Learning Linear Block Error Correction Codes

Yoni Choukroun [1] **Lior Wolf** [1]

Abstract

Error correction codes are a crucial part of the physical communication layer, ensuring the reliable transfer of data over noisy channels. The design of optimal linear block codes capable of being efficiently decoded is of major concern, especially for short block lengths. While neural decoders have recently demonstrated their advantage over classical decoding techniques, the neural design of the codes remains a challenge. In this work, we propose for the first time a unified encoder-decoder training of binary linear block codes. To this end, we adapt the coding setting to support efficient and differentiable training of the code for end-to-end optimization over the order two Galois field. We also propose a novel Transformer model in which the self-attention masking is performed in a differentiable fashion for the efficient backpropagation of the code gradient. Our results show that (i) the proposed decoder outperforms existing neural decoding on conventional codes, (ii) the suggested framework generates codes that outperform the analogous conventional codes, and (iii) the codes we developed not only excel with our decoder but also show enhanced performance with traditional decoding techniques.

1. Introduction

In the modern era of information technology, maintaining strong and reliable communication despite interference in transmission channels is a significant priority. It necessitates the development of codes designed for resilient transmission over noisy channels. Among the existing family of codes, linear block codes represent a widely used class of error correction codes (ECC), which benefits from decades of research and design. Neural methods (Nachmani et al., 2016; Gruber et al., 2017; Bennatan et al., 2018; Choukroun & Wolf, 2022) have been applied as successful heuristics to the NP-hard maximum likelihood (ML) problem of decoding. However, creating high-performance codes in conjunction with their decoders remains a challenge, particularly in the contemporary realm of finite-length codes.

Linear block codes are generally designed upon asymptotical analysis based on mathematical principles or upon pseudorandom generation. The Shannon channel capacity theorem (Shannon, 1948) demonstrated first the existence of a coding technique that allows an arbitrarily small error probability under maximum likelihood decoding when the transmission rate remains closely below the channel capacity. Then, the capacity can be asymptotically achieved using random linear block codes, encoded in polynomial time. Polar codes (Arikan, 2008) are a family of codes proven to be capacity-achieving under successive cancellation (SC) decoding with an explicit construction method based on recursive channel combination. Low-Density Parity-Check (LDPC) codes (Gallager, 1962) are provably close to the capacity generally obtained from random sparse bipartite graphs with appropriate degree distribution. LDPC codes are effectively decoded via the efficient Belief Propagation algorithm (Pearl, 1988).

While neural methods have been applied successfully to train decoders of existing codes, attempts to use machine learning for code design are far fewer. There have been attempts to learn non-linear continuous codes along with their neural decoders (O'Shea & Hoydis, 2017; Jiang et al., 2019b), but the very high degree of non-differentiability of the ubiquitous *binary linear block* codes makes their design a major challenge.

Our contributions include (i) showing for the first time that it is possible to directly optimize binary linear block codes along with a neural decoder, in a unified and differentiable fashion.

We optimize the denoising capabilities of a learned code with respect to a neural decoder that is trained jointly. A central question then becomes whether the learned code is tailored for a given neural decoder only, or whether the framework provides a universally good code in some sense. (ii) We present compelling evidence supporting that the optimized code exhibits improved performance compared

[1]The Blavatnik School of Computer Science, Tel Aviv University. Correspondence to: Yoni Choukroun <choukroun.yoni@gmail.com>, Lior Wolf <wolf@cs.tau.ac.il>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

to other codes of the same size, irrespective of the decoder used. This outcome is somewhat unexpected, in light of the significantly non-convex objective associated with the integrated optimization process.

To achieve these innovative results, we make multiple technical contributions: (iii) we adapt the error correction coding setting for efficient differentiable training, (iv) we solve the highly not-differentiable optimization over the binary finite field with binarization and polarization methods, and (v) we improve over the existing ECC Transformer-based models by creating a mask derived from the parity-check matrix in a differentiable manner, allowing the efficient backpropagation of the gradients with respect to the code through the neural decoder's layers.

2. Related Works

Neural decoder contributions generally focus on short and moderate-length codes for two main reasons. First, classical decoders are proven to reach the capacity of the channel for large codes, preventing any potential enhancement. Second, the emergence of applications driven by the Internet of Things created the requirement for optimal decoders of short to moderate-length codes. For example, 5G Polar codes have code lengths of 32 to 1024 (ESTI, 2021).

Previous work on neural decoders is generally divided into two main classes: model-based and model-free. Model-based decoders implement parameterized versions of classical Belief Propagation (BP) decoders, where the Tanner graph is unfolded into an NN in which weights are assigned to each variable edge. This results in an improvement in comparison to the baseline BP method for short codes (Nachmani et al., 2016; Nachmani & Wolf, 2019; Raviv et al., 2020; 2023; Kwak et al., 2023). While model-based decoders benefit from a strong theoretical background, the architecture is overly restrictive, which generally enforces its coupling with high-complexity NN (Nachmani & Wolf, 2021). Also, the improvement gain generally vanishes for more iterations and longer codewords (Hoydis et al., 2022).

Model-free decoders employ general types of neural network architectures. Earlier approaches (Cammerer et al., 2017; Gruber et al., 2017; Kim et al., 2018b) employed stacked fully connected (FC) networks, convolutional neural networks (CNN) (Jiang et al., 2019a) or recurrent neural networks (RNNs) that have difficulties in learning the code, since no prior can be straightforwardly established. Similarly, (Bennatan et al., 2018) extended the classical syndrome decoding by employing a channel output preprocessing, which further adds the magnitude to the syndrome vector such that the decoder remains provably invariant to the codeword, avoiding overfitting the exponential number of codewords. Recently, several works based on the Transformer (Vaswani et al., 2017) architecture have been adapted to ECC. (Choukroun & Wolf, 2022) first introduced the Error Correction Code Transformer (ECCT), obtaining state-of-the-art performance. Subsequently, (Choukroun & Wolf, 2023) extended the denoising diffusion paradigm to ECC, further improving the SOTA by large margins. (Choukroun & Wolf, 2024a) employed the ECCT to syndrome decoding for quantum error correction. Finally, (Choukroun & Wolf, 2024b) proposed a foundation neural decoder, capable of decoding and generalizing to any code, length, and rate, enabling the potential deployment of a single universal neural decoder for every type of code. At the intersection of Transformers and neural BP solutions, (Cammerer et al., 2022) proposed a graph neural network decoder built upon the Tanner graph. Recently, and following (Bennatan et al., 2018; Choukroun & Wolf, 2022; Raviv et al., 2020), (Park et al., 2023) has shown that different parity-check matrices describing the same code provide different performance on the ECCT.

While neural decoders show improved performance in various communication settings, there has been very limited success in the design of novel neural coding methods. Most of the existing works attack the unified training design in a classical deep encoder-decoder fashion (based on FC, CNN, or RNN), where the codes and the modulations are integrated in a fully classical differentiable fashion. (O'Shea & Hoydis, 2017) developed continuous (7,4) code with modulation power constraint matching the Hamming code performance. Joint designs have been proposed for feedback channels (Kim et al., 2018a) and Turbo codes (Kim et al., 2018a). End-to-end training under non-differentiable modulations has been studied in (Aoudia & Hoydis, 2018; Ye et al., 2018)

However, these methods are generally problematic, since they make use of heavy deep learning-based encoding-decoding solutions in the continuous domain, which is far from practical encoding and decoding deployment. Moreover (Jiang et al., 2019b), neural codes remain far from capacity-approaching performance because of the high level of non-differentiability, as well as the difficulties in inducing the code through the neural decoder.

3. Background

We provide the necessary background on error correction coding and the Transformer architectures for ECC.

3.1. Coding

We assume a standard transmission protocol for messages $m \in \{0,1\}^k$ using a linear code C, defined by a generator matrix $G \in \{0,1\}^{k\times n}$ and the parity check matrix $H \in \{0,1\}^{(n-k)\times n}$, such that $GH^T = 0$ over the order 2 Galois field $GF(2)$. The parity check matrix H entails what is known as a Tanner graph, which consists of n variable nodes and $(n-k)$ check nodes. The edges of this graph correspond to the on-bits in each column of the matrix H.

The input message $m \in \{0,1\}^k$ is encoded by G to a codeword $x = mG \in C \subset \{0,1\}^n$ satisfying $Hx = 0$ and is transmitted via a Binary-Input Symmetric-Output channel, e.g., an additive white Gaussian noise (AWGN) channel. Let y denote the channel output represented as $y = x_s + \varepsilon$, where x_s denotes the Binary Phase Shift Keying (BPSK) modulation of x (i.e., over $\{\pm 1\}$), and ε is a random noise independent of the transmitted x. The main goal of the decoder $f : \mathbb{R}^n \to \mathbb{R}^n$ is to provide a soft approximation $\hat{x} = f(y)$ of the codeword.

Following Bennatan et al. (2018); Choukroun & Wolf (2022; 2023), the surrogate decoder objective is defined by the prediction of the equivalent multiplicative noise $\tilde{\varepsilon}$ such that $y = x_s \odot \tilde{\varepsilon}$, with $\odot$ the Hadamard product. Extending classical syndrome decoding, the decoder preprocesses the channel output y by concatenating the provably *codeword-independent* magnitude and syndrome vectors, such that $h(y) := [|y|, s(y)] \in \mathbb{R}^{2n-k}$, where $[\cdot,\cdot]$ denotes vector/matrix concatenation and $s(y)$ denotes the syndrome defined by $s(y) = H\text{bin}(y)$ with $\text{bin}(y)$ the binary mapping of y. In this case, the soft codeword prediction is given by the denoising task defined as $\hat{x} = f(h(y)) \odot y$.

3.2. Transformers for Error Correction Code

The seminal Transformer architecture was first introduced as a novel, attention-based architecture for machine translation (Vaswani et al., 2017). The input sequence is embedded into a high-dimensional space, coupled with positional embedding for each element. The embeddings are then propagated through multiple normalized self-attention and feed-forward blocks. The self-attention mechanism is based on a trainable associative memory with (key, value) vector pairs, where a query vector $q \in \mathbb{R}^d$ is matched against a set of k key vectors using scaled inner products, such that $A(Q,K,V) = \text{Softmax}(d^{-1/2}(QK^T))V$. Here, $Q \in \mathbb{R}^{N\times d}$, $K \in \mathbb{R}^{k\times d}$ and $V \in \mathbb{R}^{k\times d}$ represent the stacked N queries, k keys and values tensors, respectively. Keys, queries, and values are obtained using linear transformations of representations of the sequence's elements, and a multi-head self-attention scheme is deployed by extending the self-attention mechanism to multiple attention heads.

The Error Correction Code Transformer (Choukroun & Wolf, 2022) is a state-of-the-art neural error decoder. Its initial embedding is defined by encoding $h(y)$, viewed as a sequence of length $2n-k$ where each bit is encoded into a high-dimensional space with its own (position-dependent) embedding vector. To integrate information about the code, a binary masking derived from the parity-check

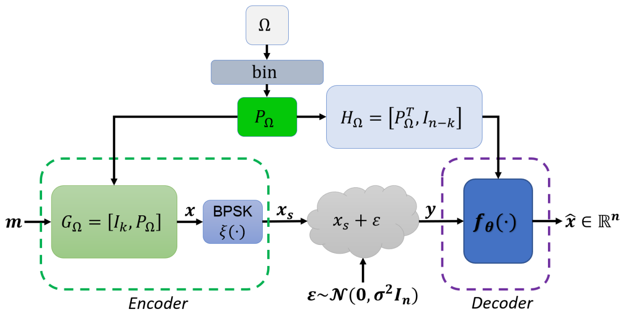

Ω
bin
P_Ω
$H_\Omega = [P_\Omega^T, I_{n-k}]$
m
$G_\Omega = [I_k, P_\Omega]$
x
BPSK $\xi(\cdot)$
x_s
$x_s + \varepsilon$
y
$f_\theta(\cdot)$
$\hat{x} \in \mathbb{R}^n$
$\varepsilon \sim \mathcal{N}(0, \sigma^2 I_n)$
Encoder
Decoder

Figure 1: Illustration of the proposed end-to-end communication system. Our work focuses on the unified design and co-training of the code induced by Ω and of the parameterized decoder f_θ.

H matrix is integrated into the self-attention mechanism $A_H(Q,K,V) = \text{Softmax}(d^{-1/2}(QK^T + g(H)))V$, where $g(H) : \{0,1\}^{(n-k)\times n} \to \{-\infty, 0\}^{(2n-k)\times(2n-k)}$. Specifically, the mask $g(H)$ is obtained as the adjacency matrix of the Tanner graph extended to two-ring connectivity. Finally, the prediction module is implemented with two standard linear layers.

Recently, the Foundation ECCT (FECCT) (Choukroun & Wolf, 2024b) matched the ECCT's performance while being fully code-, length-, and rate-invariant. FECCT enables one to train a single decoder on several codes and demonstrates strong generalization capabilities on unseen codes. The length invariance of the initial embedding is obtained using a single input encoding vector for all the magnitude elements and two position-invariant embedding vectors for representing the binary syndrome elements. The positional embedding, as well as the code, are integrated as *relative* positional encoding into the self-attention via a parameterized soft mapping of the node distances in the Tanner graph, in order to modulate the self-attention tensor such that $A_H(Q,K,V) = \big(\text{Softmax}(d^{-1/2}QK^T) \odot \psi(\mathcal{G}(H))\big)V$, where $\psi(\mathcal{G}(H))$ is the learned mapping of the distances between the nodes of the Tanner graph. Finally, to remain code-invariant, the final prediction module is conditioned on the parity-check matrix H by selecting the relevant variable nodes from the parity-check embeddings.

4. Method

We present the setting of the proposed framework and the elements of the proposed neural decoder, its complete architecture, and the training procedure.

4.1. End-to-End Optimization

We assume the *standard* (also referred to as canonical or systematic) form of the code to ensure its efficient and differentiable optimization. In the standard form, the generator matrix is defined as $G = [I_k, P]$ with $P \in \{0,1\}^{k \times n-k}$ and the parity check matrix is induced as $H = [P^T, I_{n-k}]$. Using a general matrix form for G would allow a greater degree of freedom in design, but the *differentiable* and *fast* design of the corresponding full-rank parity-check matrix defining the code kernel from G would become a challenge.

To obtain trainable codes, we propose to parameterize the matrix P such that

$$P \coloneqq P_\Omega = \text{bin}(\Omega), \tag{1}$$

where $\Omega \in \mathbb{R}^{k \times n-k}$ denotes the *trainable* parameterized version of P and $\text{bin} : \mathbb{R} \to \{0,1\}$ denotes the point-wise binarization function.

Given a neural decoder $f_\theta : \mathbb{R}^{2n-k} \to \mathbb{R}^n$ parameterized by θ, the parameterized generator matrix $G_\Omega = [I_k, P_\Omega]$ and parity check matrix $H_\Omega = [P_\Omega^T, I_{n-k}]$, the objective is now defined in a unified end-to-end encoding-decoding fashion, as opposed to the standard neural decoding optimization task in which only the decoder is trained.

Defining $\phi(\cdot,\cdot)$ the matrix multiplication over $GF(2)$ (i.e., modulo 2) and the bipolar mapping $\xi : \{0,1\} \to \{\pm 1\}$ as $\xi(u) = 1 - 2u, u \in \{0,1\}$, the objective is given by

$$\mathcal{L}(\Omega, \theta) = \mathbb{E}_{m \sim \text{Bern}^k(1/2), \varepsilon \sim \mathcal{Z}} \text{BCE}\big(f_\theta(h_\Omega(y_\Omega)), \text{bin}(\tilde{\varepsilon})\big) \tag{2}$$

Here, $y_\Omega = \xi(\phi(m, G_\Omega)) + \varepsilon$ denotes the parameterized channel output, $h_\Omega(y_\Omega) = [|y_\Omega|, H_\Omega \text{bin}(y_\Omega)]$ is the parameterized codeword-invariant preprocessing, $\mathcal{Z}$ denotes the distribution of the channel noise used for the training, BCE denotes the binary cross entropy loss, and $\tilde{\varepsilon}$ is defined in Sec. 3.1 above. An illustration of the proposed end-to-end communication system is given in Figure 1. Constraints on the code (e.g., sparsity, structure) can be further added to the training objective via its regularization.

While one could argue that the definition and optimization of the code via the decoder only (without the generator) is sufficient via its syndrome computation, we show in Appendix A that the integration of the whole encoding-decoding pipeline (i.e., G and H) is crucial for efficient backpropagation.

4.2. Optimization over $GF(2)$

The major problem in the end-to-end training objective is the use of the highly non-differentiable ϕ and bin functions.

Here, we propose to perform the optimization of the binarization function bin via the straight-through estimator (STE) (Bengio et al., 2013) defined such that

$$\begin{cases} \text{bin}(u) = \xi^{-1}\big(\text{sign}(u)\big) \\ \frac{\partial \text{bin}(u)}{\partial u} = -\frac{\mathbb{1}_{|u| \le \tau}}{2} \end{cases} \tag{3}$$

with τ the thresholding scalar that stops the weights Ω from growing overly large (Courbariaux et al., 2015).

The optimization of ϕ is obtained using a differentiable equivalence mapping of the XOR ($\oplus$; i.e., sum over $GF(2)$) operation using the following property: $\xi(u \oplus v) = \xi(u)\xi(v), \forall u, v \in \{0,1\}$. Thus, without loss of generality, with $G_{\Omega i}$ being the i-th column of G_Ω and m a binary vector, we have $\forall i \in \{1 \dots n\}$

$$\big(\phi(m, G_\Omega)\big)_i \coloneqq G_{\Omega i} \oplus m = \xi^{-1}\bigg(\Pi_{j=1}^k \xi\big((G_\Omega)_{ij} \cdot m_j\big)\bigg). \tag{4}$$

The new form defines a multilinear polynomial (potentially inducing saddle-point optimization) over the classical linear dot-product defined over $\mathbb{R}$ and the gradient can now be computed in a differentiable manner.

4.3. Differentiable Masking

The masking allows the integration of information about the code into the self-attention tensor. The masking derived from the Tanner graph connectivity can be soft (Choukroun & Wolf, 2024b) or hard (Choukroun & Wolf, 2022) and can be placed at different locations of the self-attention computation. However, existing masking methods induced from the code are extracted once in a non-differentiable fashion, i.e., no information can be backpropagated during the optimization from the mask to the code (i.e., the parity matrix).

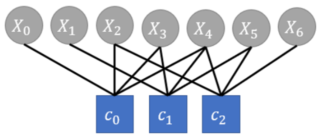
In order to allow the integration of the code through the self-attention while permitting its differentiable optimization, we learn a parameterized mapping $\psi_\gamma : \mathbb{N} \to \mathbb{R}$ of the elements constituting the mask, which is derived by the parity-check matrix, such that

$$A_H(Q, K, V) = \text{Softmax}\bigg(\frac{QK^T + \psi_\gamma\big(g(H_\Omega)\big)}{\sqrt{d}}\bigg)V, \tag{5}$$

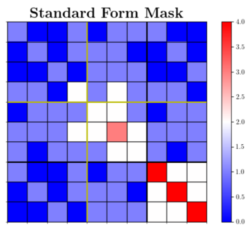
where the mask $g(H_\Omega) \in \mathbb{N}^{(2n-k)\times(2n-k)}$ is defined by

$$g(H_\Omega) = \begin{pmatrix} H_\Omega^T H_\Omega & H_\Omega \\ H_\Omega^T & H_\Omega H_\Omega^T \end{pmatrix} \tag{6}$$

Since H_Ω represents the bipartite Tanner graph, the mask diagonal block elements can be seen as the two-step transitioning connectivity, i.e., the number of paths of length two between every two nodes. The $n \times n$ top-left block matrix represents the two-step transition matrix between every two variable nodes, while the $(n-k) \times (n-k)$ bottom-right block matrix represents the two-step transition matrix between every two parity-check nodes. The diagonal elements of these matrices denote the degree of each node. Since the block off-diagonal is defined by H_Ω solely, it straightforwardly defines the relationship between the parity nodes and the variable nodes of the corresponding graph.



(a)



(b)

Figure 2: For the Hamming(7,4) code: (a) the Tanner graph, (b) the proposed differentiable masking for the standardized version of the code.

This way, the gradient $\nabla_\Omega \mathcal{L}$ can be backpropagated through the self-attention layers along the network to provide a decoder-aware code. An illustration of the proposed masking method is given in Figure 2.

4.4. Architecture

An illustration of the entire model is given in Figure 3. As with (Choukroun & Wolf, 2024b), the initial encoding is performed with a d dimensional one-hot encoding for the syndrome part and a single d-dimensional vector for the magnitude part, for a total of three d-dimensional parameters. Formally, the initial positional embedding $\Phi \in \mathbb{R}^{(2n-k)\times d}$ is given by

$$\Phi = [|y_\Omega|^T W_m, W_{s(y_\Omega)}] \tag{7}$$

with $W_m \in \mathbb{R}^d$ being the magnitude embedding vector and $W_0 \in \mathbb{R}^d$ and $W_1 \in \mathbb{R}^d$ the two one-hot encodings of each of the $n-k$ binary syndrome values.

The decoder is defined as a concatenation of N decoding layers composed of self-attention and feed-forward layers interleaved with normalization layers with d. The distance embedding $\psi_\gamma : \mathbb{N} \to \mathbb{R}$ is a fully connected neural network with a 50-dimensional hidden layer and ReLU nonlinearities mapping each number of paths to a scalar. This mapping becomes a fixed tensor at inference time. The contribution of each bit to itself (i.e. the diagonal elements) is omitted (masked) in the self-attention mechanism.

The output module is borrowed from (Choukroun & Wolf, 2024b) to allow a code-aware prediction conditioned by the parameterized H_Ω. The output module performs the following projections on the final embedding $\Phi := [\Phi_M, \Phi_S]$

$$\hat{\tilde{\varepsilon}} = \big(\Phi_M W_M + H_\Omega^T(\Phi_S W_S)\big) W_{d\to 1} \tag{8}$$

with $W_S, W_M \in \mathbb{R}^{d\times d}$ and $W_{d\to 1} \in \mathbb{R}^d$ as the embedding layers. Thus, our model remains code-, length-, and rate-invariant as well, by design.

The dimension of the feed-forward network of the transformer is four times that of the embedding d, following

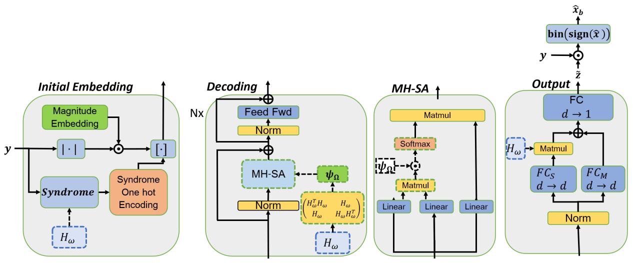


Figure 3: Illustration of the proposed architecture. The main contributions are represented with dashed lines.

(Vaswani et al., 2017), and is composed of GEGLU layers (Shazeer, 2020), with layer normalization set to the pre-layer setting, as in (Klein et al., 2017; Xiong et al., 2020). An eight-head (i.e., $h = 8$) self-attention module is used in all experiments. We note that while larger architectures would enable better performance, deepening the accuracy gap over other methods (e.g., GPT-3 (Brown et al., 2020) operates successfully on 2K inputs with a similar Transformer model, but with $N = 96, d = 12K$), ECC requires rather light and shallow models to be deployed on edge devices.

The computational complexity as well as the number of parameters of the method are the same as of the FECCT (Choukroun & Wolf, 2024b), since the code binary matrices (G and H) as well as the distance embedding functions ψ_γ are fixed after training. The number of parameters is defined by $\mathcal{O}(Nd^2)$. In comparison, the ECCT is not length invariant and has $\mathcal{O}(Nd^2 + nd)$ parameters.

4.5. Training

The training objective is the cross-entropy function as given in Eq. 2, while the estimated hard-decoded codeword is straightforwardly obtained as $\hat{x}_b = \text{bin}(\text{sign}(f_\theta(h(y_\Omega)) \cdot y_\Omega))$.

The Adam optimizer (Kingma & Ba, 2014) is used with 1024 samples per minibatch, for $1K$ epochs, with $1K$ minibatches per epoch. We initialized the learning rate to 10^{-4} coupled with a cosine decay scheduler down to 10^{-6} at the end of the training. No warmup was employed (Xiong et al., 2020). We note the optimization is stochastic and highly non-convex such that theoretical guarantees of the code performance or structure are difficult to establish.

We observed that using a large batch size ($\times 8$) greatly improves the performance of our method, as well as of other baselines (Choukroun & Wolf, 2022; 2024b). We, therefore, report in our tables new (and better) results for these baselines. We note that while using more epochs improves performance, the current modest setting already reaches state-of-the-art performance.

The initialization and optimization of Ω are of major impor-

Table 1: A comparison of the negative natural logarithm of Bit Error Rate (BER) for three normalized SNR values of our method with literature baselines. BP results are obtained after $L = 5$ BP iterations in first row and *at convergence* results in the second row are obtained after $L = 50$ BP iterations. SCL results are presented with a list length of $L = 1$ in the first row and $L = 32$ in the second row. Our performance is presented with fixed Ω (DC-ECCT) and with trained Ω (E2E DC-ECCT) for two shallow architectures: for $N = 2, d = 32$ in the first row and $N = 6, d = 128$ in the second row. The best results are in **bold**. The second best results are in *italic*. Higher is better.

Method	BP			SCL			ECCT			DC-ECCT			E2E DC-ECCT		
E_b/N_0	4	5	6	4	5	6	4	5	6	4	5	6	4	5	6
POLAR(32,11)	3.29	3.77	4.21	6.22	8.06	10.28	4.46	5.57	7.01	4.53	5.69	7.08	4.65	5.81	7.28
	3.84	4.71	5.70	*6.45*	*8.37*	*10.60*	6.37	8.12	10.19	6.41	8.09	10.57	**6.62**	**8.57**	**11.71**
POLAR(64,32)	3.53	4.02	4.45	7.24	9.74	12.91	4.56	5.93	7.75	4.40	5.76	7.67	4.72	6.22	8.13
	4.29	5.35	6.45	**8.16**	**10.73**	**13.98**	7.19	9.70	13.33	7.45	*10.49*	*13.74*	*7.59*	10.38	13.09
BCH(31,16)	4.59	5.87	7.57		NA		4.61	5.97	7.69	4.97	6.56	8.54	5.30	6.89	9.09
	5.12	6.87	9.27				6.37	8.32	10.63	*7.07*	**9.69**	*12.54*	**7.19**	*9.08*	**12.84**
BCH(63,45)	4.07	4.92	6.03		NA		4.59	6.07	8.13	4.54	6.07	8.14	4.98	6.69	8.97
	4.35	5.60	7.24				*6.25*	*8.78*	*12.45*	6.08	8.64	12.41	**6.37**	**9.09**	**13.12**
LDPC(49,24)	5.25	7.15	9.86		NA		4.21	5.32	6.56	4.08	5.29	6.55	4.95	6.46	8.33
	6.09	*8.75*	*11.91*				5.34	6.43	7.21	5.57	6.55	7.21	**6.28**	**8.77**	**12.33**
RS(60,52)	4.43	5.32	6.43		NA		4.37	5.11	6.03	5.04	6.68	8.82	5.12	6.80	9.02
	4.69	6.43	7.56				4.37	5.13	6.04	**5.61**	**7.59**	*9.82*	**5.61**	*7.56*	**9.90**

tance in our highly non-convex optimization setting. The initialization can be performed given a binary matrix Ω_0 obtained from random sampling or from a baseline standardized parity-check matrix. Thus, given an initial binary matrix Ω_0, the learnable matrix is initialized as $\Omega = c \cdot \xi(\Omega_0)$ with $c \in \mathbb{R}_+$, providing a uniform confidence to every element of Ω. The hyperparameters used for the optimization of Ω are the early stopping training of Ω (not of the decoder), and the learning rate of Ω following (Courbariaux et al., 2015). In all the experiments $c = 0.01$, $\tau = \infty$ but with Ω clamped such that $|\Omega| < 0.5$, as in (Courbariaux et al., 2015). However, other values can be more optimal for any given code, and using larger batches would further improve performance. As shown in Appendix A, it is important to train using the all ones message, i.e., $m = \mathbb{1}_k$.

Accelerating the proposed method (e.g. pruning, model quantization/binarization, distillation, low-rank approximation) (Wang et al., 2020; Lin et al., 2021) is beyond the scope of this paper and is left for future work. E.g., sparse self-attention can be induced via the regularization of the objective. Training and experiments are performed on a 12GB GeForce RTX 2080 Ti GPU. The training time ranges from 34 to 327 seconds per epoch depending on code length, rate, and model size. Testing time ranges from 2 to 3ms per sample, using one GPU without any model optimization.

5. Experiments

To evaluate our method, we train the proposed architecture with four classes of linear codes: Low-Density Parity Check (LDPC) codes (Gallager, 1962), Polar codes (Arikan, 2008), Reed Solomon codes (Reed & Solomon, 1960) and Bose–Chaudhuri–Hocquenghem (BCH) codes (Bose & Ray-Chaudhuri, 1960). All the parity check matrices are taken from (Helmling et al., 2019). Code available at `https://github.com/yoniLc/E2E_DC_ECCT`.

We compare our method with the BP algorithm (Pearl, 1988), the SCL algorithm (Tal & Vardy, 2015) for polar codes, and the SOTA ECCT (Choukroun & Wolf, 2022) neural decoder. FECCT is not tested, since while possessing important invariance properties, it reaches similar performance as the ECCT in average. We note that, similarly to FECCT, our method can be implemented as a foundation model, where a single decoder is used for decoding multiple (potentially trained) codes. However, we are not looking to optimize multiple codes at once, due to the associated training complexity. Other neural decoders are not presented, since their performance remains far from the ECCTs. Note that LDPC codes are specifically designed for BP-based decoding (Richardson et al., 2001). SCL (Tal & Vardy, 2015) is specifically designed for (short) polar codes on which it is close to ML performance.

As opposed to other methods (Nachmani et al., 2016; Nachmani & Wolf, 2019; 2021), our method as well as the other Transformer based neural decoders do not assume that the channel is known and do not then compute the LLRs as model input for improved performance. We observe that using LLR very slightly improves these methods, while removing it from classical BP induces catastrophic degradation of the performance.

The results are reported as negative natural logarithm bit error rates (BER), i.e., $-\ln(\text{BER})$, for three different normalized SNR values (Eb/N_0), following the conventional testing benchmark, e.g., (Nachmani & Wolf, 2019; Choukroun & Wolf, 2022). BP-based results are obtained after $L = 5$

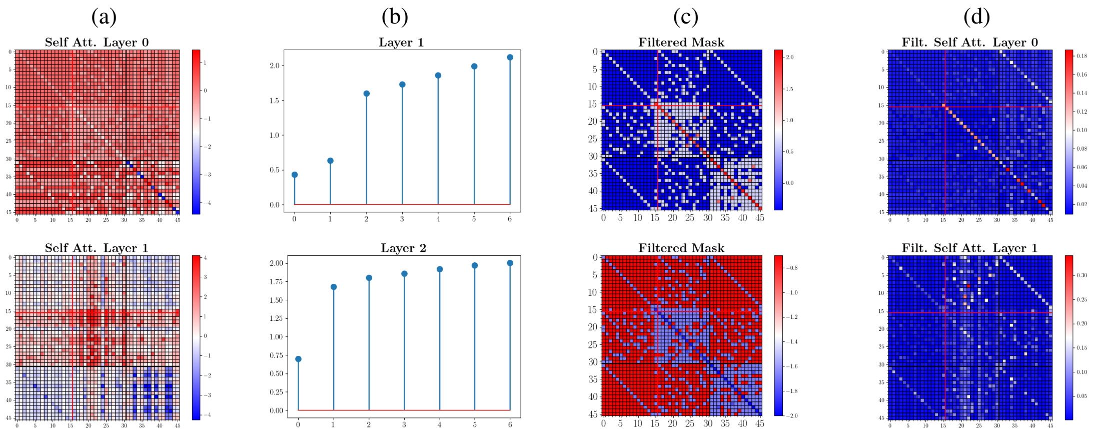


Figure 4: For a $N = 2$ layers DC-ECCT (first and second row) and a (31,16) code: (a) self-attention layer, (b) connectivity mapping ψ_γ, (c) the corresponding filtered mask $\psi_\gamma\big(g(H_\Omega)\big)$ (d) the obtained soft masked self-attention. The self-attention maps have been averaged over the heads dimension.

BP iterations in the first row (i.e., 10-layer neural network) and *at convergence* results in the second row are obtained after $L = 50$ BP iterations (i.e., 100-layer neural network). During testing, at least 10^5 random codewords are decoded, to obtain at least 50 frames with errors at each SNR value. We trained and tested all reported ECCT results ourselves to ensure that the models were trained on the same parity-check matrices. The SCL experiments are conducted by us, using the code framework of (Cassagne et al., 2019).

The hyperparameter search was performed using a validation set as follows. For all neural decoders (i.e., including ECCT), we selected the best results obtained from *a single* random initialization and with the baseline initialization of Ω. For the trained Ω setting, we tested the early stopping of Ω optimization after 800 epochs. We also experimented with training the code with a smaller learning rate defined following (Courbariaux et al., 2015; Glorot & Bengio, 2010). Results showing the impact of the initialization on the proposed decoder and encoder-decoder framework are given in Appendix C. The parity-check matrices of all neural decoders are in standard form.

In Table 1 we present the performance of our model Deep Coding Error Correction Code Transformer (DC-ECCT) on several codes. As can be seen, even for fixed (not trained) Ω our neural decoder outperforms the state-of-the-art neural decoder. Moreover, the end-to-end optimization of the code (E2E DC-ECCT) improves the performance by very large margins. We can observe that polar codes are already well-suited for the inductive bias of transformers. Also, on larger polar codes SCL gets close to ML and the code optimization seems to converge to a worse local minima. We provide BER plot visualizations in Appendix F.

6. Ablation Study and Analysis

We study the impact of the proposed method on other decoders and analyze the different modules of the method. We provide in Appendix E an ablation study of the different components of the proposed method, demonstrating the superiority of the components of the proposed solution.

6.1. Performance with Belief Propagation

In Table 2, we present the performance of the Belief Propagation decoding algorithm on different codes and rates. We compare the performance of the baselines codes, these same codes in standard form, random codes, random standardized codes (i.e., random binary Ω), and codes obtained from the Ω optimized via the E2E DC-ECCT. We can observe that the learned codes outperform other codes under the BP decoder by very large margins, even if the code is presented in standard form. This supports the claim that our method is able to provide good codes in a broader sense.

Table 2: A comparison of the negative natural logarithm of BER for three normalized SNR values of the BP method on different codes that also appeared in Table 1. Results are obtained after $L = 5$ BP iterations in the first row and *at convergence* results in the second row are obtained after $L = 50$ BP iterations. Best for each L in bold.

Code	Baseline			Standard form			Random H			Random Ω			E2E Ω		
E_b/N_0	4	5	6	4	5	6	4	5	6	4	5	6	4	5	6
(31,16)	4.59	5.87	7.57	3.97	4.67	5.57	3.23	3.72	4.36	4.22	5.01	5.86	**6.13**	**7.95**	**9.90**
	5.12	6.87	9.27	4.64	5.89	7.37	3.42	4.18	5.33	4.89	6.14	7.38	**6.42**	**8.31**	**10.24**
(63,45)	4.07	4.92	6.03	4.37	5.33	6.42	3.72	4.26	4.93	3.96	4.61	5.41	**5.55**	**7.33**	**9.23**
	4.35	5.60	7.24	4.78	6.30	8.22	3.93	4.79	5.93	4.27	5.29	6.69	**6.15**	**8.60**	**11.32**
(60,52)	4.43	5.32	6.43	4.60	5.65	6.94	4.41	5.32	6.45	4.60	5.63	6.93	**4.73**	**5.79**	**7.01**
	4.69	5.95	7.56	4.83	6.21	8.00	4.65	5.90	7.53	4.83	6.19	7.97	**4.99**	**6.45**	**8.35**
(64,32)	3.53	4.02	4.45	3.82	4.37	5.03	2.92	3.37	3.73	3.34	3.90	4.64	**6.93**	**9.49**	**12.51**
	4.29	5.35	6.45	4.81	5.74	6.71	2.96	3.54	4.17	3.59	4.53	5.87	**7.69**	**10.04**	**12.94**

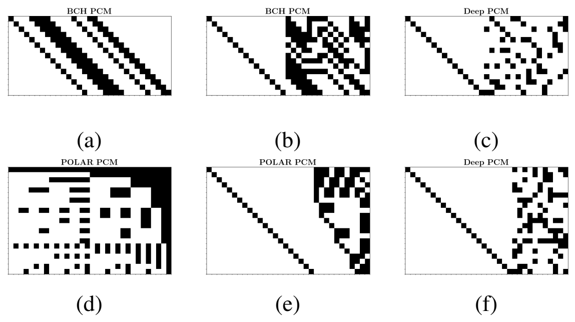


Figure 5: The original parity-check matrix (PCM) of (a) BCH(31,16), (d) POLAR(32,11) and their standard form in (b) and (e), respectively. The third column corresponds to the learned parity-check matrices of the corresponding code length and rate. The PCM sparsity is of (a) 25%, (b) 30% (c) 16%, (d) 31%, (e) 17%, and (f) 15%.

In Table 3 we also present the ML decoding performance on the shortest codes, further demonstrating that the method can learn competitive codes, independently of the decoder.

Table 3: A comparison of the maximum likelihood method on different codes. The code tested are the baseline codes, random code and, the proposed learned codes.

Code	Baseline			Random			E2E Ω		
E_b/N_0	4	5	6	4	5	6	4	5	6
(31,16)	7.40	9.81	13.11	7.25	9.14	10.80	7.39	9.54	12.19
(32,11)	6.50	8.28	10.71	7.08	8.48	11.72	7.34	9.48	11.79

6.2. Parity-check Matrix Visualization

In Figure 5, we depict several typical parity-check matrices for different codes, where we can see our optimization method generally provides sparse codes. More visualizations and explanations are given in Appendix B.

6.3. Visualization of Learned Mapping and Self-attention Maps

In Figure 4 we show the self-attention maps at the different layers of the model, at the different stages of the proposed soft masking. We first depict the classical self-attention layer in (a), then show the learned connectivity mapping ψ_γ in (b), the induced filtered mask in (c), and the resulting masked/filtered self-attention (d). We can observe from the connectivity mapping (b) that this shallow two-layer network learns to initially analyze highly connected nodes in the first layer, to finally focus on other less related nodes. This translates directly to complementary saliency regions of the filtered masks (c). Visualization for a $N = 6$ model is given in Appendix D.

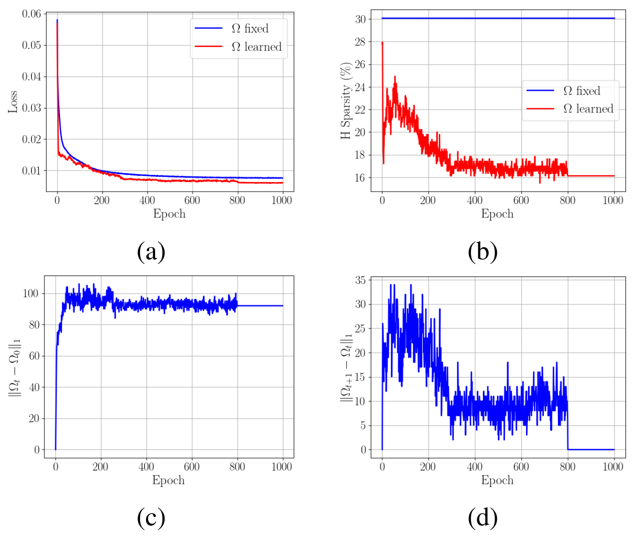


Figure 6: (a) Training loss of the proposed DC-ECCT with Ω fixed and trained for BCH(31,16) codes. (b) Evolution of Ω_t compared to Ω_0. (c) Evolution of Ω_t compared to Ω_{t-1}. (b) Parity-check matrix sparsity. In these experiments, the initial Ω_0 is random, the optimization over Ω is stopped at iteration 800, and the architecture is $N = 2, d = 32$.

6.4. Training Dynamics

We present in Figure 6 the typical training dynamics of the proposed framework. In panel (a) we show the training loss for a fixed Ω. As can be seen, the encoder-decoder models enable faster and better training. We present in (b,c) the high variation of the learned Ω at the beginning of the optimization, attenuated towards the end of the training. Finally, we can observe in (d) the level of sparsity of the code during training. The framework tends to produce sparse codes, with the main modification of the codes appearing during the first stage of training.

7. Conclusion

We present a novel end-two-end training method of the binary linear block error correction system. The proposed framework enables the effective and differentiable joint optimization of the code and of the neural decoder. The neural decoder based on the Transformer architecture allows the differentiable training of the code via the Tanner graph connectivity derivation from the parity-check matrix. The proposed neural decoder outperforms the state-of-the-art, while the unified encoding-decoding training allows further improvement of performance.

A common criticism for ML-based ECC is that the neural decoder cannot be deployed directly without the application of massive deep-learning acceleration methods. Here, we show for the first time that a code trained jointly with its decoder is also better for popular classical decoders. Looking forward, the efficient optimization of the codes may open the door to the creation of new families of codes and the establishment of new industry standards.

8. Impact Statements

This paper presents work whose goal is to advance the field of Machine Learning and Information Theory. No real societal consequences of our work can be easily established or highlighted here.

9. Acknowledgements

This project has received funding from the Tel Aviv University Center for AI and Data Science (TAD) and the Blavatnik Computer Science Research Fund. The contribution of the first author is part of a PhD thesis research conducted at Tel Aviv University.

References

Aoudia, F. A. and Hoydis, J. End-to-end learning of communications systems without a channel model. In *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, pp. 298–303. IEEE, 2018.

Arikan, E. Channel polarization: A method for constructing capacity-achieving codes. In *2008 IEEE International Symposium on Information Theory*, pp. 1173–1177. IEEE, 2008.

Bengio, Y., Léonard, N., and Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Bennatan, A., Choukroun, Y., and Kisilev, P. Deep learning for decoding of linear codes-a syndrome-based approach. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 1595–1599. IEEE, 2018.

Bose, R. C. and Ray-Chaudhuri, D. K. On a class of error correcting binary group codes. *Information and control*, 3(1):68–79, 1960.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

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

Cammerer, S., Hoydis, J., Aoudia, F. A., and Keller, A. Graph neural networks for channel decoding. In *2022 IEEE Globecom Workshops (GC Wkshps)*, pp. 486–491. IEEE, 2022.

Cassagne, A., Hartmann, O., Léonardon, M., He, K., Leroux, C., Tajan, R., Aumage, O., Barthou, D., Tonnellier, T., Pignoly, V., Le Gal, B., and Jégo, C. Aff3ct: A fast forward error correction toolbox! *Elsevier SoftwareX*, 10:100345, October 2019. ISSN 2352-7110. doi: https://doi.org/10.1016/j.softx.2019.100345. URL http://www.sciencedirect.com/science/article/pii/S2352711019300457.

Choukroun, Y. and Wolf, L. Error correction code transformer. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Choukroun, Y. and Wolf, L. Denoising diffusion error correction codes. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

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

Choukroun, Y. and Wolf, L. A foundation model for error correction codes. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024b. URL https://openreview.net/forum?id=7KDuQPrAF3.

Courbariaux, M., Bengio, Y., and David, J.-P. Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in neural information processing systems*, 28, 2015.

ESTI. 5g nr multiplexing and channel coding. etsi 3gpp ts 38.212. https://www.etsi.org/deliver/etsi_ts/138200_138299/138212/16.02.00_60/ts_138212v160200p.pdf, 2021.

Gallager, R. Low-density parity-check codes. *IRE Transactions on information theory*, 8(1):21–28, 1962.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Gruber, T., Cammerer, S., Hoydis, J., and ten Brink, S. On deep learning-based channel decoding. In *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6. IEEE, 2017.

Helmling, M., Scholl, S., Gensheimer, F., Dietz, T., Kraft, K., Ruzika, S., and Wehn, N. Database of Channel

Codes and ML Simulation Results. www.uni-kl.de/channel-codes, 2019.

Hoydis, J., Cammerer, S., Ait Aoudia, F., Vem, A., Binder, N., Marcus, G., and Keller, A. Sionna: An open-source library for next-generation physical layer research. *arXiv preprint*, Mar. 2022.

Jiang, Y., Kannan, S., Kim, H., Oh, S., Asnani, H., and Viswanath, P. Deepturbo: Deep turbo decoder. In *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–5. IEEE, 2019a.

Jiang, Y., Kim, H., Asnani, H., Kannan, S., Oh, S., and Viswanath, P. Turbo autoencoder: Deep learning based channel codes for point-to-point communication channels. *Advances in neural information processing systems*, 32, 2019b.

Kim, H., Jiang, Y., Kannan, S., Oh, S., and Viswanath, P. Deepcode: Feedback codes via deep learning. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 9436–9446, 2018a.

Kim, H., Jiang, Y., Rana, R., Kannan, S., Oh, S., and Viswanath, P. Communication algorithms via deep learning. In *Sixth International Conference on Learning Representations (ICLR)*, 2018b.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. M. Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL*, 2017. doi: 10.18653/v1/P17-4012. URL https://doi.org/10.18653/v1/P17-4012.

Kwak, H.-Y., Yun, D.-Y., Kim, Y., Kim, S.-H., and No, J.-S. Boosting learning for ldpc codes to improve the error-floor performance. *arXiv preprint arXiv:2310.07194*, 2023.

Lin, T., Wang, Y., Liu, X., and Qiu, X. A survey of transformers. *arXiv preprint arXiv:2106.04554*, 2021.

Nachmani, E. and Wolf, L. Hyper-graph-network decoders for block codes. In *Advances in Neural Information Processing Systems*, pp. 2326–2336, 2019.

Nachmani, E. and Wolf, L. Autoregressive belief propagation for decoding block codes. *arXiv preprint arXiv:2103.11780*, 2021.

Nachmani, E., Be'ery, Y., and Burshtein, D. Learning to decode linear codes using deep learning. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 341–346. IEEE, 2016.

O'Shea, T. J. and Hoydis, J. An introduction to machine learning communications systems. *arXiv preprint arXiv:1702.00832*, 2017.

Park, S.-J., Kwak, H.-Y., Kim, S.-H., Kim, S., Kim, Y., and No, J.-S. How to mask in error correction code transformer: Systematic and double masking. *arXiv preprint arXiv:2308.08128*, 2023.

Pearl, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan kaufmann, 1988.

Raviv, N., Caciularu, A., Raviv, T., Goldberger, J., and Be'ery, Y. perm2vec: Graph permutation selection for decoding of error correction codes using self-attention. *arXiv preprint arXiv:2002.02315*, 2020.

Raviv, T., Goldmann, A., Vayner, O., Be'ery, Y., and Shlezinger, N. Crc-aided learned ensembles of belief-propagation polar decoders. *arXiv preprint arXiv:2301.06060*, 2023.

Reed, I. S. and Solomon, G. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.

Richardson, T. J., Shokrollahi, M. A., and Urbanke, R. L. Design of capacity-approaching irregular low-density parity-check codes. *IEEE transactions on information theory*, 47(2):619–637, 2001.

Shannon, C. E. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Shazeer, N. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Tal, I. and Vardy, A. List decoding of polar codes. *IEEE Transactions on Information Theory*, 61(5):2213–2226, 2015.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pp. 10524–10533. PMLR, 2020.

Ye, H., Li, G. Y., Juang, B.-H. F., and Sivanesan, K. Channel agnostic end-to-end learning based communication systems with conditional gan. In *2018 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–5. IEEE, 2018.

A. Repetition Code Backpropagation Analysis

We assume the standardized (3,1) repetition code such that $G = \begin{pmatrix} 1 & c_1 & c_2 \end{pmatrix}$ and $H = \begin{pmatrix} c_1 & 1 & 0 \\ c_2 & 0 & 1 \end{pmatrix}$, where $c_1, c_2 \in \{0, 1\}$. The repetition code is optimal for $c_1 = c_2 = 1$.

Given a message $m \in \{0, 1\}$ we have $x = mG = \begin{pmatrix} m & c_1 m & c_2 m \end{pmatrix}$. Assuming no modulation, under a binary erasure channel we have $y = \begin{pmatrix} y_1 & y_2 & y_3 \end{pmatrix} = x \oplus n$ with $n \in \{0, 1\}^3$ the noise vector. The code syndrome is then of the form

$$s = \begin{pmatrix} s_0 \\ s_1 \end{pmatrix} = Hy = \begin{pmatrix} c_1 y_1 \oplus y_2 \\ c_2 y_1 \oplus y_3 \end{pmatrix} \tag{9}$$

Using polarized notation, we have

$$\begin{cases} s_1 & = 0.5 - 0.5(1 - 2c_1 y_1)(1 - 2y_2) \\ s_2 & = 0.5 - 0.5(1 - 2c_2 y_1)(1 - 2y_3) \end{cases} \tag{10}$$

Thus, we have $\frac{\partial s_1}{\partial c_2} = \frac{\partial s_2}{\partial c_1} = 0$ and, without loss of generality we have

$$\frac{\partial s_1}{\partial c_1} = y_1(1 - 2y_2) + (1 - 2c_1 y_1)\frac{\partial y_2}{\partial c_1} \tag{11}$$

We can observe that assuming y_i as independent from the codeword (i.e., not derived from G) for *every* (potentially wrong) value of c_i, the gradient is zero and no update is possible half of the time (i.e., $m = n_1 = 0$ or $m = n_1 = 1$). Thus the channel output should not be assumed as constant during backpropagation (i.e., $y_i \coloneqq y_i(c_1, c_2)$).

Assuming the channel output is dependent on the generator and considering it for the computation of the codeword we have now $y_i = x_i \oplus n_i$ and then

$$\frac{\partial s_1}{\partial c_1} = y_1(1 - 2y_2) + m(1 - 2c_1 y_1)(1 - 2n_2) \tag{12}$$

We can observe sampling zero messages induces the same gradient as the independent setting since it cancels the parity check coupled with it. Thus, we can deduct the all ones *binary* message i.e., $m = 1$ (or at least $m \neq 0$, a contrary to what is commonly done in neural decoders training) is important for efficient backpropagation since it will not cancel the information propagation in the backward pass.

B. Parity-check Matrix Visualization

In Figure 7, we depict several typical parity-check matrices for different codes. We note that the sparsity of the code is a property that emerges from training and is not enforced by the framework: sparse codes seem more appropriate for Transformer-based neural decoders' inductive bias.

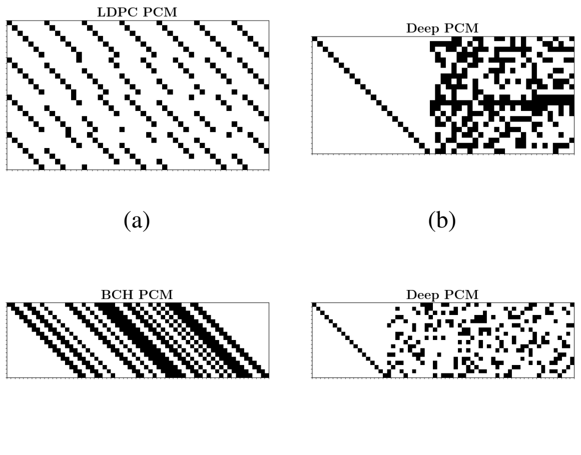


Figure 7: The original parity-check matrix (PCM) of additional codes. (a) LDPC(49,24), and (c) BCH(63,45). The second column (b,d) corresponds to the learned parity-check matrices of the corresponding code length and rate. The PCM sparsity is of (a) 15%, (b) 26%, (c) 38%, and (d) 20%

C. Impact of the Initialization

We present in Table 4 the performance of the DC-ECCT and the E2E DC-ECCT under baseline and random initialization. We can observe that the initialization has a very major impact on the DC-ECCT while the end-to-end training allows to mitigate the code impact but still presents different performance under different initialization, which is understandable in high dimensional non-convex optimization scenarios. Also, it seems Polar codes, contrary to other codes, already provide a strong initialization for the proposed framework.

From our experience, refining a learned parity check matrix most of the time did not improve or even gave worse performance. The reason should lie within the high dimensional non-convex optimization where the local minima of may be a sharp minimum [1] such that small perturbation (retraining) induces a worse local optimum. Other optimization methods may allow such refinement though.

D. Visualization of Learned Mapping and Self-attention Maps

Figure 8 depicts BER plots comparing the performance of the baselines and the proposed method for several codes.

E. Ablation Study

In Figure 5 we present an ablation study of the different elements of the proposed framework.

Table 4: A comparison of the negative natural logarithm of Bit Error Rate (BER) for three normalized SNR values of our method for $N = 2, d = 32$ with different initializations. We test the initialization of Ω with baseline codes and with random parity-check matrix. Higher is better.

Model	DC-ECCT						E2E DC-ECCT					
Code	Baseline Ω			Random Ω			Baseline Ω			Random Ω		
E_b/N_0	4	5	6	4	5	6	4	5	6	4	5	6
POLAR(32,11)	4.53	5.69	7.08	3.95	4.93	6.18	4.55	5.68	7.08	4.65	5.81	7.28
POLAR(64,32)	4.40	5.76	7.57	3.47	4.42	5.77	4.72	6.22	8.13	4.70	6.13	8.05
BCH(31,16)	4.97	6.56	8.54	4.62	5.90	7.59	5.22	6.75	8.56	5.30	6.89	9.09
BCH(63,45)	4.54	6.06	8.12	4.41	5.85	7.83	4.72	6.37	8.60	4.98	6.69	8.97
LDPC(49,24)	4.08	5.19	6.45	3.75	4.81	6.24	4.95	6.45	8.33	4.95	6.46	8.33
RS(60,52)	4.38	5.13	6.03	5.04	6.68	8.82	4.38	5.13	6.03	5.12	6.80	9.02

In order to emphasize the importance of a mask induced by the code we provide two ablation studies. Regarding the *masking* procedure itself, we first provide the performance of a fully trainable mask, such that the self-attention is multiplied similarly as in Eq 6 while the mask is a parameter matrix trained independently of the code. Regarding the training via the mask we show the performance of the proposed framework where the gradients originated from the mask are stopped.

In order to emphasize the importance of the sampling of the original message m we provide results using random sampling of m (i.e., not $m = 1$). We note here that using $m = 0$ (i.e., G is not taken into account for the optimization), as done in almost all the existing neural decoding methods, just completely fails the training.

Finally, even though there exists no known alternative, we provide an interesting insight into the gradient computation of the modulo operator. We apply the STE (Bengio et al., 2013) on the results of a regular matrix-vector multiplication. Given the modulo function g, we then have $g(x) = x\text{mod}2$ and $\frac{\partial g(x)}{\partial x} = \mathbb{1}_{|x|\leq 1}$. Since this approach is less theoretically founded than the proposed polarization (especially in our scenario where the modulo operation is performed over $\mathbb{N}$), it is interesting to see its overall training seems to bring enhanced performance.

We note here that the quality of the learned code can certainly be improved if better trained on larger noise ranges (i.e., range) and batch size, depending on available computing resources.

F. BER Visualization

Figure 9 depicts classical BER plots for three of the tested codes.

Table 5: Ablation analysis on two codes for the $N = 2, d - 32$ architecture. For fairness, the same setting is used in all experiment: clamping Ω s.t. $|\Omega| \leq 1$ and we stop the training of Ω after 800 iterations among 1000. Ω_0 is randomely sampled. We show the performance of the proposed method (Our), the performance using a fully trainable mask independently of the code (Mask V2), and the performance of using the STE for the modulo calculations instead of the polarization approximation. We show the performance when we stop the gradient from backpropagating via the mask (Mask Stop Gradient) as well as the performance of using random m instead of all ones.

Code	(31,16)			(32,11)		
E_b/N_0	4	5	6	4	5	6
Our	5.16	6.51	8.19	4.40	5.54	7.04
Mask V2	4.83	6.18	7.85	4.04	5.05	6.28
STE	5.11	6.53	8.20	4.63	5.89	7.39
Mask S.G.	4.54	5.81	7.50	4.26	5.39	6.79
Random m	5.04	6.44	8.13	4.38	5.50	6.84

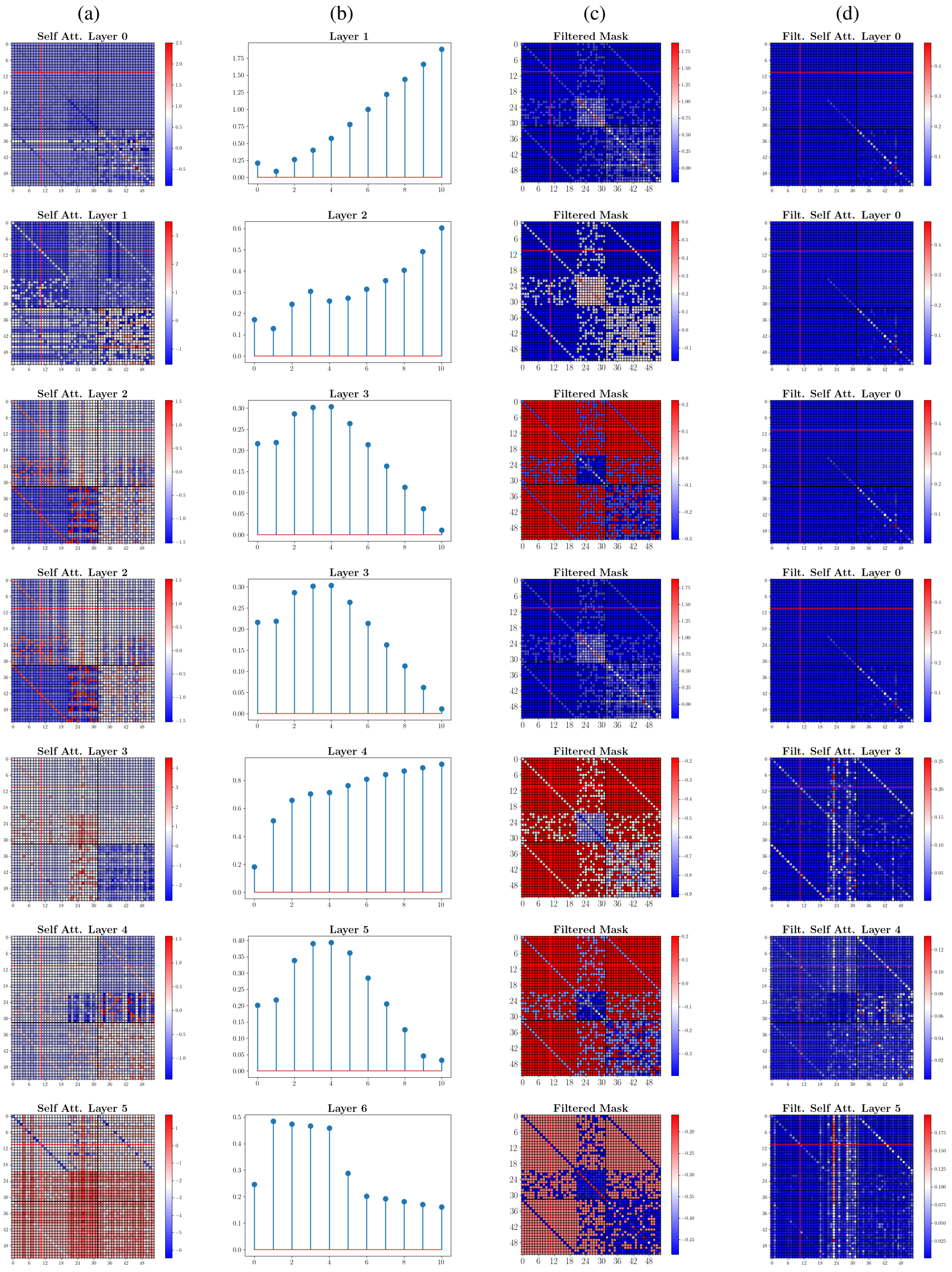


Figure 8: For a $N = 6$ layers DC-ECCT (first and second row) and a (32,11) code: (a) self-attention layer, (b) connectivity mapping ψ_γ, (c) the corresponding filtered mask $\psi_\gamma\big(g(H_\Omega)\big)$ (d) the obtained soft masked self-attention. The self-attention maps have been averaged over the heads dimension.

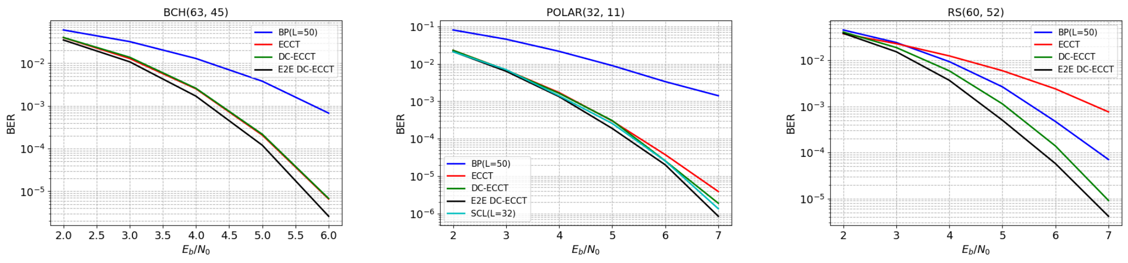


Figure 9: BER plots comparing the performance of the baselines and the proposed method for various E_b/N_0 values.