Transformer-based Convolutional Codes Decoder

Guangya Hu , **Tianyi Li** and **Chaoping Xing***
Shanghai Jiao Tong University, Shanghai, China
{hgyhang, ltetsla, xingcp}@sjtu.edu.cn,

Abstract

As a fundamental component of the physical layer, convolutional codes are instrumental in ensuring dependable transmission of information over inherently noisy communication channels. Recently, neural network-based decoding algorithms have shown remarkable progress in enhancing the performance of block error-correcting codes. However, current neural network decoders for convolutional codes often exhibit strong performance primarily on low-rate codewords, whereas maximum likelihood decoding algorithm incur substantial time overhead when decoding high-rate codewords, which impacts communication efficiency in practical communication systems. In this work, we propose a Transformer-based decoder for convolutional codes designed for soft-decision decoding over AWGN and fading channels, which serves as an efficient decoding scheme for convolutional codes with various code rates. For the unique structure of convolutional codes, we propose a Bidirectional Context Aggregation mechanism to enable the model to fully capture the long-range dependencies and the memory effect of convolutional structures. Additionally, we extend the Transformer attention matrix with a codeword-specific feature integration mechanism to allow the model to capture implicit relationships within codewords. Extensive experiments demonstrate that the accuracy performance of the proposed model outperforms that of neural network decoders. And for maximum likelihood decoding algorithm, our model achieves comparable error-correction capability and a significantly faster decoding speed.

1 Introduction

In the modern information age, ensuring strong and reliable digital transmission amidst channel interference is a top priority within the realm of communication systems. Since the birth of information theory [Shannon, 1948], much progress has been made in the design of near-optimal codes. Among these, convolutional codes [Elias, 1955] are a type of linear code capable of encoding information streams of indefinite length. As an important channel coding technology, convolutional codes have been widely used in communication systems such as GSM [ETSI Secretariat, 1996], WCDMA [3GPP, 2000], and Wi-Fi protocols [IEEE, 2021].

In recent years, deep learning methods have developed rapidly and have been applied to all layers of communication systems to improve performance greatly. Decoding of channel codes can be considered as a classification problem, where the aim is to infer the transmitted codeword from its noisy version. The recent success of deep learning methods for a wide range of classification problems and their ability to capture complex patterns has motivated exploring their applications to channel decoding. Machine learning-based decoding primarily employs two methodologies: model-based [Xu *et al.*, 2017; Doan *et al.*, 2019; Buchberger *et al.*, 2021] and model-free. The first one involves using neural networks to update relevant weights within traditional decoding algorithms such as belief propagation (BP), while the other entails entirely novel decoding methods that don't rely on existing algorithms. Neural BP (NBP) introduces learnable weights to conventional BP graphs and has gained improvements in comparison to the baseline BP method for short codes [Nachmani *et al.*, 2016; Nachmani *et al.*, 2018; Lugosch and Gross, 2017]. Neural successive cancellation for polar codes [Doan *et al.*, 2018] and Neural Viterbi (NViterbi) [Kim *et al.*, 2018] for convolutional codes perform close to the optimal. While model-based decoders benefit from a strong theoretical background, the architecture is overly restrictive, and the improvement gain generally vanishes for more iterations [Hoydis *et al.*, 2022]. Model-free decoders employ various types of neural network architectures. [Cammerer *et al.*, 2017; Gruber *et al.*, 2017] used fully connected (FC) networks; [Jiang *et al.*, 2019] employed Convolutional Neural Network for turbo codes. [Bennatan *et al.*, 2018] extended the classical syndrome decoding by adding the magnitude to the syndrome vector. [Choukroun and Wolf, 2022] implemented a Transformer-based [Vaswani *et al.*, 2017] decoding scheme (ECCT) for error-correcting codes, employing the previously discussed syndrome decoding technique, and yielded favorable performance. [Choukroun and Wolf, 2023] extended the denoising diffusion [Ho *et al.*, 2020] paradigm to an error

*Corresponding author.

correction code. [Choukroun and Wolf, 2024a] 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. [Choukroun and Wolf, 2024b] introduced a unified encoder-decoder training framework for linear block, enabling end-to-end differentiable optimization over Galois fields for code design. [Kurmukova and Gunduz, 2024] presented a "friendly attack" approach that parallels traditional adversarial attacks. It enhances the performance of error-correcting channel code decoders by applying subtle perturbations to fixed-point modulated codewords before transmission. [Park *et al.*, 2025a] used multiple masks to enhance the capability of transformer for error correction codes.[Park *et al.*, 2025b] proposed CrossMPT which ingeniously refines the ECCT model to achieve superior bit error rate (BER) performance while maintaining lower computational and memory overhead. [Xiao *et al.*, 2025] exploits a cyclic shift reuse property in Transformer-based decoders for cyclic codes, significantly reducing the number of model parameters.

Nonetheless, the aforementioned impactful studies were not explicitly developed for sequential codes (e.g., convolutional codes). Existing neural architectures specialized for this code type currently exhibit suboptimal performance, thereby limiting their efficacy in real-world scenarios. [Kim *et al.*, 2018] proposed a bi-direction Gated Recurrent Units (bi-GRU)-based approach, designed to emulate the Viterbi [Viterbi, 1967] and BCJR [Bahl *et al.*, 1974] algorithms, which is limited to specific codewords with a code rate of 1/2. Consequently, their model shows only mediocre performance on convolutional codes commonly used in the Wi-Fi (IEEE 802.11) standard. [Teich *et al.*, 2019] proposed an energy-efficient analog decoder implementation for the case of convolutional self-orthogonal codes (CSOCs) based on high-order recurrent neural networks (HORNNs). [Teich and Pan, 2023] extends the code type to nonsystematic convolutional codes by a network based on a Convolutional Neural Network. Unfortunately, their decoding performance and the upscaling property degrade severely for general convolution codes. [Yan and Dai, 2025] utilized a Long Short-Term Memory (LSTM)-based [Graves, 2012] model for decoding punctured convolutional codes of various rates, achieving promising results.

1.1 Our Contribution

In this work, we propose a convolutional codes decoder characterized by high accuracy, fast decoding speed, and applicability across various convolutional codes. Inspired by the excellent performance of a series of works like Error Correction Codes Transformer (ECCT) [Choukroun and Wolf, 2022] on linear block codes, we select the transformer architecture as our decoder. To enhance the model's adaptability and performance, we propose the following technical contributions specifically addressing the characteristics of convolutional codes: (i) we adapt the Transformer input sequence by Bidirectional Context Aggregation to more effectively leverage information from preceding and succeeding codeword blocks. (ii) We extract implicit inter-bit information to refine the attention matrix, which results in a much lower bit error rate. Applied to a variety of convolutional codes, our approach outperforms existing neural decoders across diverse convolutional codes, attaining near-maximum-likelihood error correction performance with substantially lower computational latency than the Viterbi algorithm.

2 Background

To furnish a comprehensive understanding of the proposed Transformer-based convolutional code Decoder, this section presents the necessary background on convolutional codes, the transmission scheme, and the Transformer architecture.

2.1 Convolutional Codes

Convolutional codes are linear codes that assign code bits to an incoming information bit stream continuously in a stream-oriented fashion. The convolutional code is characterized by its encoder structure, where the input bit stream passes through a series of shift registers, generating encoded bits as output. Each output bit is determined by the current input and several previous input bits, as defined by the constraint length of the code. The encoder structure can be represented as a polynomial matrix. For example, Fig. 1 shows the encoder structure of the convolutional code specified in the IEEE 802.11 standard, and its corresponding polynomial matrix is $[91, 121]$, where the binary representation of each decimal digit's value indicates whether the corresponding register participates in the output of the result.

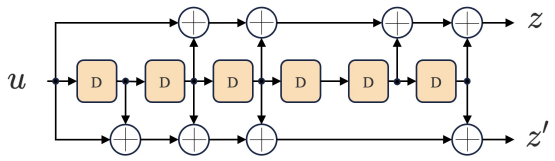

Figure 1: Encoder structure of (2,1,7) convolutional code in IEEE 802.11.

2.2 Transmission Scheme

Fig. 2 shows the transmission model. For encoding and decoding, we assume a traditional transmission setting with a (n, k, l) convolutional code $\mathcal{C}$ over a finite field $\mathbb{F}_2 = \{0, 1\}$. The sender has a K-bit message $m \in \{0, 1\}^K$, which is encoded into a codeword X^N from the code space. The modulator then maps the sequence of codeword bits into a sequence of symbols S^N. In this work, we focus on the binary phase shift keying (BPSK) modulation where $S^N = 1^N - 2X^N$. The mapped symbols are sent over a discrete-time additive white Gaussian noise (AWGN) channel, i.e., $Y^N = S^N + Z^N$, where Z^N is a sequence of independent and identically distributed Gaussian random variables $Z_i \backsim \mathcal{N}(0, \sigma^2)$. The sequence of received symbols Y^N is the input to the decoder after preprocessing. The main goal of the decoder $F : \mathbb{R}^n \to \mathbb{R}^n$ is to provide a soft approximation $\hat{X^N} = f(Y^N)$ of the codeword. In order to avoid overfitting, we extend classical syndrome decoding and employ a surrogate decoder [Bennatan *et al.*, 2018; Choukroun

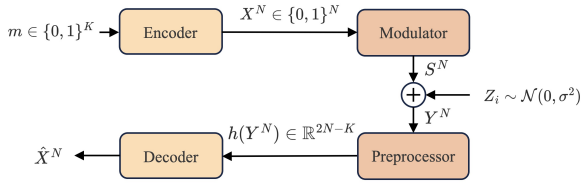

Figure 2: The communication scenario with AWGN channel.

and Wolf, 2022]. The surrogate decoder objective is defined by the prediction of the equivalent multiplicative noise $\tilde{\epsilon}$ such that $Y^N = X^N \odot \tilde{\epsilon}$, where $\odot$ is the Hadamard product. The decoder preprocesses the channel output Y^N by concatenating the provably codeword-independent magnitude and syndrome vectors, such that $h(Y^N) = [|Y^N|, s(Y^N)] \in \mathbb{R}^{2N-K}$, where $[\cdot,\cdot]$ denotes vector concatenation and $s(Y^N)$ denotes the syndrome defined by $s(Y^N) = H\text{bin}(Y^N)$ with $\text{bin}(Y^N)$ the binary mapping of Y^N. In this case, the soft codeword prediction is given by the denoising task defined as $\hat{X^N} = f(h(Y^N)) \odot Y^N$.

2.3 Transformer

The Transformer architecture [Vaswani *et al.*, 2017], which was initially proposed for machine translation, introduced a novel attention-based approach. It processes the input sequence by incorporating positional embeddings for each element and embedding it into a high-dimensional space.

The embeddings are propagated through multiple normalized self-attention and feedforward layers. 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 $\text{Attention}(Q,K,V) = \text{Softmax}(d^{-1/2}(QK^T))V$. Here $Q \in \mathbb{R}^{N\times d}, K \in \mathbb{R}^{\kappa\times d}$ and $V \in \mathbb{R}^{\kappa\times d}$ represent the stacked N queries, κ 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.

3 Convolutional Code Decoding Transformer

In this section, we introduce the crucial components of the proposed framework, the complete architecture, and the training procedure.

3.1 Bidirectional Context Aggregation

Different from linear block codes, convolutional codes handle infinite information bit streams and encode them into sequences, which means that the entire codeword cannot be fed into the decoder all at once. It is unreliable to group the continuous code sequences into different blocks directly; otherwise, the constrained information between blocks will be lost. In [Zhang *et al.*, 2020], the authors provide a decoding method that can utilize the constraint information from the previous codeword block. They train multiple different networks according to the length of the constraint bits, i.e., the number of memory register states. Based on the decoding result of the previous codeword block, they select a different network to decode the next codeword block. However, the unique structure of this method limits the parallel processing capability of the decoder. Furthermore, if an error occurs in the last several bits of the previously decoded sequence, it will directly affect the decoding of the subsequent codeword block.

To avoid this problem, we choose to increase the length of the decoder's input sequence. For instance, when we attend to decode a (n,k,l) convolutional code with block length L, the input length should be $n \times (L+l-1)$, where k/n is the code rate and l is the constraint length. This is because the last group of bits in the information sequence is directly involved in the encoding of the subsequent $l-1$ codeword groups, as shown in the left half of Fig. 3.

However, in the experiments, the results obtained by this Unidirectional Context Aggregation (UCA) method were not satisfactory. After further research and trials, we adopted the method of adding input information before and after the codeword blocks, as shown in the right half of Fig. 3. More specifically, if we want to decode a convolutional code with a block length of L, we need to take an additional $n \times (l-1)$-length bits before and after this code block, respectively, and input them together into the decoder. The effectiveness of this Bidirectional Context Aggregation (BCA) method is remarkable. The first $n \times (l-1)$ bits play an important role in the decoding procedure. Although the first $n \times (l-1)$ bits of the codeword sequence are not directly involved in the encoding process of the information sequence, they exhibit a strong correlation with the last $l-1$ groups of the previous information block. These latter $l-1$ groups are directly involved in the encoding of the current codeword block. Our Transformer-based model can effectively capture these implicit relationships and leverage them during the decoding process.

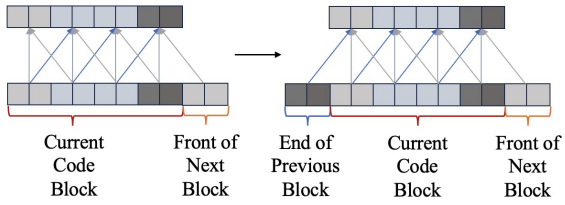

Figure 3: Unidirectional Context Aggregation and Bidirectional Context Aggregation.

3.2 Implicit Inter-bit Information Extraction

We intend to use the algebraic structure of the code to modify the self-attention weights. Since not every position in the received codeword is related to all other positions, it is necessary to design a module similar to a mask matrix based on the codeword type to weight and adjust the self-attention.

We first considered adopting the mask strategy from ECCT [Choukroun and Wolf, 2022], which directly applies binary masking based on the parity-check matrix to filter self-attention connections. But as convolutional codes naturally

have sparse parity-check matrices due to their recursive encoding structure with memory elements, applying a binary mask derived from such sparse matrices would overly restrict the attention scope and mask out too much potentially useful information. This over-constraint leads to suboptimal performance, especially when dealing with noisy channels where subtle relationships between distant bits can aid in error correction.

Intuitively, positions that are farther apart should have a weaker influence on each other. Thus, the distance between codewords is an important indicator to consider when constructing the weight matrix. As a graphical representation of the algebraic structure of codes, the Tanner graph establishes a connection between codeword bits and parity-check constraints, which is an effective tool for capturing the inherent relationships in convolutional codes. We propose to design our attention mechanism based on it to enhance the model's ability to model long-distance dependencies and improve decoding accuracy.

Inspired by the Foundation Error Correction Code Transformer (FECCT) framework [Choukroun and Wolf, 2024a], we construct a trainable single-input single-output multilayer perceptron (MLP) M to map the distance values to attention weights. The MLP M consists of a linear layer of 1-to-32 dimensions, a ReLU nonlinear activation layer, and a 32-to-1-dimensional linear layer. This mapping transforms the codeword distances into weights that reflect the strength of the structural relationship between different positions. We then apply these weights to the initially computed attention matrix, through which the mechanism focuses attention on positions with shorter codeword distances.

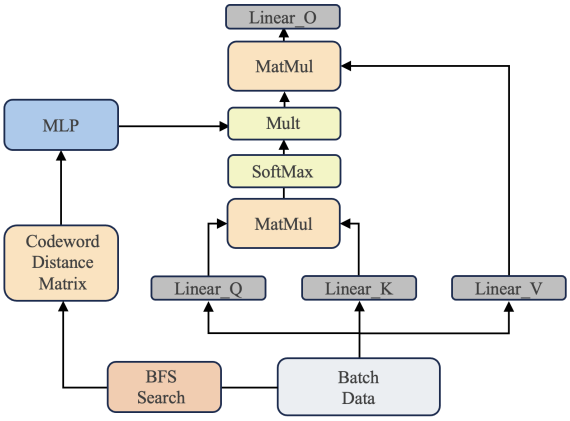

Figure 4: Implicit inter-bit correlation self-attention.

Thus, for a given code type with a constructible parity-check matrix H, we propose to define a function $\psi(H)$ with input $H \in \{0,1\}^{(2l-2+L)(n-k)\times(2l-2+L)k}$ and output weight matrix $W \in \mathbb{R}^{(2l-2+L)(2n-k)\times(2l-2+L)(2n-k)}$, which will then be applied to the raw self-attention calculation. The function ψ consists of a Breadth First Search (BFS) algorithm for Tanner graph and MLP M as afore mentioned. Thus, our implicit inter-bit correlation self-attention mechanism can be summarized as

$$T_H(Q,K,V) = \left(\text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \odot \psi(H)\right) V. \quad (1)$$

We detail the implementation of the inter-bit correlation self-attention mechanism in Fig. 4. This design offers greater flexibility in capturing structural correlations, as the learnable MLP can model non-linear relationships between distances and attention weights that extend beyond simple adjacency rules. While a standard Transformer treats all position pairs with equal potential relevance in its attention computation, the proposed distance-based weighting can be viewed as a dynamic adjacency matrix where edge strengths are determined by the algebraic distance in the Tanner graph.

3.3 Architecture

The input of the model is a $(2l-2+L)n$-dimensional vector. In the embedding stage, features are extracted through different methods and then concatenated into a $(2l-2+L)(2n-k)$-dimensional vector. Subsequently, each element of this vector is embedded into a d_{model}-dimensional word vector via a trainable embedding matrix src_emb. The decoder module is composed of N_{dec} stacked encoding layers, each with identical structure but independent parameters. Each encoding layer consists of an inter-bit correlation self-attention sublayer and a feed-forward sublayer, connected through residual connections and layer normalization. The output integration part is dual to the embedding stage: it intercepts the corresponding feature intervals, processes them separately through fully connected layers, and then integrates them to form the final output. An illustration of the model is presented in Fig. 5.

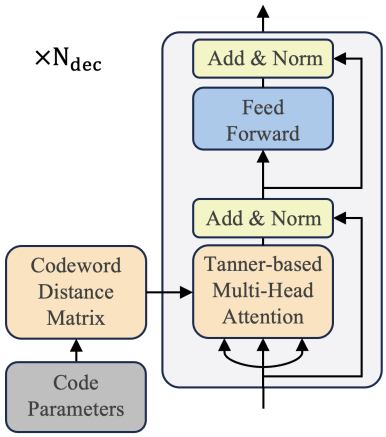

Figure 5: The architecture of the proposed framework.

The feed-forward sublayer consists of a dimension expansion layer, a gated activation layer, and a dimension compression layer. The dimension expansion layer maps the output of the self-attention sublayer to a higher dimension, which is set to $4d_{model}$ in our implementation, and the gated activation layer applies the GELU activation function to introduce non-linearity.

The output mapping module first splits the encoder output into magnitude and syndrome feature embeddings. It processes the latter with H^T before concatenating them with the former, and finally feeds the result into a fully connected layer to obtain the predicted value z_{pred}.

3.4 Training

The loss function is defined based on the cross-entropy between the model's prediction z_{pred} and the target $0.5(1 - \text{sign}(z_{mul}))$, where $z_{mul} = y \times \hat{x}$. The prediction of the original codeword is derived as $x_{pred} = 0.5(1 - \text{sign}(z_{pred} \times \text{sign}(y)))$.

We adopt the Adam optimizer [Kingma and Ba, 2014] along with a cosine annealing learning rate scheduling strategy during the training process. The learning rate is initialized to 10^{-4} and decays down to $1 \cdot 10^{-6}$ by the end of training. By default, we set the batch size to 128, $d_{model} = 32$ and $N_{\text{dec}} = 6$ for standard architectures, and the number of epochs to 1000. We typically set the number of blocks for single codeword data in the range of $80 - 120$.

In the training phase, noise with random Signal-to-Noise Ratio (SNR) values is adopted. For data usage, the training process exclusively utilizes all-zero codewords, while the testing phase employs randomly generated codewords. This training strategy is widely adopted in [Choukroun and Wolf, 2022; Park *et al.*, 2025b], as the all-zero codeword alone is sufficient for model training. The training and inference processes are performed on an NVIDIA RTX 5090 GPU with 32 GB of memory and an NVIDIA RTX 3090 Ti with 24 GB of memory. The underlying computing hardware includes multiple processors: an AMD EPYC 9754 128-Core Processor and an Intel(R) Xeon(R) Gold 5220R CPU. The system is equipped with 503 GiB of system memory. The software environment is built on Ubuntu 22.04.5 LTS with Python 3.13.2 as the main programming language. Training time ranges from 12 to 48 hours, depending on the code length.

4 Experiments

We train the proposed architecture for convolutional codes of various code rates to evaluate our method. The codes we consider include (2,1,3), (2,1,4), (2,1,7), (3,1,3), (3,2,3) and (4,3,3), covering self-orthogonal convolutional codes and non-self-orthogonal convolutional codes; systematic convolutional codes and non-systematic convolutional codes. We compare our proposed method with the Viterbi-like RNN decoder introduced by [Kim *et al.*, 2018] as well as the conventional Viterbi algorithm. Additionally, we benchmark our approach against state-of-the-art Transformer-based decoders, including MMECCT [Park *et al.*, 2025a] and CrossMPT [Park *et al.*, 2025b]. The results are all obtained by reproducing the code and running it in the same environment. All the results are reported as bit error rates (BER) for different normalized SNR values. During the testing phase, our decoder processes at least 10^5 codewords to validate data reliability.

The results compared to the Viterbi algorithm are reported in Tab. 1, where we present the negative natural logarithm of the BER. We evaluated six distinct convolutional codes, each with two distinct codeword block lengths. The Viterbi decoder can be regarded as an optimal way to implement Maximum Likelihood (ML) decoding for convolutional codes at AWGN channels. From a comprehensive perspective, the BER performance of our method is competitive with the Viterbi algorithm for codewords at lower code rates. For higher rate codewords, the performance gap of our method is acceptable given the substantial computational overhead of the Viterbi algorithm.

We then compare our method with NViterbi. The NViterbi architecture is designed for the rate-1/2 RSC code, and can be trained to decode other types of convolutional codes by adjusting network parameters. Reproducing experiments with the model parameters of the rate-1/2 convolutional code from their paper, we find that the model performed effectively for smaller constraint lengths (e.g., a constraint length of 3 or 4). However, its performance significantly deteriorated for longer constraint lengths (e.g., a constraint length of 7). Therefore, we conducted a comparative analysis of the (2,1,3) and (2,1,4) codes, evaluating both our proposed model and the NViterbi algorithm in Fig. 6. As can be seen, our method outperforms NViterbi at every normalized SNR value.

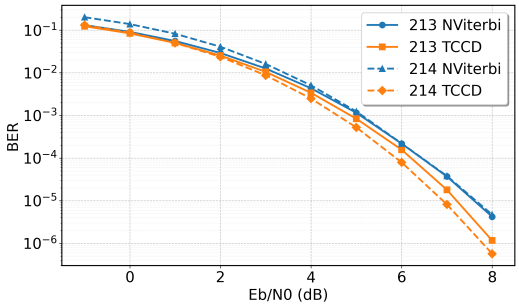

Figure 6: Comparison of BER performance for two distinct convolutional codes, (2,1,3) and (2,1,4), each with a block length of 100. Two decoding methods were evaluated: our proposed method (TCCD) and NViterbi.

In addition, we benchmark our proposed method against the state-of-the-art error correction code decoders MMECCT and CrossMPT in Fig. 7. The evaluated codewords cover six different parameter configurations, all with a block length of 40. We observe that the BER curves of MMECCT and CrossMPT are highly similar. Overall, our decoder demonstrates comparable performance to these methods at low signal-to-noise ratios, while exhibiting a clear advantage at higher signal-to-noise ratios.

We further conduct simulation experiments under Rayleigh fading channel. The received signal over the channel is given as: $\mathbf{Y} = \mathbf{Hx} + \mathbf{Z}$, where $\mathbf{H} \in \mathbb{R}^{N_r \times N_t}$ denotes the Rayleigh fading channel matrix, in which each element is an independent and identically distributed Rayleigh fading coefficient with the scale parameter $\alpha = 1$; N_t represents the number of transmit antennas and N_r denotes the number of receive antennas. We adopt a standard 2×2 multiple input multiple output (MIMO) configuration for performance evaluation, experiments on $(2, 1, 3)$ codewords are performed based on the above channel model and we compare it with Viterbi method. We evaluate the decoding accuracy of the proposed scheme

Code	TCCD						Viterbi					
	blk_len 40			blk_len 100			blk_len 40			blk_len 100		
	4	5	6	4	5	6	4	5	6	4	5	6
(2,1,3)	5.65	7.03	8.71	5.68	7.09	8.85	5.77	7.05	8.59	5.89	7.24	8.89
(2,1,4)	5.94	7.45	9.40	6.01	7.54	9.40	6.18	7.50	8.94	6.39	7.85	9.48
(2,1,7)	5.08	6.90	9.32	4.99	6.81	9.24	6.29	7.66	9.06	7.27	8.59	9.92
(3,1,3)	5.56	6.81	8.30	5.58	6.63	8.32	5.86	7.07	8.46	5.98	7.22	8.66
(3,2,3)	5.70	7.17	9.08	5.71	7.22	9.07	2.77	5.98	9.85	5.89	7.41	9.24
(4,3,3)	5.79	7.46	9.49	5.82	7.49	9.58	6.04	7.76	9.74	6.02	7.74	9.80

Table 1: A comparison of the negative natural logarithm of BER for three normalized SNR values of our method with the Viterbi algorithm. Higher is better.

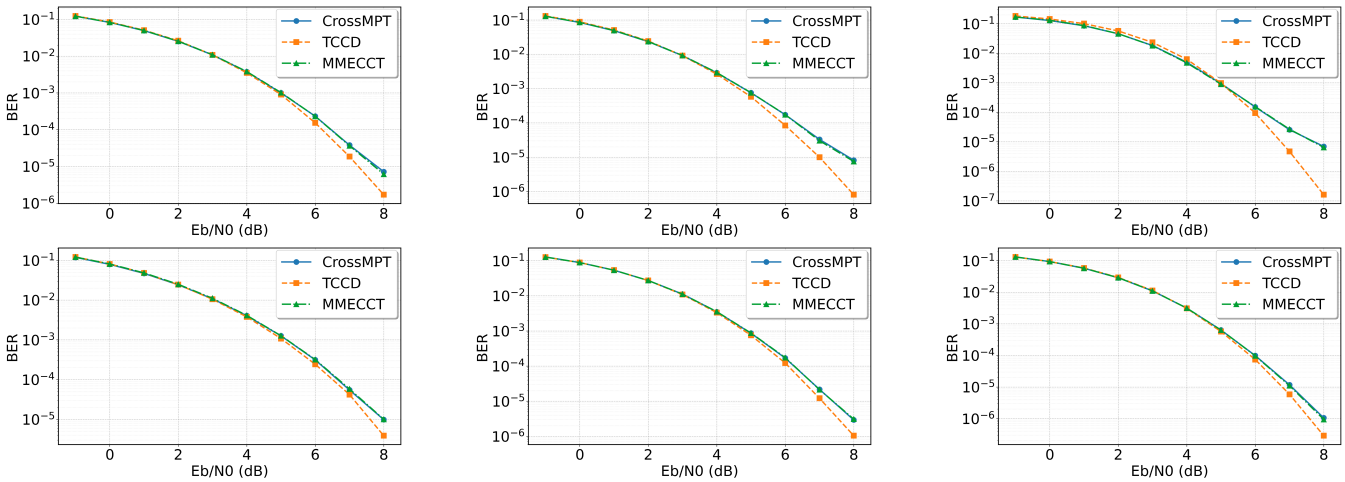

Figure 7: A comparison of bit error rate (BER) performance among CrossMPT, MMECCT, and the proposed TCCD under six distinct convolutional code configurations. The top row shows results for codes with parameters (2,1,3),(2,1,4),(2,1,7) from left to right, while the bottom row corresponds to (3,1,3), (3,2,3), (4,3,3), respectively.

under the above experimental setup, and the detailed experimental results are presented as in Fig. 8. Although there remains a gap between our BER and that of maximum likelihood decoding, the sufficiently strong performance demonstrates the applicability of our approach over Rayleigh fading channels.

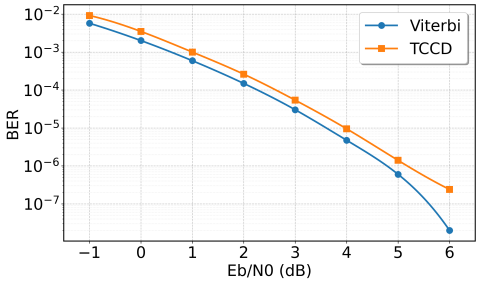

Figure 8: BER performance comparison of TCCD and Viterbi decoding algorithms for the (2,1,3,100) codeword over 2×2 MIMO Rayleigh fading channel.

5 Analysis

In this section, we study the impact of the proposed bidirectional context aggregation and implicit inter-bit information extraction. We also examine and compare the computational complexity and decoding latency with the conventional Viterbi algorithm.

5.1 Ablation Study

We first study the impact of our bidirectional context aggregation mechanism. As shown in Fig. 9, the benefit obtained by our bidirectional context aggregation mechanism is clear and understandable, which further corroborates the transformer's ability to extract such implicit relationships between adjacent code blocks.

We then demonstrated the effectiveness of the implicit inter-bit information extraction method. We compare the parity check matrix masking method in ECCT [Choukroun and Wolf, 2022] with our attention weight fine-tuning, as shown in Fig. 10. It is evident that our method yields superior performance, indicating that leveraging the hidden internal in-

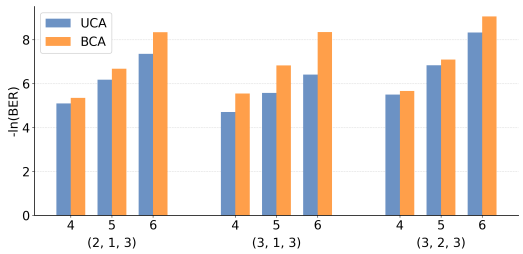

Figure 9: Comparison of the BCA and UCA input processing methods. All codewords are tested under a fixed block length of 20.

formation between bits is beneficial for improving decoding efficiency in convolutional codes.

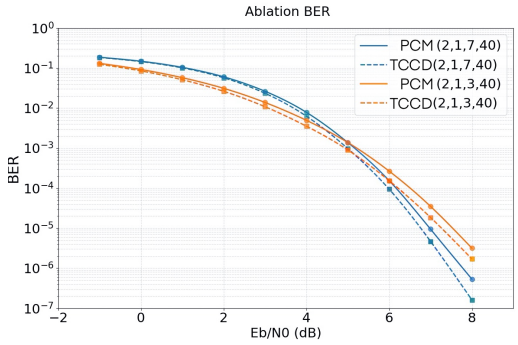

Figure 10: BER performance for parity check matrix (denoted as PCM) mask and attention weight fine-tune method used in TCCD on two convolutional codes.

We investigate the impact of the single-input single-output multi-layer perceptron size, as proposed in the implicit inter-bit information extraction mechanism, on the performance of our model. Our tests across two different codewords revealed that network size had minimal impact on model performance, as shown in Fig. 11. This suggests that a perceptron with a hidden layer width of 32 is sufficient to capture the necessary information.

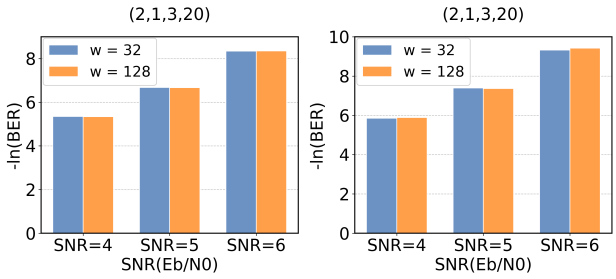

Figure 11: Comparison of different hidden layer widths in perceptron.

5.2 Computational Complexity

The complexity of the network is defined by $\mathcal{O}(N_{dec}((L+2l-2)^2(2n-k)^2 d_{model} + (L+2l-2)(2n-k)d_{model}^2))$,

Code	TCCD		MMECCT		CrossMPT	
	Parameter	FLOPs	Parameter	FLOPs	Parameter	FLOPs
(2,1,7)	8.54×10^4	2.11×10^7	1.91×10^5	1.82×10^7	8.61×10^4	9.08×10^6
(3,2,3)	9.30×10^4	2.52×10^7	2.23×10^5	2.42×10^7	9.57×10^4	1.21×10^7
(4,3,3)	1.05×10^5	3.52×10^7	2.65×10^5	3.03×10^7	1.09×10^5	1.51×10^7

Table 2: A Comparison of parameters and FLOPs for TCCD with MMECCT and CrossMPT.

	Viterbi(ms)	TCCD(ms)	MMECCT(ms)	CrossMPT(ms)
(2,1,7)	114.6	3.735	6.012	44.962
(3,2,3)	2692.0	11.395	7.036	27.266
(4,3,3)	12850.0	33.743	5.907	92.709

Table 3: A comparison of decoding latency of TCCD with Viterbi and Transformer-based decoders.

where k/n is the code rate, l is the constraint length, L is the block length and N_{dec}, d_{model} are network hyperparameters. Given the same codeword parameters and an identical decoding length, the Viterbi algorithm exhibits a decoding complexity of $(L+l-1)n \cdot 2^{k(l-1)}$, where $k(l-1)$ is the number of trellis states. Despite its high performance, our proposed method, much like the majority of neural network decoding algorithms, lacks sufficient efficiency when compared to traditional decoding algorithms, with the exception of cases where number of trellis states is the primary determinant of complexity. In Tab. 2, we evaluate the computational complexity (parameters and FLOPs) of TCCD, MMECCT, and CrossMPT across various (n, k, l) configurations with a fixed block length of 40.

5.3 Decoding Latency

Decoding latency is a crucial metric in real-time communication systems. We evaluate the proposed TCCD decoder against the conventional Viterbi decoder and Transformer-based decoders. The evaluation results were presented with a floating-point precision of float32, based on 10^5 trials with an information block length of 40. As shown in Tab. 3, Our proposed method achieves a latency between those of MMECCT and CrossMPT, and is substantially lower than that of the conventional Viterbi algorithm.

6 Conclusion

This paper proposes a convolutional code decoder based on the Transformer architecture, which is capable of efficiently processing various types of codewords. The proposed model achieves state-of-the-art BER accuracy compared to high-performance neural network decoders by leveraging the bidirectional context aggregation mechanism and implicit inter-bit information extraction. Our method achieves decoding accuracy close to that of maximum-likelihood decoding, while offering significantly higher decoding throughput. Future work can focus on optimizing the model's architecture to further reduce computational complexity and latency, as well as extending the proposed framework to other sequential codes like punctured convolutional codes or Turbo codes, collectively contributing to the development of next-generation AI-powered communication systems.

Acknowledgments

The work was supported in part by the National Natural Science Foundation of China under Grants 12426302, 12271084, 12361141818 and the National Key Research and Development Program of China under Grants 2022YFA1004900 and 2023YFE0123900.

Contribution Statement

Guangya Hu and Tianyi Li contributed equally to this work.

References

[3GPP, 2000] 3GPP. 3GPP TS 25.401: UTRAN overall description. https://www.3gpp.org/ftp/Specs/archive/25_series/25.401/, 2000. Release 4 (2000-01).

[Bahl *et al.*, 1974] Lalit Bahl, John Cocke, Frederick Jelinek, and Josef Raviv. Optimal decoding of linear codes for minimizing symbol error rate (corresp.). *IEEE Transactions on information theory*, 20(2):284–287, 1974.

[Bennatan *et al.*, 2018] Amir Bennatan, Yoni Choukroun, and Pavel Kisilev. Deep learning for decoding of linear codes-a syndrome-based approach. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 1595–1599. IEEE, 2018.

[Buchberger *et al.*, 2021] Andreas Buchberger, Christian Häger, Henry D Pfister, Laurent Schmalen, and Alexandre Graell i Amat. Learned decimation for neural belief propagation decoders. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8273–8277. IEEE, 2021.

[Cammerer *et al.*, 2017] Sebastian Cammerer, Tobias Gruber, Jakob Hoydis, and Stephan Ten Brink. Scaling deep learning-based decoding of polar codes via partitioning. In *GLOBECOM 2017-2017 IEEE global communications conference*, pages 1–6. IEEE, 2017.

[Choukroun and Wolf, 2022] Yoni Choukroun and Lior Wolf. Error correction code transformer. *Advances in Neural Information Processing Systems*, 35:38695–38705, 2022.

[Choukroun and Wolf, 2023] Yoni Choukroun and Lior Wolf. Denoising diffusion error correction codes. In *The Eleventh International Conference on Learning Representations*, 2023.

[Choukroun and Wolf, 2024a] Yoni Choukroun and Lior Wolf. A foundation model for error correction codes. In *The Twelfth International Conference on Learning Representations*, 2024.

[Choukroun and Wolf, 2024b] Yoni Choukroun and Lior Wolf. Learning linear block error correction codes. In *International Conference on Machine Learning*, pages 8801–8814. PMLR, 2024.

[Doan *et al.*, 2018] Nghia Doan, Seyyed Ali Hashemi, and Warren J Gross. Neural successive cancellation decoding of polar codes. In *2018 IEEE 19th international workshop on signal processing advances in wireless communications (SPAWC)*, pages 1–5. IEEE, 2018.

[Doan *et al.*, 2019] Nghia Doan, Seyyed Ali Hashemi, Elie Ngomseu Mambou, Thibaud Tonnellier, and Warren J Gross. Neural belief propagation decoding of crc-polar concatenated codes. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2019.

[Elias, 1955] Peter Elias. Coding for noisy channels. In *IRE WESCON Convention Record, 1955*, volume 2, pages 94–104, 1955.

[ETSI Secretariat, 1996] ETSI Secretariat. Digital cellular telecommunications system (Phase 2); General description of a GSM Public Land Mobile Network (PLMN). https://www.etsi.org/deliver/etsi_gts/04/0408/05.03.00_60/gsmts_0408v050300p.pdf, 1996. ETSI GSM 04.08 V5.3.0 (1996-07).

[Graves, 2012] Alex Graves. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45, 2012.

[Gruber *et al.*, 2017] Tobias Gruber, Sebastian Cammerer, Jakob Hoydis, and Stephan Ten Brink. On deep learning-based channel decoding. In *2017 51st annual conference on information sciences and systems (CISS)*, pages 1–6. IEEE, 2017.

[Ho *et al.*, 2020] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[Hoydis *et al.*, 2022] Jakob Hoydis, Sebastian Cammerer, Fayçal Ait Aoudia, Avinash Vem, Nikolaus Binder, Guillermo Marcus, and Alexander Keller. Sionna: An open-source library for next-generation physical layer research. *arXiv preprint arXiv:2203.11854*, 2022.

[IEEE, 2021] IEEE. IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Std., Feb 2021. IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016).

[Jiang *et al.*, 2019] Yihan Jiang, Sreeram Kannan, Hyeji Kim, Sewoong Oh, Himanshu Asnani, and Pramod Viswanath. Deepturbo: Deep turbo decoder. In *2019 IEEE 20th international workshop on signal processing advances in wireless communications (SPAWC)*, pages 1–5. IEEE, 2019.

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

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Kurmukova and Gunduz, 2024] Anastasiia Kurmukova and Deniz Gunduz. Friendly attacks to improve channel coding reliability. In *Proceedings of the AAAI Conference*

on Artificial Intelligence, volume 38, pages 13292–13300, 2024.

[Lugosch and Gross, 2017] Loren Lugosch and Warren J Gross. Neural offset min-sum decoding. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 1361–1365. IEEE, 2017.

[Nachmani *et al.*, 2016] Eliya Nachmani, Yair Be'Ery, and David Burshtein. Learning to decode linear codes using deep learning. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 341–346. IEEE, 2016.

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

[Park *et al.*, 2025a] Seong-Joon Park, Hee-Youl Kwak, Sang-Hyo Kim, Sunghwan Kim, Yongjune Kim, and Jong-Seon No. Multiple-masks error correction code transformer for short block codes. *IEEE Journal on Selected Areas in Communications*, 2025.

[Park *et al.*, 2025b] Seong-Joon Park, Hee-Youl Kwak, Sang-Hyo Kim, Yongjune Kim, and Jong-Seon No. CrossMPT: Cross-attention message-passing transformer for error correcting codes. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

[Teich and Pan, 2023] Werner G Teich and Weikun Pan. Scalable convolutional neural networks for decoding of terminated convolutional codes. In *International Work-Conference on Artificial Neural Networks*, pages 43–54. Springer, 2023.

[Teich *et al.*, 2019] Werner G Teich, Heiko Teich, and Giuseppe Oliveri. From iterative threshold decoding to a low-power high-speed analog vlsi decoder implementation. In *International Work-Conference on Artificial Neural Networks*, pages 615–628. Springer, 2019.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[Viterbi, 1967] Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theory*, 13(2):260–269, 1967.

[Xiao *et al.*, 2025] Shuai Xiao, Weijun Fang, and Bin Chen. Transformer-based decoders for cyclic codes: A tanner cycle-equivalent approach. *IEEE Transactions on Communications*, 73(12):13173–13185, 2025.

[Xu *et al.*, 2017] Weihong Xu, Zhizhen Wu, Yeong-Luh Ueng, Xiaohu You, and Chuan Zhang. Improved polar decoder based on deep learning. In *2017 IEEE International workshop on signal processing systems (SiPS)*, pages 1–6. IEEE, 2017.

[Yan and Dai, 2025] Yongli Yan and Linglong Dai. Decoding for punctured convolutional and turbo codes: A deep learning solution for protocols compliance. *arXiv preprint arXiv:2502.15475*, 2025.

[Zhang *et al.*, 2020] Zhengyu Zhang, Dongping Yao, Lei Xiong, Bo Ai, and Shuo Guo. A convolutional neural network decoder for convolutional codes. In *Communications and Networking: 14th EAI International Conference, ChinaCom 2019, Shanghai, China, November 29–December 1, 2019, Proceedings, Part II 14*, pages 113–125. Springer, 2020.