

Article

E2E DC-CrossMPT: Cross-Attention Message-Passing Transformer for Joint Design and Decoding of Linear Block Codes

Yeji Cho ¹ and Junghyun Kim ^{2,3,*} 

¹ Department of Convergence Engineering for Artificial Intelligence, Sejong University, Seoul 05006, Republic of Korea; yezy0390@sju.ac.kr

² Department of Artificial Intelligence and Data Science, Sejong University, Seoul 05006, Republic of Korea

³ Deep Learning Architecture Research Center, Sejong University, Seoul 05006, Republic of Korea

* Correspondence: j.kim@sejong.ac.kr

Abstract

In this paper, we propose a novel deep learning-based framework for the joint design and decoding of linear block codes, the end-to-end deep coding cross-attention message-passing Transformer (E2E DC-CrossMPT). To improve linear block code design and decoding, we redesign the conventional error correction code (ECC) decoder, CrossMPT, to fit within an end-to-end framework. The redesigned decoder separately utilizes magnitude and syndrome vectors obtained from the received signals as inputs. It further employs one-hot encoding based syndrome embedding and incorporates a parity-check matrix into the output layers. Experimental results demonstrate that, across various code lengths and code rates, E2E DC-CrossMPT consistently outperforms both traditional decoding algorithms and a conventional end-to-end deep coding model in terms of decoding performance. Moreover, the codes designed by E2E DC-CrossMPT achieve superior error-correction capability compared with both traditional linear block codes and those designed by the conventional end-to-end deep coding model, while requiring lower computational complexity.

Keywords: error correction code (ECC); transformer; cross-attention; code construction; linear block code

1. Introduction

Error correction codes (ECCs) are used to add redundancy to information bits to correct errors occurring during transmission [1]. Linear block codes are a class of error-correction codes, including Bose–Chaudhuri–Hocquenghem (BCH), low-density parity-check (LDPC), and Polar codes, which are typically decoded using traditional algorithms such as belief propagation (BP), min-sum (MS), and successive cancellation (SC). However, traditional decoding algorithms achieve optimal performance only under specific code configurations. For example, the BP decoder performs near-optimally for long LDPC codes but suffers from performance degradation at short code lengths. Similarly, the SC decoder achieves near-optimal performance for Polar codes but fails to provide comparable performance for other types of codes. These limitations motivate the development of new decoding approaches.

Recently, ECC decoders that apply deep learning-based Transformer structures have attracted significant attention and have shown remarkable performance improvements across various codes. First, the Transformer-based ECC decoder, error correction code Transformer (ECCT) [2] uses the concatenation of the magnitude and syndrome vectors



Academic Editor: Hector E. Nistazakis

Received: 15 May 2026

Revised: 18 June 2026

Accepted: 22 June 2026

Published: 25 June 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

obtained from the received signals as input to the model. This model achieves better performance than traditional decoding algorithms and conventional neural decoders for various linear block codes with different code rates. Subsequently, the cross-attention message-passing Transformer (CrossMPT) [3] uses the magnitude and syndrome vectors as separate inputs and introduces an iterative message update mechanism based on cross-attention. As a result, CrossMPT reduces complexity and improves decoding performance compared with ECCT.

Although deep learning-based decoders have been actively studied for decoding linear block codes, studies that attempt to utilize deep learning for code design remain limited due to the non-differentiability of binary code matrices (generator and parity-check matrices) and encoding operations. Recently, the deep learning-based linear block code design model, end-to-end deep coding ECCT (E2E DC-ECCT) [4], was proposed. This model applies a differentiable equivalence mapping to the encoding operation and employs a straight-through estimator (STE) directly on the binary code matrices to facilitate differentiation. As a result, E2E DC-ECCT achieves superior performance compared with traditional decoding algorithms and neural decoders, and the codes designed by E2E DC-ECCT demonstrate improved error-correction capability compared with conventional linear block codes.

Despite these advances, the Transformer-based decoders and the end-to-end code design framework still have limitations. Although CrossMPT demonstrates strong decoding performance with reduced computational complexity, it is limited to the decoding stage and cannot be directly extended to joint code design due to the non-differentiability of code matrices. In contrast, E2E DC-ECCT enables end-to-end learning of linear block codes, but its decoder structure leads to relatively high computational complexity compared with CrossMPT.

In this paper, we propose a novel end-to-end model, E2E DC-CrossMPT, to overcome the limitations of the two models and to achieve improved linear block code design and decoding. The proposed model uses a redesigned CrossMPT as the decoder, adapted to the end-to-end framework. The redesigned decoder incorporates the following modifications: First, a one-hot encoding-based embedding method is applied to the syndrome vectors in the initial embedding, which reduces the number of model parameters. Second, the output layers, composed of two linear layers, are modified to integrate the parity-check matrix, enabling the model to more effectively learn code-aware representations. Third, by processing the magnitude and syndrome vectors as separate inputs and employing a cross-attention mechanism with a reduced-size mask matrix, the proposed model significantly lowers the computational complexity of the attention operation, thereby improving overall computational efficiency compared with E2E DC-ECCT.

Experimental results demonstrate that, for BCH, Polar, LDPC, and Reed–Solomon (RS) codes with various code lengths and code rates, E2E DC-CrossMPT achieves superior performance compared with both traditional decoding algorithms and the conventional model, E2E DC-ECCT. The codes designed by the proposed model exhibit superior error correction capability, and the model achieves higher efficiency across all codes compared with the conventional model. In addition, E2E DC-CrossMPT improved model efficiency in all codes compared to E2E DC-ECCT, and in particular in the (63,45) code, FLOPs were reduced by about 28.7%, training time by about 30.5%, and inference time by about 82.7%.

2. Related Works

2.1. Deep Learning-Based Error Correction Code Decoder

Existing studies on neural network-based ECC decoders can be broadly divided into two categories: model-based decoders and model-free decoders. Model-based decoders are architectures that unfold conventional BP or MS decoding algorithms into neural network

structures [5–7]. These methods improve decoding performance by unfolding the iterative message-passing process performed on a Tanner graph into a deep neural network and assigning learnable weights to each edge or message. The neural BP decoder for BCH codes [8], which unfolds the BP decoding process into a neural network structure, achieved improved decoding performance compared with the conventional BP decoder. The neural MS decoder [9] was proposed for decoding protograph LDPC codes and introduced a parameter sharing mechanism to improve scalability for longer codes. Subsequent studies have shown that neural BP and neural MS decoders can achieve better performance than conventional decoders [10–15]. However, these model-based neural network decoders rely on the structure of conventional decoding algorithms, which limits architectural flexibility and further performance improvements.

Model-free decoders perform decoding using general neural network architectures without directly relying on conventional decoding algorithms. Early studies proposed decoders based on fully connected networks, convolutional neural networks (CNNs), and recurrent neural networks (RNNs) [6,16,17]. However, for linear block codes, the extremely large number of possible codewords makes it impractical to use all codewords for training. Therefore, when training is performed using only a subset of codewords, the model tends to overfit to the codewords used during training. To address this issue, a syndrome-based preprocessing method was proposed to alleviate overfitting in model-free decoders [18]. This method is designed to train the model using only the all-zero codeword and to estimate the noise by utilizing syndrome information obtained from the received signal. This preprocessing technique was applied to RNN-based decoders and improved the generalization capability of model-free decoders. Subsequently, ECCT [2] combined syndrome-based preprocessing with a masked self-attention mechanism and achieved strong decoding performance across various linear block codes. Since the introduction of ECCT, Transformer-based ECC decoders have been actively studied. For example, Accelerating ECCT (AECCT) [19] applies ternary weight quantization to ECCT and implements a decoder with linear layers that do not require multiplication operations. Denoising Diffusion ECC (DDECC) [20] reinterprets the decoding process as a diffusion process, while Multi-Mask ECCT (MM ECCT) [21] improves decoding performance by using multiple parity-check matrices for the same code to capture diverse bit relationships. In addition, Foundation ECCT (FECCT) [22] is a Transformer-based foundation model designed to perform decoding regardless of code class, block length, and code rate. Finally, CrossMPT [3] introduces a cross-attention mechanism inspired by message-passing algorithms and improves both decoding performance and computational efficiency through iterative information exchange between the magnitude and syndrome vectors.

2.2. Deep Learning-Based Code Design

Although neural network-based decoders have shown improved decoding performance in various communication environments, studies on new code designs using neural networks have achieved only limited success. First, RNNs were used in [23] to construct low-latency codes and feedback codes, respectively, achieving performance improvement. A deep learning-based Polar code design method suitable for BP decoding was also proposed by treating the frozen-bit positions of Polar codes as learnable weights [24]. In addition, a previous study learned a continuous-valued (7,4) code satisfying a power constraint and demonstrated performance comparable to that of the Hamming code [25]. Subsequently, joint design methods for feedback channels and Turbo codes were also proposed [26]. End-to-end learning methods under non-differentiable modulation or channel environments have also been studied.

However, most of these methods use complex neural network-based encoding and decoding structures in the continuous domain, which differ from the encoding and decoding structures of practical linear block codes. Therefore, they still have limitations in terms of practical deployment in communication systems. In addition, neural network-based code design often fails to achieve performance close to theoretical limits because of the high non-differentiability of the code generation process and the difficulty of optimizing codes through neural decoders. Recently, E2E DC-ECCT [4], which applies a straight-through estimator (STE)-based method, was proposed to address this non-differentiability problem. E2E DC-ECCT combines ECCT, a Transformer-based decoder, with an end-to-end code design framework, thereby achieving improved decoding performance and linear block code design performance.

3. Background

3.1. Linear Block Codes

A linear block code C is defined by a generator matrix G of size $k \times n$ and a parity-check matrix H of size $(n - k) \times n$, satisfying $GH^T = 0$ over $\{0, 1\}$ under modulo-2 addition. A codeword $x \in C \subset \{0, 1\}^n$ is obtained by encoding a message m with the generator matrix G , such that $x = mG$. We assume an additive white Gaussian noise (AWGN) channel, in which the channel output is represented as $y = x_s + \varepsilon$, where x_s denotes the binary phase-shift keying (BPSK) signal of x , y is the noisy channel output corresponding to x_s , and ε is the noise. The objective of a decoder f is to correct errors and recover the transmitted codeword x . When y is received, the decoder checks the syndrome $s(y) = Hy_b$ to determine whether the received signal has been corrupted, where $y_b = \text{bin}(\text{sign}(y))$ is the decoded signal of y , $\text{sign}(a) = 1$ if $a \geq 0$, and -1 otherwise, $\text{bin}(-1) = 1$, $\text{bin}(1) = 0$. If $s(y)$ is a non-zero vector, the decoder detects that y has been corrupted during transmission and starts the error correction process.

In this paper, we adopt the preprocessing [18] of obtaining the magnitude vectors $|y|$ and the syndrome vectors $s(y)$ of the channel output y . This preprocessing has been pivotal in enabling data-driven decoders to mitigate the overfitting issue. Accordingly, the objective of the decoder is to estimate the multiplicative noise $\tilde{\varepsilon}$, which is defined as:

$$y = x_s + \varepsilon = x_s \tilde{\varepsilon}. \quad (1)$$

3.2. Transformer-Based Decoder

ECCT [2] uses the concatenation of the magnitude vectors $|y|$ and the syndrome vectors $s(y)$ obtained from the channel output y as the input vector. ECCT consists of N decoder blocks, where each decoder block is composed of self-attention and a feed-forward neural network (FFNN). In the attention process, a binary mask matrix is used to integrate information about the code. The mask matrix of ECCT includes all relationships between magnitude–magnitude, magnitude–syndrome, and syndrome–syndrome, and a mask matrix $g(H)$ derived from the parity-check matrix H is defined, where $g(H): \{0, 1\}^{(n-k) \times n} \rightarrow \{-\infty, 0\}^{(2n-k) \times (2n-k)}$. The output of the attention passes through the FFNN, then through the remaining decoder blocks, and finally, the output of the decoder block is fed into a prediction module consisting of two linear layers.

CrossMPT [3] uses the magnitude vector $|y|$ and the syndrome vector $s(y)$ obtained from the channel output y as separate inputs to the decoder. Similar to the principles of message-passing algorithms [27], $|y|$ and $s(y)$ are iteratively updated by exchanging messages. The CrossMPT decoder consists of two parallel decoders for the magnitude and syndrome vectors, each of which is composed of N decoder blocks. Each decoder block is composed of a cross-attention block and a feed-forward neural network (FFNN). Initially,

$|y|$ is passed as the query to the masked cross-attention block, while $s(y)$ is used as the key and value. The attention matrix is of size $n \times (n - k)$, representing the magnitude-syndrome relationship. To reflect this relationship, CrossMPT employs the transpose of the parity-check matrix $H^\top$ as the mask matrix $g(H^\top)$, where $g(H^\top) : \{0, 1\}^{n \times (n-k)} \rightarrow \{-\infty, 0\}^{n \times (n-k)}$. Subsequently, $s(y)$ is updated using the updated $|y|$ in another masked cross-attention block. Here, $s(y)$ is the query, while the key and value are the updated $|y|$. In this operation, the parity-check matrix H is used as the mask matrix $g(H)$. CrossMPT updates the magnitude and syndrome vectors through the iterative message exchange process based on cross-attention, enabling it to accurately estimate multiplicative noise. Finally, the outputs of the two decoders are concatenated and passed to a prediction module consisting of two linear layers.

3.3. Transformer-Based Code Construction and Decoder

E2E DC-ECCT [4] is the first study to design linear block codes using the Transformer-based decoder. This model overcomes the limitations of conventional deep learning-based linear block code design, for which differentiating binary code matrices and encoding operations is challenging.

First, the standard form is assumed to ensure efficient and differentiable optimization of the code matrices during deep learning training. The generator matrix in standard form is expressed as $G = [I^k, P]$, and the parity-check matrix as $H = [P^\top, I^{n-k}]$. Here, I denotes the identity matrix, $P \in \{0, 1\}^{k \times (n-k)}$, and $P^\top$ represents the transpose of P . To obtain a learnable code, the matrix P is parameterized and expressed as a learnable parameter P_Ω . Accordingly, given the parameterized generator matrix $G_\Omega = [I^k, P_\Omega]$ and the parity-check matrix $H_\Omega = [P_\Omega^\top, I^{n-k}]$, the neural decoder f_θ is defined within the end-to-end framework.

Subsequently, to address the major challenges in end-to-end linear block code design, which include the non-differentiable encoding operation ϕ and the bin function, a STE [28] is introduced and defined as

$$\begin{cases} \text{bin}(u) = \zeta^{-1}(\text{sign}(u)), \\ \frac{\partial \text{bin}(u)}{\partial u} \approx \frac{1_{|u| \leq \tau}}{2}, \end{cases} \quad (2)$$

where a clipped STE gradient with threshold $\tau = 0.5$ is used, following the setting of E2E DC-ECCT [4], to stabilize training and prevent the parameters Ω from exploding. Here, ζ is a bipolar mapping, i.e., $\zeta : \{0, 1\} \rightarrow \{\pm 1\}$, $\zeta^{-1} : \{\pm 1\} \rightarrow \{0, 1\}$. In addition, to optimize the encoding operation ϕ , a differentiable equivalent mapping of the XOR ($\oplus$) operation is used, yielding $\zeta(u \oplus v) = \zeta(u)\zeta(v)$, $\forall u, v \in \{0, 1\}$, which facilitates gradient-based optimization for the encoding operation. Without loss of generality, G_{Ω_i} denotes the i -th column of G_Ω and m is a binary message. Then, for all $i \in \{1, \dots, n\}$, the differentiable encoding operation ϕ is defined as

$$(\phi(m, G_\Omega))_i := G_{\Omega_i} \oplus m = \zeta^{-1} \left(\prod_{j=1}^k \zeta((G_\Omega)_{ij} \cdot m_j) \right), \quad (3)$$

where the term $\prod_{j=1}^k \zeta((G_\Omega)_{ij} \cdot m_j)$ does not represent a product of ordinary continuous values, but rather an equivalent representation of the XOR operation in the bipolar domain. Specifically, since ζ is a mapping that converts binary values into $\{+1, -1\}$, the magnitude of each term is always 1 after binarization. Therefore, the magnitude of the product also remains 1 regardless of the code length k , and the vanishing-gradient issue caused by individual terms approaching zero does not directly occur in this formulation.

Finally, in E2E DC-ECCT, the mask matrix is derived from the parity-check matrix and used within the decoder's attention mechanism. Through this mechanism, the model can obtain information about the code and thus learn the code structure. To enable this, a differentiable method for generating the mask matrix is proposed and defined as

$$g(H_{\Omega}) = \begin{pmatrix} H_{\Omega}^{\top} H_{\Omega} & H_{\Omega} \\ H_{\Omega}^{\top} & H_{\Omega} H_{\Omega}^{\top} \end{pmatrix}, \quad (4)$$

where $g(H_{\Omega})$ is of size $(2n - k) \times (2n - k)$.

4. Methodology

E2E DC-CrossMPT is an end-to-end model that simultaneously performs linear block code design and decoding. The overall framework is illustrated in Figure 1. To enable enhanced end-to-end code design and decoding, the proposed model employs the redesigned CrossMPT as the decoder, adapted to the end-to-end framework. In the redesigned decoder, the one-hot encoding is applied to the syndrome vectors in the initial embedding, which reduces the number of model parameters. The existing prediction module, composed of two linear layers, is modified to integrate the parity-check matrix, allowing the model to learn code information more effectively. Unlike E2E DC-ECCT, which processes the input as a single sequence, the proposed model takes the magnitude vector and the syndrome vector as separate inputs and is optimized using a message-passing-based approach that enables iterative information exchange between the two vectors. This process allows the model to extract the distinctive features of each vector. In this section, we describe the overall training process for E2E DC-CrossMPT and the detailed architecture of the redesigned decoder.

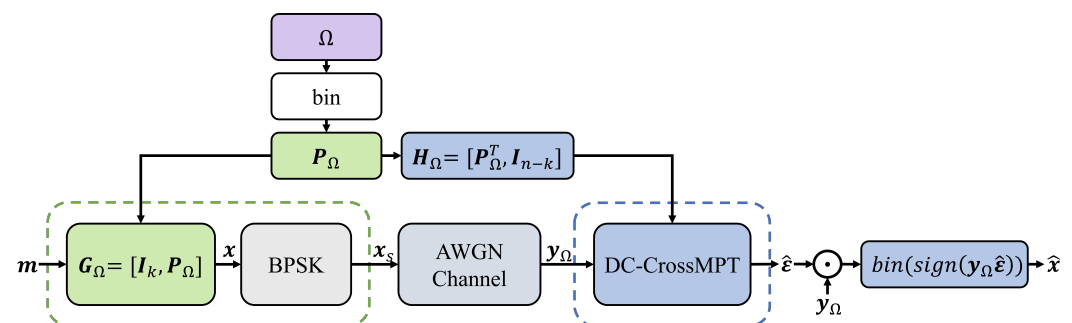


Figure 1. The proposed end-to-end communication system.

4.1. E2E DC-CrossMPT Based Code Construction

E2E DC-CrossMPT adopts the standard form of the parity-check matrix, the STE method, and the differentiable encoding operation ϕ from the conventional model to ensure efficient and differentiable optimization of the binary code matrix. In the training process, E2E DC-CrossMPT aims to estimate the multiplicative noise $\tilde{\epsilon}$. Accordingly, the objective of the neural network decoder f_{θ} is to estimate $\hat{\epsilon}$, i.e., $f_{\theta}(y_{\Omega}) = \hat{\epsilon}$, and the estimate of x is given as $\hat{x} = \text{bin}(\text{sign}(y_{\Omega} f_{\theta}(y_{\Omega})))$. If the estimated multiplicative noise $\hat{\epsilon}$ is correct, then $\text{sign}(\tilde{\epsilon}) = \text{sign}(\hat{\epsilon})$ and $\text{sign}(\tilde{\epsilon}\hat{\epsilon}) = 1$, in which case $\hat{x}$ is obtained as follows:

$$\hat{x} = \text{bin}(\text{sign}(y_{\Omega} f_{\theta}(y_{\Omega}))) = \text{bin}(\text{sign}(x_s \tilde{\epsilon}\hat{\epsilon})) = \text{bin}(\text{sign}(x_s)) = x. \quad (5)$$

Finally, the proposed model is optimized using the cross-entropy loss, defined as

$$L_{\theta, \Omega} = - \sum_{i=1}^n \left[\tilde{\epsilon}_i \log(\sigma(f_{\theta}(y_{\Omega})_i)) + (1 - \tilde{\epsilon}_i) \log(1 - \sigma(f_{\theta}(y_{\Omega})_i)) \right], \quad (6)$$

where $\sigma(\cdot)$ denotes the sigmoid function. The model predicts a binary error vector corresponding to $\text{sign}(\tilde{\epsilon})$, rather than the continuous multiplicative noise $\tilde{\epsilon}$ itself.

4.2. DC-CrossMPT-Based Decoder

The redesigned decoder of E2E DC-CrossMPT is divided into three main components: the initial embedding module, N decoder blocks, and the prediction module. The decoder architecture is illustrated in Figure 2. In Figure 2, B , k , n , and d represent the batch size, message length, codeword length, and embedding dimension, respectively. The inputs to the model consist of the magnitude vectors $|y_\Omega|$ and the syndrome vectors $s(y_\Omega)$, which are obtained from the received signals y_Ω using the same preprocessing method used in the conventional E2E DC-ECCT [4]. The query, key, and value corresponding to $|y_\Omega|$ are denoted as Q_m , K_m , and V_m , respectively, while those corresponding to $s(y_\Omega)$ are denoted as Q_s , K_s , and V_s .

First, the redesigned decoder uses the magnitude vectors $|y_\Omega|$ and the syndrome vectors $s(y_\Omega)$ as separate inputs. In this design, the initial embedding method for the syndrome vectors is modified from the conventional embedding representation to one-hot encoding. The one-hot embedding method assigns trainable parameters only to the binary values $\{0, 1\}$, resulting in a constant number of two parameters regardless of the syndrome length. Accordingly, the initial embedding of the model uses a d -dimensional one-hot encoding for the syndrome vectors $s(y_\Omega)$, and a d -dimensional embedding for the magnitude vectors $|y_\Omega|$. Thus, the initial positional embeddings $\Phi_m \in \mathbb{R}^{n \times d}$, $\Phi_s \in \mathbb{R}^{(n-k) \times d}$ are defined as

$$\Phi_m, \Phi_s = |y_\Omega|^\top W_m, s(y_\Omega) W_s, \quad (7)$$

where $W_m \in \mathbb{R}^d$ represents the weight of the magnitude embeddings, and W_s denotes the weight corresponding to the one-hot encoding of the binary syndrome values, where the weight for 0 is expressed as $W_0 \in \mathbb{R}^d$ and the weight for 1 is expressed as $W_1 \in \mathbb{R}^d$.

Subsequently, the two embedded vectors Φ_m and Φ_s are used as separate inputs to the decoder. Each of the two decoders consists of N decoder blocks, and each decoder block is composed of a cross-attention block and an FFNN. E2E DC-CrossMPT incorporates information about the code into the attention mechanism by using the mask matrix based on the parity-check matrix. The conventional E2E DC-ECCT uses the mask matrix $g(H_\Omega)$ derived from the parity-check matrix, which is of size $(2n - k) \times (2n - k)$. In contrast, E2E DC-CrossMPT uses the mask matrix H_Ω of size $(n - k) \times n$. In both models, a parameterized mapping $\psi(\cdot)$ is applied to the corresponding mask matrix to obtain a learnable attention mask. Accordingly, in the decoder for learning the magnitude embeddings (the magnitude decoder), the transpose of the parity-check matrix $H_\Omega^\top$ is used as the mask matrix, while in the decoder for learning the syndrome embeddings, the parity-check matrix H_Ω is used as the mask matrix. The cross-attention mechanism in the magnitude decoder is given by

$$A(Q_m, K_s, V_s) = \text{softmax}\left(\frac{Q_m K_s^\top + \psi(H_\Omega^\top)}{\sqrt{d}}\right) V_s, \quad (8)$$

where $\psi(\cdot)$ consists of two linear layers and an ReLU activation function.

Finally, the prediction module applies the code-aware prediction method [22] that incorporates the parameterized H_Ω , instead of the conventional structure composed of two linear layers. In this method, the two embeddings Φ_m, Φ_s obtained after passing through the decoder blocks are each processed by linear layers. The updated Φ_s is augmented with the code information by integrating the parity-check matrix H_Ω . The outputs of the two embeddings are then added, and the final linear layer reduces the output dimension from d to 1. The prediction module for the embeddings Φ_m, Φ_s is expressed as follows

$$\hat{\varepsilon} = \left(\Phi_m W_m + H_{\Omega}^{\top} (\Phi_s W_s) \right) W_{d \rightarrow 1}, \quad (9)$$

where $W_s, W_m \in \mathbb{R}^{d \times d}$ and $W_{d \rightarrow 1} \in \mathbb{R}^d$ represent the weights of the embedding layers. $d \rightarrow 1$ denotes the reduction of a d -dimensional vector to a 1-dimensional vector. The redesigned prediction module is a code-aware prediction module designed to explicitly incorporate the bit-check connectivity defined by the parity-check matrix H_{Ω} into the final error prediction stage. Specifically, the term $H_{\Omega}^{\top} (\Phi_s W_s)$ in Equation (9) projects the syndrome representation onto bit-wise predictions. This allows each bit prediction to aggregate only the syndrome representations of the parity-check nodes connected to the corresponding bit in H_{Ω} , rather than indiscriminately using the entire syndrome feature. Therefore, Equation (9) can be interpreted as being analogous to check-to-bit message aggregation in a parity-check graph, providing code structure-aware information at the final prediction stage.

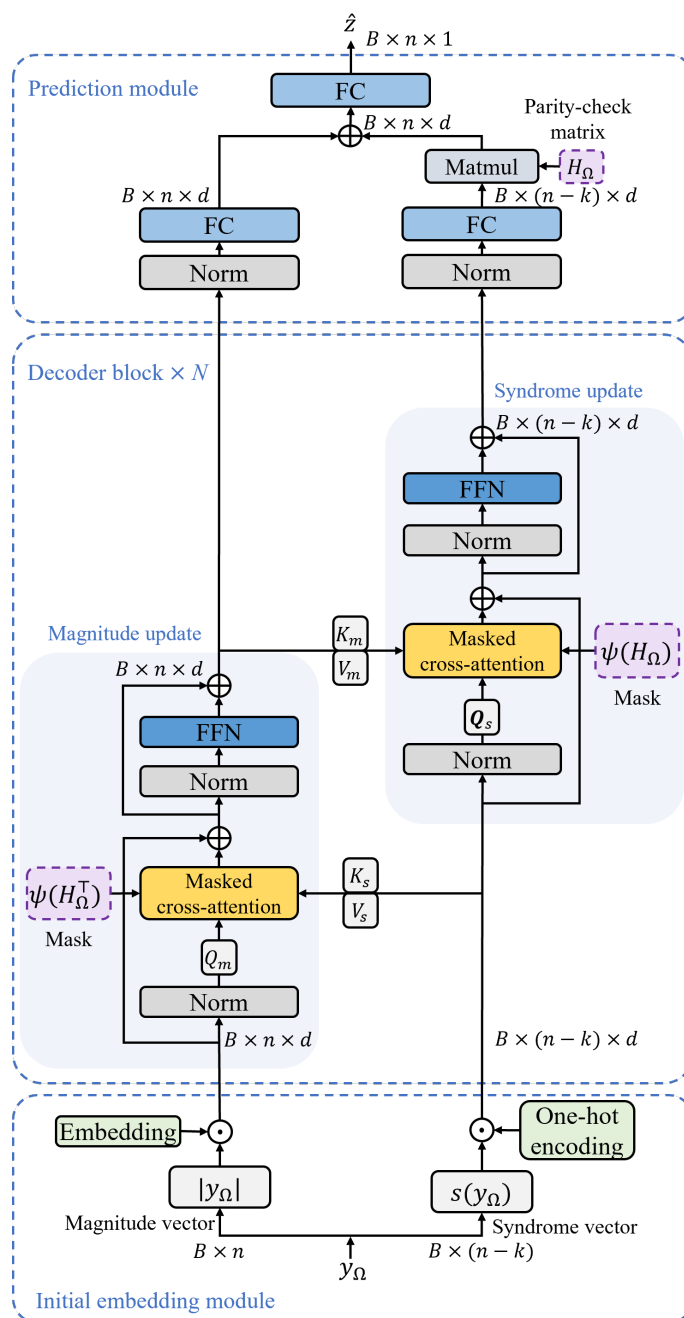


Figure 2. Architecture of modified DC-CrossMPT.

5. Results

In this section, we evaluate the performance of E2E DC-CrossMPT on BCH, LDPC, Polar, and RS codes with various code lengths and code rates. All experiments are conducted using the PyTorch 2.1.0+cu118 framework on a workstation equipped with a single NVIDIA RTX 4090 GPU. The proposed model employs decoder blocks $N = 2$, attention heads $h = 8$, and embedding dimension $d = 32$. For a fair comparison between E2E DC-CrossMPT and E2E DC-ECCT, we use the same training environment. Thus, the proposed model is trained using the all-ones message and optimized with Adam for 1000 epochs. Each epoch consists of 1000 mini-batches, and each mini-batch contains 1024 samples. The initial learning rate for optimization is set to 10^{-4} and gradually decays to 10^{-6} following a cosine annealing scheduler. After the training process, inference is performed in the SNR range from 4 dB to 6 dB until 200 frame errors are observed. For performance evaluation, all test results are obtained using randomly generated message bits, and the negative natural logarithm of the bit error rate (BER) is used as the evaluation metric.

Table 1 presents the decoding results of a traditional decoding algorithm such as BP, conventional Transformer-based decoders such as ECCT, DC-ECCT, E2E DC-ECCT, and CrossMPT, and the proposed DC-CrossMPT and E2E DC-CrossMPT. All E2E-based models jointly optimize the encoder and decoder during training but are evaluated on standard linear block codes during testing, without modifying the code structure. DC-CrossMPT fixes the parity-check matrix H_Ω of E2E DC-CrossMPT and trains only the decoder. The number of iterations for BP is set to five, consistent with the settings used in prior work [4]. For a fair comparison, we retrain CrossMPT under the same experimental environment as the proposed model. As a result, the proposed DC-CrossMPT and E2E DC-CrossMPT achieved improved decoding performance compared with the considered neural network-based decoders and the low-iteration BP performance. These results demonstrate that the redesigned decoder of E2E DC-CrossMPT effectively learns the distinctive features of the magnitude and syndrome vectors by using separate inputs. Moreover, the integration of the parity-check matrix in the prediction module enables the model to capture information about the code more effectively, leading to improved performance. However, it should be noted that BP decoders with a larger number of iterations can achieve higher decoding performance at the cost of increased decoding latency. Therefore, the proposed model is not intended to universally replace fully converged conventional decoders under all conditions, but rather to improve the trade-off between performance and complexity within a Transformer-based end-to-end code design and decoding framework. In addition, to facilitate comparison with other studies, we present additional BER-based results in Figure 3, in addition to the negative natural logarithm of BER values reported in Table 1. The BER curves were evaluated over the E_b/N_0 range from 0 dB to 6 dB. Under the same evaluation settings, the proposed model achieved better BER performance than the existing E2E DC-ECCT-based model across the considered code configurations.

Table 1. Comparison of decoding performance.

Methods	BP			ECCT			DC-ECCT			E2E DC-ECCT			CrossMPT			DC-CrossMPT			E2E DC-CrossMPT		
Codes	4 dB	5 dB	6 dB	4 dB	5 dB	6 dB	4 dB	5 dB	6 dB	4 dB	5 dB	6 dB	4 dB	5 dB	6 dB	4 dB	5 dB	6 dB	4 dB	5 dB	6 dB
BCH (31,16)	4.59	5.87	7.57	4.61	5.97	7.69	4.97	6.56	8.54	5.30	6.89	9.09	5.19	6.76	8.72	5.60	7.37	9.75	5.91	7.73	9.97
BCH (63,45)	4.07	4.92	6.03	4.59	6.07	8.14	4.54	6.07	8.14	4.98	6.69	8.97	4.88	6.64	9.05	5.06	6.85	9.38	5.52	7.48	10.15
LDPC (49,24)	5.25	7.15	9.86	4.21	5.32	6.56	4.08	5.29	6.55	4.95	6.46	8.33	4.32	5.62	7.41	4.56	5.97	7.43	5.68	7.47	9.66
LDPC (49,36)	4.09	4.48	5.34	5.13	6.57	8.55	5.18	6.76	8.80	5.51	7.30	9.61	5.43	7.12	9.32	5.39	7.05	9.33	5.51	7.52	10.02
POLAR (32,11)	3.29	3.77	4.21	4.46	5.57	7.01	4.53	5.69	7.08	4.65	5.81	7.28	5.20	6.60	8.35	5.32	6.41	8.21	5.43	6.79	8.64
POLAR (64,32)	3.53	4.02	4.45	4.56	5.93	7.75	4.40	5.76	7.67	4.72	6.22	8.13	4.95	6.47	8.40	5.15	7.07	9.18	5.55	7.23	9.49
RS (60,52)	4.43	5.32	6.43	4.37	5.11	6.03	5.04	6.68	8.82	5.12	6.80	9.02	5.09	6.82	9.15	5.23	7.05	9.50	5.36	7.37	9.37

The highest score is indicated in **bold**.

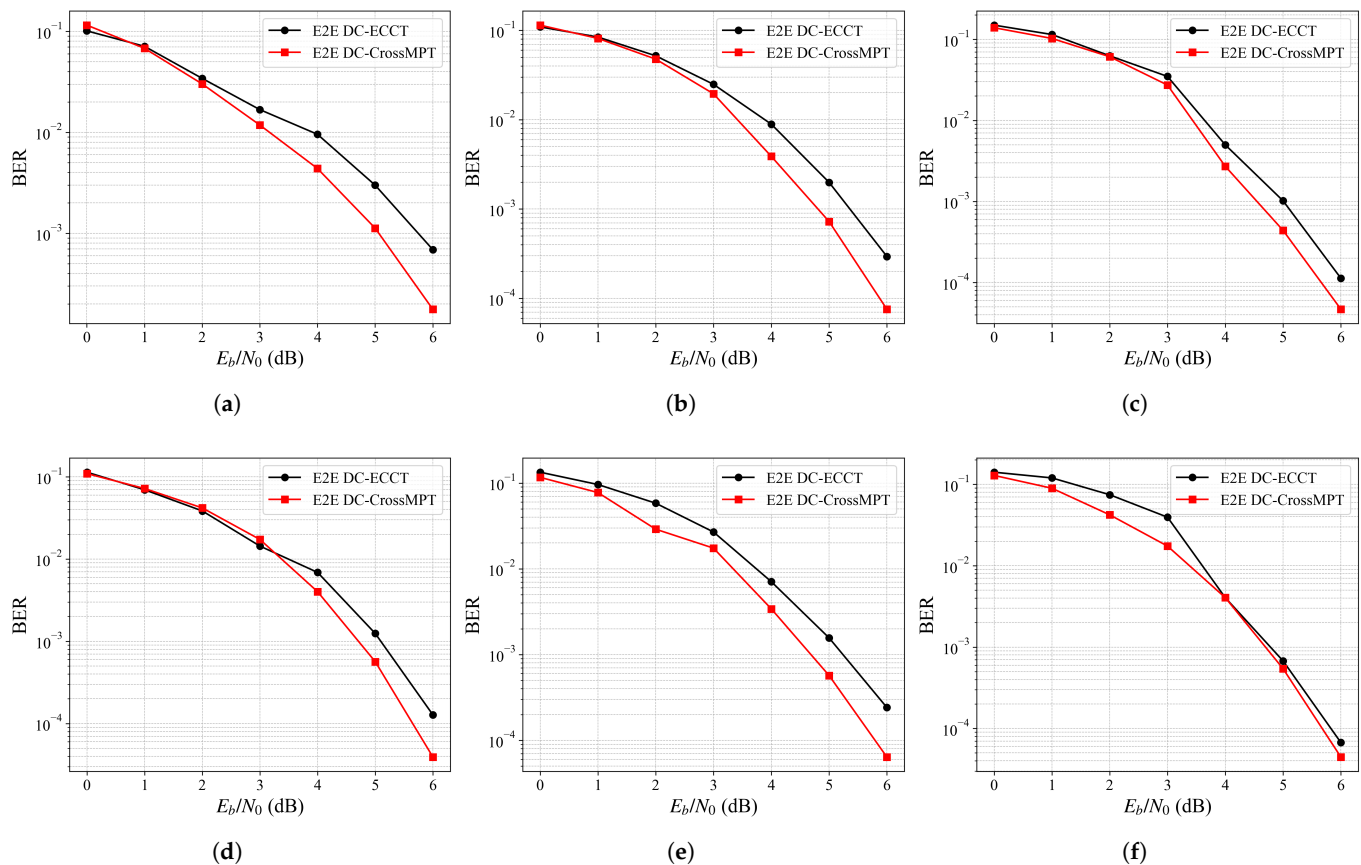


Figure 3. Comparison of BER performance between E2E DC-ECCT and E2E DC-CrossMPT: (a) BCH (31,16) code. (b) BCH (63,45) code. (c) LDPC (49,24) code. (d) LDPC (49,36) code. (e) POLAR (32,11) code. (f) POLAR (64,32) code.

Table 2 presents the decoding performance, evaluated using the BP algorithm, of the codes designed by E2E DC-CrossMPT. For a fair comparison, the BP algorithm is iterated five times, as in the conventional E2E DC-ECCT. The conventional model is retrained with $N = 2$ and $d = 32$ to design the codes. The Conventional Code in Table 2 corresponds to the systematic form of conventional linear block codes. Experimental results show that the codes designed by E2E DC-CrossMPT achieve the best performance among all compared methods. In particular, the (49,24) code shows significantly improved performance compared to the other two codes. The relatively large gain observed for the (49,24) code can be attributed to the structural similarity between the proposed CrossMPT-based decoder and Tanner graph-based message-passing decoding. In LDPC decoding, variable nodes and check nodes iteratively exchange reliability information and parity-check constraint information. Similarly, the proposed decoder separately processes the magnitude information $|y|$ and the syndrome information $s(y)$, and learns their interactions through masked cross-attention mechanisms induced by H_Ω and $H_\Omega^\top$. Since LDPC codes are characterized by sparse parity-check matrices, this structure can effectively exploit the relationship between bit-level reliability and parity-check constraints. These results indicate that the efficient and robust structure of E2E DC-CrossMPT is highly effective in designing linear block codes with superior error-correction capability.

Table 3 presents the comparison of FLOPs, inference time, and training time between E2E DC-ECCT and E2E DC-CrossMPT. Here, inference time represents the decoding time for a single codeword, and training time is defined as the time per epoch divided by the batch size. The FLOPs of E2E DC-CrossMPT are significantly lower than E2E DC-ECCT

across all codes. In particular, the proposed model achieves the largest FLOPs reduction of approximately 33.3% for the (60,52) code. E2E DC-CrossMPT also achieves substantial reductions in training and inference time across all codes compared to E2E DC-ECCT. Specifically, for the (64,32) code, the training time is reduced by about 29.21%, and the inference time is reduced by about 86.29%, demonstrating a significant improvement. The reduction in computational overhead of the proposed model mainly stems from the difference in attention mask size and input structure. The existing E2E DC-ECCT concatenates the magnitude vector and syndrome vector into a single sequence and then performs self-attention, resulting in an attention mask size of $(2n - k) \times (2n - k)$. In this case, magnitude-to-magnitude, magnitude-to-syndrome, syndrome-to-magnitude, and syndrome-to-syndrome relationships are all included in the attention operation. In contrast, the proposed E2E DC-CrossMPT keeps the magnitude vector and syndrome vector as separate inputs and performs information exchange between the two representations through masked cross-attention. Therefore, the attention mask is constructed with a size of $n \times (n - k)$ or $(n - k) \times n$ based on H_{Ω} or $H_{\Omega}^{\top}$, and the number of attention score matrices to be considered is significantly reduced compared with the self-attention-based structure. Owing to this structural difference, the FLOPs of the attention operation are reduced, and consequently, the training time and inference time are also decreased.

Table 2. Comparison of decoding performance using H_{Ω} with BP.

Methods	Conventional Code			E2E DC-ECCT			E2E DC-CrossMPT		
Codes	4 dB	5 dB	6 dB	4 dB	5 dB	6 dB	4 dB	5 dB	6 dB
(31,16)	4.00	4.60	5.59	2.95	3.37	3.70	4.69	5.43	6.27
(63,45)	4.37	5.33	6.42	3.24	3.45	3.56	4.82	5.74	6.75
(49,24)	3.49	4.12	4.88	5.41	7.02	8.80	6.15	8.10	9.44
(49,36)	3.95	4.60	5.25	3.14	3.10	3.56	4.14	4.61	5.30
(60,52)	3.91	4.99	6.32	3.70	3.99	4.32	4.45	5.14	6.35

The highest score is indicated in **bold**.

Table 3. Comparison of FLOPs, inference time, and training time.

Methods	E2E DC-ECCT			E2E DC-CrossMPT		
Codes	FLOPs	Inference Time	Training Time	FLOPs	Inference Time	Training Time
(32,11)	6.03 M	3.45 μ s	15.34 ms	4.93 M	0.95 μ s	12.59 ms
(64,32)	15.88 M	7.88 μ s	36.62 ms	11.73 M	1.08 μ s	25.93 ms
(31,16)	4.75 M	2.21 μ s	12.59 ms	3.89 M	0.90 μ s	10.09 ms
(63,45)	11.11 M	6.20 μ s	27.50 ms	7.92 M	1.07 μ s	19.12 ms
(49,24)	10.20 M	5.38 μ s	28.28 ms	7.86 M	1.08 μ s	19.44 ms
(49,36)	7.38 M	12.21 μ s	26.26 ms	5.51 M	7.14 μ s	13.40 ms
(60,52)	7.62 M	4.99 μ s	22.69 ms	5.08 M	3.36 μ s	14.78 ms

5.1. Ablation Studies

In this paper, ablation studies are conducted using the BCH (31,16), LDPC (49,24), and Polar (32,11) codes to analyze the impact of key components in the proposed E2E DC-CrossMPT on decoding performance and model complexity. Ablation 1's decoder corresponds to the conventional CrossMPT. Ablation 2's decoder applies a one-hot embedding method to the syndrome vectors in the conventional CrossMPT. Ablation 3's decoder incorporates the parity-check matrix into the prediction module of the conventional CrossMPT.

Table 4 presents the results obtained by applying each decoder to the E2E framework to analyze the impact of key components in the proposed E2E DC-CrossMPT on decoding performance. The experimental results show that the E2E model using the proposed decoder achieves the best decoding performance in most cases. In addition, compared to Ablation 1,

Ablation 2 and Ablation 3 consistently achieve improved decoding performance across all codes when the proposed components are individually applied. These results indicate that the key components introduced in the proposed model contribute to the improvement of decoding performance.

Table 5 compares the FLOPs and the number of parameters of each decoder for a single codeword to analyze the impact of the key components on model complexity. The results show that Ablation 2 has the smallest number of parameters, indicating that the syndrome one-hot embedding method reduces the number of parameters compared to the conventional CrossMPT. In contrast, Ablation 3 exhibits the largest number of parameters and FLOPs, since incorporating the parity-check matrix into the prediction module introduces additional computations and trainable parameters. In addition to the ablation results, when comparing DC-ECCT and DC-CrossMPT, DC-ECCT has a smaller number of parameters. However, despite having fewer parameters, DC-ECCT results in larger FLOPs. This is because the conventional DC-ECCT uses a larger mask size, $(2n - k) \times (2n - k)$, than the proposed DC-CrossMPT, $n \times (n - k)$, thereby increasing computational complexity during the attention mechanism. These results demonstrate that the proposed model has an efficient and robust architecture capable of achieving improved decoding performance with reduced computational complexity.

Table 4. Ablation results comparing the impact of individual components on decoding performance.

Codes	BCH (31,16)			LDPC (49,24)			Polar (32,11)		
Methods	4 dB	5 dB	6 dB	4 dB	5 dB	6 dB	4 dB	5 dB	6 dB
E2E DC-CrossMPT	5.91	7.73	9.97	5.68	7.47	9.66	5.43	6.79	8.64
Ablation1	5.24	6.92	8.90	4.86	6.29	8.19	5.15	6.53	8.07
Ablation2	5.62	7.39	9.61	5.11	6.75	8.76	5.48	6.89	8.80
Ablation3	5.42	7.13	9.31	5.16	6.93	8.96	5.27	6.69	8.49

The highest score is indicated in **bold**.

Table 5. Ablation results comparing the impact of individual components on complexity.

Codes	BCH (31,16)		LDPC (49,24)		Polar (32,11)	
Methods	FLOPs	Parameters	FLOPs	Parameters	FLOPs	Parameters
E2E DC-ECCT	4.75 M	28,175	5.10 M	28,775	3.01 M	28,406
E2E DC-CrossMPT	3.89 M	29,167	3.92 M	31,225	2.46 M	29,462
Ablation1	2.88 M	28,800	2.62 M	32,514	2.46 M	29,526
Ablation2	2.88 M	27,520	2.62 M	30,338	1.74 M	28,022
Ablation3	3.89 M	29,455	3.92 M	30,951	1.74 M	29,910

5.2. Stability Analysis

To analyze the training stability of the model, this paper presents the loss, the BER, the variation of the parity-check matrix, $\|H_t - H_0\|_1$, and the gradient norm observed during the training process of the BCH (31,16) code in Figure 4. The results show that the loss gradually decreases and the BER stabilizes during training. In addition, the variation of the parity-check matrix remains within a bounded range without abrupt divergence. These results indicate that severe training instability or parameter divergence caused by gradient mismatch was not observed during the learning process of the binary code matrix. These observations suggest that the use of clipped STE, a small learning rate, and a cosine annealing scheduler helps stabilize the learning of the binary code matrix.

Additionally, to analyze the stability and robustness of the proposed model under changes in channel conditions, we conducted experiments on the Rayleigh fading channel, as shown in Table 6. The results show that E2E DC-CrossMPT outperforms E2E DC-ECCT

for all codes, indicating that the proposed model remains more robust than the existing model even under the Rayleigh fading channel.

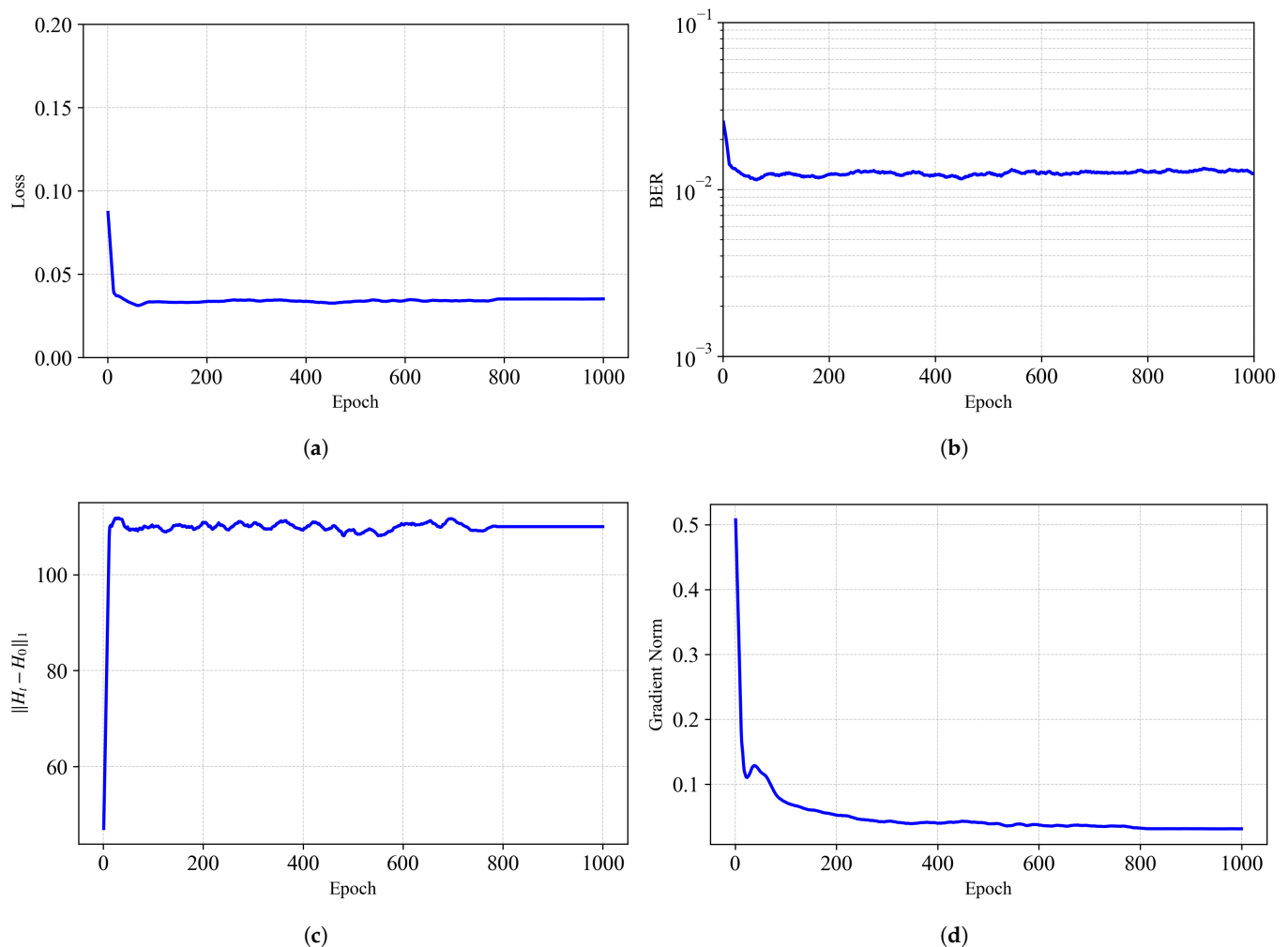


Figure 4. Training stability curves: (a) Training loss. (b) Training BER. (c) Variation of the parity-check matrix $\|H_t - H_0\|_1$. (d) Training gradient normalization.

Table 6. Comparison of decoding performance in Rayleigh fading.

Codes	BCH (31,16)			LDPC (49,24)			Polar (32,11)		
Methods	4 dB	5 dB	6 dB	4 dB	5 dB	6 dB	4 dB	5 dB	6 dB
E2E DC-CrossMPT	2.49	2.87	3.28	2.23	2.54	2.92	2.62	2.94	3.45
E2E DC-ECCT	2.37	2.71	3.05	2.14	2.50	2.69	2.55	2.82	3.35

The highest score is indicated in **bold**.

6. Discussion

In this paper, we propose E2E DC-CrossMPT, an end-to-end model designed to simultaneously improve the design and decoding performance of linear block codes. The proposed model redesigns the conventional CrossMPT architecture for an end-to-end code design framework and uses the magnitude and syndrome vectors obtained from the received signals as separate inputs. In addition, the decoder applies a one-hot encoding-based embedding to the syndrome vector and directly incorporates the parity-check matrix into the prediction module, allowing the code structure information to be reflected more effectively. The redesigned decoder independently embeds the magnitude information and syndrome information of the received signal into different representation spaces, and then

learns the interaction between these two types of information through cross-attention. This structure is conceptually similar to the message-passing-based decoding process widely used for linear block codes. For example, in the Tanner graph of an LDPC code, variable nodes represent the reliability information of the received bits, while check nodes represent parity-check constraints. A conventional message-passing decoder estimates the error pattern by iteratively exchanging messages between variable nodes and check nodes. Similarly, the cross-attention mechanism in the proposed CrossMPT learns to model the relationship between the magnitude representation, which represents bit-wise reliability, and the syndrome representation, which reflects parity-check constraints. In particular, by using a parity-check matrix-based mask, the attention mechanism is encouraged to perform information exchange that reflects the structural constraints of the code. Therefore, the proposed cross-attention-based decoder structure can be interpreted as a data-driven generalization of fixed message-passing rules such as BP, and this is one of the main factors contributing to the improved decoding performance.

Experimental results showed that the proposed E2E DC-CrossMPT achieved superior decoding performance compared with conventional decoding algorithms and existing Transformer-based models across various linear block codes, including BCH, Polar, LDPC, and RS codes. The codes designed by the proposed model also exhibited enhanced error correction capability compared with conventional linear block codes and the codes designed by E2E DC-ECCT when evaluated using the BP decoder. In addition, E2E DC-CrossMPT reduced FLOPs, training time, and inference time compared with the conventional E2E DC-ECCT. Therefore, the proposed model can be regarded as an efficient end-to-end code design and decoding framework that simultaneously improves decoding performance and reduces computational complexity.

The threshold τ of the clipped STE used in this study is an important hyperparameter that can affect the training stability of end-to-end code design. Following the existing E2E DC-ECCT, we fix $\tau = 0.5$, which enables a fair comparison under the same STE setting. The value of τ does not change the binarization result in the forward pass. Instead, it restricts the range of surrogate gradient propagation in the backward pass, thereby mitigating excessive updates of the learnable code matrix Ω . However, since STE approximates the non-differentiable sign/bin operations using surrogate gradients, a gradient mismatch exists between the forward operation and the backward gradient, and convergence to the global optimum is not theoretically guaranteed. Therefore, STE-based learning of a binary code matrix should be interpreted as a practical approximate optimization method for the discrete code design problem.

This study has several limitations in that the experiments were conducted only on a limited set of model sizes and code configurations. For larger models and longer block lengths, memory usage and training time increase significantly because the learnable code matrices and decoder parameters must be optimized simultaneously. Therefore, due to computational resource constraints, larger-scale models with deeper decoder structures or larger embedding dimensions could not be fully evaluated in this study. Further experiments are needed in future work to verify whether the performance improvements of the proposed model can be consistently maintained across larger models, longer block lengths, and various code rates. We also plan to conduct a more comprehensive complexity-performance Pareto-front analysis by considering fully converged BP/SC decoders, various numbers of iterations, and hardware-optimized implementations.

Another future research direction is to extend the proposed E2E DC-CrossMPT to integrated 6G physical-layer systems. In this study, we validated the code design and decoding performance mainly under the AWGN channel and a 1D bit-stream representation. However, in recent 6G vehicular and aerial communications, high-dimensional and

dynamic environments such as UAV-based aerial–terrestrial integration, integrated sensing and communication (ISAC), stacked intelligent metasurface (SIM)-based propagation control, distributed arrays, and intelligent reflecting surface (IRS)-assisted transmission have become increasingly important [29–31]. In such environments, spatial channel variations, mobility, interference, sensing information, and physical-layer security can directly affect decoding reliability and code stability. Therefore, future work may consider integrating environmental information, such as location, velocity, channel state, sensing features, and SIM configurations, into the decoder or the code design process. In addition, distributed array and IRS-based secure transmission techniques can improve security and interference resilience in the spatial domain, and thus may be combined with E2E DC-CrossMPT in a complementary manner, which improves reliability in the coding/decoding domain.

7. Conclusions

In this paper, we proposed E2E DC-CrossMPT, an end-to-end model that enhances linear block code design and decoding via a redesigned decoder. The redesigned decoder applied a one-hot encoding-based embedding to the syndrome and integrated the parity-check matrix into the prediction module, thereby reducing model complexity while improving decoding performance. Experimental results showed that E2E DC-CrossMPT improved decoding performance and error-correction capability and reduced complexity across the considered codes compared with conventional methods. These results indicate that the proposed decoder structure can effectively learn code-aware representations and provide an efficient framework for joint code design and decoding. In future work, once the current computational limitations are alleviated, we aim to conduct experiments on a wider range of code rates and block lengths, as well as larger-scale models.

Author Contributions: Conceptualization, Y.C. and J.K.; methodology, Y.C.; software, Y.C.; validation, Y.C.; formal analysis, Y.C.; investigation, Y.C.; writing—original draft, Y.C.; writing—review and editing, J.K.; visualization, Y.C.; supervision, J.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) under the virtual convergence support program to nurture the best talents (IITP-2026-RS-2023-00254529) grant funded by the Korea government (MSIT). This work was supported by the “Regional Innovation System & Education (RISE)” through the Seoul RISE Center, funded by the Ministry of Education (MOE) and the Seoul Metropolitan Government (2025-RISE01-019-04).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Lee, H.; Lee, B.; Yang, H.; Kim, J.; Kim, S.; Shin, W.; Shim, B.; Poor, H.V. Towards 6G hyper-connectivity: Vision, challenges, and key enabling technologies. *J. Commun. Netw.* **2023**, *25*, 344–354. [CrossRef]
2. Choukroun, Y.; Wolf, L. Error correction code transformer. In Proceedings of the Advances in Neural Information Processing Systems 35 (NeurIPS), New Orleans, LA, USA, 28 November–9 December 2022; pp. 38695–38705.
3. Park, S.J.; Kwak, H.Y.; Kim, S.H.; Kim, Y.; No, J.S. CrossMPT: Cross-attention message-passing transformer for error correcting codes. In Proceedings of the 13th International Conference on Learning Representations (ICLR), Singapore, 24–28 April 2025; pp. 1–19.
4. Choukroun, Y.; Wolf, L. Learning linear block error correction codes. In Proceedings of the 41st International Conference on Machine Learning (ICML), Vienna, Austria, 21–27 July 2024; pp. 8801–8814.

5. Nachmani, E.; Beery, Y.; Burshtein, D. Learning to decode linear codes using deep learning. In Proceedings of the 54th Annual Allerton Conference on Communication, Control, and Computing, Monticello, IL, USA, 27–30 September 2016; pp. 341–346.
6. Gruber, T.; Cammerer, S.; Hoydis, J.; Brink, T. On deep learning-based channel decoding. In Proceedings of the 51st Annual Conference on Information Sciences and Systems, Baltimore, MD, USA, 22–24 March 2017; pp. 1–6.
7. Kim, H.; Jiang, Y.; Rana, R.; Kannan, S.; Oh, S.; Viswanath, P. Communication algorithms via deep learning. In Proceedings of the 6th International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, 30 April–3 May 2018; pp. 1–17.
8. Nachmani, E.; Marciano, E.; Lugosch, L.; Gross, W.J.; Burshtein, D.; Beery, Y. Deep learning methods for improved decoding of linear codes. *IEEE J. Sel. Top. Signal Process.* **2018**, *12*, 119–131. [[CrossRef](#)]
9. Dai, J.; Tan, K.; Si, Z.; Niu, K.; Chen, M.; Poor, H.V.; Cui, S. Learning to decode protograph LDPC codes. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 1983–1999. [[CrossRef](#)]
10. Nachmani, E.; Wolf, L. Hyper-graph-network decoders for block codes. In Proceedings of the Advances in Neural Information Processing Systems 32 (NeurIPS), Vancouver, BC, Canada, 8–14 December 2019; pp. 2326–2336.
11. Nachmani, E.; Wolf, L. Autoregressive belief propagation for decoding block codes. *arXiv* **2021**, arXiv:2103.11780. [[CrossRef](#)]
12. Kwak, H.-Y.; Yun, D.-Y.; Kim, Y.; Kim, S.-H.; No, J.-S. Boosting learning for LDPC codes to improve the error-floor performance. In Proceedings of the Advances in Neural Information Processing Systems 36 (NeurIPS), New Orleans, LA, USA, 10–16 December 2023; pp. 22115–22131.
13. Kwak, H.-Y.; Kim, J.-W.; Kim, Y.; Kim, S.-H.; No, J.-S. Neural min-sum decoding for generalized LDPC codes. *IEEE Commun. Lett.* **2022**, *26*, 2841–2845. [[CrossRef](#)]
14. Kwak, H.-Y.; Yun, D.-Y.; Kim, Y.; Kim, S.-H.; No, J.-S. Boosted neural decoders: Achieving extreme reliability of LDPC codes for 6G networks. *IEEE J. Sel. Areas Commun.* **2025**, *43*, 1089–1102. [[CrossRef](#)]
15. Buchberger, A.; Hager, C.; Pfister, H.D.; Schmalen, L.; i Amat, A.G. Pruning and quantizing neural belief propagation decoders. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 1957–1966.
16. Cammerer, S.; Gruber, T.; Hoydis, J.; ten Brink, S. Scaling deep learning-based decoding of polar codes via partitioning. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Singapore, 4–8 December 2017; pp. 1–6.
17. Jiang, Y.; Kannan, S.; Kim, H.; Oh, S.; Asnani, H.; Viswanath, P. DEEPTURBO: Deep turbo decoder. In Proceedings of the IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), Cannes, France, 2–5 July 2019; pp. 1–5.
18. Bennatan, A.; Choukroun, Y.; Kisilev, P. Deep learning for decoding of linear codes—A syndrome based approach. In Proceedings of the IEEE International Symposium on Information Theory (ISIT), Vail, CO, USA, 17–22 June 2018; pp. 1595–1599.
19. Levy, M.; Choukroun, Y.; Wolf, L. Accelerating error correction code transformers. In Proceedings of the 13th International Conference on Learning Representations (ICLR), Singapore, 24–28 April 2025; pp. 1–22.
20. Choukroun, Y.; Wolf, L. Denoising diffusion error correction codes. In Proceedings of the 11th International Conference on Learning Representations (ICLR), Kigali, Rwanda, 1–5 May 2023; pp. 1–18.
21. Park, S.-J.; Kwak, H.-Y.; Kim, S.-H.; Kim, S.; Kim, Y.; No, J.-S. Multiple-masks error correction code transformer for short block codes. *IEEE J. Sel. Areas Commun.* **2025**, *43*, 2518–2529. [[CrossRef](#)]
22. Choukroun, Y.; Wolf, L. A foundation model for error correction codes. In Proceedings of the 12th International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024; pp. 1–19.
23. Jiang, Y.; Kim, H.; Asnani, H.; Kannan, S.; Oh, S.; Viswanath, P. LEARN codes: Inventing low-latency codes via recurrent neural networks. In Proceedings of the IEEE International Conference on Communications (ICC 2019), Shanghai, China, 20–24 May 2019; pp. 1–7.
24. Ebada, M.; Cammerer, S.; Elkelesh, A.; ten Brink, S. Deep learning-based polar code design. In Proceedings of the 57th Annual Allerton Conference on Communication, Control, and Computing, Monticello, IL, USA, 24–27 September 2019; pp. 177–183.
25. O’Shea, T.J.; Hoydis, J. An introduction to deep learning for the physical layer. *IEEE Trans. Cogn. Commun. Netw.* **2017**, *3*, 563–575. [[CrossRef](#)]
26. Kim, H.; Jiang, Y.; Kannan, S.; Oh, S.; Viswanath, P. DeepCode: Feedback codes via deep learning. In Proceedings of the Advances in Neural Information Processing Systems 31 (NeurIPS), Montreal, QC, Canada, 3–8 December 2018; pp. 9436–9446.
27. Richardson, T.; Urbanke, R. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Trans. Inf. Theory* **2001**, *47*, 599–618.
28. Bengio, Y.; Leonard, N.; Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv* **2013**, arXiv:1308.3432. [[CrossRef](#)]
29. Ma, Z.; Lin, Y.; Hua, B.; Mao, K.; Zeng, L.; Lian, Z.; Zhu, Q.; Wu, Q. SIM-empowered LAINs: A unified channel model-driven optimization framework. *IEEE Wirel. Commun.* **2026**, early access. [[CrossRef](#)]

30. Ma, Y.; Ma, R.; Lin, Z.; Miao, C.; Zhang, R.; Long, W.; Wu, W.; Wang, J. Distributed split single-sideband time-modulated arrays for secure communications. *IEEE Internet Things J.* **2026**, *13*, 23862–23875. [[CrossRef](#)]
31. Hu, X.; Lyu, S.; Zhao, C.; Liu, C.; Peng, M. Distributed IRS beamforming for secure transmission. *IEEE Trans. Veh. Technol.* **2026**, *early access*. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.