Toward Uncertainty-Aware and Generalizable Neural Decoding for Quantum LDPC Codes

Xiangjun Mi, Frank Mueller
North Carolina State University, Raleigh, NC 27695, USA
{xmi, fmueller}@ncsu.edu

***Abstract*—Quantum error correction (QEC) is essential for scalable quantum computing, yet decoding errors via conventional algorithms result 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 QEC codes. To address this gap, we propose a Quantum Bayesian graph Attention decoder (QuBA) that enables expressive error-pattern recognition alongside calibrated uncertainty estimates. Building on QuBA, we further develop a multi-phase training framework with enhanced cross-domain robustness enabling decoding beyond the training set called Sequential Aggregate Generalization under Uncertainty (SAGU). 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 up to a *two orders of magnitude* reduction in logical error rate (LER) under confident-decision bounds on the coprime BB code $[[154, 6, 16]]$; (ii) SAGU achieves decoding performance comparable to or even outperforming QuBA's domain-specific training approach.**

I. Introduction

Quantum error correction (QEC) [1] is an essential paradigm that enables quantum computation at negligible error rates over logical qubits [2]–[4]. 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 [1], [5]. 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 [6], [7]. Early quantum LDPC constructions included hypergraph product codes and hyperbolic codes, which demonstrated that constant-rate quantum codes with growing distance are possible [8]–[11]. 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 [12], [13]. 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 [14], 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 [15].

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 [16], [17]. However, the abundance of *short cycles* in Tanner graphs of quantum codes can severely degrade BP performance [18], while *degeneracy* (i.e., multiple distinct errors corresponding to the same syndrome) can trap BP in symmetric belief states [19], [20]. To address these limitations, several variants have been proposed. Memory-based BP introduces additional memory effects [21], while SymBreak explicitly mitigates degeneracy [22]. Other improvements include guided decimation [23], sliding window decoding [24], automorphism-ensemble decoding [25], relay-BP [26], and speculative approaches [27]. Post-processing techniques have also been developed, such as ordered statistics decoding (OSD) [28], which improves performance but at cubic computational cost and limited parallelism, as well as the more recent localized statistics decoding (LSD) [29], which offers a parallelizable alternative. More recently, the QEC community has turned toward advanced machine learning (ML)-based decoders, ranging from (recurrent or graph) neural networks [30]–[33] and generative decoding [34] to transformer architectures [35], [36], providing a new perspective on decoding techniques.

Another crucial challenge in decoding is *uncertainty*. 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) [28], etc.

Despite these advances, existing ML decoders still face

two critical limitations for practical fault tolerance. First, they generally lack reliable uncertainty quantification, 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:

- To the best of our knowledge, this work is the first in quantum ML-based decoding to jointly model and quantify both data and model uncertainty. To this end, we propose QuBA, which leverages Bayesian neural networks (BNNs) to represent predictive uncertainty, using Monte Carlo dropout at inference time to produce calibrated confidence estimates that enable adaptive decision-making. To better capture correlations in error syndromes, QuBA further integrates an attention mechanism into a graph neural network architecture, enhancing relational reasoning on Tanner graphs (see Sec. IV).
- Beyond QuBA, and inspired by the DART paradigm [37], we design SAGU (Sequential Aggregate Generalization under Uncertainty), a cross-domain training framework that consists of three phases (see Sec. V) 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. E.g., both QuBA and SAGU consistently outperform the classical baseline BP, achieving, on average, *one order of magnitude* reduction in LER and up to *two orders of magnitude* under confident-decision bounds on the coprime BB code $[[154, 6, 16]]$. In addition, QuBA surpasses state-of-the-art neural decoders, providing roughly a *one order of magnitude* advantage (e.g., for the larger BB code $[[756, 16, \leq 34]]$) even under conservative (safe) decision bounds.

II. Related Work

Recent research on 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.

1) 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 [30], [38], [39]. This line of work was extended to QEC with neural BP decoders, which adapt message-passing rules to handle degeneracy in quantum LDPC codes [31]. *A second direction* leverages message-passing mechanisms in 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 [40]–[42]. [43] introduces a hypergraph neural decoder with a two-stage message-passing mechanism for hypergraph product codes, achieving up to an 84% reduction in LER compared to BP. By combining structural priors with trainable neural layers, model-based decoders achieve high performance while usually retaining scalability and interpretability.

2) 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 for topological color codes, demonstrating the feasibility of purely supervised decoders under circuit-level noise [44]. 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 [45]–[47]. Building on this line of work, a recurrent transformer decoder was recently applied to BB codes, where a multi-stage training protocol enabled effective decoding under circuit-level noise [48]. In addition, [49] proposes a diffusion-based decoder that directly predicts the LER from detectors in STIM [50], as in the transformer method, and demonstrates performance improvements. 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 [51].

3) Our approach: Our proposed method, QuBA, belongs to the model-based category, as it builds on neural 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 classical and related ML-based approaches.

III. Background

A. Quantum Decoding

In the stabilizer formalism [6], 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

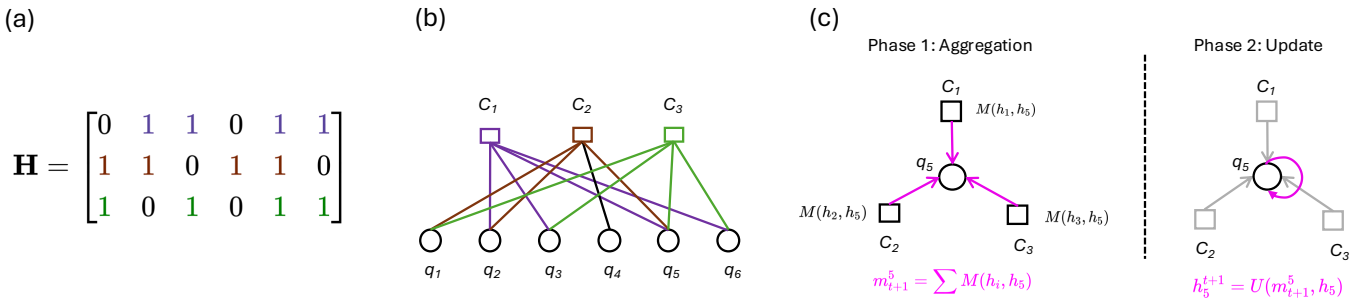


Fig. 1: Relationship between belief propagation (BP) and message passing in graph neural networks (GNNs). **(a)**. A parity-check matrix. **(b)**. The Tanner graph constructed from the parity-check matrix, showing the connections between check nodes and variable nodes (physical qubits). BP operates by 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.

S_i. An error E anticommutes with a subset of stabilizers, producing a binary syndrome vector

$$S_i = \begin{cases} 0, & ES_i = S_iE, \\ 1, & ES_i = -S_iE, \end{cases} \qquad j = 1, \ldots, n-k. \tag{1}$$

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. Given a 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 belief information.

B. Graph Neural Networks

GNNs [52]–[54] extend deep learning to graph-structured data by iteratively exchanging and aggregating information between neighboring nodes. In *message-passing neural network* (MPNN) [55], the hidden state of each node v at iteration t is updated as

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

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 propagate and transform messages in a manner that approximates maximum-likelihood decoding, thereby mitigating key limitations of BP in graphs with cycles. Specifically, GNN-based message passing relies on learned state updates, whereas BP performs iterative belief exchanges that accumulate toward a final hard decision. In the presence of short cycles, these fixed update rules in BP can give rise to trapping sets and degeneracy-induced symmetries, leading to oscillatory behavior or biased belief accumulation [56], [57].

IV. QuBA: A Quantum Bayesian Attention Decoder

A. Uncertainty Representation

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:

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

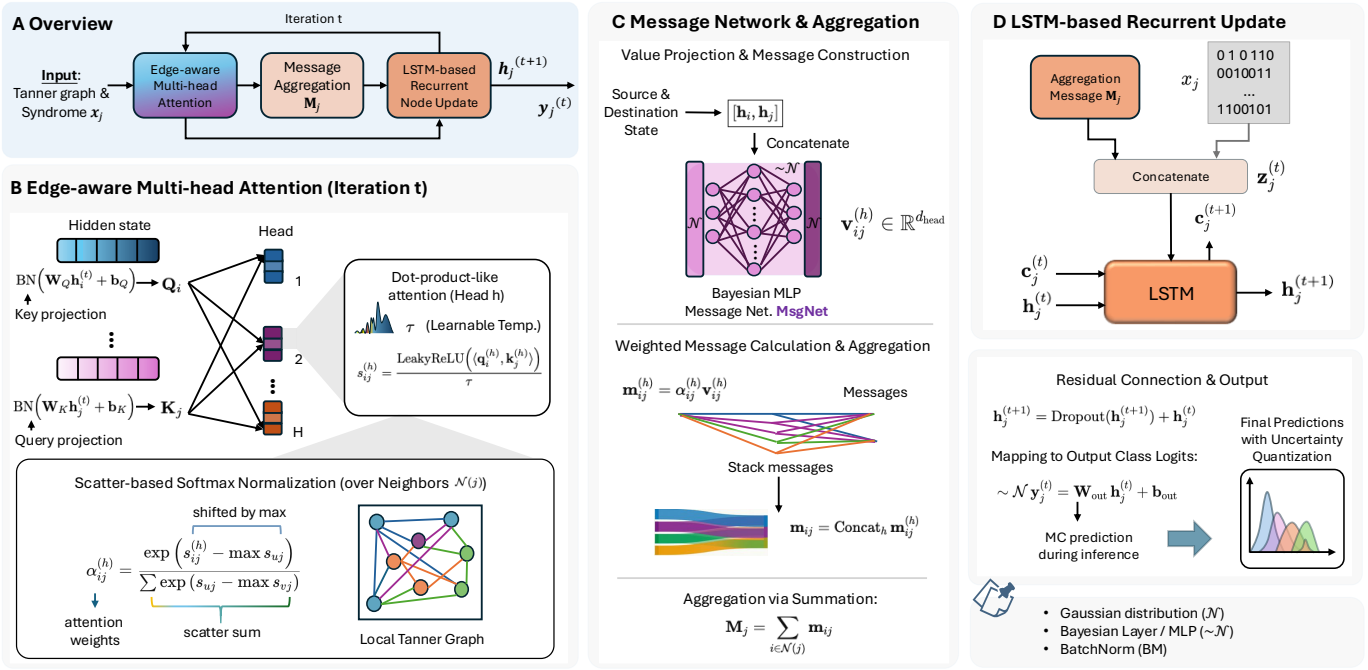


Fig. 2: Architecture of QuBA.

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 [58].

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 $q_\phi(\theta)$ [59]–[61], 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.

a) Monte Carlo (MC) prediction: At inference, we perform M independent MC forward passes. During each pass m, the decoder evaluates the same syndrome $\mathbf{s}$ while drawing fresh layerwise weight samples from $q_\phi(\theta)$ at each internal iteration (i.e., weights are resampled within the forward). This produces predictions $\hat{\mathbf{e}}^{(m)}$, whose empirical mean and variance are

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

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.

B. 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.

a) 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.

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

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

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

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

c) 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

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

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

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

e) 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

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

f) 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, (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. Fig. 2 shows an overview of our proposed QuBA and detailed modules in it.

C. Loss Function

In QEC, let $\mathbf{e} \in \{0,1\}^{2n}$ denote the true Pauli error in binary symplectic form, and let $\hat{\mathbf{e}}$ denote the decoder's predicted correction. The residual error after applying the correction is $\mathbf{e}_{\text{tot}} = \mathbf{e} + \hat{\mathbf{e}} \pmod 2$. Decoding succeeds when this residual is trivial up to stabilizers, so that it produces neither syndrome defects (i.e., stabilizer violations) nor a nontrivial logical effect on the encoded state.

a) Logical-consistency loss: To encourage globally valid corrections during training, QuBA introduces a differentiable loss term based on the residual error. Since exact mod-2 parity checks are non-differentiable, the implementation replaces them with the smooth surrogate $f(x) = \left|\sin\left(\frac{\pi}{2}x\right)\right|$ [31]. The decoder outputs logits over the four Pauli classes $\{I, X, Z, Y\}$ for each data qubit. These logits are first converted by a softmax into class probabilities. The resulting probabilities are then used to form soft X- and Z-components of the predicted correction, which are combined with the ground-truth X- and Z-components of $\mathbf{e}$ to obtain a soft version of the residual $\mathbf{e}_{\text{tot}}$. The resulting soft residual is projected onto code-constraint subspaces using the orthogonal-complement bases $H_X^{\perp}$ and $H_Z^{\perp}$, yielding the differentiable loss term $\mathcal{L}_{\text{LER}}$. This term penalizes residual errors that are inconsistent with the code-level constraint structure.

b) Error cross-entropy loss: We additionally supervise the predicted Pauli label on each data qubit with a standard cross-entropy loss, denoted $\mathcal{L}_{\text{CE,e}}$. This term promotes accurate local identification of the error type.

c) Syndrome cross-entropy loss: Because the QuBA forward pass does not explicitly enforce syndrome consistency, we also apply a cross-entropy loss $\mathcal{L}_{\text{CE,s}}$ on the syndrome-node outputs. This term encourages agreement between the predicted and measured syndromes.

d) Overall objective: The final objective averages the task losses across the T decoding iterations and adds the KL regularizer from the Bayesian neural network formulation:

$$\begin{aligned}\mathcal{L}(\theta) = \frac{1}{T}\sum_{t=1}^{T}\Big(\mathcal{L}_{\text{LER}}^{(t)} + \tfrac{1}{2}\mathcal{L}_{\text{CE,e}}^{(t)} + \tfrac{1}{2}\mathcal{L}_{\text{CE,s}}^{(t)}\Big) \\ + \beta(\tau)\,\mathrm{KL}(q_\phi(\theta) \,\|\, p(\theta))\,,\end{aligned} \tag{21}$$

where $\beta(\tau)$ is annealed during training to gradually introduce Bayesian regularization.

In summary, this composite loss encourages the decoder to generate corrections that are not only statistically accurate but also logically valid under the stabilizer formalism, while explicitly modeling epistemic uncertainty through Bayesian parameterization.

V. SAGU: Generalizable Training

To address the challenge of learning a unified decoding strategy across heterogeneous quantum code families, we develop a framework that jointly captures variations in code constructions and data distributions to ease effects such as trapping sets and quantum degeneracy by leveraging the diversity across different codes. To this end, we formalize

this setting through a Bayesian graph decoding framework that is inherently robust to domain shifts across quantum codes, and introduce SAGU (*Sequential Aggregate Generalization under Uncertainty*). SAGU enables principled, uncertainty-aware generalization across diverse code families.

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., $|\mathcal{D}_{\text{warm}}| = |\mathcal{D}_{\text{con.}}| = \sum_{k=1}^{M} |\mathcal{D}_k|$, where $\mathcal{D}_{\text{warm}}$ and $\mathcal{D}_{\text{con.}}$ 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. 21. 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 $\mathcal{D}_{\text{warm}}$ 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 $\theta_k \leftarrow \theta_{\text{start}}$. For each epoch $\tau \in [E_w, E_m)$, 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 $[E_m, E_{\text{tot}})$ 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 logical error rate (LER) fails to improve within a patience window or when the LER completely converges (i.e., reaches zero). Fig. 3 presents the SAGU pipeline.

Algorithm 1 Sequential Aggregate Generalization under Uncertainty (SAGU)

Input: Datasets $\mathcal{D}_{\text{warm}}, \mathcal{D}_{\text{con.}}, \{\mathcal{D}_k\}_{k=1}^{M}$; aggregation weights $\{w_k\}_{k=1}^{M}$; epoch budgets $E_w < E_m \leq E_{\text{tot}}$; aggregation interval λ.
Output: Final decoder parameters θ_{final}.
1. Warm-up Phase:
Initialize starting-domain decoder $f_{\theta_{\text{start}}}$.
for $\tau = 0$ **to** $E_w - 1$ **do**
  Train $f_{\theta_{\text{start}}}$ for one epoch on $\mathcal{D}_{\text{warm}}$.
end for
2. Diversify–Aggregate Phase:
Initialize M diversity domain decoders $\{f_{\theta_k}\}_{k=1}^{M}$: $\theta_k \leftarrow \theta_{\text{start}}$, $k = 1, \ldots, M$.
for $\tau = E_w$ **to** $E_m - 1$ **do**
  for $k = 1$ **to** M **do**
    Train f_{θ_k} on domain $\mathcal{D}_k$ for one epoch.
  end for
  if $(\tau + 1 - E_w) \bmod \lambda = 0$ **or** $\tau = E_m - 1$ **then**
    Aggregate: $\bar{\theta} \leftarrow \sum_{k=1}^{M} w_k \theta_k$.
    Synchronize: $\theta_k \leftarrow \bar{\theta}$ for all k.
  end if
end for
3. Consolidation Phase:
Initialize aggregation-domain decoder $f_{\theta_{\text{final}}}$ with $\bar{\theta}$.
for $\tau = E_m$ **to** $E_{\text{tot}} - 1$ **do**
  Fine-tune on $\mathcal{D}_{\text{con.}}$ with reduced learning rate.
end for
Return θ_{final}.

VI. Experiments

A. 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 under a capacity noise model. Specifically, we use an i.i.d. single-qubit depolarizing Pauli channel with noiseless syndrome extraction. For a physical error rate p, each data qubit remains error-free with probability $1 - p$, and undergoes an X, Y, or Z error with probabilities $p/3$, $p/3$, and $p/3$, respectively.

During training, p is sampled uniformly from $[0, p_{\max}]$, where $p_{\max}$ is chosen close to the theoretical noise threshold of the corresponding QEC code. During testing, p is fixed to the reported physical error rate. The syndrome/input node features are represented using a one-hot encoding, while the error labels are represented by four integer classes corresponding to $\{I, X, Z, Y\}$. Across all code families and models, results were obtained from over 1,000 syndrome samples.

B. Error Assumptions

In the literature, two types of error decoding methods are commonly considered, namely *uncorrelated decoding* and *correlated decoding* [62]. The former decodes the X and Z error components separately, and therefore does not explicitly model same-qubit correlations arising from Y errors, which may lead to suboptimal performance under depolarizing noise.

QuBA employs a correlated decoding strategy by using a four-class error alphabet corresponding to $\{I, X, Z, Y\}$. In the implementation, the classes are encoded as $0, 1, 2, 3$, where class 3 denotes a Y error. This class is interpreted as the simultaneous occurrence of its Pauli components, since $Y = iXZ$ up to a global phase. Thus, when evaluating residual and logical errors, class 3 contributes to both the X and Z components.

Although errors are independent across different qubits, the induced binary X and Z components on the same qubit are correlated under depolarizing noise. In particular, both

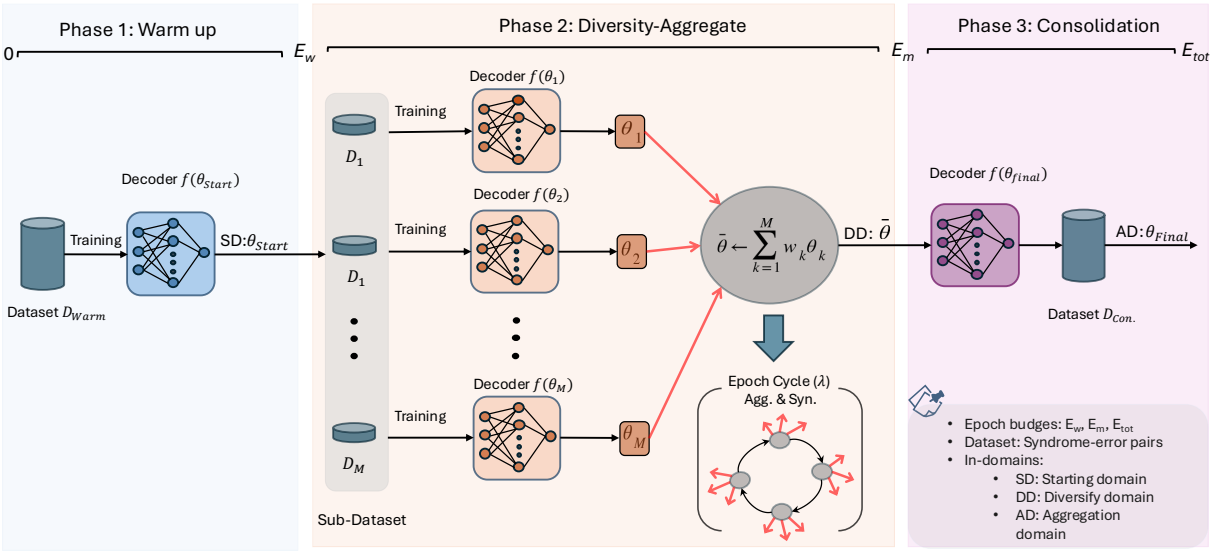


Fig. 3: A diagram of the SAGU pipeline.

$\Pr(e_X = 1)$ and $\Pr(e_Z = 1)$ are equal to $2p/3$, while $\Pr(e_X = e_Z = 1) = p/3$. This same correlated noise model is used for both training and testing.

C. Baselines

We benchmark two classical decoding methods, BP [17] and BP-OSD [28], as well as a state-of-the-art neural decoder, Astra [42], on both BB codes [14] and coprime BB codes [63], [64]. All methods are evaluated under the same capacity depolarizing Pauli channel described above. We then compare these baselines with our proposed decoders, QuBA and SAGU, evaluating performance both with and without OSD post-processing.

D. Training Details

a) Model hyperparameters: We first report the set of hyperparameters that are shared across all model variants. Model-specific hyperparameters are then detailed separately. Tab. I summarizes the training configurations for BB and coprime BB codes, while Tab. II provides the hyperparameters for training the SAGU model.

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 (MsgNet) 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. 21, 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 logical error rate (LER) reaches zero, or if no improvement in LER is observed over 20 consecutive epochs.

b) SAGU training process: Fig. 4 illustrates the training dynamics of SAGU on the validation set. Epochs 0–19 correspond to the warm-up phase, while epochs 50–52 shown in the figure fall within the consolidation phase. The intermediate epochs correspond to the diversity-aggregate phase. The results show that each phase contributes to a steady reduction in the LER, culminating in full convergence by epoch 52, at which point training is early stopped. These observations validate the effectiveness and design rationale of SAGU.

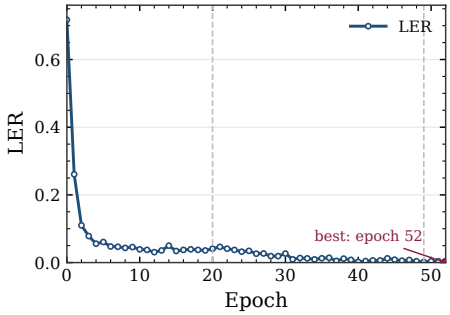


Fig. 4: LER across training epochs for SAGU. For epochs 20–49, the reported value is averaged over 3 decoders. Vertical dashed lines mark epochs 20 and 49.

E. Comparison Details

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

Code	n_{iters}	n_{node}	n_{edge}	MsgNet size	Training size	Validation 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 I: Hyperparameters for BB and Coprime BB codes during training.

Phase	BB code	n_{iters}	n_{node}	n_{edge}	MsgNet size	Training size	Validation 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 II: 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 $\lfloor 2/3E_w \rfloor$, domain step $\lfloor 2/3(E_m - E_w) \rfloor$, final step $= 10$, and $\gamma = 0.5$.

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. II, 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 (LR) is set to 5×10^{-4} instead of 1×10^{-4}.

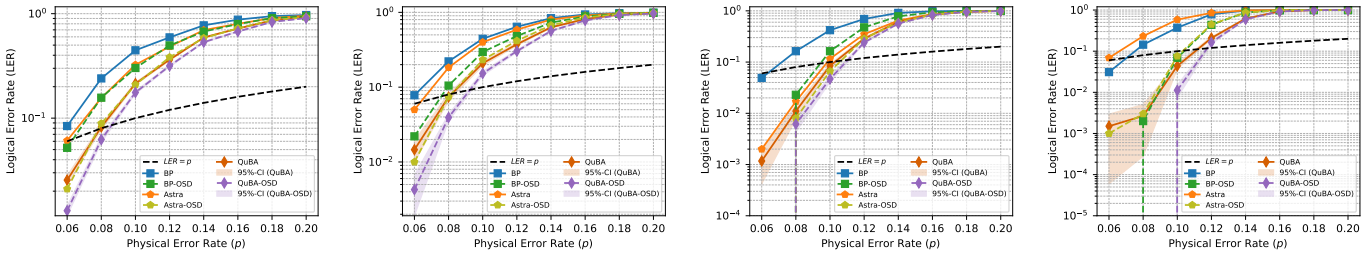
F. Results and Analysis

1) BB Codes: **Overall trends -** Across the BB family, larger codes shift the LER curves downward at fixed PER and move more QuBA/QuBA-OSD points below the break-even line (Fig. 5), with reductions from the 10^{-2} regime on $[[144, 12, 12]]$ to the 10^{-4} regime on $[[288, 12, 18]]$ and further gains on $[[756, 16, \leq 34]]$. This size-dependent shift indicates that the decoder exploits larger code distances rather than merely fitting a fixed-noise regime. For example, $[[90, 8, 10]]$ remains above 10^{-2}, while $[[144, 12, 12]]$ already drops below that level, and larger codes push more PER values below $LER = p$.

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 two 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 nearly 7%, 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.

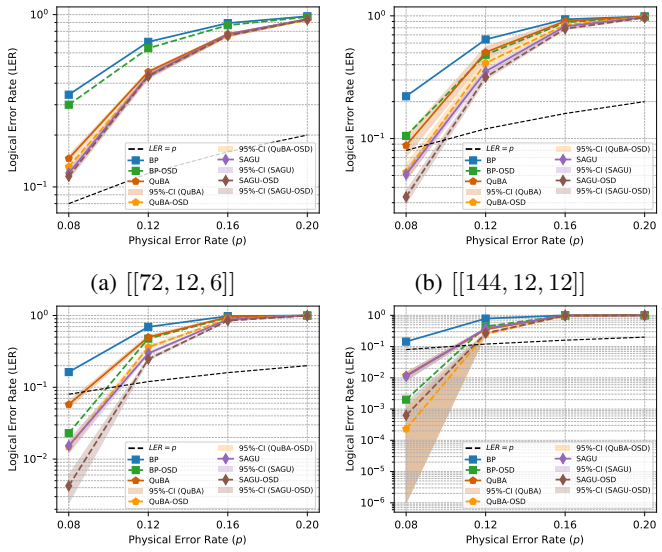
Summary: Overall, QuBA improves over BP and Astra across all BB codes, with advantages ranging from one to three orders of magnitude depending on PER and evaluation setting. With OSD, QuBA-OSD gives the strongest benchmarks, while QuBA without OSD remains competitive against post-processing-enhanced baselines, underscoring the strength of its attention design.

2) SAGU - Domain Generalization: **Overall trends:** Across domains, SAGU lowers LER as code size increases: the starting domain $[[72, 12, 6]]$ remains above 10^{-1}, the diversity domain $[[144, 12, 12]]$ falls below 10^{-1}, the aggregation domain $[[288, 12, 18]]$ reaches about 10^{-2}, and the out-of-domain $[[756, 16, \leq 34]]$ drops to 10^{-6} (Fig. 6). This nearly four-order progression preserves the expected rightward break-even shift and shows that SAGU retains the scaling behavior of domain-specific training while transferring across domains.



(a) $[[90, 8, 10]]$ (b) $[[144, 12, 12]]$ (c) $[[288, 12, 18]]$ (d) $[[756, 16, \leq 34]]$

Fig. 5: Logical vs. physical error rate (LER vs. p) for BB codes. Lines become vertical when all errors are corrected (LER=0).



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

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

Fig. 6: 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 $[[756, 16, \leq 34]]$.

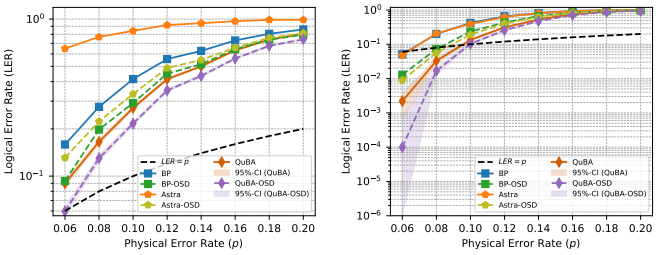
Comparison with BP and BP-OSD: SAGU consistently outperforms both BP and BP-OSD across the in-domain codes $[[72, 12, 6]]$, $[[144, 12, 12]]$, and $[[288, 12, 18]]$, with improvements reaching up to one order of magnitude. For instance, on $[[288, 12, 18]]$ at $p = 0.08$, BP attains an LER of 0.163, 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 $[[756, 16, \leq 34]]$, at $p = 0.08$, SAGU-OSD achieves an order-of-magnitude improvement compared to BP-OSD, and outperforms BP-OSD by two orders of magnitude. **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 improves over BP and BP-OSD on in-domain codes and, with OSD, surpasses QuBA-OSD on nearly all benchmarks. Compared with domain-specific QuBA, SAGU gives modest but consistent gains on smaller and intermediate codes while retaining competitive out-of-domain performance. Overall, these results highlight SAGU's cross-domain generalization, reliability, and scalability for quantum LDPC decoding.

3) Coprime BB Codes: For coprime BB codes, QuBA and QuBA-OSD follow the same size-scaling behavior observed for standard BB codes: LER drops from the 10^{-1} regime on $[[30, 4, 6]]$ to nearly 10^{-6} on $[[154, 6, 16]]$ (Fig. 7). Correspondingly, more PER points fall below the break-even line as code size increases, matching the behavior of standard BB codes. On $[[154, 6, 16]]$ at $p = 0.06$, QuBA maintains approximately an order-of-magnitude advantage over BP and BP-OSD, and confidence-bound evaluation widens its margin over Astra to nearly two orders of magnitude. This advantage also holds against Astra-OSD, showing that the confidence-bound gains are not only a consequence of classical post-processing. With OSD, QuBA-OSD exceeds BP-OSD and Astra-OSD by roughly one order of magnitude and reaches a zero LER lower bound at $p = 0.06$. These results mirror the standard BB-code trends and support QuBA's robustness across related code constructions.

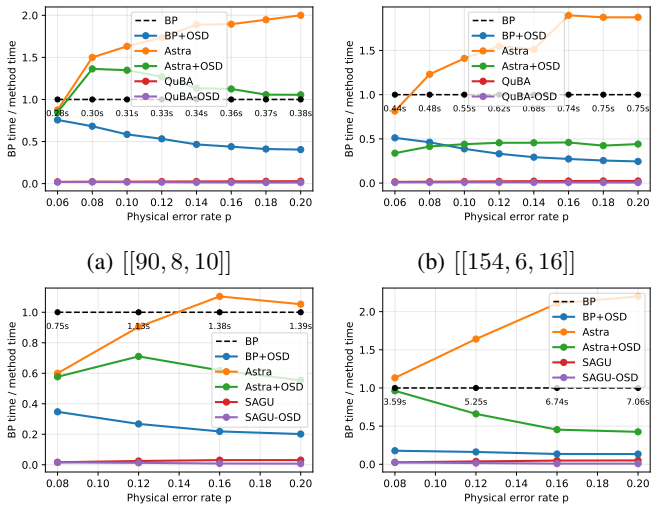
4) BB vs. Coprime BB Codes: It is instructive to compare standard BB codes with their coprime counterparts at similar distances and comparable block lengths, as this sheds light on whether or not the algebraic simplifications inherent in coprime constructions influence decoder performance.

For BB $[[144, 12, 12]]$ (see Fig. 5b) and coprime BB $[[154, 6, 16]]$ (see Fig. 7b), decoding results without OSD are largely comparable across all methods. E.g., at $p = 0.06$, BP yields LERs of 0.078 and 0.051, respectively, which are



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

Fig. 7: Logical vs. physical error rate (LER vs. p) for coprime BB codes.



(a) $[[90, 8, 10]]$ (b) $[[154, 6, 16]]$

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

Fig. 8: Runtime relative to the BP baseline across multiple codes (one BB code, one coprime BB code, and two BB codes for SAGU in-domain and out-of-domain) as a function of the physical error rate p. Values above (below) 1 indicate faster (slower) performance than BP. The dashed horizontal line marks the BP baseline, and the labels along that line show the absolute BP runtime at each p.

of the same order of magnitude. For QuBA, the BB code achieves $1.47 \times 10^{-2} \pm 4.7 \times 10^{-3}$, whereas the coprime BB code yields $2.20 \times 10^{-3} \pm 1.5 \times 10^{-3}$. **Effect of OSD:** With OSD post-processing, all decoding methods again exhibit closely matched performance. For BB $[[144, 12, 12]]$ at $p = 0.06$, the LERs are 0.022 (BP-OSD), 0.01 (Astra-OSD), and $4.27 \times 10^{-3} \pm 2.36 \times 10^{-3}$ (QuBA-OSD). 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 converges to zero.

Overall: Both BB and coprime BB codes demonstrate nearly identical decoding behavior across BP, Astra, and their OSD-enhanced variants, with differences typically remaining within the same order of magnitude. Although the coprime BB code attains a lower central value, the confidence intervals overlap, and both codes fall within a comparable performance regime. Where separations emerge, they modestly favor the coprime BB code, consistent with its lower code rate (BB: 0.0833 vs. Coprime BB: 0.0390), which introduces additional stabilizer constraints and thereby yields a larger code distance (more correctable errors), as well as structural advantages such as algebraic simplification.

5) Runtime: Fig. 8 reports runtime normalized to BP. Average runtime increases with larger block sizes within the same code family. Astra(-OSD) is generally fastest on $[[90, 8, 10]]$; for coprime BB codes, BP-OSD is faster than Astra-OSD for $p \in [0.06, 0.09]$ but slower afterward; under SAGU, QuBA(-OSD) approaches BP-OSD runtime on larger domains while remaining slower than Astra(-OSD).

The overhead reflects QuBA's Bayesian inference: Each prediction uses $M = 30$ MC forward passes, each with n_{iters} message-passing iterations and $L = 7$ Bayesian linear modules, so stochastic parameter draws scale as $M \times n_{\text{iters}} \times L$ (or $M \times n_{\text{iters}} \times 2L$ if biases are counted). This makes inference approximately M times more expensive than deterministic inference, before accounting for Bayesian parameter sampling inside each pass. The resulting accuracy–runtime tradeoff improves confidence estimation and decoding quality, but makes the current implementation less suitable for real-time decoding without lighter Bayesian inference or hardware acceleration. Thus, runtime should be interpreted alongside LER gains.

VII. 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(-OSD) consistently outperforms classical decoders (BP, BP-OSD) and state-of-the-art neural approaches (Astra), with gains reaching up to *two orders of magnitude* accuracy improvement even without considering the confidence-decision bounds, e.g., for the coprime BB $[[154, 6, 16]]$ code. 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 graph networks for scalable quantum decoding. However, given the runtime limitations discussed in Sec. VI-F5, future work will focus on developing a lightweight, uncertainty-aware decoder under the circuit-level error model, potentially leveraging convolutional neural networks [65] to predict logical failures instead of physical errors.

Acknowledgment

This work was supported in part by NSF awards OSI-2531350, OSI-2410675, PHY-2325080, and OMA-2120757.

REFERENCES

[1] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Physical Review A*, vol. 54, no. 2, p. 1098, 1996.

[2] D. Kielpinski, C. Monroe, and D. J. Wineland, "Architecture for a large-scale ion-trap quantum computer," *Nature*, vol. 417, no. 6890, pp. 709–711, 2002.

[3] X. Ye, G. Yan, and J. Yan, "Towards quantum machine learning for constrained combinatorial optimization: a quantum qap solver," in *International Conference on Machine Learning*. PMLR, 2023, pp. 39 903–39 912.

[4] Z. He, D. Amaro, R. Shaydulin, and M. Pistoia, "Performance of quantum approximate optimization with quantum error detection," *Communications Physics*, vol. 8, no. 1, p. 217, 2025.

[5] E. Knill and R. Laflamme, "Theory of quantum error-correcting codes," *Physical Review A*, vol. 55, no. 2, p. 900, 1997.

[6] D. Gottesman, *Stabilizer codes and quantum error correction*. California Institute of Technology, 1997.

[7] J.-P. Tillich and G. Zémor, "Quantum ldpc codes with positive rate and minimum distance proportional to the square root of the blocklength," *IEEE Transactions on Information Theory*, vol. 60, no. 2, pp. 1193–1202, 2013.

[8] M. H. Freedman, D. A. Meyer, and F. Luo, "Z2-systolic freedom and quantum codes," in *Mathematics of quantum computation*. Chapman and Hall/CRC, 2002, pp. 303–338.

[9] G. Zémor, "On cayley graphs, surface codes, and the limits of homological coding for quantum error correction," in *International Conference on Coding and Cryptology*. Springer, 2009, pp. 259–273.

[10] W. Zeng and L. P. Pryadko, "Higher-dimensional quantum hypergraph-product codes with finite rates," *Physical review letters*, vol. 122, no. 23, p. 230501, 2019.

[11] N. P. Breuckmann and B. M. Terhal, "Constructions and noise threshold of hyperbolic surface codes," *IEEE transactions on Information Theory*, vol. 62, no. 6, pp. 3731–3744, 2016.

[12] N. P. Breuckmann and J. N. Eberhardt, "Balanced product quantum codes," *IEEE Transactions on Information Theory*, vol. 67, no. 10, pp. 6653–6674, 2021.

[13] A. Leverrier and G. Zémor, "Quantum tanner codes," in *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2022, pp. 872–883.

[14] S. Bravyi, A. W. Cross, J. M. Gambetta, D. Maslov, P. Rall, and T. J. Yoder, "High-threshold and low-overhead fault-tolerant quantum memory," *Nature*, vol. 627, no. 8005, pp. 778–782, 2024.

[15] J. J. Postema and S. J. Kokkelmans, "Existence and characterisation of bivariate bicycle codes," *arXiv preprint arXiv:2502.17052*, 2025.

[16] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on information theory*, vol. 47, no. 2, pp. 498–519, 2002.

[17] D. Poulin and Y. Chung, "On the iterative decoding of sparse quantum codes," *arXiv preprint arXiv:0801.1241*, 2008.

[18] A. A. Kovalev and L. P. Pryadko, "Quantum kronecker sum-product low-density parity-check codes with finite rate," *Physical Review A—Atomic, Molecular, and Optical Physics*, vol. 88, no. 1, p. 012311, 2013.

[19] D. Poulin, "Optimal and efficient decoding of concatenated quantum block codes," *Physical Review A—Atomic, Molecular, and Optical Physics*, vol. 74, no. 5, p. 052333, 2006.

[20] P. Panteleev and G. Kalachev, "Degenerate quantum ldpc codes with good finite length performance," *Quantum*, vol. 5, p. 585, 2021.

[21] K.-Y. Kuo and C.-Y. Lai, "Exploiting degeneracy in belief propagation decoding of quantum codes," *npj Quantum Information*, vol. 8, no. 1, p. 111, 2022.

[22] K. Yin, X. Fang, J. Ruan, H. Zhang, D. Tullsen, A. Sornborger, C. Liu, A. Li, T. Humble, and Y. Ding, "Symbreak: Mitigating quantum degeneracy issues in qldpc code decoders by breaking symmetry," *arXiv preprint arXiv:2412.02885*, 2024.

[23] H. Yao, W. A. Laban, C. Häger, A. G. i Amat, and H. D. Pfister, "Belief propagation decoding of quantum ldpc codes with guided decimation," in *2024 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2024, pp. 2478–2483.

[24] A. Gong, S. Cammerer, and J. M. Renes, "Toward low-latency iterative decoding of qldpc codes under circuit-level noise," *arXiv preprint arXiv:2403.18901*, 2024.

[25] S. Koutsioumpas, H. Sayginel, M. Webster, and D. E. Browne, "Automorphism ensemble decoding of quantum ldpc codes," *arXiv preprint arXiv:2503.01738*, 2025.

[26] T. Müller, T. Alexander, M. E. Beverland, M. Bühler, B. R. Johnson, T. Maurer, and D. Vandeth, "Improved belief propagation is sufficient for real-time decoding of quantum memory," *arXiv preprint arXiv:2506.01779*, 2025.

[27] M. Wang, A. Li, and F. Mueller, "Fully parallelized bp decoding for quantum ldpc codes can outperform bp-osd," in *2026 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2026, pp. 1–14.

[28] J. Roffe, D. R. White, S. Burton, and E. Campbell, "Decoding across the quantum low-density parity-check code landscape," *Physical Review Research*, vol. 2, no. 4, p. 043423, 2020.

[29] T. Hillmann, L. Berent, A. O. Quintavalle, J. Eisert, R. Wille, and J. Roffe, "Localized statistics decoding for quantum low-density parity-check codes," *Nature Communications*, vol. 16, no. 1, p. 8214, 2025.

[30] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 119–131, 2018.

[31] Y.-H. Liu and D. Poulin, "Neural belief-propagation decoders for quantum error-correcting codes," *Physical review letters*, vol. 122, no. 20, p. 200501, 2019.

[32] S. Miao, A. Schnerring, H. Li, and L. Schmalen, "Neural belief propagation decoding of quantum ldpc codes using overcomplete check matrices," in *2023 IEEE Information Theory Workshop (ITW)*. IEEE, 2023, pp. 215–220.

[33] P. Baireuther, T. E. O'Brien, B. Tarasinski, and C. W. Beenakker, "Machine-learning-assisted correction of correlated qubit errors in a topological code," *Quantum*, vol. 2, p. 48, 2018.

[34] H. Cao, F. Pan, D. Feng, Y. Wang, and P. Zhang, "Generative decoding for quantum error-correcting codes," *arXiv preprint arXiv:2503.21374*, 2025.

[35] H. Wang, P. Liu, K. Shao, D. Li, J. Gu, D. Z. Pan, Y. Ding, and S. Han, "Transformer-qec: quantum error correction code decoding with transferable transformers," *arXiv preprint arXiv:2311.16082*, 2023.

[36] Y. Choukroun and L. Wolf, "Deep quantum error correction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, 2024, pp. 64–72.

[37] S. Jain, S. Addepalli, P. K. Sahu, P. Dey, and R. V. Babu, "Dart: Diversify-aggregate-repeat training improves generalization of neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 16 048–16 059.

[38] E. Nachmani, Y. Be'Ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2016, pp. 341–346.

[39] E. Nachmani and L. Wolf, "Autoregressive belief propagation for decoding block codes," *arXiv preprint arXiv:2103.11780*, 2021.

[40] A. Gong, S. Cammerer, and J. M. Renes, "Graph neural networks for enhanced decoding of quantum ldpc codes," in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2024.

[41] V. Ninkovic, O. Kundacina, D. Vukobratovic, C. Häger, and A. G. i Amat, "Decoding quantum ldpc codes using graph neural networks," in *GLOBECOM 2024-2024 IEEE Global Communications Conference*. IEEE, 2024, pp. 3479–3484.

[42] A. S. Maan and A. Paler, "Machine learning message-passing for the scalable decoding of qldpc codes," *npj Quantum Information*, vol. 11, no. 1, p. 78, 2025.

[43] A. S. Bhave, N. Choudhury, and K. Basu, "Hypernq: A hypergraph neural network decoder for quantum ldpc codes," *arXiv preprint arXiv:2511.01741*, 2025.

[44] P. Baireuther, T. E. O'Brien, B. Tarasinski, and C. Beenakker, "Neural network decoder for topological color codes with circuit level noise," *Quantum*, vol. 2, p. 48, 2018.

[45] N. Raviv, A. Caciularu, T. Raviv, J. Goldberger, and Y. Be'ery, "perm2vec: Graph permutation selection for decoding of error correction codes using self-attention," *arXiv preprint arXiv:2002.02315*, 2020.

[46] Y. Choukroun and L. Wolf, "Error correction code transformer," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, 2022, pp. 23 390–23 402.

[47] S.-e. Cohen, Y. Choukroun, and E. Nachmani, "Hybrid mamba-transformer decoder for error-correcting codes," *arXiv preprint arXiv:2505.17834*, 2025.

[48] J. Blue, H. Avlani, Z. He, L. Ziyin, and I. L. Chuang, "Machine learning decoding of circuit-level noise for bivariate bicycle codes," *Quantum*, vol. 10, p. 2149, 2026.

[49] Z. Liu, A. Gong, and B. K. Clark, "Decoding quantum low density parity check codes with diffusion," *arXiv preprint arXiv:2509.22347*, 2025.

[50] C. Gidney, "Stim: a fast stabilizer circuit simulator," *Quantum*, vol. 5, p. 497, Jul. 2021. [Online]. Available: https://doi.org/10.22331/q-2021-07-06-497

[51] M. Lange, P. Havström, B. Srivastava, I. Bengtsson, V. Bergentall, K. Hammar, O. Heuts, E. van Nieuwenburg, and M. Granath, "Data-driven decoding of quantum error correcting codes using graph neural networks," *Physical Review Research*, vol. 7, no. 2, p. 023181, 2025.

[52] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[53] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[54] G. Liu, T. Zhao, J. Xu, T. Luo, and M. Jiang, "Graph rationalization with environment-based augmentations," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 1069–1078.

[55] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*. PMLR, 2017, pp. 1263–1272.

[56] N. Raveendran and B. Vasić, "Trapping sets of quantum ldpc codes," *Quantum*, vol. 5, p. 562, 2021.

[57] D. Chytas, M. Pacenti, N. Raveendran, M. F. Flanagan, and B. Vasić, "Enhanced message-passing decoding of degenerate quantum codes utilizing trapping set dynamics," *IEEE Communications Letters*, vol. 28, no. 3, pp. 444–448, 2024.

[58] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya *et al.*, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Information fusion*, vol. 76, pp. 243–297, 2021.

[59] A. Graves, "Practical variational inference for neural networks," *Advances in neural information processing systems*, vol. 24, 2011.

[60] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *International conference on machine learning*. PMLR, 2015, pp. 1613–1622.

[61] C. Louizos and M. Welling, "Structured and efficient variational deep learning with matrix gaussian posteriors," in *International conference on machine learning*. PMLR, 2016, pp. 1708–1716.

[62] A. S. Maan, F. M. Garcia Herrero, A. Paler, and V. Savin, "Decoding correlated errors in quantum ldpc codes," *Nature Communications*, 2026.

[63] M. Wang and F. Mueller, "Coprime bivariate bicycle codes and their properties," *arXiv e-prints*, pp. arXiv–2408, 2024.

[64] ——, "Coprime bivariate bicycle codes and their layouts on cold atoms," *Quantum*, vol. 10, p. 2009, 2026.

[65] A. Gu, J. Ataides, M. D. Lukin, and S. F. Yelin, "Scalable neural decoders for practical fault-tolerant quantum computation," *arXiv preprint arXiv:2604.08358*, 2026.