TOWARD UNCERTAINTY-AWARE AND GENERALIZABLE NEURAL DECODING FOR QUANTUM LDPC CODES

Anonymous authors
Paper under double-blind review

ABSTRACT

Quantum error correction (QEC) is essential for scalable quantum computing, yet decoding errors via conventional algorithms results in limited accuracy (i.e., suppression of logical errors) and high overheads, both of which can be alleviated by inference-based decoders. To date, such machine-learning (ML) decoders lack two key properties crucial for practical fault tolerance: reliable uncertainty quantification and robust generalization to previously unseen codes. To address this gap, we propose **QuBA**, a Bayesian graph neural decoder that integrates attention to both dot-product and multi-head, enabling expressive error-pattern recognition alongside calibrated uncertainty estimates. Building on QuBA, we further develop **SAGU (Sequential Aggregate Generalization under Uncertainty)**, a multi-code training framework with enhanced cross-domain robustness enabling decoding beyond the training set. Experiments on bivariate bicycle (BB) codes and their coprime variants demonstrate that (i) both QuBA and SAGU consistently outperform the classical baseline belief propagation (BP), achieving a reduction of on average *one order of magnitude* in logical error rate (LER), and up to *two orders of magnitude* under confident-decision bounds on the coprime BB code $[[154, 6, 16]]$; (ii) QuBA also surpasses state-of-the-art neural decoders, providing an advantage of roughly *one order of magnitude* (e.g., for the larger BB code $[[756, 16, \leq 34]]$) even when considering conservative (safe) decision bounds; (iii) SAGU achieves decoding performance comparable to or even outperforming QuBA's domain-specific training approach. Our code implementation is available at `https://anonymous.4open.science/r/QuBA-SAGU-5FCD/`.

1 INTRODUCTION

Quantum error correction (QEC) (Calderbank & Shor, 1996) is an essential paradigm that enables quantum computation at negligible error rates over logical qubits (Kielpinski et al., 2002; Ye et al., 2023; He et al., 2025). By encoding a logical qubit into multiple physical qubits and measuring parity-check syndromes, QEC can diagnose and correct logical errors (bit and phase flips) without destroying a complex quantum state (Calderbank & Shor, 1996; Knill & Laflamme, 1997). Even though physical qubits are subject to noise (more frequent errors, currently around 10^{-3}), clever encoding into logical qubits reduces this error rate algorithmically via error correction to where it become negligible (up to 10^{-13}) at the logical level. Quantum low-density parity-check (LDPC) codes are a broad class of codes aimed at high error-correction performance combined with high encoding efficiency (Gottesman, 1997; Tillich & Zémor, 2013). Early quantum LDPC constructions included hypergraph product codes and hyperbolic codes, which demonstrated that constant-rate quantum codes with growing distance are possible (Freedman et al., 2002; Zémor, 2009; Zeng & Pryadko, 2019; Breuckmann & Terhal, 2016). More recent developments, such as balanced product codes and quantum Tanner codes, have achieved even stronger asymptotic guarantees, with some families proving to be good quantum codes, offering constant encoding rates and linear distance scaling (Breuckmann & Eberhardt, 2021; Leverrier & Zémor, 2022). These advances make quantum LDPC codes highly attractive for fault-tolerant quantum computing, as they can dramatically reduce the physical qubit overhead compared to surface codes while maintaining competitive thresholds. Among these, bivariate bicycle (BB) codes (Bravyi et al., 2024), published in *Nature*, have attracted particular attention for their balance between practicality and asymptotic performance, which generalize classical bicycle codes into two dimensions and achieve a threshold close to 0.8%, comparable to the surface code, but with substantially higher encoding rates (Postema & Kokkelmans, 2025).

Efficient decoding is critical for realizing the benefits of quantum codes with near-term quantum device technology. In decoding via general Tanner graphs, iterative belief propagation (BP) decoding is widely used due to its moderate computational complexity and high degree of parallelism (Kschischang et al., 2002; Poulin & Chung, 2008). However, the abundance of short cycles in Tanner graphs of quantum codes can severely degrade BP performance (Poulin & Chung, 2008; Kovalev & Pryadko, 2013), and degeneracy (i.e., multiple distinct errors corresponding to the same syndrome), can trap BP in symmetric belief states (Poulin, 2006; Panteleev & Kalachev, 2021). To address these limitations, several variants have been proposed. Memory-based BP (MBP) introduces additional memory effects (Kuo & Lai, 2022), while SymBreak explicitly mitigates degeneracy (Yin et al., 2024). Other improvements include generalized BP (Old & Rispler, 2022), guided decimation (Yao et al., 2024), sliding window decoding (Gong et al., 2024b), automorphism-ensemble decoding (Koutsioumpas et al., 2025), and speculative approaches (Wang et al.,

2025). Post-processing techniques have also been developed, such as ordered statistics decoding (OSD) (Roffe et al., 2020), which improves performance but at cubic computational cost and limited parallelism, as well as the more recent localized statistics decoding (LSD) (Hillmann et al., 2025), which offers a parallelizable alternative. More recently, the QEC community has turned toward advanced machine learning-based decoders, ranging from (recurrent or graph) neural networks (Nachmani et al., 2018; Liu & Poulin, 2019; Miao et al., 2022; Lange et al., 2025; Baireuther et al., 2017) to transformer architectures (Wang et al., 2023; Choukroun & Wolf, 2024), achieving state-of-the-art performance.

However, despite these advances, existing machine-learning decoders still face two critical limitations for practical fault tolerance. First, they generally lack *reliable uncertainty quantification* (see Sec. 4.1 for the classification of uncertainty in quantum decoding), making it difficult to assess confidence in decoding decisions or to design adaptive hybrid strategies. Second, their generalization ability across different quantum codes remains weak, as most approaches are trained domain-specifically and fail to transfer to unseen codes or varying noise conditions *without any retraining*. Motivated by these challenges, our contributions are summarized as follows:

- QuBA leverages Bayesian neural networks (BNNs) to represent predictive uncertainty, using Monte Carlo dropout at inference time to provide calibrated confidence estimates that enable adaptive decision-making. To better capture correlations in error syndromes, QuBA integrates dot-product and multi-head attentions within a graph neural network (GNN) architecture, enhancing relational reasoning on Tanner graphs (see Appendix A.1 for more detailed explanations and the connection between the attentions and quantum decoding).
- Beyond QuBA, and inspired by the Diversify-Aggregate-Repeat Training (DART) paradigm (Jain et al., 2023), we design SAGU (Sequential Aggregate Generalization under Uncertainty), a cross-domain training framework that consists of three phases (see Sec. 5) designed to strengthen generalization across heterogeneous quantum codes, by exploiting the complementary strengths of diverse code architectures and training data distributions in the quantum decoding setting.
- Together, QuBA and SAGU deliver not only improved decoding accuracy (i.e., suppression of logical errors) but also uncertainty awareness and strong cross-domain robustness. Experimental results across both standard and coprime BB codes illustrate the superiority of the proposed methods. For instance, QuBA achieves an improvement of nearly *two orders of magnitude* over BP for the coprime BB code $[[154, 6, 16]]$. Compared to the state-of-the-art neural decoder Astra, QuBA maintains almost *one order of magnitude* advantage on standard BB codes, even under safe decision bounds.

2 Related Work

Recent research on machine learning (ML)-based decoders has pushed the boundaries of classical and quantum decoding. Broadly, neural decoders can be grouped into two categories: model-based and model-free approaches.

Model-based decoding: Model-based approaches explicitly incorporate the Tanner graph structure of quantum LDPC codes into the neural architecture. Two main directions have emerged. *The first* integrates belief propagation (BP) with neural design by unfolding iterative BP updates into a differentiable architecture, allowing the update rules to be optimized through data-driven training (Nachmani et al., 2016; 2018; Nachmani & Wolf, 2021). This line of work was extended to QEC with neural BP decoders, which adapt message-passing rules to handle degeneracy in quantum LDPC codes (Liu & Poulin, 2019). *A second direction* leverages message-passing mechanisms in graph neural networks (GNNs), directly embedding Tanner graph connectivity into learned aggregation and update functions. Recent works demonstrated the effectiveness of GNN-based decoders for both classical and quantum LDPC codes, with notable progress on quantum LDPC decoding at scale (Gong et al., 2024a; Ninkovic et al., 2024; Maan & Paler, 2025). By combining structural priors with trainable neural layers, model-based decoders achieve high performance while retaining scalability and interpretability.

Model-free decoding: In contrast, model-free methods treat decoding as a purely data-driven task, without embedding explicit BP or Tanner graph mechanics. Early works employed neural networks trained directly on error-syndrome pairs to map syndromes to corrections, demonstrating the feasibility of purely supervised decoders under circuit-level noise (Baireuther et al., 2018). Subsequent advances introduced attention-based architectures such as self-attention and Transformers, which are capable of capturing global correlations in syndrome data and have shown strong decoding performance across classical and quantum codes (Raviv et al., 2020; Choukroun & Wolf, 2022; Wang et al., 2023; Cohen et al., 2025). Building on this line of work, a recurrent Transformer decoder was recently applied to bivariate bicycle codes, where a multi-stage training protocol enabled effective decoding under circuit-level noise (Blue et al., 2025). In parallel, GNNs have also been explored in fully data-driven settings, where decoding is formulated as a graph-classification problem and the network directly predicts the most likely logical error class (Lange et al., 2025).

Our approach: Our proposed method, QuBA, belongs to the model-based category, as it builds on GNN message passing while augmenting it with Bayesian attention mechanisms. Unlike prior model-based decoders, QuBA

provides explicit predictive uncertainty estimates and integrates attention to capture heterogeneous syndrome-qubit interactions. Together with the sequential training strategy SAGU, our framework achieves both stronger error suppression and broader cross-domain generalization than existing model-based or model-free approaches.

3 BACKGROUND

Quantum decoding: In the stabilizer formalism (Gottesman, 1997), a $[[n, k, d]]$ quantum code, encoding k logical qubits into n physical qubits with a code distance d (i.e., the minimum number of physical qubit errors that can cause an undetectable logical error), is defined by a stabilizer group $\mathcal{S} = \langle S_1, \ldots, S_{n-k} \rangle$ of commuting Pauli operators. The code space is the joint $+1$ eigenspace of all S_j. An error E anticommutes with a subset of stabilizers, producing a binary syndrome vector

$$s_j = \begin{cases} 0, & ES_j = S_jE, \\ 1, & ES_j = -S_jE, \end{cases} \quad j = 1, \ldots, n-k. \tag{1}$$

The quantum decoding problem is to find a correction E_{corr} such that

$$E_{\text{corr}}E \in \mathcal{S}, \tag{2}$$

i.e., E_{corr} differs from E by a stabilizer and thus restores the code state up to a global phase. Since multiple errors can yield the same syndrome $\mathbf{s}$, a maximum-likelihood decoder selects

$$\hat{E} = \arg\max_{E \in \mathcal{P}_n} P(E \,|\, \mathbf{s}), \tag{3}$$

where $\mathcal{P}_n$ is the n-qubit Pauli group. Graph-based quantum decoders realize Eq. 3 as belief propagation (BP) on the Tanner graph, where variable and check nodes exchange probabilistic information.

Graph neural networks: GNNs (Scarselli et al., 2008; Wu et al., 2020; Liu et al., 2022) extend deep learning to graph-structured data by iteratively exchanging and aggregating information between neighboring nodes. In *message-passing neural network* (MPNN) (Gilmer et al., 2017), the hidden state of each node v at iteration t is updated as

$$\mathbf{m}_v^{(t)} = \square_{u \in \mathcal{N}(v)} \psi^{(t)}\left(\mathbf{h}_v^{(t)}, \mathbf{h}_u^{(t)}, \mathbf{e}_{uv}\right), \quad \mathbf{h}_v^{(t+1)} = \phi^{(t)}\left(\mathbf{h}_v^{(t)}, \mathbf{m}_v^{(t)}\right), \tag{4}$$

where $\psi^{(t)}$ is the message function, $\phi^{(t)}$ is the node update function, $\mathbf{e}_{uv}$ encodes edge features, and $\square$ denotes a permutation-invariant aggregation (e.g., sum, mean, or max). This iterative scheme allows information to propagate over multi-hop neighborhoods, enabling the network to capture both local and global structural patterns.

Fig. 1 illustrates the relationship between BP and neural message-passing networks on the Tanner graph. For decoding, the Tanner graph of a quantum LDPC code naturally provides the input graph, where variable nodes correspond to physical qubits, check nodes correspond to syndrome bits, and edges encode qubit-stabilizer incidence. The GNN learns to pass and transform messages in a way that approximates maximum-likelihood decoding, potentially overcoming the limitations of hand-designed BP schedules in loopy graphs, which result in oscillations or bias accumulation (Raveendran & Vasić, 2020; Chytas et al., 2024).

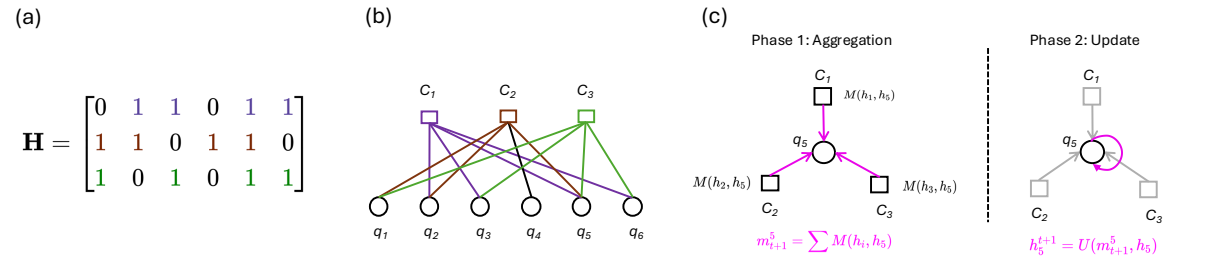


Figure 1: Relationship between belief propagation (BP) and message passing in graph neural networks (GNNs). **(a)**. A parity-check matrix. **(b)**. The Tanner graph constructed from the parity-check matrix, showing the connections between check nodes and variable nodes (physical qubits). BP operates by exchanging information between check and variable nodes (from check nodes to variable nodes, and vice versa) over T iterations. **(c)**. A message-passing neural network (MPNN) on the Tanner graph. At each iteration, message passing proceeds in two steps. In the aggregation step, every node v computes messages for its neighbors $u \in \mathcal{N}(v)$ by applying a learnable message function $M(\cdot)$. All incoming messages are aggregated at the receiving node using a permutation-invariant operator such as element-wise summation. In the update step, the hidden state of node v is updated by an update function $U(\cdot)$ that combines the previous state with the aggregated messages. After T such iterations, the hidden representation of each node reflects information from its T-hop neighborhood.

4 QUBA: A QUANTUM BAYESIAN ATTENTION DECODER

4.1 UNCERTAINTY REPRESENTATION

In QEC, both the physical error process and the decoding model introduce uncertainty. *Physical uncertainty* arises from the stochastic nature of the Pauli noise channel, which produces different physical error patterns even under identical circuit operations. *Model uncertainty* stems from limited training data, code degeneracy, and imperfect generalization to unseen syndromes. Accurately representing both types of uncertainty is essential. It allows the decoder to output well-calibrated confidence estimates, improves robustness to distribution shifts (e.g., changes in error rate), and supports downstream decision-making such as hybrid decoding with ordered statistics decoding (OSD).

To enable uncertainty-aware decoding, we adopt a Bayesian neural network (BNN) formulation for our GNN-based decoder. Unlike conventional deep neural networks (DNNs) that learn a single point estimate for each parameter, in a BNN every model parameter $\theta \in \Theta$ is treated as a random variable rather than a fixed value. A prior distribution $p(\theta)$, chosen as a standard Gaussian $\mathcal{N}(0,1)$, encodes our initial belief about these parameters before observing any data. Given a training dataset $\mathcal{D} = \{(\mathbf{s}_i, \mathbf{e}_i)\}_{i=1}^N$, where $\mathbf{s}_i$ denotes the measured stabilizer syndrome and $\mathbf{e}_i$ is the corresponding physical error pattern on data qubits, the posterior distribution over parameters is obtained via Bayes' theorem:

$$p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)\, p(\theta)}{p(\mathcal{D})} = \frac{\prod_{i=1}^N p(\mathbf{e}_i \mid \mathbf{s}_i, \theta)\, p(\theta)}{p(\mathcal{D})}. \tag{5}$$

For a new measured syndrome $\mathbf{s}^*$, the predictive distribution over the corresponding physical error pattern $\mathbf{e}^*$ is obtained by marginalizing over the posterior as

$$p(\mathbf{e}^* \mid \mathbf{s}^*, \mathcal{D}) = \int_\theta p(\mathbf{e}^* \mid \mathbf{s}^*, \theta)\; p(\theta \mid \mathcal{D})\; d\theta, \tag{6}$$

and the prediction of such a marginal distribution incorporates both data and model uncertainties (Abdar et al., 2021).

In practice, however, the exact posterior $p(\theta \mid \mathcal{D})$ in Eq. 5 is intractable for gradient-based optimization, since it requires integration over the entire parameter space Θ. To make learning feasible, we approximate the posterior with a factorized Gaussian variational distribution $q_\phi(\theta)$ (Graves, 2011; Blundell et al., 2015; Louizos & Welling, 2016), parameterized by variational parameters ϕ. This approximation is optimized by minimizing the Kullback–Leibler (KL) divergence

$$\min_\phi \; \mathrm{KL}(q_\phi(\theta) \,\|\, p(\theta \mid \mathcal{D})) = \min_\phi \int_{\theta \in \Theta} q_\phi(\theta) \log \frac{q_\phi(\theta)}{p(\theta \mid \mathcal{D})}\, d\theta. \tag{7}$$

Under the Gaussian assumption in each BNN layer, this KL term has the closed-form expression

$$KL = \frac{1}{2} \sum_j \left[\frac{\sigma_j^2}{\sigma_p^2} + \frac{\mu_j^2}{\sigma_p^2} - 1 - \log \frac{\sigma_j^2}{\sigma_p^2} \right], \tag{8}$$

where (μ_j, σ_j^2) are the variational parameters of $q_\phi(\theta)$ and σ_p^2 is the prior variance. The total KL regularizer is summed over all Bayesian layers in the decoder, ensuring posterior distributions remain close to the prior.

Monte Carlo prediction: At inference, we draw M independent weight samples $\{\theta^{(m)}\}_{m=1}^M$ from $q_\phi(\theta)$ and run the decoder on the same syndrome $\mathbf{s}$ to obtain $\hat{\mathbf{e}}^{(m)} = f_{\theta^{(m)}}(\mathbf{s})$. From these predictions, we compute the empirical mean and variance

$$\hat{\mu} = \frac{1}{M} \sum_{m=1}^M \hat{\mathbf{e}}^{(m)}, \quad \hat{\sigma}^2 = \frac{1}{M} \sum_{m=1}^M \left(\hat{\mathbf{e}}^{(m)} - \hat{\mu}\right)^2. \tag{9}$$

A 95% confidence interval for each predicted error probability is then approximated by $CI^{0.95} \approx \hat{\mu} \pm 2\hat{\sigma}$. This MC-based prediction procedure captures epistemic uncertainty through weight sampling and predictive variability from the decoder's output distribution, both of which are crucial for robust and reliable QEC decoding.

4.2 DECODER DESIGN

Our decoder is a graph neural network that integrates (i) *edge-aware multi-head attention* and (ii) *LSTM-based recurrent state updates* with Bayesian parameterization for uncertainty quantification.

Node initialization: Each node i begins from a shared learnable embedding

$$\mathbf{h}_i^{(0)} = \mathbf{e}_0 \in \mathbb{R}^{d_h}, \tag{10}$$

where d_h is the hidden dimension. This provides a uniform initialization for iterative message passing.

Edge-aware multi-head attention: At iteration t, hidden states are projected into queries and keys using Bayesian linear layers with BatchNorm

$$\mathbf{Q}_i = \mathrm{BN}\Big(\mathbf{W}_Q \mathbf{h}_i^{(t)} + \mathbf{b}_Q\Big), \quad \mathbf{K}_j = \mathrm{BN}\Big(\mathbf{W}_K \mathbf{h}_j^{(t)} + \mathbf{b}_K\Big), \tag{11}$$

where $\mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{d_h \times (H d_{\text{head}})}$. The vectors are reshaped into H heads of dimension d_{head}.

For each edge $(i \to j)$ and head h, the scaled dot-product attention score is

$$s_{ij}^{(h)} = \frac{\mathrm{LeakyReLU}\Big(\langle \mathbf{q}_i^{(h)}, \mathbf{k}_j^{(h)} \rangle\Big)}{\tau}, \tag{12}$$

where τ is a learnable temperature. To stabilize training, scores are shifted by the maximum value at each destination and normalized with a scatter-based softmax

$$\alpha_{ij}^{(h)} = \frac{\exp(s_{ij}^{(h)} - \max_{u \in \mathcal{N}(j)} s_{uj}^{(h)})}{\sum_{u \in \mathcal{N}(j)} \exp(s_{uj}^{(h)} - \max_{v \in \mathcal{N}(j)} s_{vj}^{(h)})}. \tag{13}$$

Message network: Values are produced by a deep Bayesian MLP operating on concatenated source and destination states

$$\mathbf{v}_{ij} = \mathrm{MsgNet}\Big([\mathbf{h}_i^{(t)}, \mathbf{h}_j^{(t)}]\Big) \in \mathbb{R}^{H d_{\text{head}}}, \tag{14}$$

with per-head values $\mathbf{v}_{ij}^{(h)} \in \mathbb{R}^{d_{\text{head}}}$, and messages are then scaled by attention weights

$$\mathbf{m}_{ij}^{(h)} = \alpha_{ij}^{(h)} \mathbf{v}_{ij}^{(h)}, \quad \mathbf{m}_{ij} = \mathrm{Concat}_h \mathbf{m}_{ij}^{(h)}. \tag{15}$$

Aggregation: Finally, the messages are aggregated by summation over incoming edges

$$\mathbf{M}_j = \sum_{i \in \mathcal{N}(j)} \mathbf{m}_{ij}. \tag{16}$$

LSTM-based recurrent update: Each node update concatenates aggregated messages with static node inputs $\mathbf{x}_j$

$$\mathbf{z}_j^{(t)} = [\mathbf{M}_j, \mathbf{x}_j]. \tag{17}$$

The Long Short-Term Memory (LSTM) cell then updates the hidden and cell states

$$\mathbf{h}_j^{(t+1)}, \mathbf{c}_j^{(t+1)} = \mathrm{LSTM}\Big(\mathbf{z}_j^{(t)}, (\mathbf{h}_j^{(t)}, \mathbf{c}_j^{(t)})\Big), \tag{18}$$

where $\mathbf{h}_j^{(t+1)}$ denotes the hidden state of node j at iteration $t+1$ (short-term representation), while $\mathbf{c}_j^{(t+1)}$ denotes the corresponding cell state (long-term representation), which preserves long-range dependencies across multiple syndrome updates.

A residual connection with dropout stabilizes the dynamics

$$\mathbf{h}_j^{(t+1)} = \mathrm{Dropout}(\mathbf{h}_j^{\text{new}}) + \mathbf{h}_j^{(t)}. \tag{19}$$

Final output: At each iteration, hidden states are mapped to class logits via a Bayesian linear output layer

$$\mathbf{y}_j^{(t)} = \mathbf{W}_{\text{out}} \mathbf{h}_j^{(t)} + \mathbf{b}_{\text{out}}. \tag{20}$$

Overall, this design enables the decoder to (i) adaptively weight syndrome-qubit interactions through attention (for more details, see Appendix A.1), (ii) propagate parameter uncertainty through Bayesian layers, and (iii) maintain long-range temporal consistency with recurrent memory in quantum decoding. Together, these mechanisms provide robustness to degeneracy and improved generalization on large Tanner graphs with circular dependencies.

4.3 Loss Function

In QEC, let $\mathbf{e} \in \{0,1\}^n$ denote the true error in binary symplectic form, corresponding to a Pauli error operator E_1, and let $E_2 = E_1 S_j$ denote another Pauli error differing from E_1 by a stabilizer S_j. Since E_1 and E_2 act identically on all code states, it suffices for the total error $\mathbf{e}_{\text{tot}} = \mathbf{e} + \mathbf{e}_{\text{inf}} \pmod 2$ to belong to the stabilizer group, i.e., the set of all Pauli operators generated by the rows of the parity-check matrix H. To verify that $\mathbf{e}_{\text{tot}}$ belongs to the stabilizer group, one needs to check that it commutes with all stabilizers (Liu & Poulin, 2019)

$$H^{\perp} M \mathbf{e}_{\text{tot}} \equiv 0 \pmod 2, \tag{21}$$

where $H^{\perp}$ is the matrix generating the orthogonal complement of H with respect to the symplectic inner product, and M is the symplectic form. In practice, the non-differentiable parity check $\mathrm{parity}(x) = x \bmod 2$ is replaced by a smooth surrogate $f(x) = |\sin \pi x/2|$, which facilitates gradient-based learning while retaining the stabilizer consistency constraint.

The decoder produces logits for both qubit error variables and check (syndrome) variables at each message-passing iteration $t \in \{1, \ldots, T\}$. These outputs are physically linked through the stabilizer constraint $\mathbf{s} = H\mathbf{e} \pmod 2$, where $\mathbf{s}$ denotes the syndrome vector. Because the forward pass does not explicitly enforce $\hat{\mathbf{s}} = H\hat{\mathbf{e}}$, we include a *syndrome cross-entropy loss* $\mathcal{L}_{\mathrm{CE,s}}$ to penalize a mismatch between predicted and measured syndromes. This serves two purposes: (1) It reinforces physical consistency between predicted errors and the corresponding stabilizer measurements. (2) It provides an auxiliary training signal, particularly valuable in early epochs, that guides the network toward the correct relationship between error and syndrome. Our final composite Bayesian objective averages task losses over all T iterations and adds a KL regularizer from the BNN formulation in Eq. 8 yielding

$$\mathcal{L}(\theta) = \frac{1}{T} \sum_{t=1}^{T} \left(\mathcal{L}_{\mathrm{LER}}^{(t)} + \tfrac{1}{2}\mathcal{L}_{\mathrm{CE,e}}^{(t)} + \tfrac{1}{2}\mathcal{L}_{\mathrm{CE,s}}^{(t)} \right) + \beta(\tau)\, \mathrm{KL}(q_\phi(\theta) \,\|\, p(\theta \mid \mathcal{D}))\,, \tag{22}$$

where $\mathcal{L}_{\mathrm{LER}}^{(t)}$ is a differentiable logical error rate loss, incorporating the stabilizer-group consistency check, $\mathcal{L}_{\mathrm{CE,e}}^{(t)}$ is the cross-entropy loss over predicted error bits, and $\mathcal{L}_{\mathrm{CE,s}}^{(t)}$ is the cross-entropy loss over predicted syndrome bits. The syndrome loss is consistent with the loss in (Liu & Poulin, 2019), but it operates on syndromes rather than errors. The KL regularizes the variational posterior in each Bayesian linear layer toward its prior, with $\beta(\tau)$ annealed over training to progressively introduce Bayesian regularization.

In summary, this loss encourages the network to produce error predictions that are not only statistically accurate but also physically consistent with the stabilizer formalism, while explicitly modeling epistemic uncertainty through the Bayesian parameterization.

5 SAGU: Generalizable Training

We extend *Diversify-Aggregate-Repeat Training* (DART) (Jain et al., 2023) to the setting of Bayesian graph decoding under domain shift across quantum codes. We refer to this variant as SAGU (*Sequential Aggregate Generalization under Uncertainty*), which focuses on uncertainty-aware generalization across heterogeneous code families. In contrast to the original DART, which focuses on domain specialization, our goal is to integrate different code constructions together with diverse data properties in quantum decoding, including structural effects such as trapping sets in the Tanner graphs of different codes, to enhance robustness and generalization.

Within our SAGU framework, training is organized into three phases: *Warm-up*, *Diversify-Aggregate*, and *Consolidation*. Each model is trained on distinct datasets in the corresponding phase, while the training and validation sets are of equal size in every phase, i.e., $|\mathcal{D}_{\mathrm{warm}}| = |\mathcal{D}_{\mathrm{cons}}| = \sum_{k=1}^{M} |\mathcal{D}_k|$, where $\mathcal{D}_{\mathrm{warm}}$ and $\mathcal{D}_{\mathrm{cons}}$ denote the training or validation sets used in the warm-up and consolidation phases, respectively, and $\mathcal{D}_k$ indicates the training or validation set for the k-th model in the diversify-aggregate phase. Note that the three phases use the same loss objective defined in Eq. 22. We refer to the trained models (i.e., the decoders for specific QEC codes) in the three phases as the *starting domain*, the *diversity domain*, and the *aggregation domain*. Collectively, these three domains are referred to as *in-domain*, while models outside of them are considered *out-of-domain*. More specifically, the details and roles of each phase are described in the following paragraphs, and the three-phase training procedure is summarized in Alg. 1

Warm-up: We first optimize a single decoder f_θ, serving as the *starting domain*, on $\mathcal{D}_{\mathrm{warm}}$ for E_m epochs using AdamW with a phase-specific StepLR schedule. This stage yields parameters θ_{start} that capture general decoding structure and serve as the initialization and the starting point for all domain-specific models. Typically, the starting point is a small QEC code, i.e., one with a smaller code distance that can correct fewer errors. **Diversify-Aggregate:** We instantiate M *diversity domain* decoders $\{f_{\theta_k}\}_{k=1}^{M}$ with the trained model in the previous phase $\theta_k \leftarrow \theta_{\mathrm{start}}$. For each epoch $\tau \in [E_m, E_m)$, every model is trained independently on its domain $\mathcal{D}_k$ with its own optimizer and StepLR. After every λ epochs (or at $\tau = E_m - 1$,), parameters are synchronized by a weighted average $\bar{\theta} = \sum_{k=1}^{M} w_k\, \theta_k$ with $\sum_k w_k = 1$, where the weights bias aggregation toward harder domains (larger d_k). This balances domain-specific specialization with cross-domain sharing of structural knowledge, including patterns arising from different code structures and quantum degeneracy. **Consolidation:** The final centralized $\bar{\theta}$ initializes a single *aggregation domain* decoder. We fine-tune it for the remaining epochs $[E_m, E_{\mathrm{tot}})$ on the target dataset (same size as warm-up) with a reduced learning rate and StepLR, selecting checkpoints by validating logical-error metrics and applying early stopping when the total logical error rate (LER) fails to improve within a patience window or when $\mathrm{LER}_{\mathrm{tot}}$ completely converges (i.e., reaches zero).

Algorithm 1: Sequential Aggregate Generalization under Uncertainty-SAGU

Input: The training/validation data across all phases: $\mathcal{D}_{\text{warm}}$, $\mathcal{D}_{\text{cons}}$, and $\mathcal{D}_k$ for $k = 1, \ldots, M$; aggregation weights $w \in \mathbb{R}^M$ with $\sum_k w_k = 1$; epoch budgets E_w, E_m ($E_w < E_m$), and E_{tot}; aggregation interval λ; AdamW optimizers and *per-phase* StepLR schedulers.

Output: Final parameters θ_{final} on the aggregation domain.

```
// Warm-up Phase
```
Initialize the starting domain decoder $f_{\theta_{start}}$;
for $\tau = 0, \ldots, E_w - 1$ **do**
  Train $f_{\theta_{start}}$ for one epoch on $\mathcal{D}_{warm}$; step the warm-up StepLR.
Set $\theta_k \leftarrow \theta_{start}$ for all $k \in \{1, \ldots, M\}$.
```
// Diversify-Aggregate Phase
```
Instantiate the diversity domain decoders $\{f_{\theta_k}\}$, each with its own optimizer and StepLR;
for $\tau = E_w, \ldots, E_m - 1$ **do**
  for $k = 1, \ldots, M$ **do**
    Train f_{θ_k} for one epoch on $\mathcal{D}_k$; step domain StepLR.
  if $(\tau{+}1{-}E_m) \bmod \lambda = 0$ ***or*** $\tau = E_m{-}1$ **then**
    $\bar{\theta} \leftarrow \sum_{k=1}^{M} w_k \, \theta_k$
    for $k = 1, \ldots, M$ **do**
      $\theta_k \leftarrow \bar{\theta}$
```
// Consolidation Phase
```
Initialize the aggregation domain $f_{\theta_{final}}$ on $\mathcal{D}_{\text{cons}}$, load $\bar{\theta}$;
for $\tau = E_m, \ldots, E_{tot} - 1$ **do**
  Train $f_{\theta_{final}}$ for one epoch on target data with reduced LR; step StepLR.
  Evaluate LER and save if improved; early stop if LER shows no improvement within patience or if $\text{LER}_{\text{tot}} = 0$.
return θ_{final}.

6 Experiments

6.1 Setup

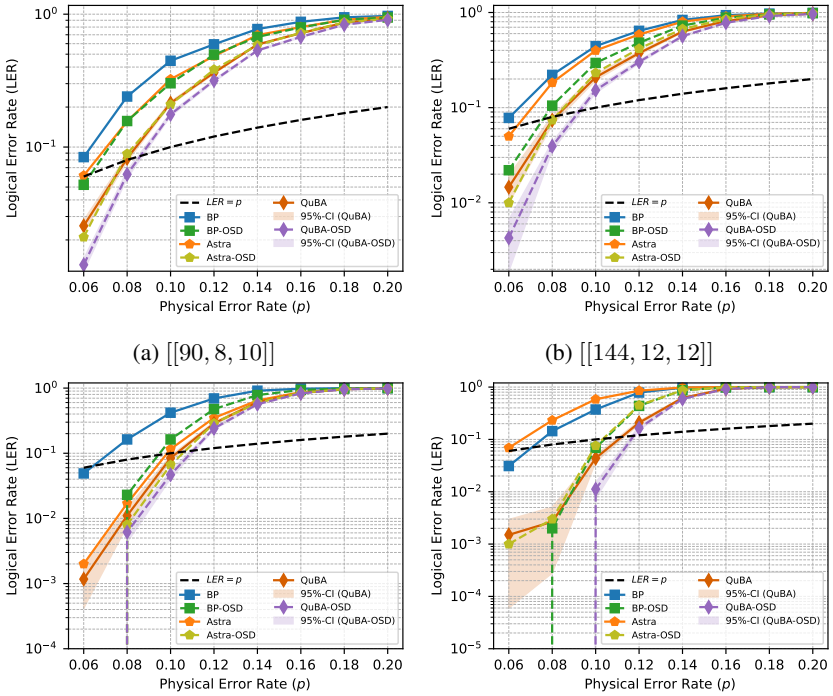
In line with prior literature, we benchmark two classical decoding methods (namely BP (Poulin & Chung, 2008) and BP-OSD (Roffe et al., 2020)) as well as a state-of-the-art neural decoder, Astra (Maan & Paler, 2025), on both BB and coprime BB codes under the depolarizing error model, where each Pauli operator (X, Y, or Z) flips with probability $1/3$ (see Appendix A.2 for the assumptions on errors). We then compare these baselines with our proposed decoders, QuBA and SAGU, evaluating performance both with and without OSD post-processing. The specific constructions of BB codes and coprime BB codes are provided in Appendix A.3. The training data and hyperparameters (selected via grid search) are reported in Appendix A.4, and the settings and details for model comparisons are summarized in Appendix A.5.

6.2 Result Analysis

Decoding results for different BB codes (see subcaption) are presented in Fig. 2 for varying physical error rates (PER, on the x-axis) ideally resulting in lower logical error rates (LER, on the y-axis) over the set of frameworks discussed before (see legend). Results for coprime BB codes are given in Appendix A.6. Appendix A.7 provides a comparison and discussion of a pair of BB codes and coprime BB codes of approximately equal scale. Fig. 3 evaluates the generalization ability of SAGU across different BB code families for the same parameters.

6.2.1 BB Codes

Overall trends: Across consecutive BB codes (from smaller to larger), the LER for across all schemes (but particularly for our QuBA/QuBA-OSD) exhibit clear order-of-magnitude shifts at fixed PER p (see Fig. 2). E.g., $[[90, 8, 10]]$ maintains a LER above 10^{-2}, while $[[144, 12, 12]]$ drops below 10^{-2}, representing a reduction of one order. A similar shift is observed from $[[288, 12, 18]]$ to $[[756, 16, \leq 34]]$. Particularly striking is the transition from $[[144, 12, 12]]$ to $[[288, 12, 18]]$, where the LER decreases by nearly two orders, from the 10^{-2} regime to the 10^{-4} regime. Furthermore, from $[[90, 8, 10]]$ to $[[756, 16, \leq 34]]$, both our QuBA and QuBA-OSD consistently push more LER values below the break-even line ($LER = p$). This indicates that the proposed method effectively exploits the advantage of larger code distances, correcting more physical errors and thereby lowering LER across regimes.



(a) $[[90, 8, 10]]$ (b) $[[144, 12, 12]]$

(c) $[[288, 12, 18]]$ (d) $[[756, 16, \leq 34]]$

Figure 2: Logical error rate (LER) vs. physical error rate (p) for BB codes. Lines become vertical when all errors are corrected (LER=0).

Comparison with BP and BP-OSD: Across all BB codes, QuBA consistently outperforms BP, with advantages up to two orders of magnitude. E.g., at $p = 0.06$, QuBA achieves 0.00140 ± 0.00122 on $[[756, 16, \leq 34]]$, compared to BP's 0.031. Under high-confidence decisions (uncertainty lower bound), the gain increases to three orders of magnitude. Even with OSD post-processing, QuBA outperforms BP-OSD on smaller codes such as $[[90, 8, 10]]$ and $[[144, 12, 12]]$. For larger codes like $[[288, 12, 18]]$ and $[[756, 16, \leq 34]]$, QuBA still surpasses BP-OSD at most PER values, except at $p = 0.06$ where BP-OSD saturates to zero. For instance, on $[[288, 12, 18]]$ at $p = 0.10$, QuBA achieves 0.08430 ± 0.00914, nearly an order lower than BP-OSD's 0.163. These results highlight that QuBA's robustness derives primarily from its model architecture rather than reliance on heavy post-processing.

Comparison with Astra: Against Astra, QuBA demonstrates similar or even larger advantages. Across all tested codes, QuBA outperforms Astra, often by one to two orders of magnitude. E.g., at $p = 0.06$ on $[[756, 16, \leq 34]]$, QuBA achieves 0.00140 ± 0.00122, compared to Astra's 0.07, a difference of nearly two orders. Under confidence-based evaluation, this gap widens to three orders, even surpassing Astra-OSD (0.001). With OSD, QuBA establishes superiority over all baselines. For instance, on $[[756, 16, \leq 34]]$ at $p = 0.08$, QuBA-OSD fully converges with no uncertainty (0.00000 ± 0.00000), compared to Astra-OSD's 0.003, achieving an improvement of three orders.

Summary: Overall, QuBA demonstrates systematic improvements over BP and Astra across all BB codes, with or without OSD. Its advantages range from one to three orders of magnitude depending on PER and evaluation setting. With OSD, QuBA sets the strongest benchmarks, often achieving complete convergence. Importantly, even without OSD, QuBA remains competitive against post-processing-enhanced baselines, underscoring the strength of its attention design.

6.2.2 SAGU: Domain Generalization

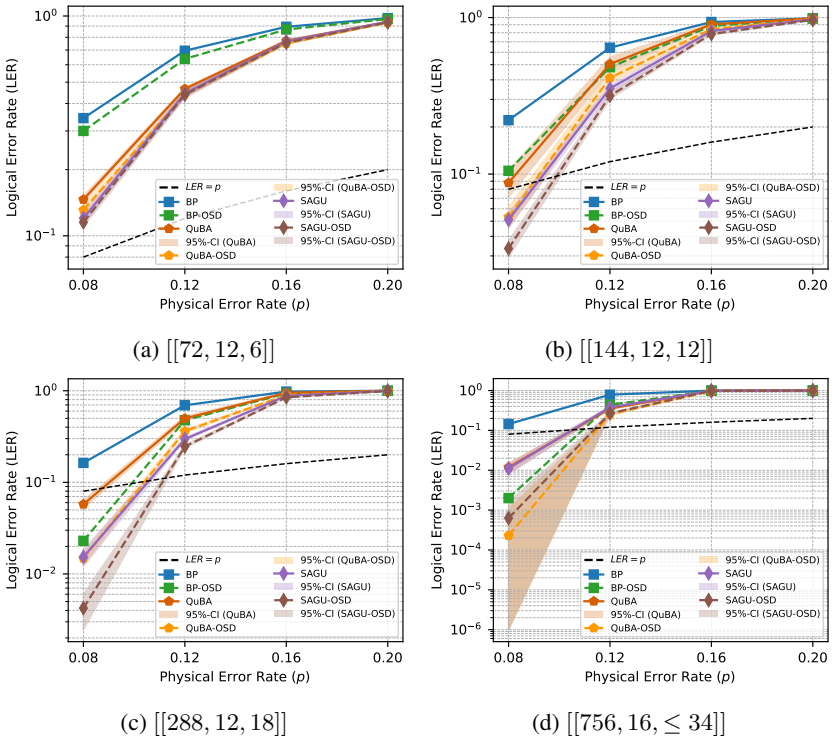
Overall trends: Across different domains, the LER for the our generalized SAGU method shows pronounced variation as the code size increases under the same PER. Fig. 3 shows that in the starting domain $[[72, 12, 6]]$, the LER remains above 10^{-1}. As the code size increases to the diversity domain $[[144, 12, 12]]$, the LER falls below 10^{-1}. For the aggregation domain $[[288, 12, 18]]$, the LER further decreases to about 10^{-2}, and for the out-of-domain case $[[756, 16, \leq 34]]$, it drops as low as 10^{-6}. This progression reflects nearly four orders of magnitude of improvement in error suppression. Moreover, the break-even line ($LER = p$) shifts rightward as the code size increases, reflecting the expected improvement in error suppression with larger codes. Importantly, SAGU mirrors the advantage of domain-specific training methods such as QuBA in pushing the break-even PER higher, demonstrating its ability to generalize effectively across domains.

Comparison with BP and BP-OSD: SAGU consistently outperforms both BP and BP-OSD across the in-domain codes $[[72, 12, 6]]$, $[[144, 12, 12]]$, and $[[288, 12, 18]]$, with improvements reaching up to one order of magnitude.

For instance, on $[[288, 12, 18]]$ at $p = 0.08$, BP-OSD attains an LER of 0.023, while SAGU achieves 0.01533 ± 0.00320. Confidence-based evaluation further amplifies this advantage, and even under conservative estimates (upper confidence bounds), SAGU maintains its lead, underscoring model reliability. With OSD post-processing, SAGU surpasses all competitors, including BP-OSD and QuBA-OSD, across all codes. For the larger codes $[[288, 12, 18]]$ and $[[756, 16, \leq 34]]$, SAGU-OSD achieves improvements of up to two orders of magnitude at $p = 0.08$.

Comparison with QuBA: As discussed in previous sections, QuBA already outperforms BP with and without OSD. Here, we focus on the additional benefits of cross-domain training with SAGU relative to the domain-specific QuBA. At $p = 0.08$, SAGU consistently improves upon QuBA by about 0.02–0.03 in LER on $[[72, 12, 6]]$ and $[[144, 12, 12]]$, both with and without OSD, within the confidence bounds. On $[[288, 12, 18]]$, SAGU achieves nearly 0.04 improvement over QuBA, and when enhanced with OSD, SAGU-OSD gains a full order of magnitude advantage over QuBA-OSD (0.00423 ± 0.00177 vs. 0.01480 ± 0.00233). On the out-of-domain code $[[756, 16, \leq 34]]$, SAGU performs only marginally worse than QuBA, with differences confined to the fourth decimal place, while SAGU-OSD and QuBA-OSD show nearly identical performance.

Summary: SAGU consistently outperforms BP and BP-OSD across all in-domain codes, and with OSD post-processing, it surpasses both QuBA-OSD and BP-OSD on nearly all benchmarks. On larger codes, SAGU achieves improvements of up to two orders of magnitude in LER, and even under conservative confidence bounds it maintains a clear advantage. Compared to the domain-specific QuBA, SAGU achieves modest but consistent gains on smaller and intermediate codes, and maintains competitive performance for the out-of-domain case. Overall, these results highlight SAGU's strong cross-domain generalization, reliability, and scalability for decoding quantum LDPC codes.



(a) $[[72, 12, 6]]$ (b) $[[144, 12, 12]]$

(c) $[[288, 12, 18]]$ (d) $[[756, 16, \leq 34]]$

Figure 3: Performance comparison of SAGU and other decoding methods across all domains, with and without OSD, on BB codes. (a): starting domain $[[72, 12, 6]]$; (b): diversity domain $[[144, 12, 12]]$; (c): aggregation domain $[[288, 12, 18]]$; and (d): out-of-domain $[[756, 16, \leq 34]]$.

7 Conclusions

We presented QuBA, a Bayesian graph neural network decoder that combines edge-aware attention with recurrent memory, enabling both uncertainty-aware predictions and effective multi-round reasoning. Building on this architecture, we introduced SAGU, a sequential training paradigm that promotes generalization across quantum LDPC codes and noise regimes. Our experiments on BB and coprime BB codes demonstrate that QuBA consistently outperforms classical decoders (BP, BP-OSD) and state-of-the-art neural approaches (Astra), achieving on *average nearly one order of magnitude* improvement in LER, with gains reaching up to *two orders of magnitude* under confident-decision bounds. Notably, these advantages hold even in the absence of OSD post-processing, highlighting QuBA's robustness. Moreover, SAGU generalizes successfully across domains, maintaining high performance on codes previously unseen during training. These results highlight the promise of Bayesian-attention GNNs for scalable quantum decoding. Despite these advantages, our approaches still have limitations, which are discussed in Appendix A.8.

REFERENCES

Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information fusion*, 76:243–297, 2021.

Paul Baireuther, Thomas E O'Brien, Brian Tarasinski, and Carlo WJ Beenakker. Machine-learning-assisted correction of correlated qubit errors in a topological code. *arXiv preprint arXiv:1705.07855*, 2017.

Paul Baireuther, Thomas E O'Brien, Brian Tarasinski, and CWJ Beenakker. Neural network decoder for topological color codes with circuit level noise. *Quantum*, 2:48, 2018.

John Blue, Harshil Avlani, Zhiyang He, Liu Ziyin, and Isaac L Chuang. Machine learning decoding of circuit-level noise for bivariate bicycle codes. *arXiv preprint arXiv:2504.13043*, 2025.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pp. 1613–1622. PMLR, 2015.

Sergey Bravyi, Andrew W Cross, Jay M Gambetta, Dmitri Maslov, Patrick Rall, and Theodore J Yoder. High-threshold and low-overhead fault-tolerant quantum memory. *Nature*, 627(8005):778–782, 2024.

Nikolas P Breuckmann and Jens N Eberhardt. Balanced product quantum codes. *IEEE Transactions on Information Theory*, 67(10):6653–6674, 2021.

Nikolas P Breuckmann and Barbara M Terhal. Constructions and noise threshold of hyperbolic surface codes. *IEEE transactions on Information Theory*, 62(6):3731–3744, 2016.

A Robert Calderbank and Peter W Shor. Good quantum error-correcting codes exist. *Physical Review A*, 54(2):1098, 1996.

Yoni Choukroun and Lior Wolf. Error correction code transformer. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 23390–23402, 2022.

Yoni Choukroun and Lior Wolf. Deep quantum error correction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 64–72, 2024.

Dimitris Chytas, Michele Pacenti, Nithin Raveendran, Mark F Flanagan, and Bane Vasić. Enhanced message-passing decoding of degenerate quantum codes utilizing trapping set dynamics. *IEEE Communications Letters*, 28(3):444–448, 2024.

Shy-el Cohen, Yoni Choukroun, and Eliya Nachmani. Hybrid mamba-transformer decoder for error-correcting codes. *arXiv preprint arXiv:2505.17834*, 2025.

Michael H Freedman, David A Meyer, and Feng Luo. Z2-systolic freedom and quantum codes. In *Mathematics of quantum computation*, pp. 303–338. Chapman and Hall/CRC, 2002.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Anqi Gong, Sebastian Cammerer, and Joseph M. Renes. Graph neural networks for enhanced decoding of quantum ldpc codes. In *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2024a.

Anqi Gong, Sebastian Cammerer, and Joseph M Renes. Toward low-latency iterative decoding of qldpc codes under circuit-level noise. *arXiv preprint arXiv:2403.18901*, 2024b.

Daniel Gottesman. *Stabilizer codes and quantum error correction*. California Institute of Technology, 1997.

Alex Graves. Practical variational inference for neural networks. *Advances in neural information processing systems*, 24, 2011.

Zichang He, David Amaro, Ruslan Shaydulin, and Marco Pistoia. Performance of quantum approximate optimization with quantum error detection. *Communications Physics*, 8(1):217, 2025.

Timo Hillmann, Lucas Berent, Armanda O Quintavalle, Jens Eisert, Robert Wille, and Joschka Roffe. Localized statistics decoding for quantum low-density parity-check codes. *Nature Communications*, 16(1):8214, 2025.

Samyak Jain, Sravanti Addepalli, Pawan Kumar Sahu, Priyam Dey, and R Venkatesh Babu. Dart: Diversify-aggregate-repeat training improves generalization of neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16048–16059, 2023.

David Kielpinski, Chris Monroe, and David J Wineland. Architecture for a large-scale ion-trap quantum computer. *Nature*, 417(6890):709–711, 2002.

Emanuel Knill and Raymond Laflamme. Theory of quantum error-correcting codes. *Physical Review A*, 55(2):900, 1997.

Stergios Koutsioumpas, Hasan Sayginel, Mark Webster, and Dan E Browne. Automorphism ensemble decoding of quantum ldpc codes. *arXiv preprint arXiv:2503.01738*, 2025.

Alexey A Kovalev and Leonid P Pryadko. Quantum kronecker sum-product low-density parity-check codes with finite rate. *Physical Review A—Atomic, Molecular, and Optical Physics*, 88(1):012311, 2013.

Frank R Kschischang, Brendan J Frey, and H-A Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2002.

Kao-Yueh Kuo and Ching-Yi Lai. Exploiting degeneracy in belief propagation decoding of quantum codes. *npj Quantum Information*, 8(1):111, 2022.

Moritz Lange, Pontus Havström, Basudha Srivastava, Isak Bengtsson, Valdemar Bergentall, Karl Hammar, Olivia Heuts, Evert van Nieuwenburg, and Mats Granath. Data-driven decoding of quantum error correcting codes using graph neural networks. *Physical Review Research*, 7(2):023181, 2025.

Anthony Leverrier and Gilles Zémor. Quantum tanner codes. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 872–883. IEEE, 2022.

Gang Liu, Tong Zhao, Jiaxin Xu, Tengfei Luo, and Meng Jiang. Graph rationalization with environment-based augmentations. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1069–1078, 2022.

Ye-Hua Liu and David Poulin. Neural belief-propagation decoders for quantum error-correcting codes. *Physical review letters*, 122(20):200501, 2019.

Christos Louizos and Max Welling. Structured and efficient variational deep learning with matrix gaussian posteriors. In *International conference on machine learning*, pp. 1708–1716. PMLR, 2016.

Arshpreet Singh Maan and Alexandru Paler. Machine learning message-passing for the scalable decoding of qldpc codes. *npj Quantum Information*, 11(1):78, 2025.

Sisi Miao, Alexander Schnerring, Haizheng Li, and Laurent Schmalen. Neural belief propagation decoding of quantum ldpc codes using overcomplete check matrices. *arXiv preprint arXiv:2212.10245*, 2022.

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

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

Vukan Ninkovic, Ognjen Kundacina, Dejan Vukobratovic, Christian Häger, and Alexandre Graell i Amat. Decoding quantum ldpc codes using graph neural networks. In *GLOBECOM 2024-2024 IEEE Global Communications Conference*, pp. 3479–3484. IEEE, 2024.

Josias Old and Manuel Rispler. Generalized belief propagation algorithms for decoding of surface codes. *arXiv preprint arXiv:2212.03214*, 2022.

Pavel Panteleev and Gleb Kalachev. Degenerate quantum ldpc codes with good finite length performance. *Quantum*, 5:585, 2021.

Jasper Johannes Postema and Servaas JJMF Kokkelmans. Existence and characterisation of bivariate bicycle codes. *arXiv preprint arXiv:2502.17052*, 2025.

David Poulin. Optimal and efficient decoding of concatenated quantum block codes. *Physical Review A—Atomic, Molecular, and Optical Physics*, 74(5):052333, 2006.

David Poulin and Yeojin Chung. On the iterative decoding of sparse quantum codes. *arXiv preprint arXiv:0801.1241*, 2008.

Nithin Raveendran and Bane Vasić. Trapping sets of quantum ldpc codes. *arXiv preprint arXiv:2012.15297*, 2020.

Nir Raviv, Avi Caciularu, Tomer Raviv, Jacob Goldberger, and Yair Be'ery. perm2vec: Graph permutation selection for decoding of error correction codes using self-attention. *arXiv preprint arXiv:2002.02315*, 2020.

Joschka Roffe, David R White, Simon Burton, and Earl Campbell. Decoding across the quantum low-density parity-check code landscape. *Physical Review Research*, 2(4):043423, 2020.

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, 2008.

Jean-Pierre Tillich and Gilles Zémor. Quantum ldpc codes with positive rate and minimum distance proportional to the square root of the blocklength. *IEEE Transactions on Information Theory*, 60(2):1193–1202, 2013.

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

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Hanrui Wang, Pengyu Liu, Kevin Shao, Dantong Li, Jiaqi Gu, David Z Pan, Yongshan Ding, and Song Han. Transformer-qec: quantum error correction code decoding with transferable transformers. *arXiv preprint arXiv:2311.16082*, 2023.

Ming Wang and Frank Mueller. Coprime bivariate bicycle codes and their properties. *arXiv e-prints*, pp. arXiv–2408, 2024.

Ming Wang, Ang Li, and Frank Mueller. Fully parallelized bp decoding for quantum ldpc codes can outperform bp-osd. *arXiv preprint arXiv:2507.00254*, 2025.

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, 2020.

Hanwen Yao, Waleed Abu Laban, Christian Häger, Alexandre Graell i Amat, and Henry D Pfister. Belief propagation decoding of quantum ldpc codes with guided decimation. In *2024 IEEE International Symposium on Information Theory (ISIT)*, pp. 2478–2483. IEEE, 2024.

Xinyu Ye, Ge Yan, and Junchi Yan. Towards quantum machine learning for constrained combinatorial optimization: a quantum qap solver. In *International Conference on Machine Learning*, pp. 39903–39912. PMLR, 2023.

Keyi Yin, Xiang Fang, Jixuan Ruan, Hezi Zhang, Dean Tullsen, Andrew Sornborger, Chenxu Liu, Ang Li, Travis Humble, and Yufei Ding. Symbreak: Mitigating quantum degeneracy issues in qldpc code decoders by breaking symmetry. *arXiv preprint arXiv:2412.02885*, 2024.

Gilles Zémor. On cayley graphs, surface codes, and the limits of homological coding for quantum error correction. In *International Conference on Coding and Cryptology*, pp. 259–273. Springer, 2009.

Weilei Zeng and Leonid P Pryadko. Higher-dimensional quantum hypergraph-product codes with finite rates. *Physical review letters*, 122(23):230501, 2019.

A Appendix

A.1 Attention Mechanisms

Attention mechanisms (Vaswani et al., 2017; Veličković et al., 2017) were originally developed in the context of sequence modeling to enable neural networks to dynamically focus on the most relevant parts of their input. In the general form, attention computes a weighted combination of input features, where the weights are determined by a learned compatibility function between a *query* vector and a set of *key* vectors. This allows the model to capture long-range dependencies and context-specific relationships, in contrast to fixed, uniform aggregation rules such as those used in standard message-passing networks. In GNNs, attention enables each node to adaptively modulate the influence of its neighbors, making the aggregation operation content-dependent rather than purely structural.

Multi-head attention: A single attention head models one notion of similarity or relevance between elements, which may be insufficient to capture diverse structural patterns in the data. *Multi-head attention* addresses this by performing h independent attention computations (Vaswani et al., 2017; Veličković et al., 2017)

$$\mathrm{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Concat}(\mathbf{O}_1, \ldots, \mathbf{O}_h)\, \mathbf{W}^O, \tag{23}$$

$$\mathbf{O}_i = \mathrm{Attention}(\mathbf{Q}\mathbf{W}_i^Q,\ \mathbf{K}\mathbf{W}_i^K,\ \mathbf{V}\mathbf{W}_i^V), \tag{24}$$

where $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V$ are head-specific projection matrices and $\mathbf{W}^O$ is the output projection. Different heads can specialize in different aspects of the input space (e.g., local neighborhoods, long-range dependencies, or rare structural motifs) leading to a richer learned representation.

Scaled dot-product attention: Given a set of queries $\mathbf{Q}$, keys $\mathbf{K}$, and values $\mathbf{V}$—typically obtained via learned linear projections of node or edge embeddings—the scaled dot-product attention (Vaswani et al., 2017) computes

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\intercal}{\sqrt{d_k}}\right)\mathbf{V}, \tag{25}$$

where d_k is the key dimension. The scaling factor $\sqrt{d_k}$ prevents the dot products from growing too large in magnitude, which could otherwise push the softmax function into saturated regimes and impede gradient-based learning. This formulation allows each query to produce a context-dependent weighting over all values.

Relevance to quantum decoding: In QEC, the Tanner graph representing the code often contains many short cycles and heterogeneous connectivity patterns due to the underlying stabilizer structure (Gottesman, 1997). Not all syndrome-data qubit connections carry equal importance. Some checks may be highly informative about likely error configurations, while others may be weakly correlated or redundant due to degeneracy. Incorporating attention into a GNN decoder allows the network to learn these importance patterns directly from data. *Scaled dot-product attention* enables variable and check nodes to selectively emphasize or suppress messages from particular neighbors based on their learned relevance, while *multi-head attention* allows simultaneous modeling of multiple correlation patterns, e.g., one head focusing on local trapping sets (Raveendran & Vasić, 2020), another on long-range stabilizer dependencies. This adaptive message weighting can mitigate the correlation build-up seen in uniform BP schedules and improve decoding accuracy on complex quantum Tanner graphs.

A.2 Error assumptions

In the literature, two types of error decoding methods are commonly considered, namely *uncorrelated decoding* and *correlated decoding*. The former decodes only the X and Z error channels. However, it cannot directly decode correlated Y errors, which leads the decoder to suboptimal performance.

In more realistic settings, our decoder QuBA employs a correlated decoding strategy. Specifically, we use a *1-bit hot encoding scheme* to represent each error type. For consistency across different methods, we decompose the Y error into its Pauli components, $Y = iXZ$, and treat it as a simultaneous occurrence of X and Z errors.

A.3 Quantum Codes

In this section, we introduce some quantum codes used in this paper.

Bivariate bicycle (BB) codes: BB codes (Bravyi et al., 2024) are Calderbank-Shor-Steane (CSS) quantum LDPC codes defined on a two-dimensional torus with quasi-cyclic structure and bounded stabilizer weight. Let S_{ℓ_x} and S_{ℓ_y} be the $\ell_x \times \ell_x$ and $\ell_y \times \ell_y$ cyclic shift matrices. Define the commuting 2D shift operators

$$x \;=\; S_{\ell_x} \otimes I_{\ell_y}, \qquad y \;=\; I_{\ell_x} \otimes S_{\ell_y}, \qquad xy = yx,$$

which generate translations along the two torus directions. Choose two polynomials $p(x, y)$ and $q(x, y)$ over $\mathbb{F}_2$ (each monomial specifies a shifted copy), and set

$$A \;=\; p(x, y), \qquad B \;=\; q(x, y).$$

The X- and Z-type parity-check matrices of the BB code of length $n = 2\ell_x\ell_y$ are

$$H_X \;=\; \big[\, A \,\big|\, B \,\big], \qquad H_Z \;=\; \big[\, B^\top \,\big|\, A^\top \,\big].$$

Because x and y commute, A and B commute in the group algebra, which implies $H_X H_Z^\top = AB^\top + BA^\top = 0 \pmod 2$. Each stabilizer is a cyclic shift of the base patterns defined by p and q, the row/column weight equals the number of monomials in the corresponding polynomial, and the Tanner graph is quasi-cyclic with block size $\ell_x\ell_y$. BB codes achieve finite rate and distance scaling $\Theta(\sqrt{n})$ while preserving 2D locality.

Co-prime BB codes: Co-prime BB (Wang & Mueller, 2024) codes are a subclass of BB codes where the torus dimensions ℓ_x and ℓ_y are coprime. In this case, the 2D shift group $\mathbb{Z}_{\ell_x} \times \mathbb{Z}_{\ell_y}$ is cyclic of order $\ell_x\ell_y$, allowing the bivariate polynomials to be mapped to univariate polynomials over a single $(\ell_x\ell_y)$-cycle. This enables an algebraic prediction of the number of logical qubits without exhaustive search.

With $\gcd(\ell_x, \ell_y) = 1$, one can define a univariate shift T of length $\ell_x\ell_y$ such that $X = T^{\ell_y}$ and $Y = T^{\ell_x}$. The generators $p(X, Y)$ and $q(X, Y)$ become univariate polynomials $p'(T)$, $q'(T)$ in $\mathbb{F}_2[T]/(T^{\ell_x\ell_y} - 1)$. The stabilizer matrices retain the same block form

$$H_X = \big[\, p'(T) \,\big|\, q'(T) \,\big], \quad H_Z = \big[\, q'(T) \,\big|\, p'(T) \,\big],$$

but the dimension k can be computed directly from

$$\gcd\big(p'(T), q'(T), T^{\ell_x\ell_y} - 1\big).$$

A.4 TRAINING DETAILS

This section summarizes the datasets and training hyperparameters used across all models.

Data: Each model (including those trained on BB codes, coprime BB codes, and within the SAGU framework) was trained using paired error-syndrome data generated across a range of physical error rates p. Errors were represented using a *1-bit hot encoding* scheme for X, Y, and Z errors on data qubits. The values of p were sampled uniformly from the interval $[0, p_{\max}]$, where $p_{\max}$ is chosen close to the theoretical noise threshold of the QEC code.

Hyperparameters: We first report the set of hyperparameters that are shared across all model variants. Model-specific hyperparameters are then detailed separately. Tab. 1 summarizes the training configurations for BB and coprime BB codes, while Tab. 2 provides the hyperparameters for training the SAGU model. *Common hyperparameters:* All models are trained using PyTorch's distributed data parallel (DDP) framework on a workstation equipped with three A5000 Ada GPUs. Each node is initialized with $n_{\text{node\_inputs}} = 4$ input features, and the final Bayesian output layer produces predictions of size $n_{\text{node\_outputs}} = 4$. The number of attention heads is set to 4. Dropout rates are fixed at 0.1 for both the message network (msg_net) and the LSTM. The maximum physical error rate is $p_{\max} = 0.15$, and the test error rate is fixed at $p_{\text{test}} = 0.05$. An AdamW optimizer is employed, with weight decay 10^{-4} and learning rates specified separately in the corresponding tables for each model. The batch size is set to 16. The loss function, described in Eq. 22, incorporates KL annealing over 10 epochs with a final scaling factor of 10^{-5}. Training is performed using automatic mixed precision (AMP), and gradient clipping is applied with threshold $\|g\| \leq 1.0$. Early stopping is triggered when the total logical error rate (LER_{tot}) reaches zero, or if no improvement in LER_{tot} is observed over 20 consecutive epochs.

Code	n_{iters}	n_{node}	n_{edge}	Msg_net size	Train size	Test size	LR
BB codes							
$[[90, 8, 10]]$	40	64	32	256	50,000	3,000	5×10^{-4}
$[[144, 12, 12]]$	50	32	32	256	80,000	4,000	5×10^{-4}
$[[288, 12, 18]]$	65	64	32	128	100,000	5,000	5×10^{-4}
$[[756, 16, \leq 34]]$	50	64	32	128	50,000	3,000	5×10^{-4}
Coprime BB codes							
$[[30, 4, 6]]$	40	32	32	256	30,000	1,000	5×10^{-4}
$[[154, 6, 16]]$	60	64	32	128	100,000	5,000	5×10^{-4}

Table 1: Hyperparameters for BB and Coprime BB codes during training.

Phase	BB code	n_{iters}	n_{node}	n_{edge}	Msg_net size	Train size	Test size	LR
Warm-up	$[[72, 12, 6]]$	35	64	32	128	24,000	1,200	5×10^{-4}
Diversify-Aggregate	$[[90, 8, 10]]$	40	64	32	128	6,000	300	5×10^{-4}
	$[[144, 12, 12]]$	50	64	32	128	8,000	400	5×10^{-4}
	$[[288, 12, 18]]$	65	64	32	128	10,000	500	5×10^{-4}
Consolidation	$[[288, 12, 18]]$	50	64	32	128	24,000	1,200	1×10^{-4}

Table 2: Hyperparameters across the training phases in the SAGU schedule. The total training budget is $E_{\text{total}} = 90$ epochs, with a warm-up $E_w = 20$ and a mid-phase $E_m = 50$. Aggregation occurs every $\lambda = 10$ epochs using weighted averaging (weights $[0.1, 0.2, 0.7]$). A StepLR scheduler is used with warm-up step $\lfloor 2/3 E_w \rfloor$, domain step $\lfloor 2/3(E_m - E_w) \rfloor$, final step $= 10$, and $\gamma = 0.5$.

A.5 COMPARATIVE DETAILS

In this section, more comparative settings and details of the experiments are provided.

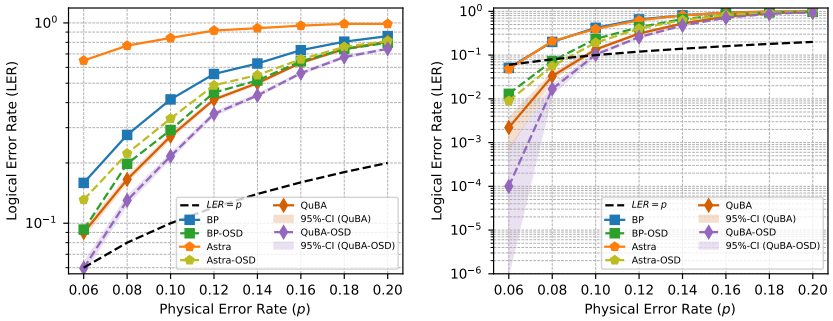
For a fair comparison, Astra and QuBA were trained using identical hyperparameters and the same training and test datasets. Comparisons with BP were performed on the same test sets. To balance computational depth, we allowed twice as many message-passing iterations in the learned models (Astra, QuBA, and SAGU) as in BP, since BP performs bidirectional message updates, while the learned models employ unidirectional message passing, i.e., from syndrome nodes to variable nodes. Finally, in experiments with OSD post-processing (applied independently to both X- and Z-decoders), all methods used the same configuration given by `schedule` = `serial`, `bp_method` = `ms`, `ms_scaling_factor` = 0.725, and `osd_method` = `osd0`. Furthermore, we assess SAGU under a domain-shift protocol with four BB domains (starting, diversity and aggregation domains), and an out-of-domain evaluation. We compare our approach against the baseline methods BP and BP-OSD. To evaluate the performance gains of SAGU over the QuBA code-specific training, each domain code is trained using the hyperparameters listed in Tab. 2, with fixed training and testing dataset sizes of 24,000 and 1,200, respectively, for all codes. For the out-of-domain BB code $[[756, 16, \leq 34]]$, the hyperparameters are identical to those of the consolidation phase, except that the learning rate is set to 5×10^{-4} instead of 1×10^{-4}.

A.6 Results on Coprime BB Codes

Similar to standard BB codes, coprime BB codes exhibit a clear trend of decreasing LER as the code size increases. From Fig. 4, the LER reduces from 10^{-1} for the smaller code $[[30, 4, 6]]$ down to nearly 10^{-6} for the larger code $[[154, 6, 16]]$. Correspondingly, the PERs lying below the break-even line ($LER = p$) increase with code size, confirming that larger coprime codes, like their standard BB counterparts, provide stronger error suppression.

Comparison with classical baselines: Across both coprime codes, QuBA consistently outperforms classical BP and BP-OSD. For the larger code $[[154, 6, 16]]$, QuBA maintains approximately an order of magnitude advantage at $p = 0.06$ compared to both BP and BP-OSD. *Comparison with Astra:* QuBA also consistently surpasses Astra across both codes, maintaining at least an order of magnitude improvement. When evaluated under confidence bounds (i.e., conservative decision-making), this advantage becomes even more pronounced. Moreover, QuBA outperforms Astra-OSD as well. Specifically, for $[[154, 6, 16]]$, the margin widens to nearly two orders of magnitude under confidence-bound evaluation. *Effect of OSD:* With OSD post-processing, QuBA-OSD achieves the strongest performance among all tested decoders, exceeding BP-OSD and Astra-OSD by roughly one order of magnitude across both codes. For the larger coprime code $[[154, 6, 16]]$, QuBA-OSD converges fully at $p = 0.06$, reaching 0.00000 ± 0.00000, thus demonstrating its robustness and reliability.

Overall, QuBA and QuBA-OSD exhibit consistent improvements for coprime BB codes, mirroring the trends observed in standard BB codes. These results underscore both the scalability of QuBA and its ability to maintain reliable error suppression across different code constructions.



(a) $[[30, 4, 6]]$ (b) $[[154, 6, 16]]$

Figure 4: Logical error rate (LER) vs. physical error rate (p) for coprime BB codes.

A.7 BB vs. Coprime BB Codes

It is instructive to compare standard BB codes with their coprime counterparts at similar distances and comparable block lengths, as this sheds light on whether or not the algebraic simplifications inherent in coprime constructions (see Appendix A.3) influence decoder performance.

Comparison at intermediate sizes: For BB $[[144, 12, 12]]$ and coprime BB $[[154, 6, 16]]$, the decoding results without OSD are remarkably similar. E.g., at $p = 0.06$, BP yields LERs of 0.049 and 0.051, respectively. QuBA shows nearly identical performance as well with $1.17 \times 10^{-3} \pm 9.1 \times 10^{-4}$ for the BB code versus $2.20 \times 10^{-3} \pm 1.5 \times 10^{-3}$ for the coprime BB code. Even when accounting for confidence intervals and conservative decision bounds, the two codes yield comparable results. *Effect of OSD:* With OSD post-processing, all methods again exhibit very similar performance. For BB $[[144, 12, 12]]$, BP-OSD, Astra-OSD, and QuBA-OSD all converge to zero at $p = 0.06$. For the coprime BB code $[[154, 6, 16]]$, the corresponding results are 0.013 (BP-OSD), 0.009 (Astra-OSD), and $1.0 \times 10^{-4} \pm 6.0 \times 10^{-4}$ (QuBA-OSD). Under confidence-bound evaluation, QuBA-OSD also converges to zero.

Overall, the BB code exhibits marginally stronger performance, though differences across decoding methods remain small. These findings confirm that coprime BB codes preserve the favorable decoding behavior of standard BB codes while offering structural advantages such as algebraic simplification. Both families achieve nearly identical results, suggesting that general-purpose models such as SAGU can generalize to coprime BB codes as effectively as to standard BB codes.

A.8 Limitations

Our work has two main limitations.

Runtime overhead. In QuBA's architecture, BNNs are employed instead of linear layers to quantify model uncertainty, which inevitably introduces additional computational cost. At inference time, estimating confidence intervals requires drawing $M = 30$ independent weight samples, resulting in approximately a $30\times$ runtime increase compared to single-pass inference without Monte Carlo sampling. Such overhead makes the current implementation impractical for real-time decoding with contemporary classical hardware, but future hardware advantages may mitigate

this aspect. Nonetheless, on-going work is investigating the design of lightweight, efficient decoders that preserve uncertainty awareness while reducing computational demands.

Circuit-level error models. In this work, we adopt the depolarizing error model, where errors occur only on physical qubits, effectively assuming that circuit-level faults can be absorbed into qubit-level depolarization. However, realistic quantum devices are subject to full circuit-level error processes, including syndrome measurement errors, gate errors, and reset errors, which require more sophisticated circuit-level graphical representations beyond Tanner graphs. Extending QuBA and SAGU to handle such circuit-level noise models remains an important avenue for future work.