COMBINING PROGRESSIVE IMAGE COMPRESSION AND RANDOM ACCESS IN DNA DATA STORAGE

Xavier Pic⋆, Raja Appuswamy⋆

⋆ EURECOM, Data Science Department, Sophia Antipolis, France

ABSTRACT

The exponential increase in storage demand and low lifespan of data storage devices has resulted in long-term archival and preservation emerging as a critical bottlenecks in data storage. In order to meet this demand, researchers are now investigating novel forms of data storage media. The high density, long lifespan and low energy needs of synthetic DNA make it a promising candidate for long-term data archival. However, current DNA data storage technologies are facing challenges with respect to cost (writing data to DNA is expensive) and reliability (reading and writing data is error prone). Thus, data compression and error correction are crucial to scale DNA storage. Additionally, the DNA molecules encoding several files are very often stored in the same place, called an oligo pool. For this reason, without random access solutions, it is relatively impractical to decode a specific file from the pool, because all the oligos from all the files need to first be sequenced, which greatly deteriorates the read cost.

This paper introduces PIC-DNA–a novel JPEG2000-based progressive image coder adapted to DNA data storage. This coder directly includes a random access process in its coding system, allowing for the retrieval of a specific image from a pool of oligos encoding several images. The progressive decoder can dynamically adapt the read cost according to the user's cost and quality constraints at decoding time. Both the random access and progressive decoding greatly improve on the read-cost of image coders adapted to DNA.

Index Terms— DNA data storage, JPEG 2000, progressive, random access, JPEG DNA VM

1. INTRODUCTION

The ever-increasing demand for data storage poses unprecedented challenges for data storage providers, as they have to store increasingly large-amounts of data over very long periods of time while maintaining retrieval guaranties. Conventional storage devices, have fundamental durability and density limitations that make long-term data storage infeasible. As a result, novel data storage solutions, such as synthetic DNA molecules, have emerged as an alternative for overcoming these challenges [1], mainly due to their long lifespan, high density and low energy needs.

A classic solution for storing data onto DNA molecules is composed of both biochemical and computational processes. The two main biochemical processes are synthesis, to create DNA molecules with the desired sequences of nucleotides, and sequencing, to retrieve the sequence of nucleotides that it contains. The main computational process is the DNA-adapted codec, that transforms data into quaternary sequences for synthesis, and back during sequencing. The codec also compresses the data to reduce synthesis and sequencing cost.

Modern data storage systems that are used for housing large image collections also support efficient random access that enables retrieval of a subset of images. Such random access is used extensively by modern-day web applications to support novel features like adaptive selection of image resolution based on the client (desktop or mobile) used to view the image. Unfortunately, DNA does not directly support such random access mechanisms. Prior work on DNA data storage has demonstrated the use of Polymerase Chain Reaction (PCR) for enabling random access. PCR requires the use of primers, short sequences of nucleotides attached to DNA strands, that can be used to biochemically select, copy and sequence only the oligos that include this specific primer pair. By attaching different primers with different oligos, prior work has demonstrated the ability to achieve random access on DNA[**?**]. However, such random access has been applied to arbitrary binary files and has not been used into the context of image storage for features like adaptive selection of resolution based on progressive image coding.

In this paper, we present Progressive Image Compression for DNA storage (PIC-DNA)–a novel codec that integrates random access at the image coding level, to enable new access paths over image collections stored in DNA and reduce the reading cost. Our solution is based on the JPEG 2000 [2] progressive coder, but can easily be adapted to any progressive coder that separates the encoded bitstream into resolution or quality layers. PIC-DNA encodes and stores each quality layer separately, using primers to enable selective amplification of specific layers. By doing so, PIC-DNA enables several access modalities. First, it enables the use of thumbnails encoded in DNA as an interface between the user and the oligo memory; through the visual selection of the thumbnail, the user can select the image to be decoded, and with the help of the oligos encoding the selected thumbnail, the rest of the

image data can be retrieved. Second, it enables each image retrieval from DNA storage to be customized based on the desired resolution. For instance, applications that need only a low quality version of the image can use primers tagged for lower resolution to selectively amplify only a few layers. This results in a lower cost of sequencing, as fewer oligos need to be sequenced, and lower computational complexity of decoding, as fewer reads need to be processed. To our knowledge, the solution presented in this paper is the first DNA image coder that directly includes a random access mechanism.

2. CONTEXT

2.1. DNA-adapted coding

The field of DNA data storage has emerged as a very active research field over the past decade. In this section, we provide an overview of a few pioneering approaches and refer the reader to a recent survey [3] for a detailed comparison of various approaches that focus on storing generic, binary data using DNA. In 2012, Church et al. [1], introduced an approach to enable large-scale encoding and decoding of data into synthetic DNA molecules. This work also identified some of the constraints that need to be respected when coding data for synthesized DNA molecules. Later, in 2013, Goldman et al.[4] provided one of the first encoders capable of respecting some of these constraints. The contribution of that work, an entropy coder, allowed for the encoding of any file into DNA-like data. In 2015, Grass et al.[5] introduced the the first error correction codes into a DNA data storage solution. This error correction mechanism makes the whole storage process robust against the different biochemical operations that often introduce errors (substitutions, insertions, deletions) in the encoded DNA-like data. Following this, other works introduced other error correction solutions [6, 7, 8, 9, 10, 11, 12] with the focus of adding redundancy to binary data to detect and correct errors. Finally, random access solutions aiming to enable selective access to binary data stored in DNA and improve read cost have also been investigated [13, 14].

2.2. DNA-adapted image coding

While the solutions described above focused on generic binary data storage, several solutions have also emerged to customize the encoding and storage of images specifically on DNA. For instance, Dimopoulou et al.[15] developed a JPEG-based image coder adapted to DNA data storage that uses the Goldman encoder to encode runcat values into DNA. Similarly, in [16], the authors introduce a DNA-adapted coding scheme based on JPEG. Pic el al. used an improved DNA-adapted entropy coder[17] to increase the performance of this JPEG-based DNA-adapted image coder. Lazzarotto et al.[18] developed the JPEG DNA VM software that encodes data with a system based on Raptor codes[19]. It was output by the JPEG DNA ad-hoc group as a verification model for later developments. Learning-based image compression methods have also been studied [20, 21], where the authors use a variational autoencoder to compress the image into a latent space

that is later encoded into DNA. In [22], the authors leverage pixel domain representation to reconstruct the images with better compression performance.

3. PROPOSED METHOD

In this section, we introduce the PIC-DNA codec which adapts progressive image compression to DNA data storage.

3.1. Coding solution and oligo organization

Data compression. The general encoding process (Fig. 2a) relies on the binary progressive encoding process of JPEG2000. The image is encoded, with a fixed number of resolution layers N_{levels}. The layer with the smallest rate is used as thumbnail. The bitstream is cut into binary files, one for each resolution layer. Each layer file is encoded into DNA with a DNA coder, in our case, the JPEG DNA VM coder.

Random access: When data is stored into DNA, several files are stored in the same pool. Reading all the files is very costly, so directly accessing the oligos specific to the desired file is crucial. Further, in order to enable progressive decoding, it is also necessary to be able to access each layer separately. For this, PIC-DNA uses two pairs of primers, concatenated at each end of each oligo (Fig. 1). The first pair of primers (image primers) $(P_{I,i,l}, P_{I,i,r}), i \in [\![0, N_{images} - 1]\!]$ are used to identify each image uniquely. The second pair of primers (layer primers) $(P_{L,k,l}, P_{L,k,r}), k \in [\![0, N_{levels} - 1]\!]$ is used to identify each layer uniquely.

Further, in order enable the thumbnail-based image search functionality, PIC-DNA encodes the thumbnails of all images in a separate collection. These thumbnail oligos follow a similar structure to other data oligos (Fig. 1). However, PIC-DNA uses a specific pair of primers $(P_{L,0,l}, P_{L,0,r})$ in place of layer primers to identify the oligos as thumbnail oligos. Additionally, PIC-DNA uses the same image primer pair for image oligos and thumbnail oligos corresponding to the same image. Thus, in the image primer positions $(P_{I,i,l}, P_{I,i,r}), i \in [\![0, N_{images} - 1]\!]$ of the thumbnail oligos, PIC-DNA stores the primers corresponding to the thumbnail's parent image.

3.2. Decoding process

The introduced decoding process can be decomposed into three separate subprocesses: (i) thumbnail extraction, (ii) secondary primer identification, and (iii) decoding.

1. **Thumbnails extraction** (Fig. 2b): With the help of the pair of primers, $(P_{L,0,l}, P_{L,0,r})$ associated to all the thumbnails, the oligos related to all the thumbnails in the pool are selected. They are then classified image by image, with the help of the secondary image primers, used here as an offset code. The DNA decoder then decodes all the thumbnail oligos into compressed binary files, that are then decoded with the image codec.

2. **Secondary primers identification** (Fig. 2c): A visual inspection of all the thumbnails allows for the selection of the desired image. Once this image has been identified, the related thumbnail oligos are inspected, and the secondary
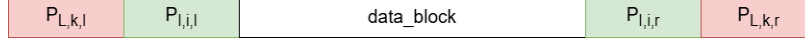
| $P_{L,k,l}$ | $P_{I,i,l}$ | data_block | $P_{I,i,r}$ | $P_{L,k,r}$ |

Fig. 1. Organization of an oligo in our solution : the pair of primers $(P_{L,k,l}, P_{L,k,r})$ describes the resolution level to which the oligo belongs, and the pair of primers $(P_{I,i,l}, P_{I,i,r})$ describes the image the oligo is related to.
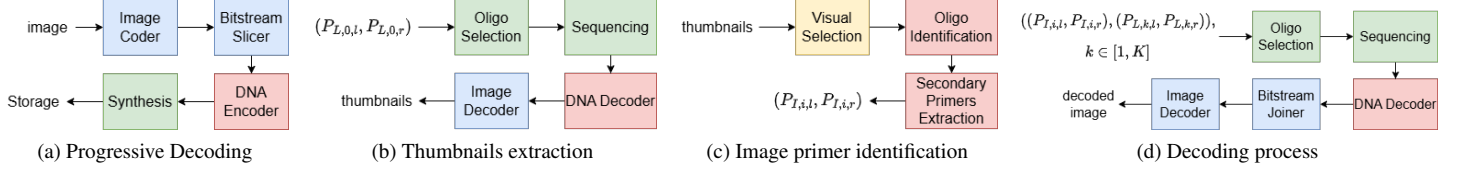

(a) Progressive Decoding   (b) Thumbnails extraction   (c) Image primer identification   (d) Decoding process

Fig. 2. Different components of the general Progressive encoding and decoding workflow

image primers $(P_{I,i,l}, P_{I,i,r})$ related to the image (Fig. 1) can be extracted.

3. **Image decoding** (Fig. 2d): To decode an image at a certain quality or resolution, the user has to decode all the layers until the chosen resolution layer is reached. With PIC-DNA, this has to be done in several stages. First, the user has to run a PCR round to select and augment oligos related to the desired image. Then multiple rounds of PCR can be performed using this amplified pool to isolate further and augment the oligos of each layer until the target resolution is reached. More specifically, once the primers corresponding to the image have been identified from the previous stage, with the help of these primers $(P_{I,i,l}, P_{I,i,r})$ and the pairs of primers related to the different levels to be decoded $\{(P_{L,k,l}, P_{L,k,r}), k \in [\![0, K_{levels\_to\_decode} - 1]\!]\}$, the necessary oligos for decoding the image can now be retrieved. It is important to note that this decoding process, like what is being done in classic binary progressive decoders, can be done iteratively, until a satisfactory version of the image is retrieved. The progressive decoder provides to the end user a trade off between the read-cost (closely linked to the sequencing price and duration), and the reconstruction quality. Once the image is decoded, this trade-off can even be re-evaluated, to decode additional resolution layers.

3.3. Read Cost

Under these conditions, the general reading cost necessary to retrieve a specific image I until the resolution level K can be evaluated in three ways, without random access or progressive decoding (Eq. 1), with just progressive decoding (Eq. 2), with both progressive decoding and random access (Eq. 3):

$$R_c(I, K) = \frac{\sum_{i=0}^{N_{images}} \sum_{k=0}^{N_{levels}} nucs(i,k)}{input\_image\_pixels} \quad (1)$$

$$R_{c\_pd}(I, K) = \frac{\sum_{i=0}^{N_{images}} \sum_{k=0}^{K} nucs(i,k)}{input\_image\_pixels} \quad (2)$$

$$R_{c\_ra}(I, K) = \frac{\sum_{i=0}^{N_{images}} nucs(i,0) + \sum_{k=1}^{K} nucs(I,k)}{input\_image\_pixels} \quad (3)$$

The $nucs(i,k)$ value represents the number of nucleotides to sequence to be able to decode a layer:

$$nucs(i,k) = coverage(i,k) \times number\_oligos(i,k) \quad (4)$$

With this, we can define a read-cost gain as in two different ways, depending on whether or not Random Access is enabled (in both cases, Progressive decoding is used) as:

$$G_{pd}(I, K) = \frac{R_c(I, K)}{R_{c\_pd}(I, K)} \quad (5)$$

$$G_{ra}(I, K) = \frac{R_c(I, K)}{R_{c\_ra}(I, K)} \quad (6)$$

4. EXPERIMENTAL RESULTS

4.1. Software implementation details

Our PIC-DNA implementation is currently utilizing the OpenJPEG implementation of the JPEG2000 codec, in its progressive coding configuration. In the version 1.5 of the OpenJPEG software, the `image_to_j2k` executable can output a `.Idx` file, that gives all the information necessary to localize the resolution layers in the general output bitstream. The bitstream is cut at the beginning of every new layer.

4.2. Performance evaluation

The performance of any DNA encoding method can be evaluated with respect to a series of metrics such as RD-curves, reading cost and writing cost. As our work primarily focuses on progressive compression and random access, the main metric we focus on in our study is the reading cost. We especially study the evolution of the reading cost for a given encoded image, as we read through each resolution layer. The image is encoded into a series of resolutions layers, the smallest one being the thumbnail. The resolution layers each divide the size of the image by a factor of 2 in each dimension (4 in total). The results presented here were obtained by encoding the images of the kodak[1] dataset, with 5 resolution levels. The oligos had data blocks of length 200 to which the primers were added. A theoretical read-cost gain was observed (lines 3 and 7 of Table 1), for a given layer, setting the coverage to 1 in the gain metrics previously defined. A noise simulator was used to introduce substitutions, deletions and insertions. Erroneous data was then passed through a clustering and consensus process to estimate a minimal coverage for decodability. Table 1 depicts the read cost gains obtained by enabling progressive decoding and random access.

The top part of Table 1 represents the gains obtained when only progressive decoding is enabled. In this situation, we

---

[1]https://r0k.us/graphics/kodak/

decode all images until a certain resolution level, without any random access. In these conditions, the progressive decoder provides gains of up to $60\times$, when only the initial layer is targeted. This gain quickly decreases if more layers are targeted, and if all the layers are targeted, no gain can be leveraged from progressive decoding, because all oligos need to be sequenced. The gains, though, depend on the dimensions of the chosen resolution layers: smaller thumbnails or layers will leverage better gains, at the cost of more distorted images. The bottom part of Table 1 represents the gains obtained when both progressive decoding and random access are enabled. In these gains, the cost necessary to retrieve the thumbnail is included, as described in the second member of the sum in Equation (3). The random access process further improves the read cost, especially in layers with better resolution where the gain is multiple times larger than the one measured without random access. This gain highly depends on the number of images encoded in the pool of DNA molecules, and on the size of the different resolution layers.

Additionally, Figures 3a and 3b depict the evolution of the reconstruction quality (PSNR here) of the image, as a function of the read cost (which depends on the resolution layers that are selected). With the reduction factors that we used in the progressive coding parameters, it is possible to decode a degraded version of the image for a fraction of the read-cost necessary to decode the whole picture. For instance, the thumbnail shown in Figure 4a has a PSNR of 18.3dB[2](to compute the PSNR, the reduced image was resized to the original image's dimensions with a bi-cubic interpolation). The last resolution level (Fig. 4b), in contrast, shows a PSNR of 52dB. Further, the thumbnail is of good enough quality to be used as a visual reference in the random access process (Fig. 2c). Moreover, the gains we obtain here are orthogonal to the improvements in coding performance and read cost of the different resolution layers that can be achieved by tweaking the encoding options of JPEG2000 (specifically the targeted quality) to further degrade the image. Finally, the JPEG DNA VM software parameters can be adjusted differently for each resolution layer, especially the redundancy, so that lower resolution layers are better protected against errors.

5. CONCLUSION AND PERSPECTIVES

In this paper, we introduced PIC-DNA–a novel image compression process adapted to DNA data storage that enable new access paths and reduction in read costs. More specifically, PIC-DNA provides a random access solution based on thumbnails that are encoded in specific oligos of the whole oligo pool where data is stored. It additionally provides a trade-off between read-cost and reconstruction quality, by utilizing the progressive functionality of the JPEG2000 image coder. This last functionality can be implemented in DNA storage with any image coders that includes a progressive decoder.

---

[2]Additional data can be found here:
https://gitlab.eurecom.fr/pic/jp2dnaprogressiveres

	Layer	L_0	L_1	L_2	L_3	L_4
Progressive Decoding	# Oligos	2878	8539	25288	69358	151958
	Theoretical G_{pd} Read-cost gain	52.8	17.8	6.01	2.19	1
	Coverage	1.67	2	1.96	1.88	1.5
	Observed G_{pd} Read-cost gain	**52.5**	**15.6**	**5.13**	**1.93**	**1**
Random Access & Progressive Decoding	# Oligos	2878	3114	3812	5648	9090
	Theoretical G_{ra} Read-cost gain	52.8	48.8	39.9	27.1	17.0
	Coverage	1.67	2	1.96	1.88	1.5
	Observed G_{ra} Read-cost gain	**52.5**	**47.8**	**38.4**	**26.5**	**17.8**

Table 1. Theoretical and observed average read-cost gains G_{pd} and G_{ra} for each target resolution level, averaged over all the images of the kodak dataset. The level L_0 refers to the thumbnail while the level L_4 refers to the full image.

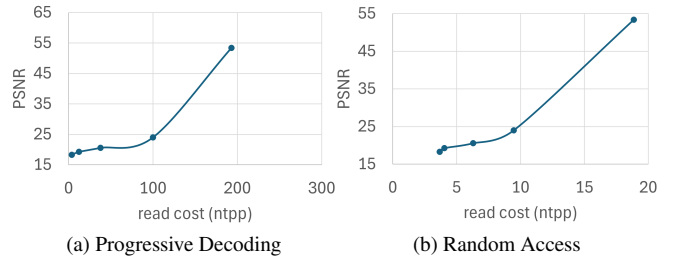(a) Progressive Decoding (b) Random Access

Fig. 3. Evolution of the distortion in function of the read-cost through the different resolution layers of the encoded image, with and without Random Access enabled.

There are several avenues of future work that can build on our work. First, PIC-DNA could be used to experiment with other progressive decoders such as JPEG XL. AI-based image coders that include a hierarchical representation of the data [23] could be considered as solutions to separate the data into sets of oligos of different layers. Second, this work can be extended to include progressive storage of other unstructured data like video or audio. Third, in this work we focused primarily on the rate gains for each resolution layer while ignoring the coding rate of the codec. The JPEG DNA VM was made to be very redundant to simplify clustering and consensus algorithms. Further work could try to tackle this issue and make the codec more competitive in terms of coding rate as well. Fourth, progressive decoding performed by PIC-DNA is well-aligned with some new APIs provided by modern sequencers. For instance, Nanopore devices offer a ReadUntil API that can be used to selectively sequence only specific DNA strands that match a template. By using layer primers as templates, PIC-DNA can be extended to perform even PCR-free, on-the-fly, adaptive selection of resolution.
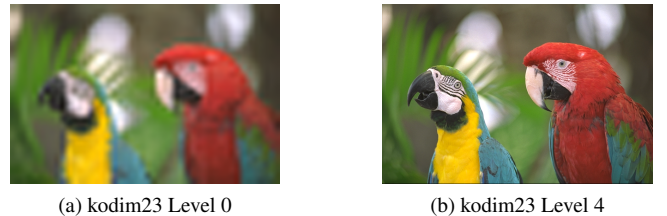

(a) kodim23 Level 0 (b) kodim23 Level 4

Fig. 4. Reconstruction of 2 different resolution levels of the kodim23 image.

6. REFERENCES

[1] George M. Church, Yuan Gao, and Sriram Kosuri, "Next-generation digital information storage in dna," Science, vol. 337, no. 6102, pp. 1628–1628, 2012.

[2] David S. Taubman and Michael W. Marcellin, "Jpeg2000 image compression fundamentals, standards and practice," Springer.

[3] Thomas Heinis, Roman Sokolovskii, and Jamie J. Alnasir, "Survey of information encoding techniques for dna," ACM Comput. Surv., vol. 56, no. 4, Nov. 2023.

[4] N. Goldman, P. Bertone, and S. Chen, "Towards practical, high-capacity, low-maintenance information storage in synthesized dna," Nature, 2013.

[5] 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, 2015.

[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] S. M. 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] Marius Welzel, Peter Michael Schwarz, Hannah F. Löchel, Tolganay Kabdullayeva, Sandra Clemens, Anke Becker, Bernd Freisleben, and Dominik Heider, "Dna-aeon provides flexible arithmetic coding for constraint adherence and error correction in dna storage," Nature Communications, 2023.

[9] 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.

[10] Ben Cao, Bin Wang, and Qiang Zhang, "Gcnsa: Dna storage encoding with a graph convolutional network and self-attention," iScience, vol. 26, no. 3, pp. 106231, 2023.

[11] Ben Cao, Xiaokang Zhang, Shuang Cui, and Qian Zhang, "Adaptive coding for dna storage with high storage density and low coverage," NPJ Systems Biology and Applications, vol. 8, 2022.

[12] Omer Sabary, Alexander Yucovich, Guy Shapira, and Eitan Yaakobi, "Reconstruction algorithms for dna-storage systems," Scientific Reports, vol. 14, 2024.

[13] L. Organick, S. Ang, and YJ. et al. Chen, "Random access in large-scale dna data storage," Nature Biotechnology, 2018.

[14] Eugenio Marinelli, Yiqing Yan, Lorenzo Tattini, Virginie Magnone, Pascal Barbry, and Raja Appuswamy, "Cmoss: A reliable, motif-based columnar molecular storage system," in Proceedings of the 17th ACM International Systems and Storage Conference, New York, NY, USA, 2024, SYSTOR '24, p. 178–192, Association for Computing Machinery.

[15] M. Dimopoulou, E. G. San Antonio, and M Antonini, "A JPEG-based image coding solution for data storage on DNA," EUSIPCO, 2021.

[16] Bingzhe Li, Li Ou, and David Du, "Img-dna: approximate dna storage for images," in Proceedings of the 14th ACM International Conference on Systems and Storage, New York, NY, USA, 2021, SYSTOR '21, Association for Computing Machinery.

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

[18] Davi Lazzarotto, Jorge Encinas Ramos, Michela Testolina, and Touradj Ebrahimi, "Technical description of the epfl submission to the jpeg dna cfp," arXiv preprint, 2023.

[19] Amin Shokrollahi, "Raptor codes," IEEE Transactions on Information Theory, vol. 52, pp. 2551 – 2567, 06 2006.

[20] Yanfen Zheng, Ben Cao, Xiaokang Zhang, Shuang Cui, Bin Wang, and Qian Zhang, "Dna-qlc: an efficient and reliable image encoding scheme for dna storage," BMC Genomics, vol. 25, 2024.

[21] Giulio Franzese, Yiqing Yan, Giuseppe Serra, Ivan D'Onofrio, Raja Appuswamy, and Pietro Michiardi, "Generative dna: Representation learning for dna-based approximate image storage," in 2021 International Conference on Visual Communications and Image Processing (VCIP), 2021, pp. 01–05.

[22] Seongjun Seo, Anshula Tandon, Keun Woo Lee, Lee Jee-Hyong, and Sung Ha Park, "Information density enhancement using lossy compression in dna data storage.," 2024.

[23] Jooyoung Lee, Se Yoon Jeong, and Munchurl Kim, "Deephq: Learned hierarchical quantizer for progressive deep image coding," 2024.