A Low-Complexity Unified Error Correction Transformer

Yongli Yan
Tsinghua University
Beijing, China
yanyongli@tsinghua.edu.cn

Jieao Zhu, Tianyue Zheng, Zhuo Xu, Chao Jiang
Tsinghua University
Beijing, China
zja21, zhengty22, xz23, jiangc24@mails.tsinghua.edu.cn

Linglong Dai
Tsinghua University
Beijing, China
daill@tsinghua.edu.cn

ICC 2026 - IEEE International Conference on Communications | 979-8-3195-4209-0/26/$31.00 ©2026 IEEE | DOI: 10.1109/ICC59461.2026.11587038

***Abstract*—Channel coding is vital for reliable sixth-generation (6G) data transmission, employing diverse error correction codes for various application scenarios. Traditional decoders require dedicated hardware for each code, leading to high hardware costs. Recently, artificial intelligence (AI)-driven approaches, such as the error correction code Transformer and its enhanced version, the foundation error correction code Transformer, have been proposed to reduce the hardware cost by leveraging the Transformer to decode multiple codes. However, their excessively high computational complexity of $\mathcal{O}(N^2)$ due to the self-attention mechanism in the Transformer limits scalability, where N represents the sequence length. To reduce computational complexity, we propose a unified Transformer-based decoder that handles multiple linear block codes within a single framework. Specifically, a standardized unit is employed to align code length across different code types, while a redesigned low-rank unified attention module, with computational complexity of $\mathcal{O}(N)$, is shared across various heads in the Transformer. Extensive experimental results demonstrate that the proposed unified Transformer-based decoder outperforms existing methods and provides a high-performance, low-complexity solution for next-generation wireless communication systems.**

***Index Terms*—Channel coding, error correction, unified decoder, Transformer, LDPC, Polar, BCH, 6G.**

I. Introduction

With the continuous evolution of wireless communication technologies, the imminent arrival of sixth-generation (6G) wireless communication will mark a paradigm shift toward ultra-reliable communication [1], [2]. To meet the stringent performance demands of 6G, channel codes with strong error correction capabilities are expected to play a pivotal role. Specifically, low-density parity-check (LDPC) codes [3] and Polar codes [4] have proven effective in fifth-generation (5G) wireless communications. Due to their excellent performance, 3GPP has just made the decision to continue to use LDPC and Polar codes for future 6G in October 2025 [5]. Furthermore, algebraic geometry codes, such as Bose-Chaudhuri-Hocquenghem (BCH) codes [6], are commonly employed as outer codes in concatenated schemes with probabilistic codes to enhance decoding performance.

In wireless communication systems, multiple coding schemes are supported, such as LDPC and Polar codes in 5G [7], each requiring customized hardware for their decoding algorithms. For instance, LDPC codes employ iterative message passing for ***parallel*** data processing [8], while Polar codes rely on the successive cancellation (SC) decoding algorithm [4], which follows a ***sequential*** approach by traversing a decision tree. Both LDPC and Polar codes utilize ***soft*** decoding methods. In contrast, BCH codes employ algebraic decoding techniques classified as ***hard*** decoding [9], which rely on mathematical structures to correct errors. Due to fundamental differences in decoding algorithms, each code type requires customized hardware circuits, resulting in significant hardware resource cost.

To reduce hardware resource cost, AI-driven decoders have emerged as a promising alternative. For example, inspired by the success of Transformers in natural language processing [10], the error correction code Transformer (ECCT) was introduced to reduce hardware resource and enhance decoding performance for short codes [11]. ECCT leverages the Transformer's self-attention mechanism, where multiple attention heads operate in parallel to process sequential data and capture long-range dependencies, effectively modeling the constrained relationships among the log-likelihood ratios (LLRs) of decoder inputs. Building on ECCT, the foundation error correction code Transformer (FECCT) enhances generalization to unseen codewords by integrating the distance matrix, derived from the parity-check matrix $\mathbf{H}$, into the self-attention mechanism [12]. However, ECCT and FECCT rely on the Transformer's powerful self-attention mechanism, which has a computational complexity of $\mathcal{O}(N^2)$, where N represents the sequence length. Their high computational complexity makes them difficult to scale to long codes.

To reduce computational complexity, we propose a unified Transformer-based decoder capable of handling multiple linear block codes, including LDPC, Polar, and BCH, within a single framework. Specifically, the main contributions of this paper are summarized as follows.

- We introduce a unified Transformer-based decoder capable of seamlessly integrating various linear block codes. This architecture incorporates a standardized unit to align code lengths, and a unified attention module that compresses the structural information of all codewords. These components achieve a code-agnostic decoder, eliminate the need for distinct hardware circuits for different decoding algorithms, and reduce computational complexity.
- We redesign the self-attention module into a unified attention mechanism that shares attention scores across all

heads. By eliminating per-head projections of queries and keys, it reduces computational complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$, where N is the sequence length. This substantial reduction enables efficient training and inference even for long codes.

The rest of this paper is organized as follows. Section II provides background knowledge, including the vanilla Transformers, channel encoders. Section III details our proposed unified Transformer-based decoder. Section IV presents the results of training and inference. Section V analyzes the computational complexity the proposed decoder. Finally, Section VI concludes.

II. Background

For a better understanding of the proposed unified error correction code Transformer, this section will provide background knowledge on vanilla Transformers and forward error correction (FEC) codes.

A. Vanilla Transformers

The vanilla Transformer is a sequence-to-sequence model consisting of an encoder and decoder, each with L identical blocks [10]. Each block includes multi-head self-attention (MHA), a position-wise feed-forward network (FFN), layer normalization, and residual connections.

The input $\boldsymbol{x} \in \mathbb{R}^N$ is embedded into $\boldsymbol{X} \in \mathbb{R}^{N\times d_k}$ via a trainable weight matrix. The MHA computes correlations using

$$\begin{aligned}&\text{Attn}\left(\boldsymbol{W}^Q,\boldsymbol{W}^K,\boldsymbol{W}^V\right)=\\&\text{Softmax}\left(\frac{\left(\boldsymbol{X}\boldsymbol{W}^Q\right)\cdot\left(\boldsymbol{X}\boldsymbol{W}^K\right)^T}{\sqrt{d_k}}\right)\cdot\left(\boldsymbol{X}\boldsymbol{W}^V\right),\end{aligned}\tag{1}$$

where $\boldsymbol{W}^Q,\boldsymbol{W}^K,\boldsymbol{W}^V\in\mathbb{R}^{d_k\times d_k}$ are learnable projections.

The FFN applies position-wise transformations

$$\text{FFN}\left(\boldsymbol{X}\right)=\boldsymbol{W}_2\cdot\text{ReLU}\left(\boldsymbol{W}_1\cdot\boldsymbol{X}+\boldsymbol{b}_1\right)+\boldsymbol{b}_2,\tag{2}$$

with $\boldsymbol{W}_1\in\mathbb{R}^{d_k\times d_f}$, $\boldsymbol{W}_2\in\mathbb{R}^{d_f\times d_k}$, $\boldsymbol{b}_1\in\mathbb{R}^{d_f}$, and $\boldsymbol{b}_2\in\mathbb{R}^{d_k}$.

Finally, the output of a single encoder layer is

$$\begin{aligned}\boldsymbol{X}_1&=\text{Attn}\left(\text{LayerNorm}\left(\boldsymbol{X}\right)\right)+\boldsymbol{X}\\\boldsymbol{X}_o&=\text{FFN}\left(\text{LayerNorm}\left(\boldsymbol{X}_1\right)\right)+\boldsymbol{X}_1.\end{aligned}\tag{3}$$

B. Forward Error Correction Codes

Forward error correction reduces bit error rate over noisy channels using linear block codes defined by a generator matrix $\mathbf{G}\in\mathbb{F}_2^{k\times n}$ and parity-check matrix $\mathbf{H}\in\mathbb{F}_2^{(n-k)\times n}$, with $\mathbf{G}\cdot\mathbf{H}^T=\mathbf{0}$. A message $\boldsymbol{m}\in\{0,1\}^k$ is encoded as $\boldsymbol{x}=\boldsymbol{m}\cdot\mathbf{G}$, satisfying $\mathbf{H}\cdot\boldsymbol{x}=\mathbf{0}$.

The codeword is binary phase shift keying (BPSK)-modulated to yield $\boldsymbol{x}_s$ and transmitted over an additive white Gaussian noise (AWGN) channel, producing the received signal $\boldsymbol{y}=\boldsymbol{x}_s+\boldsymbol{n}$, where $\boldsymbol{n}\sim\mathcal{N}(0,\sigma^2)$. The decoder recovers $\boldsymbol{m}$ from $\boldsymbol{y}$.

Following [13], we employ an equivalent training model $\boldsymbol{y}=\boldsymbol{x}_s\cdot\boldsymbol{z}$ with multiplicative noise $\boldsymbol{z}$ and use only the all-zero codeword to prevent overfitting. The complete decoder framework is shown in Figure 1. Pre-processing concatenates the LLR magnitudes with the syndrome:

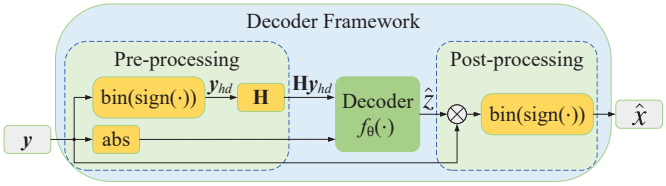

Fig. 1. Decoder framework with pre- and post-processing.

$$\widetilde{\boldsymbol{y}}=\left[|\boldsymbol{y}|,\text{s}\left(\boldsymbol{y}\right)\right].\tag{4}$$

Post-processing recovers the estimated codeword as:

$$\hat{\boldsymbol{x}}=\text{bin}\left(\text{sign}\left(\boldsymbol{y}\cdot\hat{\boldsymbol{z}}\right)\right).\tag{5}$$

Our work focuses on designing and training a parameterized decoding function f_θ.

III. Proposed Unified Error Correction Code Transformer

In this section, we present the detailed architecture and training process of the proposed **unified error correction code Transformer (UECCT)**.

A. Shared Unified Attention

The vanilla self-attention mechanism in Transformer architectures establishes dense correlations among all input features within a sequence, resulting in a computational complexity of $\mathcal{O}(N^2\cdot d_k+N\cdot d_k^2)$, where N is the sequence length and d_k is the embedding dimension. This approach, while effective for capturing long-range dependencies, is computationally intensive. Moreover, prior studies suggest that the input to the softmax function in self-attention is probabilistically sparse, with only a few weights dominating the attention distribution [14]. This sparsity implies that a low-rank approximation could effectively capture the dominant correlations, reducing redundancy without compromising performance.

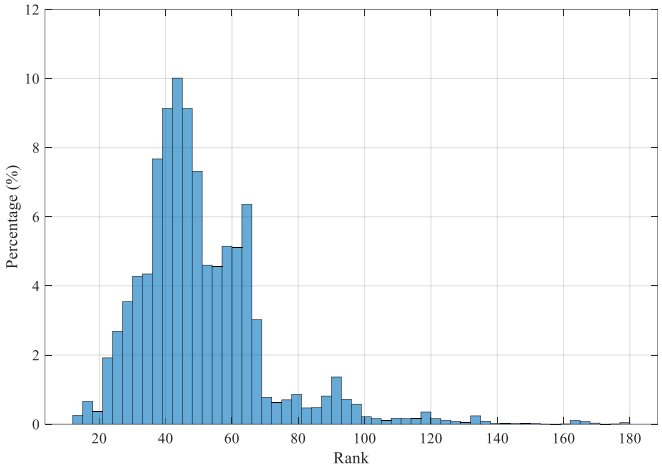

Fig. 2. Rank distribution of the $\boldsymbol{Q}\cdot\boldsymbol{K}^T$ matrix in the ECCT model.

To investigate this hypothesis, we trained an ECCT model [11] with parameters $L = 6$ (number of Transformer encoder layers), $H = 8$ (number of attention heads), $d_k = \max(2n - k) = 182$, and $d_f = 2912$ on a diverse set of linear block codes, including LDPC(49,24), LDPC(121,60), LDPC(121,80), Polar(32,16), Polar(64,32), Polar(128,86), BCH(31,16), BCH(63,36), and BCH(127,120). We analyzed the rank distribution of the $\boldsymbol{Q} \cdot \boldsymbol{K}^T$ matrix across 1000 batches for the six-layer Transformer encoder. As illustrated in Figure 2, the rank ranges from 12 to 181, with over 90% of values below 69, consistently indicating a low-rank structure across all layers and code types. This observation suggests that the high-dimensional self-attention matrix can be approximated with a lower-rank representation, offering a pathway to reduce computational overhead.

Fig. 3. Distribution of Jensen-Shannon divergence between attention heads.

Furthermore, the conventional multi-head self-attention mechanism computes independent attention scores for each head using distinct weight matrices $\boldsymbol{W}^Q$, $\boldsymbol{W}^K$, and $\boldsymbol{W}^V$. However, we hypothesize that the attention distributions across heads may exhibit high similarity, particularly in channel decoding, where the sparse structure of the parity-check matrix $\mathbf{H}$ constrains bit interactions. To measure attention head similarity, we employed the Jensen-Shannon Divergence (JSD) [15], a symmetric measure of similarity between probability distributions, defined as:

$$\begin{aligned} &\mathrm{JSD}(\mathbf{A}_i[k] \| \mathbf{A}_j[k]) = \\ &\frac{1}{2}\left[D_{KL}(\mathbf{A}_i[k] \| \mathbf{M}[k]) + D_{KL}(\mathbf{A}_j[k] \| \mathbf{M}[k])\right], \end{aligned} \tag{6}$$

where k is the row index, $\mathbf{A}_i$ and $\mathbf{A}_j \in \mathbb{R}^{N \times N}$ are the softmax outputs of attention heads i and j, $\mathbf{M}[k] = \frac{1}{2}(\mathbf{A}_i[k] + \mathbf{A}_j[k])$ is the average distribution, and $D_{KL}(\mathbf{P} \| \mathbf{Q}) = \sum_{l=1}^{N} \mathbf{P}[l] \log \frac{\mathbf{P}[l]}{\mathbf{Q}[l]}$ is the Kullback-Leibler divergence [16]. Using the same simulation setup as the low-rank validation ($L = 6$, $H = 8$, $d_k = 182$, $d_f = 2912$), we computed the per-row JSD for all head pairs and averaged the results across layers. As shown in Figure 3, JSD values are consistently below 0.16, indicating strong alignment in attention distributions.

Motivated by these findings, we propose a novel **unified attention module (UAM)** to address the computational inefficiency and lack of cross-code generalization in traditional self-attention. The UAM introduces learnable memory components, $\boldsymbol{A}_l$ and $\boldsymbol{V}_l \in \mathbb{R}^{N \times d_l}$, where d_l is a hyperparameter controlling the low-rank approximation. The UAM is defined as:

$$\mathrm{UniAttn}\left(\boldsymbol{A}_l, \boldsymbol{V}_l\right) = \mathrm{Softmax}\left(\boldsymbol{A}_l\right) \cdot \left(\boldsymbol{V}_l^T \boldsymbol{X}\right), \tag{7}$$

where $\boldsymbol{X} \in \mathbb{R}^{N \times d_k}$ is the input to the attention layer.

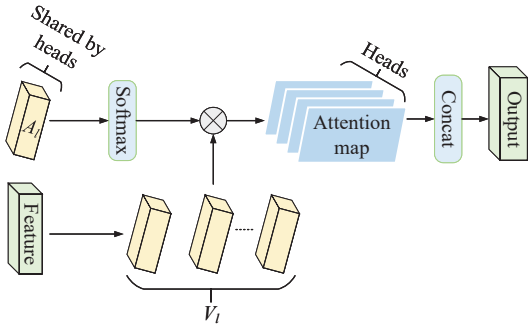

Fig. 4. Architecture of the proposed multi-head unified-attention.

The architecture of the proposed multi-head unified-attention is depicted in Figure 4. By eliminating the projections $\boldsymbol{W}^Q$ and $\boldsymbol{W}^K$, the UAM reduces the computational complexity from $\mathcal{O}(N^2 \cdot d_k + N \cdot d_k^2)$ to $\mathcal{O}(N \cdot d_l \cdot d_k)$, where $d_l \ll N$. The **shared attention matrix** $\mathbf{A}_l$ is applied across all heads, leveraging the observed head similarity to promote a unified decoding architecture.

B. Sparse Masked Unified Attention

To further exploit the sparsity of linear block codes, we propose a sparse masked unified attention mechanism that directly incorporates the parity-check constraints into the shared memory $\boldsymbol{A}_l$.

Algorithm 1 Pseudocode for Sparse Mask Construction

```
Input: H    # shape = (n − k, n)
H̄ = zeros(2n − k, n − k)
H̄[0 : n, 0 : n − k] = H.transpose(0, 1)
H̄[n : 2n − k, 0 : n − k] = eye(n − k)
Output: (−∞) · (¬ H̄)
```

For a code with parity-check matrix $\mathbf{H} \in \mathbb{F}_2^{(n-k) \times n}$, we construct an extended matrix

$$\overline{\mathbf{H}} = \begin{bmatrix} \mathbf{H}^T \\ \mathbf{I}_{n-k} \end{bmatrix} \in \mathbb{R}^{(2n-k) \times (n-k)} \tag{8}$$

and generate a sparse mask $\mathrm{M}(\overline{\mathbf{H}}) \in \{-\infty, 0\}^{N \times d_l}$ ($N = 2n - k$, $d_l = n - k$) via Algorithm 1. The masked attention is then computed as

$$\mathrm{UniAttn}(\boldsymbol{A}_l, \boldsymbol{V}_l) = \mathrm{Softmax}\big(\boldsymbol{A}_l + \mathrm{M}(\overline{\mathbf{H}})\big) \cdot (\boldsymbol{V}_l^T \boldsymbol{X}). \tag{9}$$

This sparse mask enforces attention only on positions corresponding to parity-check connections and syndrome bits, thereby eliminating redundant computations compared to dense self-attention.

C. Architecture and Training Methodology

The architecture of the proposed unified error correction code Transformer is illustrated in Figure 5. The initial layer compresses the multi-head output into a one-dimensional vector of size $2n-k$, with the subsequent layer transforming it into an n-dimensional vector representing the softly decoded noise. Finally, the post-processing module acts on the estimated noise to derive the transmitted original codeword.

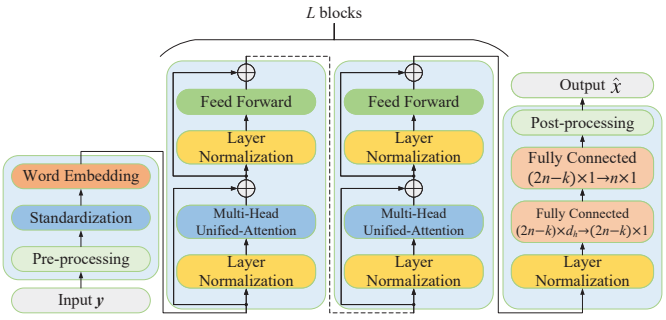

Fig. 5. Illustration of the proposed unified error correction code Transformer.

We employ binary cross-entropy as the loss function with the objective of learning to predict the multiplicative noise $\boldsymbol{z}$. The corresponding target binary multiplicative noise is denoted as $\tilde{\boldsymbol{z}} = \text{bin}(\text{sign}(\boldsymbol{z}))$. Therefore, the loss calculation for a received individual codeword $\boldsymbol{y}$ is given by:

$$\text{loss} = -\sum_{i=1}^{n} \tilde{z}_i \log(f_\theta(y)) + (1-\tilde{z}_i)\log(1-f_\theta(y)). \quad (10)$$

To accommodate codewords of varying lengths within a unified architecture, we have designed a standardization unit that performs padding on $|\boldsymbol{y}|$ and $s(\boldsymbol{y})$ as described in Equation (4). Assuming that the maximum length of the codeword is $N_{\max}$ and the maximum length of the syndrome is $S_{\max}$, the dataset is standardized according to:

$$\widetilde{\boldsymbol{y}} = [\ [|\boldsymbol{y}|,\ \mathbf{0}_c]\,,\ [s(\boldsymbol{y}),\ \mathbf{0}_s]\ ]\,, \quad (11)$$

where $\mathbf{0}_c$ and $\mathbf{0}_s$ represent zero vectors with sizes $N_{\max} - \text{size}(\boldsymbol{y})$ and $S_{\max} - \text{size}(s(\boldsymbol{y}))$, respectively. Through this unit, the Transformer neural decoder can accommodate any codeword within the maximum length, allowing for joint training in a unified manner.

In the training phase, we conducted 1000 epochs, each consisting of 1000 minibatches, which in turn contained 512 samples each. Samples for each minibatch were randomly selected from a pool of LDPC, Polar, and BCH codewords. The learning rate was initialized at 10^{-3}, and a cosine decay scheduler was applied without warmup [17], gradually reducing it to 10^{-6} by the end of the training process. We employ the Adam optimizer to dynamically adjust the learning rate, leveraging its efficacy in managing diverse data types and model architectures [18]. The AWGN noise introduced to the links had its signal-to-noise ratio (SNR) randomly chosen between 3 and 7 for each batch. All experiments were conducted on NVIDIA GeForce RTX 4090 24GB GPUs.

IV. Experiments

We conducted experiments on linear block codes, specifically LDPC, Polar, and BCH, to assess the effectiveness of the proposed architecture. The parity-check matrices utilized in the paper were sourced from [19]. The results are presented in the form of bit error rate (BER) and block error rate (BLER) at various normalized SNR (i.e. $\text{E}_\text{b}/\text{N}_0$) values, with at least 10^5 random codewords tested per SNR. We define aggregate BER (ABER) and aggregate BLER (ABLER) as the overall error rates spanning all evaluated codewords.

In addition, to enable direct comparison with ECCT and FECCT, whose simulation results were sourced from their respective papers [11], [12], we fine-tuned the pre-trained UECCT models on consultative committee for space data systems (CCSDS) [20] and MacKay [21] codewords using jointly trained LDPC, Polar, and BCH models. Given GPU memory constraints and the fact that the performance of long codes has been thoroughly explored [22], while the error correction of short codes remains far from the Shannon limit [23], this work primarily focuses on training and validating ***medium-to-short codes***.

TABLE I
NEGATIVE NATURAL LOGARITHM OF BER FOR ECCT, FECCT, AND UECCT ACROSS POLAR, LDPC, BCH, CCSDS, AND MACKAY CODES AT $\text{E}_\text{b}/\text{N}_0$ = 4, 5, 6 DB, WITH BEST RESULTS IN BOLD (HIGHER IS BETTER).

Method	ECCT			FECCT			Ours		
	$L=6, H=8$ $d_k=64, d_f=2048$			$L=6, H=8$ $d_k=128, d_f=4096$			$L=6, H=8$ $d_k=64, d_f=2048$		
$\text{E}_\text{b}/\text{N}_0$ [dB]	4	5	6	4	5	6	4	5	6
Polar (32, 16)	–	–	–	**6.36**	**8.36**	**11.49**	6.32	8.07	10.37
Polar (64, 32)	6.48	8.60	11.43	5.88	7.91	10.76	**6.58**	**8.91**	**11.72**
Polar (64, 48)	6.15	8.20	10.86	6.06	8.21	10.90	**6.21**	**8.31**	**10.96**
Polar (128, 64)	5.12	7.36	10.48	–	–	–	**6.50**	**9.23**	**12.46**
Polar (128, 86)	5.75	8.16	11.29	5.53	7.90	11.29	**6.60**	**9.43**	**12.84**
Polar (128, 96)	5.88	8.33	11.49	–	–	–	**6.36**	**9.09**	**12.39**
LDPC (49, 24)	**5.91**	**8.42**	**11.90**	–	–	–	5.70	8.18	11.40
LDPC (121, 60)	**5.02**	7.94	**12.72**	–	–	–	5.00	**7.98**	11.98
LDPC (121, 70)	**6.28**	**10.12**	**15.57**	–	–	–	6.21	9.74	14.16
LDPC (121, 80)	**7.17**	**11.21**	**16.31**	–	–	–	7.04	11.12	15.21
BCH (31, 16)	**5.85**	7.52	10.08	–	–	–	5.79	**7.64**	**10.34**
BCH (63, 36)	4.62	6.24	8.44	4.53	6.38	9.10	**4.83**	**6.76**	**9.75**
BCH (63, 45)	5.41	7.49	10.25	5.18	7.32	10.31	**5.42**	**7.63**	**10.86**
BCH (63, 51)	5.46	7.57	10.51	**5.71**	**8.07**	**11.31**	5.68	8.00	11.12
BCH (127, 92)	–	–	–	4.11	5.84	8.79	**4.23**	**6.00**	**8.82**
BCH (127, 120)	–	–	–	4.62	6.33	8.95	**4.70**	**6.41**	**9.06**
CCSDS (32, 16)	–	–	–	5.23	7.00	9.21	**5.29**	**7.09**	**9.39**
CCSDS (128, 64)	**6.65**	**10.40**	**15.46**	6.52	9.67	15.01	6.45	10.10	13.28
MacKay (96, 48)	**7.10**	**10.12**	**14.21**	–	–	–	6.68	9.57	12.91

We present the performance of our framework with hyperparameters are set to $L = 6$, $H = 8$, $d_k = 64$, $d_l = 64$, and $d_f = 4 \cdot H \cdot d_k$. The value of $d_l = 64$ corresponds to the maximum syndrome length among the jointly trained

codewords. Table I presents the simulation results, specifically showing the negative natural logarithm of the BER at $\mathrm{E_b/N_0}$ values of 4, 5, and 6 dB. It is noted that higher values in this context indicate better performance, with the best results for each SNR highlighted in bold. Simulation results demonstrate that our proposed UECCT outperforms both the individually trained ECCT [11] and the jointly trained FECCT [12] across the majority of codewords, validating the effectiveness of the proposed architecture. Notably, UECCT even surpasses FECCT's performance with an embedding length of 128 when operating at a more compact embedding length of 64, further highlighting the computational efficiency of our design.

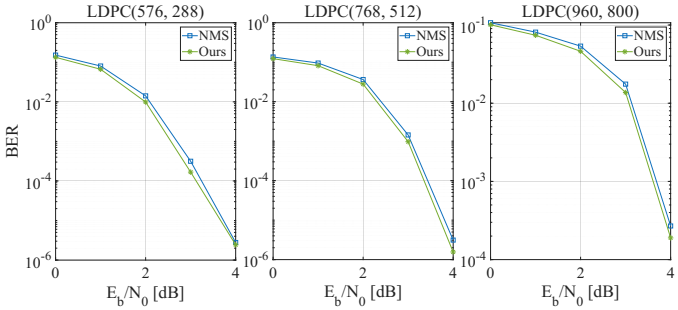

Fig. 6. BER comparison between the NMS algorithm and the proposed UECCT for medium-to-long LDPC codes.

To further validate the performance and scalability of the proposed method on medium-to-long codes, we conducted joint training using LDPC(576, 288), LDPC(768, 512), and LDPC(960, 800) codewords with hyperparameters set to $L = 12$, $H = 8$, $d_k = 16$, $d_l = 288$, and $d_f = 512$. The neural network training and inference were performed under an AWGN channel with BPSK modulation. During training, the SNR was randomly selected between 2 dB and 3 dB, with a batch size of 192, while other training methods were consistent with those described in Section III-C. During inference, 10^5 code blocks were tested per SNR. For comparison with traditional decoding algorithms, we utilized the normalized min-sum (NMS) algorithm for LDPC codes [24]. The NMS algorithm was configured with a normalization factor of 0.875 and a maximum of 15 iterations. Additionally, an early stopping mechanism was implemented to halt decoding once the parity-check matrix $\mathbf{H}$ is satisfied. Figure 6 presents the BER performance comparison between the NMS algorithm and the proposed UECCT. The simulation results demonstrate that the proposed method achieves performance comparable to the traditional NMS decoding algorithm. This indicates that our decoding architecture effectively scales to medium-to-long codes.

V. Computational Complexity Analysis

Before delving into the computational complexity, let's first review the unified error correction code Transformer presented in Figure 5, which incorporates L layers of Transformer encoders. The principal computational complexity within each layer is found in the multi-head self-attention module, which is predominantly responsible for matrix multiplication and is defined by the hyperparameters H, d_k and d_l. Consequently, the computational complexity of the proposed architecture is given by $\mathcal{O}\left(L \times H \times (N \times d_l \times d_k \times H_d)\right)$, where $N = 2n - k$, and $d_l \ll N$. Thanks to the sparsity of the parity-check matrix, H_d is usually quite small. In our experiment, H_d is approximately 0.14, which significantly reduces the computational complexity of the neural network. By contrast, the vanilla Transformer architectures in ECCT and FECCT have complexities of $\mathcal{O}\left(L \times H \times \left(N^2 \times d_k \times M_d + N \times d_k^2\right)\right)$ and $\mathcal{O}\left(L \times H \times \left(N^2 \times d_k + N \times d_k^2 + N^2 \times M_d\right)\right)$, respectively, where M_d represents the mask density in these models.

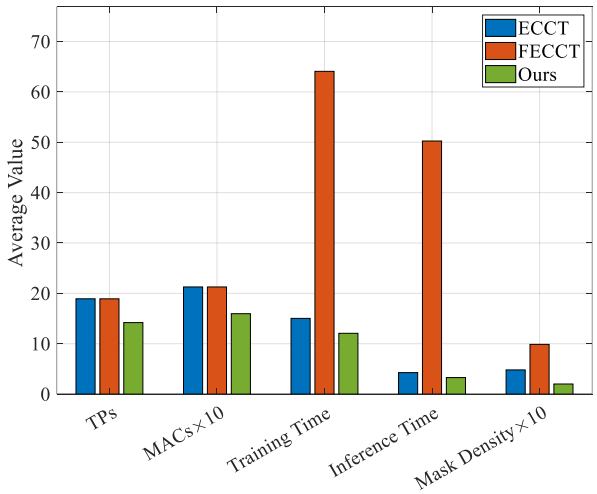

Fig. 7. Comparison of average TPs, MACs, training time, inference time, and mask density between ECCT, FECCT and UECCT.

Table II compares the trainable parameters (TPs), multiply-accumulate operations (MACs), training time, inference time, and mask density for ECCT, FECCT, and UECCT. Lower values are preferable, with the best performing values highlighted in bold. The metrics for TPs and MACs are obtained using Thop [25], a neural network profiling tool. UECCT reduces parameters and MACs by 25.0% compared to both ECCT and FECCT through low-rank approximation, shared attention, and omission of $\boldsymbol{Q}$ and $\boldsymbol{K}$ projections. This results in training times 19.7% and 81.2% lower, and inference times 23.1% and 93.5% lower than ECCT and FECCT, respectively. Mask densities are influenced by design choices: FECCT has the highest density due to Tanner graph distance calculations, ECCT has a moderate density from extended bit-pair relations, and UECCT has the lowest density, 58.2% and 79.7% lower than ECCT and FECCT, achieved through the direct utilization of $\mathbf{H}$ and the extension of the identity matrix. For a more intuitive representation, the average values of each metric across the codewords in Table II are provided in Figure 7.

VI. Conclusions

We proposed a unified Transformer-based decoder that seamlessly integrates multiple linear block codes within a single framework. By introducing standardized unit and a unified

TABLE II

COMPARISON OF TRAINABLE PARAMETERS, MACS, TRAINING TIME, INFERENCE TIME, AND MASK DENSITY AMONG ECCT, FECCT, AND UECCT FOR VARIOUS CODES, ALL WITH IDENTICAL PARAMETERS $L = 6$, $H = 8$, $d_k = 64$, $d_l = 64$, AND $d_f = 2048$.

Codes	Trainable Parameters (M)			MACs Per Codeword (G)			Training Time Per 256 Codewords (ms)			Inference Time Per 256 Codewords (ms)			Mask Density		
	ECCT	FECCT	Ours	ECCT	FECCT	Ours	ECCT	FECCT	Ours	ECCT	FECCT	Ours	ECCT	FECCT	Ours
Polar (32,16)	18.917	18.915	**14.189**	0.907	0.907	**0.681**	15.089	30.756	**11.972**	3.922	18.699	**3.270**	0.615	0.979	**0.266**
Polar (64,32)	18.919	18.915	**14.191**	1.815	1.814	**1.362**	14.605	52.738	**11.667**	3.825	39.965	**3.132**	0.573	0.979	**0.198**
Polar (128,86)	18.931	18.915	**14.203**	3.213	3.213	**2.411**	15.693	95.680	**12.327**	4.442	79.946	**3.663**	0.668	0.990	**0.208**
LDPC (49,24)	18.918	18.915	**14.190**	1.455	1.455	**1.092**	14.641	44.202	**11.538**	3.958	30.977	**3.026**	0.277	0.994	**0.104**
LDPC (121,60)	18.927	18.915	**14.199**	3.535	3.534	**2.652**	15.456	104.541	**13.009**	4.652	88.661	**3.547**	0.254	0.987	**0.064**
LDPC (121,80)	18.929	18.915	**14.201**	3.119	3.119	**2.340**	15.222	92.742	**12.636**	4.827	77.364	**3.376**	0.219	0.995	**0.073**
BCH (31,16)	18.917	18.915	**14.189**	0.869	0.869	**0.652**	14.396	30.733	**11.478**	4.020	18.159	**3.076**	0.390	0.994	**0.196**
BCH (63,36)	18.919	18.915	**14.191**	1.701	1.701	**1.276**	14.413	50.945	**11.583**	4.043	37.555	**3.022**	0.485	0.978	**0.211**
BCH (127,120)	18.932	18.915	**14.204**	2.533	2.533	**1.901**	15.786	74.412	**12.426**	4.704	60.915	**3.419**	0.841	0.989	**0.485**

attention module, we achieved enhanced decoding accuracy and reduced computational complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. This work delivers a high-performance, low-complexity solution, addressing hardware overhead and scalability challenges in next-generation wireless systems like 6G. Future work will focus on improving memory efficiency and power consumption through techniques such as pruning and quantization to enhance deployability in resource-constrained environments.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China (Youth Program) under Grant 62401332, in part by the Postdoctoral Fellowship Program of CPSF under Grant Number GZC20231377, and in part by the National Natural Science Foundation of China (Distinguished Young Scholars Program) under Grant 62325106.

REFERENCES

[1] E. Björnson, L. Sanguinetti, H. Wymeersch, J. Hoydis, and T. L. Marzetta, "Massive MIMO is a reality—What is next? Five promising research directions for antenna arrays," *Digit. Signal Process.*, vol. 94, pp. 3–20, Nov. 2019.

[2] H. Zhang and W. Tong, "Channel coding for 6G extreme connectivity—requirements, capabilities, and fundamental tradeoffs," *IEEE BITS Inf. Theory Mag.*, vol. 3, no. 1, pp. 1–12, Mar. 2024.

[3] R. Gallager, "Low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[4] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, Jul. 2009.

[5] 3GPP, "3gpp TSG RAN WG1 meeting #122-bis," 3rd Generation Partnership Project (3GPP), Technical Report, Oct. 2025.

[6] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Inf. Control*, vol. 3, no. 1, pp. 68–79, Mar. 1960.

[7] 3GPP, "Technical specification group radio access network; multiplexing and channel coding," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.212, Sep. 2023, 3GPP TS 38.212, V18.0.0.

[8] M. P. C. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. Commun.*, vol. 47, no. 5, pp. 673–680, May 1999.

[9] P. Mathew, L. Augustine, G. Sabarinath, and T. Devis, "Hardware implementation of (63,51) BCH encoder and decoder for WBAN using LFSR and BMA," *arXiv:1408.2908*, Aug. 2014.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS'17)*, vol. 30, Long Beach, CA, USA, Dec. 2017.

[11] Y. Choukroun and L. Wolf, "Error correction code transformer," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS'22)*, vol. 35, New Orleans, LA, USA, Nov. 2022, pp. 38 695–38 705.

[12] ——, "A foundation model for error correction codes," in *Proc. 12th Int. Conf. Learn. Represent. (ICLR'24)*, Vienna, Austria, Apr. 2024.

[13] A. Bennatan, Y. Choukroun, and P. Kisilev, "Deep learning for decoding of linear codes - a syndrome-based approach," in *Proc. 2018 IEEE Int. Symp. Inf. Theory (ISIT'18)*, Vail, CO, USA, Jun. 2018, pp. 1595–1599.

[14] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 12, pp. 11 106–11 115, May 2021.

[15] M. L. Menéndez, J. A. Pardo, L. Pardo, and M. d. C. Pardo, "The Jensen-Shannon divergence," *J. Franklin Inst.*, vol. 334, no. 2, pp. 307–318, Mar. 1997.

[16] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Stat.*, vol. 22, no. 1, pp. 79–86, Mar. 1951.

[17] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T. Liu, "On layer normalization in the transformer architecture," in *Proc. 37th Int. Conf. Mach. Learn. (ICML'20)*, vol. 119, Virtual Event, Jul. 2020, pp. 10 524–10 533.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, Jan. 2017.

[19] M. Helmling, S. Scholl, F. Gensheimer, T. Dietz, K. Kraft, S. Ruzika, and N. Wehn, "Database of channel codes and ML simulation results," 2019. [Online]. Available: https://www.uni-kl.de/channel-codes

[20] CCSDS, "Recommendation for space data system standards," CCSDS, Technical Report, 2003, cCSDS 131.0-B-1, Blue Book.

[21] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.

[22] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[23] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, Jul. 1948.

[24] J. Chen and M. P. C. Fossorier, "Decoding low-density parity check codes with normalized APP-based algorithm," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM'01)*, vol. 2, San Antonio, TX, USA, Nov. 2001, pp. 1026–1030.

[25] L. Zhu, "THOP: A tool for measuring the FLOPs of neural networks," 2020. [Online]. Available: https://github.com/Lyken17/pytorch-OpCounter