifting towards Asymmetric Numeral Systems (ASNs), specially its Finite State Entropy (FSE) realizations. Some important examples of ASN are:

- Zstandard (*a.k.a.* zstd) (18),
- LZFSE (19),
- JPEG XL (20), and
- CCSDS 123.0-B-2 (21), among many others.

4.3 Universality and Adaptativity

The most basic versions of the entropy coders mentioned in Chapter 4.2 rely on two main assumptions: the probability distribution of the source $\mathcal{S}$ is known *a priori*, and that probability does not change throughout the compression process. Neither of these assumptions hold generally in practice, and a two-pass approach is not necessarily practical.

In the late 70's, Ziv and Lempel introduced the LZ77 (1, §5.4.1) (22) and LZ78 (1, §5.4.1) algorithms, often referred to as zip compression, which dynamically build a dictionary of previously seen sequences. As new symbols are input, symbols can be efficiently represented referencing that dictionary. These algorithms seeded a vast class of compressors that is still relevant and growing today.

Algorithms such as Huffman and Arithmetic Coding can be extended so that they use adaptive probability models. While still reading each input sample only once, these algorithms change their estimation of each symbol's probability based on previously observed samples:

- With Huffman, for instance, input samples can be divided into blocks, and the statistics gathered for one block can be used to generate the code table for the next block.

- With an Arithmetic Coder, each sample can be used to update the probability range division characteristic of this algorithm. It is also typical to employ contexts to further refine the probability model and better exploit any redundancy present in the inputs, like CABAC does, and as previously mentioned in Chapter 3.2.

Further reading and practice

- Sayood's book (1) provides invaluable and very accessible insight on most topics of this unit, particularly in Sections 2.4, 3.2, 3.5 and 3.6.
- Taubman and Marcellin's book (2) offers a somewhat more rigorous (not that Sayood's isn't!) treatment of these algorithms, often from the perspective of image compression, in Sections 2.2, 2.3, and 2.4.
- A thorough analysis of universal coders (including Arithmetic and LZ coding) can be found in (13, §13).
- The book by McAnlis and Haecky provides a very clear description of tANS, one of the main variants of asymmetric numeral systems (3, §5.5). Further mathematical insight, including its relation to computer security (cryptography), can be found in Duda's paper (17).

Exercise 4.1 Consider the four following codes for a four-symbol alphabet $\mathcal{A} = \{a, b, c, d\}$:

Code $\mathbf{C}$	$\mathbf{C(a)}$	$\mathbf{C(b)}$	$\mathbf{C(c)}$	$\mathbf{C(d)}$
C_1	0	0	0	0
C_2	0	010	01	10
C_3	10	00	11	110
C_4	0	10	110	111

- Which of the codes are usable?
- Which compresses best?

Exercise 4.2 What's the minimum and maximum overhead of Huffman coding? When's the minimum overhead achieved? How about the maximum?

Exercise 4.3 Is RLE better suited for high-entropy or low entropy sources?

Exercise 4.4 What type of distribution are Golomb codes optimal for?

Exercise 4.5 Find the distribution of English characters. Based on that probability model, encode your name using

1. Huffman coding
2. Arithmetic coding

If needed, romanize your name so that it contains only English caracters (*e.g.*, using Pinyin).

Exercise 4.6 Find implementations of a couple of entropy coders among the described in this section, *e.g.*, at:

- `https://github.com/nayuki/Reference-arithmetic-coding`
- `https://github.com/nayuki/Reference-Huffman-coding`
- `https://github.com/Cyan4973/FiniteStateEntropy`
- `https://github.com/miguelinux314/pyac`

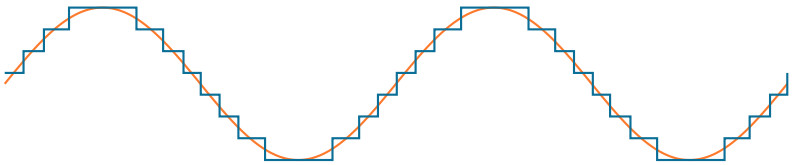
Apply them to the `mandrill-u8be-3x512x512.raw` image and compare the actual compressed size against its entropy.

5. Quantization

This chapter presents quantization as a key stage of lossy compression pipelines, and introduces some of the main examples used in practice.

5.1 Symbol grouping

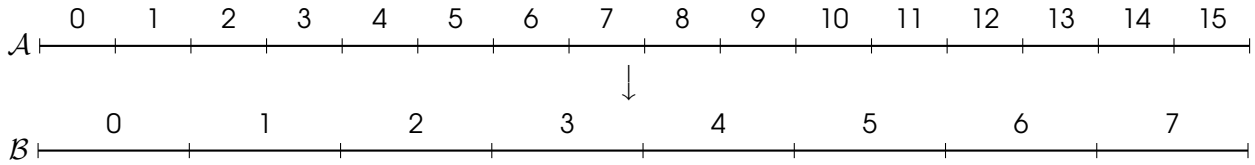
Consistent with Chapters 2.1 and 3.1, digital data can be regarded as a collection of samples from a finite alphabet $\mathcal{A}$. When those data represent a physical signal such as light intensity or sound pressure, the more elements in $\mathcal{A}$, the more precise that representation can be.



Quantization is a process that splits the input alphabet into groups, and outputs the same quantization index for all elements of the same group. This can be expressed as a scalar function:

$$\begin{aligned} Q : \mathcal{A} &\longrightarrow \mathcal{B} \\ x &\longrightarrow Q(x). \end{aligned}$$

An example quantizer function is shown next:



Applying quantization to the symbols produced by a source $\mathcal{S}$ can be regarded as having a new source $\mathcal{T} = Q(\mathcal{S})$ that produces symbols (quantization indices) from smaller alphabet $\mathcal{B} = Q(\mathcal{A})$. The new source $\mathcal{T}$ has a different probability distribution. In general, $\mathcal{H}(\mathcal{T}) < \mathcal{H}(\mathcal{S})$, which is why quantization is included in data compression pipelines.

Along with a quantization function Q, one needs to define a dequantization function $Q^{-1} : \mathcal{B} \longrightarrow \mathcal{A}$ that decides the reconstruction point of each quantization interval. This reconstruction point is normally one of the input symbols that is quantized to that interval, *i.e.*, $Q^{-1}(y) \in \{x \in \mathcal{A} : Q(x) = y\}$.

After a symbol is quantized, one can only know what group that symbol belongs to, but not which symbol exactly. Therefore, quantization is a lossy process (often the only one in a lossy compression pipeline (2, §3.1)). Once Q and Q^{-1} are defined, one can compute the expected quadratic error as $E[(X - Q^{-1}(Q(X)))^2] = \sum_{\sigma \in \mathcal{A}} P(\sigma) \left(\sigma - Q^{-1}(Q(\sigma))\right)^2$.

5.2 Popular quantizers

The most simple, nontrivial quantizer is uniform scalar quantization (1, §9.4). As the name implies, all quantization intervals have the same length. This length is often referred to as the quantization step, *a.k.a.* qstep, and typically denoted Δ. In spite of its simplicity, uniform quantization is close to optimal (in terms of rate-distortion) for high enough rates (*i.e.*, for small enough intervals) (2, §3.2.4, Fig. 3.5). When signed inputs are considered, there are two possibilities:

- Midrise (1, Fig. 9.3): when $\nexists y \in \mathcal{B} \ : \ Q^{-1}(y) = 0$, and
- Midtread (1, Fig. 9.5): when $\exists y \in \mathcal{B} \ : \ Q^{-1}(y) = 0$.

One very desirable feature of compression systems is the possibility of sending some data first (which enables the reconstruction at a certain quality), and later on sending more data to enhance that quality. This feature is called progressive transmission (or progressive reconstruction, depending on the point of view). An efficient way of achieving this is by using embedded quantization:

- To build an embedded quantizer, a quantizer Q_1 is defined on the original alphabet $\mathcal{A}$. After that, a second quantizer Q_2 is defined on the quantization intervals produced by Q_1, *i.e.*, on $Q_1(\mathcal{A})$. As a result, the quantization intervals defined by Q_1 are contained –embedded– in Q_2's. This process is repeated until the last (and coarsest) quantizer Q_D is defined on the intervals given by $Q_{D-1}(Q_{D-2}(\cdots Q_1(\mathcal{A})))$.
- To apply an embedded quantizer, the emitter applies the coarsest quantizer Q_D first. The resulting quantization intervals can be transmitted, and used for reconstruction by the receiver. After that, for each transmitted quantization index, additional information can be send so that the receiver knows to which of Q_{D-1}'s intervals the original sample belongs.
- The most common embedded quantizer is one where each Q_i is a uniform scalar quantizer with qstep 2, and the last quantizer has 2 quantization intervals (2, §3.2.6). In this case, a single bit is sufficient to select the first quantization interval. Each additional bit allows selecting an interval of the next quantizer, effectively halving the qstep. This process is equivalent to successively transmitting the bitplanes of the original data, from most to least significant. These bitplanes can (and generally are) entropy coded with a binary entropy coder. Note that embedding is not restricted to uniform scalar quantizers, and generalizations have been described (23).

Nothing forces us to have all quantization intervals have the same size. In fact, non-uniform quantizers are required to achieve optimality in the general case.

- One very elegant example of a non-uniform scheme is deadzone quantization (2, Eq. 3.30). Usually, this approach is constructed as a uniform quantizer with an enlarged (*e.g.*, twice the size), zero-centered interval. This case is called a deadzone uniform scalar quantizer – but it is not actually uniform as defined above because of that enlarged central interval. This approach yields enhanced performance for zero-mean probability distributions, which is often the case after prediction (see Chapter 7) and transforms such as the DWT (see Chapter 8).
- If the probability distribution of the source $\mathcal{S}$ is known, one can use this information to design the quantizer that minimizes $E[\left(X - Q^{-1}(Q(X))\right)^2]$. The Lloyd-Max algorithm (2, §3.2.1), (1, §9.6.1) can be used for this purpose, which yields optimal quantizers (in the rate-distortion sense) assuming no entropy coding is applied after quantization. For the case where entropy coding *is* applied, there exists a generalized version of this algorithm (2, §3.2.3), (1, §9.7.1).

Further reading and practice

- Sayood's book (1) is an excellent companion for this section, particularly §9.1–9.4, 9.6 and 9.7.
- A generalization of embedded quantization was described in detail in (23).
- Another example of a non-uniform quantizer designed for a particular purpose can be found in (24).
- Taubman and Marcellin's book extends the contents of this Chapter including non-scalar quantizers (2, §3.4) and Trellis Quantization (2, §3.5).

Exercise 5.1 Mathematically define the quantization function Q and its corresponding reconstruction function Q^{-1} for each of the main types mentioned in the chapter.

Exercise 5.2 A source $\mathcal{S}$ produces elements from an alphabet $\mathcal{A}$ with entropy $\mathcal{H}(\mathcal{S})$. We apply a uniform scalar quantizer with step 2. What is the minimum and maximum entropy of the output?

Exercise 5.3 Given a source $\mathcal{S}$ and a quantization function Q, how would you choose the optimal reconstruction point for each quantization interval?

Exercise 5.4 What's the relation between $\mathcal{H}(Q(\mathcal{S}))$ and $\mathcal{H}(Q^{-1}(Q(\mathcal{S})))$?

Exercise 5.5 Looking only at the value distributions (histograms), how would you identify the original data from a source $\mathcal{S}$, the quantization indices produced by a quantizer Q, and the reconstructed data $Q^{-1}(Q(\mathcal{S}))$?

Exercise 5.6 Split the mandrill image sample (`mandrill-u8be-3x512x512.raw`) into its bitplane constituents. What qualitative differences do you observe between the most and least significant bitplanes?

6. Performance analysis

The goal of this chapter is to present the main performance metrics (objective and subjective) employed to evaluate lossless and lossy compression pipelines.

6.1 Spacetime

Lossless compression pipelines are relatively straightforward to analyze. By definition, they preserve all information and allow a perfect reconstruction of the original data. Therefore, the compressed data size and required computational resources typically offer sufficient insight.

One of the most useful metrics to assess compressed data size is the compressed data rate, expressed in bits per sample (bps), and defined simply as $\frac{\text{compressed data size (bits)}}{\text{number of original samples}}$. Depending on the context, the generic term "sample" is substituted by the specific sample type. For instance, the term bits per pixel (bpp) is used in image compression, whereas bits per base (bpb) is used in DNA sequence compression.

> In the context of image compression, sometimes the term "pixel" refers to a spatial position (x, y). For color and multispectral images that have multiple bands, there are multiple samples at each spatial position. To prevent potential ambiguity, one can use the bits per pixel per component (bpppc) and even bits per sample so that each (x, y, z) position is considered a separate sample.

One advantage of the bits per sample metric is an easy comparison with the source's entropy $\mathcal{H}(\mathcal{S})$. This way, it is easy to compute the efficiency of a compression pipeline, defined as $\eta = \frac{\mathcal{H}(\mathcal{S})}{\text{bits per sample}}$.

Another popular metric for assessing compression performance is the compression ratio. This is often defined as $\text{CR} = \frac{\text{original data size}}{\text{compressed data size}}$ and requires no units. With this definition, if compression compacts data to half the original size, the compression ratio would be 2, or equivalently $2:1$. Although simple, this metric has some potential pitfalls:

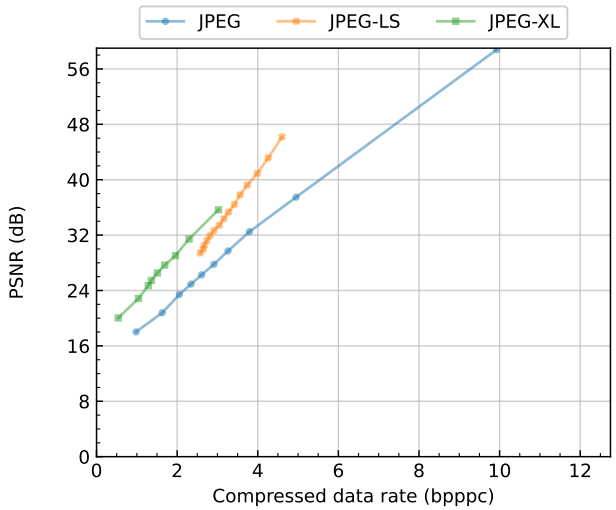
- Some authors define compression ratio as the inverse of the above definition. In the previous example, the compression ratio would be $1/2$ instead of 2.
- The "original data size" is sensitive to the way they are originally stored. Some factors that may affect this value are:
  - the bitdepth of the raw representation,
  - the presence or absence of headers, and
  - the original data be available in an already compressed format.

Another critical dimension for evaluating compression pipelines is the required computational resources. Of particular importance are compression/decompression time and memory usage. Compression time is particularly important in contexts such as remote sensing because of the limited time to process each sample, but also because time is a good surrogate metric to battery usage. Other derived metrics exist to evaluate compression speed, *e.g.*throughput, that combine execution time and the number of samples compressed.

To properly evaluate compression time, it is necessary to run multiple executions for each sample. After that, the average, median and minimum statistics can be used to summarize the measurements, although a distribution analysis can be even more informative.

6.2 Distortion

Lossy compression pipelines permit almost arbitrary compressed data size reductions, easily outperforming lossless schemes in that regard. However, this is at the cost of introducing some degree of distortion in the reconstructed data. In order to properly evaluate lossy compression, the compressed file size and the introduced distortion must be analyzed simultaneously. This rate-distortion analysis be performed for instance with a 2D plot such as the following one:



- The x axis displays a metric of compressed data size (*e.g.*, the compressed data rate).
- The y axis displays a metric of distortion (see some options below).
- Each marker in the plot corresponds to one compression pipeline with one specific set of parameters. The different markers of the same pipeline are often linked with a line for better clarity.
- For a test corpus with more than one sample (*e.g.*, more than one image), the x and y values can be the average rate and distortion across the corpus. It is also possible to include insight on the dispersion "hidden" by each marker (*e.g.*, displaying ± 1 standard deviation, or plotting a point of clouds instead of a single line per pipeline).

Rate-distortion plots allow quick comparison of different pipelines. In particular, they can help determine what pipeline yields a lower distortion (higher fidelity) for a given rate, and what pipeline achieves a certain quality with the smallest compressed data volumes.

A key element of these plots is the choice of distortion metric for the y axis. These metrics can be objective or subjective. Objective metrics perform a deterministic computation using the original X and the reconstructed $\hat{X}$ data. Subjective metrics, on the other hand, require humans to give their opinion on their perceived quality of the reconstructed data.

Most objective distortion metrics are directly linked to physical signal notions (such as error power). Some of the most relevant are listed next (1, §8.1):

- the mean squared error (MSE),
- the peak signal-to-noise ratio (PSNR),
- the peak absolute error (PAE), and
- the spectral angle (hyperspectral imagery (25, "The Spectral Angle Mapper (SAM)")).

While useful (26), the metrics above do not directly take into account the psychovisual and psychoacoustic properties of humans, so they often fail to predict the results of subjective distortions. Other metrics have been developed to address this problem, involving more or less complex models of the human sensory systems, including:

- the structural similarity (SSIM (27)),
- the multiscale structural similarity (MS-SSIM (28)) metrics, and
- the HDR-VDP-3 metric (29).

If interested, make sure to request a full presentation on perceptual coding from your closest data compression professor.

Further reading and practice

- Sayood's book (1, §8.3) offers a very clear overview of the most popular distortion criteria.
- Taubman and Marcellin's book (2) contains a brief but nice introduction to information irrelevance in §1.2.2, part of which is further developed in §4.3.4 and 16.1.
- This paper (26) provides invaluable insight and food for thought on the most popular distortion metric in many areas of data compression (MSE).

Exercise 6.1 What's the minimum and maximum efficiency η achievable by a scalar entropy coder? How about a complete compression pipeline?

Exercise 6.2 Provide matemathical expressions for the objective distortion metrics based on physical signal properties mentioned in this chapter.

Exercise 6.3 Find the distortion metrics used in these papers, paying close attention to the way results are presented:

a) (30)
b) (31)
c) (32)
d) (33)
e) (34)
f) (35)
g) (36)
h) (37)
i) (38)
j) (39)
k) (40)
l) (41)
m) (42)
n) (43)
o) (44)
p) (45)
q) (46)
r) (47)
s) (48)
t) (49)

Exercise 6.4 Rank the compression time of the entropy coders you used in Chapter 4 for the `mandrill-u8be-3x512x512.raw` image. Also provide the throughput in each case.

Exercise 6.5 Perform your own rate-distortion analysis using the mandrill image and at least two lossy compressors. For instance, you can use the JPEG and JPEG-LS codecs available at `https://github.com/thorfdbg/libjpeg/`.

7. Prediction

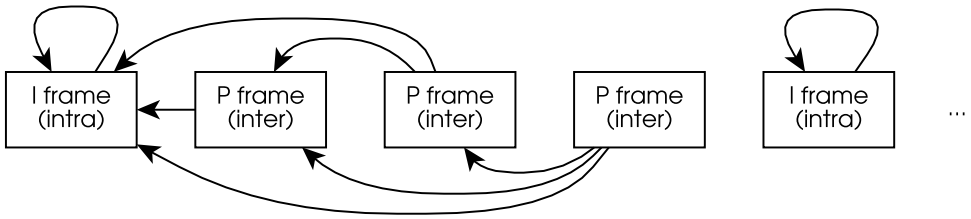
The goal of this chapter is to illustrate the utility of a prediction stage in data compression pipelines, and some of the main realizations.

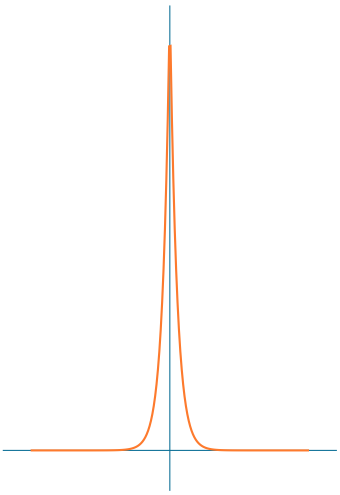
7.1 Residuals

Except for a brief digression in Chapter 4.3, symbols produced by a source $\mathcal{S}$ have been assumed to be independent. However, this does not hold for most real sources. For instance, the next word in a sentence is strongly biased by the previous characters, and image pixels tend to be similar to their neighbors and across bands. We can then say that there is redundancy present among those samples.

A prediction stage can be inserted before entropy coding to remove some of that redundancy with the hope that entropy is reduced. For each input sample x, the prediction stage "bets" on its most likely value $\hat{x}$ looking only at previously transmitted data (*i.e.*, using a causal context). Then, the prediction residual $\Delta = x - \hat{x}$ is computed, encoded and transmitted. The receiver should be able to reproduce the same predicted values $\hat{x}$ and combine them with the transmitted residuals to reconstruct the original value as $x = \Delta + \hat{x}$.

The context used for prediction can be as small or as large as the computational and memory constraints permit. These contexts can be 1D, 2D, 3D, 4D and beyond depending on the dimensionality of the input data (see for instance an example video prediction scheme below). They can also use a priori information about the dataset (*e.g.*, the average sample value). Whatever the context, the causality constraint is critical for the receiver to be able to reproduce the same predicted values $\hat{x}$.





A good predictor, assuming samples are highly correlated (high redundancy), produces many zero-valued residuals. The probability of ± 1 residuals, ± 2 residuals, etc., decays quickly, yielding a distribution akin to a Laplacian.
The better the predictor, the sharper the distribution shape and the lower the entropy of the resulting residuals.

Getting acquainted with this distribution is very useful in the context of data compression because of its high prevalence: it is not only produced by good predictions, but is also a good description of the coefficients produced by the discrete cosine transforms (DCT) and discrete wavelet transforms (DWT) explored in Chapter 8.

Information about this distribution is critical to design well-performing entropy coders, adaptive or not, as discussed in Chapter 4. It also motivates the use of deadzone quantization, as introduced in Chapter 5.2.

7.2 Linear and nonlinear

Prediction functions are often linear because of their implementation and optimization properties. Notwithstanding, they can also be non-linear as long as they can be perfectly replicated by the decoder.

The causal context can be employed to choose one among several possible predictors. The PPM (50) and LZ (22) (see Chapter 4.3) families of compressors exploit this fact, attempting to find perfect matches of variable length. These are chiefly useful for the compression of text data and other sources with small alphabets, especially when produced by a grammar. In turn, the PAQ (51) family of compressors tries multiple approaches and combines them using context mixing.

> Some of the algorithms in the previous paragraph are used to drive the probabilities of an adaptive arithmetic coder, but are included in this section by affinity.

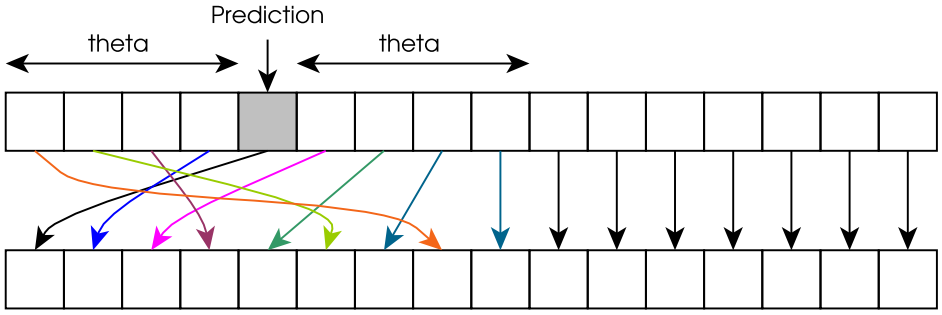
Another popular, nonlinear approach to prediction is to define piece-wise functions. Depending on which of a number of mutually-exclusive conditions is met by the causal context, a different prediction expression is used. This condition can, for instance, attempt to identify "flat" versus "edge" regions, and predict accordingly. Two main examples of this approach are CALIC (52) and JPEG-LS (53). The prediction function of the latter can be expressed as follows:

C	B	D
A	$\hat{x}$	

$$\hat{x} = \begin{cases} \min(A, B) & \text{if } C \geq \max(A, B) \\ \max(A, B) & \text{if } C \leq \min(A, B) \\ A + B - C & \text{otherwise} \end{cases} .$$

Regardless of the prediction method employed (assuming those predictions are reasonable), it is possible to map each residual Δ into the original data's dynamic range, $\Delta \mapsto \delta$. Equations (55) and (56) of (21, §4.11) describe one such way – a slightly simplified version for positive integers is shown next following the previously employed notation. Note how this mapping function requires not only the prediction residual, but also the value of the prediction itself.

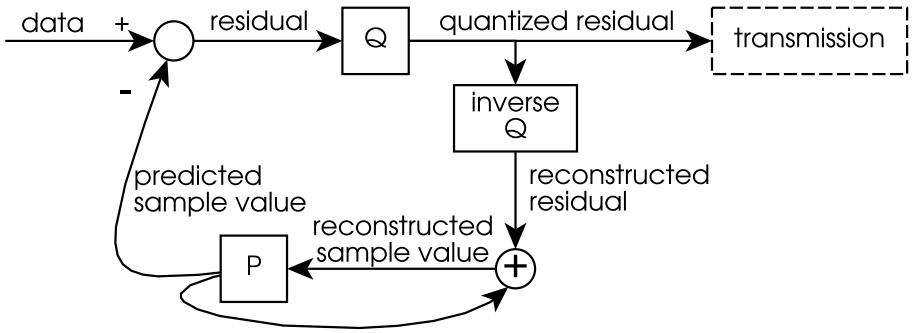
$$\begin{aligned} \theta(\hat{x}) &= \min(\hat{x} - x_{\min}, x_{\max} - \hat{x}) \\ \Delta(x, \hat{x}) &= x - \hat{x} \\ \delta(x, \hat{x}) &= \begin{cases} \theta + |\Delta| & \text{if } \hat{x} > \theta \\ 2|\Delta| & \text{if } |\Delta| \leq \theta \text{ and } |\Delta| \text{ is even} \\ 2|\Delta| - 1 & \text{if } |\Delta| \leq \theta \text{ and } |\Delta| \text{ is odd} \end{cases} \end{aligned}$$



A correctly applied prediction scheme does not introduce any data loss. That said, prediction can also be part of lossy compression pipelines that involve a quantization function Q. Quantization can be applied before or after prediction, introducing the same range of errors in both cases. Doing it before is computationally simpler, but rate-distortion theory predicts a lower mean squared error if quantization is applied afterwards (2, §3.3). Specifically, $\mathrm{MSE}_{\text{after}} = (\sigma^2_{\text{residuals}}/\sigma^2_{\text{data}})\mathrm{MSE}_{\text{before}}$, where $\sigma^2_{\text{residuals}}$ and σ^2_{data} are the variances of the original data and the prediction residuals, respectively.

The main drawback of applying quantization after prediction is the fact that the encoder needs to replicate the dequantization process in the compression loop, and use the reconstructed sample values for prediction instead of the original. Otherwise, the encoder and the decoder would be using a different set of values as inputs to the prediction function, and the

transmitted residuals (either Δ or δ) would not guarantee proper reconstruction. The modified encoder-decoder loop is described in (2, §3.3) and (1, §11.3) and illustrated next:



Further reading and practice

- Cleary and Witten's paper (50) introduced one of the most successful and widely extended prediction approaches. A gentler description is also available in (3, §9.2).
- The PAQ series, created by Mahoney, of algorithms use context mixing to provide slow but powerful prediction models. The original PAQ1 is introduced in (51), and further versions are referenced in the author's webpage `https://mattmahoney.net/dc/`. A good introduction to this family of algorithms is available in Salomon and Motta's book (54, §5.15).
- The CCSDS 123.0-B-2 standard (21) contains a highly efficient (albeit somewhat complex) predictor that adapts its weights based on the previously observed samples (and therefore the produced residuals).
- Sayood's book (1, §8.6.2) explains how to choose optimal linear prediction weights (in the MSE sense) based on the signal's autocorrelation.

Exercise 7.1 What's the highest and lowest entropy possible for a scalar prediction scheme applied to a source $\mathcal{S}$?

Exercise 7.2 Using the notation $I_{x,y,z}$ to refer to a pixel in a color image, provide the mathematical expression for a 1D, a 2D and a 3D predictor $\hat{I}_{x,y,z}$.

Exercise 7.3 In addition to natural text, what sources do you think could benefit from compression with LZ and/or PPM algorithms?

Exercise 7.4 How does the CALIC predictor compare to the LOCO-I used in JPEG-LS?

Exercise 7.5 Grab the mandrill image sample (`mandrill-u8be-3x512x512.raw`) and perform a simple, causal prediction.

- What is the dynamic range of the residuals?
- What is the entropy of the resulting residuals?
- What parameters (μ and b) best model the resulting distribution? You may want to use the `curve_fit` function described at `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html`.

Exercise 7.6 Consider a sequence of data that begins with $137, 39081, 4829, \ldots$. Apply the $\hat{x}_i = x_{i-1}$ prediction function followed by a uniform scalar quantizer with $qstep = 3$. Compare the obtained reconstruction errors for:

- the correct case that replicates the reconstruction process in the encoder, and
- the incorrect case that doesn't.

8. Transforms

This unit aims to motivate the use of transforms as a stage of the data compression pipeline and to get you acquainted with some important transform types and concepts: color, block, subband, etc.

8.1 Transform coding

Transform coding refers to a very common data compression pipeline structure. In it, the original data samples undergo a transform stage, which represents them in an alternative domain (the transform domain). After that, data are (normally) quantized and finally entropy coded. Transform coding is most useful for lossy compression (whereas prediction is most useful for lossless or near-lossless compression). The two main advantages of applying a good transform are:

- Independent transform, quantization and entropy coding stages is the fact that they simplify the design, implementation and execution of compression pipelines. Most modern, practical data compression algorithms follow this scheme for that reason.
- Data can be better compressed by reducing statistical dependencies is advantageous because it tends to compact the energy/information in some parts of the transformed data (bands/subbands), leaving other parts with little energy/information (2, §4.3.1). This compaction allows obtaining better fidelity at the same rate, and the gain is larger the more we compact energy/information in fewer bands (1, §13.3, eq. 27)(2, p. 207). This is because we can quantize lesser energy/information bands and cause lower impact on the fidelity than if we quantize the higher energy/information bands (1, §13.3)(2, "Principle Components and the KLT").
- Data can be better compressed by separating important and unimportant information allows improving compression ratios while discarding only little-to-no relevant details. One main way of doing this is by separating different frequencies into different bands. This way, high-frequency noise and noise-like information can be discarded while keeping the core of the signal at a higher fidelity. Frequency separation is particularly useful for images and other signal types because high frequencies tend to have small power , i.e., less energy/information (2, eq. 4.32). Therefore, this separation is in symbiosis with the "remove statistical dependencies" goal.

Among all possible image transforms, the ones considered in data compression are those that produce as many transformed coefficients as samples the original signal has (also known as non-expansive or critical transforms). Transforms that do expand the signal hinder compression ratios, while transforms that reduce the number of samples tend to discard information in a suboptimal way (2, §4.1.1).

Within non-expansive transforms, linear transforms (usually corresponding to a matrix or tensor multiplication) are the most common, due to their relative simplicity in design, implementation and execution. Orthogonal or biorthogonal transforms are arguably the most popular linear transforms, as these ensure that quantization errors in the transform domain do not explode in magnitude in the signal domain.

Transforms can be applied independently for different blocks (block transforms) or using a sliding window (subband transforms). Block transforms are generally faster than subband transforms, but they tend to provide worse decorrelation capabilities, and often the block structure can be seen in the reconstructed data. In this unit we touch on the discrete cosine transform (DCT) as an example of block transform (1, §13.4.1, §13.4.2, §13.3) and the discrete

wavelet transform as an example of subband transform (1, §14.1, §14.2, §14.3, §14.4, §14.8, §15.1, §15.2, §16)(2, §6.2). Color transforms (a type of spectral or point transforms) are a very common transform type in image compression.

Further reading and practice

- David Salomon's data compression book (55) contains several sections on transforms for data compression. Sections 4.4, 4.5, and 4.6 introduce transforms for image compression, and Chapter 5 provides a deep dive on the wavelet transform and its usage in data compression.
- Goyal's magazine article on the theoretical foundations of transform coding is an easy read on the topic (56).

Exercise 8.1 Let $a_1, \ldots, a_n \in \mathbb{Z}$ be a set of randomly-selected integers (for example, uniformly randomly selected from a fixed-sized interval). Consider the set of vectors $\mathcal{X} = \{(a_i, 2a_i - 1)\}$.

1. What is the entropy of each component of the vectors of $\mathcal{X}$? Is it higher for one component than the other?
2. Consider a transform $f : \mathcal{X} \to \mathbb{Z}^2$ defined as $f((a_i, 2a_i - 1)) = (a_i, 0)$. What is the entropy of each component of the transformed vectors? Is it different from the previous exercise? Is this transform good to encode $\mathcal{X}$?
3. Consider another transform $g : \mathcal{X} \to \mathbb{Z}^2$, now defined as $g((a_i, 2a_i - 1)) = (a_i \mod 4, 0)$. What is the entropy of each component of the transformed vectors now? Is this transform better than f to encode the vectors in $\mathcal{X}$?

Exercise 8.2 Let $x_1, \ldots, x_n \in [-16, 16]$ and $y_1, \ldots, y_n \in [-2, 2]$ be two sets of uniformly-distributed random real numbers, and let $\mathcal{X} = \{(x_i + y_i, y_i - x_i)\}$ be a set of vectors in $\mathbb{R}^2$. We define $\mathrm{Round}(x, y) = \{(\lfloor x \rceil, \lfloor y \rceil)\}$.

1. What is the entropy of each component of the vectors of $\mathrm{Round}(\mathcal{X})$? Is it higher for one component than the other?
2. Consider a transform $f : \mathcal{X} \to \mathbb{Z}^2$ defined as $f(x, y) = (\lfloor 2x \rceil, \lfloor y \rceil)$. What is the entropy of each component of the transformed vectors? Is it different from the previous exercise? Is this transform good to encode $\mathcal{X}$?
3. Consider another transform $g : \mathcal{X} \to \mathbb{Z}^2$, now defined as $g(x, y) = (\lfloor \frac{x}{16} \rceil, \lfloor \frac{y}{2} \rceil)$. What is the entropy of each component of the transformed vectors now? Is this transform better than f to encode the vectors in $\mathcal{X}$?
4. Find a transform that maps $\mathcal{X}$ to $\{(x_i, y_i)\}$. If we apply Round to this resulting data set and reverse the transform, what is the expected reconstruction error? How could you modify that transform to ensure the error from rounding is, at most, 1?

9. Machine Learning data compression

This unit makes a general introduction to Machnine Learning (ML) applied to data compression, and in particular the usage of autoencoders for lossy image compression.

9.1 Introduction to neural networks

This lecture series introducing neural networks is highly recommended to follow this section: https://youtu.be/aircAruvnKk?si=wLi-pbPuG3ZVliG2

A neural network is a parametric map, $f_\theta : \mathbb{R}^n \to \mathbb{R}^m$, whose parameters, $\theta \in \mathbb{R}^p$, are optimised to minimise a loss function, $L(\theta, x) : \mathbb{R}^p \times \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} \subset \mathbb{R}^n$ is a data set for which we optimise our network. To train a neural network is to find a set of parameters $\hat{\theta}$ that minimise the loss function across our data set. We call inference the application of $f_{\hat{\theta}}$ on real data, which in general is not in $\mathcal{X}$.

Typically, neural networks are structured into nodes, which are real numbers. These are either inputs, x, or are calculated from other nodes whose value is already known, based on the connections in the network. Typically, the operations described by these links are linear operations, with parameters from θ as weights, followed by a non-linear function referred to as an activation function. Nodes are typically arranged into layers which are computed in sequence, with the nodes in a given layer being all calculated out of the values of nodes in the previous layer.

Backpropagation is the recursive algorithm by which we can compute the gradient of a *differentiable* loss function with respect to the parameters (weights and bias) of the neural network. The gradient of the loss function is key for the training of the network under this algorithm, hence the requirement for it to be differentiable. Backpropagation is, in essence, an efficient way of computing the gradient of the loss function with respect to θ, so that a local minimum of that function may be found through stochastic gradient descent. The main mathematical concept behind backpropagation is the chain rule from calculus, which is recursively applied from the output layer (ground case) towards the input layer. As such, backpropagation iteratively updates the parameters in the neural network by:

1. Calculating the gradient of the loss function at a given data point and the current parameters, $\nabla L(\theta_i, x)$
2. Updating the parameters by adding that gradient, $\theta_{i+1} = \theta_i - \alpha \nabla L(\theta_i, x)$, where $\alpha > 0$ is the learning rate.

The kwy property of neural networks is that they can be used as universal approximators of any mapping: a *sufficiently large* neural network can be optimised to be arbitrarily close to any arbitrary mapping $\mathbb{R}^n \to \mathbb{R}^m$ (57, 58).

9.2 Autoencoders for image compression

An autoencoder is a type of neural network used to learn efficient codings of unlabeled data (unsupervised learning). An autoencoder learns two functions: an encoding transform that maps the input data, $x \in \mathbb{R}^n$, into a latent representation, $y \in \mathbb{R}^m$, and a decoding transform that recreates the input data from the latent representation, obtaining $\hat{x} \in \mathbb{R}^n$. The autoencoder learns an efficient representation (encoding) for a set of data, typically for dimensionality reduction, to generate lower-dimensional embeddings for subsequent use by other machine learning algorithms. This overall structure is optimised to minimise the *distortion*

between the original and reconstructed data, $D(x, \hat{x})$, for example under mean squared error (MSE).

For data compression, lower dimensionality alone is insufficient for efficient data compression, and entropy coding is applied to the latent representation. To that effect, the autoencoder may be also optimised to minimise the coding rate of the latent representation, $R(y)$. Observe that the entropy of y is not a differentiable function, so it cannot be directly used in the loss function. Instead, as proposed by Ballé *et al.*, a prior probability distribution is used, and the distribution of y is fit to that prior (59, 60). Given the prior probability mass function P, the expected codeword length for each individual value of y is $-\log(P(y_i))$, thus $R(y) = -\sum_i \log(P(y_i))$.

There are, therefore, two contradictory terms we wish to optimise our network for: rate and distortion. Clearly, minimum rate is achieved if we send no data (or constant data), while minimum distortion is achieved if the original data is transmitted as is, with no reduction. To accommodate this trade-off, a relaxation of the problem is used, setting the loss function to the Lagrangian $L(\theta, x) = R(y) + \lambda D(x, \hat{x})$, where λ is a constant regulating the trade-off between rate and distortion.

This rate-distortion minimisation scheme using autoencoders has been highly successful in image compression, outperforming a wide variety of conventional image compression techniques over the years (59, 61, 62, 63, 64, 65, 66, 67, 68, 69).

Further reading and practice

- Ballé's 2021 paper on Non-Linear Transform Coding provides a good general description of the transform coding paradigm using neural networks and what is theoretically being pursued (60).
- Yibo Yang's Introduction to Neural Data Compression provides a complete reference to the topic (70).

Exercise 9.1 Try the `TDS2526_Session_3.ipynb` notebook to acquaint yourself with tools such as Tensorflow and neural networks. You may run these in Google Colab (`http://colab.research.google.com`) or locally using Jupyter Notebooks:

```
# Install the environment using PIP
pip install notebook
# Run the environment. This should automatically open a window in your browser.
python -m notebook
```

Bibliography

(1) K. Sayood, *Introduction to data compression*. Elsevier, 2006. (Online). Available: https://bibcercador.uab.cat/permalink/34CSUC_UAB/1pvhgf7/alma991010367035106709

(2) D. S. Taubman and M. W. Marcellin, *JPEG2000: Image compression fundamentals, standards and practice*. Springer Science & Business Media, 2002. (Online). Available: https://bibcercador.uab.cat/permalink/34CSUC_UAB/1eqfv2p/alma991001464109706709

(3) C. McAnlis and A. Haecky, *Understanding compression: Data compression for modern developers*. O'Reilly Media, Inc., 2016. (Online). Available: https://bibcercador.uab.cat/permalink/34CSUC_UAB/1eqfv2p/alma991011182709906709

(4) C. Hock-Chua, "A tutorial on data representation integers, floating-point numbers, and characters," 2014, accessed: 2025-07-11. (Online). Available: https://www3.ntu.edu.sg/home/ehchua/programming/java/datarepresentation.html

(5) Joseph Collins-Unruh, "Understanding rasters," 2006, accessed: 2025-07-11. (Online). Available: https://cdn.safe.com/fmepedia/attachments/000001943/UnderstandingRasters.pdf

(6) Library of Congress, "Band Sequential (BSQ) Image File," 2021, accessed: 2025-07-11. (Online). Available: https://www.loc.gov/preservation/digital/formats/fdd/fdd000289.shtml

(7) ——, "Band Interleaved by Line (BIL) Image File Format," 2021, accessed: 2025-07-11. (Online). Available: https://www.loc.gov/preservation/digital/formats/fdd/fdd000304.shtml

(8) ——, "Band Interleaved by Pixel (BIP) Image File Format," 2021, accessed: 2025-07-11. (Online). Available: https://www.loc.gov/preservation/digital/formats/fdd/fdd000305.shtml

(9) W. S. Rasband, "Imagej," U.S. National Institutes of Health, Bethesda, MD, USA, 1997–2018. (Online). Available: https://imagej.net/

(10) R. M. Siegfried, "Notes on bytes and bytes," 2017, class notes, Adelphi University. (Online). Available: https://home.adelphi.edu/~siegfried/cs170/notes.html

(11) C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948. (Online). Available: https://dx.doi.org/10.1002/j.1538-7305.1948.tb01338.x

(12) J. V. Stone, "Information theory: A tutorial introduction," 2019. (Online). Available: https://arxiv.org/abs/1802.05968

(13) T. M. Cover, *Elements of information theory*. John Wiley & Sons, 1999. (Online). Available: https://bibcercador.uab.cat/permalink/34CSUC_UAB/avjcib/alma991006376339706709

(14) D. R. Stinson, *Cryptography: theory and practice*, 4th ed. Chapman and Hall/CRC, 2006. (Online). Available: https://doi.org/10.1201/9781315282497

(15) *Lossless Data Compression. Blue Book. Issue 3.*, Consultative Committee for Space Data Systems (CCSDS) Std. CCSDS 121.0-B-3, Aug. 2020. (Online). Available: https://ccsds.org/wp-content/uploads/gravity_forms/5-448e85c647331d9cbaf66c096458bdd5/2025/01//121x0b3.pdf

(16) S. Golomb, "Run-length encodings (corresp.)," *IEEE transactions on information theory*, vol. 12, no. 3, pp. 399–401, 1966. (Online). Available: https://doi.org/10.1109/TIT.1966.1053907

(17) J. Duda, "Asymmetric numeral systems," *CoRR*, vol. abs/0902.0271, 2009. (Online). Available: https://arxiv.org/abs/0902.0271

(18) Y. Collet and M. Kucherawy, "Zstandard Compression and the 'application/zstd' Media Type," RFC 8878, Feb. 2021. (Online). Available: https://doi.org/10.17487/RFC8878

(19) E. Bainville, "lzfse: Reference C implementation of LZFSE (Lempel–Ziv + Finite State Entropy)," 2016, commit tag "lzfse-1.0" (introduced June 7, 2016); BSD-3-Clause license. (Online). Available: https://github.com/lzfse/lzfse

(20) Joint Photographic Experts Group, "JPEG XL Image Coding System," 2021. (Online). Available: https://jpeg.org/jpegxl/index.html

(21) *Low-Complexity Lossless and Near-Lossless Multispectral and Hyperspectral Image Compression. Blue Book. Issue 2*, Consultative Committee for Space Data Systems (CCSDS) Std. CCSDS 123.0-B-2, Feb. 2019. (Online). Available: https://ccsds.org/wp-content/uploads/gravity_forms/5-448e85c647331d9cbaf66c096458bdd5/2025/01//123x0b2e2c3.pdf

(22) J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337–343, 1977. (Online). Available: https://doi.org/10.1109/TIT.1977.1055714

(23) F. Auli-Llinas, "General embedded quantization for wavelet-based lossy image coding," *IEEE Transactions on Signal Processing*, vol. 61, no. 6, pp. 1561–1574, 2013. (Online). Available: https://doi.org/10.1109/TSP.2012.2236323

(24) M. Hernández-Cabronero, I. Blanes, A. J. Pinho, M. W. Marcellin, and J. Serra-Sagristà, "Analysis-driven lossy compression of dna microarray images," *IEEE Transactions on Medical Imaging*, vol. 35, no. 2, pp. 654–664, February 2016. (Online). Available: https://dx.doi.org/10.1109/TMI.2015.2489262

(25) F. A. Kruse, A. B. Lefkoff, J. W. Boardman, K. B. Heidebrecht, A. Shapiro, P. Barloon, and A. F. Goetz, "The spectral image processing system (sips)—interactive visualization and analysis of imaging spectrometer data," *Remote sensing of environment*, vol. 44, no. 2-3, pp. 145–163, 1993. (Online). Available: https://doi.org/10.1016/0034-4257(93)90013-N

(26) Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures," *IEEE Signal Processing Magazine*, vol. 26, no. 1, pp. 98–117, 2009. (Online). Available: https:/dx.doi.org/10.1109/MSP.2008.930649

(27) Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004. (Online). Available: https://doi.org/10.1109/TIP.2003.819861

(28) Z. Wang, E. Simoncelli, and A. Bovik, "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conference on Signals, Systems and Computers, 2003*, vol. 2, 2003, pp. 1398–1402. (Online). Available: https://doi.org/10.1109/ACSSC.2003.1292216

(29) R. K. Mantiuk, D. Hammou, and P. Hanji, "HDR-VDP-3: A multi-metric for predicting image differences, quality and contrast distortions in high dynamic range and regular content," *arXiv preprint arXiv:2304.13625*, 2023. (Online). Available: https://doi.org/10.48550/arXiv.2304.13625

(30) F. Auli-Llinas, C. de Cea-Domínguez, and M. Hernandez-Cabronero, "Accelerating bpc-paco through visually lossless techniques," *Journal of Visual Communication and Image Representation*, vol. 89, p. 103672, 2022. (Online). Available: https://doi.org/10.1016/j.jvcir.2022.103672

(31) O. Maireles-Gonzalez, J. Bartrina-Rapesta, M. Hernandez-Cabronero, and J. Serra-Sagrista, "Efficient lossless compression of integer astronomical data,"

Publications of the Astronomical Society of the Pacific, vol. 135, no. 1051, p. 094502, 2023. (Online). Available: https://doi.org/10.1088/1538-3873/acf6e0

(32) J. Bartrina-Rapesta, M. Hernandez-Cabronero, and J. Serra-Sagrista, "Reducing data dependencies in the feedback loop of the ccsds 123.0-b-2 predictor," *IEEE Transactions on Geoscience and Remote Sensing*, 2022. (Online). Available: https://ddd.uab.cat/pub/artpub/2022/275899/IEEE_TGRS_IUMA_UAB_Review.pdf

(33) K. Chow, D. E. O. Tzamarias, M. Hernandez-Cabronero, I. Blanes, and J. Serra-Sagrista, "Performance improvement on k^2-raster compact data structure for hyperspectral scenes," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2021. (Online). Available: https://doi.org/10.1109/LGRS.2021.3084065

(34) D. E. O. Tzamarias, K. Chow, I. Blanes, and J. Serra-Sagrista, "Fast run-length compression of point cloud geometry," *IEEE Transactions on Image Processing*, vol. 31, pp. 4490–4501, 2022. (Online). Available: https://doi.org/10.1109/TIP.2022.3185541

(35) S. M. i Verdu, J. Ballé, V. Laparra, J. Bartrina-Rapesta, M. Hernandez-Cabronero, and J. Serra-Sagristà, "A scalable reduced-complexity compression of hyperspectral remote sensing images using deep learning," *Remote Sensing*, vol. 15, no. 18, p. 4422, 2023. (Online). Available: https://doi.org/10.3390/rs15184422

(36) X. Fan, Y. Chen, and P. Frossard, "Probability models for highly parallel image coding architecture," *Image Communication*, vol. 107, p. 116914, 2022. (Online). Available: https://doi.org/10.1016/j.image.2022.116914

(37) C. Veller, P. B. Budge, S. P. Burns, F. Faghri, F. Verdier, and F. Tufvesson, "Redundancy and optimization of tans entropy encoders," *IEEE Transactions on Multimedia*, vol. 23, pp. 2427–2440, 2021. (Online). Available: https://doi.org/10.1109/TMM.2020.3040547

(38) J. Bartrina-Rapesta, M. Hernandez-Cabronero, and J. Serra-Sagrista, "The ccsds 123.0-b-2 "low-complexity lossless and near-lossless multispectral and hyperspectral image compression" standard: A comprehensive review," *IEEE Geoscience and Remote Sensing Magazine*, vol. 9, no. 3, pp. 20–34, 2021. (Online). Available: https://doi.org/10.1109/MGRS.2020.3048443

(39) G. Martin, J. Serra-Sagristà, and M. Hernandez-Cabronero, "An efficient algorithm for segmenting quasi-periodic digital signals into pseudo cycles: Application in lossy audio compression," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 1414–1426, 2022. (Online). Available: https://doi.org/10.1109/TASLP.2022.3171969

(40) N. Zeghidour, A. V. D. Oord, Y. Adi, S. Dieleman, A. Skerry-Ryan, D. Sotelo, A. Vinyals, and O. Vinyals, "Soundstream: An end-to-end neural audio codec," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 171–186, 2022. (Online). Available: https://doi.org/10.1109/TASLP.2021.3129994

(41) D. Marpe and T. Wiegand, "An efficient low complexity encoder for mpeg advanced audio coding," in *Proceedings of the International Conference on Advanced Communication Technology (ICACT)*, 2006, pp. 1313–1317. (Online). Available: https://doi.org/10.1109/ICACT.2006.206269

(42) S. Kim, T. Kim, and Y. Kwon, "A novel scheme for svac audio encoder," in *Proceedings of the International Symposium on Communications and Information Technologies (ISCIT)*, 2014, pp. 122–126. (Online). Available: https://doi.org/10.1109/ISCIT.2014.7011970

(43) Y. Li, X. Wang, B. Liu, Z. Wei, and K. Wu, "Compression of multiple dna sequences using intra-sequence and inter-sequence similarities," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 12, no. 5, pp. 1029–1039, 2015. (Online). Available: https://doi.org/10.1109/TCBB.2015.2403370

(44) B. Liu, Q. Huang, J. Huang, and K. Wu, "Lossy compression of quality values in sequencing data," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 16, no. 3, pp. 866–877, 2019. (Online). Available: https://doi.org/10.1109/TCBB.2019.2959273

(45) Q. Huang, B. Liu, and K. Wu, "Clustering-based compression for population dna sequences," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 14, no. 5, pp. 1123–1135, 2017. (Online). Available: https://doi.org/10.1109/TCBB.2017.2762302

(46) M. F. de Moura, D. L. de Faria, L. de Almeida, and R. B. de Queiroz, "Wavelet compression of ecg signals by jpeg2000," in *Proceedings of the Data Compression Conference (DCC)*, 2004, p. 385. (Online). Available: https://doi.org/10.1109/DCC.2004.1281503

(47) X. Mao, X. Wang, and Y. Yan, "Ecg compression using compressed sensing with lempel-ziv-welch technique," in *Proceedings of the National Graduate Conference on Telecommunications (NGCT)*, 2015, pp. 1–6. (Online). Available: https://doi.org/10.1109/NGCT.2015.7375242

(48) J. Rodríguez-Fernández, M. Fernández-Moral, M. Fernández-Cardador, J. A. Gómez-González, P. Sánchez-Sánchez, and A. Ríos-Navarro, "Rate-distortion and complexity comparison of hevc and vvc video encoders," *IEEE Latin America Transactions*, vol. 18, pp. 1412–1424, 2020. (Online). Available: https://doi.org/10.1109/LASCAS45839.2020.9069036

(49) Y. Li, Y. Sun, Z. He, Q. Shen, and S. Shan, "Transform coding in the vvc standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 11, pp. 4152–4164, 2021. (Online). Available: https://doi.org/10.1109/TCSVT.2021.3087706

(50) J. Cleary and I. Witten, "Data Compression Using Adaptive Coding and Partial String Matching," *IEEE Transactions on Communications*, vol. 32, no. 4, pp. 396–402, 1984. (Online). Available: https://doi.org/10.1109/TCOM.1984.1096090

(51) M. V. Mahoney, "The paq1 data compression program," Florida Institute of Technology, Computer Science Department, Technical Report (draft), January 2002. (Online). Available: https://mattmahoney.net/dc/paq1.pdf

(52) X. Wu and N. Memon, "CALIC-a context based adaptive lossless image codec," in *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 4, 1996, pp. 1890–1893. (Online). Available: https://doi.org/10.1109/ICASSP.1996.544819

(53) International Telecommunication Union, *Information technology – Lossless and near-lossless compression of continuous-tone still images (JPEG-LS)*, International Telecommunication Union Standard ITU-T Recommendation T.87, 1998. (Online). Available: https://www.itu.int/rec/T-REC-T.87

(54) D. Salomon and G. Motta, *Handbook of data compression*. Springer Science and Business Media, 2010. (Online). Available: https://bibcercador.uab.cat/permalink/34CSUC_UAB/1eqfv2p/alma991010484105706709

(55) D. Salomon, *Data Compression: The Complete Reference*. Berlin, Heidelberg: Springer-Verlag, 2006. (Online). Available: https://bibcercador.uab.cat/permalink/34CSUC_UAB/avjcib/alma991011183977706709

(56) V. K. Goyal, "Theoretical foundations of transform coding," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 9–21, Sept. 2001. (Online). Available: https://doi.org/10.1109/79.952802

(57) K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991. (Online). Available: https://doi.org/10.1016/0893-6080(91)90009-T

(58) P. Kidger and T. Lyons, "Universal approximation with deep narrow networks," in *Conference on learning theory*. PMLR, 2020, pp. 2306–2327. (Online). Available: https://arxiv.org/abs/1905.08539

(59) J. Ballé, V. Laparra, and E. Simoncelli, "End-to-End Optimised Image Compression," *International Conference on Learning Representations (ICLR)*, 2017. (Online). Available: https://arxiv.org/abs/1611.01704

(60) J. Ballé, P. A. Chou, D. Minnen, S. Singh, N. Johnston, E. Agustsson, S. J. Hwang, and G. Toderici, "Nonlinear Transform Coding," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 2, pp. 339–353, 2021. (Online). Available: https://arxiv.org/abs/2007.03034

(61) J. Ballé, D. C. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational Image Compression with a Scale Hyperprior," *International Conference on Learning Representations (ICLR)*, 2018. (Online). Available: https://arxiv.org/abs/1802.01436

(62) D. C. Minnen, J. Ballé, and G. Toderici, "Joint Autoregressive and Hierarchical Priors for Learned Image Compression," in *NeurIPS*, 2018. (Online). Available: https://arxiv.org/abs/1809.02736

(63) Y. Choi, M. El-Khamy, and J. Lee, "Variable Rate Deep Image Compression With a Conditional Autoencoder," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10 2019, pp. 3146–3154. (Online). Available: https://doi.org/10.1109/ICCV.2019.00324

(64) Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Deep Residual Learning for Image Compression," *Conference on Computer Vision and Pattern Recognition (CVPR) Workshop*, 2019. (Online). Available: https://arxiv.org/abs/1906.09731

(65) F. Yang, L. Herranz, J. v. d. Weijer, J. A. I. Guitián, A. M. López, and M. G. Mozerov, "Variable Rate Deep Image Compression With Modulated Autoencoder," *IEEE Signal Processing Letters*, vol. 27, pp. 331–335, 2020. (Online). Available: https://doi.org/10.1109/LSP.2020.2970539

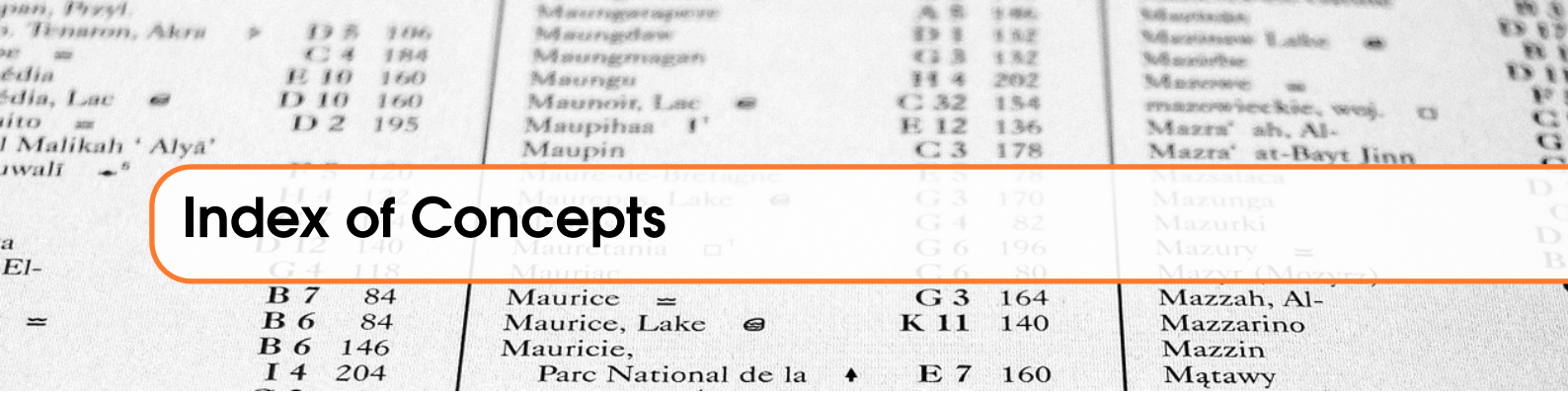
(66) Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Learned Image Compression With Discretized Gaussian Mixture Likelihoods and Attention Modules," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 7936–7945. (Online). Available: https://arxiv.org/abs/2001.01568

(67) F. Yang, L. Herranz, Y. Cheng, and M. G. Mozerov, "Slimmable Compressive Autoencoders for Practical Neural Image Compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 4998–5007. (Online). Available: https://arxiv.org/abs/2103.15726

(68) D. He, Z. Yang, W. Peng, R. Ma, H. Qin, and Y. Wang, "ELIC: Efficient Learned Image Compression with Unevenly Grouped Space-Channel Contextual Adaptive Coding," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5708–5717, 2022. (Online). Available: https://api.semanticscholar.org/CorpusID:247594672

(69) H. Fu, F. Liang, J. Liang, Y. Wang, Z. Fang, G. Zhang, and J. Han, "Fast and High-Performance Learned Image Compression With Improved Checkerboard Context Model, Deformable Residual Module, and Knowledge Distillation," *IEEE Transactions on Image Processing*, vol. 33, pp. 4702–4715, 2024. (Online). Available: https://dx.doi.org/10.1109/TIP.2024.3445737

(70) Y. Yang, S. Mandt, and L. Theis, *An introduction to neural data compression*. Foundations and Trends in Computer Graphics and Vision, 2023, vol. 15, no. 2. (Online). Available: https://theis.io/publications/0600000107.pdf



Index of Concepts

activation function, 24
adaptive, 11
alphabet, 7, 13
arithmetic coding, 10, 11
Asymmetric Numeral Systems, 11
audio, 5
autoencoder, 24
average, 16

backpropagation, 24
band, 5, 16, 19
bandwidth, 7
big endian, 6
BIL, 5
binary entropy coder, 14
BIP, 5
bit, 5, 7
bitdepth, 5
bitplane, 14
bits per base, 16
bits per pixel, 16
bits per pixel per component, 16
bits per sample, 16
block transforms, 22
BSQ, 5
byte, 5

CABAC, 11
causal context, 19
channel, 7
codeword, 10
codeword length, 10
color image, 16
color transform, 23
compact, 10
component, 5
compressed data rate, 16
compressibility, 7
compression, 6
compression ratio, 16
compression time, 16
computational complexity, 11
computer security, 7
connections, 24
context, 8
context mixing, 20
corpus, 17
correlation, 19
cross entropy, 10
cryptographical keys, 8
cryptography, 7

data, 5, 7
data compression, 7
data type, 6
deadzone quantization, 14, 19
deadzone uniform scalar quantizer, 14
decoding transform, 24
dequantization, 13
discrete cosine transform (DCT), 19
discrete wavelet transform (DWT), 19
distortion metric, 17
dynamic range, 5, 20

efficiency, 7, 11, 16
embedded quantization, 14
emitter, 7
encoding transform, 24
entropy, 19
error correcting code, 7
error power, 17
exactness, 7
expected length, 10

fidelity, 17
Finite State Entropy, 11
first-order entropy, 7
fixed-length, 5
floating point, 5

geometry, 6
Golomb-Rice, 11
grammar, 20
grayscale, 5

header, 6
Huffman's algorithm, 10, 11
hyperspectral, 5

independence, 7, 8, 19
inference, 24
information, 7
instantaneous decoding, 10

integer, 5
intensity, 5
invertible, 10

JPEG XL, 11

Kullback-Leibler, 10

Laplacian distribution, 19
latent representation, 24
layers, 24
learning rate, 24
linear, 20
little endian, 6
Lloyd-Max algorithm, 14
logarithm, 7
lossless compression, 7, 11, 16
lossy compression, 7, 11, 13, 20
LSB, 5
LZ77, 11
LZ78, 11

maximum entropy, 8
mean squared error, 17, 20
median, 16
memory usage, 16
message, 10
metainformation, 6
midrise quantizer, 14
midtread quantizer, 14
minimum, 16
minimum entropy, 8
monochrome, 5
MQ, 11
MSB, 5
multiscale structural similarity, 17
multispectral, 5, 16

neural network, 24
nodes, 24
non-expansive, 22
non-uniform quantization, 14

objective distortion metric, 17
Orthogonal, 22
overhead, 10

PAQ, 11
peak absolute error, 17
peak signal-to-noise ratio, 17
perceptual coding, 17
physical, 13
pipeline, 11
pixel, 5, 16
prediction, 19
prediction residual, 19
prefix code, 10
privacy/authenticity, 7
probability distribution, 7, 13
progressive transmission, 14
psychoacoustic, 17
psychovisual, 17

qstep, 13
quantization, 13, 20
quantization index, 13
quantization step, 13

range coding, 11
raster, 5
rate-distortion, 13
rate-distortion analysis, 17
raw, 5
receiver, 7
reconstruction point, 13
redundancy, 19
relative entropy, 10
remote sensing, 16
representation, 10
RGB, 5
Run-Length Encoding, 11

samples, 5
scalar, 13
self-information, 7
serialized, 5
side information, 6
signed, 6
source, 7
spectral angle, 17
stationary, 7
structural similarity, 17
subband transforms, 22
subjective distortion metric, 17
surprisal, 7

throughput, 16
train, 24
transform domain, 22
transform stage, 22
transmission, 7
two-pass, 11

uniform distribution, 8
uniform scalar quantization, 13
unique decoding, 10
unsigned, 6

variable-length code, 10
video, 5

wavelengths, 5

zip, 11
Zstandard, 11