Disturbance-based Discretization, Differentiable IDS Channel, and an IDS-Correcting Code for DNA-based Storage

Alan J.X. Guo[1,2*], **Mengyi Wei**[1], **Yufan Dai**[1], **Yali Wei**[1], **Pengchen Zhang**[1]

[1]Center for Applied Mathematics, Tianjin University, Tianjin 300072, China
[2]State Key Laboratory of Synthetic Biology, Tianjin University, Tianjin 300072, China
{jiaxiang.guo, mengyi.wei, daiyufan, yaliwei222__, zhangpengchen}@tju.edu.cn

Abstract

With recent advancements in next-generation data storage, especially in biological molecule-based storage, insertion, deletion, and substitution (IDS) error-correcting codes have garnered increased attention. However, a universal method for designing tailored IDS-correcting codes across varying channel settings remains underexplored. We present an autoencoder-based approach, THEA-code, aimed at efficiently generating IDS-correcting codes for complex IDS channels. In the work, a disturbance-based discretization is proposed to discretize the features of the autoencoder, and a simulated differentiable IDS channel is developed as a differentiable alternative for IDS operations. These innovations facilitate the successful convergence of the autoencoder, producing channel-customized IDS-correcting codes that demonstrate commendable performance across complex IDS channels, particularly in realistic DNA-based storage channels.

1 Introduction

Biological molecule-based storage, a method that uses the synthesis and sequencing of biological molecules for information storage and retrieval, has attracted significant attention (Church, Gao, and Kosuri 2012; Goldman et al. 2013; Grass et al. 2015; Erlich and Zielinski 2017; Organick et al. 2018; Dong et al. 2020; Chen et al. 2021; El-Shaikh et al. 2022; Welzel et al. 2023). Currently, most applications in this field are focused on DNA-based information storage (Meiser et al. 2022).

Due to the involvement of biochemical procedures, the storage pipeline can be viewed as an insertions, deletions, or substitutions (IDS) channel (Blawat et al. 2016) over 4-ary sequences with the alphabet $\{\mathrm{A}, \mathrm{T}, \mathrm{G}, \mathrm{C}\}$. Consequently, an IDS-correcting encoding/decoding method plays a key role in biological molecule-based storage.

However, despite the existence of excellent combinatorial IDS-correcting codes (Varshamov and Tenenholtz 1965; Levenshtein 1965; Sloane 2000; Mitzenmacher 2009; Cai et al. 2021; Gabrys et al. 2023; Bar-Lev, Etzion, and Yaakobi 2023), applying them in DNA-based storage remains challenging. The biochemical channel in DNA-based storage is more complex than those studied in previous works, with factors such as inhomogeneous error probabilities across error types, base indices, and even sequence patterns (Hirao et al. 1992; Press et al. 2020; Blawat et al. 2016; Cai et al. 2021; Hamoum et al. 2021). Additionally, most of the aforementioned combinatorial codes focus on correcting either a single error or a burst of errors, whereas multiple independent errors within the same DNA sequence are common in DNA-based storage. To address this, an outer code is usually employed to correct residual errors that are beyond the capability of the inner IDS code.

Given the complexity of the IDS channel, we leverage the universality of deep learning methods by employing an autoencoder (Baldi 2012) as the foundation for an end-to-end IDS-correcting code. This approach enables researchers to train customized codes tailored to various IDS channels through a unified training procedure, rather than manually designing specific combinatorial codes for each IDS channel setting, many of which remain unexplored.

To realize this approach, two novel techniques are developed, which we believe offer greater contributions to the communities than the code itself.

Firstly, the discretization effect of applying disturbance in a non-generative model is investigated in this work. It is observed that introducing disturbance to the logistic feature forces the non-generative model to reduce the disturbance caused indeterminacy by producing more confident logits, thereby achieving discretization. This aligns with the discrete codewords of an error-correcting code (ECC) in this work, and provides an alternative approach for bridging the gap between continuous models and discrete applications.

Secondly, a differentiable IDS channel using a Transformer-based model (Vaswani et al. 2017) is developed. The non-differentiable nature of IDS operations presents a key challenge for deploying deep learning models that rely on gradient descent training. To tackle this, a model is trained in advance to mimic the IDS operations according to a given error profile. It can serve as a plug-in module for the IDS channel and is backpropagable within the network. This differentiable IDS channel has the potential to act as a general module for addressing IDS or DNA-related problems using deep learning methods. For instance, researchers could build generative models on this module to simulate the biochemical processes involved in

*Corresponding author.

manipulating biosequences.

Overall, this work implements a heuristic end-to-end autoencoder as an IDS-correcting code, referred to as THEA-Code. The encoder maps the source DNA sequence into a longer codeword sequence. After introducing IDS errors to the codeword, a decoder network is employed to reconstruct the original source sequence from the codeword. During the training of this autoencoder, disturbance-based discretization is applied to the codeword sequence to produce one-hot-like vectors, and the differentiable IDS channel serves as a substitute for conventional IDS channel, enabling gradient backpropagation.

To the best of our knowledge, this work presents the first end-to-end autoencoder solution for an IDS-correcting code. It introduces the disturbance-based discretization, and proposes the first differentiable IDS channel. It is also the first universal method for designing tailored IDS-correcting codes across varying channel settings. Experiments across multiple complex IDS channels, particularly in the realistic DNA-based storage channel, demonstrate the effectiveness of the proposed THEA-Code.

2 Related Works

Many established IDS-correcting codes are rooted in the Varshamov-Tenengolts (VT) code (Varshamov and Tenengoltz 1965; Levenshtein 1965), including (Calabi and Hartnett 1969; Tanaka and Kasai 1976; Sloane 2000; Cai et al. 2021; Gabrys et al. 2023). These codes often rely on rigorous mathematical deduction and provide firm proofs for their coding schemes. However, the stringent hypotheses they use tend to restrict their practical applications. Heuristic IDS-correcting codes for DNA-based storage, such as those proposed in (Pfister and Tal 2021; Yan, Liang, and Wu 2022; Maarouf et al. 2022; Welzel et al. 2023), usually incorporate synchronization markers (Sellers 1962; Srinivasavaradhan et al. 2021; Haeupler and Shahrasbi 2021), watermarks (Davey and Mackay 2001), or positional information (Press et al. 2020) within their encoded sequences. Recently, directly correcting errors in retrieved DNA reads without sequence reconstruction has been investigated, demonstrating promising performance (Welter et al. 2024).

In recent years, deep learning methods have found increasing applications in coding theory (Ibnkahla 2000; Simeone 2018; Akrout et al. 2023; Park et al. 2025). Several architectures have been employed as decoders or submodules of conventional codes on the additive white Gaussian noise (AWGN) channel. In (Cammerer et al. 2017), the authors applied neural networks to replace sub-blocks in the conventional iterative decoding algorithm for polar codes. Recurrent neural networks (RNN) were used for decoding convolutional and turbo codes (Kim et al. 2018). Both RNNs and Transformer-based models have served as belief propagation decoders for linear codes (Nachmani et al. 2018; Choukroun and Wolf 2022, 2023, 2024a,b,c). Hypergraph networks were also utilized as decoders for block codes in (Nachmani and Wolf 2019). Despite these advancements, end-to-end deep learning solutions remain relatively less explored. As mentioned in (Jiang et al. 2019), direct applications of multi-layer perceptron (MLP) and convolutional neural network (CNN) are not comparable to conventional methods. To address this, the authors in (Jiang et al. 2019) used deep models to replace sub-modules of a turbo code skeleton, and trained an end-to-end encoder-decoder model. Similarly, in (Makkuva et al. 2021), neural networks were employed to replace the Plotkin mapping for the Reed-Muller code. Both of these works inherit frameworks from conventional codes and utilize neural networks as replacements for key modules. In (Balevi and Andrews 2020), researchers proposed an autoencoder-based inner code with one-bit quantization for the AWGN channel. Confronting challenges arising from quantization, they utilized interleaved training on the encoder and decoder.

3 Disturbance-based Discretization

In this work, it is observed that introducing disturbance to the categorical distribution feature produced by a non-generative model causes the feature to resemble a one-hot vector.

Intuitively, the non-generative model may attempt to reduce the indeterminacy introduced by the disturbance by generating more confident logits. When a logit x is perturbed by a noise term ϵ before producing the categorical distribution, a fully converged model, aiming to generate outputs with high certainty, have to produce logits x with significantly larger magnitudes to diminishing the relative proportion of the disturbance ϵ. From this perspective, the logit x becomes more confident, producing probabilities that are closer to one-hot vectors and exhibit lower entropy. This effect is confirmed by monitoring the entropy of the categorical distribution in the experiments presented in Appendix C.1.

Let $\boldsymbol{x}$ be the logits that produce the probabilities $\boldsymbol{\pi} = \{\pi_1, \pi_2, \ldots, \pi_k\}$ via the softmax function,

$$\pi_i = \frac{\exp x_i}{\sum_{j=1}^{k} \exp x_j}, \quad i = 1, 2, \ldots, k. \tag{1}$$

In this work, the non-generative disturbance is introduced to $\boldsymbol{\pi}$ by sampling from the Gumbel distribution (Gumbel 1935). It follows the same formula as the Gumbel-Softmax, which has been widely used in generative models for generating samples (Jang, Gu, and Poole 2017; Maddison, Mnih, and Teh 2017). Specifically, the non-generative disturbance is applied to $\boldsymbol{x}$ using the following formula:

$$\mathrm{GS}(\boldsymbol{x})_i = \frac{\exp\left((x_i + g_i)/\tau\right)}{\sum_{j=1}^{k} \exp\left((x_j + g_j)/\tau\right)}, \quad i = 1, 2, \ldots, k, \tag{2}$$

where $g_1, g_2, \ldots, g_k$ are i.i.d. samples drawn from the Gumbel distribution $G(0, 1)$ and τ is the temperature that controls the entropy.

Applying $\mathrm{GS}(\boldsymbol{x})$ in a non-generative model is found to induce the model to produce more confident logits $\boldsymbol{x}$ and, consequently, probabilities $\boldsymbol{\pi}$ that resemble one-hot vectors, as stated in Proposition 3.1.

Proposition 3.1. *By introducing disturbance to a non-generative autoencoder's feature logits $\boldsymbol{x}$ via $\mathrm{GS}(\boldsymbol{x})$, the au-*

toencoder, upon non-trivial convergence, produces confident logits $\boldsymbol{x}$, resulting in one-hot-like probabilities $\boldsymbol{\pi}$.

Brief proof:. Consider the binary case with temperature $\tau = 1$, and let $\boldsymbol{x} = (x_1, x_2)$ be the logits from the upstream model, with Gumbel noise added to compute $\boldsymbol{y} = \mathrm{GS}(\boldsymbol{x})$. At convergence, the gradient of the loss $\mathcal{L} = f(\boldsymbol{y})$ with respect to $\boldsymbol{x}$ approaches zero. By computing $\partial\mathcal{L}/\partial x_1$, we find that it depends on $y_1 y_2$ and the derivatives of f, implying that either the output probabilities y_i are near 0/1, or $f(\boldsymbol{y})$ is insensitive to its inputs. The former leads to low-entropy, one-hot-like distributions in $\boldsymbol{y}$. In the latter case, since $\boldsymbol{y}$ varies due to the Gumbel noise, an $f(\boldsymbol{y})$ that is insensitive to its inputs implies that the model has converged to a trivial solution, contradicting the hypothesis. Further, the logits $\boldsymbol{x}$ are bounded by the probability that $\boldsymbol{y}$ deviates from a one-hot-like distributions, indicating that the model produces confident logits to suppress the effect of the Gumbel noise. □

A full version of the proof is provided in Appendix A. Based on this, a converged model that applies the disturbance in Equation (2) to its feature logits $\boldsymbol{x}$ will be constrained to produce one-hot-like probability vectors when Equation (2) is replaced with the $\mathrm{softmax}$ during inference.

4 Differentiable IDS Channel on 3-Simplex Δ^3

It is evident that the operations of insertion and deletion are not differentiable. Consequently, a conventional IDS channel, which modifies a sequence by directly applying IDS operations, hinders gradient propagation and cannot be seamlessly integrated into deep learning-based methods.

Leveraging the logical capabilities inherent in Transformer-based models, a sequence-to-sequence model is employed to simulate the conventional IDS channel. Built on deep models, this simulated IDS channel is differentiable. In the following discussion, we use the notation $\mathrm{CIDS}(\cdot,\cdot)$ to represent the Conventional IDS channel, and $\mathrm{DIDS}(\cdot,\cdot;\theta)$ for the simulated Differentiable IDS channel. The simulated channel is trained independently before being integrated into the autoencoder, whose learned parameters remain fixed during the optimization of the autoencoder.

As the model utilizes probability vectors rather than discrete letters, we need to promote conventional IDS operations onto the 3-simplex Δ^3, where Δ^3 is defined as the collection 4-dimentional probability vectors

$$\Delta^3 = \{\boldsymbol{\pi} |\, \pi_i \geq 0, \sum_{i=1}^{4} \pi_i = 1, i = 1,2,3,4\}. \quad (3)$$

For a sequence of probability vectors $\boldsymbol{C} = (\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots, \boldsymbol{\pi}_k)$, where each $\boldsymbol{\pi}_i$ is an element from the simplex Δ^3, the IDS operations are promoted as follows.

Insertion at index i involves adding a one-hot vector representing the inserted symbol from the alphabet $\{\mathrm{A},\mathrm{T},\mathrm{G},\mathrm{C}\}$ before index i. Deletion at index i simply removes the vector $\boldsymbol{\pi}_i$ from $\boldsymbol{C}$. For substitution, the probability vector $\boldsymbol{\pi}_i$ is rolled by corresponding offsets for the three types of substitutions, which correspond to substitute $\#1, \#2$, and $\#3$ in Figure 12 from Appendix F. For example, applying a type-$\#1$ substitution at index i rolls the original vector $\boldsymbol{\pi}_i = (\pi_{i1}, \pi_{i2}, \pi_{i3}, \pi_{i4})$ into $(\pi_{i4}, \pi_{i1}, \pi_{i2}, \pi_{i3})$. It is straightforward to verify that the promoted IDS operations degenerate to standard IDS operations when the probability vectors are constrained to a one-hot representation.

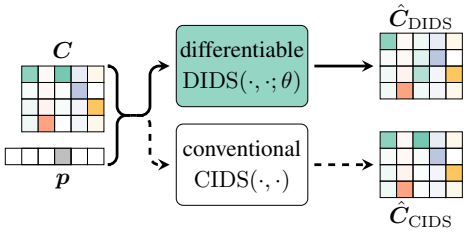

Figure 1: The differentiable IDS channel. The $\hat{\boldsymbol{C}}_{\mathrm{DIDS}}$ and $\hat{\boldsymbol{C}}_{\mathrm{CIDS}}$ are generated by the differentiable and conventional IDS channels, respectively. Optimizing the difference between $\hat{\boldsymbol{C}}_{\mathrm{DIDS}}$ and $\hat{\boldsymbol{C}}_{\mathrm{CIDS}}$ trains the differentiable channel.

As illustrated in Figure 1, both the conventional IDS channel CIDS and the simulated IDS channel DIDS take the sequence $\boldsymbol{C}$ of probability vectors and an error profile $\boldsymbol{p}$ as their inputs. The error profile consists of a sequence of letters that record the types of errors encountered while processing $\boldsymbol{C}$. Complicated IDS channels can be deduced by specifying the rules for generating error profiles. The probability sequence $\boldsymbol{C}$ is expected to be modified by the simulated IDS channel to $\hat{\boldsymbol{C}}_{\mathrm{DIDS}} = \mathrm{DIDS}(\boldsymbol{C}, \boldsymbol{p}; \theta)$ according to the error profile $\boldsymbol{p}$ in the upper stream of Figure 1. In the lower stream, the sequence $\boldsymbol{C}$ is modified as $\hat{\boldsymbol{C}}_{\mathrm{CIDS}} = \mathrm{CIDS}(\boldsymbol{C}, \boldsymbol{p})$ with respect to the error profile $\boldsymbol{p}$ using the previously defined promoted IDS operations.

To train the model $\mathrm{DIDS}(\cdot,\cdot;\theta)$, the Kullback–Leibler divergence (Kullback 1997) of $\hat{\boldsymbol{C}}_{\mathrm{DIDS}}$ from $\hat{\boldsymbol{C}}_{\mathrm{CIDS}}$ can be utilized as the optimization target

$$\mathcal{L}_{\mathrm{KLD}}(\hat{\boldsymbol{C}}_{\mathrm{DIDS}}, \hat{\boldsymbol{C}}_{\mathrm{CIDS}}) = \frac{1}{k} \sum_{i} \hat{\boldsymbol{\pi}}_{i\mathrm{CIDS}}^{T} \log \frac{\hat{\boldsymbol{\pi}}_{i\mathrm{CIDS}}}{\hat{\boldsymbol{\pi}}_{i\mathrm{DIDS}}}. \quad (4)$$

By optimizing Equation (4) on randomly generated probability vector sequences $\boldsymbol{C}$ and error profiles $\boldsymbol{p}$, the parameters θ of the differentiable IDS channel are trained to $\hat{\theta}$. Following this, the model $\mathrm{DIDS}(\cdot,\cdot;\hat{\theta})$ simulates the conventional IDS channel $\mathrm{CIDS}(\cdot,\cdot)$. The significance of such an IDS channel lies in its differentiability. Once optimized independently, the parameters of the IDS channel are fixed for downstream applications. In the following text, we use $\mathrm{DIDS}(\cdot,\cdot)$ to refer to the trained IDS channel for simplicity.

In practice, the differentiable IDS channel is implemented as a sequence-to-sequence model, employing one-layer Transformers for both its encoder and decoder.[1] The

[1]Here, the encoder and decoder refer specifically to the modules of the sequence-to-sequence model, not the modules of the autoencoder. We trust that readers will be able to distinguish between them based on the context.

model takes a padded vector sequence and error profile, whose embeddings are concatenated along the feature dimension as its input. To generate the output, that represents the sequence with errors, learnable position embedding vectors are utilized as the queries (omitted from Figure 1).

5 THEA-Code

5.1 Framework

The flowchart of the proposed code is illustrated in Figure 2. Based on the principles of DNA-based storage, which synthesizes DNA molecules of fixed length, the proposed model is designed to handle source sequences and codewords of constant lengths. Essentially, the proposed method encodes source sequences into codewords; the IDS channel introduces IDS errors to these codewords; and a decoder is employed to reconstruct the recovered sequences according to the corrupted codewords.

Let $f_{\mathrm{en}}(\cdot;\phi)$ denote the encoder, where ϕ represents the encoder's parameters. The source sequence $\boldsymbol{s}$ is first encoded into the codeword $\boldsymbol{c} = f_{\mathrm{en}}(\boldsymbol{s};\phi)$ by the encoder,[2] where the codeword $\boldsymbol{c}$ is obtained using Equation (2) during the training phase and argmax during the testing phase. Next, a random error profile $\boldsymbol{p}$ is generated, which records the positions and types of errors that will occur on codeword $\boldsymbol{c}$. Given the error profile $\boldsymbol{p}$, the codeword $\boldsymbol{c}$ is transformed into the corrupted codeword $\hat{\boldsymbol{c}} = \mathrm{DIDS}(\boldsymbol{c}, \boldsymbol{p}; \hat{\theta})$ by the simulated differentiable IDS channel, implemented as a sequence-to-sequence model with trained parameters $\hat{\theta}$. Finally, a decoder $f_{\mathrm{de}}(\cdot;\psi)$ with parameters ψ decodes the corrupted codeword $\hat{\boldsymbol{c}}$ back into the recovered sequence $\hat{\boldsymbol{s}} = f_{\mathrm{de}}(\hat{\boldsymbol{c}};\psi)$.

Following this pipeline, a natural optimization target is the cross-entropy loss

$$\mathcal{L}_{\mathrm{CE}}(\hat{\boldsymbol{s}}, \boldsymbol{s}) = -\sum_i \sum_j \mathbb{1}_{j=s_i} \log \hat{s}_{ij}, \tag{5}$$

which evaluates the reconstruction disparity of the source sequence $\boldsymbol{s}$ (in its label representation) by the recovered sequence $\hat{\boldsymbol{s}}$ (in its one-hot probability distribution).

However, merely optimizing such a loss function will not yield the desired outcomes. While the encoder and decoder of an autoencoder typically collaborate on a unified task in most applications, in this work, we expect them to follow distinct underlying logic. Particularly, when imposing constraints to enforce greater discreteness in the codeword, the joint training of the encoder and decoder becomes challenging, where the optimization of each relies on the other during the training phase.

5.2 Auxiliary reconstruction of source sequence by the encoder

To address the aforementioned issue, we introduce a supplementary task exclusively for the encoder, aimed at initializing it with some foundational logical capabilities. Inspired by the systematic code which embed the input message within the codeword, a straightforward task for the encoder is to replicate the input sequence at the output, ensuring that the model preserves all information from its input without reduction. With this in mind, we incorporate a reconstruction task into the encoder's training process.

In practice, the encoder is designed to output a longer sequence, which is subsequently split into two parts: the codeword representation $\boldsymbol{c}$ and an auxiliary reconstruction $\boldsymbol{r}$ of the input source sequence, as shown in Figure 2. The auxiliary reconstruction loss is calculated using the cross-entropy loss as

$$\mathcal{L}_{\mathrm{Aux}}(\boldsymbol{r}, \boldsymbol{s}) = -\sum_i \sum_j \mathbb{1}_{j=s_i} \log r_{ij}, \tag{6}$$

which quantifies the difference between the reconstruction $\boldsymbol{r}$ (in its one-hot probability distribution) and the input sequence $\boldsymbol{s}$ (in its label representation).

Considering that the auxiliary loss may not have negative effects on the encoder for its simple logic, we don't use a separate training stage for optimizing the $\mathcal{L}_{\mathrm{Aux}}$. The auxiliary loss defined in Equation (6) is incorporated into the overall loss function and applied consistently throughout the entire training phase.

5.3 The encoder and decoder

In this approach, both the encoder and decoder are implemented using Transformer-based sequence-to-sequence models. Each consists of (3+3)-layer Transformers with sinusoidal positional encoding. The embedding of the DNA bases is implemented through a fully connected layer without bias to ensure compatibility with probability vectors. Learnable position index embeddings are employed to query the outputs.

5.4 Training phase

The training process is divided into two phases. Firstly, the differentiable IDS channel is fully trained by optimizing

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}_{\mathrm{KLD}}(\hat{\boldsymbol{C}}_{\mathrm{DIDS}}, \hat{\boldsymbol{C}}_{\mathrm{CIDS}}) \tag{7}$$

on randomly generated codewords $\boldsymbol{c}$ and profiles $\boldsymbol{p}$. Once the differentiable IDS channel is trained, its parameters are fixed. The remaining components of the autoencoder are then trained by optimizing a weighted sum of Equation (5) and Equation (6),

$$\hat{\phi}, \hat{\psi} = \arg\min_{\phi,\psi} \mathcal{L}_{\mathrm{CE}}(\hat{\boldsymbol{s}}, \boldsymbol{s}) + \mu \mathcal{L}_{\mathrm{Aux}}(\boldsymbol{r}, \boldsymbol{s}), \tag{8}$$

where μ is a hyperparameter representing the weight of the auxiliary reconstruction loss. The autoencoder is trained on randomly generated input sequences $\boldsymbol{s}$ and profiles $\boldsymbol{p}$.

5.5 Testing phase

In the testing phase, the differentiable IDS channel is replaced with the conventional IDS channel. The process begins with the encoder mapping the source sequence $\boldsymbol{s}$ to the codeword $\boldsymbol{c}$ in the form of probability vectors. An argmax function is then applied to convert $\boldsymbol{c}$ into a discrete letter

[2] For simplicity, we do not distinguish between notations for sequences represented as letters, one-hot vectors, or probability vectors in the following text.

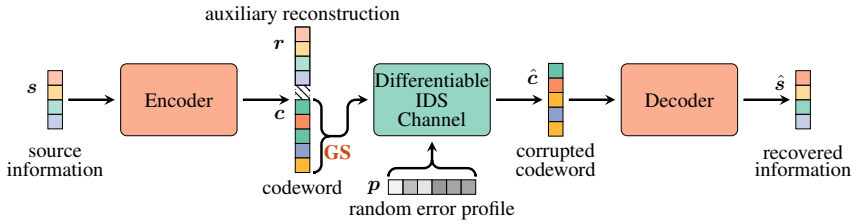

Figure 2: The flowchart of THEA-Code, including the encoder, the pretrained IDS channel, and the decoder. All of these modules are implemented using Transformer-based models. The "GS" is where the disturbance based discretization applied in the pipeline.

sequence, removing any extra information from the probability vectors. Next, the conventional IDS operations are performed on $\hat{\boldsymbol{c}} = \mathrm{CIDS}(\boldsymbol{c}, \boldsymbol{p})$ according to a randomly generated error profile $\boldsymbol{p}$. The one-hot representation of $\hat{\boldsymbol{c}}$ is then passed into the decoder, which reconstructs the recovered sequence $\hat{\boldsymbol{s}}$. Finally, metrics are computed to measure the differences between the original source sequence $\boldsymbol{s}$ and the reconstructed sequence $\hat{\boldsymbol{s}}$, providing an evaluation of the method's performance.

Since the sequences are randomly generated from an enormous pool of possible terms, the training and testing sets are separated using different random seeds. For example, in the context of this work, the source sequence is a 100-long 4-ary sequence, providing 1.6×10^{60} possible sequences. Given this vast space, sets of randomly generated sequences using different seeds are unlikely to overlap.

6 Experiments on the Differentiable IDS Channel

6.1 Accuracy of the channel

The differentiable IDS channel is expected to faithfully modify the input sequence according to the given profiles. To explicitly demonstrate the performance, accuracy is evaluated under various profile settings.

The results is illustrated in Figure 3. It is suggested that the differentiable IDS channel edits the input sequence faithfully according to the profile when the total channel error rate is no more than 20%. When the error rate exceeds 20%, the accuracy of the differentiable IDS channel declines as the channel error rate increases. It is worth noting that realistic DNA-based storage channels typically do not exhibit error rates above 20%.

7 Experiments on the IDS-Correcting Code

Commonly used methods for synthesizing DNA molecules in DNA-based storage pipelines typically yield sequences of lengths ranging from 100 to 200 (Welter et al. 2024). In this study, we choose the number 150 as the codeword length, aligning with these established practices. Unless explicitly stated otherwise, all the following experiments adhere to the default setting: source sequence length $\ell_s = 100$, codeword length $\ell_c = 150$, auxiliary loss weight $\mu = 1$, and the error profile is generated with a 1% probability of errors occurring

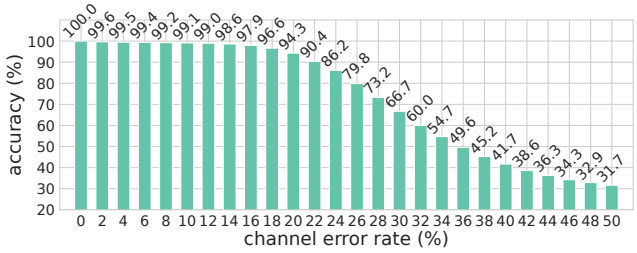

Figure 3: The accuracy of the differentiable IDS channel under various channel error rates. Accuracy is calculated by comparing the outputs of the differentiable IDS channel with those of the conventional IDS channel.

at each position, with insertion, deletion, and substitution errors equally likely.

To evaluate performance, the nucleobase error rate (NER) is employed as a metric, analogous to the bit error rate (BER), but replacing bits with nucleobases. For a DNA sequence $\boldsymbol{s}$ and its decoded counterpart $\hat{\boldsymbol{s}}$, the NER is defined as

$$\mathrm{NER}(\boldsymbol{s}, \hat{\boldsymbol{s}}) = \frac{\#\{s_i \neq \hat{s}_i\}}{\#\{s_i\}}. \tag{9}$$

The NER represents the proportion of nucleobase errors corresponding to base substitutions in the source DNA sequence. It's worth noting that these errors can be post-corrected using a mature conventional outer code.

The source code is uploaded at https://github.com/aalennku/THEA-Code.

7.1 Performance with different channel settings

The code rate is the proportion of non-redundant data in the codeword, calculated by dividing the source length ℓ_s by the codeword length ℓ_c. We explored variable source lengths ℓ_s while keeping the codeword length $\ell_c = 150$ fixed. The results in Table 1 reveal a trend that the NER increases from 0.09% to 2.81% as the code rate increases from 0.33 to 0.83.

By applying an outer conventional ECC to address the remaining NER, which is a common technique in DNA-based storage (Press et al. 2020; Pfister and Tal 2021; Yan, Liang, and Wu 2022; Welzel et al. 2023), a complete solution for DNA-based storage is achieved. Here, the IDS-correcting code is focused.

ℓ_s	50	75	100	125
code rate	0.33	0.50	0.67	0.83
NER(%)	0.12 ± 0.03	0.51 ± 0.03	1.15 ± 0.08	3.71 ± 0.59

Table 1: The testing NER for different source lengths ℓ_s, with the codeword length fixed at $\ell_c = 150$. The code rate is calculated as ℓ_s/ℓ_c, ranging from 0.33 to 0.83.

By controlling the generation process of the error profile $\boldsymbol{p}$ for different channel settings, we can evaluate whether THEA-Code learns channels' attributes and produces customized codes based on the models' performance.

Results on IDS channels with position related errors. Along with the default setting, where error rates are position-insensitive (denoted as Hom), two other IDS channels parameterized by ascending (Asc) and descending (Des) error rates along the sequence are considered.[3] The Asc channel has error rates increasing from 0% to 2% along the sequence, with the average error rate matching that of the default setting Hom. The Des channel follows a similar pattern but has decreasing error rates along the sequence.

To verify that the proposed method customizes codes for different channels, cross-channel testing was conducted, with the results shown in Table 2. The numbers in the matrix represent the NER of a model trained with the channel of the row and tested on the channel of the column.

The diagonal of Table 2 shows the results of the model trained and tested with a consistent channel, suggesting that the learned THEA-Code exhibits varying performance depending on the specific channel configuration. The columns of Table 2 suggest that, for each testing channel, models trained with the channel configuration consistently achieve the best performance among the three channel settings. Considering the Hom channel is a midway setting between Asc and Des, the first and third columns (and rows) show that the more dissimilar the training and testing channels are, the worse the model's performance becomes, even though the overall error rates are the same across the three channels. These findings verify that the deep learning-based method effectively customizes codes for specific channels, which could advance IDS-correcting code design into a more fine-grained area.

Results on IDS channels with various IDS error rates. IDS channels with larger error probabilities were also tested. The experiments were extended to include channels with error probabilities in $\{0.5\%, 1\%, 2\%, 4\%, 8\%, 16\%\}$, with results listed in Table 3.

It is suggested that models trained on channels with higher error probabilities exhibit compatibility with channels with lower error probabilities. In most cases, models trained and tested on similar channels achieve better performance.

Results on realistic IDS channels. We also conducted experiments using IDS channels that more closely resemble realistic IDS channels in DNA-based storage. A memory channel was proposed in (Hamoum et al. 2021), relying on statistical data obtained via a realistic storage pipeline. It models the IDS errors based on the k-mers of sequences and adjacent edits. In this work, we utilize the publicly released trained memory channel from (Hamoum et al. 2021), filtering out apparent outlier sequences with Levenshtein distance greater than 20. This simulated channel is referred to as MemSim.

In practice, a DNA sequence $\boldsymbol{c}$ is input into MemSim to produce the output sequence $\hat{\boldsymbol{c}}$ from the channel. By comparing $\boldsymbol{c}$ and $\hat{\boldsymbol{c}}$, an error profile $\boldsymbol{p}$ is inferred. Using the sequence $\boldsymbol{c}$ and the error profile $\boldsymbol{p}$ in the procedure depicted in Figure 2, an IDS-correcting code for MemSim is customized.

For comparison, two simple channels, partially aligned with MemSim, were also considered. The overall IDS error rate for MemSim is 10.36%, with the proportions of insertion, deletion, and substitution being 1.66%, 5.31%, and 3.38%, respectively. We refer to the context-free channel with these specific error proportions as channel C253. Channel C111 is defined as having the same overall IDS error rate 10.36%, but with equal proportions of insertion, deletion, and substitution. It is evident that MemSim is the closest approximation to a realistic channel, followed by C253, while

NER(%)	Asc	Hom	Des
Asc	$\mathbf{0.90 \pm 0.09}$	1.46 ± 0.08	2.09 ± 0.44
Hom	1.03 ± 0.20	$\mathbf{1.15 \pm 0.08}$	1.30 ± 0.03
Des	1.72 ± 0.12	1.32 ± 0.07	$\mathbf{1.01 \pm 0.05}$

Table 2: The testing NER across different channels. Each entry is the NER of a model trained (resp. tested) with the row (resp. column) header channel.

NER(%)	0.5%	1%	2%	4%	8%	16%
0.5%	**0.68**	1.59	4.26	11.67	26.87	45.61
1%	0.52	**1.15**	2.90	8.12	21.19	41.03
2%	0.67	1.43	**3.16**	7.79	18.7	36.89
4%	1.25	1.76	2.88	**5.53**	12.39	28.31
8%	2.74	3.24	4.30	6.62	**12.2**	25.41
16%	11.57	11.93	12.61	14.4	17.22	**25.51**

Table 3: The testing NER across different IDS error probabilities. The row and column headers correspond to channels configured with respective probabilities of errors. Each entry represents the NER of a model trained (resp. tested) on the channel specified by the row (resp. column) header.

code rate	0.33	0.50	0.6	0.67	0.75	0.83
Cai	0.44	1.00	-	2.53	-	8.65
DNA-LM	0.55	1.03	-	2.29	-	7.43
HEDGES	0.28	**0.25**	**0.65**	-	3.43	-
THEA-Code	**0.09**	0.46	1.00	**1.06**	**2.03**	**2.81**

Table 4: The testing error rates compared with different established codes, through the default 1% IDS channel.

[3]These settings simplify DNA-based storage channels, as a DNA sequence is marked with a 3' end and a 5' end. Some researchers believe that the error rate accumulates towards the sequence end during synthesis (Meiser et al. 2020).

	$r = 0.33$			$r = 0.50$			$r = 0.67$		
NER(%)	C111	C253	MemSim	C111	C253	MemSim	C111	C253	MemSim
C111	**2.28**	3.02	15.9	**7.60**	8.77	24.85	**15.19**	16.96	34.46
C253	2.73	**2.93**	17.3	9.15	**9.13**	25.09	16.87	**16.90**	32.86
MemSim	5.60	6.64	**1.55**	14.78	16.62	**6.11**	24.89	25.91	**12.02**

Table 5: The testing NER across different channels including C111, C253, and MemSim, under varying code rates. Each entry represents the NER of a model trained (resp. tested) on the channel specified by the row (resp. column) header.

	$r = 0.33$			$r = 0.50$			$r = 0.67$		
	C111	C253	MemSim	C111	C253	MemSim	C111	C253	MemSim
Cai	17.01	17.52	72.74	29.00	29.57	74.40	40.12	42.62	73.90
DNA-LM	32.24	37.33	60.13	45.32	51.13	64.27	56.34	60.22	68.72
HEDGES	3.21	4.56	29.42	27.22	27.79	99.56	54.35	55.66	99.62
THEA-Code	**2.28**	**2.93**	**1.55**	**7.60**	**9.13**	**6.11**	**15.19**	**16.90**	**12.02**

Table 6: The testing error rates compared with established code through channels including C111, C253, and MemSim, under varying code rates.

C111 deviates the most from a realistic channel, despite all having the same overall IDS error rate.

The results across channels, including C111, C253, and MemSim, are presented in Table 5. The results suggest that THEA-Code performs better when the model is trained on the same channel used for testing. Specifically, for the realistic channel, codes trained on the simpler channels C253 and C111 fail to deliver satisfactory results. Overall, THEA-Code trained and tested with MemSim achieves the best results, demonstrating that the proposed model significantly benefits from customizing the code for the realistic channel.

7.2 Comparison experiments

Comparison experiments were conducted against prior works include: the combinatorial code from (Cai et al. 2021), the segmented code method DNA-LM from (Yan, Liang, and Wu 2022), and the efficient heuristic method HEDGES from (Press et al. 2020).

Such methods are typically designed to operate under discrete, fixed configurations, making it challenging to align them within the same setting. We made every effort to align these methods, and present a subset of the comparison results in Table 4, which is tested through the default 1% error channel. Detailed configurations and results across multiple channels are provided in Appendix B.

Table 4 demonstrates the effectiveness of the proposed method. The performance of THEA-Code and HEDGES outperform the other methods by a large margin. At lower code rates, THEA-Code achieves a comparable error rate to HEDGES. At higher code rates, the proposed method outperforms HEDGES, achieving much lower error rates.

Comparison through the realistic channel. We also compared these codes across the channels C111, C253, and MemSim introduced in Section 7.1, all of which have an overall channel error rate of 10.36%. Specifically, MemSim simulates the IDS channel from a realistic storage pipeline.

The results are illustrated in Table 6. It can be observed that high-error-rate channels severely degrade the performance of compared codes, while the proposed THEA-Code outperforms them by a significant margin. Moreover, the compared codes, lacking the ability to adapt to specific channels, show a noticeable decline in performance as the channel transitions from the simpler C111/C253 to the more realistic MemSim. In contrast, THEA-Code leverages customized channel-specific designs, achieving the best performance on MemSim across all three channels.

8 More Experiments in the Appendices

In this work, the disturbance-based discretization, the differentiable IDS channel, and the auxiliary reconstruction loss are newly proposed. Comprehensive experiments on these modules are presented in the Appendices provided in https://arxiv.org/abs/2407.18929. Following is a brief overview.

The full proof of Proposition 3.1 is given in Appendix A. Details of the comparison experiments discussed in Section 7.2 are provided in Appendix B.

Disturbance-based discretization. The ablation studies and hyperparameter optimization for the disturbance-based discretization, including the discretization effect compared to vanilla softmax, the optimization of temperature τ, and the results of potential alternative, are in Appendix C.

Differentiable IDS channel. Experiments on the differentiable IDS channel, including the gradient trace under specific error profiles, and the gradients with respect to the error profile through an identity channel, are in Appendix D.

Auxiliary reconstruction loss. For the auxiliary reconstruction loss, ablation studies, weight optimization of the loss term μ, and experiments on different auxiliary patterns are provided in Appendix E.

Dataset and model. The construction of datasets and the definition of error profiles are detailed in Appendix F. A brief introduction to the Transformer model, as well as complexity analysis and time consumption, is presented in Appendix G.

Acknowledgements

This work was supported by the National Key Research and Development Program of China under Grant 2020YFA0712100 and 2025YFC3409900, the National Natural Science Foundation of China, and the Emerging Frontiers Cultivation Program of Tianjin University Interdisciplinary Center.

References

Akrout, M.; Feriani, A.; Bellili, F.; Mezghani, A.; and Hossain, E. 2023. Domain Generalization in Machine Learning Models for Wireless Communications: Concepts, State-of-the-Art, and Open Issues. *IEEE Communications Surveys & Tutorials*.

Baldi, P. 2012. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, 37–49. JMLR Workshop and Conference Proceedings.

Balevi, E.; and Andrews, J. G. 2020. Autoencoder-Based Error Correction Coding for One-Bit Quantization. *IEEE Transactions on Communications*, 68(6): 3440–3451.

Bar-Lev, D.; Etzion, T.; and Yaakobi, E. 2023. On the Size of Balls and Anticodes of Small Diameter Under the Fixed-Length Levenshtein Metric. *IEEE Transactions on Information Theory*, 69(4): 2324–2340.

Blawat, M.; Gaedke, K.; Huetter, I.; Chen, X.-M.; Turczyk, B.; Inverso, S.; Pruitt, B. W.; and Church, G. M. 2016. Forward error correction for DNA data storage. *Procedia Computer Science*, 80: 1011–1022.

Cai, K.; Chee, Y. M.; Gabrys, R.; Kiah, H. M.; and Nguyen, T. T. 2021. Correcting a single indel/edit for DNA-based data storage: Linear-time encoders and order-optimality. *IEEE Transactions on Information Theory*, 67(6): 3438–3451.

Calabi, L.; and Hartnett, W. 1969. A family of codes for the correction of substitution and synchronization errors. *IEEE Transactions on Information Theory*, 15(1): 102–106.

Cammerer, S.; Gruber, T.; Hoydis, J.; and Ten Brink, S. 2017. Scaling deep learning-based decoding of polar codes via partitioning. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, 1–6. IEEE.

Chen, W.; Han, M.; Zhou, J.; Ge, Q.; Wang, P.; Zhang, X.; Zhu, S.; Song, L.; and Yuan, Y. 2021. An artificial chromosome for data storage. *National Science Review*, 8(5): nwab028.

Choukroun, Y.; and Wolf, L. 2022. Error Correction Code Transformer. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems*, volume 35, 38695–38705. Curran Associates, Inc.

Choukroun, Y.; and Wolf, L. 2023. Denoising Diffusion Error Correction Codes. In *The Eleventh International Conference on Learning Representations*.

Choukroun, Y.; and Wolf, L. 2024a. Deep quantum error correction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 64–72.

Choukroun, Y.; and Wolf, L. 2024b. A Foundation Model for Error Correction Codes. In *The Twelfth International Conference on Learning Representations*.

Choukroun, Y.; and Wolf, L. 2024c. Learning Linear Block Error Correction Codes. In *International Conference on Machine Learning*, 8801–8814. PMLR.

Church, G. M.; Gao, Y.; and Kosuri, S. 2012. Next-generation digital information storage in DNA. *Science*, 337(6102): 1628–1628.

Davey, M.; and Mackay, D. 2001. Reliable communication over channels with insertions, deletions, and substitutions. *IEEE Transactions on Information Theory*, 47(2): 687–698.

Dong, Y.; Sun, F.; Ping, Z.; Ouyang, Q.; and Qian, L. 2020. DNA storage: research landscape and future prospects. *National Science Review*, 7(6): 1092–1107.

El-Shaikh, A.; Welzel, M.; Heider, D.; and Seeger, B. 2022. High-scale random access on DNA storage systems. *NAR Genomics and Bioinformatics*, 4(1): lqab126.

Erlich, Y.; and Zielinski, D. 2017. DNA Fountain enables a robust and efficient storage architecture. *Science*, 355(6328): 950–954.

Gabrys, R.; Guruswami, V.; Ribeiro, J.; and Wu, K. 2023. Beyond Single-Deletion Correcting Codes: Substitutions and Transpositions. *IEEE Transactions on Information Theory*, 69(1): 169–186.

Goldman, N.; Bertone, P.; Chen, S.; Dessimoz, C.; LeProust, E. M.; Sipos, B.; and Birney, E. 2013. Towards practical, high-capacity, low-maintenance information storage in synthesized DNA. *Nature*, 494(7435): 77–80.

Grass, R. N.; Heckel, R.; Puddu, M.; Paunescu, D.; and Stark, W. J. 2015. Robust chemical preservation of digital information on DNA in silica with error-correcting codes. *Angewandte Chemie International Edition*, 54(8): 2552–2555.

Gumbel, E. J. 1935. Les valeurs extrêmes des distributions statistiques. In *Annales de l'institut Henri Poincaré*, volume 5, 115–158.

Haeupler, B.; and Shahrasbi, A. 2021. Synchronization strings and codes for insertions and deletions—A survey. *IEEE Transactions on Information Theory*, 67(6): 3190–3206.

Hamoum, B.; Dupraz, E.; Conde-Canencia, L.; and Lavenier, D. 2021. Channel model with memory for DNA data storage with nanopore sequencing. In *2021 11th International Symposium on Topics in Coding (ISTC)*, 1–5. IEEE.

Hirao, I.; Nishimura, Y.; Tagawa, Y.-i.; Watanabe, K.; and Miura, K.-i. 1992. Extraordinarily stable mini-hairpins: Electrophoretical and thermal properties of the various sequence variants of d (GCFAAAGC) and their effect on DNA sequencing. *Nucleic acids research*, 20(15): 3891–3896.

Ibnkahla, M. 2000. Applications of neural networks to digital communications–a survey. *Signal Processing*, 80(7): 1185–1215.

Jang, E.; Gu, S.; and Poole, B. 2017. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*.

Jiang, Y.; Kim, H.; Asnani, H.; Kannan, S.; Oh, S.; and Viswanath, P. 2019. Turbo Autoencoder: Deep learning based channel codes for point-to-point communication channels. In *Advances in Neural Information Processing Systems*, 2754–2764.

Kim, H.; Jiang, Y.; Rana, R. B.; Kannan, S.; Oh, S.; and Viswanath, P. 2018. Communication Algorithms via Deep Learning. In *International Conference on Learning Representations*.

Kullback, S. 1997. *Information theory and statistics*. Courier Corporation.

Levenshtein, V. I. 1965. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics. Doklady*, 10: 707–710.

Maarouf, I.; Lenz, A.; Welter, L.; Wachter-Zeh, A.; Rosnes, E.; and i Amat, A. G. 2022. Concatenated codes for multiple reads of a DNA sequence. *IEEE Transactions on Information Theory*, 69(2): 910–927.

Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *International Conference on Learning Representations*.

Makkuva, A. V.; Liu, X.; Jamali, M. V.; Mahdavifar, H.; Oh, S.; and Viswanath, P. 2021. Ko codes: inventing nonlinear encoding and decoding for reliable wireless communication via deep-learning. In *International Conference on Machine Learning*, 7368–7378. PMLR.

Meiser, L. C.; Koch, J.; Antkowiak, P. L.; Stark, W. J.; Heckel, R.; and Grass, R. N. 2020. DNA synthesis for true random number generation. *Nature communications*, 11(1): 5869.

Meiser, L. C.; Nguyen, B. H.; Chen, Y.-J.; Nivala, J.; Strauss, K.; Ceze, L.; and Grass, R. N. 2022. Synthetic DNA applications in information technology. *Nature communications*, 13(1): 352.

Mitzenmacher, M. 2009. A survey of results for deletion channels and related synchronization channels. *Probability Surveys*, 6(none): 1 – 33.

Nachmani, E.; Marciano, E.; Lugosch, L.; Gross, W. J.; Burshtein, D.; and Be'ery, Y. 2018. Deep Learning Methods for Improved Decoding of Linear Codes. *IEEE Journal of Selected Topics in Signal Processing*, 12(1): 119–131.

Nachmani, E.; and Wolf, L. 2019. Hyper-Graph-Network Decoders for Block Codes. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Organick, L.; Ang, S. D.; Chen, Y.-J.; Lopez, R.; Yekhanin, S.; Makarychev, K.; Racz, M. Z.; Kamath, G.; Gopalan, P.; Nguyen, B.; et al. 2018. Random access in large-scale DNA data storage. *Nature Biotechnology*, 36(3): 242–248.

Park, S.-J.; Kwak, H.-Y.; Kim, S.-H.; Kim, Y.; and No, J.-S. 2025. CrossMPT: Cross-attention Message-passing Transformer for Error Correcting Codes. In *The Thirteenth International Conference on Learning Representations*.

Pfister, H. D.; and Tal, I. 2021. Polar Codes for Channels with Insertions, Deletions, and Substitutions. In *2021 IEEE International Symposium on Information Theory (ISIT)*, 2554–2559.

Press, W. H.; Hawkins, J. A.; Jones, S. K.; Schaub, J. M.; and Finkelstein, I. J. 2020. HEDGES error-correcting code for DNA storage corrects indels and allows sequence constraints. *Proceedings of the National Academy of Sciences*, 117(31): 18489–18496.

Sellers, F. 1962. Bit loss and gain correction code. *IRE Transactions on Information Theory*, 8(1): 35–38.

Simeone, O. 2018. A very brief introduction to machine learning with applications to communication systems. *IEEE Transactions on Cognitive Communications and Networking*, 4(4): 648–664.

Sloane, N. J. 2000. On single-deletion-correcting codes. *Codes and designs*, 10: 273–291.

Srinivasavaradhan, S. R.; Gopi, S.; Pfister, H. D.; and Yekhanin, S. 2021. Trellis BMA: Coded trace reconstruction on IDS channels for DNA storage. In *2021 IEEE International Symposium on Information Theory (ISIT)*, 2453–2458. IEEE.

Tanaka, E.; and Kasai, T. 1976. Synchronization and substitution error-correcting codes for the Levenshtein metric. *IEEE Transactions on Information Theory*, 22(2): 156–162.

Varshamov, R. R.; and Tenenholtz, G. 1965. A code for correcting a single asymmetric error. *Automatica i Telemekhanika*, 26(2): 288–292.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Welter, L.; Sokolovskii, R.; Heinis, T.; Wachter-Zeh, A.; Rosnes, E.; et al. 2024. An End-to-End Coding Scheme for DNA-Based Data Storage With Nanopore-Sequenced Reads. *arXiv preprint arXiv:2406.12955*.

Welzel, M.; Schwarz, P. M.; Löchel, H. F.; Kabdullayeva, T.; Clemens, S.; Becker, A.; Freisleben, B.; and Heider, D. 2023. DNA-Aeon provides flexible arithmetic coding for constraint adherence and error correction in DNA storage. *Nature Communications*, 14(1): 628.

Yan, Z.; Liang, C.; and Wu, H. 2022. A Segmented-Edit Error-Correcting Code With Re-Synchronization Function for DNA-Based Storage Systems. *IEEE Transactions on Emerging Topics in Computing*, 1–13.