GUMBEL-SOFTMAX DISCRETIZATION CONSTRAINT, DIFFERENTIABLE IDS CHANNEL, AND AN IDS-CORRECTING CODE FOR DNA STORAGE

Anonymous authors
Paper under double-blind review

ABSTRACT

Insertion, deletion, and substitution (IDS) error-correcting codes have garnered increased attention with recent advancements in DNA storage technology. However, a universal method for designing IDS-correcting codes across varying channel settings remains underexplored. We present an autoencoder-based method, THEA-code, aimed at efficiently generating IDS-correcting codes for complex IDS channels. In the work, a Gumbel-Softmax discretization constraint 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, resulting in channel-customized IDS-correcting codes with commendable performance across complex IDS channels.

1 INTRODUCTION

DNA storage, a method that utilizes the synthesis and sequencing of DNA molecules for information storage and retrieval, has attracted significant attention (Church et al., 2012; Goldman et al., 2013; Grass et al., 2015; Erlich & Zielinski, 2017; Organick et al., 2018; Dong et al., 2020; Chen et al., 2021; El-Shaikh et al., 2022; Welzel et al., 2023).

Due to the involvement of biochemical procedures, the DNA 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 DNA storage.

However, despite the existence of excellent combinatorial IDS-correcting codes (Varshamov & Tenenholtz, 1965; Levenshtein, 1965; Sloane, 2000; Mitzenmacher, 2009; Cai et al., 2021; Gabrys et al., 2023; Bar-Lev et al., 2023), applying them in DNA storage remains challenging. The IDS channel in DNA 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 storage.

Given the complexity of the IDS channel, we leverage the universality of deep learning methods by employing a heuristic end-to-end autoencoder (Baldi, 2012) as the foundation for an IDS-correcting code. This approach allows researchers to train customized codes tailored to different IDS channels through the same training procedure, rather than to design specific combinatorial codes for each IDS channel setting, many of which have not yet been explored.

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 Gumbel-Softmax in a non-generative model is investigated in this work. Originally proposed as a differentiable approximation for categorical sampling (Jang et al., 2017; Maddison et al., 2017; Huijben et al., 2023), Gumbel-Softmax has been widely used as a reparameterization trick, particularly in variational autoencoders. It is found that

applying Gumbel-Softmax in a non-generative model induces discretized features, which align with the discrete codewords of an error-correcting code (ECC). This discovery may offer an alternative method for bridging the gap between continuous deep 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 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, the Gumbel-Softmax discretization constraint 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 introduces the first end-to-end autoencoder solution for an IDS-correcting code. Additionally, it marks the first application of Gumbel-Softmax as a discretization constraint, and the first proposal of a differentiable IDS channel.

2 Related Works

Many established IDS-correcting codes are rooted in the Varshamov-Tenengolts (VT) code (Varshamov & Tenenholtz, 1965; Levenshtein, 1965), including (Calabi & Hartnett, 1969; Tanaka & 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 storage, such as those proposed in (Pfister & Tal, 2021; Yan et al., 2022; Maarouf et al., 2022; Welzel et al., 2023), usually incorporate synchronization markers (Sellers, 1962; Srinivasavaradhan et al., 2021; Haeupler & Shahrasbi, 2021), watermarks (Davey & 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). Several architectures have been employed as decoders or sub-modules of conventional codes on the 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 & Wolf, 2022; 2023; 2024b;a;c). Hypergraph networks were also utilized as decoders for block codes in (Nachmani & 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 & 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 GUMBEL-SOFTMAX DISCRETIZATION CONSTRAINT

The Gumbel-Softmax method was introduced as a differentiable approximation for sampling from categorical distributions. Let $\boldsymbol{x}$ be the logits that produce the probabilities $\boldsymbol{\pi} = \{\pi_1, \pi_2, \ldots, \pi_k\}$ via the softmax function,

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

The Gumbel-Softmax is the softmax variant of the Gumbel-Max trick (Gumbel, 1954)

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

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. The Gumbel-Softmax is commonly applied in generative models, such as variational autoencoders (VAE), to sample from a categorical distribution of latent variables while retaining the ability to compute gradient.

In this work, it is found that applying Gumbel-Softmax in a non-generative model induces the categorical distribution to resemble a one-hot vector. Intuitively, Gumbel-Softmax introduces indeterminacy in its output $\mathrm{GS}(\boldsymbol{x})$ by sampling from the Gumbel distribution. In a non-generative model, the network may attempt to eliminate this indeterminacy by producing more confident logits $\boldsymbol{x}$.

For simplicity, the binary case is considered as an example, with the temperature set to $\tau = 1$. Let $\boldsymbol{x} = (x_1, x_2)$ represent the logits outputted by the upstream model, and let $\boldsymbol{y} = \mathrm{GS}(\boldsymbol{x})$ denote the Gumbel-Softmax of $\boldsymbol{x}$, where

$$y_i = \frac{\exp(x_i + g_i)}{\exp(x_1 + g_1) + \exp(x_2 + g_2)}, \quad i = 1, 2. \tag{3}$$

Let $\mathcal{L} = f(y_1, y_2)$ be the optimization target of the model, which represents the composite function of the downstream model and the loss function.

The partial derivative of the optimization target with respect to x_1 is calculated using the chain rule:

$$\frac{\partial \mathcal{L}}{\partial x_1} = \frac{\partial f}{\partial y_1}\frac{\partial y_1}{\partial x_1} + \frac{\partial f}{\partial y_2}\frac{\partial y_2}{\partial x_1} = y_1 y_2 \left(\frac{\partial f}{\partial y_1} - \frac{\partial f}{\partial y_2}\right). \tag{4}$$

The calculations for x_2, y_2 are analogous to those for x_1, y_1 and are omitted here and in the following text. It is known that a model converges to a local minimum has a zero gradient, thus according to Equation (4), when the model converges, either y_1, y_2 or $\left|\frac{\partial f}{\partial y_1} - \frac{\partial f}{\partial y_2}\right|$ should be zero or less than a minimal value ϵ.

Consider the case where $\left|\frac{\partial f}{\partial y_1} - \frac{\partial f}{\partial y_2}\right| < \epsilon$. Noting that $y_2 = 1 - y_1$, the partial derivative of $\mathcal{L}$ with respect to y_1 is given by

$$\left|\frac{\partial \mathcal{L}}{\partial y_1}\right| = \left|\frac{\partial f}{\partial y_1} + \frac{\partial f}{\partial y_2}\frac{\partial y_2}{\partial y_1}\right| = \left|\frac{\partial f}{\partial y_1} - \frac{\partial f}{\partial y_2}\right| < \epsilon. \tag{5}$$

Since y_1 is calculated by sampling a random variable as described in Equation (3), either y_1 remains constant with respect to different g_i, or Equation (5) holds for a variable y_i, in which case the downstream model degenerates into a trivial model, as its optimization target becomes insensitive to different inputs. The partial derivative of y_1 with respect to g_1 is

$$\frac{\partial y_1}{\partial g_1} = y_1 y_2. \tag{6}$$

If y_1 is not sensitive to g_1, either y_1 or y_2 should be zero or less than a minimal value ϵ.

All the above cases indicate that the converged model should have either y_1 or y_2 less than ϵ. Taking $y_1 < \epsilon_1$ as an example, it can be reformulated as

$$\frac{1}{y_1} = 1 + \exp(x_2 - x_1 + g_2 - g_1) > M_1, \tag{7}$$

where $M_1 = 1/\epsilon_1$. Since g_1, g_2 independently follow the Gumbel distribution $G(0,1)$, whose probability density function (PDF) is

$$f_{G(0,1)}(x) = \exp(-x)\exp(-\exp(-x)), \tag{8}$$

the distribution of $g_2 - g_1$ can be calculated by convolution, resulting in a logistic distribution $\mathrm{Logistic}(0,1)$ with PDF

$$f_{\mathrm{Logistic}(0,1)}(x) = \frac{\exp(-x)}{(1+\exp(-x))^2}. \tag{9}$$

Thus, the probability of $1/y_1$ being greater than $M_1 = 1/\epsilon_1$ is

$$P_{g_2-g_1}\left(\frac{1}{y_1} > M_1\right) = 1 - \frac{M_1 - 1}{\exp(x_2 - x_1) + (M_1 - 1)}. \tag{10}$$

Letting this probability be greater than $1 - \epsilon_2$, the x_1, x_2 should follow the restriction

$$\exp(x_2 - x_1) > (M_1 - 1)(M_2 - 1), \tag{11}$$

where $M_2 = 1/\epsilon_2$. This indicates that the upstream network should produce confident logits $\boldsymbol{x}$, as applying softmax to $\boldsymbol{x}$ results in

$$\pi_1 = \frac{\exp x_1}{\exp x_1 + \exp x_2} < \frac{1}{(M_1 - 1)(M_2 - 1) + 1} < \frac{2}{M_1 M_2} = 2\epsilon_1\epsilon_2, \tag{12}$$

when $\epsilon_1 + \epsilon_2 < 0.5$.

Based on the above analysis, it can be inferred that a converged model using the Gumbel-Softmax on its feature $\boldsymbol{x}$ instead of the vanilla softmax will constrain the logits $\boldsymbol{x}$ to produce one-hot-like probability vectors.

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

$$\Delta^3 = \{\boldsymbol{\pi} |\, \pi_i \geq 0, \sum_{i=1}^{4} \pi_i = 1, i = 1,2,3,4\}. \tag{13}$$

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 (type-$1, 2, 3$). 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.

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

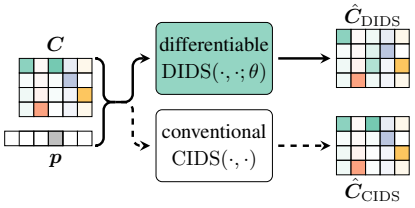


Figure 1: The differentiable IDS channel. The $\hat{\boldsymbol{C}}_{\mathrm{DIDS}}$ and $\hat{\boldsymbol{C}}_{\mathrm{CIDS}}$ are generated by the differentiable and conventional IDS channels, respectively. Optimizing the difference between $\hat{\boldsymbol{C}}_{\mathrm{DIDS}}$ and $\hat{\boldsymbol{C}}_{\mathrm{CIDS}}$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

$\boldsymbol{C}$. 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

$$\mathcal{L}_{\mathrm{KLD}}(\hat{\boldsymbol{C}}_{\mathrm{DIDS}}, \hat{\boldsymbol{C}}_{\mathrm{CIDS}}) = \frac{1}{k} \sum_{i} \hat{\boldsymbol{\pi}}_{i\mathrm{CIDS}}^{T} \log \frac{\hat{\boldsymbol{\pi}}_{i\mathrm{CIDS}}}{\hat{\boldsymbol{\pi}}_{i\mathrm{DIDS}}}. \tag{14}$$

By optimizing Equation (14) 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 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 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 sink 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 Gumbel-Softmax during the training phase and $\arg\max$ 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-

[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.

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

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 sink sequence $\hat{\boldsymbol{s}} = f_{\mathrm{de}}(\hat{\boldsymbol{c}};\psi)$.

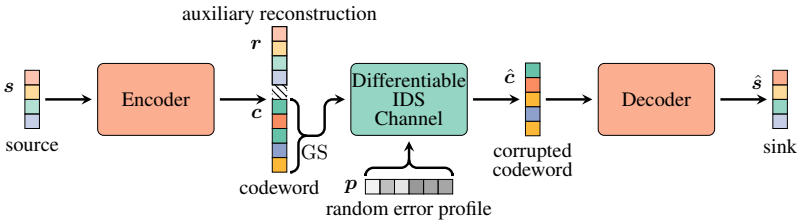


Figure 2: The flowchart of THEA-Code, including the deep learning-based encoder, the pretrained IDS channel, and the decoder.

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

$$\mathcal{L}_{\mathrm{CE}}(\hat{\boldsymbol{s}}, \boldsymbol{s}) = -\sum_{i}\sum_{j} \mathbb{1}_{j=s_i} \log \hat{s}_{ij}, \tag{15}$$

which evaluates the reconstruction disparity of the source sequence $\boldsymbol{s}$ by the sink sequence $\hat{\boldsymbol{s}}$.

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 resembles a chicken-and-egg dilemma, 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 a 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

$$\mathcal{L}_{\mathrm{Aux}}(\boldsymbol{r}, \boldsymbol{s}) = -\sum_{i}\sum_{j} \mathbb{1}_{j=s_i} \log r_{ij}, \tag{16}$$

which quantifies the difference between the reconstruction $\boldsymbol{r}$ and the input sequence $\boldsymbol{s}$.

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 (16) 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

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}_{\mathrm{KLD}}(\hat{\boldsymbol{C}}_{\mathrm{DIDS}}, \hat{\boldsymbol{C}}_{\mathrm{CIDS}}) \tag{17}$$

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 (15) and Equation (16),

$$\hat{\phi}, \hat{\psi} = \arg\min_{\phi,\psi} \mathcal{L}_{\mathrm{CE}}(\hat{\boldsymbol{s}}, \boldsymbol{s}) + \mu \mathcal{L}_{\mathrm{Aux}}(\boldsymbol{r}, \boldsymbol{s}), \tag{18}$$

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 $\arg\max$ function is then applied to convert $\boldsymbol{c}$ into a discrete letter 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 sink sequence $\hat{\boldsymbol{s}}$. Finally, metrics are computed to measure the differences between the original source sequence $\boldsymbol{s}$ and the reconstructed sink 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, we do not distinguish between a training set and a testing set. For example, in the context of this work, the source sequence is a 100-long 4-ary sequence, approximately providing 1.6×10^{60} possible sequences. Given this vast space, sets of randomly generated sequences using different seeds are highly unlikely to overlap.

6 EXPERIMENTS AND ABLATION STUDY

Commonly used methods for synthesizing DNA molecules in DNA storage pipelines typically yield sequences of lengths ranging from 100 to 200. 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 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

$$\mathrm{NER}(\boldsymbol{s}, \hat{\boldsymbol{s}}) = \frac{\#\{s_i \neq \hat{s}_i\}}{\#\{s_i\}}. \tag{19}$$

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.

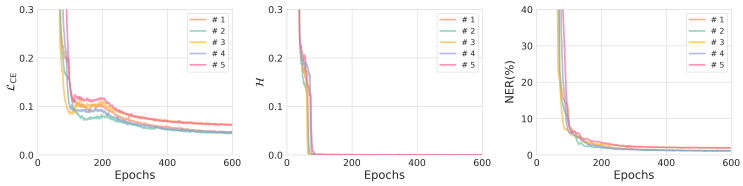
The source code is uploaded at `https://anonymous.4open.science/r/THEA-Code`, and will be made publicly accessible upon the manuscript's publication.

6.1 EFFECTS OF THE GUMBEL-SOFTMAX DISCRETIZATION CONSTRAINT

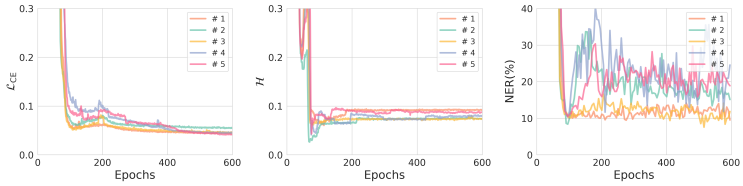
The ablation study on utilizing the Gumbel-Softmax discretization constraint was conducted to observe its impact on the codeword discreteness. During training, the entropy of the codewords

$$H(\boldsymbol{\pi}) = -\sum_{i=1}^{k} \pi_i \log \pi_i \tag{20}$$

was recorded. This entropy measures the level of discreteness in the codewords. Lower entropy implies a distribution that is closer to a one-hot style probability vector, which indicates greater discreteness. In addition to entropy, two other metrics were also recorded, as they are the reconstruction loss $\mathcal{L}_{\text{CE}}$ and the NER. The results, plotted in Figure 3, compare the default Gumbel-Softmax setting against a vanilla softmax approach.



(a) Gumbel-Softmax



(b) vanilla softmax

Figure 3: The reconstruction loss, codeword entropy, and validation NER comparing the Gumbel-Softmax setting against a vanilla softmax approach. Each subfigure shows 5 experiment runs.

The first column of Figure 3 indicates that using Gumbel-Softmax marginally increases the reconstruction loss $\mathcal{L}_{\text{CE}}$ in the continuous mode, which is expected since Gumbel-Softmax introduces additional noise into the system. When comparing the average entropy $\mathcal{H}$ of the learned codeword, applying Gumbel-Softmax significantly reduces the entropy, suggesting that the codewords behave more like one-hot vectors. This lower entropy reflects increased discreteness in the codewords. The NER is calculated in the discrete mode by replacing the Gumbel-Softmax or softmax with an $\arg\max$ operation on the codewords. The third column clearly shows that when codewords are closer to a one-hot style, the model is more consistent between the continuous and discrete modes, leading to better performance during the testing phase.

6.2 Gradients to the differentiable IDS channel

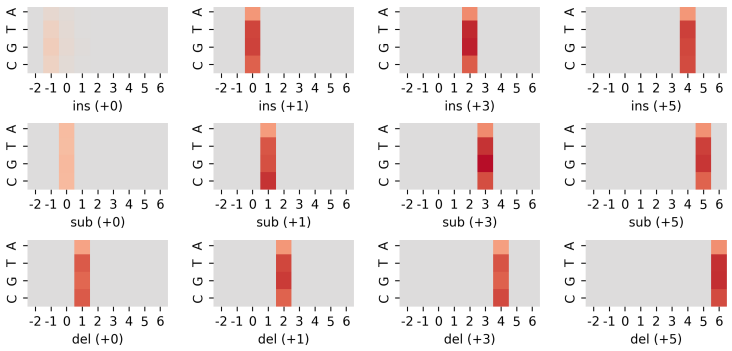


Figure 4: The averaged absolute gradients with respect to the input $\boldsymbol{c}$ over 100 runs. The corresponding IDS operations were performed at an aligned $\text{index} = 0$ by the simulated differentiable IDS channel, the gradients were back-propagated from position $+k$ of the channel output $\hat{\boldsymbol{c}}$. It is suggested that the gradients identify their corresponding position in the input: $+k-1$ for insertion, $+k$ for substitution, and $+k+1$ for deletion.

To investigate whether the simulated IDS channel back-propagates the gradient reasonably, the channel output $\hat{\boldsymbol{c}} = \text{DIDS}(\boldsymbol{c})$ is modified by altering one base to produce $\hat{\boldsymbol{c}}'$. The absolute values of the gradients of $\mathcal{L}(\hat{\boldsymbol{c}}, \hat{\boldsymbol{c}}')$ with respect to the input $\boldsymbol{c}$ after back-propagation are presented in Figure 4. For

instance, subfigure $\mathrm{del}(+3)$ indicates that the IDS channel modifies $\boldsymbol{c}$ to $\hat{\boldsymbol{c}}$ by performing a deletion at index 0. The output $\hat{\boldsymbol{c}}$ is then manually modified by applying a substitution at position $+3$. The gradients of $\mathcal{L}(\hat{\boldsymbol{c}}, \hat{\boldsymbol{c}}')$ with respect to $\boldsymbol{c}$ are plotted over the window $[-2, +6]$.

It is suggested in Figure 4 that the proposed differentiable IDS channel back-propagates gradients reasonably. The gradients shift by one base to the left (resp. right) when the IDS channel performs an insertion (resp. deletion) on $\boldsymbol{c}$. When the IDS channel operates $\boldsymbol{c}$ with a substitution, the gradients stay at the same index. This behavior demonstrates that the channel is able to trace the gradients through the IDS operations. Specifically, in the case $\mathrm{ins}(+0)$, the channel-inserted base in $\hat{\boldsymbol{c}}$ at idx is manually modified. As a result, no specific base in $\boldsymbol{c}$ has a connection to the manually modified base, leading to a diminished gradient in this scenario.

Additionally, we have also conducted experiments on gradients with respect to codewords and empty profiles, presented in Appendix B.

6.3 Performance with different channel settings

Table 1: The testing NER for different source lengths ℓ_s. The source length ℓ_s ranges from 50 to 125. Best NER among 5 runs are reported in row NER*, while the average NER and standard derivative are reported in row NER in format $\mathrm{mean} \pm \mathrm{std}$.

ℓ_s	50	75	100	125
coderate	0.33	0.50	0.67	0.83
NER*(%)	0.09	0.46	1.06	2.81
NER(%)	0.12 ± 0.03	0.51 ± 0.03	1.15 ± 0.08	3.71 ± 0.59

The coderate 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 ℓ_c fixed. For each source length ℓ_s, the experiments were conducted 5 times, with results reported in Table 1. The results reveal a trend where the NER increases from 0.09% to 2.81% as the coderate 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 storage (Press et al., 2020; Pfister & Tal, 2021; Yan et al., 2022; Welzel et al., 2023), a complete solution for DNA storage is achieved. Here, we focus specifically on the IDS-correcting code.

Results on complex IDS channels. 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.

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

[3]These settings simplify DNA 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).

Table 2: The testing NER across different channels. Each number represents the NER of a code tested with its column channel, while trained with its row channel. The average NER and standard derivative are reported in format $\text{mean} \pm \text{std}$ over 5 training runs of each channel.

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

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.

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.

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

Ch-Error	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**	2.9	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**

6.4 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 et al., 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 present a subset of the comparison results in Table 4, with detailed configurations and results across multiple channels provided in Appendix A.

Table 4: The testing error rates compared with different established codes. The coderates are not fully aligned, as HEDGES supports only a limited set of fixed coderates. The experiments on Cai have only the coderates matched, with the code length determined according to the coderate. The experiments on DNA-LM have code length around 150, and the coderates were tuned by varying the number of segments.

coderate	0.33	0.50	0.6	0.67	0.75	0.83
Cai	0.44	1.00	-	2.53	-	8.65
DNA-LM	0.70	1.32	-	3.11	-	9.10
HEDGES	0.28	0.25	0.65	-	3.43	-
THEA-Code	0.09	0.46	-	1.06	-	2.81

Table 4 demonstrates the effectiveness of the proposed method. The performance of THEA-Code and HEDGES outperforms 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 a lower error rate at a higher code rate, specifically 2.81% error rate at 0.83 coderate for THEA-Code v.s. 3.43% error rate at 0.75 coderate for HEDGES.

REFERENCES

Mohamed Akrout, Amal Feriani, Faouzi Bellili, Amine Mezghani, and Ekram Hossain. Domain generalization in machine learning models for wireless communications: Concepts, state-of-the-art, and open issues. *IEEE Communications Surveys & Tutorials*, 2023.

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

Eren Balevi and Jeffrey G. Andrews. Autoencoder-based error correction coding for one-bit quantization. *IEEE Transactions on Communications*, 68(6):3440–3451, 2020. doi: 10.1109/TCOMM.2020.2977280.

Daniella Bar-Lev, Tuvi Etzion, and Eitan Yaakobi. 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, 2023. doi: 10.1109/TIT.2022.3227128.

Meinolf Blawat, Klaus Gaedke, Ingo Hütter, Xiao-Ming Chen, Brian Turczyk, Samuel Inverso, Benjamin W. Pruitt, and George M. Church. Forward error correction for DNA data storage. *Procedia Computer Science*, 80:1011 – 1022, 2016. ISSN 1877-0509. International Conference on Computational Science 2016, ICCS 2016, 6-8 June 2016, San Diego, California, USA.

Kui Cai, Yeow Meng Chee, Ryan Gabrys, Han Mao Kiah, and Tuan Thanh Nguyen. 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, 2021.

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

Sebastian Cammerer, Tobias Gruber, Jakob Hoydis, and Stephan Ten Brink. Scaling deep learning-based decoding of polar codes via partitioning. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–6. IEEE, 2017.

Weigang Chen, Mingzhe Han, Jianting Zhou, Qi Ge, Panpan Wang, Xinchen Zhang, Siyu Zhu, Lifu Song, and Yingjin Yuan. An artificial chromosome for data storage. *National Science Review*, 8 (5):nwab028, 02 2021. ISSN 2095-5138. doi: 10.1093/nsr/nwab028.

Yoni Choukroun and Lior Wolf. Error correction code transformer. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 38695–38705. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/fcd3909db30887ce1da519c4468db668-Paper-Conference.pdf.

Yoni Choukroun and Lior Wolf. Denoising diffusion error correction codes. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=rLwC0_MG-4w.

Yoni Choukroun and Lior Wolf. A foundation model for error correction codes. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=7KDuQPrAF3.

Yoni Choukroun and Lior Wolf. Deep quantum error correction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 64–72, 2024b.

Yoni Choukroun and Lior Wolf. Learning linear block error correction codes. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024c. URL https://openreview.net/forum?id=Kf9CqdI8Rb.

George M Church, Yuan Gao, and Sriram Kosuri. Next-generation digital information storage in DNA. *Science*, 337(6102):1628–1628, 2012.

M.C. Davey and D.J.C. Mackay. Reliable communication over channels with insertions, deletions, and substitutions. *IEEE Transactions on Information Theory*, 47(2):687–698, 2001. doi: 10.1109/18.910582.

Yiming Dong, Fajia Sun, Zhi Ping, Qi Ouyang, and Long Qian. DNA storage: research landscape and future prospects. *National Science Review*, 7(6):1092–1107, 01 2020. ISSN 2095-5138. doi: 10.1093/nsr/nwaa007.

Alex El-Shaikh, Marius Welzel, Dominik Heider, and Bernhard Seeger. High-scale random access on DNA storage systems. *NAR Genomics and Bioinformatics*, 4(1):lqab126, 01 2022. ISSN 2631-9268. doi: 10.1093/nargab/lqab126.

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

Ryan Gabrys, Venkatesan Guruswami, João Ribeiro, and Ke Wu. Beyond single-deletion correcting codes: Substitutions and transpositions. *IEEE Transactions on Information Theory*, 69(1):169–186, 2023. doi: 10.1109/TIT.2022.3202856.

Nick Goldman, Paul Bertone, Siyuan Chen, Christophe Dessimoz, Emily M. LeProust, Botond Sipos, and Ewan Birney. Towards practical, high-capacity, low-maintenance information storage in synthesized DNA. *Nature*, 494(7435):77–80, 2013. doi: 10.1038/nature11875.

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*, 54(8):2552–2555, 2015. doi: 10.1002/anie.201411378.

Emil Julius Gumbel. *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office, 1954.

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

Belaid Hamoum, Elsa Dupraz, Laura Conde-Canencia, and Dominique Lavenier. Channel model with memory for dna data storage with nanopore sequencing. In *2021 11th International Symposium on Topics in Coding (ISTC)*, pp. 1–5. IEEE, 2021.

Ichiro Hirao, Yoshifumi Nishimura, Yon-ichi Tagawa, Kimitsuna Watanabe, and Kin-ichiro Miura. 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, 1992.

Iris A. M. Huijben, Wouter Kool, Max B. Paulus, and Ruud J. G. van Sloun. A review of the gumbel-max trick and its extensions for discrete stochasticity in machine learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1353–1371, 2023. doi: 10.1109/TPAMI.2022.3157042.

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

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=rkE3y85ee`.

Yihan Jiang, Hyeji Kim, Himanshu Asnani, Sreeram Kannan, Sewoong Oh, and Pramod Viswanath. Turbo autoencoder: Deep learning based channel codes for point-to-point communication channels. In *Advances in Neural Information Processing Systems*, pp. 2754–2764, 2019.

Hyeji Kim, Yihan Jiang, Ranvir B Rana, Sreeram Kannan, Sewoong Oh, and Pramod Viswanath. Communication algorithms via deep learning. In *International Conference on Learning Representations*, 2018.

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

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

Issam Maarouf, Andreas Lenz, Lorenz Welter, Antonia Wachter-Zeh, Eirik Rosnes, and Alexandre Graell i Amat. Concatenated codes for multiple reads of a dna sequence. *IEEE Transactions on Information Theory*, 69(2):910–927, 2022.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=S1jE5L5gl.

Ashok V Makkuva, Xiyang Liu, Mohammad Vahid Jamali, Hessam Mahdavifar, Sewoong Oh, and Pramod Viswanath. Ko codes: inventing nonlinear encoding and decoding for reliable wireless communication via deep-learning. In *International Conference on Machine Learning*, pp. 7368–7378. PMLR, 2021.

Linda C Meiser, Julian Koch, Philipp L Antkowiak, Wendelin J Stark, Reinhard Heckel, and Robert N Grass. Dna synthesis for true random number generation. *Nature communications*, 11(1):5869, 2020.

Michael Mitzenmacher. A survey of results for deletion channels and related synchronization channels. *Probability Surveys*, 6(none):1 – 33, 2009. doi: 10.1214/08-PS141. URL https://doi.org/10.1214/08-PS141.

Eliya Nachmani and Lior Wolf. Hyper-graph-network decoders for block codes. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/a9be4c2a4041cadbf9d61ae16dd1389e-Paper.pdf.

Eliya Nachmani, Elad Marciano, Loren Lugosch, Warren J. Gross, David Burshtein, and Yair Be'ery. Deep learning methods for improved decoding of linear codes. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):119–131, 2018. doi: 10.1109/JSTSP.2017.2788405.

Lee Organick, Siena Dumas Ang, Yuan-Jyue Chen, Randolph Lopez, Sergey Yekhanin, Konstantin Makarychev, Miklos Z Racz, Govinda Kamath, Parikshit Gopalan, Bichlien Nguyen, et al. Random access in large-scale DNA data storage. *Nature Biotechnology*, 36(3):242–248, 2018.

Henry D. Pfister and Ido Tal. Polar codes for channels with insertions, deletions, and substitutions. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pp. 2554–2559, 2021. doi: 10.1109/ISIT45174.2021.9517755.

William H. Press, John A. Hawkins, Stephen K. Jones, Jeffrey M. Schaub, and Ilya J. Finkelstein. HEDGES error-correcting code for DNA storage corrects indels and allows sequence constraints. *Proceedings of the National Academy of Sciences*, 117(31):18489–18496, 2020. doi: 10.1073/pnas.2004821117.

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

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

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

Sundara Rajan Srinivasavaradhan, Sivakanth Gopi, Henry D Pfister, and Sergey Yekhanin. Trellis bma: Coded trace reconstruction on ids channels for dna storage. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pp. 2453–2458. IEEE, 2021.

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

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 5998–6008. Curran Associates, Inc., 2017.

Lorenz Welter, Roman Sokolovskii, Thomas Heinis, Antonia Wachter-Zeh, Eirik Rosnes, et al. An end-to-end coding scheme for dna-based data storage with nanopore-sequenced reads. *arXiv preprint arXiv:2406.12955*, 2024.

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*, 14(1):628, 2023.

Zihui Yan, Cong Liang, and Huaming Wu. A segmented-edit error-correcting code with re-synchronization function for DNA-based storage systems. *IEEE Transactions on Emerging Topics in Computing*, pp. 1–13, 2022. doi: 10.1109/TETC.2022.3225570.

A Comparison Experiments

To evaluate the effectiveness of the proposed methods, we conducted comparison experiments against three prior works, which are:

- a combinatorial code that can correct single IDS errors over a 4-ary alphabet from Cai (Cai et al., 2021);
- a segmented method for correcting multiple IDS errors, called DNA-LM from (Yan et al., 2022);
- a well-known, efficient heuristic method, called HEDGES, from a DNA storage research (Press et al., 2020).

These methods typically offer only a few discrete, fixed configurations. We made efforts to align their settings as closely as possible. For Cai's combinatorial code, the coderates are fixed based on the code lengths. In our experiments on Cai, only the coderates are matched, with the code length determined according to the coderate[4]. For DNA-LM, we maintained the codeword length around 150, adjusting the number of segments to match coderates. For HEDGES, only binary library is publicly available, and it supports fixed coderates in $\{0.75, 0.6, 0.5, 1/3, 0.25, 1/6\}$. HEDGES' inner code was tested independently for comparison. We list all the source lengths ℓ_s, codeword lengths ℓ_c, and coderate r used in the experiments in Table 5.

Table 5: The testing configurations for the comparison experiments. Each cell includes the code rate, message length, and code length. The settings are tried to be aligned, except the Cai configuration has a code length that does not align with 150, and HEDGES uses fixed coderates of 0.60 and 0.75, which are not aligned.

	$r_1 = \ell_{s1}/\ell_{c1}$	$r_2 = \ell_{s2}/\ell_{c2}$	$r_3 = \ell_{s3}/\ell_{c3}$	$r_4 = \ell_{s4}/\ell_{c4}$
Cai	0.33=7/21	0.50=16/32	0.67=32/48	0.83=85/102
DNA-LM	0.34=50/148	0.51=77/152	0.68=96/142	0.84=124/148
HEDGES	0.34=52/155	0.50=76/152	*0.60=92/153*	*0.75=115/155*
THEA-Code	0.33=50/150	0.50=75/150	0.67=100/150	0.83=125/150

The experiments were conducted on the default IDS channel with 1% error probability, as well as its variations, Asc and Des, introduced in Section 6.3. The results is illustrated in Figure 5. The experiments handled failed corrections by directly using the corrupted codeword as the decoded message.

[4]It is important to note that code length plays a critical role in these experiments, as longer codewords are more likely to encounter multiple errors that cannot be corrected. Thus, Cai's performance here is just a baseline statistic of multi-errors with respect to the length, and performance may degrade with increased length.

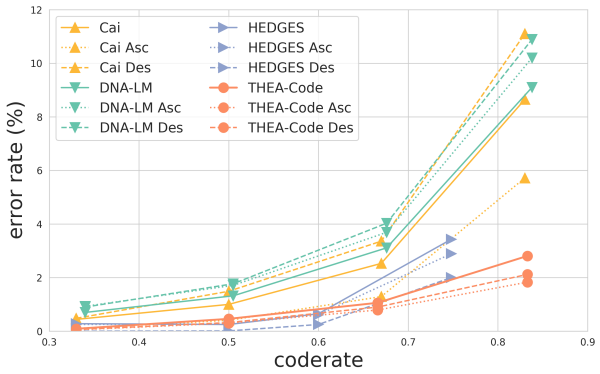


Figure 5: The error rates of the comparison experiments. Results for Cai, DNA-LM, HEDGES, and THEA-Code are shown across Hom, Asc, and Des channels, with respect to their coderates.

The results for Cai's method indicate that directly applying classical combinatorial codes to a 1% IDS error probability channel with a codeword length of 150 is impractical. The observed error rates are high, even though these values were obtained with shorter code lengths than 150. The segmented method with sync markers in DNA-LM supports a codeword length of 150 and can correct multiple errors across different segments. However, it also exhibits a high error rate, indicating a nonnegligible likelihood of multi-errors occurring within the same segment. For HEDGES, while the results are commendable, the coderate is restricted to a limited set of fixed values. The results of THEA-Code demonstrate the effectiveness of the proposed method. At lower code rates, THEA-Code achieves a comparable error rate to HEDGES. At higher code rates, the proposed method outperforms HEDGES, achieving a lower error rate at a higher code rate, specifically 2.81% error rate at 0.83 coderate for THEA-Code v.s. 3.43% error rate at 0.75 coderate for HEDGES.

B More on the gradients to differentiable IDS channel

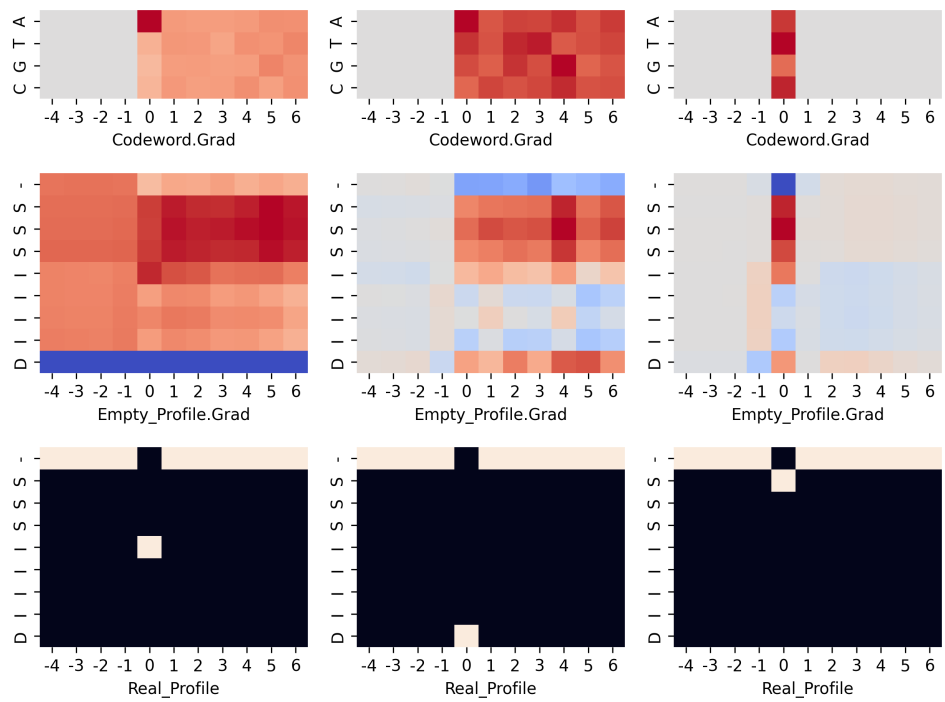
In Section 6.2, we illustrated that the differentiable IDS channel can effectively trace gradients through the IDS operations. In this section, we focus on evaluating the channel's capability to recover the error profile through gradient-based optimization.

Given a codeword $\boldsymbol{c}$, an empty profile $\boldsymbol{p}_0$ which defines the identity transformation of the IDS channel such that $\hat{\boldsymbol{c}} = \boldsymbol{c} = \mathrm{DIDS}(\boldsymbol{c}, \boldsymbol{p}_0)$, and a modified codeword $\hat{\boldsymbol{c}}'$ which is produced by manually modifying $\boldsymbol{c}$ through an insertion, deletion, or substitution at position idx, the gradients of $\mathcal{L}(\hat{\boldsymbol{c}}, \hat{\boldsymbol{c}}')$ are computed with respect to both the input codeword $\boldsymbol{c}$ and the empty profile $\boldsymbol{p}_0$. The average gradients, calculated over 100 runs, are plotted in Figure 6 with position idx aligned to 0.

In Figure 6, it is suggested that, when performing an insertion or deletion, the gradients with respect to the codeword are distributed after the error position idx. This aligns with the fact that synchronization errors (insertions or deletions) can be interpreted as successive substitutions starting from the error position, especially when the actual error profile is unknown. When performing a substitution, the gradients naturally concentrate at the error position idx.

Regarding the empty profile, $\boldsymbol{p}_0 = \mathbf{0}$, the gradients also exhibit meaningful patterns. For an insertion, the substitution area after idx is lighted by the gradients, supporting the view that an insertion can be seen as a sequence of substitutions if error constraints are absent. Additionally, the insertion area of the profile is also lighted, which makes sense since an insertion may also be interpreted as a series of substitutions followed by an ending insertion. For deletion errors, similar patterns are observed: the gradients are distributed in the areas of substitutions and deletions after the error position idx, since the deletion can also be viewed as a series of substitutions, or as several substitutions and an ending deletion. For substitution errors, the gradients again concentrate at the error position idx, as substitutions do not cause sequence mismatches.

Utilizing energy constraints on the profile may be helpful for specific profile applications. In this work, only the gradients with respect to the codeword participate in the training phase, the existing version of the simulated differentiable IDS channel is assumed to be adequate.

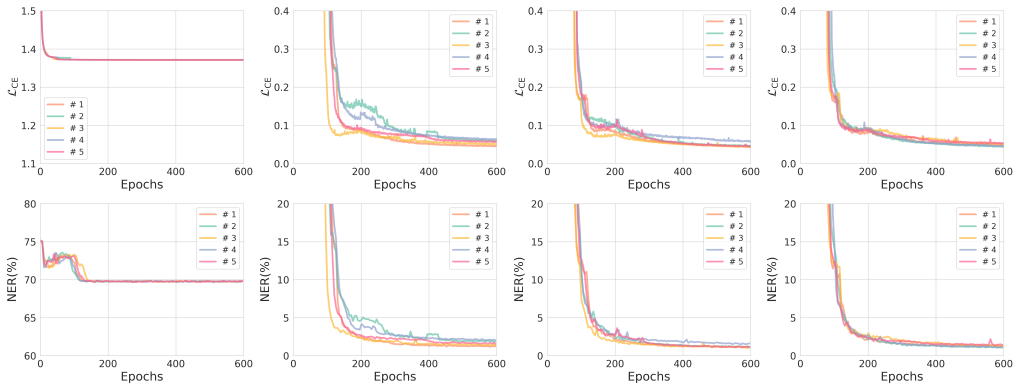


(a) insertion (b) deletion (c) substitution

Figure 6: The gradient distribution with respect to the input codeword and the empty profile, when the output codeword is manually modified. The figures display the averaged gradients over 100 runs, visualizing how the gradients were back-propagated in different cases of insertions, deletions, and substitutions in the output codeword.

C Ablation study on the auxiliary reconstruction loss

C.1 Effects of the auxiliary reconstruction loss



(a) $\mu = 0$ (b) $\mu = 0.5$ (c) $\mu = 1$ (d) $\mu = 1.5$

Figure 7: The reconstruction loss $\mathcal{L}_{\text{CE}}$ between the source and sink sequences, and the validation NER for various choices of $\mu \in \{0, 0.5, 1, 1.5\}$. Each curve in the subfigures represents one of the 5 runs conducted in the experiment and is plotted against the training epochs.

Experiments with different choices of the hyperparameter μ were conducted, which are $\mu = 0$ indicating the absence of the auxiliary reconstruction loss, and $\mu \in \{0.5, 1, 1.5\}$ for different weights for the auxiliary loss. The validation NER and the reconstruction loss between source and sink

sequences are plotted against the training epochs. Additionally, we have also explored on different types of auxiliary reconstruction sequences, results presented in Appendix C.2.

The first column of Figure 7 indicates that without the auxiliary loss, all 5 runs of the training fail, producing random output. By comparing the first column with the other three, the effectiveness of introducing the auxiliary loss can be inferred. In the subfigures corresponding to $\mu \in \{0.5, 1, 1.5\}$, all the models converge well, and the NERs also exhibit a similar convergence. This suggests the application of the auxiliary loss is essential, but the weight of this loss has minimum influence on the final performance.

C.2 Auxiliary loss on patterns beyond sequence reconstruction

In Appendix C.1, the necessity of introducing a auxiliary reconstruction task to the encoder is verified. After these experiments, a natural question arises: How about imparting the encoder with higher initial logical ability through a more complicated task rather than replication? Motivated by this, we adopted commonly used operations from existing IDS-correcting codes and attempted to recover the sequence from these operations using the encoder. In practice, we employed the forward difference $\mathrm{Diff}(\boldsymbol{s})$, where

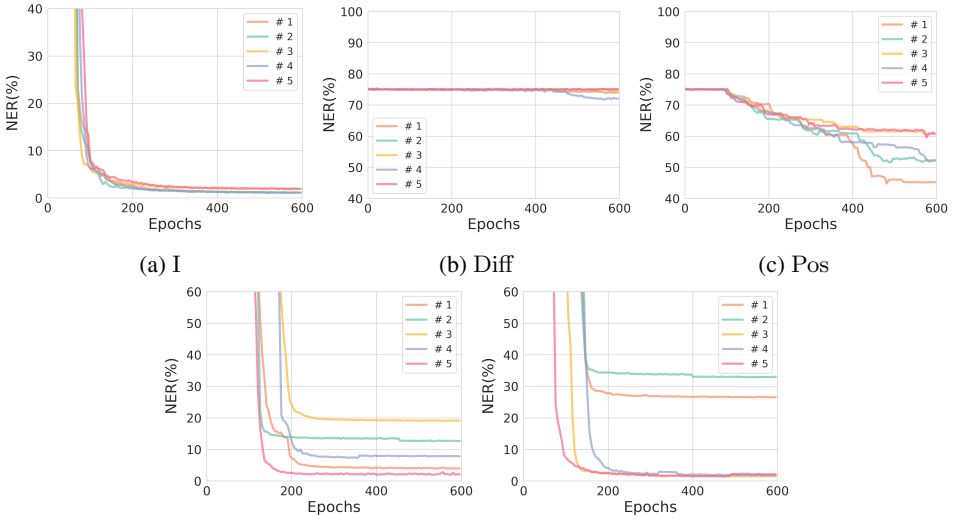
$$\mathrm{Diff}(\boldsymbol{s})_i = s_i - s_{i+1} \mod 4, \tag{21}$$

the position information-encoded sequence $\mathrm{Pos}(\boldsymbol{s})$, where

$$\mathrm{Pos}(\boldsymbol{s})_i = s_i + i \mod 4, \tag{22}$$

and their combinations as the reconstructed sequences.

The evaluation NERs against training epochs are plotted in Figure 8 under different combinations of the identity mapping I, Diff, and Pos. It is clear that the reconstruction of the identity mapping I outperforms Diff and Pos. Introducing the identity mapping I to Diff and Pos helps improving the convergence of the model, but final results have illustrated that they are still worse than simple applying the identify mapping I as the auxiliary task. These variations may be attributed to the capabilities of the transformers in our setting or the disordered implicit timings during training.



(a) I (b) Diff (c) Pos

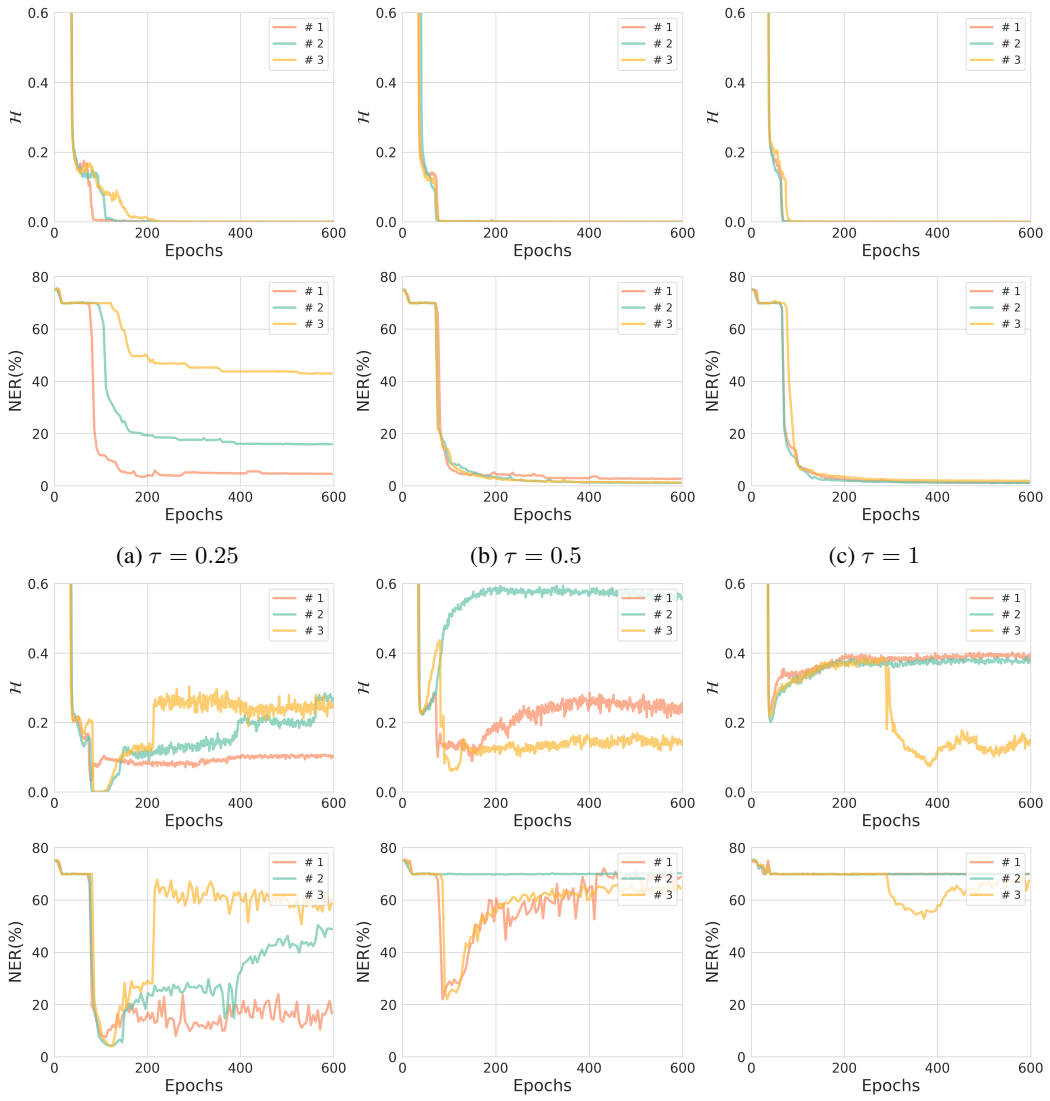
(d) I + Diff (e) I + Pos

Figure 8: The validation NER against the training epochs with different choices of auxiliary reconstruction. The reconstructed sequences are produced by combinations of the identity mapping I, Diff, and Pos, where + denotes sequence concatenating.

D Optimization of hyperparameter temperature τ in the Gumbel-Softmax formula

To examine the impact of different temperature values τ in Equation (2), experiments were conducted with various settings of $\tau \in \{0.25, 0.5, 1, 2, 4, 8\}$. Since the Gumbel-Softmax Discretization Constraint is designed to encourage greater discretization of the codeword, the codeword entropy $\mathcal{H}$, as defined in Equation (20), and the validation NER were tracked throughout the training phase

As shown in Figure 9, lower temperature ($\tau = 0.25$) has an effect in discretization, but result in unstable and poor model performance, while higher temperatures ($\tau \in \{2, 4, 8\}$) lead to both poor discretization and high NER.



(a) $\tau = 0.25$ (b) $\tau = 0.5$ (c) $\tau = 1$

(d) $\tau = 2$ (e) $\tau = 4$ (f) $\tau = 8$

Figure 9: The codeword entropy $\mathcal{H}$ and the validation NER for various choices of $\tau \in \{0.25, 0.5, 1, 2, 4, 8\}$. Each curve in the subfigures represents one of the 3 runs conducted in the experiment and is plotted against the training epochs.

E TRANSFORMER, COMPLEXITY, AND TIME CONSUMPTION

Transformers (Vaswani et al., 2017), well-known deep learning architectures, rely on the attention mechanism. Each head of a Transformer model processes features according to the following formula:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{23}$$

In this work, each layer comprises 16 attention heads with an embedding dimension 512, and a total of $3+3$ attention layers are used for the sequence-to-sequence model. Both the encoder and decoder are implemented as such sequence-to-sequence models. For the differentiable IDS channel, a $1+1$ layered sequence-to-sequence model is employed.

Since attention is calculated globally over the sequence in Equation (23), it has a complexity of $O(n^2)$. Without delving into the many efficient transformer architectures, the time consumption was measured by decoding $1,280,000$ codewords using an RTX3090. The encoder, which shares the same structure, exhibits similar performance. The results are acceptable and are presented in Table 6.

Table 6: Time consumption of decoding $1,280,000$ codewords for each source length ℓ_s by an RTX3090.

source length	$\ell_s = 50$	$\ell_s = 75$	$\ell_s = 100$	$\ell_s = 125$
time (s)	521.94	573.87	623.92	687.76