University of Beira Interior

Departament of Informatics

Nº 2024: *Study and evaluation of the HiDNA codec*

Author:

Artur Alexandre Putyato

Mentor:

Professor Doctor Maria Manuela Pereira de Sousa

July 2, 2024

Acknowledgements

I would like to express my deepest gratitude to my mentor, Professor Doctor Maria Manuela de Sousa, for her invaluable guidance and support throughout this journey.

I am also deeply thankful to my colleagues at university for their camaraderie and support, especially Tiago Silva, Miguel Marques, Gabriel Lázaro, David Quadrado, and Alexandre Cunha. Their friendship and collaboration have made this experience enriching and enjoyable.

To my family, whose boundless love, unwavering encouragement, and profound understanding have been my constant source of strength and inspiration, I am profoundly thankful.

I also want to honor the memory of my late uncle, Oleg. Though he is no longer with us, his wisdom and kindness continue to inspire me.

Thank you.

Contents

List of Figures

List of Tables

Acronyms

AC	Alternating Current term
CNR	Coordinate-based Neural Representation
COIN	Compression with Implicit Neural Representations
DC	Direct Current term
DCT	Discrete Cosine Transform
DPCM	Differential Pulse Code Modulation
DNA	Deoxyribonucleic Acid
FSIM	Feature Similarity
IW-SSIM	Information Content Weighted SSIM
JPEG	Joint Photographic Experts Group
MDC	Multiple Description Coding
MLP	Multi-layer Perceptron
MSE	Mean Squared Error
MS-SSIM	Multi-Scale SSIM
NLPD	Normalised Laplacian Pyramid Distance
PCR	Polymerase Chain Reaction
PSNR	Peak Signal-to-Noise Ratio
PSNR-HVS-M	PSNR-Human Visual System
RLC	Run-Length Coding
SARS-CoV-2	Severe Acute Respiratory Syndrome Coronavirus-2
SSIM	Structural Similarity Index
VIF	Visual Information Fidelity
VMAF	Video Multimethod Assessment Fusion
ReLU	Rectified Linear Unit

Chapter

1

Introduction

1.1 Context

The current world demands a high consumption of information due to the advancement of the digital economy and corporations in this sector. Each year, with the increasing number of users and their volume of consumption, we witness an exponential growth in information generated globally. According to Statista, in 2024, a volume of information created, captured, copied, and consumed of 147 zettabytes is projected, representing a 22.5% increase compared to 2023. This value equates to about 400 exabytes per day - approximately 330 billion pictures.

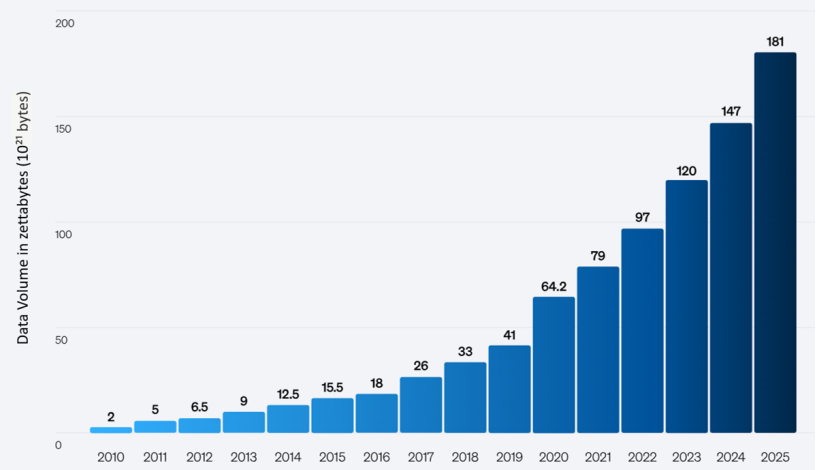

Figure 1.1: Volume of data consumed per year (Taylor and Petroc [2023])

To store the necessary amount of information, it is imperative to contin-

ually improve storage methods. However, the current rate of improvement is only 20% per year, while consumption increases by about 25% per year. Unfortunately, this gap between storage capacity and production is not a problem that can be easily resolved. The existing approach calls for the expensive and resource-intensive building of additional data centres.

In response, this report explores an innovative substitute: DNA storage. We will explore the mechanics of the HiDNA codec, an original solution that promises to revolutionise how we store vast amounts of information efficiently and sustainably.

1.2 General Objective

The main objective of this work is to study and evaluate the HiDNA codec, focusing on its technical aspects, performance metrics, and potential applications. The HiDNA codec represents a significant advancement in image compression using artificial intelligence algorithms, with the goal of storing the compressed image in Deoxyribonucleic Acid (DNA).

1.3 Document Organisation

To reflect the work that has been done, this document is structured as follows:

1. The first chapter – **Introduction** – presents the project, the motivation for its choice, the background, the objectives, and the organization of the document.

2. The second chapter – **Concepts** – will introduce the necessary concepts to understand the following chapters.

3. The third chapter – **State of Art** – discusses the current state of research on DNA coding, and relevant studies.

4. The fourth chapter – **HiDNA Codec** – explores the step-by-step methods of how the HiDNA codec itself works and the details the most relevant studies on the subject.

5. The fifth chapter – **Evaluation of the HiDNA Codec** – evaluates the results obtained from the execution of the codec and compare them to the anchors' results.

6. The sixth chapter – **General conclusions and Future Work** – summarises the findings, reflects on the work done, and discusses potential future work.

Chapter

2

Concepts

2.1 Introduction

In this chapter, we will explore the necessary methods and concepts that are important to the State-of-Art (chapter 3) and used by the HiDNA codec that will be explained in the chapter 4.

Firstly, biological concepts will be presented as they are important to have full understanding of the codecs that will be presented since their general objective is to translate data to DNA sequences (see section 2.2).

The first methods of Image DNA coding were based on the Joint Photographic Experts Group (JPEG) standard (see section 2.3) while the most recent one is developed in the context of Multiple Description Coding (MDC) (see section 2.4).

In recent years, following the tendency to incorporate deep learning in image coding, the area of DNA Image coding is also not exempt from this trend. The required deep learning concepts will be explained in the section 2.5.

The codec which is subject of study in this report uses bicubic interpolation method which is a popular upscaling method of images to avoid aggressive visual quality distortions (see section 2.6).

2.2 Biological Concepts

Nowadays, it became very important to find efficient ways to store large quantity of data. An interesting approach involves the use of DNA.

DNA is a molecule formed by two intertwined strands, that contain sugar molecules called deoxyribose and phosphate groups. In addition, four nitrogenous bases are associated with these sugar molecules: adenine (A), cyto-

sine (C), guanine (G) and thymine (T). A unique feature of DNA is the pairing of these bases in two chains: adenine with thymine and cytosine with guanine; this ensures the stability and integrity of the molecule.

Given the explosive growth of digital data and the limitations of traditional storage methods, exploring alternatives such as DNA storage has become increasingly important (Dey et al. [2022]) . This chapter will explore how DNA storage works and its potential as a solution to the increasing data storage requirements specially of images, therefore there will be a section explaining the JPEG Standard, as it is currently the most popular and widely used image encoding method.

2.2.1 Synthesis Methods of DNA

DNA (oligonucleotide) synthesis is an essential process for several laboratory applications. There are basically two main ways to do it: one is adding one building block at a time, kind of like building with Lego bricks - base-by-base synthesis, and the other is joining together shorter, already-made pieces - ligation synthesis. Although base-by-base synthesis is the most popularly used, it has limitations in strand length, in contrast to ligation methods that offer potential solutions to overcome these limitations.

2.2.1.1 Phosphoramidite method

Phosphoramidite method was developed by Beaucage and Caruthers in 1981 (Sandahl et al. [2021]) and it still remains one of the most popular methods in DNA synthesis. It's basically a step-by-step process where you add special building blocks called phosphoramidites onto a solid support matrix, like Polystyrene or Controlled Pore Glass.

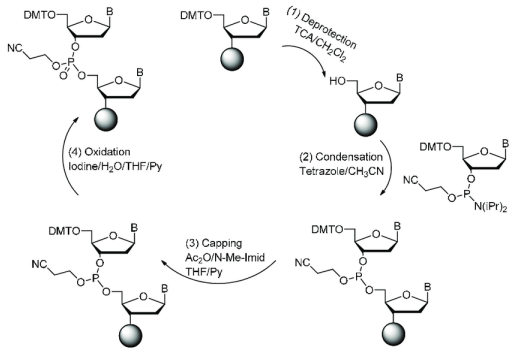

Figure 2.1: Phosphoramidite Method (Ni et al. [2017])

In this process, each cycle entails distinct chemical steps:

1. **Deprotection**: The removal of a protective Dimethoxytrityl (DMT) group from the nucleotide precursor, to allow further additions. The DMT group shields the hydroxyl group on the 5' carbon of the sugar molecule, from undergoing reactions prematurely. Once the DMT group is removed, the exposed hydroxyl group can participate in subsequent reactions.

2. **Coupling**: The attachment of a nucleotide to the growing DNA chain, activated by an appropriate reagent.

3. **Capping**: The prevention of unintended strand elongation by blocking any unreacted sites.

4. **Oxidation**: Strengthening the sugar-phosphate backbone to stabilize the newly formed DNA strand.

Repeated cycles of these reactions gradually will build the desired DNA sequence. Once synthesized, the DNA strands are separated from the solid support for further use.

This method, known for its precision and scalability and it's easier to automate in laboratory environment. Additionally, emerging techniques such as micro-array synthesis offer increased throughput and cost efficiency, further advancing DNA synthesis technologies.

2.2.1.2 Polymerase Chain Reaction

The Polymerase Chain Reaction (PCR) is a proccess based on the activity of DNA polymerases (Khehra et al. [2023]), which are enzymes responsible for generating DNA.

In PCR, DNA polymerase enzymes function as conductors, as they are the ones that will "slip" complementary nucleotides into the template, that is, these enzymes lengthen the primers by sequentially attaching complementary nucleotides to the template strands, with the primers outlining the initiation points for the DNA replication (YourGenome.org [2024]).

The advantage of this kind of methods is that they operate under soft conditions, which reduces the risk of DNA damage and avoids the need for complex chemical safety precautions and complex purification protocols. However, these methods have certain limitations. Challenges can arise, especially when dealing with long DNA sequences (200nt - 300nt) or when accurately incorporating modified nucleotides.

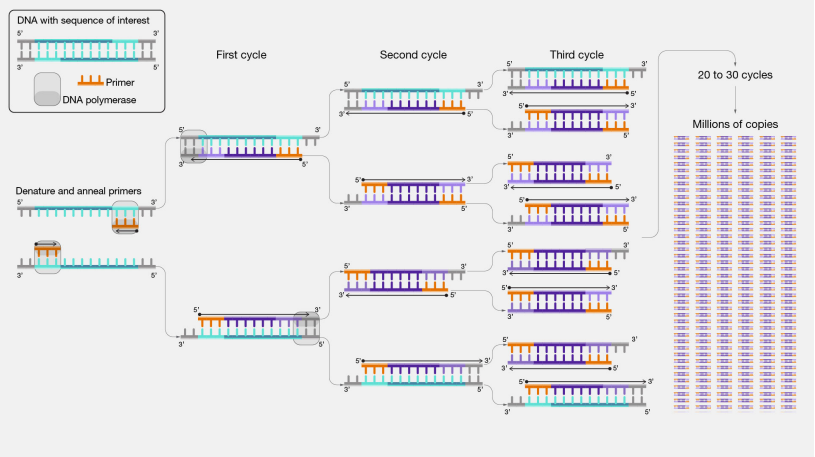

Figure 2.2: PCR Process (Smith [2024])

Despite these restrictions, enzymatic DNA replication remains indispensable in the areas of molecular biology, diagnostics and biotechnology due to its simplicity, efficiency and compatibility with aqueous environments.

2.2.1.3 Synthesis by ligation

Synthesis ligation is a method of DNA synthesis that uses enzymes called DNA ligases to figure out the order of nucleotides in a DNA strand without needing DNA polymerase (Sciences [2024]).

Here's how it works: DNA ligase enzymes join the ends of DNA molecules together. To do this, they rely on short sequences of nucleotides called "anchor" strands. These anchors are attached to known parts of the DNA we want to analyse. After adding fluorescent labeled probes (small DNA pieces about eight to nine bases long) to the mix, these probes stick to the target DNA next to the anchor sequence. When the probes find their matching spots on the DNA, the DNA ligase comes into action. It joins the DNA pieces together only where they match up.

One of the most important things to consider is sequencing orientation: are we reading DNA from the 5' end to the 3' end or vice versa (3' to 5'). While both directions work, going from 3' to 5' is usually better because it is more efficient for completing multiple rounds of connection. However, this method may have difficulty analysing palindromic sequences, in which the DNA reads the same way forward and backward.

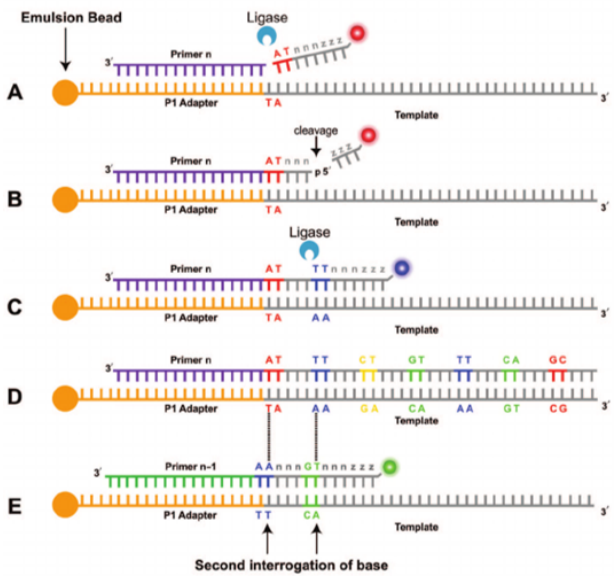

Figure 2.3: Synthesis by ligation (Voelkerding et al. [2009])

2.2.2 Sequencing methods

Determining a DNA molecule's nucleotide sequence is known as DNA sequencing. It's similar to reading an organism's genetic instruction handbook.

The first machine to automatically sequence DNA was created in 1986 as a result of research conducted in the 1970s by scientists such as Sanger, who began inventing techniques to read DNA sequences. After almost 40 years of use, the Sanger sequencing technology was recently applied to sequence the Severe Acute Respiratory Syndrome Coronavirus-2 (SARS-CoV-2) component that is essential to the virus's ability to infect cells. The accuracy and speed of DNA sequencing have increased with time due to developments in computers and nanotechnology.

Although other sequencing techniques have been established, for the purpose of this work, we will concentrate on two crucial ones: sequencing by synthesis and nanopore sequencing.

2.2.2.1 Illumina Dye Sequencing

Illumina sequencing is recognised as one of the most accurate high-throughput technology, it uses a sequencing by synthesis approach to identify the nucleotide sequence of DNA. It's derived from Sanger sequencing with some modifications that allow massive parallelisation. The process follows these steps:

1. **Library Preparation**: DNA samples are fragmented into smaller pieces (usually between 200 and 500 nucleotides). Then, special adapters are ligated to the ends of these fragment, these adapters are critical for later steps, allowing the DNA to bind to the flow cell and be amplified. After adapter ligation, the fragments are amplified via PCR to create multiple copies, and then they are purified, resulting in a sequencing library ready for the next step.

2. **Cluster Generation**: The sequencing library is attached to a flow cell. The flow cell is a glass slide with multiple lanes, each coated with oligos that are complementary to the adapters in the sequencing library. The attached DNA strands undergo "bridge PCR," creating clonal clusters. Bridge PCR involves the attached strands looping over to bind to other oligos on the flow cell, then being amplified to form dense clusters of identical DNA strands.

3. **Sequencing**: Firstly, specific connector primers are used to initiate the synthesis process at specific points on the DNA template. As the DNA polymerase moves along the template strand, it incorporates complementary nucleotides into the growing strand (T-A, C-G), if the template nucleotide contains Adenine, the complementary nucleotide (that contains thymine) labeled with the fluorescent marker corresponding to Adenine will be incorporated into the new DNA strand. The four different nucleotides are labeled with distinct fluorescent dyes so they can be distinguised by the the different radiation emitted and recorded.

The most common errors in Illumina-sequenced data are substitutions that can occur when one base is mistakenly identified as another, often due to similarities in fluorescence between certain base pairs.

2.2.2.2 Nanopore sequencing

Nanopore sequencing is a significant advancement in DNA sequencing technology because it allows long DNA fragments to be sequenced and relatively cheaper and faster than Illumina. The core principle behind nanopore sequencing involves the identification of DNA bases by measuring changes in electrical conductivity as DNA strands pass through a biological pore that are usually based on bacteria protein channels that can be found in the outer membrane ("shell").

This process can be broken down into three main steps:

1. **Sample Preparation**: DNA sequences are prepared by attaching adapters to their ends, which guide the sequences through the nanopores.

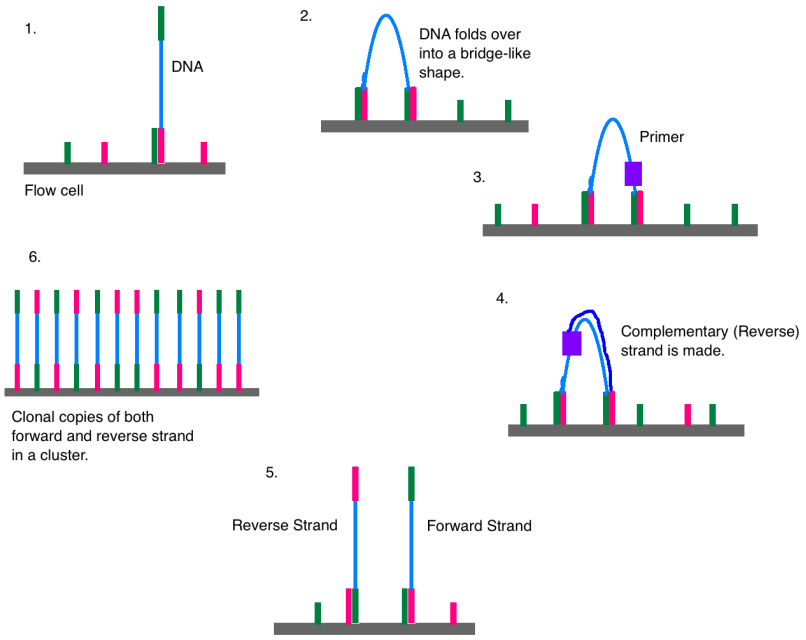

Figure 2.4: Illumina Process (DMLapato [2015])

2. **Signal Measurement**: The DNA sequences are immersed in an electrolytic solution, that means it contains ions which are atoms (or molecules) with electrical charge, and a constant voltage is applied at the nanopore generating an ionic current.

   When a single-stranded DNA molecule passes through the nanopore, it temporally disrupt the ionic current, these disruptions are detected by sensors positioned around the nanopore that continuously monitor these changes at constant sampling frequency, generating a series of electrial signals.

3. **Basecalling**: After the previous step we end up with these electrical signals that hold the information about the DNA sequence inside. It's used machine learning algorithms to translate these electrical signals into meaningful DNA sequences by analysis of the electrical signal data and decode it into sequences of DNA bases. This process involves using reference databases and k-mer tables, which essentially map specific patterns in the electrical signals to corresponding sequences of nucleotides. By sequentially interpreting these patterns, the algorithms reconstruct the original DNA sequence from the electrical signals.

Oxford Nanopore Technology provides a range of sequencing devices based

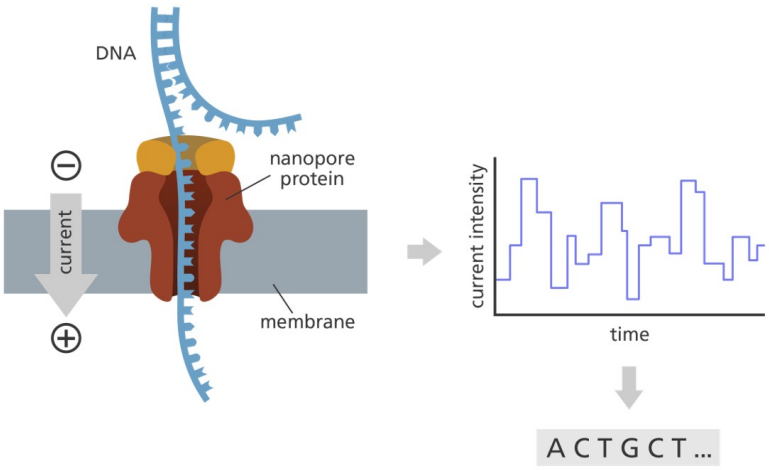

Figure 2.5: Nanopore Sequencing (yourgenome)

on this sequencing method, including MinION and PromethION. Both devices utilize nanopore technology, but PromethION is designed for high-volume applications, ideal for large-scale projects requiring extensive sequencing, while MinION is better suited for smaller-scale or fieldwork scenarios where portability and cost-efficiency matters most.

2.2.3 Biological Constraints of DNA Coding

DNA Coding is an innovative approach that bridges digital data storage with the molecular world of DNA, it involves the translation of digital data, such as text or images, into the DNA molecules. This translation is achieved through DNA synthesis when encoding data into DNA, and DNA sequencing is required to retrieve the information. However, these methods are prone to errors. Therefore, to optimize DNA coding, it is crucial to establish constraints that mitigate these errors and enhance reliability.

- **Strand length limitations** - In DNA data storage, when synthesizing DNA strands for encoding information, longer strands can increase the likelihood of errors during synthesis. The longer the strand, the higher the probability of errors such as misincorporation of nucleotides or incomplete synthesis. By imposing a limit on the length of each DNA strand, typically around 200 nucleotides per strand, it becomes more manageable to control the synthesis process and reduce the occurrence of errors.

- **Homopolymers** - Homopolymers are sequences of nucleotides that are repeated consecutively within a DNA strand. While short they are generally well-tolerated, longer homopolymers can pose significant challenges. This is because sequencing technologies may struggle to accurately resolve long stretches of identical nucleotides, leading to errors in the determination of the DNA sequence. Additionally, during DNA synthesis or replication, polymerase enzymes may encounter difficulties or errors when copying long homopolymer regions, further contributing to potential inaccuracies in the DNA sequence.

- **CG content balance** - Maintaining an appropriate balance of CG content (cytosine, guanine) is important for the stability and functionality of the DNA molecule. Extreme deviations from the optimal CG content range of 40% to 50% can affect the stability of DNA duplexes and may interfere with processes such as DNA replication and transcription.

- **Repetition of patterns** - Repetitive patterns within DNA sequences, consisting of short sequences of 3 to 5 nucleotides repeated multiple times, can pose challenges during data storage and analysis. It can complicate sequencing and analysis processes, as repetitive sequences may be prone to errors or misinterpretation. By limiting the repetition of such patterns to a maximum of 4 times helps to mitigate potential errors and ensures the accuracy of the stored DNA data.

2.3 JPEG Standard

JPEG is a popular method used to compress digital images, it is especially effective for photographs and images with smooth colour transitions. It's a lossy compression algorithm which means that some information is lost during compression, but those variations are hard to notice to the human eye (the consumer of the image).

1. **Colour Space Conversion** - The image is converted from RGB colour space to Y-Cb-Cr. Y represents luminance (brightness), and Cb-Cr represents the chrominance (colour, Cb - blue, Cr - red).

2. **Downsampling** - This step allows more efficient compression because we can lose some chrominance information, because the human eye is less sensitive to it. The most common downsampling ratios in JPEG compression are 4:2:0 and 4:2:2:

- 4:2:0 ratio: For every 4 luminance samples, there are 2 chrominance samples horizontally and vertically. This means that every 2x2 block of pixels in the chrominance planes is averaged into a single value, resulting in a reduction in resolution.
- 4:2:2 ratio: For every 2 luminance samples, there are 2 chrominance samples horizontally, but only 1 chrominance sample vertically. This means that the resolution of the chrominance information is reduced horizontally but remains the same vertically, resulting in less aggressive downsampling compared to 4:2:0.

3. **Block Splitting** - The image is then divided into small blocks of 8x8 pixels in size

4. **Discrete Cosine Transform (DCT)** - DCT is applied to each block of pixels. DCT is mathematical transformation that converts the spacial domain information (pixel values) into frequencies. This allows to separate the image into different frequency components, with lower frequencies representing the general image structure and higher frequencies representing finer details. The DCT transform is defined as follow:

$$F(u,v) = \frac{1}{4}C(u)C(v)\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right]$$

The Inverse Discrete Cosine Transformation (IDCT) is used to reconstruct the image back to original spacial domain:

$$f(x,y) = \frac{1}{4}C(u)C(v)\sum_{u=0}^{7}\sum_{v=0}^{7} F(u,v)\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right]$$

where:

- $F(u,v)$ is the DCT coefficient at frequency index (u,v)
- $f(x,y)$ represents the pixel value at position (x,y) within the block
- $C(u), C(v)$ are normalisation constants:
  $C(u) = \frac{1}{\sqrt{2}}, u = 0$
  $C(u) = 1, u > 0$
  $C(v)$ follows the same way

The next figure (2.7) depicts the DCT application process applied to an image of Lena. As we see, the difference between the original image and the reconstructed one is quite small (Mean Squared Error (MSE) $\approx$ 1.61e-27).

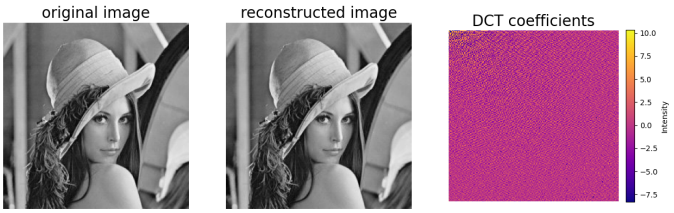

Figure 2.6: Lena, Application of DCT and IDCT, DCT coefficients

5. **Quantisation** - In this step it's used quantisation matrices, in the previous step we got the matrix of all $F(u,v)$ for a 8x8 block, now we will divide those matrices element-by-element, similary to the Hadamard Product (2.4) but using division.

$$\widehat{Q} = B \oslash Q$$

Where:

- $\widehat{Q}$ is the quantised block
- B is the 8x8 block
- Q is the quantisation matrix

Figure 2.7: Lena, Application of DCT, Quantisation and Reconstructed image

The image reduced in size about 5 times and then reconstructed with an error of about 10%, the quality of compression depends on the quantisation matrix.

6. **Direct Current term (DC) Coding and Zig-Zag Sequence** - The first element of each block after DCT is called DC, it has the information about

the average colour of all pixels in the block. The remaining 63 elements are called Alternating Current term (AC) that represent the colour changes. The DCs of all blocks are encoded using Differential Pulse Code Modulation (DPCM), this is a simple method where each DC coefficient is encoded as the difference from the DC term of the previous block in the encoding order.

$$DC_i = DC_i - DC_{i-1}$$

Where:

- DC_i is the DC coefficient of the $i-th$ block
- DC_{i-1} is the the DC coefficient of the previous block
- for the first block, we don't apply the algorithm because it's there is no blocks before it

Then the AC coefficients are ordered in a Zig-Zag sequence for the next step.

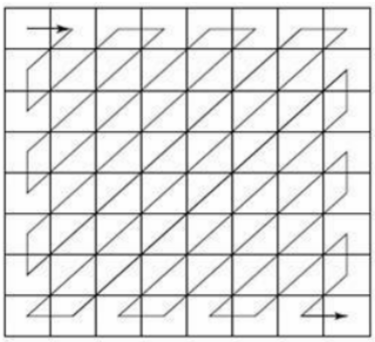

Figure 2.8: Zig-Zag Sequence (Hasan et al. [2012])

7. **Entropic Encoding** - Entropic Encoding is the type of encoding that is based on the concept of Entropy. In the Theory Information, Entropy measures the average amount of information contained in each message or symbol in a data stream. Mathematically, it's defined by the following formula:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \cdot \log_2(P(x_i))$$

where:

- $H(X)$ is the entropy of a text X
- $P(x_i)$ is the probability of the i-th character x_i

- $\log_2$ denotes the base-2 logarithm
- n is the number of distinct characters

So the Entropic Encoding algorithms basically try to minimise the Entropy of some text. In JPEG Standard are usually used three algorithms: Run-length Coding, Huffman Encoding and Mixed Algorithm (Huffman Encoding applied to Run-length Coding).

- **Run-Length Coding (RLC)**

  RLC is a lossless compression algorithm used in JPEG to avoid repetitive data, in this case, the blocks from the quantisation step have a lot of zeros, so RLC is used to spend less bits storing that information. A RLC unit is defined as:

  $$(z,l)n$$

  where:

  - z is the number of zeros before the next nonzero entry
  - l is the length of that nonzero entry
  - n is the nonzero entry

  Let's make an example. Suppose after DCT and quantisation we have this matrix:

  $$\begin{bmatrix} 298 & -31 & -17 & 5 & -1 & -3 & -1 & 2 \\ 4 & 1 & 1 & -2 & 1 & 0 & -1 & 0 \\ -6 & -2 & 2 & -1 & 1 & -1 & 1 & -1 \\ 0 & 1 & -2 & 0 & -1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

  The Zig-Zag sequence will be [298, -31, 4, -6, 1, ..., -1, 0, ..., 0], some of the parts are omitted for simplicity, the RLC of it will be:

  (9)298, (0,5)-31, (0,3)4, (0,3)-6, (0,1)-1, ..., (1,1)-1, (0,0)

  The first element is the DC coefficient and, since it's the first element of any block, we don't need to store the zero quantity before it because it's always 0.

- **Huffman Encoding**

Huffman Encoding is also a lossless compression algorithm used in JPEG, it uses the character frequency to optimize data representation.

The following diagram depicts the Huffman tree generated for the message 'STANDARD'. To obtain the code for each character, traverse from the root to the leaves. For instance, 'S' corresponds to 000, and 'A' corresponds to 10.

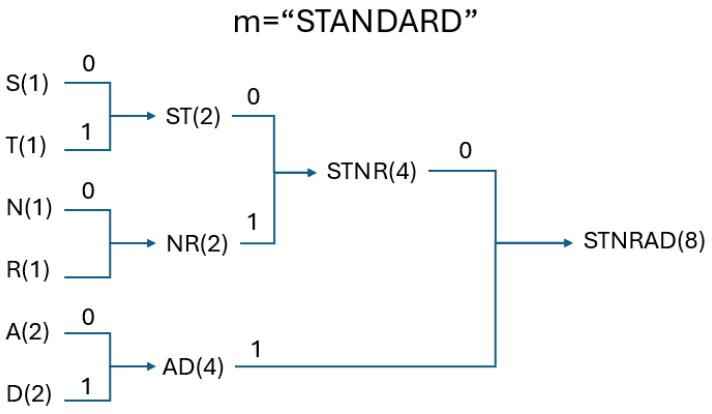

Figure 2.9: Huffman tree for the message "STANDARD"

2.4 MDC

In the current digital era, robust transmission methods are necessary to ensure the efficient transmission of high-quality data over a potentially unstable network. One method to get around these problems is to use MDC which is a technique that creates several distinct representations, or descriptions, of some source data. Since each description can be independently decoded, the original data can be reconstructed even in cases where some descriptions are missing or were corrupted during transmission. As a result, the quality of the reconstructed data will be higher if more descriptions are received.

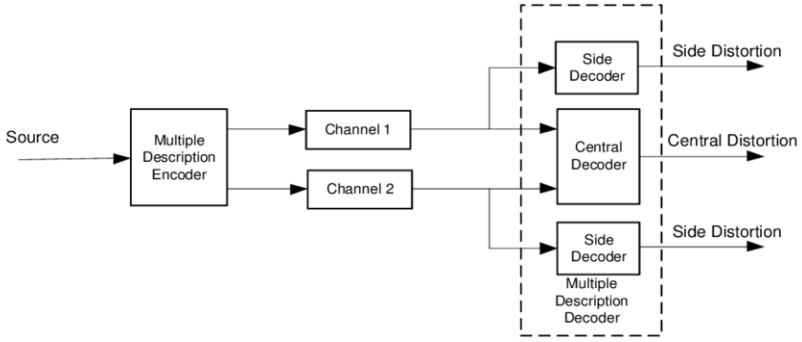

Figure 2.10: MDC with two descriptions (Kazemi et al. [2017])

There are several ways to generate those descriptions:

- **Transform-based method**: The signal is transformed into different domain and divided into descriptions. A great way to do that is by using transformations such as the DCT that are particularly effective for this purpose, as they convert spatial domain data into frequency domain data, allowing for the separation of higher frequency components from lower ones (2.3).

- **Quantisation-based method**: The signal is quantised differently to create multiple descriptions. For example, by applying different quantization steps, we will generate distinct descriptions, since each description can independently approximate the original signal, and if all descriptions are received, they can be combined to reconstruct the original data with higher accuracy.

- **Prediction-based method**: Predictive coding generates different descriptions by predicting signal parts. Predictive coding is a technique used to efficiently encode a signal by predicting its values based on previously

encoded information, the main idea is to achieve compression by lowering the signal's redundancy.

2.5 Deep Learning

Deep Learning is a field of machine learning where patterns/relationships are represented by algebraic circuits with varying strengths for each connection. These circuits, organised into multiple layers, allow for deep computation paths from inputs to outputs, simulating the human's brain activity.

This method is widely applied in many different fields, including speech recognition, machine translation, and visual object detection. Its efficacy in managing high-dimensional data sets distinguishes it from techniques such as decision trees or linear regression. Deep learning models can capture complex relationships between input variables, making them highly expressive for real-world data. The approach involves training circuits with long computation paths, allowing for intricate interactions between input variables.

2.5.1 Supervised Learning

When the learning agent is given input-output pairs, it will learn the relationship between the inputs and the outputs so then it can use it to estimate the output values of new data that will follow the same relationship.

2.5.1.1 Decision Trees

Decision trees are intuitive models for decision-making processes, the decisions are made by going through a series of tests, where each non-terminal node is a test of the value of a certain attribute and the results are marked on the edges and each leaf node specifies the decision to be made if it is reached.

For example, let's make a decision like whether to wait for a table at a restaurant, we can consider to the decision the wait time, reservation availability, and restaurant rating. These attributes help to structure the decision-making process, where the goal is to get a decision or a classification.

The figure 2.11 shows an example of a decision structured by a Decision Tree. If we make will pass to that Decision Tree the input [0,0,1], we will get the decision to wait for a table.

2.5.1.2 Linear Regression

Linear Regression is basically a way to connect dots on an N-dimension plane infering about the relationship between a dependent variable y and one or

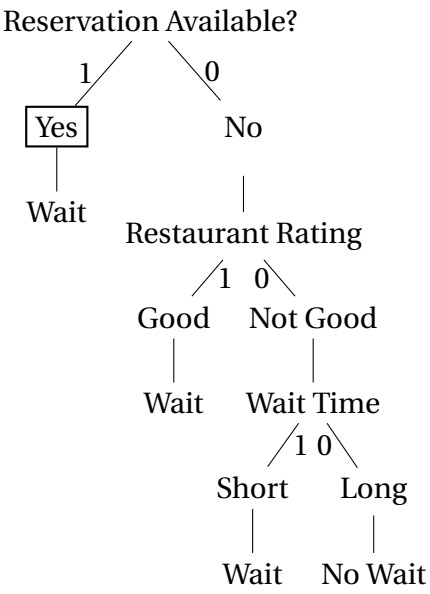

Figure 2.11: Decision tree for deciding whether to wait for a table

more independent variables x:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \epsilon$$

Where:

- y is the dependent variable (the variable we are trying to predict),
- $x_1, x_2, \ldots, x_n$ are the independent variables (features),
- $\beta_0, \beta_1, \ldots, \beta_n$ are the coefficients (parameters) representing the weight of each independent variable on the dependent variable,
- ϵ is the error term, representing the difference between the predicted value and the actual value of y.

The goal of linear regression is to estimate the coefficients that minimize the error between the predicted values and the actual values of the dependent variable. This is typically done by minimizing the cost function such as the MSE.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2.1}$$

Where:

- n is the number of observations in the dataset,
- y_i is the actual value of the dependent variable for the i-th observation,
- $\hat{y}_i$ is the predicted value of the dependent variable for the i-th observation.

Once the coefficients are estimated, the linear regression model can be used to make predictions on new data by plugging in the values of the independent variables into the equation.

Let's imagine a simple example, we have a dataset of heights and weights of 5 people and we want to be able to predict a person's height by the weight. For example, $(w(kg), h(cm)) = \{(80,180),(70,175),(60,170),(90,186),(50,160)\}$, we want to find a formula of this type: $h(w) = aw + b$

The formula to estimate the parameters a and b when we have only one independent variable is:

$$a = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{2.2}$$

$$b = \bar{y} - a\bar{x} \tag{2.3}$$

Where:

- n is the number of data points.
- (x_i, y_i) are the observed data points.
- $\bar{x}$ and $\bar{y}$ are the average values of x and y respectively.

Applying this formula, we will get to the result like: $h(w) = 1.78w + 17.71$, of course it's not very accurate because the dataset is very small, if we did this experiment with at least 100 samples we would get a model that is more corrected than this one. For example, if a new person says to me that he weights 90kg, I would tell him that his height is about 177cm as predicts this simple model that we created.

2.5.2 Gradient

The gradient is a vector that shows how a function changes. In the case of a function with a single independent variable $f(x)$, the gradient is the derivative function $f'(x)$. For more complex functions, the gradient has that role:

$$\nabla f(x_1, x_2, ..., x_n) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

Where $\frac{\partial f}{\partial x_i}$ is the partial derivative, meaning that in this operation, x_i is taken as the independent variable.

For better readability, we can write $\nabla f(x_1, x_2, ..., x_n) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n}\right)$.

Given the function $f(x_1, x_2) = ax_1 + bx_2^2 + c$:

$$\frac{\partial f}{\partial x_1} = a, \quad \frac{\partial f}{\partial x_2} = 2bx_2$$

So, the gradient of f is:

$$\nabla f = \begin{bmatrix} a \\ 2bx_2 \end{bmatrix}$$

2.5.3 Gradient for Minimization

In many scenarios, determining the minimum value of a function is crucial, and the gradient offers a powerful tool for achieving this. In elementary calculus, we've learned that to locate the minimum value of a function, we set its derivative equal to zero. This fundamental principle extends to multi-variable functions, where setting the gradient equal to zero leads us to critical points or extrema.

Let's apply this concept to finding the optimal parameters a and b for Linear Regression:

We define our function as: $f(x) = ax + b$. Our objective is to minimize the error function, for which the MSE (2.1) serves as a suitable measure:

$$\nabla \text{MSE}(a, b) = 0 \iff \nabla \text{MSE}(a, b) = \left(\frac{\partial \text{MSE}}{\partial a}, \frac{\partial \text{MSE}}{\partial b}\right) = (0, 0)$$

$$\frac{\partial \text{MSE}}{\partial b} = \frac{\partial}{\partial b}\left(\frac{1}{n}\sum_{i=1}^{n}(y_i - (\hat{y}_i))^2\right) \overset{\hat{y}_i = ax_i + b}{=} \frac{\partial}{\partial b}\left(\frac{1}{n}\sum_{i=1}^{n}(y_i - (ax_i + b))^2\right) = 0 \iff$$

$$\iff \frac{-2}{n}\sum_{i=1}^{n}(y_i - ax_i - b) = 0 \iff \sum_{i=1}^{n}(y_i - ax_i - b) = 0 \iff$$

$$\iff \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} ax_i - \sum_{i=1}^{n} b = 0 \iff \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} ax_i - nb = 0 \iff$$

$$\iff b = \frac{\sum_{i=1}^{n} y_i - \sum_{i=1}^{n} ax_i}{n} \iff b = \bar{y} - a\bar{x}$$

Here, $\bar{y}$ represents the average value of y, and $\bar{x}$ is the average value of x.

$$\frac{\partial \text{MSE}}{\partial a} = 0 \iff \frac{\partial \text{MSE}}{\partial a} = \frac{\partial}{\partial a}\left(\frac{1}{n}\sum_{i=1}^{n}(y_i - (ax_i + b))^2\right) = 0 \iff \frac{-2}{n}\sum_{i=1}^{n} x_i(y_i - ax_i - b)$$

$$\overset{b=\bar{y}-a\bar{x}}{=} \frac{-2}{n}\sum_{i=1}^{n} x_i(y_i - ax_i - (\bar{y} - a\bar{x})) = 0 \iff \sum_{i=1}^{n}(x_i y_i - \bar{y}x_i) - a\sum_{i=1}^{n}(\bar{x}x_i - x_i^2) = 0 \iff$$

$$\iff a = \frac{\sum_{i=1}^{n}(x_i y_i - \bar{y}x_i)}{\sum_{i=1}^{n}(\bar{x}x_i - x_i^2)} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

We conclude that we have found the optimal parameters formula for Linear Regression, which we have previously used (2.2, 2.3).

2.5.4 Gradient Descent

Gradient Descent is an iterative optimization algorithm used to minimize an error function (or cost function), but more generally, it's used to find minimum values of any function. Instead of setting the gradient to zero, we progressively move towards the minimum value step-by-step. This method is particularly useful for complicated functions where it's impractical or impossible to find the minimum analytically by setting the gradient equal to zero.

The Descent Gradient algorithm is basically:

1. Initialize w_0
2. Repeat until a stopping criterion is met:
    a) Calculate $w_i \leftarrow w_{i-1} - \alpha \nabla f$, where α is the step that we want to perform in the direction of the minimum.
    b) Check stopping criterion (e.g., $||\nabla f|| < \epsilon$ or maximum iterations reached).

Quick Solved Exercise

Let's assume we did a Linear Regression on some dataset and we need to optimise our parameters, by that we need to minimise the error function. Let's assume that the error function we obtained was $E = (\beta_0 - 8)^2 + \beta_1^4$, where β_0, β_1 are parameters of a function of the type $f(x) = \beta_1 x + \beta_0$. We easily can make the derivations and get the gradient vector:

$\nabla E = \begin{bmatrix} 2(-8+\beta_0) \\ 4\beta_1^3 \end{bmatrix}$

Let's use then the Gradient Descent algorithm with $\alpha = 0.5$ and Stopping Criteria as $\nabla E = 0$:

Let's start with $w_0 = (10, 0)$, for example.

$$w_1 = \begin{bmatrix} 10 \\ 0 \end{bmatrix} - 0.5 \times \nabla E(10,0) = \begin{bmatrix} 10 \\ 0 \end{bmatrix} - 0.5 \times \begin{bmatrix} 4 \\ 0 \end{bmatrix} = \begin{bmatrix} 8 \\ 0 \end{bmatrix}$$

Checking if $\nabla E = 0$. It's not, so we proceed:

$$w_2 = \begin{bmatrix} 8 \\ 0 \end{bmatrix} - 0.5 \times \nabla E(8,0) = \begin{bmatrix} 8 \\ 0 \end{bmatrix} - 0.5 \times \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 8 \\ 0 \end{bmatrix}$$

Since $\nabla E = 0$, we stop the iteration. Hence, $w = \begin{bmatrix} 8 \\ 0 \end{bmatrix}$.

So we found the values for β_0 and β_1 that minimize the error function, $E = (8-8)^2 + 0^2 = 0$. Even though this is a very simplistic case, it shows why this method is used.

2.5.5 Hadamard Matrix Multiplication

The next topics will use matrices, so this subsection will just explain really quick what the Hadamard Product is. When we have two vectors of the same dimensions, we can get the Hadamard Product just multiplying the element-by-element on both matrices at the same position, that means that $a \odot b = (a \odot b)_j = a_j b_j$.

For example:

$$\begin{bmatrix} 2 \\ 3 \end{bmatrix} \odot \begin{bmatrix} 5 \\ 7 \end{bmatrix} = \begin{bmatrix} 10 \\ 21 \end{bmatrix} \tag{2.4}$$

2.5.6 Autoregressive Model

Autoregressive Probabilistic Models predict the next value based on the previous ones, these models are primarily used in stock markets to forecast the future prices of assets by analyzing a company's historical data.

Mathematically, the Autoregressive model can be expressed as:

$$X_t = c + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \ldots + \phi_p X_{t-p} + \epsilon_t$$

where:

- X_t is the value at time t,
- c is a constant (bias),
- $\phi_1, \phi_2, \ldots, \phi_p$ are the weights/importance of X_t's,
- ϵ_t is the white noise error term.

Even though the formula is the model's definition, it's often used another formula based on probabilities:

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i | X_1, X_2, ..., X_{i-1})$$

This formula indicates that the probability of observing a dataset $\mathbf{X}$ of length n is the product of the conditional probabilities of each value given the previous values.

Let's say we have the following historical data for a stock price:

Time (t)	Price (X)
1	100
2	102
3	101
4	103
5	105

Assuming $c = 0$, $\phi_5 = 0.5$, $\phi_4 = 0.3$, $\phi_3 = 0.2$, ϕ_2 and ϕ_1 are 0, and the white noise term ϵ_t is negligible, the model can predict the next value X_6:

$$X_6 = 0.5 \times X_5 + 0.3 \times X_4 + 0.2 \times X_3 = 103.6$$

Thus, the predicted stock price at time $t = 6$ is 103.6.

2.5.7 Single Perceptron - artificial neuron

Frank Rosenblatt introduced the perceptron concept in the 1950's (Neil [2015]). The perceptron is an algorithm for supervised learning, specifically for binary classification tasks. It learns from a set of training examples, adjusting its parameters to correctly classify the input data into one of two categories. The

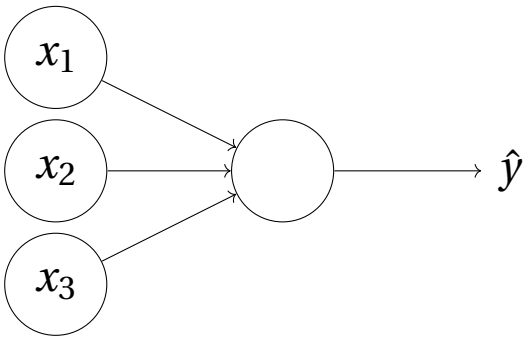

Figure 2.12: Example of a Perceptron

model reminds a real neuron cell, just like a neuron in the brain, a perceptron receives input "signals", processes them, and generates an output "signal".

In the example shown, the perceptron is illustrated with three inputs x_1, x_2, x_3. However, the Perceptron model doesn't inherently have a fixed input quantity. The core principle of its operation lies in assigning different weights to each input. For instance, x_1 is assigned a weight denoted as w_1, and similarly for the other inputs. The perceptron then computes the weighted sum of all inputs. If this sum exceeds or equals a certain threshold, denoted as q, the perceptron outputs 1; otherwise, it outputs 0:

$$\hat{y} = \begin{cases} 0 & , \sum w_j x_j \leq q \\ 1 & , \sum w_j x_j > q \end{cases}$$

Instead of using sums, we can simplify the representation using matrices:

$$\sum w_j x_j = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \cdot \begin{bmatrix} x_1 x_2 ... x_n \end{bmatrix} = \boldsymbol{w} \cdot \boldsymbol{x}$$

To normalise the sum and enhance flexibility, we can introduce a bias term b resulting in the expression $\boldsymbol{w} \cdot \boldsymbol{x} + b$:

$$\hat{y} = \begin{cases} 0 & , \boldsymbol{w} \cdot \boldsymbol{x} + \mathrm{b} \leq 0 \\ 1 & , \boldsymbol{w} \cdot \boldsymbol{x} + \mathrm{b} > 0 \end{cases}$$

Individual perceptrons are not sufficient for more complex tasks, so usually multiple perceptrons are arranged into a network structure, where the perceptrons are arranged into layers, each of which has a specific role to play in processing the incoming data. By setting up this kind of structure, the system creates a hierarchy of understanding. With this arrangement, information can be represented hierarchically, lower layers are focused on recognizing simple features, and then higher layers build more abstract characteristics

on top of the representations that lower layers have learnt.

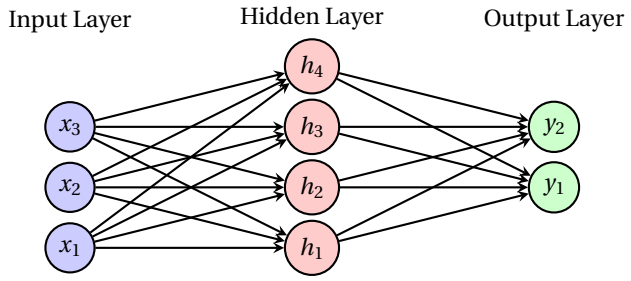

Figure 2.13: Neural Network Diagram

2.5.8 Sigmoid neurons

Previously, we've explored linear functions as a solid starting point. However, there is a problem: small changes in input weights could lead to wide differences in outputs. To overcome this, we introduce a powerful tool called the Sigmoid Function, $\sigma(z)$, in the neurons, so that's why they are called sigmoid neurons.

The Sigmoid Function is defined as the follow:

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Where z is the input.

Sigmoid neurons are similar to perceptrons, but since they use a sigmoid function instead of a simple linear one, little variations in their weights and biases result in only minor variations in their output.

Moreover, without this non-linearity, it would be impossible or non-efficient to represent non-linear functions. This is because composed linear functions yield a linear function, limiting the capacity of neural networks to capture complex relationships from the data. By introducing non-linear activation functions, neural networks can efficiently model and approximate a wide range of non-linear functions.

2.5.9 Popular Activation Functions

Even though the sigmoid function is a great activation function, there are many others used in neural networks. Sometimes, the task of choosing an activation function is subjective or empirically determined (i.e., testing a set of functions and selecting the best one). In this subsection, we will mention

two activation functions that are referred to in the next subsections: Rectified Linear Unit (ReLU) and sinusoidal. The figure 2.14 shows the graphic of these different activation functions.

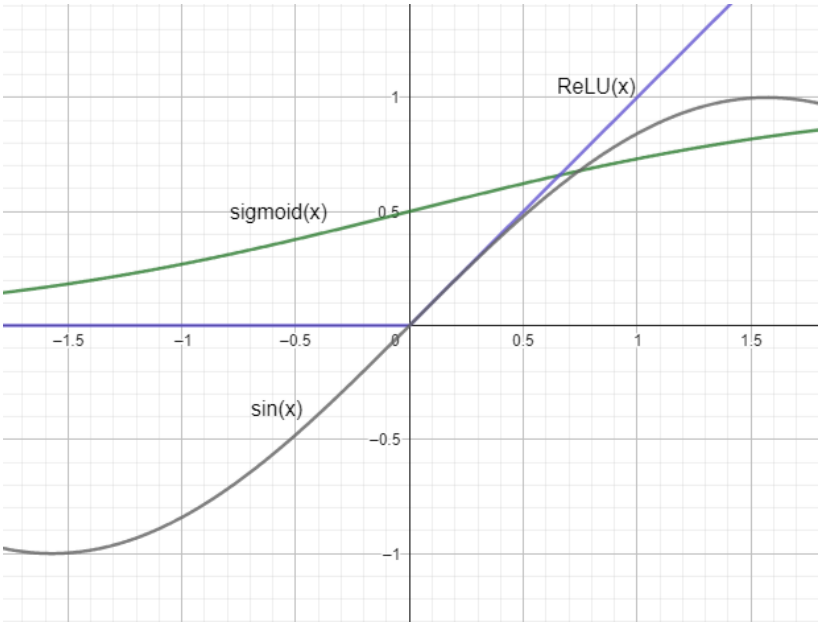

Figure 2.14: Activation functions

1. **ReLU**

   This function is widely used because it's fast to compute, since for negative numbers it will just output 0 and for positive inputs it will return the value back as output, it is defined mathematically as:

   $$\text{ReLU}(x) = \max(0, x)$$

   However, ReLU is not without its drawbacks. One common issue is the "dying ReLU" problem, where neurons can become inactive and only output zero, effectively preventing learning, this happens when the neurons might have large negative values, since ReLU outputs zero for negative values, these neurons can end up outputting zero.

2. **Sinus Activation Function**

   The sinusoidal activation function is inspired by sinusoidal functions and, although less common, offers unique benefits in specific contexts related to periodic patterns.

2.5.10 Backpropagation

Backpropagation is an algorithm in training neural networks that facilitates the learning of a model from data and enhancement of performance, it's similar to the Gradient Descent Algorithm, but much faster.

The goal of Backpropagation is to get the partial derivates of a cost function with respect to any weight of the network or bias, that means that we want to understand how changing the weights and biases changes the cost function. Usually it's used the quadratic cost function:

$$C = \frac{1}{2n} \sum_x ||y(x) - a^L(x)||^2$$

Where:

- n - total number of inputs
- y - desired output for x
- L - number of layers in the network
- $a^L(x)$ - the actual output for x in the output layer

The error function for each neuron j in layer l, δ^l_j, can be represented as the partial derivative of the cost function related to the input in that neuron. This is a good approach since that if the partial derivative is close to zero, we can say that the error is close to zero as well, and if the derivative is large, then there is a way to reduce it.

$$\delta^l_j \equiv \frac{\partial C}{\partial z^l_j}$$

Usually, this formula is not used, because makes more sense to estimate the error by the weights and by the bias rather than by the input value itself. In the following algorithm, there will be used equivalent formulas for the error.

The backpropagation algorithm can be summarized as follows:

1. **Input** x
2. **Feedforward**: Compute the activations a^l for each layer l

$$z^l = w^l a^{l-1} + b^l; a^l = \sigma(z^l)$$

3. **Output error**: Calculate the error term δ^L for the output layer L using the formula:

$$\delta^L = \nabla_a C \odot \sigma'(z^L)$$

4. **Backpropagate the error**: For each $l = L-1, L-2, \ldots, 2$ get:

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$$

5. **Output**: The gradient of the cost function is given by

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j; \frac{\partial C}{\partial b^l_j} = \delta^l_j$$

6. **Update weights**: Similarly to the Gradient Descent algorithm, we update the weights, where α is the learning rate.

$$w^l_{jk} \leftarrow w^l_{jk} - \alpha \times \frac{\partial C}{\partial w^l_{jk}}$$

2.5.11 Autoencoders

An autoencoder is a special neural network that basically receives an input, transforms it to a less dimensional representation, and then reconstructs the the input value using the less dimensional representation (Dertat [2017]).

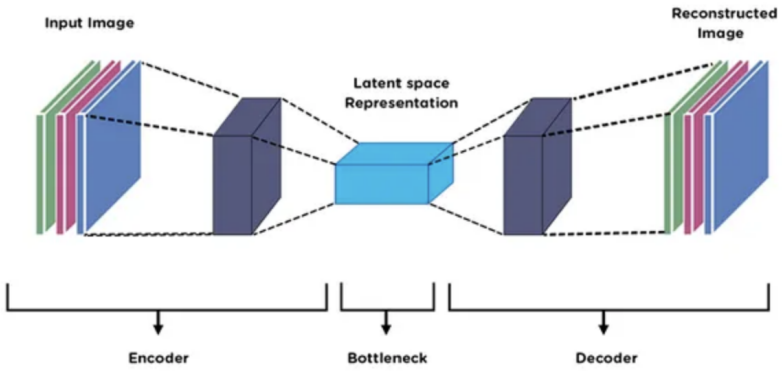

Figure 2.15: Autoencoder Architecture (Dertat [2017])

This is very useful when compressing images, because if we can to represent the same information with less data then we are saving memory.

The structure of an autoencoder consists of an encoder and a decoder, which are typically neural networks.

Architecture

An autoencoder consists of three main components:

1. Encoder ϕ - This component compresses the input data into latent space representation

$$z = \phi(x)$$

   Where:

   - x is the original image
   - z is the encoded image (latent space)

2. Decoder ψ - The decoder takes the encoded representation from the latent space and reconstructs the original input data.

$$\hat{x} = \psi(z)$$

   Where:

   - $\hat{x}$ is the reconstructed image
   - z is the encoded image (latent space)

2.6 Upscaling an image

Sometimes, when we want to enlarge an image (zoom in), it inevitably reduces the quality. For instance, when we zoom in on a computer an image, it reaches a point where individual pixels become visible, degrading the image's overall sharpness.

To mitigate this quality loss, interpolation techniques are employed to insert new pixel values into the existing grid of the image.

In this section we will talk about four popular algorithms: Nearest Neighbour, Linear Interpolation, Bilinear Interpolation and Bicubic Interpolation. Bicubic Interpolation is specially important because it is mentioned in the next chapter.

2.6.1 Nearest Neighbour

"Nearest Neighbour" is a simplistic interpolation algorithm where new pixel values are assigned based on their proximity to existing pixels. However, this method tends to produce pixelated or blocky images due to its simplistic approach.

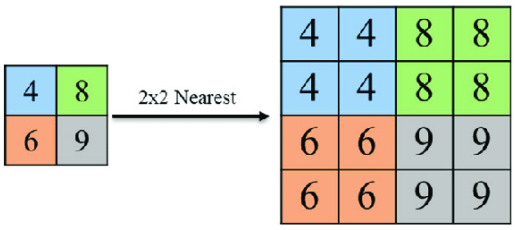

Figure 2.16: "Nearest Neighbour" Example (Sheu et al. [2022])

2.6.2 Linear Interpolation

Linear interpolation assigns new pixel values by "drawing" a straight line between two known pixel values. It calculates a linear function that connects these two values. When a new pixel is inserted between the known values, its value is determined by this linear function, attempting to make a smooth transition between the original pixels. The generic formula of this function can be defined as:

$$y = y_1 + \frac{(y_1 - y_2)}{(x_1 - x_2)}(x - x_1)$$

where:

- y_1, y_2 are the values of the pixels x_1 and x_2
- x is the relative distance from the pixel x_1 to x_2 of the new pixel

Let's see this example, we have two known pixels with values 20 and 100, then we upscale it and we need to insert a new pixel, so we need to calculate its value, let's assume that the first pixel is in position 0 and the second one when upscaled in position 2:

$$y = 20 + \frac{(20 - 100)}{(0 - 2)}(x - 0) = 20 + 40x$$

We need to calculate the value $y(1)$ for the new pixel, $y(1) = 20 + 40(1) = 60$

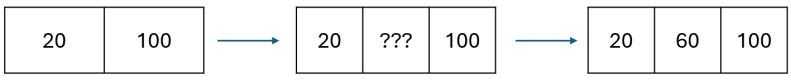

Figure 2.17: Example of Linear Interpolation

2.6.3 Bilinear Interpolation

Bilinear Interpolation is basically the same thing as the Linear one, but in two dimensions, that means we need to do Linear Interpolations horizontally and vertically:

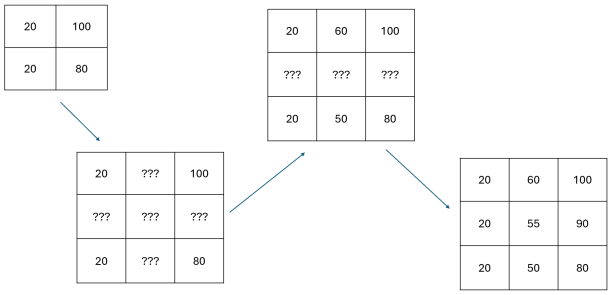

Figure 2.18: Example of Bilinear Interpolation

2.6.4 Bicubic Interpolation

Bicubic Interpolation is similar to the Bilinear one, but instead of using linear functions, it uses the cubic one horizontally and vertically:

$$y = ax^3 + bx^2 + cx + d$$

To solve this equation, we need the values of four pixels in each dimension, resulting in a total of 16 pixels. Bicubic interpolation provides even smoother and more visually appealing results compared to linear and bilinear methods, making it a preferred choice for high-quality image resizing.

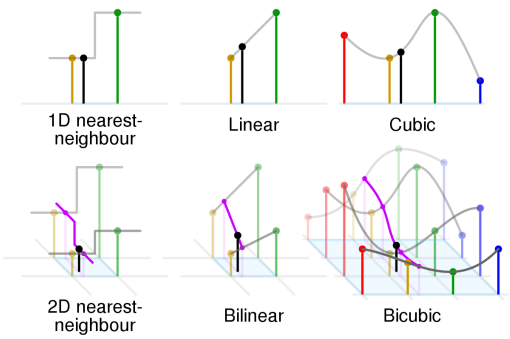

Figure 2.19: Comparison between interpolations (Cmglee [2019])

2.7 Conclusion

In this chapter, we explored the necessary concepts that are relevant to understand the next chapters.

Chapter

3

State of Art

3.1 Introduction

Initially, the DNA encoding methods weren't focused on the type of data to be encoded. These works showed the importance of defining constraints and not simply translate directly binary code into nucleotic quaternary. Only the most important methods will be referred in section 3.2.

In recent years, have emerged methods of DNA encoding of images following the global tendency to consume visual content. This methods already took in consideration more seriously the biological constraints and adaptations of JPEG standards of compressing images to DNA. The history of these methods will be referred in section 3.3 and the most important to explain the HiDNA codec will be detailed.

Lately, it started to appear encoding methods using artificial intelligence concepts to efficiently compress images before translating them to DNA. The HiDNA codec is an example of them that uses Implicit Neural Representations, therefore the most important works on this subject will explained in the section 3.3.

3.2 General DNA Coding

3.2.1 Church, Gao and Kosuri in 2012

In 2012, it was proposed a pioneering method for storing digital data in DNA by Church, Gao and Kosuri (Church et al. [2012]). Their method involved encoding binary sequences into a quaternary representation by randomly as-

signing adenine (A) or cytosine (C) to 0 and thymine (T) or guanine (G) to 1. They used this approach to encode a 659-Kbyte book into synthetic DNA.

This work represented a significant advancement in the field, but it also revealed insights into sequencing errors and established key constrains for encoding data into DNA. The goal of further research was to address these issues, especially by minimizing sequencing errors with the introduction of error correction methods.

3.2.2 Goldman Encoding

In 2013, Goldman et al. [2013] proposed to use Ternary Huffman Encoding to avoid homopolymers and there was presented a way to transcode the resulting outputs from the Huffman Code to nucleotides. This approach was particularly advantageous in avoiding homopolymers, which are sequences of identical nucleotides that can cause errors in DNA synthesis and sequencing processes because they can lead to higher error rates in sequencing, making data retrieval less reliable.

Previous nucleotide	0	1	2
A	T	C	G
T	A	C	G
C	A	T	G
G	A	T	C

Table 3.1: Goldman's transcoding table

The next table, there are some examples of transcoding the Huffman ternary code into DNA.

Huffman 3-nary Code	DNA Representation
01100102010122	TCTATCAGACACGC
10012211101001	CATCGCTCTACATC
22211100022211	GCGTCTATAGCGTC

In this work, they also included a error correction procedure based on redundancy, it's based on overlapping bases between adjacent fragments so that every bit was represented by various oligos.

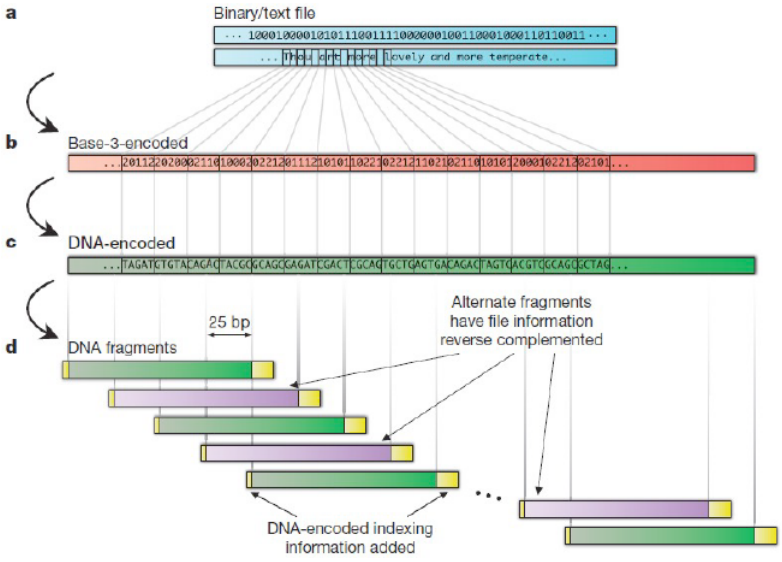

Figure 3.1: Oligos overlapping (Limbachiya et al. [2015])

3.3 Image DNA coding

In recent years, with visual content now comprising at least 80% of all online content (Personify [2024]), several methods have emerged for encoding images in DNA, highlighting its potential impact on the majority of the world's content.

In recent advances, since 2019, more specialised coding techniques adapted to image data have been developed. The Mediacoding team has introduced several digital image encoding solutions, emphasizing adaptive compression techniques.

One notable contribution is the Paircode algorithm (Dimopoulou et al. [2019]) which is an algorithm that generates robust quaternary code in a way to avoid homopolymers and the quantity of CG content is less or equal to the quantity of AT content, this approach is different from the earlier methods that were based on transcoding directly the binary sequences to DNA. Later proposal (Dimopoulou and Antonini [2021]) improved the compression rate of the algorithm by using vector quantisation.

In 2022, Pic and Antonini [2022] proposed a variable-length quaternary encoder based on the Shannon-Fano algorithm adapted to DNA data storage, the algorithm successfully avoids homopolymers longer than 3 nucleotides. This novel entropy coder was introduced in the JPEG based image coding with increased performance over the state-of-art JPEG image codec (Dimopoulou et al. [2021]).

Recently, some works started using neural networks to compress images efficiently. In Strebel et al. [2023] by combining a convolutional autoencoder (see section 2.5.11) with a Goldman encoder. In Le et al. [2023], the first MDC scheme (see section 2.4) using Implicit Neural Representation (INR) (Dupont et al. [2021], Tancik et al. [2020]) is proposed. The proposed scheme achieves high performance and flexible redundancy tuning. Moreover, a generalized model training is unnecessary. The goals of implementing MDC in DNA data storage are twofold: minimising the reading cost and enhancing noise robustness. In Le et al. [2024] was proposed a Spatial Frequency Multiple Description based on Le et al. [2024] but generalized to N descriptions.

For the purpose of this work, more detailed will be explained the Paircode algorithm (Dimopoulou et al. [2019]) and the adapted Shannon-Fano algorithm (Pic and Antonini [2022]).

Since the HiDNA codec uses Implicit Neural Representations to represent images in a compact way, the Dupont et al. [2021] and Tancik et al. [2020] works will be also explained more detailed.

3.3.1 A biologically constrained encoding solution for long-term storage of images onto synthetic DNA

In 2019, Dimopoulou et al. proposed a simple fixed-length code construction algorithm that provides the quaternary representation of the data after quantisation - PAIRCODE.

PAIRCODE Algorithm:

1. **Creation of the codewords** - Codewords are unique sequence of nucleotides that represent some piece of information.

   To form this codewords the algorithm has two dictionaries:

   $C_1 = \{AT, AC, AG, TA, TC, TG, CA, CT, GA, GT\}$

   $C_2 = \{A, T, C, G\}$

   The codewords of even length are constructed by selecting $\frac{l}{2}$ elements of C_1, the codewords of odd size are constructed by selecting $\frac{l-1}{2}$ elements of C_1 and one element of C_2

2. **Mapping values to codewords** - In the thesis of Melpomeni Dimopoulou (Dimopoulou [2020]) is proposed to map the same value to two different codewords each one in different "class" - A and B. For example, if we have the K values, the class A would contain all codewords since c_0 to c_K and class B would contain codewords since c_{K+1} to c_{2K}. The idea

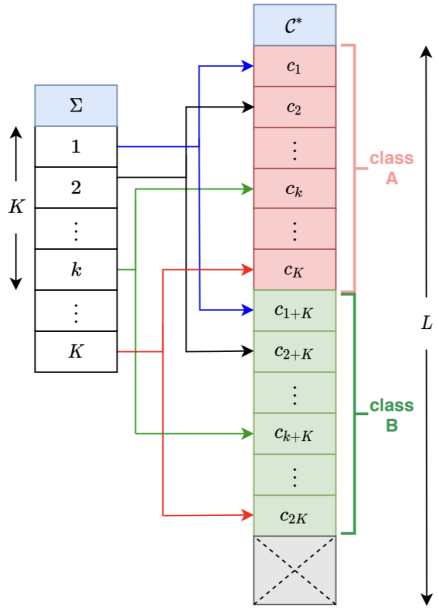

Figure 3.2: PAIRCODE Mapping (Dimopoulou [2020])

is then to assign one codeword from each class to the same value, that would increase robustness and error correction.

3.3.2 A constrained Shannon-Fano entropy coder for image storage in synthetic DNA

In 2022, Xavier Pic and Marc Antonini (Pic and Antonini [2022])proposed a novel variable-length quaternary encoder adapted to DNA data storage based on the Shannon-Fano algorithm adapted to operate within a quaternary alphabet comprising the nucleotide bases A, C, G, T, while following to biochemical constraints. The work focused more in addressing constraints on homopolymers.

The main difference from the Standard JPEG codec is that it uses the proposed variable length quaternary encoder based on the classic Shannon-Fano algorithm.

Classic Shannon-Fano Algorithm:

1. Create a list of probabilities for each character of the text

2. Sort the list of characters in decreasing order of probability

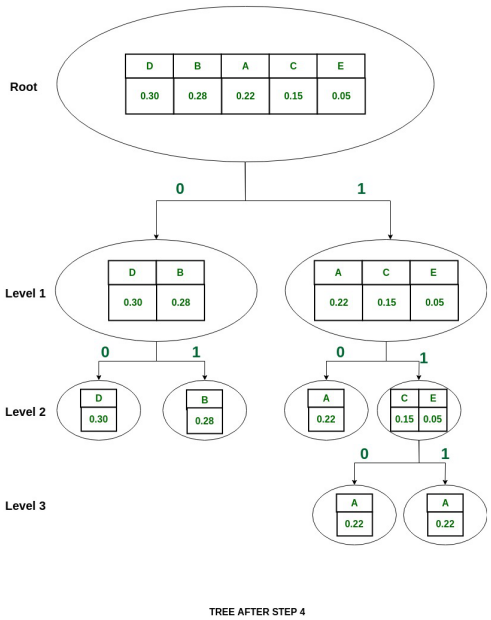

Figure 3.3: Classic Binary Shannon-Fano Algorithm (gee [2024])

3. Split the list into two parts, with the total probability of both parts being as close to each other as possible

4. Assign the value 0 to the left part and 1 to the right part

5. Repeat steps 3 and 4 for each part until all the symbols are split into individual subgroups

This coder basically have two stages:

1. **Quaternary Coding** - The Algorithm used in this codec instead of splitting in 2 (standard binary representation), it splits in 4 (quaternary representation) but in 3-to-3 layers of the tree it splits in 3 instead of 4, as shown in the figure 3.4, that helps to avoid homopolymers.

2. **Transcoding to DNA** - The output from the previous step is not the DNA representation but a quaternary code that has two types of symbols: the constrained ones - those that are encoded in ternary representation (each 3th layer) and the unconstrained ones - the rest of the symbols. The constrained ones are encoded using the Goldman table that it was already explained earlier (subsection 3.2.2) and the rest are directly transcoded using the following table:

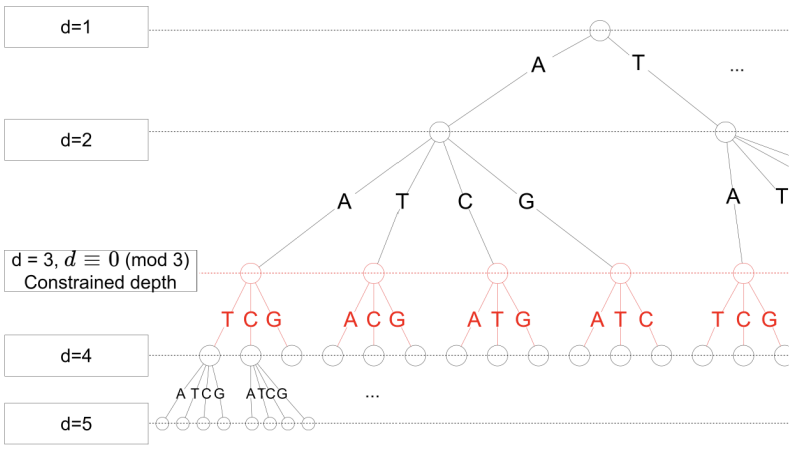

Figure 3.4: Proposed Algorithm based on the classic Shannon-Fano in Pic and Antonini [2022]

Symbol	0	1	2	3
Nucleotide	A	T	C	G

Table 3.2: Quaternary Transcoding Table

3.3.3 Coordinate-based neural representations

Coordinate-based Neural Representation (CNR), sometimes called implicit neural representations or neural fields, transfer input coordinates directly to signal values using fully-connected neural networks, usually Multi-layer Perceptron (MLP) (see figure 3.5). These neural networks provide a continuous and adaptable means of representing signals, in contrast to discrete techniques that depend on set grids (inr [2024], Tancik et al. [2020]).

The core of this approach is the function f_θ, an MLP parameterized by θ, that maps a coordinate x (e.g., a 2D pixel coordinate) to the signal value at that point (e.g., RGB colour). Formally, this can be expressed as:

$$f_\theta : x \rightsquigarrow T(x)$$

where $T(x)$ is the actual signal value at that point x and the idea of f_θ is to be close to that value and θ is the vector of weights of a MLP. For that the MPL f_θ is trained to minimise the loss function L, typically MSE (2.1).

$$\theta^* = \arg\left(\min_\theta \sum_{i=1}^{N} L(f_\theta(x_i), T(x_i))\right)$$

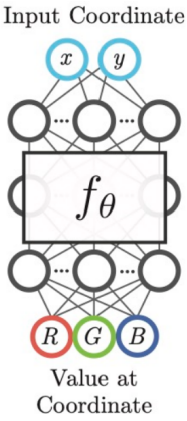

Figure 3.5: CNR model (Tancik et al. [2020])

The training process is usually done by Gradient Descent (2.5.4) in order to find θ the overall Loss function.

3.3.4 Compression with Implicit Neural Representations (COIN)

COIN is referred here because it's a fundamental component of the HiDNA codec, therefore it's essential to mention how this works. COIN instead of saving the RGB values of each pixel of the image, it stores the weights of a neural network that can generate that image, this is like it was described earlier at the previous section (3.3.3). The primary concept of COIN was published in a publication (Dupont et al. [2021]) that left room for optimisations for later work.

Key Processes of COIN

For consistency, the author's (Dupont et al.) nomenclature is used in the following sections to explain the main COIN procedures.

1. **Encoding**

   To encode an image I, the COIN method fits an MLP f_θ to the image. The MLP maps pixel locations (x, y) to RGB values, $f_\theta(x, y) = (r, g, b)$. The encoding process minimises the MSE between the output from the MLP and the image's actual RGB values:

$$\min_\theta \sum_{x,y} \| f_\theta(x, y) - I[x, y] \|_2^2$$

The parameters θ of the MLP are quantised and that is the compressed representation of the image.

2. **Decoding**

   The decoding process involves evaluating f_θ at each pixel location to reconstruct the image, this allows for progressive decoding by evaluating the f_θ at various pixel locations, providing flexibility for resource-constrained devices.

COIN instead of using standard activation functions like sigmoid or ReLU, it uses sinusoidal activation functions, that enhances its capacity to get high-frequency data.

Due to the per image learning process that needs optimisation mechanisms, COIN performs slower than state-of-the-art approaches, but it has potential to improve in the future. Additionally, the compact size of the model is a notable advantage, as it reduces storage requirements significantly, because it's basically needed the θ parameter (of a small network) to reconstruct the image, as it can be verified in the figure 3.6.

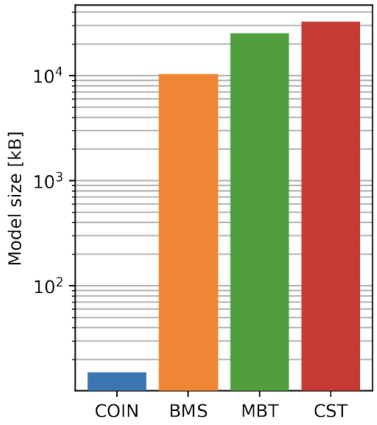

Figure 3.6: Model sizes (Dupont et al. [2021])

3.4 Conclusion

In conclusion, there is an urgent need for creative and effective storage solutions given the recent exponential rise in digital data. The intriguing prospect of using DNA as a data store medium presents an intriguing way of solving this

problem, because it exhibits exceptional characteristics like the enormous storage capacity incorporated in its molecular structure, which make it a promising substitute for conventional storage techniques.

Furthermore, the discussion on DNA storage in the context of the JPEG Standard underscores the practical implications of this technology for image storage and retrieval. We can open up new possibilities for effectively storing and retrieving large amounts of digital material by fusing the concepts of DNA storage with accepted practices in digital photography.

Chapter

4

HiDNA codec

4.1 Introduction

The HiDNA codec was proposed for the JPEG DNA call of proposal, developed by the I3S laboratory in partnership with the Pearcode Company. This software compresses images and stores them in formats for nucleic recording (FASTA) and, obviously, also performs the reverse process. It is based on a Neural Implicit Representation model, which overfits the MLP to reconstruct the image to code give the latency spaces as input. Furthermore, it uses an autoregressive model that estimates the distribution of quantised latency spaces.

4.2 HiDNA Codec

In this section, it will be introduced the HiDNA codec, its architecture is shown in the Figure 4.1.

In short, the process involves the following steps: During training, the N latent spaces are initially divided into 8x8 blocks. Each block is then individually quantised with added uniform noise before being fed into the synthesis module. This module generates the reconstructed output and calculates the corresponding distortions $D_0, ..., D_N$. The latent space is updated through back-propagation based on the measured distortion using MSE. At the same time, the auto-regressive model is refined to better predict the distribution of the quantised latent space, which is crucial for entropy encoding processes.

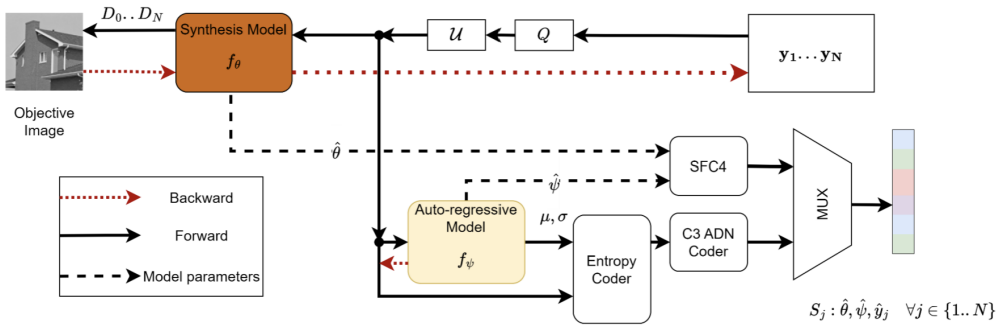

Figure 4.1: Architecture of the HiDNA codec (Pic et al. [2023])

4.2.1 Step-by-step Analysis

4.2.1.1 Synthesis Model - f_θ

The synthesis model is based on COIN technology (see section 3.3.4) with some modifications. We start with a set of latent spaces organised hierarchically by resolution y_k, where k is the resolution level.

Then, each latent space y_k is divided into M blocks, each of size 8×8, the block is denoted as y_k^b, where b is the block index $b \in \{0, 1, \ldots, M-1\}$. Then each block is quantised with a unique quantisation step Δy_k^b:

$$\hat{y}_k^b = Q(y_k^b, \Delta y_k^b)$$

Thus, the quantised latent space $\hat{y}_k$ is defined as the set of all quantised blocks:

$$\hat{y}_k = \{\hat{y}_k^b \in \mathbb{Z}^{8\times 8} : b \in \{0, 1, \ldots, M-1\}\}$$

And, the latent space is defined as a set of quantised latent spaces $\hat{y}_k$:

$$\hat{y} = \{\hat{y}_k \in \mathbb{Z}^{H_k \times W_k} : k \in \{0, 1, \ldots, L-1\}\}$$

where $H_k = \frac{H}{2^k}$, $W_k = \frac{W}{2^k}$ and L is the hierarchical level of $\hat{y}_k$. Then the latent spaces are upsampled with a bicubic interpolation to match the image dimensions.

In order to facilitate optimisation during training, the quantisation is replaced by noise addition to the latent space $\hat{y}_k$ before it is fed into the synthesis model, this is done to make the quantisation process differentiable, because for Gradient Descent it's imperative to differentiate the loss function for a continuous variable (see section 2.5.4). Specifically, during training, a uniform noise u is sampled from the range $[-0.5, 0.5]$:

$$\hat{y}_k = y_k + u, \quad u \sim U[-0.5, 0.5]$$

But, when we need to entropy encode, we need discrete values, so it's used a uniform quantiser Q. This is also important for reconstructing the image, as we need to map to discrete RGB values.

$$\hat{y}_k = Q(y_k)$$

The synthesis model then reconstructs the image based on the received latent spaces.

4.2.1.2 Autoregressive Model - f_ψ

To entropically encode and decode the latent spaces, it is essential to determine their probability distributions, because this kind of codifications aim to compress data by reducing redundancy, making the encoded data more efficient.

Therefore, in order to get that distribution probabilities it's used an autoregressive model because the pixel value is of course conditioned with its neighbours:

$$p_\psi(\hat{y}) = \prod_{i,k} p_\psi(\hat{y}_{ik}|c_{ik})$$

where $\hat{y}_{ik}$ is the latent pixel at position i of level k and c_{ik} is the set of decoded neighbouring pixels of $\hat{y}_{ik}$.

$$p_\psi(\hat{y}_{ik}|c_{ik}) = \int_{\hat{y}_{ik}-0.5}^{\hat{y}_{ik}+0.5} g(y)\,dy \quad \text{with} \quad g(y) \sim \mathcal{L}(\mu_{ik}, \sigma_{ik})$$

In order to get the μ_{ik} and σ_{ik} it's used an MLP f_ψ that will learn the those parameters, that means that:

$$\mu_{ik}, \sigma_{ik} = f_\psi(c_{ik})$$

4.2.1.3 Entropy Encodings

After the retrieval of the probabilities, it's time to entropically encode the parameters of the MPLs and the latent spaces.

1. **Range Coder** - C3

   The Range coder, Pic et al. [2023], is used to compress the latent spaces, it uses an entropy encoding method used in data compression, which efficiently encodes symbols based on their probabilities reducing redundancy. Let's see an example that will explain this concept better:

   So let's imagine we need to encode the following sequence of characters "AATC" and the probabilities are P([A,T,C,G]) = [0.3, 0.3, 0.2, 0.2].

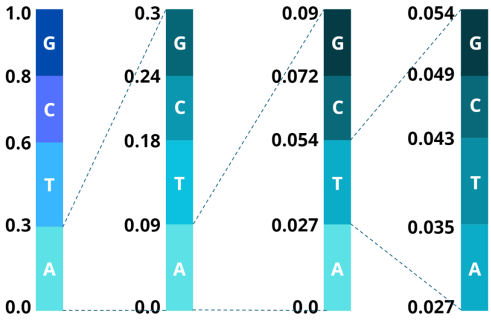

Figure 4.2: Range Encoding Example

The process of range encoding the sequence "AATC" can be broken down as follows:

a) **Initialisation:** The initial range is set to $[0,1[$, because the probability possible range is from 0 to 1.

b) **Encoding 'A':** The symbol 'A' has a probability of 0.3. Therefore, the range for 'A' is $[0,0.3[$.

c) **Encoding 'A':** The current range is $[0,0.3[$. We subdivide this range using the same probabilities:

- **A**: $[0,0.3\times 0.3[= [0,0.09[$
- **T**: $[0.09,0.09+0.3\times 0.3[= [0.09,0.18[$
- **C**: $[0.18,0.18+0.3\times 0.2[= [0.18,0.24[$
- **G**: $[0.24,0.24+0.3\times 0.2[= [0.24,0.3[$

The range for the second 'A' is now $[0,0.09[$.

d) **Encoding 'T':** The current range is $[0,0.09[$. We subdivide this range using the same probabilities:

- **A**: $[0,0.09\times 0.3[= [0,0.027[$
- **T**: $[0.027,0.027+0.09\times 0.3[= [0.027,0.054[$
- **C**: $[0.054,0.054+0.09\times 0.2[= [0.054,0.072[$
- **G**: $[0.072,0.072+0.09\times 0.2[= [0.072,0.09[$

The range for 'T' is now $[0.027,0.054[$.

e) **Encoding 'C':** The current range is $[0.027,0.054[$:

- Length of current interval: $0.054-0.027=0.027$
- **A**: $[0.027,0.027+0.027\times 0.3[= [0.027,0.0351[$
- **T**: $[0.0351,0.0351+0.027\times 0.3[= [0.0351,0.0432[$

- **C**: $[0.0432, 0.0432 + 0.027 \times 0.2[= [0.0432, 0.0486[$
- **G**: $[0.0486, 0.0486 + 0.027 \times 0.2[= [0.0486, 0.054[$

The range for 'C' is now $[0.0432, 0.0486[$.

In this example, the sequence "AATC" is encoded into a final range of $[0.0432, 0.0486[$ from which we get a random value, for example 0.045. This example illustrated the efficiency of range coding in compressing extensive data into a single value.

To decompress the data, the same algorithm is applied in reverse using the compressed value, the number of encoded characters, and their probabilities; by knowing these three variables, it becomes straightforward to reconstruct the original data. I encourage the reader to mentally perform this reconstruction by watching the Figure 4.2.

This kind of coders is used at the HiDNA codec, but instead of initialising the range with $[0,1[$, it's done $[0, 2^{32} - 1[$, this choice prioritizes precision and efficiency in operations involving integers over floating-point numbers, while maintaining the same underlying principle.

2. **Adapted MQ coder**

   The next step involves transcoding the output to DNA using the MQ coder (Pic et al. [2023]). At this stage, the output from the previous step is not a floating point number but an unsigned 32-bit integer. For the transcoding process, a codebook consisting of 48 codewords, each of length 3, is used. These codewords are specifically designed to avoid homopolymers.

$$C3 = \{0 : \text{"AAT"}, 1 : \text{"AAC"}, 2 : \text{"AAG"}, ..., 46 : \text{"GGT"}, 47 : \text{"GGC"}\}$$

   MQ coder, in the way that is implemented, will convert the 32 bit unsigned integer in base 48 and at the same time translate it to DNA based on $C3$:

$$\text{MQ\_encode}(x) = \text{transcode}(x, 5)$$

   where:

$$\text{transcode}(x, i) = \begin{cases} C3[\text{to\_base48}(x, 5)], & \text{if } i = 0 \\ C3[\text{to\_base48}(x, i)] \oplus \text{transcode}(x, i-1) & \text{if } i > 0 \end{cases}$$

   where:

- $\oplus$ is the concatenation operator
- to_base48 is defined as:

$$\text{to\_base48}(x,i) = \left(\left\lfloor \frac{x}{48^{5-i}} \right\rfloor \mod 48 \right)$$

Thus, the final output of the MQ coder will be the DNA representation.

3. **Adapted Shannon-Fano Encoder - SFC4**

SFC4 is used to encode the synthesis model parameter θ and the autoregressive model parameter ψ. This encoder was designed by Xavier Pic and Marc Antonini in 2022 (Pic and Antonini [2022]). The details of SFC4 were already explained in the *State of Art* section (page 41).

4.2.1.4 Rate and Cost function

Distortion The HiDNA codec to get the metric for image reconstruction uses the MSE defined in the following way:

For an image with C colour channels, width W and height H, the distortion based on MSE can be defined as:

$$D_k = \frac{1}{C \times W \times H} \sum_{i=1}^{C \times W \times H} (\hat{x}_{i|k} - x_i)^2$$

where $\hat{x}_{i|k}$ is the $i-th$ pixel of the reconstructed image $\hat{x}$ from the resolution k.

Rate Rate of an image is the quantity of bits needed to encode a pixel on average, usually higher rates indicate better quality, but that also means that the image size would be bigger. After entropy encoding, the redundancy of the data is removed, so the rate will be different. For a quantised latent space $\hat{y}$, the rate is defined as:

$$R(\hat{y}) = -\sum_{i,k} \log_2 p_\psi(\hat{y}_{ik}|c_{ik})$$

where:

- $\hat{y}$ is the quantised latent space
- $p_\psi(\hat{y}_{ik}|c_{ik})$ is the probability of the quantised latent pixel $\hat{y}_{ik}$ given its context c_{ik}

4.2.1.5 Cost Function

The previous two components define the cost function. Generally, a higher rate corresponds to lower distortion, and greater distortion corresponds to a lower rate. The goal is to find the best trade-off between distortion and rate:

$$J_{(\lambda)}(\theta, \psi, \hat{y}) = D + \lambda(R(\hat{y}) + R(\psi) + R(\theta))$$

However, in practice, since the parameters θ and ψ are small, the bit-rate cost associated with them is considered negligible during training and only taken into account after the training process. Therefore, the cost function can be simplified to:

$$J_{(\lambda)}(\hat{y}) = D + \lambda R(\hat{y})$$

Thus, for an image, the HiDNA codec training process consists of minimising this cost function.

4.2.1.6 Formatting

In order to store data corresponding to the input image, it is necessary to serialize the data, encapsulating it in a format that stores all the necessary variables, it follows the format showed in the picture 4.3.

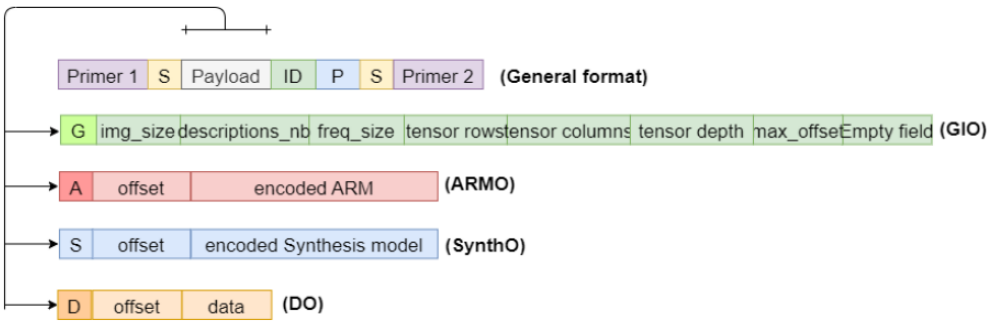

Figure 4.3: Format used by the HiDNA software (Le et al. [2023])

According to the content, there are 4 types of oligos:

- **Global Information** (GIO) - it stores the general info like the sizes of different variables such as the descriptions and models parameters (synthesis and autoregressive);
- **Autoregressive Model Parameters** (ARMO) - it stores the Autoregressive model parameters;

- **Synthesis Model Parameters** (SynthO) - it stores the Synthesis model parameters;
- **Latent Space** (DO) - it stores the quantised latent space.

HiDNA codec encodes the image into a set of small sequences stored in a FASTA file, the FASTA format is a text-based format for representing sequence of nucleotides, it uses the same letters as science for the nucleotides (A,T,C,G).

4.2.1.7 Decoding

During decoding, the process starts with decoding the header (GIO), followed by decoding the network parameters (ARMO and SynthO). Next, the latent spaces are extracted from the bitstream (DO) using an entropy coder that utilizes statistics estimated by the decoded autoregressive model. Then, the image is reconstructed using the latent spaces received.

4.3 Conclusion

The HiDNA codec undoubtedly has a complex and elegant theoretical basis. In this chapter, we detailed the codec step-by-step, the CNR and entropy coding techniques used by this codec to effectively compress and optimise data using advanced algorithms.

Since this method uses complex artificial intelligence concepts, a brief explanation of the concepts had to be included in the chapter 2 to give the readers a deeper understanding of the more complex aspects.

In the next chapter, we will discuss evaluating codec performance. The discussion will review the methodologies and procedures used to evaluate the effectiveness and efficiency of the HiDNA codec and provide insight into its practical applications and benefits.

Chapter

5

Evaluation of the HiDNA codec

5.1 Introduction

This chapter presents the implementation of the evaluation for the HiDNA codec, including the metrics and statistical analysis employed. The evaluation was based on "JPEG DNA Common Test Conditions" paper (Antonini and Ebrahimi [2023]), which specifies the dataset, image formats, and metrics applied.

5.2 JPEG DNA Dataset

The JPEG DNA dataset used for the performance evaluation of the codecs (HiDNA and EPFL) consists of a set of 10 images, they were chosen because of their diversity in terms of content, colour, and spatial resolution.

Figure 5.1: Common dataset for JPEG DNA codecs

The images are differentiated by their number and resolution, as indicated in the table below:

IMAGE NUMBER	CONTENT	RESOLUTION (pixels)
00001	Object	1192x832
00002	Human face	853x945
00003	Food	945x840
00004	Computer-generated	2000x2496
00005	Animal	560x888
00006	Water	2048x1536
00007	Night scene	1600x1200
00008	Fabric/fine texture	1430x1834
00009	Landscape	2048x1536
00010	Buildings	2592x1946

Table 5.1: Image content and resolutions

The dataset used is available for download through FTP:

FTP credentials

```
Protocol: FTP
FTP address: tremplin.epfl.ch
Username: jpegdna@mmspgdata.epfl.ch
Password: fT76Dl
FTP port: 21
Folder name: 04-2023
```

5.3 Objective Quality Evaluation

5.3.1 Structural Similarity Index (SSIM)

The SSIM is a widely used metric for evaluating the perceptual quality of images and measuring the similarity between two images (Ravindra [2020]). Unlike traditional methods that compute pixel-wise differences (such as Mean Squared Error), SSIM considers changes in structural information, luminance, and contrast.

- **Luminance** - Luminance of an image is calculated by getting the average value of all pixels:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i$$

The function that compares two images by luminance $l(x,y)$ can be defined as:

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$

where:

- μ_x, μ_y - average value of the luminance for the images x and y
- C_1 - small constant to avoid division by 0

- **Contrast** - Contrast can be defined as the standard deviation of all the pixels of an image.

$$\sigma_x = \sqrt{\left(\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2\right)}$$

The function that compares two images by contrast $c(x,y)$ can be defined as:

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

where:

- σ_x, σ_y - value of the contrast for the images x and y
- C_2 - small constant to avoid division by 0

- **Structure** - Structural comparison can be understood as the attempt to compare the overall tendency of the pixel values of two images, this can be achieved through using the covariance:

$$\sigma_{xy} = \left(\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y)\right)$$

And the overall comparison is done by the function $s(x,y)$ that can be defined as:

$$s(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

Thus, SSIM(x,y) is defined as:

$$\text{SSIM}(x,y) = [l(x,y)]^{\alpha} \cdot [c(x,y)]^{\beta} \cdot [s(x,y)]^{\gamma}$$

Where α, β, γ are the importance weights for each parameter, typically they are all set to 1.

Even though, this metric isn't used at the evaluation directly, it is still the basis of other metrics used.

5.3.2 Multi-Scale SSIM (MS-SSIM)

MS-SSIM is a perception based model that compares details across different resolutions, providing a comprehensive evaluation of the quality relation between reference and distorted images, it is based on SSIM and it is particularly effective for learning-based image codecs due to its flexibility in handling variations in image resolution and viewing conditions (Wang et al. [2016]). Higher score means better image quality.

Mathematically, it can be defined as:

$$\text{MS-SSIM}(x,y) = [l_m(x,y)]^{\alpha_M} \cdot \prod_{j=1}^{M} [c_j(x,y)]^{\beta_j} \cdot [s_j(x,y)]^{\gamma_j}$$

Where:

- j - resolution j that is obtained by downsampling by factor of 2 the previous
- M - worst scale, that is the maximum value of downsamplings

5.3.3 Information Content Weighted SSIM (IW-SSIM)

Not all regions of an image are perceived the same way to the human eye, then it makes sense to split the image in regions and weight each region by it's importance to the total perception. This metric assumes that perceptual weights should be proportional to the local information content, thereby improving its correlation with subjective scores from established databases. So it's basically a comparison between the weighted averages of all regions (Wang et al. [2003]).

$$\text{IW-SSIM}(x,y) = \frac{\sum_i w_i \cdot \text{SSIM}(x_i, y_i)}{\sum_i w_i}$$

Where SSIM(x_i, y_i) is the SSIM for the i-th region.

5.3.4 Video Multimethod Assessment Fusion (VMAF)

VMAF is a quality assessment metric developed to estimate video quality by combining scores from various quality assessment algorithms using a support vector machine (SVM)(Zhang et al. [2021]). Though originally designed for videos, VMAF is also effective for evaluating single images, especially in the context of learning-based image codecs. Higher score means better image quality.

5.3.5 Visual Information Fidelity (VIF)

VIF quantifies the loss of visual information perceived by humans during image compression (sci, VIF). It operates in the wavelet domain and it measures information fidelity, making it highly aligned with human perceptual response. It's a similar approach to IW-SSIM, but instead of doing a weighted average between regions, it's done between frequency bands. Higher score means better image quality.

5.3.6 PSNR-Human Visual System (PSNR-HVS-M)

PSNR-HVS-M is a model that incorporates characteristics of the human visual system using DCT basis functions. It calculates the maximum distortion that remains invisible to the human eye due to between-coefficient masking, along with a contrast sensitivity function (Vitor [2023]).

5.3.7 Peak Signal-to-Noise Ratio (PSNR)-Y and PSNR-YUV

PSNR-Y and PSNR-YUV are traditional metrics that measure the peak signal-to-noise ratio between the original and reconstructed components of an image.

The YUV colour space is designed to separate brightness (luminance) from colour information (chrominance), reflecting the human visual system's greater sensitivity to changes in luminance than to changes in chrominance. The formula to convert RGB values into YUV is pretty simple:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

where:

- Y represents the luminance (brightness),
- U represents the blue-difference chrominance,
- V represents the red-difference chrominance.

PSNR-Y specifically measures the peak signal-to-noise ratio of the luminance component, providing insight into how well the brightness information is preserved during reconstruction. On the other hand, PSNR-YUV is a comprehensive metric that combines PSNR values for all three components: luminance (Y), and chrominance (U and V). This combined metric provides an overall evaluation of the image quality in the YUV colour space (Antonini and Ebrahimi [2023]):

$$\text{PSNR-YUV} = \frac{\text{PSNR-U} + \text{PSNR-V} + 6 \times \text{PSNR-Y}}{8}$$

5.3.8 Normalised Laplacian Pyramid Distance (NLPD)

NLPD, Laparra et al. [2016], is based on local luminance subtraction and contrast gain control, reflecting aspects of the human visual system. It uses a Laplacian pyramid decomposition to compare the quality of a distorted image to its reference, with a lower score indicating better image quality.

The Laplacian pyramid shows details of an image at different scales. NLPD calculates the Laplacian pyramid for both the reference and distorted images, then measures the distance between the two pyramids. The smaller the distance, the smaller the difference between the images.

5.3.9 Feature Similarity (FSIM)

FSIM evaluates image quality based on two low-level features: phase congruency, representing local structure (edges and texture patterns), and gradient magnitude, accounting for contrast information as result of image intensity variations (Jiang et al. [2019]). FSIM is particularly effective for colour images, with higher scores indicating that the distorted image is more similar to the reference image.

5.4 Testing of the HiDNA codec Process

5.4.1 Introduction

This study presents a detailed statistical analysis of the HIDNA codec's performance in compressing images, transcoding them to FASTA and their recon-

struction. This codec was evaluated by analysing the distortions for a dataset of images, performing multiple trials with different rates defined in the appendix A, and using statistical methods to derive insights.

5.4.2 Definitions and Notations

Let us define the necessary components and notations:

- I is the set of images: $I = \{I_1, I_2, \ldots, I_n\}$, here $n = 10$.
- m is the number of trials for each image, here $m = 10$.
- $\hat{I}_{ri|j}$ denotes the reconstructed image I_i in the j-th trial for the rate r.
- $\bar{D}_r$ represents the mean distortion rate for the rate r.
- σ_r represents the standard deviation in a set of images of rate r.

5.4.3 Data Collection

For each image I_i in the dataset I, collect the reconstructed images. This gives a set of matrices of reconstructed images $\hat{I}$:

$$\hat{I} = \{\hat{I}_j : j \in \{1,2,3,...,9,10\}\}$$

where j is the trial and $\hat{I}_j$ can be defined as:

$$\hat{I}_j = \begin{bmatrix} \hat{I}_{11|j} & \hat{I}_{12|j} & \cdots & \hat{I}_{1,10|j} \\ \hat{I}_{21|j} & \hat{I}_{22|j} & \cdots & \hat{I}_{2,10|j} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{I}_{71|j} & \hat{I}_{72|j} & \cdots & \hat{I}_{7,10|j} \end{bmatrix}$$

5.4.4 Procedure's Algorithm

1. **Data Collection**
   - For each image I_i in the dataset I and each rate r, perform 10 trials.
   - Collect the reconstructed images $\hat{I}_{r|ij}$ for each trial j.
2. **Calculate the distortion per image**

   For each image I_i and rate r, compute the distortions $D_{r|ij} = D(\hat{I}_{r|ij}, I_i)$.

3. **Get the averages per trial**

   For each image $\hat{I}_i$ and rate r, compute its mean distortion for $\bar{D_{i|r}}$ across all images and trials:

$$\bar{D}_{i|r} = \frac{1}{10} \sum_{j=1}^{10} D_{r\,i|j}$$

4. **Get the averages per rate**

   For each rate r, compute its mean distortion:

$$\bar{D}_r = \frac{1}{10} \sum_{i=1}^{10} \bar{D}_{i|r}$$

5. **Get the standard deviations per rate**

   For each rate r, calculate the standard deviation σ_r of the distortions:

$$\sigma_r = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} \left(\bar{D}_{i|r} - \bar{D_r}\right)^2}$$

   where $n = 10$

6. **Calculate the Confidence Interval at 95% per rate**

   For each rate r, determine the 95% confidence interval for the mean distortion $\bar{D}_r$ using the t-distribution:

$$\mathrm{CI}_{95\%} = \bar{D_r} \pm t_{\alpha/2,n-1} \cdot \frac{\sigma_r}{\sqrt{n}}$$

   where $t_{\alpha/2,n-1}$ is the critical value from the t-distribution corresponding to a 95% confidence level and $n-1$ degrees of freedom. The t-distribution is used here because it provides a more accurate estimate for small sample sizes (less than 30), in this case 10 (because each rate has 10 averages).

5.5 Results

5.5.1 Summary

When running the codec 10 times, it failed to make and to reconstruct certain FASTA files into images. On average only 64 images could be successfully reconstructed out of theoretical 70. The problematic files were predominantly those associated with large-scale images. Specifically, in none of the tests was it possible to obtain images for:

- **Image "00006_2048x1536"**
  - **HiDNA_00006_2048x1536_24bit_sRGB_004.png**
    Unable to obtain result to rebuild image "00006_2048x1536" for rate 4
- **Image "00010_2592x1946"**
  - **HiDNA_00010_2592x1946_24bit_sRGB_001.png**
    Unable to obtain result to rebuild image "00010_2592x1946" for rate 1
  - **HiDNA_00010_2592x1946_24bit_sRGB_002.png**
    Unable to obtain result to rebuild image "00010_2592x1946" for rate 2
  - **HiDNA_00010_2592x1946_24bit_sRGB_004.png**
    Unable to obtain result to rebuild image "00010_2592x1946" for rate 4
  - **HiDNA_00010_2592x1946_24bit_sRGB_006.png**
    Unable to obtain result to rebuild image "00010_2592x1946" for rate 6
  - **HiDNA_00010_2592x1946_24bit_sRGB_007.png**
    Unable to obtain result to rebuild image "00010_2592x1946" for rate 7

Moreover, the images that were successfully reconstructed often approached or even exceeded the values obtained by the authors. All the results obtained at this experience will be at github repository that can be accessed by the following link:
`https://github.com/Pobedinsky/Study-and-Evaluation-of-the-HiDNA-codec.git`

5.5.2 Graphics

In this section, we will conduct a detailed comparative analysis between the distortion metrics graphs generated in this study and those presented in the "Report on Crosschecking of the BioCoder and HiDNA Submissions to the JPEG DNA CfP" (Testolina et al. [2023]). The purpose of this comparison is to evaluate the accuracy, consistency, and differences between the results obtained by different entities using similar methodologies.

FSIM Performance:
In terms of FSIM, the codec for all the rates performed well. Typically FSIM > 0.98 indicates very high similarity, where distortions are almost imperceptible to the human eye, as we see in all the rates the codec outperformed that barrier.

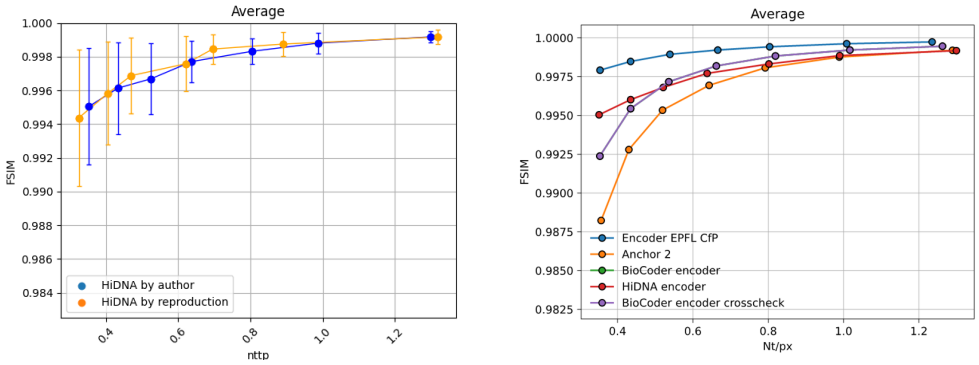

Figure 5.2: FSIM: left - this study, right - Crosschecking Report

IW-SSIM Performance:
In terms of IW-SSIM, the HiDNA codec also performed well, even though with the Interval of Confidence at 95%, the codec doesn't surpass or it is on the edge of the minimum barrier of 0.98 at the first rate, an IW-SSIM value of 0.98 or higher is generally considered sufficient to ensure that distortions are not visible to the human eye in most practical applications.

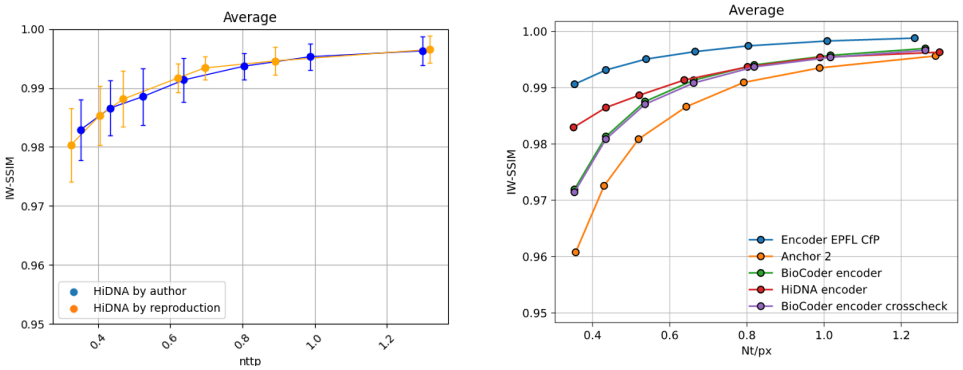

Figure 5.3: IW-SSIM: left - this study, right - Crosschecking Report

MS-SSIM Performance:
In terms of MS-SSIM, the codec performs well, it surpasses the minimum barrier of 0.98 for all the rates even taking in consideration the Interval of Confidence at 95%.

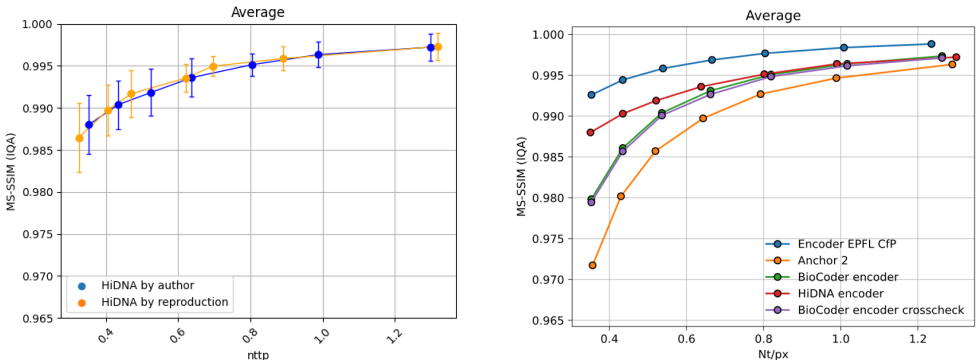

Figure 5.4: MS-SSIM (IQA): left - this study, right - Crosschecking Report

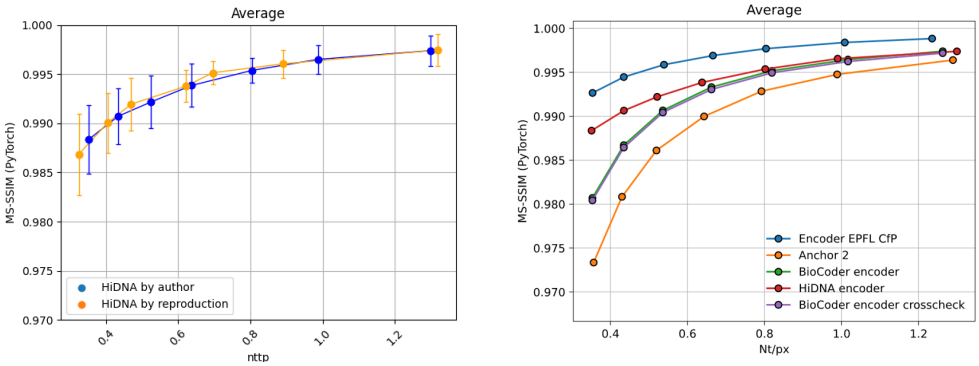

Figure 5.5: MS-SSIM (PyTorch): left - this study, right - Crosschecking Report

NLPD Performance

In terms of NLPD, there is no a consensus in terms of what is the maximum value of the distance that makes distortions non-perceptible, but the codec performs well because the values are close to 0 (perfect, no distortion).

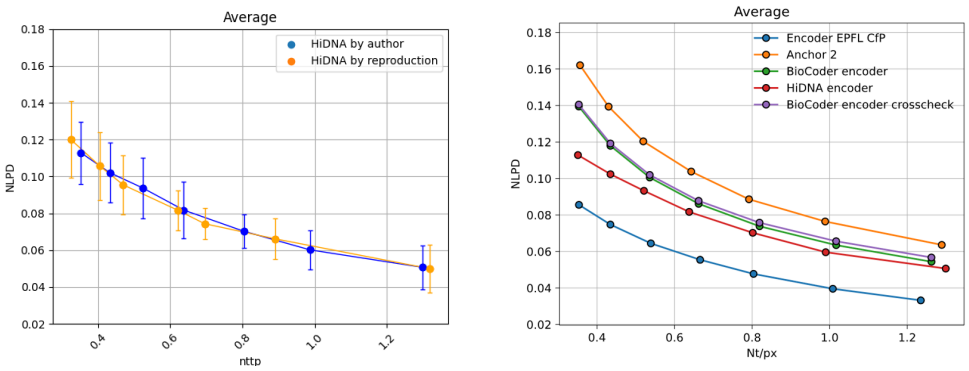

Figure 5.6: NLPD: left - this study, right - Crosschecking Report

PSNR-HVS-M Performance

In terms of PSNR-HVS-M, the HiDNA codec performed well for all rates, usually for PSNR values it's considered values above 30 as non-perceptible to human eye the distortions, as we see the codec surpasses that minimum barrier.

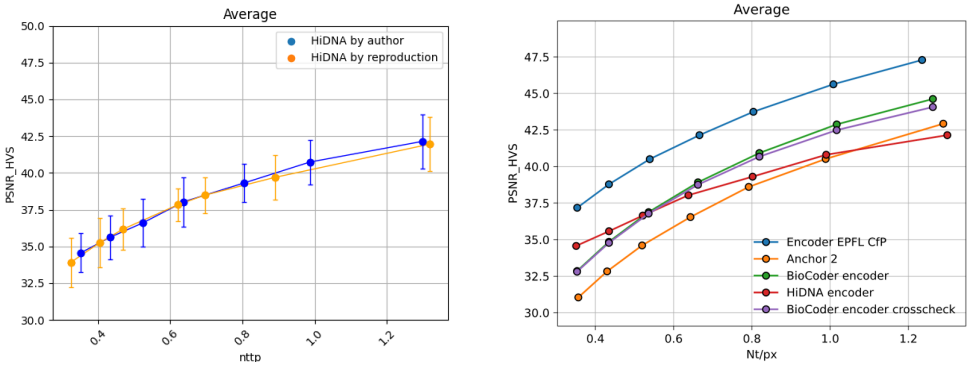

Figure 5.7: PSNR HVS: left - this study, right - Crosschecking Report

PSNR-YUV Performance

In terms of PSNR-YUV, the HiDNA codec consistently performs well across all rates, maintaining values above the crucial barrier of 30. Although the results for the 5th and 6th rates show a greater deviation from the author's results compared to the other rates, this deviation is not significant.

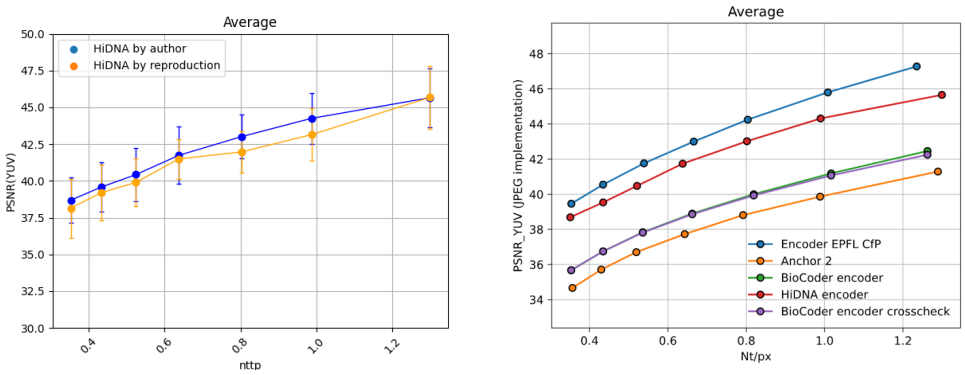

Figure 5.8: PSNR YUV: left - this study, right - Crosschecking Report

PSNR-Y Performance

In terms of PSNR-Y, the HiDNA codec also performs well across all rates, maintaining values above 30. The results of this experiment are pretty close to the ones that were obtained using the images from the HiDNA submission.

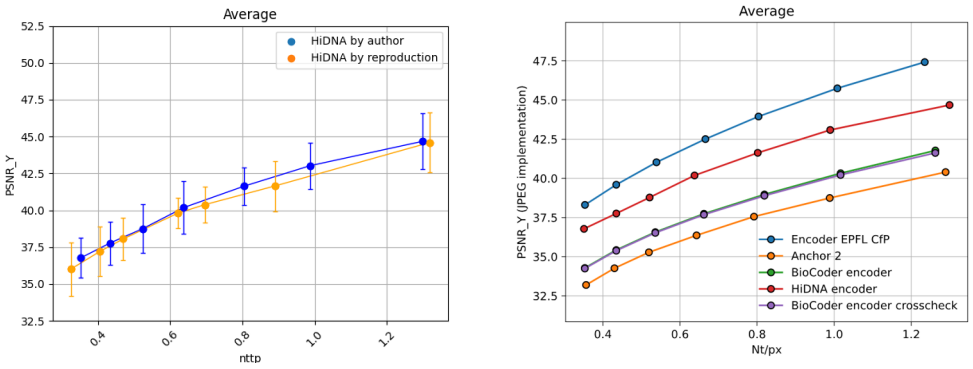

Figure 5.9: PSNR Y: left - this study, right - Crosschecking Report

VIF Performance

In terms of VIF, Sheikh and Bovik [2004], there is no a globally accepted minimum, it's generally considered a VIF score of 0.96 as barrier below which distortions became perceptible to human eye. This implication would suggest that codecs performing below this barrier might generally exhibit poorer performance in terms of preserving visual fidelity. However, further research is necessary to conclude for sure that.

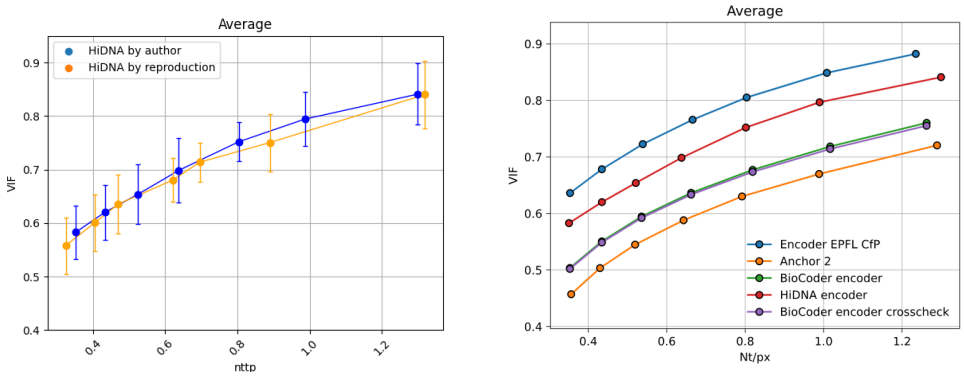

Figure 5.10: VIF: left - this study, right - Crosschecking Report

VMAF Performance

In terms of VMAF, there isn't a universally agreed minimum VMAF score for imperceptibility, generally studies suggest that VMAF scores above 90 correspond to imperceptible distortions for videos. Although VMAF was originally developed for video quality assessment, it can also be applied to images. In terms of the evaluation of the HiDNA codec, while the first three rates fell below the VMAF barrier, the deviations were minor, indicating overall satisfactory performance.

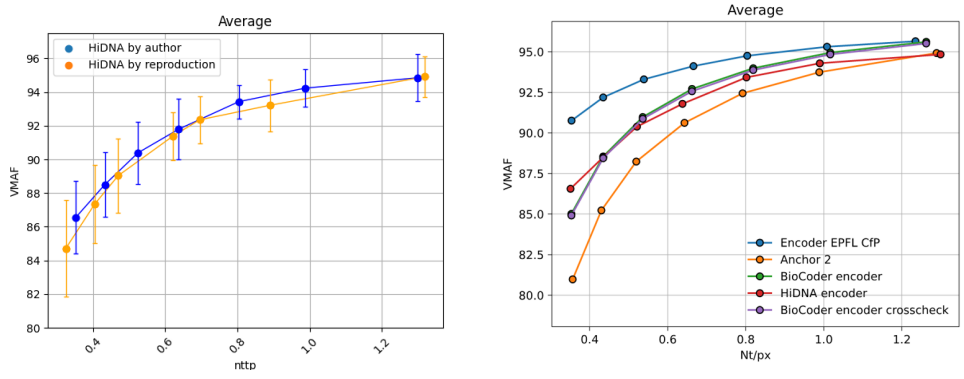

Figure 5.11: VMAF: left - this study, right - Crosschecking Report

The graphs show that the results obtained in this experiment are consistent with those presented in the Crosschecking Report (Testolina et al. [2023]), and in some cases, even surpass them. This consistency validates the effectiveness of the HiDNA codec in various testing situations. Although the HiDNA codec did not exhibit the best overall performance, it demonstrated competitive performance in terms of compression efficiency and visual fidelity. The EPFL codec, tested by my colleague Alexandre Nunes (Nunes [2024]), achieved the best overall performance, while HiDNA codec achieved the second overall performance.

5.6 Conclusion

The evaluation of the HiDNA codec revealed a tremendous capability to give good compression ratios without affecting the accuracy and quality of the reconstructed images, this codec is very successful in keeping high visual fidelity, which has been extensively examined using many different metrics. These impressive results make it an appropriate choice for any application that requires efficient data storage and high resolution image reproduction such as DNA storage considering the high cost of DNA synthesis.

However, one should note that despite having a good level of effectiveness overall, the HiDNA codec had some limitations when dealing with larger images; after encoding them, it could not reconstruct these images meaning that more work needs to be done to refine and optimise it further.

Chapter

6

General Conclusions and Future Work

6.1 Conclusions

The HiDNA codec represents a significant advance in the context of DNA JPEG. It is surprising to see how the constant and continuous efforts of European experts have produced a simple implementation and a conceptually elegant answer. This technique, which undoubtedly aligns with emerging trends that merge the biological domain of DNA with the computational domain of artificial intelligence, offers efficiency and relevance despite some inconsistencies.

The application of the concept of latency space allows this codec to achieve high compression rates, as the image is represented using fewer variables and stores not the image itself, but rather the recipe for recreating it, that means that the HiDNA codec translates complex image data into a compressed format that can be stored efficiently and later reconstructed with high fidelity.

Furthermore, the innovative integration of DNA-based storage methods with artificial intelligence algorithms exemplifies an innovative approach to solving contemporary data compression challenges. The seamless combination of these two advanced fields shows the potential of interdisciplinary research to produce transformative solutions.

Working on this project expanded my knowledge of image compression techniques and the complex algorithms behind them. I learned a lot about the beneficial uses of artificial intelligence in this industry, especially how it helps optimise data storage procedures. Moreover, this project broadened my view of the possibilities for future innovation at this crossroads, highlighting the complex relationships between biological concepts and computer systems.

6.2 Future Work

This report provides an analysis of the HiDNA codec's performance results along with a thorough description of how it operates. Although addressing software problems was not the main goal of this study, it is important to note that several difficulties were found during the evaluation process. While not within the purview of this work, debugging these difficulties would be an intriguing endeavour. Improving the software's functionality and reliability may maximise the codec's performance even further, increasing its total usefulness in real-world scenarios.

Since the evaluation of this codec was made without considering errors, a possible and interesting future development would be testing the codec with the existence of errors to make the study more complete.

Appendix

A

Rates Definition per image

The rates were defined at the codec submission in the "HiDNA_Rates.xlsx" file as follows:

Image	**r1**	**r2**	**r3**	**r4**	**r5**	**r6**	**r7**
00001_1192x832.png	0.0000288	0.000073	0.00013	0.00016	0.00032	0.0005	0.00056
00002_853x945.png	0.000035	0.00005	0.00008	0.0001	0.00013	0.00015	0.00046
00003_945x840.png	0.000025	0.00005	0.00007	0.0002	0.0003	0.0004	0.0005
00004_2000x2496.png	0.0002	0.0002255	0.000226	0.00025	0.00045	0.0004	0.0005
00005_560x888.png	0.00005	0.0001	0.00013	0.00025	0.0005	0.0007	0.001
00006_2048x1536.png	0.00002	0.00005	0.0001	0.00015	0.0002	0.0005	0.0006
00007_1600x1200.png	0.000039	0.00005	0.0001	0.00015	0.0002	0.0003	0.0005
00008_1430x1834.png	0.00001	0.00005	0.0001	0.00017	0.0002	0.0003	0.0005
00009_2048x1536.png	0.000006	0.000015	0.00003	0.00004	0.00006	0.00007	0.0001
00010_2592x1946.png	0.00005	0.00007	0.00008	0.0001	0.00012	0.00015	0.0002

Bibliography

Information fidelity. https://www.sciencedirect.com/topics/engineering/information-fidelity.

Shannon-fano algorithm for data compression, 2024. URL https://www.geeksforgeeks.org/shannon-fano-algorithm-for-data-compression/. Accessed 5th April 2024.

Implicit neural representation for vision (inrv). https://inrv.github.io/, 2024. Accessed 1st June 2024.

Marc Antonini and Touradj Ebrahimi. Jpeg dna common test conditions version 2.0. 99th JPEG Meeting, Online, April 2023. ICQ SG, JPEG DNA Exploration, Final, Public Distribution.

George Church, Yuan Gao, and Sriram Kosuri. Next-generation digital information storage in dna. *Science (New York, N.Y.)*, 337:1628, 08 2012. doi: 10.1126/science.1226355.

Cmglee. Comparison of 1d and 2d interpolation. Wikimedia Commons, 2019. URL https://en.wikipedia.org/wiki/Linear_interpolation#/media/File:Comparison_of_1D_and_2D_interpolation.svg. [Online; accessed 5th June 2024].

Arden Dertat. Applied deep learning - part 3: Autoencoders, 2017. URL https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798.

Seemanti Dey, Tuhin Subhra Chakraborty, Shadi Jafari, Maha V. Agha, Bilal R. Malik, Christopher Ayyub, Aneil F. Dey, Seán Corcoran, Jenny Drnevich, Edward W. Green, and Peter D. Robinson. Genetic dissection of olfactory behavior in drosophila larvae using a new, fully automated testing and analysis platform. *G3: Genes, Genomes, Genetics*, 12(11):jkab378, 2022. doi: 10.1534/g3.121.401279. URL https://doi.org/10.1534/g3.121.401279.

Melpomeni Dimopoulou. *Encoding techniques for long-term storage of digital images into synthetic DNA.* Phd thesis, Université Côte d'Azur, 2020. Image Processing [eess.IV].

Melpomeni Dimopoulou and Marc Antonini. Image storage in dna using vector quantization. In *EUSIPCO 2020*, 2021. doi: 10.23919/Eusipco47968.2020.9287470. URL https://hal.science/hal-02959330.

Melpomeni Dimopoulou, Marc Antonini, Pascal Barbry, and R. Appuswamy. A biologically constrained encoding solution for long-term storage of images onto synthetic dna. pages 1–5, 09 2019. doi: 10.23919/EUSIPCO.2019.8902583.

Melpomeni Dimopoulou, Eva Gil San Antonio, and Marc Antonini. A jpeg-based image coding solution for data storage on dna. pages 786–790, 08 2021. doi: 10.23919/EUSIPCO54536.2021.9616020.

DMLapato. Cluster generation.png. Digital image, 2015. URL https://commons.wikimedia.org/wiki/File:Cluster_Generation.png. Accessed 10th May 2024.

Emilien Dupont, Adam Golinski, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. Coin: Compressed implicit neural representations. https://arxiv.org/abs/2103.03123v2, 2021. University of Oxford.

Nick Goldman, Paul Bertone, Siyuan Chen, Christophe Dessimoz, Emily M LeProust, Botond Sipos, and Ewan Birney. Towards practical, high-capacity, low-maintenance information storage in synthesized dna. *Nature*, 494:77–80, 2013. doi: 10.1038/nature11875.

Mahmud Hasan, Kamruddin Nur, and Tanzeem Noor. Computational complexity reduction of jpeg images. *International Journal of Scientific & Technology Research*, 1, 05 2012.

Qingbo Jiang, Ran Wei, and Shiqian Wu. Features similarity index matrix (fsim) for image quality assessment. *Journal of Applied Mathematics and Physics*, 7(1), 2019. doi: 10.4236/jamp.2019.71018. URL https://www.scirp.org/journal/paperinformation.aspx?paperid=90911.

Mohammad Kazemi, Razib Iqbal, and Shervin Shirmohammadi. Joint intra and multiple description coding for packet loss resilient video transmission. *IEEE Transactions on Multimedia*, PP:1–1, 10 2017. doi: 10.1109/TMM.2017.2758578.

Navleen Khehra, Indermeet S. Padda, and Cameron J. Swift. *Polymerase Chain Reaction (PCR)*. StatPearls [Internet]. StatPearls Publishing, Treasure Island (FL), updated 2023 mar 6 edition, 2023. URL https://www.ncbi.nlm.nih.gov/books/NBK589663/. Accessed 1st April 2024.

Vicenç Laparra, Jesús Balle, and Eero P. Simoncelli. Perceptual image quality assessment using multiscale statistical models. 2016. https://www.cns.nyu.edu/pub/eero/laparra16a-preprint.pdf.

Trung Hieu Le, Xavier Pic, and Marc Antonini. Inr-mdsqc: Implicit neural representation multiple description scalar quantization for robust image coding. In *2023 25th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE, 2023. doi: 10.1109/MMSP59012.2023.10337691. I3S laboratory, Côte d'Azur University CNRS, UMR 7271, Sophia Antipolis, France.

Trung Hieu Le, Xavier Pic, Jeremy Mateos, and Marc Antonini. Implicit neural multiple description for dna-based data storage. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8000–8004, 2024. doi: 10.1109/ICASSP48485.2024.10446157.

Dixita Limbachiya, Vijay Dhameliya, Madhav Khakhar, and Manish Gupta. On optimal family of codes for archival dna storage. 09 2015. doi: 10.1109/IWSDA.2015.7458386.

Michael Neil. Neural networks and deep learning, 2015. URL http://neuralnetworksanddeeplearning.com. Last Accessed at 20th June.

Shuaijian Ni, Houzong Yao, Lili Wang, Jun Lu, Feng Jiang, Aiping Lu, and Ge Zhang. Chemical modifications of nucleic acid aptamers for therapeutic purposes. *International Journal of Molecular Sciences*, 2017. URL https://www.researchgate.net/publication/318869969_Chemical_Modifications_of_Nucleic_Acid_Aptamers_for_Therapeutic_Purposes. Accessed 1st April 2024.

Alexandre Cunha Nunes. Study and evaluation of epfl_jpeg_dna codec, June 2024. URL https://github.com/AlexandreNunes0/Study-and-evaluation-of-EPFL_JPEG_DNA-codec. Mentor: Prof. Dr. Maria Manuela Areias da Costa Pereira Sousa.

Personify. Video marketing in 2024: Trends and statistics you can't afford to ignore, 2024. URL

https://personifycorp.com/blog/video-marketing\
-in-2024-trends-and-statistics-you-cant-afford-to-ignore/.

Xavier Pic and Marc Antonini. A constrained shannon-fano entropy coder for image storage in synthetic dna. *European Signal Processing Conference (EUSIPCO)*, 03 2022.

Xavier Pic, Melpomeni Dimopoulou, Eva Gil San Antonio, and Marc Antonini. Technical description of the hidna submission to the jpeg dna cfp. ISO/IEC JTC 1/SC 29/WG 1, October 30-November 3 2023. 101th Meeting, Online.

Abhinav Ravindra. All about structural similarity index (ssim) - theory & code in pytorch. https://medium.com/srm-mic/all-about-structural-similarity-index-ssim-theory-code-in-\pytorch-6551b455541e, 2020. Accessed 1st June 2024.

Alexander F. Sandahl, Thuy J. D. Nguyen, Rikke A. Hansen, Martin B. Johansen, Troels Skrydstrup, and Kurt V. Gothelf. On-demand synthesis of phosphoramidites. *Nature Communications*, 12, 2021. doi: 10.1038/s41467-021-22945-z. URL https://www.nature.com/articles/s41467-021-22945-z.

Danaher Life Sciences. Unveiling dna ligation, 2024. URL https://lifesciences.danaher.com/us/en/library/unveiling-dna-ligation.html. Accessed: 2024-06-22.

H. R. Sheikh and A. C. Bovik. Image information and visual quality. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Montreal, Canada, August 2004.

Ming-Hwa Sheu, S M Salahuddin Morsalin, Szu-Hong Wang, Lin-Keng Wei, Shih-Chang Hsia, and Chuan-Yu Chang. Fhi-unet: Faster heterogeneous images semantic segmentation design and edge ai implementation for visible and thermal images processing. *IEEE Access*, 10:1–1, 01 2022. doi: 10.1109/ACCESS.2022.3151375.

Mike Smith. Polymerase chain reaction (pcr), 2024. URL https://www.genome.gov/genetics-glossary/Polymerase-Chain-Reaction. Accessed 10th May 2024.

Sophie Strebel, Noémie Monnier, Davi Lazzarotto, Michela Testolina, and Touradj Ebrahimi. Towards learning-based image compression for storage on DNA support. In Andrew G. Tescher and Touradj Ebrahimi, editors, *Applications of Digital Image Processing XLVI*, volume 12674, page

126740V. International Society for Optics and Photonics, SPIE, 2023. doi: 10.1117/12.2677356. URL https://doi.org/10.1117/12.2677356.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. *arXiv preprint arXiv:2012.02189*, 2020.

Taylor and Petroc. Amount of data created, consumed, and stored 2010-2020, with forecasts to 2025 (statista), 2023. URL https://www.statista.com/statistics/871513/worldwide-data-created/. Accessed 1st April 2024.

Michela Testolina, Davi Lazzarotto, and Touradj Ebrahimi. Report on cross-checking of the biocoder and hidna submissions to the jpeg dna cfp. https://www.jpeg.org, 2023. ISO-IEC JTC 1-SC 29-WG 1, wg1m101156.

Jean Vitor. Human vision image quality - psnr-hvs. https://jeanvitor.com/human-vision-image-quality-psnr-hvs/, 2023.

Karl Voelkerding, Shale Dames, and Jacob Durtschi. Next-generation sequencing: From basic research to diagnostics. *Clinical chemistry*, 55:641–58, 03 2009. doi: 10.1373/clinchem.2008.112789.

Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *Arabian Journal for Science and Engineering*, 41(2):213–229, 2016. doi: 10.1016/j.ajse.2016.03.002. URL https://www.sciencedirect.com/science/article/pii/S1319157816300088.

Zhou Wang, Eero P. Simoncelli, and Alan C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. IEEE, 2003. URL https://ece.uwaterloo.ca/~z70wang/publications/IWSSIM.pdf.

yourgenome. What is oxford nanopore technology (ont) sequencing? URL https://yourgenome.org/theme/what-is-oxford-nanopore-technology-ont-sequencing/. Accessed 10th May 2024.

YourGenome.org. What is pcr (polymerase chain reaction)?, 2024. URL https://www.yourgenome.org/theme/what-is-pcr-polymerase-chain-reaction/. Accessed: 2024-06-22.

Anne Aaron Zhang, Christos G. Bampis Han, Lei Zhi, Harini Jin, and Jay Qian Lin. Toward a practical perceptual video quality metric. https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652, 2021.