Image storage in DNA using Vector Quantization

Melpomeni Dimopoulou, Marc Antonini

Image storage in DNA using Vector Quantization

Melpomeni Dimopoulou
I3S, Université Côte d'Azur, CNRS
Sophia Antipolis, France
dimopoulou@i3s.unice.fr

Marc Antonini
I3S, Université Côte d'Azur, CNRS
Sophia Antipolis, France
am@i3s.unice.fr

***Abstract*—Rapid technological advances and the increasing use of social media has caused a tremendous increase in the generation of digital data, a fact that imposes nowadays a great challenge for the field of digital data storage due to the short-term reliability of conventional storage devices. Hard disks, flash, tape or even optical storage have a durability of 5 to 20 years while running data centers also require huge amounts of energy. An alternative to hard drives is the use of DNA, which is life's information-storage material, as a means of digital data storage. Recent works have proven that storing digital data into DNA is not only feasible but also very promising as the DNA's biological properties allow the storage of a great amount of information into an extraordinary small volume for centuries or even longer with no loss of information. In this work we present an extended end-to-end storage workflow specifically designed for the efficient storage of images onto synthetic DNA. This workflow uses a new encoding algorithm which serves the needs of image compression while also being robust to the biological errors which may corrupt the encoding.**

I. Introduction

Data explosion is one of the biggest challenges for the digital world as the development of efficient storage devices has reached a certain limit improving the storage density at a 20% every year while the equivalent data growth per year is estimated to increase by 60%. In short, we are about to have a serious data-storage problem that will only become more severe over time. Furthermore, a great percentage of the generated data is "cold" (very rarely accessed) but for legal and regulatory compliance reasons needs to be stored for a long duration (more than 50 years). Unfortunately conventional means of cold data storage such as hard disk drives or tapes have a very limited life-span from 5 to 20 years equivalently, requiring frequent maintenance and replacement of the storage devices. Some pioneer, yet very interesting works have studied the use of DNA, the genetic code of living organisms as an alternative approach for the storage of digital data. This can be achieved by encoding the digital information into a quaternary stream of A, T, C and G. One of the first attempts of encoding digital data into DNA is described in [1] by the works of Church *et* al. proposing a simple encoding of each binary bit to one nucleotide. To improve the encoding density as well as the robustness of the encoding to errors following works have adopted some more complex and efficient encoding algorithms. More precisely Goldman et al. in [2] applies a ternary huffman algorithm to compress the binary sequence into a ternary stream of three symbols (trits). Then each of the trits is encoded into a symbol from the dictionary $\{A, T, C, G\}$ each time avoiding the symbol that has been previously used. Blawat *et* al [3] propose the use of 5 nucleotides to encode 8 bits of information while also using forward error correction in the decoding to deal with the sequencing error. In the works of Grass *et* al [4], the encoding is performed using Reed Solomon codes while Bornholt *et* al in [5] have applied the same encoding as in [2] improving the encoding scheme and avoiding the fourfold redundancy which is suggested by the latter. Erlich *et* al [6] have implemented an encoding using Fountain codes to reach a high coding potential while in [7] Yazdi *et* al. proposed a new encoding method to allow re-writable random access while also introducing error correction to handle sequencing errors.

All the prior works are doing basically transcoding, i.e., converting to a quaternary code a binary sequence coming from the output of an encoder, without considering the characteristics of the input signal being encoded, nor the transcoding cost. To reduce the high DNA synthesis cost we propose an extended end-to-end encoding workflow which optimally compresses an image before it is stored into DNA, allowing control of the encoding rate and thus reducing the synthesis cost which is unfortunately relatively high (around 0.02$ per nucleotide). Another strong advantage of our proposed workflow is the use of an encoding algorithm, which has been proposed and analyticaly described in [8], which can be extended to the encoding of more than 8 bits of information and can be applied to any type of input data (binary or not).

II. Overview of the overall storage workflow

As shown in figure 1, our proposed workflow consists of 4 main parts. The first part is image compression where the data has to be compressed using some compression protocol. In this work we are using a discrete wavelet transform (DWT) and quantizing each subband of the wavelet decomposition independently using Vector Quantization (VQ). In order to optimize the compression we also use an optimal nucleotide allocation algorithm, which works similar to the bit allocation algorithms described in [9], designed to serve the needs of the DNA coding process as described in the section IV-C. This allocation serves to the choice of an optimal quantization codebook for each wavelet subband for storing the maximum possible bits in one nucleotide for a given encoding rate. Consequently, the number of DNA sequences (oligos) to be synthesized is minimized for a specified image quality. In the second part of the workflow the quantized subbands are

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 863320.

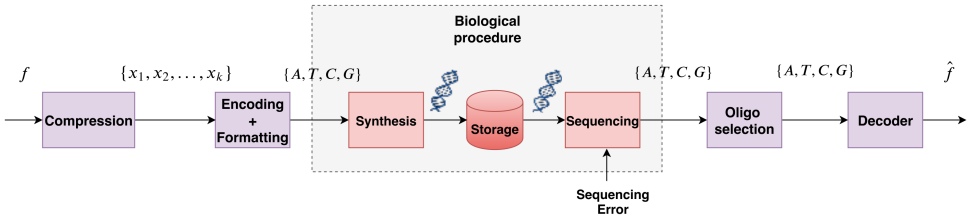

Fig. 1. Proposed workflow for storing images into DNA.

independently encoded into a quaternary code of A, T, C and G using a novel encoding algorithm which is robust to the error-prone process of DNA sequencing (reading) by respecting some constraints which will be further described in section IV-B. The encoded sequences are then cut into chunks and formatted adding headers to produce the final oligos (small pieces of DNA strands). The encoded oligos are then processed biologically in vitro so that in the third part of the process they are being synthesized into DNA and stored safely into special capsules which keep the DNA safe from corruptions and data loss for hundreds of years. DNA synthesis is generally an error-free process as long as the oligos to be synthesized are not longer than 300 nucleotides (nts). This justifies the need for cutting our oligos into smaller DNA chunks with the process of formatting. The reading of the stored data is then performed using a process which is called sequencing. The main challenge of the DNA data storage lies in the fact that this is an error prone procedure which can introduce errors in the synthesized DNA strands while reading them by sequencing. This yields that the addition of extra redundancy is necessary to protect the reliability of the decoding. This redundancy is added thanks to a biological process called Polymerase Chain Reaction (PCR) amplification before sequencing by creating many copies of the synthesized oligos. After amplification the oligos are read using a sequencing device (as for example Illumina NextSeq or Oxford Nanopore MinIon). The description of the sequencing process is out of the scope of the paper so it will not be further described. Interested readers may refer to [10]. The sequencing provides us a set of many copies of our synthesized oligos retrieved. Similarly to channel coding in which packets are sent in many copies selecting the most frequent ones as the most reliable, in the fourth step of the DNA coding workflow the oligos which are found in many copies are selected as the most representative ones. Finally, the last part of the scheme corresponds to the decoding and reconstruction of the initial image from the sequenced oligos. At this point it is important to denote that apart from the addition of the compression sub-part, this is a classical workflow for DNA data coding. Our novelty lies in the fact that, to our knowledge, this is the first work to include the compression process into the DNA storage schema so to allow control of the DNA synthesis cost by optimizing the compression parameters.

III. Our contribution

Until this point it is clear that the encoding of data into DNA is not a simple process and is constrained by many biological factors. Previous works until today have been transcoding binary information from many types of data into DNA, without taking into consideration the data characteristics. Thus, when encoding an image into DNA, the state of the art works have been using the protocol of JPEG which optimally compresses an image given some arbitrary target rate of compression expressed in bits/pixel. Then, the output bitstream of JPEG is transcoded to a quaternary stream leading to an "open loop" coding solution. However, as the DNA synthesis process is an expensive process requiring several dollars per synthesized oligo, it is fundamental to control this cost by including the compression into the process of encoding. Therefore In our approach we don't use some fixed black box of compression. We rather designed a new codec (fully controllable) embedding a quaternary encoder leading to a "closed loop" coding solution so to be able to perform an optimal source allocation algorithm that can optimally compress the image given a target rate expressed in nucleotides/pixel (or bits/nucleotide) and not in bits/pixel. Indeed, our approach doesn't perform "transcoding" from binary to quaternary (as state of the art is doing) but rather we are directly encoding using a quaternary code the quantized coefficients allowing optimal rate/distortion control. Therefore, the encoding process in DNA is considered as a main factor for the optimal compression, something which is not possible when using the standard JPEG to compress.

In [8] , we have made a first attempt to optimally compress an image into DNA by using a simple scalar quantizer. The proposed solution has been validated by a wet lab experiment, generating real synthetic oligonucleotides that were stored in a DNAShell capsule. Although very promising, our results could be improved by using some more sophisticated method of compression. Attempting to improve our previous results, we are making a first step in using Vector Quantization for DNA image coding using a nucleotide allocation algorithm for the optimization of the compression. The techniques used for the encoding are clearly described in the next sections.

IV. The proposed encoding process

A. Vector Quantization

The compression of the input image is performed using a Vector Quantization (VQ) on the coefficients of the subbands $b \in \{1, 2, \ldots, B\}$, with $B = 3i + 1$ produced by an $i-$level DWT decomposition. The advantages of using VQ instead of scalar quantization is that it can provide good compression performance while associated to a fixed length code. In our case, the choice of using a fixed length code rather than a variable length one, was motivated by the fact it is an efficient

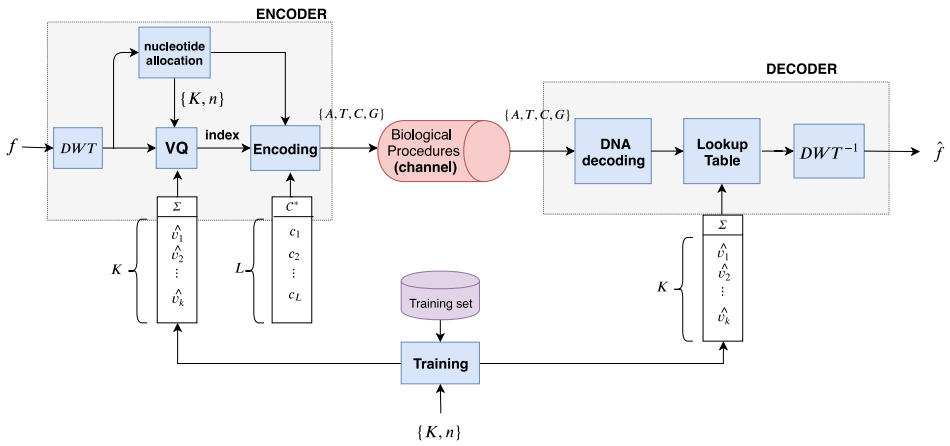

Fig. 2. Coding/decoding workflow using Vector Quantization.

way to be more robust to the sequencing noise.

The purpose of VQ is to map n-dimensional vectors in the vector space $\mathbb{R}^n$ into a finite set of vectors Σ (the codebook) where $\Sigma = \{\hat{v_j} \in \mathbb{R}^n | j = 1, 2, \ldots, K\}$. The vector set Σ contains the centroid vectors that have been computed after clustering a set of training vectors according to the generalized Lloyd algorithm presented in [11].

B. Construction of the code

The purpose of the proposed DNA coding workflow is the encoding of some input quantization vectors using a quaternary code composed by the alphabet $\{A, T, C, G\}$ to be later synthesized into DNA. Let $\Sigma = \{\hat{v_1}, \hat{v_2}, \ldots, \hat{v_K}\}$, with $|\Sigma| = K$, be a set of symbols to be encoded into a set $\mathcal{C}^* = \{c_1, c_2, \ldots, c_L\}$ of L quaternary codewords of length $l \in \mathbb{N} \setminus \{0, 1\}$. In [8], we have analytically described an efficient encoding algorithm for the encoding of digital data into quaternary words, which is robust to the sequencing errors which can occur if the encoded sequences contain homopolymer runs (same nucleotide repeated more than 3 times) and a percentage of G,C nucleotides which is higher than the percentage of A and T nucleotides. The encoding proposed in [8] proposes the creation of codewords from a set of duplets (pairs of symbols) which create an acceptable sequence when assembled in a longer strand. This means that this assembly creates no homopolymer runs and contains a percentage of G and C which is lower or equal to the percentage of A and T. More precisely, the codewords are constructed by selecting elements from the following dictionaries:

- $\mathcal{C}_1 = \{AT, AC, AG, TA, TC, TG, CA, CT, GA, GT\}$
- $\mathcal{C}_2 = \{A, T, C, G\}$

Dictionary $\mathcal{C}_1$ is composed only by pairs of symbols that when assembled in a longer strand will not create homopolymers, or a high G,C content. Consequently, the dictionary $\mathcal{C}_1$ does not contain pairs of the same symbol and the pairs GC and CG are omitted. Codewords of an even length are constructed by only selecting pair symbols from $\mathcal{C}_1$ while codewords of an odd length also require the addition of a symbol from $\mathcal{C}_2$ in their ending. The code size L that can be constructed is then given by the following relation:

$$L = \begin{cases} 10^{\frac{l}{2}}, & \text{if } l \text{ is even} \\ 10^{\frac{l-1}{2}} * 4, & \text{if } l \text{ is odd} \end{cases} \quad (1)$$

It is obvious that the code length increases at an order of $O(n)$. More specifically, in the worst case where one needs to add an extra pair of symbols to the codeword length to cover the needed size K of symbols to be encoded into quaternary, the code size L will be multiplied by 10. This code extension can be relatively big compared to the encoding needs and can possibly leave a big part of codewords unused. However, this unused part can be exploited to deal with an extra biological restriction of pattern repetitions. According to [12], the existence of repetition of short DNA sequences that can create patterns can affect the performance of the sequencer. This is a restriction that had not been taken into consideration by previous works and can occur in the case where the same symbol is repeated many times in the initial sequence. The idea is that every symbol is represented at least twice in the DNA code so that in the case of repeated occurency of the same symbol, one can alternate the two different representations to avoid repetition of short patterns that can cause a problem in the encoding. To this aim, we propose to use the remaining unused codewords of the code for a double representation of the most frequent symbols of the source in such a way so to make use of the full code length. This is an important characteristic of our encoding as in the case of image compression, while strongly quantizing the coefficients of the subbands of some sparse transform, the quantized output can contain multiple repetitions of the same value which can cause patterns in the encoded strand.

Pattern repetition problem: VQ is an efficient way for compressing images as blocks of information are being packed into quantization vectors which are then encoded to a codeword of $\mathcal{C}^*$. While this quantization is very efficient in terms of coding potential, the fact that more than one elements can be represented by the same codeword can easily cause the problem of pattern repetitions which is an ill-case for DNA

5' end of payload - ATGT ATAT ATAT ATAT ATGT ATGT GTGT GTAT ATAT ATAT GTGT ATGT ATGT ATAT ACAT ATGT GTAT ATGT - 3' end of payload
5' end of payload - TGAAG TTGAA GCATA TGATG ACTCT GATCG AGCTC GTCGG TGCTT TGACT CTGAA TAAGC CTTCT TATAG - 3' end of payload

Fig. 3. Pattern Repetitions: Example of an oligo encoded using one to one mapping (top oligo) and avoiding pattern repetition (bottom case).

coding as it can produce errors in the sequencing process. Figure 3 depicts an example of the same information encoded in an oligo using VQ with (bottom oligo) and without (top oligo) treating the case of pattern repetitions. Therefore, by applying a double representation to the most frequent vectors, the pattern repetition can be avoided. This is due to the fact that in the case of repetition of the same vector one can alternate the two associated codewords increasing this way the robustness of the encoded sequence. More precisely, we propose two different cases of mapping to deal with patterns. In the first case which is depicted in figure 4a all the vectors in Σ are mapped once to the K first codewords of $\mathcal{C}^*$.Then the p most probable vectors from Σ are also mapped to a second codeword from the $L-K$ remaining vectors of $\mathcal{C}^*$. In the second case we ensure that all vectors in Σ can be encoded by at least m different codevectors in $\mathcal{C}^*$ $(L=2K)$. In our experiments we used m=2. This second case of mapping is described in figure 4b. For further details in the encoding algorithm and the mapping process the reader can refer to [8].

C. Nucleotide allocation

This paragraph describes the source allocation algorithm, which we call nucleotide allocation and can provide the optimal quantization parameters for a given rate R_T. The purpose of the nucleotide allocation is the minimization of the total image distortion D_T such that the total rate R_T is lower than a given target rate R_{target}. In other words the goal is to find an optimal number ${K_b}^*$ as well as an optimal length ${n_b}^*$ of quantization vectors so to achieve the best possible image quality for a given compression rate. This can be mathematically described by the following relation:

$$\min_{\{K,n\}} D_T = \sum_{b=1}^{B} w_b D_b(K_b, n_b), \text{ w.r.t.}$$

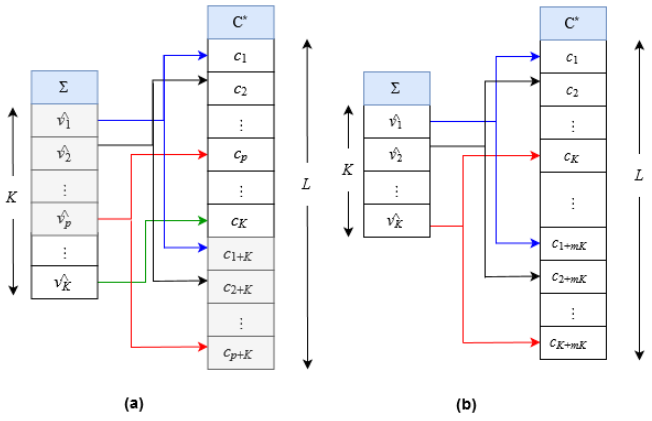

Fig. 4. Mapping of symbols to DNA words.

$$R_T = \sum_{b=1}^{B} a_b R_b(K_b, n_b) \leq R_{target}$$

where w_b is the weight of each subband b taking into consideration the non orthogonality of the DWT in case of biorthogonal transform, a_b the weight that takes into account the subband size, D_b and R_b the subband distortion and subband rate respectively, $K_b = \frac{\alpha}{100} L_b$ with $\alpha \in [0, 100]$ the number of quantization vectors (codebook size) and n_b the length of each vector (vector space dimension) for a subband b. For the minimization of the distortion in every subband we will employ the method described in [9]. This work proposes an unconstrained minimization of a function of the form $J = D(B) + \lambda R(B)$ where λ denotes the slope of the rate-distortion (R-D) curve and the minimization is over all subbands B. They noted that for any $\lambda \geq 0$, the solution $B^*(\lambda)$ to the unconstrained problem of minimizing J over all B is also the solution of the problem of minimizing $D(B)$ over all subbands B while satisfying the constraint that $R(B) \leq R(B^*(\lambda))$. Consequently, assuming that R_b is a function of n_b and K_b (and consequently of the code size L_b), and D_b is a function of R_b, our optimization problem can be written into the following minimization criterion:

$$\min_{R_b} J = \sum_{b=1}^{B} w_b D_b(R_b) + \lambda \sum_{b=1}^{B} a_b R_b.$$

For a given λ, the solution is obtained when:

$$\frac{\partial D_b(R)}{\partial R}\Big|_{R=R_b^*} = -\lambda \frac{a_b}{w_b}.$$

To get the slopes λ of the R-D curves for each subband b we exhaustively compute the rate and the distortion produced by the vector quantizer for each set of parameters (K_b, n_b). Then, from the set of produced points on the R-D curve, we select the points lying on the convex hull. As the convex hull may not be smooth enough to allow correct computation of the slopes λ we use an exponential decay approximation which can provide a fair estimation of the slope at each computed point. Figure 5 shows a typical example of the R-D curve for one subband of a 9-7 biorthogonal wavelet decomposition along with the convex hull and its exponential approximation. Finally, by selecting the points that correspond to the desired slope for every subband one can select the optimal tuning, i.e. the values of K_b^* and n_b^* for $b \in \{1, 2, \ldots, B\}$, which provides the lowest distortion D_b for a given subband rate R_b. The global R-D curve for encoding an 512x512 image of Lena [1] is depicted in figure 6.

V. Results

This last curve depicts the results of comparison of our compression scheme proposed in [8] (scalar case) and 4 different cases of VQ:

[1]The performance has been tested in other images too but due to lack of space we only present results on one image.

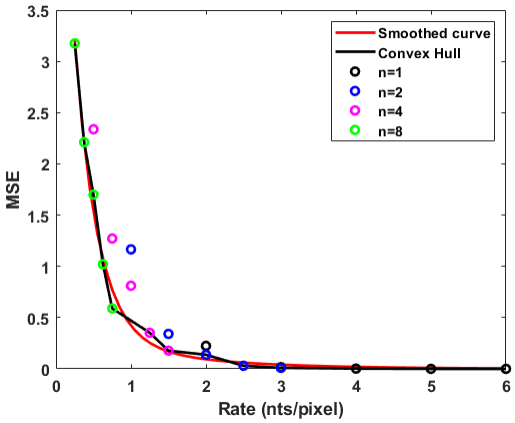

Fig. 5. Behaviour of the rate-distortion curve in one wavelet subband of Lena image. Each point corresponds to different value of K_b and n_b. The convex hull is plotted in black and its continuous exponential approximation in red.

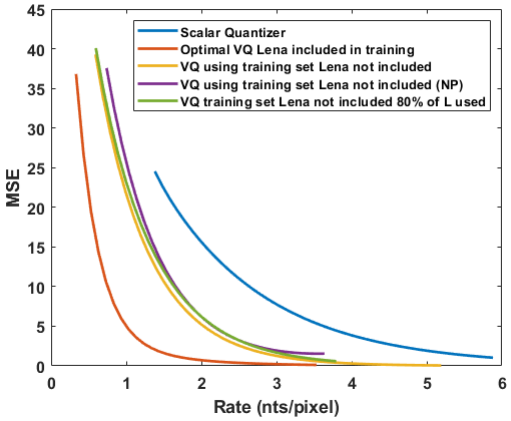

Fig. 6. Comparison of the global rate-distortion curves for 5 different cases of quantization of Lena image. Blue: scalar quantizer, Orange curve: VQ, Lena inside the training set with pattern repetition ($K_b = L_b$ for all b). Yellow: VQ, Lena outside the training set with pattern repetition. Purple: VQ, Lena outside the training set no patterns (NP) with L=2K. Green: VQ, Lena outside the training set using only 80% of the code (K_b is 80% of L_b for all b).

- Without treating patterns: Lena inside or outside the training set.
- Treating patterns (for Lena outside the training set):
  - Using only 80% of the code $\mathcal{C}^*$ for the encoding ($K = \frac{80}{100}L$) and using the 20% remaining codewords ($K - L$) to allow double representation of the most frequent quantized vectors.
  - Mapping 100% of the code ($L = 2K$).

The results show significant improvement in comparison to the previously published work.

At this point it is important to denote the fact that the oligos produced using the method proposed in this paper avoid patterns and follow a structure and statistical content very similar to the oligos produced in our previous biological experiment which was carried out with success [8]. Thus, even if there is no wet lab experiment provided in this work we are hopeful to believe that if one uses the same techniques for synthesis and sequencing as the ones used in [8] there is a very strong probability that this new experiment will be successful too. Furthermore, since this work proposes a close-loop solution of source allocation to provide an optimal compression which is specific for DNA coding, a fair comparison with classical compression protocols such as JPEG which specialise in bit/rate optimization is not possible except if transcoding is performed. A comparison of our proposed quaternary code construction to state of the art methods using transcoding along with JPEG and JPEG2000 can be found in our study in [13].

VI. Conclusions

In an attempt to improve our previous results presented in [8], in this work we have proposed an efficient workflow to be used for storing images into DNA which uses vector quantization. Our results have been very encouraging since they have shown significant improvement allowing an even better control of the compression rate which can lead to the reduction of the synthesis cost. As this is only just a first attempt to use a more sophisticated model for the compression, further experiments will allow a more complete study of this promising workflow, hoping that it can bring interesting advantages to the field of DNA data storage.

References

[1] George M Church, Yuan Gao, and Sriram Kosuri, "Next-generation digital information storage in DNA," Science, p. 1226355, 2012.

[2] 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, vol. 494, no. 7435, pp. 77, 2013.

[3] Meinolf Blawat, Klaus Gaedke, Ingo Huetter, Xiao-Ming Chen, Brian Turczyk, Samuel Inverso, Benjamin W Pruitt, and George M Church, "Forward error correction for DNA data storage," Procedia Computer Science, vol. 80, pp. 1011–1022, 2016.

[4] Robert N Grass, Reinhard Heckel, Michela Puddu, Daniela Paunescu, and Wendelin J Stark, "Robust chemical preservation of digital information on DNA in silica with error-correcting codes," Angewandte Chemie International Edition, vol. 54, no. 8, pp. 2552–2555, 2015.

[5] James Bornholt, Randolph Lopez, Douglas M Carmean, Luis Ceze, Georg Seelig, and Karin Strauss, "A DNA-based archival storage system," ACM SIGOPS Operating Systems Review, vol. 50, no. 2, pp. 637–649, 2016.

[6] Yaniv Erlich and Dina Zielinski, "DNA fountain enables a robust and efficient storage architecture," Science, vol. 355, no. 6328, pp. 950–954, 2017.

[7] SM Hossein Tabatabaei Yazdi, Ryan Gabrys, and Olgica Milenkovic, "Portable and error-free dna-based data storage," Scientific reports, vol. 7, no. 1, pp. 1–6, 2017.

[8] Melpomeni Dimopoulou, Marc Antonini, Pascal Barbry, and Raja Appuswamy, "A biologically constrained encoding solution for long-term storage of images onto synthetic DNA," in EUSIPCO, 2019.

[9] Yair Shoham and Allen Gersho, "Efficient bit allocation for an arbitrary set of quantizers (speech coding)," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 36, no. 9, pp. 1445–1453, 1988.

[10] I Illumina, "An introduction to next-generation sequencing technology," 2015.

[11] Yoseph Linde, Andres Buzo, and Robert Gray, "An algorithm for vector quantizer design," IEEE Transactions on communications, vol. 28, no. 1, pp. 84–95, 1980.

[12] Todd J Treangen and Steven L Salzberg, "Repetitive DNA and next-generation sequencing: computational challenges and solutions," Nature Reviews Genetics, vol. 13, no. 1, pp. 36, 2012.

[13] Melpomeni Dimopoulou, Marc Antonini, Pascal Barbry, and Raja Appuswamy, "Storing digital data into dna: A comparative study of quaternary code construction," in ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2020, pp. 4332–4336.