# I-FGMP: FACTORIZED GATED MESSAGE PASSING FOR EFFICIENT LDPC DECODING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Decoding low-density parity-check (LDPC) codes is a fundamental task in modern communication systems. Recent neural LDPC decoders have achieved strong decoding performance, but often require relatively large models and costly inference. In particular, diffusion- and score-based decoders can improve with additional inference steps, yet this test-time scaling incurs substantial computational cost. In this work, we propose Iterative Factorized Gated Message Passing (I-FGMP), which combines neural adaptability with two established principles of conventional message-passing decoding: sparse graph-local message exchange and reuse of a common update rule across iterations. Its core FGMP operator decomposes adaptive message exchange into source-side emission modulation, graph-structured aggregation, and destination-side acceptance gating. This design provides state-dependent message control without computationally expensive pairwise interaction modeling, while repeated application of the shared operator keeps the parameter count independent of inference depth. We train I-FGMP by finite-depth weight-tied unrolling, with supervision applied to the output state after each iteration. Experiments focused primarily on short codes show that I-FGMP trained for 12 iterations outperforms recent Transformer-based decoders with less than 1/10 the parameters and at most 4/5 the floating-point operations (FLOPs) of all baselines. When trained for 30 iterations, I-FGMP continues to improve with additional test-time iterations beyond its training depth, achieving performance comparable to the best diffusion- and score-based decoders while requiring less than 1/10 the parameters and roughly 1/3 the maximum allowable test-time FLOPs of the most competitive baseline.

## 1 INTRODUCTION

Reliable and efficient decoding of low-density parity-check (LDPC) codes is essential to modern communication systems (Richardson & Urbanke, 2008). Recent neural decoders have adopted increasingly expressive modeling approaches, such as Transformer-based architectures and generative models, achieving substantial gains over conventional decoders for short- and moderate-length codes (Choukroun & Wolf, 2022; Park et al., 2025; 2026; Choukroun & Wolf, 2023; Helvits & Nachmani, 2026). Among these, diffusion- and score-based decoders offer test-time scalability by allowing additional inference steps to improve decoding performance, enabling a trained model to operate at different performance–computation trade-offs (Choukroun & Wolf, 2023; Helvits & Nachmani, 2026). Despite these performance gains, such neural decoders generally introduce substantial parameter overhead and require substantially more computation than conventional decoders.

Conventional message-passing decoders offer a complementary set of strengths. Algorithms such as belief propagation (BP) (MacKay & Neal, 1996) and min-sum (MS) variants (Fossorier et al., 1999; Chen & Fossorier, 2002; Chen et al., 2005) use fixed, non-learned update rules, require low computation, and can improve decoding performance with additional iterations. Their low computational cost and test-time scalability follow from two shared computational principles. First, messages are exchanged only along the sparse connections of the Tanner graph (Tanner, 1981), allowing the code structure to guide inference. Second, the same update mechanism is reused across iterations, making the update rule independent of the number of iterations performed at test time. Sparse connectivity keeps each iteration inexpensive, while update reuse enables the inference depth to be adjusted without changing the underlying update mechanism. Together, these principles provide test-time



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.

scalability at low incremental computational cost. The limitation is that handcrafted updates provide substantially less representational and adaptive capacity than learned neural operators.

These observations highlight two complementary aspects of decoder design: learned neural operators provide strong representational and adaptive capacity, while iterative message passing offers an efficient organization of computation. Building on these established principles, we propose the Factorized Gated Message Passing (FGMP) operator, which factorizes adaptive message exchange into source-side emission modulation, fixed sparse graph aggregation, and destination-side acceptance gating, thereby enabling adaptive source- and destination-conditioned message control without costly pairwise interaction modeling. Repeated application of the FGMP operator, with parameters shared across iterations, yields the Iterative FGMP (I-FGMP) decoder. Its parameter count is independent of the iteration budget, allowing additional inference iterations without introducing additional trainable parameters.

To enable single-codeword training, we adopt the established magnitude-and-syndrome preprocessing proposed by Bennatan et al. (2018). To provide an intuitive view of these quantities and their interactions, we introduce a corresponding factor-graph representation, over which I-FGMP performs iterative message passing. We train the shared FGMP operator through finite-depth unrolling, with supervision applied along the resulting iterative trajectory.

We evaluate the parameter and computational efficiency of I-FGMP, together with its test-time scalability, using two independently trained configurations at computational budgets comparable to those of different baseline groups. I-FGMP-12 is trained by unrolling 12 iterations and compared with ECCT (Choukroun & Wolf, 2022), CrossMPT (Park et al., 2025), and EfficientMPT (Park et al., 2026). It outperforms the best baseline while using less than 1/10 the parameters and at most 4/5 the floating-point operations (FLOPs) of all baselines. I-FGMP-Scale is separately trained by unrolling 30 iterations and evaluated for up to 60 iterations at test time. Compared with the diffusion-based DDECC (Choukroun & Wolf, 2023) and score-based SB-ECC (Helvits & Nachmani, 2026), which also support test-time scalability, I-FGMP-Scale achieves performance comparable to the best diffusion- and score-based decoders while requiring less than 1/10 the parameters and roughly 1/3 the maximum allowable test-time FLOPs of the most competitive baseline. In particular, its decoding performance continues to improve with additional inference iterations beyond its training depth.

## 2 Related Work

### 2.1 Neural Decoders

Neural decoders are commonly divided into model-based and model-free approaches (Choukroun & Wolf, 2022). Model-based decoders introduce learnable parameters into conventional algorithms such as BP and MS decoding, often by unfolding a fixed number of iterations into a trainable network (Lugosch & Gross, 2017; Nachmani et al., 2018; Nachmani & Wolf, 2019; Dai et al., 2021; Buchberger et al., 2021; Kwak et al., 2023; 2025). This approach preserves the inductive biases of the underlying algorithm, but also constrains the learned model to the predefined structure of the underlying algorithm. Model-free decoders instead learn the decoding mapping using more expressive neural architectures, providing greater flexibility in representation and modeling.

Early model-free decoders can suffer from severe overfitting because training must cover an exponentially large codeword space (Gruber et al., 2017; Cammerer et al., 2017; Kim et al., 2018). Magnitude-and-syndrome preprocessing alleviates this issue by enabling single-codeword training (Bennatan et al., 2018) and has since been widely adopted by subsequent neural decoders (Choukroun & Wolf, 2022; 2023; Park et al., 2025; Lau et al., 2025). Building on this preprocessing, recent model-free decoders have employed increasingly expressive architectures while exploiting code structure to improve decoding performance. ECCT combines magnitudes and syndromes using self-attention masked by the parity-check matrix (Choukroun & Wolf, 2022); CrossMPT alternately updates their separate representations through masked cross-attention (Park et al., 2025). EfficientMPT introduces efficient error-correcting (EEC) attention, using a global query vector and element-wise query–key interactions, together with parity-check-matrix-based transformations between the magnitude and syndrome domains (Park et al., 2026). FGMP instead factorizes adaptive

interaction into node-wise message control and fixed graph-structured propagation, thereby separating learned message processing from topology-defined information exchange.

Diffusion- and score-based decoders provide another direction for model-free decoding by formulating inference as a sequence of denoising steps (Choukroun & Wolf, 2023; Helvits & Nachmani, 2026). Their inference budgets can be adjusted at test time, allowing additional computation to improve decoding performance. However, each additional step requires another evaluation of the underlying neural model, leading to substantial total inference cost. Repeated application of FGMP similarly enables I-FGMP to progressively refine its decoding result. However, each FGMP iteration provides a lightweight refinement, enabling fine-grained test-time scaling at low incremental computational cost.

### 2.2 Graph Neural Networks

Graph neural networks (GNNs) commonly learn representations on graph-structured data by transforming and aggregating information from neighboring nodes (Wu et al., 2021). Representative feedforward architectures include GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), and GAT (Veličković et al., 2018), which typically stack message-passing layers with layer-specific parameters. Recurrent GNNs (Gori et al., 2005; Scarselli et al., 2009) instead reuse a shared update operator across propagation steps, while Gated Graph Sequence Neural Networks (Li et al., 2016) further introduce recurrent gates to control node-state updates. For channel decoding, Cammerer et al. (2022) learn node and directed-edge updates on Tanner graphs, whereas Tian et al. (2023) use a shared network to assign reliability-dependent weights to BP-style edge messages. In contrast, I-FGMP factorizes adaptation into node-wise emission modulation and acceptance gating around fixed sparse aggregation, avoiding separate neural evaluations on individual edges.

## 3 Preliminaries

### 3.1 Low-Density Parity-Check Codes and Tanner Graphs

An $(n, k)$ LDPC code is specified by a parity-check matrix $H \in \{0,1\}^{(n-k)\times n}$, where $n$ and $k$ denote the code length and the number of information bits, respectively. A binary vector $\mathbf{x}^b \in \{0,1\}^n$ is a valid codeword if

$$H\mathbf{x}^b = \mathbf{0} \pmod 2. \tag{1}$$

The parity-check matrix induces the Tanner graph (Tanner, 1981), a bipartite factor graph with variable nodes representing code bits and check nodes representing parity constraints. Variable node $i$ is connected to check node $j$ if and only if $H_{ji} = 1$. In BP and MS decoding, variable and check nodes iteratively exchange messages along the graph edges to refine estimates of the transmitted bits (Richardson & Urbanke, 2008; Ryan & Lin, 2009).

### 3.2 Syndrome-Based Decoding

We assume uniformly distributed codewords, binary phase-shift keying (BPSK), and a memoryless binary-input symmetric-output (BISO) channel with input alphabet $\{-1, 1\}$ and real-valued output alphabet $\mathcal{Y}$. Channel outputs are assumed nonzero almost surely, as is the case for binary-input AWGN channels. Denoting the channel's conditional probability mass function or density by $p_{Y|X}$, output symmetry requires

$$p_{Y|X}(y \mid 1) = p_{Y|X}(-y \mid -1), \qquad \forall y \in \mathcal{Y}. \tag{2}$$

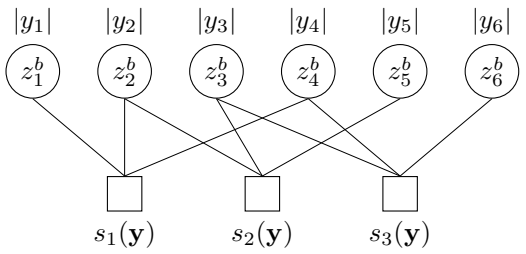
Such a channel admits the equivalent multiplicative-noise representation

$$Y = XZ, \tag{3}$$

where the multiplicative noise $Z \sim p_{Y|X}(\cdot \mid 1)$ is independent of the BPSK symbol $X$. Applying $\mathrm{bipolar}(u) = 1 - 2u$ element-wise to $\mathbf{x}^b$ gives $\mathbf{x} = \mathrm{bipolar}(\mathbf{x}^b)$; hereafter, scalar functions applied to vectors or matrices are understood element-wise. Memorylessness then gives the vector model $\mathbf{y} = \mathbf{x} \odot \mathbf{z}$, where $\odot$ denotes the Hadamard (element-wise) product.

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

(a) Parity-check matrix.




(b) Syndrome-based factor graph.

Figure 1: A parity-check matrix and the corresponding syndrome-based factor graph for estimating $\mathbf{z}^b$ from $|\mathbf{y}|$ and $s(\mathbf{y})$.

The observation syndrome $s(\mathbf{y}) \in \{0,1\}^{n-k}$ is defined by

$$s(\mathbf{y}) \triangleq H\mathbf{y}^b \pmod 2. \tag{4}$$

Here, $\mathbf{y}^b = \mathrm{bin}(\mathrm{sign}(\mathbf{y}))$, where $\mathrm{sign}(u) = 1$ for $u \geq 0$ and $-1$ otherwise, and $\mathrm{bin}(u) = (1-u)/2$. Define $\mathbf{z}^b = \mathrm{bin}(\mathrm{sign}(\mathbf{z}))$ similarly. Under the assumptions above, $(|\mathbf{y}|, s(\mathbf{y}))$ is sufficient for estimating the multiplicative noise $\mathbf{z}$ (Bennatan et al., 2018). We call its computation magnitude-and-syndrome preprocessing. This result leads to an optimality-preserving two-step estimation strategy: first estimate the multiplicative noise $\mathbf{z}$ from $(|\mathbf{y}|, s(\mathbf{y}))$, and then reconstruct the transmitted codeword from the resulting estimate $\hat{\mathbf{z}}$ as

$$\hat{\mathbf{x}}^b = \mathbf{y}^b + \hat{\mathbf{z}}^b \pmod 2 \tag{5}$$

$$= \mathrm{bin}(\mathrm{sign}(\mathbf{y}) \odot \mathrm{sign}(\hat{\mathbf{z}}))\,. \tag{6}$$

With an appropriate noise estimator, this two-step procedure can achieve bitwise maximum a posteriori (MAP) decoding without intrinsic loss of optimality (Bennatan et al., 2018). Our I-FGMP decoder adopts the same estimation strategy.

The distribution of $(|\mathbf{y}|, s(\mathbf{y}))$ is invariant to the transmitted codeword since, for any valid $\mathbf{x}^b$,

$$|\mathbf{y}| = |\mathbf{z}|, \qquad s(\mathbf{y}) = H(\mathbf{x}^b + \mathbf{z}^b) = H\mathbf{z}^b \pmod 2, \tag{7}$$

where the last equality follows from (1). Thus, the joint distribution of the sufficient statistic and multiplicative noise is codeword-invariant. The noise estimator can therefore be trained on samples generated from any fixed codeword, including the all-zero codeword, without sampling the full codebook.

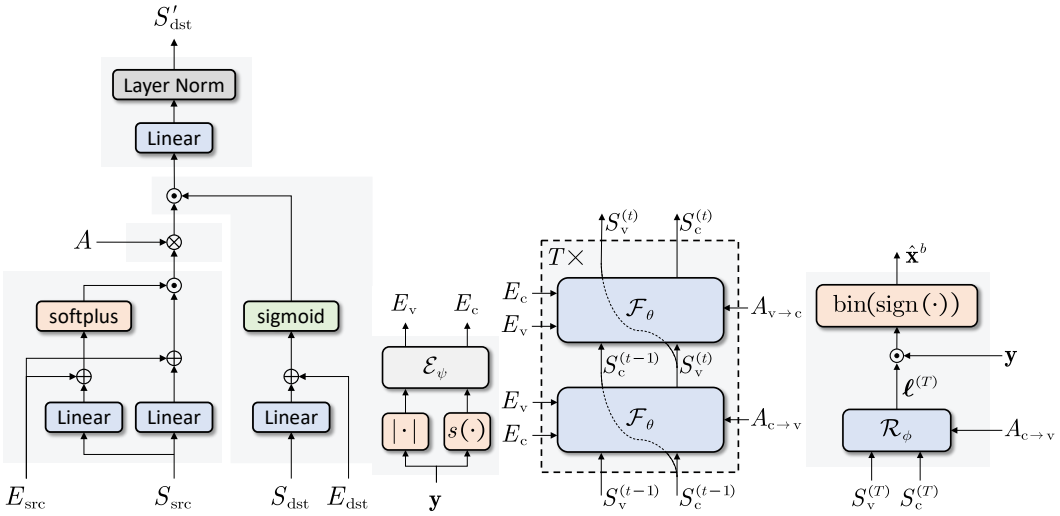
### 3.3 Syndrome-Based Factor Graph

A conventional Tanner graph represents the posterior factorization for directly estimating $\mathbf{x}^b$. In contrast, syndrome-based decoding estimates $\mathbf{z}$ from $(|\mathbf{y}|, s(\mathbf{y}))$. Since $|\mathbf{z}| = |\mathbf{y}|$, this reduces to estimating the binary multiplicative noise $\mathbf{z}^b$. Appendix A derives the posterior factorization of $\Pr(\mathbf{z}^b \mid |\mathbf{y}|, s(\mathbf{y}))$ for this noise-estimation problem.

Figure 1 illustrates this factorization for an example parity-check matrix $H$. Each $z_i^b$ is represented by a variable node, $|\mathbf{y}|$ determines the local evidence factors, and $s(\mathbf{y})$ determines the parity-check factors. As in a conventional Tanner graph, $H$ specifies the sparse bipartite connectivity. I-FGMP then performs message passing on this graph.

## 4 Iterative Factorized Gated Message Passing

Building on the syndrome-based factor graph introduced above, we first specify the FGMP operator for directed information exchange between two node sets and then construct I-FGMP by applying this operator alternately in both directions of the graph.




(a) FGMP operator $\mathcal{F}_\theta$. (b) I-FGMP decoder.

Figure 2: FGMP operator and I-FGMP decoder. (a) Factorized gated message passing for a directed node-state update. (b) The I-FGMP decoding pipeline, comprising evidence embedding, alternating check-to-variable and variable-to-check updates, and readout with hard decision for codeword reconstruction.

### 4.1 Factorized Gated Message Passing Operator

Consider a directed update from $N_{\mathrm{src}}$ source nodes to $N_{\mathrm{dst}}$ destination nodes. In I-FGMP, check nodes serve as sources and variable nodes as destinations for the check-to-variable update, with these roles reversed for the variable-to-check update. Let $E_{\mathrm{src}} \in \mathbb{R}^{N_{\mathrm{src}}\times d}$ and $E_{\mathrm{dst}} \in \mathbb{R}^{N_{\mathrm{dst}}\times d}$ denote their fixed evidence embeddings, and let $S_{\mathrm{src}} \in \mathbb{R}^{N_{\mathrm{src}}\times d}$ and $S_{\mathrm{dst}} \in \mathbb{R}^{N_{\mathrm{dst}}\times d}$ denote their current node states, where $d$ is the hidden dimension.

Let $A \in \mathbb{R}^{N_{\mathrm{dst}}\times N_{\mathrm{src}}}$ denote a fixed, non-learnable, nonnegative row-normalized aggregation matrix. Its entry $A_{ij}$ specifies the contribution of source node $j$ to destination node $i$, with $A_{ij} = 0$ when the two nodes are not connected. For LDPC decoding, $A$ assigns equal weights to connected source nodes in each row, as detailed in the next subsection. The matrix $A$ remains unchanged across iterations, as do the evidence embeddings; only the node states evolve.

FGMP updates the destination states as

$$\begin{aligned} S'_{\mathrm{dst}} &= \mathcal{F}_\theta(S_{\mathrm{dst}}, S_{\mathrm{src}}; E_{\mathrm{dst}}, E_{\mathrm{src}}, A) \\ &\triangleq \mathrm{LN}(\{G \odot [A(M \odot P)]\}\, W_o)\,, \end{aligned} \tag{8}$$

where

$$\begin{aligned} P &= S_{\mathrm{src}}U_p + E_{\mathrm{src}}, \\ M &= \mathrm{softplus}(S_{\mathrm{src}}U_m + E_{\mathrm{src}}), \\ G &= \sigma(S_{\mathrm{dst}}U_g + E_{\mathrm{dst}}). \end{aligned} \tag{9}$$

The functions $\sigma(\cdot)$, $\mathrm{softplus}(\cdot)$, and $\mathrm{LN}(\cdot)$ denote sigmoid activation, softplus activation, and layer normalization, respectively. The matrices $U_p, U_m, U_g, W_o \in \mathbb{R}^{d\times d}$ are trainable. Layer normalization has trainable scale and bias vectors $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^d$, and $\theta \triangleq \{U_p, U_m, U_g, W_o, \boldsymbol{\gamma}, \boldsymbol{\beta}\}$.

Figure 2(a) illustrates the three stages. At each source node, $P$ combines the current state with fixed evidence, and the positive modulator $M$ scales the emitted representation. The matrix $A$ then aggregates these representations over graph neighbors. At each destination node, $G$ gates the aggregate using the destination state and evidence. To examine message passing over the $d$-dimensional feature space, consider feature channel $r$ and let $\mathbf{p}_r = P_{:,r}$, $\mathbf{m}_r = M_{:,r}$, and $\mathbf{g}_r = G_{:,r}$. Before the output projection,

$$\big[G \odot A(M \odot P)\big]_{:,r} = \mathrm{diag}(\mathbf{g}_r) A\, \mathrm{diag}(\mathbf{m}_r)\mathbf{p}_r. \tag{10}$$

Thus, the effective weight from source $j$ to destination $i$ in channel $r$ is $G_{ir}A_{ij}M_{jr}$. The endpoint-dependent factors adapt message exchange, while $A$ fixes the graph connectivity. Finally, the projection $W_o$ mixes the $d$ feature channels, and layer normalization produces the updated destination states.

### 4.2 Iterative FGMP Decoder

We next instantiate FGMP on the syndrome-based factor graph. Figure 2(b) summarizes the resulting decoding pipeline, from evidence embedding through alternating FGMP updates to readout and codeword reconstruction. The embedding operator maps the preprocessed observation to fixed variable- and check-node evidence embeddings, following the code-specific embedding design (Choukroun & Wolf, 2022; Park et al., 2025; Choukroun & Wolf, 2023; Helvits & Nachmani, 2026):

$$(E_{\mathrm{v}}, E_{\mathrm{c}}) = \mathcal{E}_\psi(|\mathbf{y}|, s(\mathbf{y})) \triangleq (\mathrm{diag}(|\mathbf{y}|)W_v, \mathrm{diag}\left(\mathrm{bipolar}\left(s(\mathbf{y})\right)\right) W_c), \tag{11}$$

where $W_v \in \mathbb{R}^{n\times d}$ and $W_c \in \mathbb{R}^{(n-k)\times d}$ are trainable and $\psi \triangleq \{W_v, W_c\}$. Define the diagonal degree matrices

$$D_{\mathrm{c}} \triangleq \mathrm{diag}(H\mathbf{1}_n), \;\; D_{\mathrm{v}} \triangleq \mathrm{diag}(H^\top \mathbf{1}_{n-k}). \tag{12}$$

Assuming all graph nodes have positive degree, the directed aggregation matrices are

$$A_{\mathrm{v}\to\mathrm{c}} \triangleq D_{\mathrm{c}}^{-1}H, \;\; A_{\mathrm{c}\to\mathrm{v}} \triangleq D_{\mathrm{v}}^{-1}H^\top. \tag{13}$$

Thus, each row averages the source-node representations over the corresponding graph neighborhood.

Starting from $(S_{\mathrm{v}}^{(0)}, S_{\mathrm{c}}^{(0)}) = (\mathbf{0}_{n\times d}, \mathbf{0}_{(n-k)\times d})$, one I-FGMP iteration consists of a check-to-variable update followed by a variable-to-check update:

$$\begin{aligned} S_{\mathrm{v}}^{(t)} &= \mathcal{F}_\theta\left(S_{\mathrm{v}}^{(t-1)}, S_{\mathrm{c}}^{(t-1)}; E_{\mathrm{v}}, E_{\mathrm{c}}, A_{\mathrm{c}\to\mathrm{v}}\right), \\ S_{\mathrm{c}}^{(t)} &= \mathcal{F}_\theta\left(S_{\mathrm{c}}^{(t-1)}, S_{\mathrm{v}}^{(t)}; E_{\mathrm{c}}, E_{\mathrm{v}}, A_{\mathrm{v}\to\mathrm{c}}\right), \qquad t = 1, \ldots, T. \end{aligned} \tag{14}$$

The same parameters $\theta$ are shared across both directions and all iterations. As detailed in Appendix C, untying parameters improves performance but increases model size, particularly across iterations. We retain both forms of sharing for parameter efficiency.

After iteration $t$, the readout combines variable-node states with aggregated check-node states to produce logits for the events $z_i^b = 0$:

$$\boldsymbol{\ell}^{(t)} = \mathcal{R}_\phi(S_{\mathrm{v}}^{(t)}, S_{\mathrm{c}}^{(t)}; A_{\mathrm{c}\to\mathrm{v}}) \triangleq S_{\mathrm{v}}^{(t)}\mathbf{w}_v + A_{\mathrm{c}\to\mathrm{v}}S_{\mathrm{c}}^{(t)}\mathbf{w}_c, \tag{15}$$

where $\mathbf{w}_v, \mathbf{w}_c \in \mathbb{R}^d$ are trainable vectors and $\phi \triangleq \{\mathbf{w}_v, \mathbf{w}_c\}$. At the final iteration, the binary-noise and codeword estimates are

$$\begin{aligned} \hat{\mathbf{z}}^{b,(T)} &= \mathrm{bin}(\mathrm{sign}(\boldsymbol{\ell}^{(T)})), \\ \hat{\mathbf{x}}^b &= \mathbf{y}^b + \hat{\mathbf{z}}^{b,(T)} \pmod 2 = \mathrm{bin}\Big(\mathrm{sign}(\mathbf{y}) \odot \mathrm{sign}(\boldsymbol{\ell}^{(T)})\Big). \end{aligned} \tag{16}$$

Both forms of $\hat{\mathbf{x}}^b$ agree under the convention $\mathrm{sign}(0) = 1$, including when a logit is zero. This decision rule implements the two-step estimation strategy introduced in Subsection 3.2.

### 4.3 Training

We train I-FGMP by unrolling (14) for a finite depth $T$ and backpropagating through the resulting trajectory.

For a training sample $(\mathbf{x}, \mathbf{y})$, let $\mathbf{z}^b = \mathrm{bin}(\mathrm{sign}(\mathbf{x} \odot \mathbf{y}))$ be the target binary multiplicative noise, and let $\mathbf{q}^{(t)} = \sigma(\boldsymbol{\ell}^{(t)})$, where $q_i^{(t)}$ represents the estimated probability that $z_i^b = 0$. We supervise the readout after every iteration using

$$\mathcal{L}(\theta, \psi, \phi; \mathbf{x}, \mathbf{y}) = -\frac{1}{nT}\sum_{t=1}^{T}\sum_{i=1}^{n}\Big[(1 - z_i^b)\log q_i^{(t)} + z_i^b \log(1 - q_i^{(t)})\Big]. \tag{17}$$

Table 1: Comparison with fixed-computation decoders: ECCT (Choukroun & Wolf, 2022), CrossMPT (Park et al., 2025), and EfficientMPT (Park et al., 2026). I-FGMP-12 is trained and evaluated for 12 iterations. BER is computed from the final-iteration output. For I-FGMP-12, the reported FLOPs correspond to the complete 12-iteration inference.

(a) $-\ln(\mathrm{BER})$ at $E_b/N_0 \in \{4, 5, 6\}$ dB (higher is better). Best and second-best results are **bold** and underlined, respectively; dashes denote unavailable results.

| Code | $(n, k)$ | ECCT | | | CrossMPT | | | EfficientMPT | | | I-FGMP-12 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 4 | 5 | 6 | 4 | 5 | 6 | 4 | 5 | 6 | 4 | 5 | 6 |
| LDPC | $(49, 24)$ | 6.13 | 8.71 | 12.10 | 6.68 | 9.52 | 13.19 | – | – | – | **6.85** | **9.86** | **13.65** |
| | $(121, 60)$ | 5.17 | 8.31 | 13.30 | 5.74 | 9.26 | 14.78 | 5.62 | 9.04 | 14.73 | **6.02** | **9.85** | **15.22** |
| | $(121, 70)$ | 6.40 | 10.21 | 16.11 | 7.06 | 11.39 | 17.52 | 6.96 | 11.31 | 16.97 | **7.48** | **11.99** | **17.78** |
| | $(121, 80)$ | 7.41 | 11.51 | 16.44 | 7.99 | 12.75 | 18.15 | 7.95 | 12.40 | 17.90 | **8.46** | **13.03** | **18.44** |
| LDPC (MacKay) | $(96, 48)$ | 7.38 | 10.72 | 14.83 | 7.97 | 11.77 | 15.52 | – | – | – | **8.48** | **11.99** | **15.64** |
| LDPC (CCSDS) | $(128, 64)$ | 6.88 | 10.90 | 15.90 | 7.68 | 11.88 | 17.50 | – | – | – | **8.26** | **12.97** | **18.20** |

(b) FLOPs per decoded codeword and total trainable parameters. M and K denote million and thousand, respectively; lowest and second-lowest values are **bold** and underlined, respectively.

| Code | $(n, k)$ | ECCT | | CrossMPT | | EfficientMPT | | I-FGMP-12 | |
|---|---|---|---|---|---|---|---|---|---|
| | | FLOPs | Params | FLOPs | Params | FLOPs | Params | FLOPs | Params |
| LDPC | $(49, 24)$ | 95.7 M | 1.20 M | 91.1 M | 1.20 M | 81.9 M | 1.09 M | **62.0 M** | **75.5 K** |
| | $(121, 60)$ | 266 M | 1.24 M | 237 M | 1.24 M | 208 M | 1.09 M | **166 M** | **89.3 K** |
| | $(121, 70)$ | 248 M | 1.23 M | 222 M | 1.23 M | 195 M | 1.09 M | **154 M** | **88.1 K** |
| | $(121, 80)$ | 231 M | 1.23 M | 206 M | 1.23 M | 183 M | 1.09 M | **143 M** | **86.8 K** |
| LDPC (MacKay) | $(96, 48)$ | 201 M | 1.22 M | 184 M | 1.22 M | 163 M | 1.09 M | **127 M** | **84.5 K** |
| LDPC (CCSDS) | $(128, 64)$ | 283 M | 1.24 M | 252 M | 1.24 M | 220 M | 1.09 M | **176 M** | **90.6 K** |

As shown in Subsection 3.2, the joint distribution of the sufficient statistic and the multiplicative noise is independent of the transmitted codeword. We may therefore fix $\mathbf{x}$ during training to the BPSK image of the all-zero codeword, i.e., $\mathbf{x} = \mathbf{1}_n$. Retaining only the term at $t = T$ in (17) gives supervision of the final output alone. As detailed in Appendix C, full-trajectory supervision consistently improves performance over final-only supervision.

## 5 Experimental Results

### 5.1 Experimental Protocol

**Training and inference.** We consider BPSK transmission over an additive white Gaussian noise (AWGN) channel, with channel inputs fixed in $\{-1, +1\}$ and $E_b/N_0$ controlled through the AWGN variance. Our main experiments focus on six short LDPC codes for which the papers introducing the baseline methods report BER results, enabling direct comparison; additional results for medium- and long-length codes are reported in Appendix E. All parity-check matrices used in our experiments are obtained from the website provided by Helmling et al. (2025). All neural decoders in the comparisons use hidden dimension $d = 128$. As justified in Subsection 4.3, all I-FGMP models are trained using the BPSK image of the all-zero codeword. Each I-FGMP variant is trained for 1000 epochs of 1000 mini-batches with batch size 128. For each training sample, $E_b/N_0$ is drawn uniformly from $\{3, 4, 5, 6, 7\}$ dB. We use Adam with a cosine learning-rate schedule from $10^{-4}$ to $5 \times 10^{-7}$. I-FGMP-12 uses 12 iterations during both training and inference. I-FGMP-Scale is trained for 30

Table 2: Comparison with test-time-scalable diffusion- and score-based decoders: DDECC (Choukroun & Wolf, 2023) and SB-ECC (Helvits & Nachmani, 2026). All methods stop at the first iterate satisfying $H\hat{\mathbf{x}}^b = 0 \pmod 2$. I-FGMP-Scale is trained for 30 iterations and allows at most 60 iterations at test time. Reported FLOPs correspond to the maximum number of test-time iterations allowed for each method.

(a) $-\ln(\mathrm{BER})$ at $E_b/N_0 \in \{4, 5, 6\}$ dB (higher is better). Best and second-best results are **bold** and underlined, respectively.

| Code | $(n, k)$ | DDECC | | | SB-ECC | | | I-FGMP-Scale | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 4 | 5 | 6 | 4 | 5 | 6 | 4 | 5 | 6 |
| LDPC | $(49, 24)$ | 6.01 | 8.74 | 11.92 | **7.74** | **10.88** | **14.63** | 7.25 | 10.35 | 14.03 |
| | $(121, 60)$ | 5.38 | 8.73 | 14.17 | 6.33 | 10.38 | 16.38 | **6.53** | **10.63** | **16.54** |
| | $(121, 70)$ | 6.79 | 11.13 | 16.93 | 7.75 | 12.69 | **19.24** | **8.11** | **12.92** | 18.80 |
| | $(121, 80)$ | 7.71 | 12.17 | 17.55 | 8.84 | 14.02 | **20.42** | **9.12** | **14.12** | 19.47 |
| LDPC (MacKay) | $(96, 48)$ | 8.12 | 11.88 | 15.93 | 8.97 | 12.82 | 16.25 | **9.33** | **13.02** | **16.26** |
| LDPC (CCSDS) | $(128, 64)$ | 7.81 | 12.48 | 17.66 | 8.60 | 13.43 | 19.20 | **9.24** | **14.48** | **19.38** |

(b) Maximum test-time FLOPs per decoded codeword and total trainable parameters. G, M, and K denote billion, million, and thousand, respectively; lowest and second-lowest values are **bold** and underlined, respectively.

| Code | $(n, k)$ | DDECC | | SB-ECC | | I-FGMP-Scale | |
|---|---|---|---|---|---|---|---|
| | | FLOPs | Params | FLOPs | Params | FLOPs | Params |
| LDPC | $(49, 24)$ | 2.39 G | 1.20 M | 911 M | 1.20 M | **310 M** | **75.5 K** |
| | $(121, 60)$ | 16.2 G | 1.24 M | 2.37 G | 1.24 M | **829 M** | **89.3 K** |
| | $(121, 70)$ | 12.6 G | 1.23 M | 2.22 G | 1.23 M | **771 M** | **88.1 K** |
| | $(121, 80)$ | 9.47 G | 1.23 M | 2.06 G | 1.23 M | **713 M** | **86.8 K** |
| LDPC (MacKay) | $(96, 48)$ | 9.65 G | 1.22 M | 1.84 G | 1.22 M | **637 M** | **84.5 K** |
| LDPC (CCSDS) | $(128, 64)$ | 18.1 G | 1.24 M | 2.52 G | 1.24 M | **881 M** | **90.6 K** |

iterations, evaluated with a 60-iteration cap in the baseline comparison, and studied with budgets up to 200 iterations.

**BER estimation.** For each I-FGMP operating point, we test at least $10^5$ frames and then continue until 100 frame errors are observed or $10^9$ frames have been tested. At $E_b/N_0 = 6$ dB, we sample from a $5.5$ dB proposal distribution and estimate BER using standard importance sampling with target-to-proposal density-ratio weights. In this case, all frame and frame-error counts above are measured under the proposal distribution.

**FLOPs and parameters.** For I-FGMP, the dense-matrix FLOPs per codeword are $T[4(2n-k)d^2 + 2n(n-k)d]$, counting each multiply-add as one operation; baseline FLOPs use the same convention. This count includes feature transformations and two $H$-derived aggregations per iteration, but excludes lower-order element-wise operations, activations, normalization, embedding, readout, and early stopping. The trainable-parameter count is $4d^2 + (2n - k + 4)d$, comprising four FGMP transformations, evidence embeddings, layer normalization, and readout; sharing across directions and iterations makes it independent of $T$.

### 5.2 Accuracy, Efficiency, and Test-Time Scaling

Table 1 compares I-FGMP-12 with three fixed-computation decoders, ECCT (Choukroun & Wolf, 2022), CrossMPT (Park et al., 2025), and EfficientMPT (Park et al., 2026). I-FGMP-12 achieves

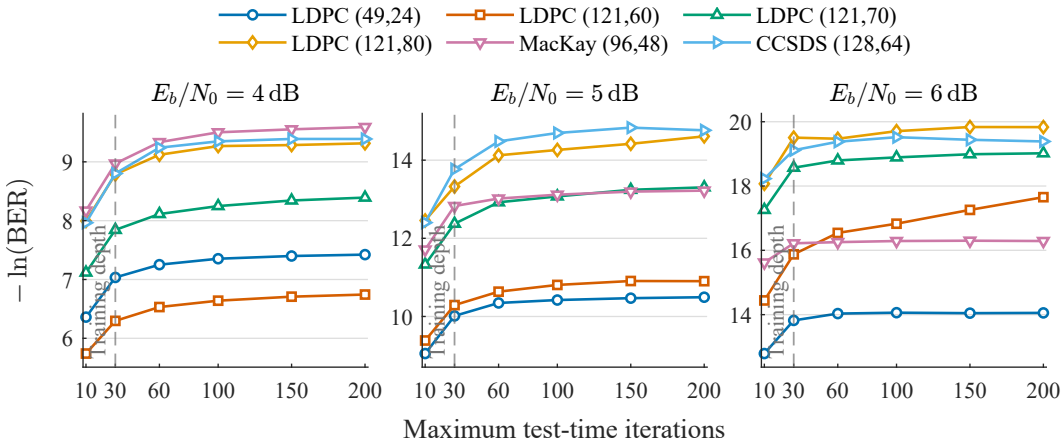



Figure 3: Test-time scaling of I-FGMP-Scale with syndrome-based early stopping at the first iterate satisfying $H\hat{\mathbf{x}}^b = 0 \pmod 2$. Dashed lines mark the 30-iteration training depth.

the highest $-\ln(\mathrm{BER})$ at all 18 code–$E_b/N_0$ operating points, with 91–93% fewer parameters and 20–24% fewer FLOPs than the lowest corresponding baseline value for each code. These results show that the accuracy gains do not rely on a larger model or inference budget, supporting both the effectiveness and efficiency of the proposed factorized message-passing design.

Table 2 compares I-FGMP-Scale with the diffusion-based DDECC (Choukroun & Wolf, 2023) and score-based SB-ECC (Helvits & Nachmani, 2026) decoders. All three use syndrome-based early stopping at the first iterate satisfying $H\hat{\mathbf{x}}^b = 0 \pmod 2$. This condition certifies codeword validity, but does not guarantee recovery of the transmitted codeword. I-FGMP-Scale allows up to 60 test-time iterations in this comparison. The FLOPs reported in the table correspond to the maximum number of test-time iterations allowed for each method and thus represent the maximum rather than the average inference cost. I-FGMP-Scale outperforms DDECC at all 18 operating points and achieves the highest $-\ln(\mathrm{BER})$ at 13 of them, while SB-ECC performs best at the remaining 5. I-FGMP-Scale uses approximately 7% of the parameters of either baseline, while its maximum test-time FLOPs are only 34–35% of those of SB-ECC and 5–13% of those of DDECC. These results demonstrate that I-FGMP-Scale attains competitive or superior decoding accuracy while requiring substantially fewer parameters and a much smaller maximum test-time computational cost. Moreover, the stopping statistics in Appendix B show that I-FGMP-Scale typically terminates well before 60 iterations, indicating that its average inference cost under early stopping is substantially lower than the maximum cost reported in the table.

Figure 3 evaluates the same trained checkpoint for each code with test-time budgets up to 200 iterations. All 18 operating points benefit from extending the budget beyond the 30-iteration training depth, without retraining or adding parameters, allowing a single model to operate at different performance–computation trade-offs. Gains generally diminish at larger budgets, and some curves plateau or fluctuate slightly. The useful iteration budget therefore depends on the operating point; parameter sharing alone does not guarantee monotonic improvement.

## 6 Conclusion

We introduced I-FGMP, a neural LDPC decoder that embeds the proposed FGMP operator in an iterative message-passing structure. FGMP provides adaptive source- and destination-conditioned message control around fixed graph aggregation, without pairwise interaction modeling. Reusing the operator keeps the parameter count independent of inference depth.

Experimentally, I-FGMP-12 outperforms the compared fixed-computation decoders with substantially fewer parameters and FLOPs. I-FGMP-Scale remains competitive with diffusion- and score-based decoders using fewer parameters and lower maximum test-time FLOPs, and benefits from iterations beyond its training depth. These results support factorized message control within weight-tied iterative architectures as a promising design for efficient neural decoding.

## AI USE STATEMENT

Generative AI tools were used to assist with code development, literature review, and manuscript preparation. The research problem, proposed method, experimental design, and interpretation of the results were developed by the authors. All AI-assisted work was reviewed and, where applicable, tested or verified by the authors. The authors take full responsibility for the final content of this work, including all text, claims, code, and other artifacts produced with the aid of generative AI.

## REFERENCES

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)*, pp. 1595–1599, 2018. doi: 10.1109/ISIT.2018.8437530.

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. doi: 10.1109/JSAC.2020.3041392.

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*, pp. 1–6, 2017. doi: 10.1109/GLOCOM.2017.8254811.

Sebastian Cammerer, Jakob Hoydis, Fayçal Aït Aoudia, and Alexander Keller. Graph neural networks for channel decoding. In *2022 IEEE Globecom Workshops (GC Wkshps)*, pp. 486–491, 2022. doi: 10.1109/GCWkshps56602.2022.10008601.

Jinghu Chen and Marc P. C. Fossorier. Near optimum universal belief propagation based decoding of low-density parity check codes. *IEEE Transactions on Communications*, 50(3):406–414, 2002. doi: 10.1109/26.990903.

Jinghu Chen, Ajay Dholakia, Evangelos Eleftheriou, Marc P. C. Fossorier, and Xiao-Yu Hu. Reduced-complexity decoding of LDPC codes. *IEEE Transactions on Communications*, 53(8): 1288–1299, 2005. doi: 10.1109/TCOMM.2005.852852.

Yoni Choukroun and Lior Wolf. Error correction code transformer. In *Advances in Neural Information Processing Systems*, volume 35, pp. 38695–38705, 2022.

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

Jincheng Dai, Kailin Tan, Zhongwei Si, Kai Niu, Mingzhe Chen, H. Vincent Poor, and Shuguang Cui. Learning to decode protograph LDPC codes. *IEEE Journal on Selected Areas in Communications*, 39(7):1983–1999, 2021. doi: 10.1109/JSAC.2021.3078488.

Marc P. C. Fossorier, Miodrag Mihaljević, and Hideki Imai. Reduced complexity iterative decoding of low-density parity check codes based on belief propagation. *IEEE Transactions on Communications*, 47(5):673–680, 1999. doi: 10.1109/26.768759.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings of the 2005 IEEE International Joint Conference on Neural Networks (IJCNN)*, volume 2, pp. 729–734, 2005. doi: 10.1109/IJCNN.2005.1555942.

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)*, pp. 1–6, 2017. doi: 10.1109/CISS.2017.7926071.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30, pp. 1024–1034, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html.

Michael Helmling, Stefan Scholl, Florian Gensheimer, Tobias Dietz, Kira Kraft, Oliver Griebel, Stefan Ruzika, and Norbert Wehn. Database of Channel Codes and ML Simulation Results. www.rptu.de/channel-codes, 2025.

Alon Helvits and Eliya Nachmani. Score based error correcting code decoder. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026. URL https://openreview.net/forum?id=uQiLoAF0bM.

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. URL https://openreview.net/forum?id=ryazCMbR-.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Hee-Youl Kwak, Dae-Young Yun, Yongjune Kim, Sang-Hyo Kim, and Jong-Seon No. Boosting learning for LDPC codes to improve the error-floor performance. In *Advances in Neural Information Processing Systems*, volume 36, pp. 22115–22131, 2023.

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.

Chin Wa Lau, Xiang Shi, Ziyan Zheng, Haiwen Cao, and Nian Guo. Interplay between belief propagation and transformer: Differential-attention message passing transformer. In *2025 IEEE International Symposium on Information Theory (ISIT)*, pp. 1668–1673, 2025. doi: 10.1109/ISIT63088.2025.11195462. URL https://arxiv.org/abs/2509.15637.

Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. Gated graph sequence neural networks. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1511.05493.

Loren Lugosch and Warren J. Gross. Neural offset min-sum decoding. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pp. 1361–1365, 2017. doi: 10.1109/ISIT.2017.8006751.

David J. C. MacKay and Radford M. Neal. Near Shannon limit performance of low-density parity-check codes. *Electronics Letters*, 32(18):1645–1646, 1996. doi: 10.1049/el:19961141.

Eliya Nachmani and Lior Wolf. Hyper-graph-network decoders for block codes. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

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. doi: 10.1109/JSTSP.2017.2788405.

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*, pp. 56710–56728, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/8f0942c43fcfba4cc66a859b9fcb1bba-Paper-Conference.pdf.

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 *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Xk8cwnwu2e.

Tom Richardson and Rüdiger Urbanke. *Modern Coding Theory*. Cambridge University Press, 2008. ISBN 9780521852296. doi: 10.1017/CBO9780511791338.

William E. Ryan and Shu Lin. *Channel Codes: Classical and Modern*. Cambridge University Press, 2009. ISBN 9780521848688. doi: 10.1017/CBO9780511803253.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. doi: 10.1109/TNN.2008.2005605.

R. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, 1981. doi: 10.1109/TIT.1981.1056404.

Kou Tian, Chentao Yue, Changyang She, Yonghui Li, and Branka Vucetic. A scalable graph neural network decoder for short block codes. In *2023 IEEE International Conference on Communications (ICC)*, pp. 1268–1273, 2023. doi: 10.1109/ICC45041.2023.10278711.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021. doi: 10.1109/TNNLS.2020.2978386.

## A Posterior Factorization

For $|y_i| > 0$ for all $i$, the binary-noise posterior factorizes as[1]

$$\begin{aligned}
&\Pr(\mathbf{z}^b \mid |\mathbf{y}|, s(\mathbf{y})) \\
&\overset{(a)}{\propto} \Pr(s(\mathbf{y}) \mid \mathbf{z}^b, |\mathbf{y}|) \Pr(\mathbf{z}^b \mid |\mathbf{y}|) \\
&\overset{(b)}{=} \mathbf{1}\big\{H\mathbf{z}^b = s(\mathbf{y}) \pmod 2\big\} \prod_{i=1}^{n} \Pr(z_i^b \mid |y_i|) \\
&\overset{(c)}{=} \mathbf{1}\big\{H\mathbf{z}^b = s(\mathbf{y}) \pmod 2\big\} \prod_{i=1}^{n} \frac{p_{Y|X}((1-2z_i^b)|y_i| \mid 1)}{p_{Y|X}(|y_i| \mid 1) + p_{Y|X}(-|y_i| \mid 1)} \\
&\overset{(d)}{=} \prod_{j=1}^{n-k} \psi_j(\mathbf{z}^b_{\mathcal{N}(j)}; s_j) \prod_{i=1}^{n} p_i^{z_i^b} (1-p_i)^{1-z_i^b},
\end{aligned} \tag{18}$$

where (a) is Bayes' rule; (b) uses memorylessness and $s(\mathbf{y}) = H\mathbf{z}^b \pmod 2$, obtained from $\mathbf{y}^b = \mathbf{x}^b + \mathbf{z}^b \pmod 2$ and (1); (c) uses the BISO noise law $p_{Y|X}(\cdot \mid 1)$ and $z_i = (1-2z_i^b)|y_i|$; and (d) combines local parity and Bernoulli factors, defined by

$$\psi_j(\mathbf{z}^b_{\mathcal{N}(j)}; s_j) \triangleq \mathbf{1}\left\{\sum_{i \in \mathcal{N}(j)} z_i^b = s_j \pmod 2\right\}, \qquad j = 1, \ldots, n-k,$$

$$p_i \triangleq \frac{p_{Y|X}(-|y_i| \mid 1)}{p_{Y|X}(|y_i| \mid 1) + p_{Y|X}(-|y_i| \mid 1)}, \qquad i = 1, \ldots, n.$$

Here, $s_j = [s(\mathbf{y})]_j$, $\mathcal{N}(j) = \{i : H_{j,i} = 1\}$, $\mathbf{z}^b_{\mathcal{N}(j)} = (z_i^b)_{i \in \mathcal{N}(j)}$, and $\mathbf{1}\{\cdot\}$ denotes the indicator function. The factorization in (18) induces the Tanner graph in Figure 1: $H$ determines connectivity, magnitudes provide local evidence, and syndrome entries specify the check constraints.

## B Early-Stopping Statistics

Table 3 reports the iterations executed by I-FGMP-Scale under the early-stopping rule of Table 2. With caps of 60 or 100, the mean is at most 1.44 iterations at 5 dB and 1.06 at 6 dB. At 4 dB, means range from 1.74 to 4.37, with greater variation in stopping times. Raising the cap from 60 to 100 has little effect on the mean at 5 and 6 dB and a larger effect at 4 dB, particularly for LDPC$(121, 60)$. Thus, most frames stop after few iterations, while allowing more iterations primarily affects difficult low-$E_b/N_0$ samples.

Table 3: Iterations executed by I-FGMP-Scale with early stopping when $H\hat{\mathbf{x}}^b = 0 \pmod 2$ and iteration cap $T_{\max}$. Columns give $E_b/N_0$ in dB; entries report the mean and standard deviation, $\mu \pm \sigma$. Frames that reach the cap without satisfying the parity constraints are recorded as $T_{\max}$.

| **Code** | $(n, k)$ | $T_{\max} = 60$ | | | $T_{\max} = 100$ | | |
|---|---|---|---|---|---|---|---|
| | | 4 | 5 | 6 | 4 | 5 | 6 |
| LDPC | $(49, 24)$ | $1.96 \pm 4.43$ | $1.17 \pm 1.11$ | $1.03 \pm 0.23$ | $2.10 \pm 6.47$ | $1.18 \pm 1.46$ | $1.03 \pm 0.25$ |
| | $(121, 60)$ | $3.80 \pm 8.02$ | $1.43 \pm 1.42$ | $1.05 \pm 0.27$ | $4.37 \pm 12.24$ | $1.44 \pm 1.91$ | $1.05 \pm 0.29$ |
| | $(121, 70)$ | $2.26 \pm 4.23$ | $1.18 \pm 0.67$ | $1.02 \pm 0.15$ | $2.38 \pm 6.14$ | $1.18 \pm 0.79$ | $1.02 \pm 0.16$ |
| | $(121, 80)$ | $1.74 \pm 2.75$ | $1.10 \pm 0.45$ | $1.01 \pm 0.10$ | $1.79 \pm 3.88$ | $1.10 \pm 0.50$ | $1.01 \pm 0.11$ |
| LDPC (MacKay) | $(96, 48)$ | $1.84 \pm 2.24$ | $1.20 \pm 0.61$ | $1.03 \pm 0.21$ | $1.87 \pm 2.96$ | $1.20 \pm 0.66$ | $1.03 \pm 0.23$ |
| LDPC (CCSDS) | $(128, 64)$ | $2.28 \pm 2.58$ | $1.34 \pm 0.68$ | $1.06 \pm 0.26$ | $2.32 \pm 3.55$ | $1.34 \pm 0.71$ | $1.06 \pm 0.27$ |

[1] AWGN outputs are nonzero almost surely; channels with positive mass at zero require separate treatment.

## C Ablation Study

Table 4 examines endpoint control in FGMP, parameter tying across iterations and message-passing directions, and supervision of intermediate outputs.

Table 4: Ablation of endpoint control, parameter tying, and intermediate supervision. Entries report final-iteration $-\ln(\mathrm{BER})$ at $E_b/N_0 \in \{4, 5, 6\}$ dB for variants trained and evaluated for 12 iterations. The reference is the configuration used in the main experiments; each variant changes only the indicated design choice. Higher is better; best and second-best results are **bold** and underlined, respectively.

| **Design choice** | **Variant** | LDPC $(49, 24)$ | | | LDPC $(121, 60)$ | | | MacKay $(96, 48)$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 4 | 5 | 6 | 4 | 5 | 6 | 4 | 5 | 6 |
| Reference | I-FGMP-12 | 6.85 | 9.86 | 13.65 | 6.02 | 9.85 | 15.22 | 8.48 | 11.99 | 15.64 |
| Endpoint control | W/o emission modulation $M$ | 6.65 | 9.48 | 13.25 | 5.85 | 9.67 | 14.82 | 8.29 | 11.76 | 15.25 |
| | W/o acceptance gate $G$ | 6.34 | 8.95 | 12.79 | 5.68 | 9.24 | 14.30 | 7.92 | 11.43 | 15.21 |
| | W/o $M$ and $G$ | 4.07 | 5.53 | 7.56 | 3.57 | 5.23 | 7.93 | 4.44 | 6.13 | 8.46 |
| Parameter tying | Untied across iterations | 6.92 | **10.21** | 13.74 | 6.08 | **9.94** | 15.51 | **8.68** | **12.47** | **16.15** |
| | Untied across directions | **6.96** | 10.02 | **14.02** | **6.13** | 9.88 | **15.60** | 8.52 | 12.29 | 15.71 |
| Supervision | Final iteration only | 6.51 | 9.27 | 12.99 | 5.91 | 9.67 | 15.06 | 8.39 | 11.90 | 15.63 |

**Variants.** I-FGMP-12 uses emission modulation $M$ and acceptance gating $G$, ties the FGMP parameters across all 12 iterations and both message-passing directions, and supervises every iteration. The endpoint-control variants remove $M$, $G$, or both. The parameter-tying variants use a distinct operator at each iteration or for each direction. The final-iteration-only variant removes intermediate supervision. All other settings remain unchanged.

**Results.** As shown in Table 4, removing either $M$ or $G$ degrades performance at every operating point, with removal of $G$ consistently causing the larger loss. Removing both causes a substantially larger degradation, showing that source-side emission and destination-side acceptance control both contribute. Together with the parameter counts in Table 5, these results show that untying parameters across iterations improves $-\ln(\mathrm{BER})$ by $0.06$–$0.51$ but raises the total parameter count to roughly $9$–$11\times$ that of I-FGMP-12. Untying the two directions yields gains by $0.03$–$0.38$ while keeping the total parameter count below $2\times$ that of I-FGMP-12, making it a viable alternative when a larger model is acceptable. Finally, final-iteration-only supervision is consistently worse than full-trajectory supervision, although the difference is code-dependent. Overall, I-FGMP-12 provides a favorable balance between decoding performance and parameter efficiency.

Table 5: Trainable parameter counts under different tying schemes.

| **Tying scheme** | LDPC $(49, 24)$ | LDPC $(121, 60)$ | MacKay $(96, 48)$ |
|---|---|---|---|
| I-FGMP-12 | 75.5 K | 89.3 K | 84.5 K |
| Untied iterations | 799 K | 813 K | 808 K |
| Untied directions | 141 K | 155 K | 150 K |

## D Comparison with Belief Propagation

Tables 6 and 7 compare I-FGMP with BP under fixed-computation and syndrome-early-stopping protocols, respectively. The I-FGMP-12 and I-FGMP-Scale entries reproduce the corresponding results from Tables 1(a) and 2(a). The BP experiments use the same decoding algorithm and iteration caps in both comparisons; their only protocol difference is whether syndrome early stopping is enabled.

Table 6: Fixed-computation comparison between I-FGMP-12 and BP on the six short codes used in the main experiments. I-FGMP-12 is evaluated at its final output after 12 iterations, whereas BP-$T$ is evaluated after exactly $T$ iterations; no syndrome early stopping is applied. Entries report $-\ln(\mathrm{BER})$ at the indicated $E_b/N_0$ in dB. Higher is better; the best and second-best results are **bold** and underlined, respectively.

| Code | $(n,k)$ | BP-5 | | | BP-50 | | | BP-100 | | | I-FGMP-12 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 4 | 5 | 6 | 4 | 5 | 6 | 4 | 5 | 6 | 4 | 5 | 6 |
| LDPC | $(49,24)$ | 5.28 | 7.19 | 9.71 | 6.09 | 8.76 | 11.82 | 6.24 | 9.00 | 12.30 | **6.85** | **9.86** | **13.65** |
| | $(121,60)$ | 4.81 | 7.19 | 10.73 | 5.66 | 8.95 | 13.35 | 5.77 | 9.33 | 14.10 | **6.02** | **9.85** | **15.22** |
| | $(121,70)$ | 5.87 | 8.70 | 12.64 | 6.93 | 10.67 | 15.61 | 7.14 | 11.07 | 16.41 | **7.48** | **11.99** | **17.78** |
| | $(121,80)$ | 6.62 | 9.64 | 13.83 | 7.87 | 11.74 | 17.13 | 8.05 | 12.12 | 17.90 | **8.46** | **13.03** | **18.44** |
| LDPC (MacKay) | $(96,48)$ | 6.72 | 9.40 | 12.77 | 8.05 | 11.14 | 14.54 | 8.29 | 11.61 | 15.16 | **8.48** | **11.99** | **15.64** |
| LDPC (CCSDS) | $(128,64)$ | 6.48 | 9.64 | 13.85 | 7.97 | 12.04 | 16.60 | 8.22 | 12.73 | 17.93 | **8.26** | **12.97** | **18.20** |

Table 7: Syndrome-early-stopping comparison between I-FGMP-Scale and BP on the six short codes used in the main experiments. Each decoder stops at the first iteration whose hard decision satisfies all parity checks and otherwise returns the output at its iteration cap: 60 for I-FGMP-Scale and $T$ for BP-$T$. Entries report $-\ln(\mathrm{BER})$ at the indicated $E_b/N_0$ in dB. Higher is better; the best and second-best results are **bold** and underlined, respectively.

| Code | $(n,k)$ | BP-5 | | | BP-50 | | | BP-100 | | | I-FGMP-Scale | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 4 | 5 | 6 | 4 | 5 | 6 | 4 | 5 | 6 | 4 | 5 | 6 |
| LDPC | $(49,24)$ | 5.26 | 7.19 | 9.82 | 6.10 | 8.75 | 11.55 | 6.23 | 9.03 | 12.22 | **7.25** | **10.35** | **14.03** |
| | $(121,60)$ | 4.79 | 7.13 | 10.44 | 5.60 | 8.76 | 13.47 | 5.73 | 9.13 | 13.99 | **6.53** | **10.63** | **16.54** |
| | $(121,70)$ | 5.81 | 8.81 | 12.72 | 6.94 | 10.83 | 15.96 | 7.10 | 11.25 | 16.46 | **8.11** | **12.92** | **18.80** |
| | $(121,80)$ | 6.62 | 9.79 | 14.13 | 7.85 | 11.93 | 17.17 | 8.04 | 12.30 | 18.31 | **9.12** | **14.12** | **19.47** |
| LDPC (MacKay) | $(96,48)$ | 6.77 | 9.52 | 12.98 | 8.14 | 11.18 | 14.55 | 8.28 | 11.73 | 15.37 | **9.33** | **13.02** | **16.26** |
| LDPC (CCSDS) | $(128,64)$ | 6.49 | 9.79 | 14.05 | 8.08 | 12.22 | 16.65 | 8.38 | 12.61 | 18.00 | **9.24** | **14.48** | **19.38** |

**Results.** Under fixed-computation evaluation, I-FGMP-12 achieves the best performance across all tested code–$E_b/N_0$ pairs, with BP-100 consistently ranking second. The same pattern holds under syndrome-based early stopping, where I-FGMP-Scale achieves the best performance throughout and BP-100 again ranks second. These results show that the advantage over BP is maintained under both stopping protocols for the tested short codes.

## E Performance on Medium- and Long-Length Codes

Table 8 reports the performance of our method on five medium- and long-length codes and compares it with BP. I-FGMP-12 follows exactly the same training procedure as in the main experiments.

Table 8: Fixed-computation comparison between I-FGMP-12 and BP on five medium- and long-length codes. I-FGMP-12 values are taken from the final output after 12 iterations, and BP-$T$ is evaluated after exactly $T$ iterations. Entries report $-\ln(\mathrm{BER})$ for $E_b/N_0$ from 2 to 5 dB in 0.5 dB increments. Higher is better; the best and second-best results are **bold** and underlined, respectively.

| Code | Decoder | $E_b/N_0$ (dB) | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | **2.0** | **2.5** | **3.0** | **3.5** | **4.0** | **4.5** | **5.0** |
| MacKay $(408, 204)$ | BP-5 | 3.71 | 4.71 | 6.05 | 7.74 | 9.77 | 12.03 | 14.47 |
| | BP-50 | 5.05 | 6.91 | 9.11 | 11.53 | 13.00 | 14.64 | 15.73 |
| | BP-100 | **5.11** | **7.11** | **9.49** | 11.84 | 13.30 | 15.35 | 17.74 |
| | I-FGMP-12 | 4.67 | 6.51 | 9.20 | **12.06** | **15.61** | **17.48** | **17.82** |
| CCSDS $(512, 256)$ | BP-5 | 3.14 | 4.14 | 5.70 | 7.91 | 10.84 | 14.50 | 18.73 |
| | BP-50 | 4.05 | 6.03 | 8.79 | 12.20 | 16.12 | 19.16 | 22.23 |
| | BP-100 | **4.09** | **6.16** | **9.07** | **12.78** | **17.17** | **20.70** | **22.77** |
| | I-FGMP-12 | 3.75 | 5.63 | 8.61 | 12.58 | 16.96 | 17.92 | 18.30 |
| LDPC $(529, 440)$ | BP-5 | **3.04** | 3.39 | 4.04 | 5.19 | 6.89 | 9.11 | 11.77 |
| | BP-50 | **3.04** | **3.43** | 4.25 | 5.80 | 8.16 | 11.24 | 14.80 |
| | BP-100 | **3.04** | **3.43** | **4.26** | **5.83** | **8.29** | 11.57 | 15.40 |
| | I-FGMP-12 | 3.02 | **3.43** | 4.25 | 5.79 | 8.22 | **11.65** | **16.05** |
| LDPC $(648, 540)$ | BP-5 | **3.05** | 3.42 | 4.09 | 5.32 | 7.21 | 9.68 | 12.61 |
| | BP-50 | **3.05** | **3.47** | 4.37 | 6.13 | 8.88 | 12.07 | 15.58 |
| | BP-100 | **3.05** | **3.47** | **4.38** | **6.17** | **9.02** | 12.34 | 15.94 |
| | I-FGMP-12 | 3.04 | 3.46 | 4.34 | 6.04 | 8.83 | **12.58** | **16.55** |
| LDPC $(1056, 880)$ | BP-5 | **3.09** | 3.48 | 4.24 | 5.68 | 7.95 | 10.85 | 13.99 |
| | BP-50 | **3.09** | **3.54** | 4.72 | 7.14 | 10.62 | 13.63 | 16.22 |
| | BP-100 | **3.09** | **3.54** | **4.73** | **7.22** | **10.86** | 13.87 | 16.56 |
| | I-FGMP-12 | 3.07 | 3.52 | 4.60 | 6.90 | 10.60 | **14.64** | **16.58** |

**Results.** Overall, I-FGMP-12 and BP achieve comparable performance on the tested medium- and long-length codes, with their relative performance varying across operating points. For most codes, I-FGMP-12 remains close to BP at lower $E_b/N_0$ and becomes better at higher $E_b/N_0$. CCSDS(512, 256) is an exception, for which BP-100 performs better across the entire tested range, with the gap becoming more pronounced at higher $E_b/N_0$. The close performance between I-FGMP and BP is consistent with the behavior of LDPC decoding at larger block lengths, where finite-length effects become less pronounced and BP more closely approaches its asymptotic iterative-decoding performance, potentially leaving less room for learned decoders to improve upon it.