EMBEDDED NEURAL BELIEF PROPAGATION: MORE DIMENSIONS, BETTER BELIEFS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recently, Transformer-based decoders have achieved state-of-the-art performance on short codes, surpassing conventional belief propagation (BP) decoders. However, processing high-dimensional representations through attention incurs substantial computational cost. In this paper, we propose Embedded Neural Belief Propagation (E-NBP), which combines the efficiency of BP with the expressive power of high-dimensional representations. E-NBP embeds channel information into d-dimensional vectors and propagates message vectors along Tanner-graph edges. A key design choice is to perform intra-dimensional check-node updates to maintain low complexity, while performing cross-dimensional variable-node updates to improve decoding performance. On CCSDS$(128, 64)$ at $E_b/N_0 = 6$ dB, E-NBP reduces BER by 98.1% relative to CrossMPT while requiring 73.7% fewer FLOPs than EfficientMPT. These findings show that vector messages enable BP to compete with Transformer-based decoders while retaining its computational efficiency.

1 INTRODUCTION

Error-correcting codes (ECCs) enable reliable communication over noisy channels by introducing structured redundancy that allows receivers to decode transmitted information. Belief propagation (BP) is a widely used iterative decoding algorithm, offering strong error-correction performance at low computational cost (Richardson & Urbanke, 2001). BP exchanges scalar messages along the edges of a Tanner graph and is particularly effective for long and sparse codes. For short or dense codes, however, Tanner graphs often contain many short cycles, causing repeated messages to become correlated and thereby limiting BP's decoding performance (Richardson & Urbanke, 2001; Nachmani et al., 2018).

Neural belief propagation (NBP) decoders address this limitation by introducing learnable weights into BP updates to mitigate the effects of short cycles (Nachmani et al., 2016; 2018). Subsequent studies have further improved NBP through advances in parameterization, conditioning, compression, and training strategies (Nachmani & Wolf, 2019; 2021; Buchberger et al., 2021; Kwak et al., 2025). Graph neural network (GNN)-based decoders use high-dimensional node representations and learned message updates to improve decoding performance (Cammerer et al., 2022). Despite these advances, BP-based iterative decoding has yet to achieve universally strong performance across diverse code families.

Transformer-based decoders, including ECCT, CrossMPT, and EfficientMPT, have opened a promising direction for addressing these limitations by combining high-dimensional embeddings with attention mechanisms (Choukroun & Wolf, 2022; Park et al., 2025; 2026). On short- and moderate-length code benchmarks, these decoders outperform conventional BP-based decoders. High-dimensional representations play an important role in these performance gains, providing greater expressive capacity than scalar BP messages. However, processing these representations through attention incurs substantial computational and memory costs, limiting their practical applicability.

Motivated by these developments, we augment NBP with high-dimensional representations to surpass the performance of transformer-based decoders while retaining efficient Tanner-graph-based iterative decoding. We propose *Embedded Neural Belief Propagation* (E-NBP), which embeds each channel log-likelihood ratio (LLR) into a d-dimensional vector and propagates vector messages along Tanner-graph edges. This extension increases the expressive capacity of BP messages beyond scalar

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

representations. In the variable-to-check (V2C) update, E-NBP separately transforms the extrinsic message aggregate and self-edge feedback, enabling cross-dimensional information exchange to improve decoding performance. In the check-to-variable (C2V) update, E-NBP applies the conventional sum-product (SP) operation independently in each dimension to maintain low computational complexity. Together, these updates enable high-dimensional message processing while preserving the iterative BP update structure.

Extensive experiments across BCH, polar, and LDPC codes demonstrate the effectiveness and efficiency of E-NBP. On CCSDS$(128, 64)$ at $E_b/N_0 = 6$ dB, E-NBP achieves a 98.1% lower BER than CrossMPT, the best-performing Transformer baseline on this code, while reducing FLOPs by 73.7% relative to EfficientMPT, the most computationally efficient Transformer baseline. On BCH$(63, 45)$, E-NBP with 12 iterations outperforms scalar BP even with up to 768 decoding iterations. These results suggest that increasing the message dimension can improve decoding performance more effectively than further increasing the iteration count. Together, these results show that augmenting BP with high-dimensional representations can outperform strong Transformer-based decoders and exceed the performance plateau of scalar BP even as the number of iterations increases.

Main contributions. Our contributions are summarized as follows.

- **Vector-message extension of NBP.** We propose E-NBP, which extends scalar BP messages to d-dimensional vectors through learnable LLR embeddings, bringing high-dimensional representations to Tanner-graph-based iterative decoding.
- **Cross-dimensional information exchange.** We enable cross-dimensional information exchange through separate transformations of the extrinsic message aggregate and self-edge feedback in V2C updates, while retaining intra-dimensional independent SP operations in C2V updates.
- **Toward universal BP-based decoding.** Without attention or general neural message transformations, E-NBP retains conventional BP-style sum and product operations while achieving strong performance across BCH, polar, and LDPC codes with diverse graph structures, demonstrating the broad applicability of high-dimensional BP decoding.

2 Related Work

We categorize existing neural decoders into *BP-based decoders* and *Transformer-based decoders*. Table 1 summarizes their message representations and update operations, together with those of E-NBP for comparison.

2.1 BP-Based Decoders

BP decoders iteratively update decoding messages using SP operations, providing low computational complexity and a hardware-friendly decoding structure (Richardson & Urbanke, 2001). For short or dense codes, however, short cycles induce correlations among exchanged messages and can limit decoding performance. NBP mitigates this limitation by introducing learnable weights into BP message updates (Nachmani et al., 2016; 2018). Subsequent studies have further improved NBP through alternative parameterizations, hypernetworks, autoregressive conditioning, pruning, and improved training strategies (Nachmani & Wolf, 2019; 2021; Buchberger et al., 2021; Kwak et al., 2025). Despite these advances, the underlying BP messages remain scalar, limiting their expressive capacity.

Neural enhanced BP combines BP with a factor-graph GNN that learns to improve BP messages (Garcia Satorras & Welling, 2021), while Cammerer et al. (2022) proposed a GNN decoder that employs high-dimensional node representations and vector messages with MLP-based message updates. E-NBP also employs vector messages, but differs in retaining the underlying BP update structure on the Tanner graph. In particular, E-NBP retains the conventional BP message-update operations while extending the messages themselves to high-dimensional representations, avoiding general neural transformations in the update rules. This preserves the hardware-friendly structure of BP while increasing its message expressiveness, enabling strong decoding performance across both sparse and dense code structures.

Table 1: Architectural comparison by decoder category, message representation, and update operation.

Category	Decoder	Message representation	Update operation
BP	BP (Richardson & Urbanke, 2001)	Scalar edge messages	BP
	NBP (Nachmani et al., 2016; 2018)		Weighted BP
	GNN (Cammerer et al., 2022)	Vector edge messages	MLP-based
	E-NBP (ours)		**Weighted BP + cross-dimensional exchange**
Transformer	ECCT (Choukroun & Wolf, 2022)	Embedded tokens	Masked self-attention
	CrossMPT (Park et al., 2025)		Masked cross-attention
	EfficientMPT (Park et al., 2026)		EEC attention

2.2 Transformer-Based Decoders

Transformer-based decoders embed channel magnitudes and syndromes into tokens and process them through attention mechanisms. ECCT uses self-attention to exchange information among these tokens (Choukroun & Wolf, 2022). CrossMPT separates magnitude and syndrome representations into two streams and exchanges information through cross-attention, achieving strong decoding performance across various code classes (Park et al., 2025). EfficientMPT, a recent efficiency-oriented Transformer decoder, introduces EEC attention to reduce the computational cost of Transformer-based decoding and extend it to longer codes (Park et al., 2026). Despite these advances, processing high-dimensional token representations remains computationally expensive. In contrast, E-NBP avoids attention-based token processing and retains BP-based Tanner-graph message passing, aiming to combine expressive vector messages with efficient decoding operations.

3 Background

3.1 Channel Coding Preliminaries

Consider a binary linear block code of length n, dimension k, and rate $R = k/n$. A binary codeword $\mathbf{x} = [x_1, \ldots, x_n]^\mathsf{T}$ satisfies $\mathbf{Hx} = \mathbf{0} \pmod 2$, where $\mathbf{H} \in \{0,1\}^{m \times n}$ is a parity-check matrix. Binary phase-shift keying (BPSK) maps x_v to $b_v = (-1)^{x_v}$, and transmission over an AWGN channel gives $y_v = b_v + \nu_v$ with $\nu_v \sim \mathcal{N}(0, \sigma^2)$. For unit-energy BPSK at $E_b/N_0 = \rho_b$ dB, $\sigma^2 = (2R\,10^{\rho_b/10})^{-1}$. The channel LLR is

$$m_v^{\mathrm{ch}} = \log \frac{p(y_v \mid x_v = 0)}{p(y_v \mid x_v = 1)} = \frac{2y_v}{\sigma^2}. \tag{1}$$

The parity-check matrix induces a Tanner graph $\mathcal{G} = (\mathcal{V}, \mathcal{C}, \mathcal{E})$, where $v \in \mathcal{V}$ denotes a variable node (VN), $c \in \mathcal{C}$ denotes a check node (CN), and $(v, c) \in \mathcal{E}$ if and only if $H_{c,v} = 1$. We denote the number of edges by $E = |\mathcal{E}|$ and the neighborhood of a node by $\mathcal{N}(\cdot)$. Decoding iterations are indexed by $\ell \in \{1, \ldots, \bar{\ell}\}$. The scalar variable to check (V2C) and check to variable (C2V) messages are denoted by $m_{v\to c}^{(\ell)}$ and $m_{c\to v}^{(\ell)}$, respectively; their directions are distinguished by the subscripts.

3.2 Neural Belief Propagation

Neural belief propagation (NBP) unfolds BP message passing into a trainable recursion by introducing learnable weights into the message updates (Nachmani et al., 2016; 2018). For the first V2C update at $\ell = 1$, we initialize the C2V messages as $m_{c\to v}^{(0)} = 0$, such that $m_{v\to c}^{(1)} = m_v^{\mathrm{ch}}$. For $\ell \in \{1, \ldots, \bar{\ell}\}$, the V2C update is

$$m_{v\to c}^{(\ell)} = m_v^{\mathrm{ch}} + \sum_{c' \in \mathcal{N}(v)\setminus\{c\}} m_{c'\to v}^{(\ell-1)}. \tag{2}$$

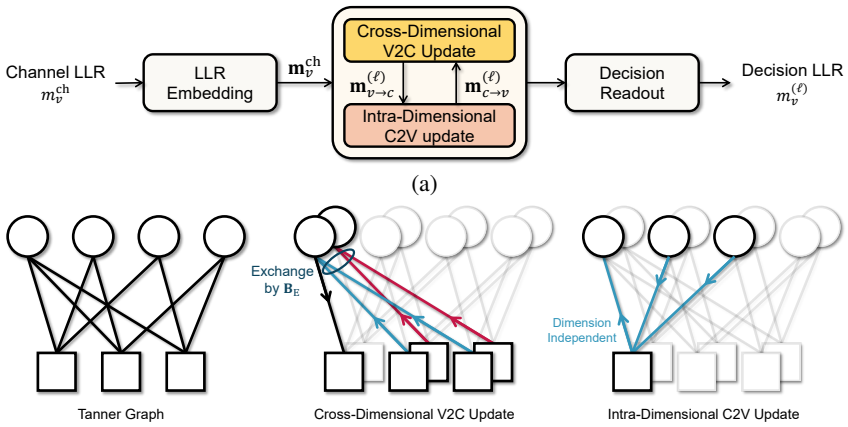


Figure 1: Overview of E-NBP. (a) Channel LLRs are embedded into vector messages $\mathbf{m}_v^{\mathrm{ch}}$, processed through V2C and C2V updates, and mapped to decision LLRs. (b) Tanner graph and message updates for $d = 2$, with overlapping layers representing the two message dimensions. Cross-dimensional V2C updates exchange information across dimensions by applying $\mathbf{B}_{\mathrm{E}}$ to the vector of dimension-wise extrinsic message sums, whereas C2V updates operate independently in each dimension.

The exclusion of the target check c gives the conventional extrinsic message update. For SP decoding, we apply an iteration- and edge-specific learnable weight to each C2V message:

$$m_{c\to v}^{(\ell)} = w_{c\to v}^{(\ell)}\, 2\tanh^{-1}\left[\prod_{v'\in\mathcal{N}(c)\setminus\{v\}} \tanh\left(\frac{m_{v'\to c}^{(\ell)}}{2}\right)\right], \tag{3}$$

where $w_{c\to v}^{(\ell)}$ denotes the learnable C2V weight. The decision LLR at iteration ℓ is

$$m_v^{(\ell)} = m_v^{\mathrm{ch}} + \sum_{c\in\mathcal{N}(v)} m_{c\to v}^{(\ell)}, \tag{4}$$

and the final hard decision is $\widehat{x}_v = \mathbf{1}[m_v^{(\bar{\ell})} < 0]$.

E-NBP extends this scalar recursion to vector messages, introducing cross-dimensional information exchange in the V2C update while retaining intra-dimensional weighted SP C2V operations.

4 Embedded Neural Belief Propagation

4.1 Architecture Overview and LLR Embedding

E-NBP extends each scalar edge message to a d-dimensional vector,

$$\mathbf{m}_{v\to c}^{(\ell)} = \left[m_{v\to c,1}^{(\ell)},\ldots,m_{v\to c,d}^{(\ell)}\right]^{\mathsf{T}} \in \mathbb{R}^d,\quad \mathbf{m}_{c\to v}^{(\ell)} = \left[m_{c\to v,1}^{(\ell)},\ldots,m_{c\to v,d}^{(\ell)}\right]^{\mathsf{T}} \in \mathbb{R}^d, \tag{5}$$

where $m_{v\to c,i}^{(\ell)}$ and $m_{c\to v,i}^{(\ell)}$ denote the ith components of the V2C and C2V messages at iteration ℓ, respectively. The C2V messages are initialized as $\mathbf{m}_{c\to v}^{(0)} = \mathbf{0}$.

E-NBP assigns each variable node v a trainable embedding $\mathbf{h}_v \in \mathbb{R}^d$ and maps the scalar channel LLR m_v^{ch} to $\mathbf{m}_v^{\mathrm{ch}} = m_v^{\mathrm{ch}}\mathbf{h}_v \in \mathbb{R}^d$. For $d > 1$, the embedding vectors are initialized as one plus Xavier-normal perturbations to break symmetry across message dimensions (Glorot & Bengio, 2010).

Figure 1 summarizes the E-NBP decoding flow and its two message updates. As shown in Figure 1(a), each channel LLR is embedded into a d-dimensional representation, processed through cross-dimensional V2C and intra-dimensional C2V updates, and mapped by the decision readout to a scalar decision LLR. Figure 1(b) first shows the underlying Tanner graph (left) and the V2C (middle) and C2V (right) updates for E-NBP with $d = 2$. In the V2C and C2V panels, the overlapping

graph layers represent the two message dimensions. The V2C panel shows that incoming extrinsic messages are first aggregated independently within each dimension, after which $\mathbf{B}_{\mathrm{E}}^{(\ell)}$ exchanges information across dimensions. In contrast, the C2V panel shows that the weighted SP operation is applied independently within each message dimension.

4.2 Cross-Dimensional V2C Update

Applied independently to each dimension, the conventional extrinsic V2C update becomes

$$m_{v\to c,i}^{(\ell)} = m_{v,i}^{\mathrm{ch}} + \sum_{c'\in\mathcal{N}(v)\setminus\{c\}} m_{c'\to v,i}^{(\ell-1)}, \qquad i = 1,\dots,d. \tag{6}$$

E-NBP augments this update with cross-dimensional information exchange:

$$\widetilde{\mathbf{m}}_{v\to c}^{(\ell)} = \mathbf{m}_{v}^{\mathrm{ch}} + \mathbf{B}_{\mathrm{E}}^{(\ell)} \sum_{c'\in\mathcal{N}(v)\setminus\{c\}} \mathbf{m}_{c'\to v}^{(\ell-1)} + \mathbf{B}_{\mathrm{S}}^{(\ell)} \mathbf{m}_{c\to v}^{(\ell-1)}. \tag{7}$$

The second and third terms correspond to the *extrinsic message aggregate* and *self-edge feedback*, respectively. For the extrinsic message aggregate, incoming C2V messages are first summed independently within each dimension, and the resulting d-dimensional vector is transformed by $\mathbf{B}_{\mathrm{E}}^{(\ell)} \in \mathbb{R}^{d\times d}$ to exchange information across dimensions. The self-edge feedback reintroduces the target-edge C2V message, which is excluded from the conventional extrinsic sum, and mixes its dimensions through $\mathbf{B}_{\mathrm{S}}^{(\ell)} \in \mathbb{R}^{d\times d}$ before incorporating it into the V2C update.

Both matrices are shared across edges and learned independently at each iteration. We initialize $\mathbf{B}_{\mathrm{E}}^{(\ell)} = \mathbf{I}$ and $\mathbf{B}_{\mathrm{S}}^{(\ell)} = \mathbf{0}$, so the V2C update initially follows the conventional BP structure without cross-dimensional information exchange. Conventional BP excludes the self-edge message to form an extrinsic V2C update. E-NBP instead exploits the self-edge messages by mixing their information across dimensions, a capability enabled by its high-dimensional message representation. A related use of self-edge information appears in memory belief propagation for quantum codes, where the self-edge message is reintroduced through a weighted inhibition term (Kuo & Lai, 2022).

Building on the learnable successive relaxation method of Nachmani et al. (2018), we update the multidimensional V2C messages using a learnable relaxation factor:

$$\mathbf{m}_{v\to c}^{(\ell)} = g_{v\to c}^{(\ell)} \widetilde{\mathbf{m}}_{v\to c}^{(\ell)} + \left(1 - g_{v\to c}^{(\ell)}\right) \mathbf{m}_{v\to c}^{(\ell-1)}. \tag{8}$$

Here, $g_{v\to c}^{(\ell)}$ is a learnable relaxation factor associated with each Tanner edge and shared across the d message dimensions. We set $\mathbf{m}_{v\to c}^{(0)} = \mathbf{0}$ and initialize $g_{v\to c}^{(\ell)} = 1$. The relaxation factors are projected to $[0, 2]$ after each optimization step.

4.3 Intra-Dimensional C2V Update and Decision Readout

The C2V update applies the NBP-weighted SP operation independently to each message dimension:

$$m_{c\to v,i}^{(\ell)} = w_{c\to v,i}^{(\ell)}\, 2\tanh^{-1}\left[\prod_{v'\in\mathcal{N}(c)\setminus\{v\}} \tanh\left(\frac{m_{v'\to c,i}^{(\ell)}}{2}\right)\right], \qquad i = 1,\dots,d. \tag{9}$$

The weight $w_{c\to v,i}^{(\ell)}$ is learned independently for each iteration, Tanner edge, and message dimension and is initialized to one. The SP C2V update is typically more computationally demanding than the V2C summation because it involves products and nonlinear operations over the incoming messages, with the computational burden increasing as the CN degree grows. E-NBP therefore performs the C2V update independently in each message dimension, so that the number of incoming messages processed by each CN remains the same as in the original Tanner graph within each dimension. This design limits the increase in C2V complexity to linear scaling with the message dimension.

After each C2V update, an iteration-specific readout maps the accumulated d-dimensional messages to the decision LLR:

$$m_{v}^{(\ell)} = \mathbf{r}^{(\ell)} \cdot \left(\mathbf{m}_{v}^{\mathrm{ch}} + \sum_{c\in\mathcal{N}(v)} \mathbf{m}_{c\to v}^{(\ell)}\right), \qquad \mathbf{r}^{(\ell)} \in \mathbb{R}^{d}. \tag{10}$$

The readout is shared across variable nodes, initialized as $r_i^{(\ell)} = 1/d$, and learned independently at each iteration. The final hard decision is $\widehat{x}_v = \mathbf{1}[m_v^{(\bar{\ell})} < 0]$.

Following Nachmani et al. (2018), we employ a multi-loss objective that averages the binary cross-entropy over bits and iterations:

$$\mathcal{L} = -\frac{1}{n\bar{\ell}} \sum_{\ell=1}^{\bar{\ell}} \sum_{v=1}^{n} \log \left(\text{sigmoid}\Big((-1)^{x_v} m_v^{(\ell)} \Big) \right). \tag{11}$$

5 Experiments

5.1 Experimental Setup

Codes and channel. Following the ECCT and CrossMPT benchmarks (Choukroun & Wolf, 2022; Park et al., 2025), we evaluate BCH, polar and LDPC codes using parity-check matrices from Helmling et al. (2019). The long-code experiments use WiMAX LDPC$(576, 432)$, WiMAX LDPC$(1056, 880)$, and WiMAX LDPC$(2304, 1920)$ from the IEEE 802.16 standard (IEEE, 2009). We use BPSK transmission over AWGN.

Baselines and decoder settings. We compare E-NBP with BP, NBP, GNN, ECCT, CrossMPT, and EfficientMPT (Nachmani et al., 2016; 2018; Cammerer et al., 2022; Choukroun & Wolf, 2022; Park et al., 2025; 2026). For short codes in Fig. 2, E-NBP uses message dimension $d = 64$ and $\bar{\ell} = 12$ iterations, while for long codes in Fig. 3, it uses $d = 128$ and $\bar{\ell} = 10$. We evaluate BP and NBP using the same number of iterations as E-NBP for the main short-code comparison. The GNN decoder is trained and evaluated using its public implementation. For the Transformer-based decoders (ECCT, CrossMPT, and EfficientMPT), we follow their recommended configurations, using embedding dimension $d = 128$ with $N = 6$ layers for short codes and $N = 10$ layers for long codes. Reference results for ECCT and CrossMPT are taken from Park et al. (2025), while EfficientMPT results combine published values with additional evaluations using its official implementation. Other analysis-specific configurations are stated alongside the corresponding experiments.

Training and evaluation. E-NBP is trained offline using samples drawn from an E_b/N_0 range of 3 to 7 dB. We use Adam (Kingma & Ba, 2015) with mini-batches of 128 frames and 1000 mini-batches per epoch. Code-specific training settings are detailed in Table 3 and the accompanying text. At each test SNR, we evaluate decoding performance using randomly generated codewords until both at least 10^5 frames and 300 frame errors have been observed, or until 10^9 frames have been evaluated, whichever occurs first. Reported SNR gains are obtained by linear interpolation in log-BER.

5.2 Decoding Performance

As shown in Figure 2, E-NBP achieves lower BER than the evaluated Transformer-based decoders on both BCH codes and both LDPC codes across the evaluated SNRs, while remaining competitive on Polar$(64, 32)$. On BCH$(63, 45)$ at 5 dB, E-NBP reduces BER by more than one order of magnitude relative to NBP, the BP-based baseline without message embedding. Compared with CrossMPT, the strongest Transformer baseline on this code, E-NBP achieves a 67.4% BER reduction at 5 dB and provides an SNR gain of 0.36 dB at a BER of 10^{-5}. The performance gap becomes even larger on CCSDS$(128, 64)$: at 6 dB, E-NBP achieves more than a two-order-of-magnitude BER reduction over NBP and BER reductions of 98.1% and 98.4% relative to CrossMPT and EfficientMPT, respectively. At a BER of 10^{-7}, E-NBP provides SNR gains of 0.60 dB and 0.62 dB over CrossMPT and EfficientMPT, respectively. On Polar$(64, 32)$, E-NBP also addresses a key limitation of conventional BP on dense code structures, achieving performance comparable to CrossMPT.

Furthermore, E-NBP extends its performance advantage to long codes, as shown in Figure 3. On WiMAX LDPC$(1056, 880)$ at 5 dB, E-NBP reduces BER by 71.2% relative to EfficientMPT. These results demonstrate the effectiveness of E-NBP across short and long codes, including a practical LDPC code of length 2304.

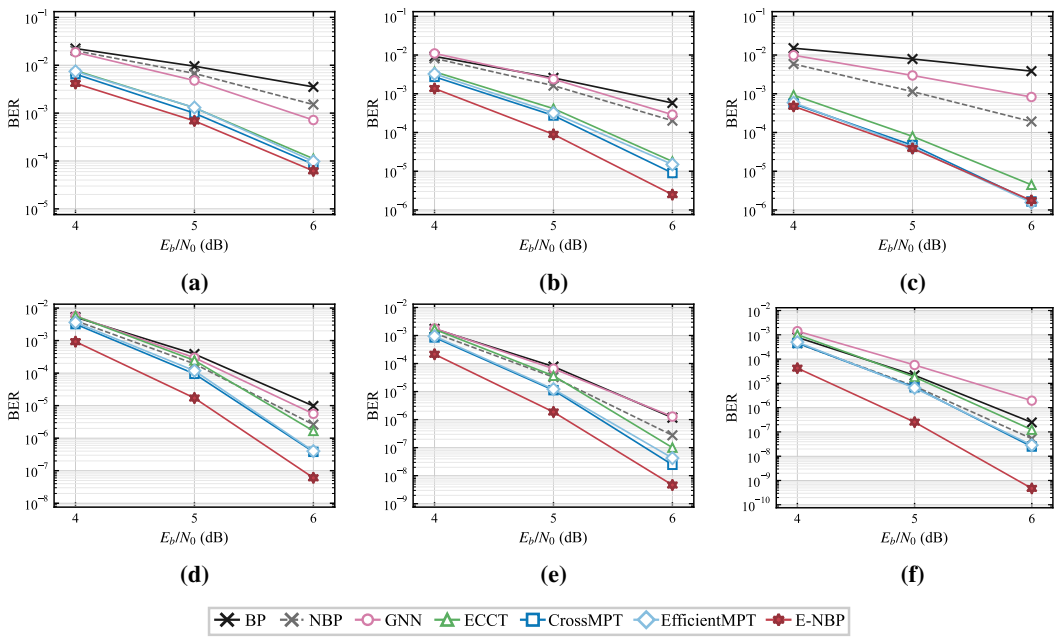


Figure 2: BER performance at $E_b/N_0 \in \{4, 5, 6\}$ dB: (a) BCH$(63, 36)$, (b) BCH$(63, 45)$, (c) Polar$(64, 32)$, (d) LDPC$(121, 60)$, (e) LDPC$(121, 70)$, and (f) CCSDS$(128, 64)$.

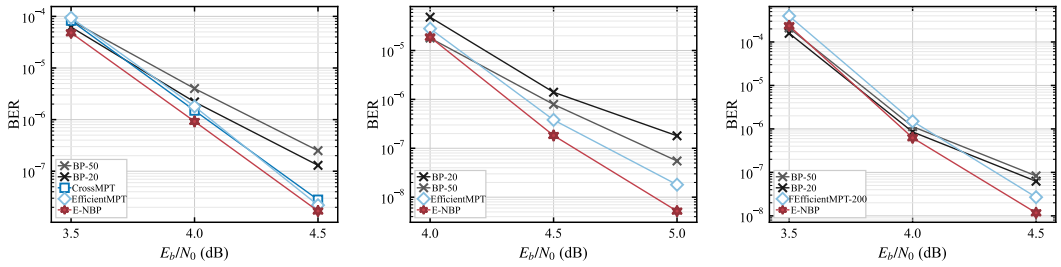


Figure 3: BER comparison for long codes: (a) WiMAX LDPC$(576, 432)$, (b) WiMAX LDPC$(1056, 880)$, and (c) WiMAX LDPC$(2304, 1920)$. FEfficientMPT-200 denotes the EfficientMPT model fine-tuned for 200 epochs (Park et al., 2026).

5.3 Complexity Analysis

At each decoding iteration, E-NBP applies the two $d \times d$ transformations $\mathbf{B}_{\mathrm{E}}^{(\ell)}$ and $\mathbf{B}_{\mathrm{S}}^{(\ell)}$ to the vector messages associated with each Tanner edge. These edge-wise matrix–vector operations result in a dominant arithmetic complexity of $\mathcal{O}\big(\ell E d^2\big)$, where E is the number of Tanner edges. For sparse code families with $E = \mathcal{O}(n)$, E-NBP therefore retains linear complexity in the block length when the message dimension and number of iterations are fixed.

Figure 4 compares the FLOPs of E-NBP and Transformer-based decoders for short codes in panel (a) and long codes in panel (b). For short codes, E-NBP reduces FLOPs by 55.3% on BCH$(63, 45)$ and 73.7% on CCSDS$(128, 64)$ relative to EfficientMPT, the most computationally efficient Transformer baseline. Compared with ECCT, E-NBP achieves FLOPs reductions ranging from 53.5% to 81.2% across the evaluated short codes. For long codes, E-NBP continues to provide substantial computational savings, reducing FLOPs by 44.9%–46.3% relative to EfficientMPT and by 75.8%–90.3% relative to ECCT. Further calculation details are provided in Appendix C.

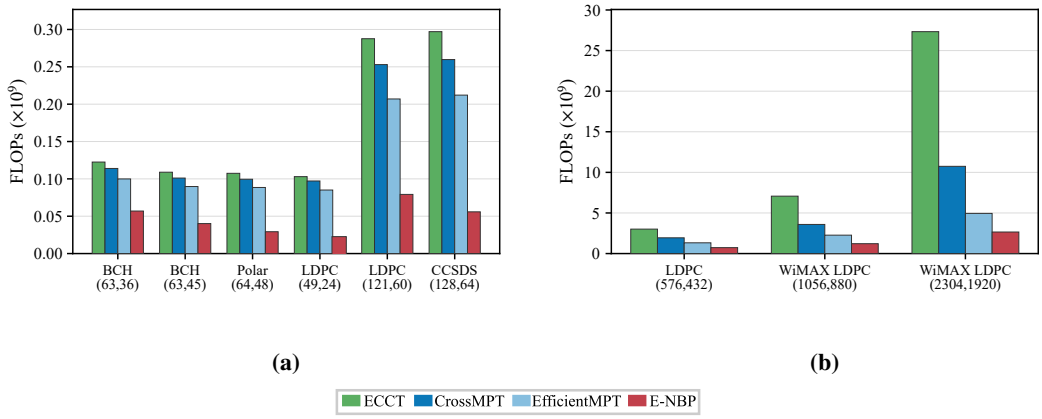


Figure 4: FLOPs comparison: (a) short codes with E-NBP $(12, 64)$ and MPT $(6, 128)$; (b) long codes with $(10, 128)$ for all models. Configurations denote (iterations/layers, dimension).

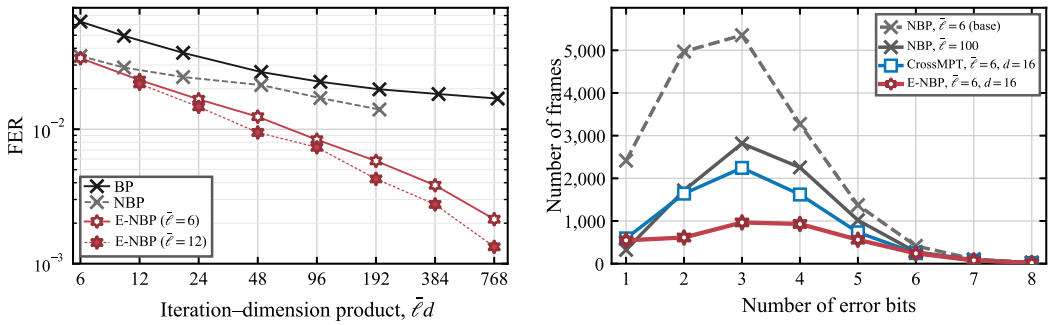


Figure 5: BCH$(63, 45)$ decoding at $E_b/N_0 = 5$ dB. (a) FER versus the iteration–dimension product for BP, NBP, and E-NBP with $\bar{\ell} \in \{6, 12\}$. (b) Residual error-bit distributions on the 17,935 frames that NBP-6 fails to decode. Only bins with one to eight residual errors are shown.

6 Analysis and Ablation Studies

6.1 Iteration–Dimension Analysis

The message-processing capacity of E-NBP can be increased along two axes: additional decoding iterations provide repeated message refinement, whereas a larger message dimension enriches the representation processed at each iteration. To examine these effects on a common structural axis, Figure 5 (a) uses the iteration–dimension product $\bar{\ell}d$. For scalar BP and NBP, $d = 1$, so this product reduces to the number of decoding iterations. We report FER on BCH$(63, 45)$ at 5 dB for BP with up to 800 iterations, NBP with up to 200 iterations, and E-NBP dimension sweeps with $\bar{\ell} \in \{6, 12\}$. NBP with 400 or more iterations could not be trained under the same hyperparameter settings due to gradient explosion.

Both additional scalar-message iterations and higher-dimensional E-NBP messages improve decoding performance. Across the shared iteration–dimension products, the E-NBP configuration with $\bar{\ell} = 12$ generally improves upon that with $\bar{\ell} = 6$, showing that the allocation between iterative refinement and message representation affects decoding performance. More importantly, both E-NBP configurations achieve lower FER than BP and NBP achieve even with substantially more iterations, indicating that vector-message processing provides a complementary source of gain beyond repeated scalar-message updates.

To examine how these performance gains translate into the recovery of decoding failures, Figure 5 (b) compares the residual-error distributions on frames that NBP with $\bar{\ell} = 6$ (denoted by NBP-6) fails

to decode. From a common test set of 500,000 received frames, we select the 17,935 frames that NBP-6 fails to decode and evaluate NBP-6, NBP-100, CrossMPT, and E-NBP. NBP-100 uses 100 iterations, whereas CrossMPT/E-NBP use six layers/iterations, respectively, with $d = 16$. On this failure set, NBP-100 recovers 52.5% of the frames, CrossMPT recovers 60.0%, and E-NBP recovers 77.9%. E-NBP achieves the highest recovery rate and, in particular, reduces the number of frames with two to five residual bit errors. Such low-weight residual errors are commonly associated with trapping-set-related decoding failures (Richardson, 2003; Kwak et al., 2025), suggesting that E-NBP is effective at resolving these persistent errors. Appendix E further examines how E-NBP resolves trapped low-weight errors.

6.2 Component Ablation

We evaluate the contributions of the multidimensional message representation and the two V2C transformations on BCH$(63, 45)$. Starting from scalar NBP, we successively introduce LLR embedding into vector messages, $\mathbf{B}_{\mathrm{E}}$ for the extrinsic message aggregate, and $\mathbf{B}_{\mathrm{S}}$ for the self-edge feedback. All variants are trained and evaluated using $\bar{\ell} = 6$, and the multidimensional configurations use $d = 128$. The embedding-only E-NBP fixes $\mathbf{B}_{\mathrm{E}} = \mathbf{I}$ and $\mathbf{B}_{\mathrm{S}} = \mathbf{0}$, so the message dimensions propagate independently after LLR embedding and are combined only by the learned readout.

Table 2: Component-wise gains on BCH$(63, 45)$. Check marks indicate enabled components, and Embedding denotes the LLR embedding into vector messages. The configuration without any enabled components corresponds to scalar NBP. Gain is the relative BER reduction from the preceding configuration at the same SNR.

Components			4 dB		5 dB		6 dB	
Embedding	$\mathbf{B}_{\mathrm{E}}$	$\mathbf{B}_{\mathrm{S}}$	BER	Gain	BER	Gain	BER	Gain
×	×	×	8.33×10^{-3}	—	1.61×10^{-3}	—	1.80×10^{-4}	—
✓	×	×	2.53×10^{-3}	69.6%	3.38×10^{-4}	79.0%	2.13×10^{-5}	88.2%
✓	✓	×	2.26×10^{-3}	10.7%	2.00×10^{-4}	40.8%	9.61×10^{-6}	54.9%
✓	✓	✓	2.09×10^{-3}	7.5%	1.88×10^{-4}	6.0%	8.78×10^{-6}	8.6%

The embedding-only configuration substantially improves upon scalar NBP, and the extrinsic transformation $\mathbf{B}_{\mathrm{E}}$ further lowers BER through cross-dimensional information exchange. The self-edge transformation $\mathbf{B}_{\mathrm{S}}$ complements this processing by further reducing the errors that remain after embedding and extrinsic transformation. At 6 dB, $\mathbf{B}_{\mathrm{S}}$ lowers BER from 9.61×10^{-6} to 8.78×10^{-6}, improving decoding performance even in this low-BER regime. The complete configuration achieves the lowest BER at all three evaluated SNRs.

7 Conclusion

BP decoders achieve strong performance with low computational complexity through iterative message passing for long and sparse codes, but their performance can be limited on short or dense codes. To address this limitation, we proposed E-NBP, which brings the embedding principle of Transformer-based decoders into BP by extending scalar messages to high-dimensional vector representations. These additional message dimensions enable richer beliefs to be formed and propagated, allowing E-NBP to surpass the performance reached by scalar BP even with substantially more iterations and to outperform Transformer-based decoders without relying on attention. At the same time, E-NBP retains BP-style message updates, resulting in substantially lower computational cost than Transformer-based decoders.

The strong performance of E-NBP across BCH, polar, and LDPC codes suggests three broader implications. First, high-dimensional message representations can extend BP toward more universal decoding across diverse code structures. Second, message dimension provides a new design axis for BP decoders beyond simply increasing the number of iterations or introducing learnable message weights. Third, strong neural decoding performance does not necessarily require attention: with sufficiently expressive high-dimensional messages, conventional BP-style updates can be enough. In this sense, we suggest that *belief propagation is all you need*.

AI USE STATEMENT

We did not use generative AI tools for any task that requires disclosure under the ICLR 2027 AI Policy for Authors. The research questions, the design of E-NBP and of the experiments, the implementation, the complexity and FLOPs analyses, and the analysis and interpretation of all results are the authors' own work and were carried out without generative AI assistance. Generative AI tools were used only to aid and polish the writing: to improve the grammar, clarity, and readability of text written by the authors, and to assist with formatting and organizing the manuscript in LaTeX. Every AI-suggested edit was reviewed by the authors before being accepted, and no experimental results, technical claims, mathematical derivations, or references were introduced by these tools. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies simulated channel decoding and does not involve human participants, personal data, or deployment decisions. Its potential benefits and risks arise from the broader use of reliable and computationally efficient communication systems.

REPRODUCIBILITY STATEMENT

The model equations specify the message updates and parameter sharing. Section 5.1 and Appendix A describe training, evaluation, and the provenance of reference results. Appendices C and D specify the complexity and graph protocols; Appendix E describes the readout analysis. We will provide the source code through an anonymized repository link during the rebuttal period.

REFERENCES

Andreas Buchberger, Christian Häger, Henry D. Pfister, Laurent Schmalen, and Alexandre Graell i Amat. Pruning and quantizing neural belief propagation decoders. *IEEE Journal on Selected Areas in Communications*, 39(7):1957–1966, 2021.

Sebastian Cammerer, Jakob Hoydis, Fayçal Aït Aoudia, and Alexander Keller. Graph neural networks for channel decoding. In *IEEE Globecom Workshops*, pp. 486–491, 2022.

Yoni Choukroun and Lior Wolf. Error correction code transformer. In *Advances in Neural Information Processing Systems*, 2022.

Robert G. Gallager. Low-density parity-check codes. *IRE Transactions on Information Theory*, 8(1):21–28, 1962. doi: 10.1109/TIT.1962.1057683.

Víctor Garcia Satorras and Max Welling. Neural enhanced belief propagation on factor graphs. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, pp. 685–693, 2021.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9, pp. 249–256, 2010.

Michael Helmling, Stefan Scholl, Florian Gensheimer, Tobias Dietz, Kira Kraft, Stefan Ruzika, and Norbert Wehn. Database of channel codes and ML simulation results, 2019. `https://www.uni-kl.de/channel-codes`.

IEEE. IEEE standard for local and metropolitan area networks—Part 16: Air interface for broadband wireless access systems. *IEEE Std 802.16-2009*, 2009. doi: 10.1109/IEEESTD.2009.5062485.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kao-Yueh Kuo and Ching-Yi Lai. Exploiting degeneracy in belief propagation decoding of quantum codes. *npj Quantum Information*, 8:111, 2022. doi: 10.1038/s41534-022-00623-2.

Hee-Youl Kwak, Dae-Young Yun, Yongjune Kim, Sang-Hyo Kim, and Jong-Seon No. Boosted neural decoders: Achieving extreme reliability of LDPC codes for 6G networks. *IEEE Journal on Selected Areas in Communications*, 43(4):1089–1102, 2025. doi: 10.1109/JSAC.2025.3531553.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

Eliya Nachmani, Yair Be'ery, and David Burshtein. Learning to decode linear codes using deep learning. In *Proceedings of the 54th Annual Allerton Conference on Communication, Control, and Computing*, pp. 341–346, 2016.

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.

Eliya Nachmani and Lior Wolf. Hyper-graph-network decoders for block codes. In *Advances in Neural Information Processing Systems*, 2019.

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

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 *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=gFvRRCnQvX.

Seong-Joon Park, Taewoo Park, Hee-Youl Kwak, Sang-Hyo Kim, Yongjune Kim, and Jong-Seon No. Efficient message-passing transformer for error correcting codes. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Xk8cwnwu2e.

Tom Richardson. Error floors of LDPC codes. In *Proceedings of the 41st Annual Allerton Conference on Communication, Control, and Computing*, pp. 1426–1435, 2003.

Thomas J. Richardson and Rüdiger L. Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Transactions on Information Theory*, 47(2):599–618, 2001.

Matthew C. Valenti and Jian Sun. The UMTS turbo code and an efficient decoder implementation suitable for software-defined radios. *International Journal of Wireless Information Networks*, 8 (4):203–215, 2001. URL https://community.wvu.edu/~mcvalenti/documents/valenti01.pdf.

Table 3: E-NBP parameterization and training settings.

Parameter	Shape	Initialization
LLR embedding $\mathbf{h}_v$	$n \times d$	$1+$ Xavier-normal
Extrinsic transformation $\mathbf{B}_{\mathrm{E}}^{(\ell)}$	$\bar{\ell} \times d \times d$	$\mathbf{I}$
Self-feedback transformation $\mathbf{B}_{\mathrm{S}}^{(\ell)}$	$\bar{\ell} \times d \times d$	$\mathbf{0}$
C2V weights $w_{c\to v,i}^{(\ell)}$	$\bar{\ell} \times E \times d$	1
Relaxation factors $g_{v\to c}^{(\ell)}$	$\bar{\ell} \times E$	1
Readout $\mathbf{r}^{(\ell)}$	$\bar{\ell} \times d$	$1/d$
Hyperparameter	**Setting**	
Iterations / dimension	Short: $(12, 64)$; long: $(10, 128)$	
Optimizer	Adam	
Training epochs	Short: 1,000; long: 50	
Batch size	128 frames	
Batches per epoch	1,000	
Learning rate: short codes	Cosine decay (Loshchilov & Hutter, 2017), $10^{-3} \to 10^{-5}$	
Learning rate: long codes	Constant, 10^{-3}	
Training E_b/N_0	3–7 dB	

A Experimental Settings and Additional Short-Code Results

Parameterization and training settings. Table 3 summarizes the trainable parameters and the settings used in the main short- and long-code performance comparisons. Both short- and long-code models are trained using randomly generated codewords. All iteration-indexed parameters are unshared across ℓ. The total parameter count is $nd + \bar{\ell}d + 2\bar{\ell}d^2 + \bar{\ell}Ed + \bar{\ell}E$. For BCH$(63, 45)$ with $E = 368$, $\bar{\ell} = 12$, and $d = 64$, this gives 390,144 trainable parameters.

The component ablation uses $\bar{\ell} = 6$ and $d = 128$ for the vector-message variants, with successive relaxation disabled for all variants. The residual-error analysis uses $(\bar{\ell}, d) = (6, 16)$ for E-NBP. The iteration–dimension study uses the configurations specified in the corresponding experiment.

Table 4: Short-code $-\ln(\mathrm{BER})$ comparison with BP-derived decoders at $E_b/N_0 = 4, 5, 6$ dB.

Family	Code	BP			Hyp-BP			AR-BP			E-NBP		
		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	$(31, 16)$	4.63	5.88	7.60	5.05	6.64	8.80	5.48	7.37	9.60	**7.04**	**9.52**	**12.44**
BCH	$(63, 36)$	4.03	5.42	7.26	4.29	5.91	8.01	4.57	6.39	8.92	**5.48**	**7.28**	**9.69**
BCH	$(63, 45)$	4.36	5.55	7.26	4.64	6.27	8.51	4.97	6.90	9.41	**6.60**	**9.32**	**12.90**
BCH	$(63, 51)$	4.50	5.82	7.42	4.80	6.44	8.58	5.17	7.16	9.53	**6.02**	**8.23**	**11.55**
Polar	$(64, 32)$	4.26	5.38	6.50	4.59	6.10	7.69	5.57	7.43	9.82	**7.65**	**10.16**	**13.25**
Polar	$(64, 48)$	4.74	5.94	7.42	4.92	6.44	8.39	5.41	7.19	9.30	**6.49**	**8.48**	**11.09**
Polar	$(128, 86)$	4.49	5.65	6.97	4.95	6.84	9.28	5.39	7.37	10.13	**7.60**	**10.83**	**14.57**
Polar	$(128, 96)$	4.61	5.79	7.08	4.94	6.76	9.09	5.27	7.44	10.20	**7.06**	**9.87**	**12.63**
LDPC	$(49, 24)$	6.23	8.19	11.72	6.23	8.54	11.95	6.58	9.39	12.39	**7.73**	**10.83**	**14.47**
LDPC	$(121, 60)$	4.82	7.21	10.87	5.22	8.29	13.00	5.22	8.31	13.07	**6.98**	**10.97**	**16.62**
LDPC	$(121, 70)$	5.88	8.76	13.04	6.39	9.81	14.04	6.45	10.01	14.77	**8.44**	**13.19**	**19.18**
LDPC	$(121, 80)$	6.66	9.82	13.98	6.95	10.68	15.80	7.22	11.03	15.90	**9.44**	**14.47**	**20.77**
MacKay	$(96, 48)$	6.84	9.40	12.57	7.19	10.02	13.16	7.43	10.65	14.65	**9.71**	**13.23**	**16.12**
CCSDS	$(128, 64)$	6.55	9.65	13.78	6.99	10.57	15.27	7.25	10.99	16.36	**10.06**	**15.18**	**21.46**
Turbo	$(132, 40)$	2.94	3.31	3.57	–	–	–	–	–	–	**7.08**	**9.90**	**13.21**

Reference results and official implementations. Tables 4 and 5 report the complete short-code results across all evaluated codes. The E-NBP and Transformer configurations follow the main short-code comparison in Section 5.1. In each table, the best and second-best results for each code and SNR are shown in bold and underlined, respectively. Higher values are better; ties at the displayed precision receive the same formatting.

BER values of BP, Hyp-BP, AR-BP, ECCT, CrossMPT, and EfficientMPT are taken from Park et al. (2025; 2026). Baseline results not reported in these sources are obtained by evaluating the respective official implementations, including those of EfficientMPT and GNN (Cammerer et al., 2022). GNN

Table 5: Short-code $-\ln(\mathrm{BER})$ comparison with GNN and Transformer-based decoders.

Family	Code	ECCT			CrossMPT			EfficientMPT			GNN			E-NBP		
		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	$(31, 16)$	6.39	8.29	10.66	6.98	9.25	12.48	7.00	9.38	**12.54**	4.89	6.46	8.40	**7.04**	**9.52**	12.44
BCH	$(63, 36)$	4.86	6.65	9.10	5.03	6.91	9.37	4.90	6.64	9.23	3.98	5.34	7.24	**5.48**	**7.28**	**9.69**
BCH	$(63, 45)$	5.60	7.79	10.93	5.90	8.20	11.62	5.73	8.05	11.11	4.51	6.04	8.15	**6.60**	**9.32**	**12.90**
BCH	$(63, 51)$	5.66	7.89	11.01	5.78	8.08	11.41	5.69	7.95	11.11	4.72	6.23	8.28	**6.02**	**8.23**	**11.55**
Polar	$(64, 32)$	6.99	9.44	12.32	7.50	9.97	13.31	7.44	10.10	**13.37**	4.62	5.82	7.10	**7.65**	**10.16**	13.25
Polar	$(64, 48)$	6.36	8.46	11.09	**6.51**	**8.70**	**11.31**	6.36	8.58	11.02	4.92	6.33	8.03	6.49	8.48	11.09
Polar	$(128, 86)$	6.31	9.01	12.45	7.51	**10.83**	**15.24**	6.44	9.32	13.07	4.57	5.96	7.47	**7.60**	**10.83**	14.57
Polar	$(128, 96)$	6.31	9.12	12.47	**7.15**	**10.15**	**13.13**	6.20	9.11	12.57	4.61	6.00	7.46	7.06	9.87	12.63
LDPC	$(49, 24)$	6.13	8.71	12.10	6.68	9.52	13.19	6.78	9.70	13.68	5.86	8.05	11.06	**7.73**	**10.83**	**14.47**
LDPC	$(121, 60)$	5.17	8.31	13.30	5.74	9.26	14.78	5.62	9.04	14.73	5.17	8.12	12.09	**6.98**	**10.97**	**16.62**
LDPC	$(121, 70)$	6.40	10.21	16.11	7.06	11.39	17.52	6.96	11.31	16.97	6.32	9.63	13.56	**8.44**	**13.19**	**19.18**
LDPC	$(121, 80)$	7.41	11.51	16.44	7.99	12.75	18.15	7.95	12.40	17.90	7.04	10.54	14.72	**9.44**	**14.47**	**20.77**
MacKay	$(96, 48)$	7.38	10.72	14.83	7.97	11.77	15.52	7.85	11.31	15.47	7.08	9.78	12.66	**9.71**	**13.23**	**16.12**
CCSDS	$(128, 64)$	6.88	10.90	15.90	7.68	11.88	17.50	7.59	11.94	17.36	6.55	9.76	13.14	**10.06**	**15.18**	**21.46**
Turbo	$(132, 40)$	4.74	6.54	9.06	5.55	7.92	10.94	4.36	6.04	8.46	2.25	2.53	2.85	**7.08**	**9.90**	**13.21**

models follow the authors' released training configurations where available and the closest available configuration otherwise, using unrolling depths of eight for BCH codes and ten for the remaining codes. For the results reported here, all GNN models use six decoding iterations at inference.

E-NBP yields lower BER than CrossMPT on multiple BCH and LDPC codes, as well as the evaluated CCSDS and Turbo codes. These results demonstrate substantial improvements over BP and competitive performance against strong Transformer-based decoders.

B Zero-Shot Robustness to Rayleigh Fading

To assess zero-shot robustness, we evaluate AWGN-trained E-NBP checkpoints directly under Rayleigh fading without channel-specific adaptation. E-NBP uses six decoding iterations and message dimension $d = 128$, matching the layer count and embedding dimension of CrossMPT. Each symbol follows $y_v = h_v(1 - 2x_v) + z_v$, where $h_v \overset{\text{i.i.d.}}{\sim} \mathrm{Rayleigh}(s = 1.0)$ is the fading coefficient and $z_v \sim \mathcal{N}(0, \sigma^2)$. The decoder receives the channel LLR obtained from the conditional Gaussian likelihood (Valenti & Sun, 2001),

$$m_v^{\mathrm{ch}} = \log \frac{p(y_v \mid x_v = 0, h_v)}{p(y_v \mid x_v = 1, h_v)} = \frac{2h_v}{\sigma^2} y_v. \tag{12}$$

Evaluation uses random codewords and the stopping rule in Section 5.1.

Table 6: Zero-shot Rayleigh-fading performance measured by $-\ln(\mathrm{BER})$. Larger values indicate lower BER. The best and second-best results are shown in bold and underlined, respectively.

Family	Code	ECCT			CrossMPT			E-NBP		
		4 dB	5 dB	6 dB	4 dB	5 dB	6 dB	4 dB	5 dB	6 dB
BCH	$(31, 16)$	5.18	6.04	6.92	5.53	6.55	7.61	**6.13**	**7.28**	**8.46**
Polar	$(64, 32)$	5.53	6.62	7.80	5.91	7.17	8.48	**6.60**	**7.77**	**9.08**
Polar	$(128, 64)$	4.31	5.37	6.63	4.70	5.93	7.34	**5.58**	**7.02**	**8.60**
Polar	$(128, 86)$	4.02	4.81	5.70	4.41	5.38	6.46	**4.59**	**5.67**	**6.89**
LDPC	$(121, 70)$	3.91	4.97	6.31	4.25	5.53	7.11	**5.63**	**7.50**	**9.74**
CCSDS	$(128, 64)$	2.46	3.97	5.79	5.25	6.94	8.92	**7.27**	**9.66**	**12.25**

Table 6 shows successful zero-shot transfer across BCH, polar, LDPC, and CCSDS codes without adapting the learned parameters to Rayleigh fading. E-NBP achieves lower BER than ECCT and CrossMPT across all evaluated codes and SNRs under Rayleigh fading without retraining. These results suggest that explicitly retaining BP message-passing operations may reduce reliance on training-channel statistics and improve robustness to channel mismatch relative to the evaluated Transformer-based decoders.

C Additional Complexity Analysis

Inference FLOPs. For all models, each multiply–accumulate, scalar arithmetic operation, and function evaluation is counted as one operation. Total counts include the remaining computations required for inference, while Table 7 summarizes the principal costs.

In E-NBP, each V2C update requires $2Ed^2$ operations for the $\mathbf{B}_{\mathrm{E}}$ and $\mathbf{B}_{\mathrm{S}}$ transformations, $4Ed$ for extrinsic aggregation and message combination, and $3Ed + E$ for SR, totaling $2Ed^2 + 7Ed + E$. Each C2V update requires $7Ed$ operations: $2Ed$ for two evaluations of Gallager's function (Gallager, 1962), $2Ed$ for extrinsic aggregation, $2Ed$ for XOR-based sign processing, and Ed for learned weighting.

Table 7: Principal FLOPs per forward pass (N: Transformer layers; $\bar{\ell}$: E-NBP iterations).

Operation	ECCT	CrossMPT	EfficientMPT	E-NBP
Embedding	$(2n-k)d$	$(2n-k)d$	$(2n-k)d$	nd
V2C update	$N\big[4(2n-k)d^2 + 2(2n-k)^2 d\big]$	$N\big[2(2n-k)d^2 + 2n(n-k)d\big]$	$N\big[(3n-k)d^2 + Ed\big]$	$\bar{\ell}\big[2Ed^2 + 7Ed + E\big]$
C2V update		$N\big[2(2n-k)d^2 + 2n(n-k)d\big]$	$N\big[(3n-2k)d^2 + Ed\big]$	$7\bar{\ell}Ed$
FFN	$8N(2n-k)d^2$	$8N(2n-k)d^2$	$8N(2n-k)d^2$	–
Readout	$(2n-k)d + n(2n-k)$	$(2n-k)d + n(2n-k)$	$Ed + 2nd$	$2nd$

For Transformer-based decoders, the principal costs arise from embedding, attention updates, FFNs, and readout. For CrossMPT and EfficientMPT, V2C and C2V denote magnitude-to-syndrome and syndrome-to-magnitude updates, respectively. EfficientMPT aggregates information through the PCM (Park et al., 2026); we count each aggregation as Ed operations using its E nonzero entries. Readout is counted only at the final iteration/layer for all models.

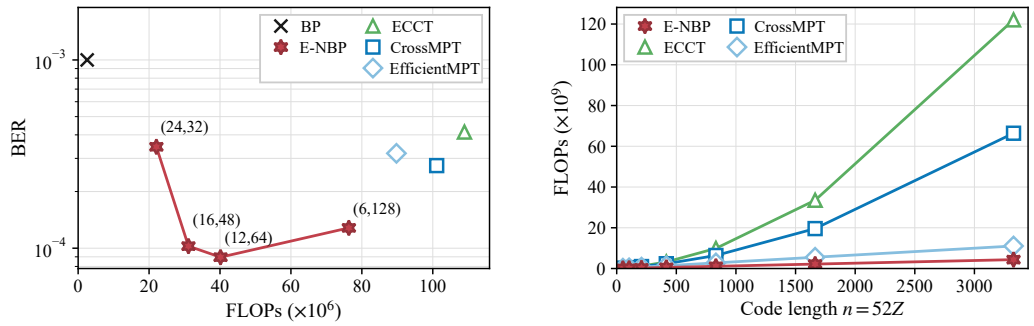
E-NBP limits its d^2-dependent computation to two V2C transformations, while EfficientMPT performs both attention projections and FFNs. These operations cost $2Ed^2$ per iteration for E-NBP and $11(2n-k)d^2$ per layer for EfficientMPT. Thus, for sufficiently sparse Tanner graphs satisfying $2E < 11(2n-k)$, E-NBP has a lower cost for these operations at the same dimension and iteration/layer count.

Table 8: FLOPs across codes. Short codes: E-NBP $(12, 64)$ and MPT $(6, 128)$; long codes: all models $(10, 128)$. Parentheses are normalized to ECCT; lower is better.

Short codes					Long codes				
Code	E-NBP	EfficientMPT	CrossMPT	ECCT	Code	E-NBP	EfficientMPT	CrossMPT	ECCT
BCH	15.3 M	50.9 M	56.8 M	58.9 M	WiMAX LDPC	0.73 G	1.33 G	1.94 G	3.01 G
$(31, 16)$	(26%)	(86%)	(96%)	(100%)	$(576, 432)$	(24%)	(44%)	(64%)	(100%)
BCH	40.1 M	89.8 M	101.1 M	108.9 M	802.11n LDPC	0.82 G	1.39 G	1.93 G	3.25 G
$(63, 45)$	(37%)	(82%)	(93%)	(100%)	$(648, 540)$	(25%)	(43%)	(60%)	(100%)
Polar	37.5 M	106.2 M	122.4 M	131.7 M	WiMAX LDPC	1.22 G	2.27 G	3.59 G	7.07 G
$(64, 32)$	(28%)	(81%)	(93%)	(100%)	$(1056, 880)$	(17%)	(32%)	(51%)	(100%)
Polar	103.9 M	188.7 M	223.1 M	256.2 M	WiMAX LDPC	2.52 G	6.35 G	23.04 G	43.21 G
$(128, 86)$	(41%)	(74%)	(87%)	(100%)	$(2304, 1152)$	(6%)	(15%)	(53%)	(100%)
LDPC	66.0 M	194.7 M	235.0 M	267.2 M	5G NR LDPC	4.36 G	11.06 G	66.40 G	122.04 G
$(121, 70)$	(25%)	(73%)	(88%)	(100%)	$(3328, 640)$	(4%)	(9%)	(54%)	(100%)

Table 8 additionally includes codes beyond the BER benchmarks to examine computational scaling; these FLOP counts are calculated analytically without model training.

Figure 6(a) compares the BER–FLOPs trade-offs of E-NBP configurations on BCH$(63, 45)$ at 5 dB. The $(12, 64)$ configuration achieves lower BER and fewer FLOPs than $(6, 128)$. Panel (b) shows the FLOP scaling of the 5G NR LDPC family with all models using $(10, 128)$; at $Z = 64$, E-NBP requires 60.6% fewer operations than EfficientMPT. Table 8 reports code-wise FLOPs, with E-NBP shaded.

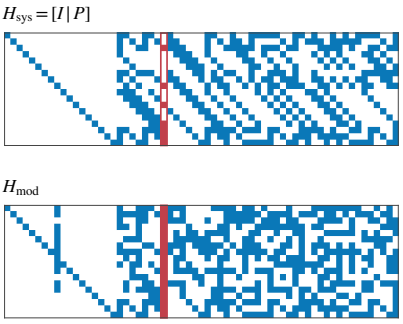


(a) **(b)**

Figure 6: (a) BER–FLOPs trade-off for BCH$(63, 45)$ at 5 dB. E-NBP labels indicate $(\bar{\ell}, d)$; the MPT baselines use $(6, 128)$ and BP uses 768 decoding iterations. (b) FLOPs for the 5G NR LDPC family, with all models using $(10, 128)$.

D Dependence on Parity-Check Representation

We compare the systematic PCM $H_{\text{sys}} = [I \mid P]$ of BCH$(63, 45)$ with a row-equivalent H_{mod} constructed to intentionally increase a target VN degree. Both PCMs have rank 18 and define the same code, but the transformation increases the edge count from 368 to 456 and the number of 4-cycles from 5,349 to 13,570. Figure 7 reports the graph statistics, $-\ln(\text{BER})$, and the percentage BER change $\Delta\%$ from H_{sys} to H_{mod}; positive values indicate degradation. For this experiment, E-NBP uses $(\bar{\ell}, d) = (6, 128)$, and BP is evaluated with 6 and 50 iterations.



Statistic	H_{sys}	H_{mod}
Maximum VN degree	11	18
Maximum CN degree	28	32
Number of edges	368	456
Number of 4-cycles	5,349	13,570

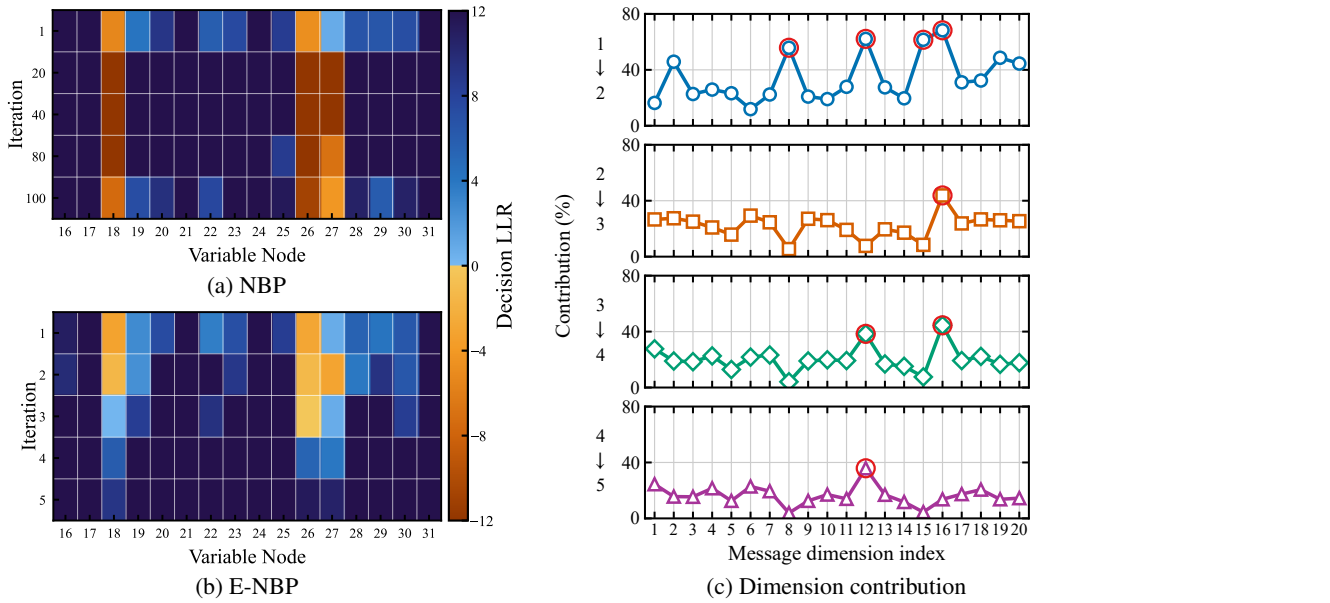
	4 dB			5 dB			6 dB		
Decoder	H_{sys}	H_{mod}	$\Delta\%$	H_{sys}	H_{mod}	$\Delta\%$	H_{sys}	H_{mod}	$\Delta\%$
BP-6	4.56	4.16	49.8	5.68	5.04	89.2	7.05	6.04	174.6
BP-50	4.77	4.21	75.6	6.25	5.10	215.7	8.32	6.28	665.1
ECCT	5.60	5.67	−7.1	7.79	7.86	−7.1	10.93	11.02	−8.8
CrossMPT	5.90	5.89	1.1	8.20	8.24	−3.8	11.62	11.49	13.5
E-NBP	6.38	5.96	52.1	8.97	8.30	94.8	12.35	11.26	197.4

Figure 7: Comparison of systematic and modified parity-check representations for BCH$(63, 45)$. The highlighted column is the target of the row-equivalent modification. Performance entries are $-\ln(\text{BER})$; $\Delta\%$ is the relative BER change from H_{sys} to H_{mod}.

At 6 dB, the modified PCM substantially degrades BP and E-NBP, whereas the Transformer changes remain within 13.5%. The modified representation therefore causes larger degradation for the evaluated BP-family decoders. Since BP-based decoders exchange messages along Tanner-graph edges, their decoding performance strongly depends on the graph structure. This highlights parity-check representation selection as an important design factor for E-NBP; optimizing it is left for future work.

E Resolving Persistent Errors and Intra-Dimensional Readout Behavior

Figure 8(a) and (b) show the bit-wise decision LLRs during decoding of a received word that NBP fails to decode but E-NBP decodes successfully. The example uses BCH$(31, 16)$ at 5 dB and displays bits 16–31. NBP uses $\bar{\ell} = 100$, whereas E-NBP uses $\bar{\ell} = 5$ and $d = 20$, giving both configurations $\bar{\ell}d = 100$. An all-zero codeword is transmitted, so a positive decision LLR indicates a correct decision for the corresponding bit.



(a) NBP (b) E-NBP (c) Dimension contribution

Figure 8: BCH$(31, 16)$ at 5 dB: decision-LLR trajectories for (a) NBP and (b) E-NBP, and (c) iteration-wise dimension contributions.

For this failure frame, some bits remain incorrect throughout NBP decoding, exhibiting persistent errors consistent with trapping-set behavior. E-NBP resolves these trapped errors within a few decoding iterations and decodes the entire codeword correctly by iteration 4.

Beyond this individual example, panel (c) aggregates corrected-bit events across frames to identify the message dimensions that contribute most frequently to error correction at each iteration transition. Decomposing the readout aggregation in Eq. (10), the contribution of dimension i to the decision LLR of bit v at iteration ℓ is

$$m_{v,i}^{(\ell)} = r_i^{(\ell)} \left(m_{v,i}^{\mathrm{ch}} + \sum_{c \in \mathcal{N}(v)} m_{c \to v,i}^{(\ell)} \right). \tag{13}$$

These contributions sum to the decision LLR, $m_v^{(\ell)} = \sum_{i=1}^{d} m_{v,i}^{(\ell)}$.

For each iteration transition, we define the escape contribution of dimension i as the percentage of corrected-bit events for which its readout contribution $m_{v,i}^{(\ell)}$ changes from strictly negative to strictly positive, after sign alignment by $(-1)^{x_v}$. A corrected-bit event denotes an incorrect bit decision that becomes correct at that transition. Multiple dimensions can contribute to the same correction, so their frequencies need not sum to 100%.

The dominant contribution pattern changes across decoding iterations. Dimensions 8, 12, 15, and 16 are prominent at the $1 \to 2$ transition. Dimension 16 contributes most frequently at the $2 \to 3$ and $3 \to 4$ transitions, whereas dimension 12 contributes most frequently at $4 \to 5$. These changing patterns indicate that different message dimensions play distinct roles in correcting errors across decoding iterations. This suggests that expanding the message dimension in E-NBP provides a direction for performance improvement distinct from increasing the number of iterations in conventional BP decoding.