# PRIME: A NEURAL-PRIOR BP-OSD DECODER FOR QUANTUM LDPC CODES

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

## ABSTRACT

Quantum error correction protects logical information by encoding it redundantly across physical qubits, and a decoder infers a correction from the measured syndrome without introducing a logical error. For bivariate bicycle (BB) quantum low-density parity-check (qLDPC) codes, belief propagation with ordered-statistics decoding (BP-OSD) is widely used, and its channel prior, the assumed per-qubit error probability, comes from a predefined noise model. We introduce PRIME, which augments BP-OSD with a graph neural network (GNN) that maps each measured syndrome to a syndrome-conditioned channel prior over physical-qubit errors. The underlying BP-OSD procedure is used unchanged, which preserves syndrome consistency while letting the decoder exploit noise structure learned from data, and because the GNN operates directly on the Tanner graph, one model trained on small BB codes generalizes without retraining to larger BB codes and toric codes. We further introduce Jitter-PRIME, a perturbed-prior ensemble whose candidates are aggregated into logical equivalence classes, together with a learned confidence gate that sends only the riskiest instances to it. At physical error rate $p = 0.10$, Jitter-PRIME reduces logical error rate (LER) over BP-OSD-CS-6 by $6.6\times$, $9.0\times$, and $57.8\times$ on BB codes with 288, 360, and 756 qubits. On the two held-out codes, the gate keeps the full ensemble gain with about 15% of the always-on ensemble's solver calls. Under circuit-level noise, Jitter-PRIME reduces LER over BP-OSD-CS-6 by $1.5$–$15\times$ on the length-72 and length-144 BB codes.

## 1 INTRODUCTION

Fault-tolerant quantum computing requires protection from continual physical noise. Quantum error correction encodes logical qubits across physical qubits, where parity-check outcomes, called syndromes, reveal errors without measuring the encoded state. The surface code (Kitaev, 2003; Fowler et al., 2012) supports local checks but uses many physical qubits per logical qubit, while quantum low-density parity-check (qLDPC) codes offer a lower-overhead alternative (Breuckmann & Eberhardt, 2021b), bivariate bicycle (BB) codes (Bravyi et al., 2024) are a practical example, but short cycles and many logically equivalent errors make decoding difficult. Belief propagation (BP) with ordered-statistics decoding (OSD), BP-OSD (Panteleev & Kalachev, 2021; Roffe et al., 2020), is a strong and widely used baseline for these codes. BP passes messages on the Tanner graph to estimate how likely each qubit is to be in error, and OSD uses those estimates to repair the output into a correction that matches the measured syndrome. Both stages depend on the *channel prior*, the initial per-qubit error probabilities, which standard BP-OSD takes from a fixed noise model. Every instance therefore starts from the same uninformed guess, even though the measured syndrome itself carries information about where errors are likely to have occurred.

In this work, we introduce PRIME, a new decoder that augments BP-OSD with a learned, syndrome-conditioned channel prior. Rather than replacing the classical decoder, PRIME improves its input: a graph neural network (GNN) reads the $X$ and $Z$ syndromes together on a single Tanner graph and predicts, for every qubit, the probability of an $X$ and of a $Z$ error given that syndrome. These predictions replace the fixed noise-model prior, and BP-OSD then runs unchanged, so every correction remains consistent with the syndrome. Because the GNN shares its node parameters across the graph, its size does not depend on the code length, and a single model trained on small BB codes can



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

decode larger BB codes, and codes from another family, without retraining. Building on PRIME, we further introduce Jitter-PRIME to recover failures that a single decode leaves behind. It perturbs the channel prior several times, placing larger perturbations on qubits whose predictions are uncertain, and runs BP-OSD once per perturbed prior, a single network evaluation is shared across them, so the additional cost is confined to the classical solver. Under code-capacity noise the resulting candidates are aggregated into logical equivalence classes and the most probable class is returned, while under circuit-level noise the perturbations act at fixed scales on the native fault-mechanism priors and the most likely candidate is returned. Since PRIME already decodes most instances correctly, a learned confidence gate sends only the riskiest instances to the ensemble. Our specific contributions are as follows:

- PRIME, a GNN that predicts a syndrome-conditioned channel prior for BP-OSD, preserving its BP updates and OSD post-processing. Because its parameters are shared across the Tanner graph and are independent of code length, one trained model decodes codes unseen during training, including codes larger than any it was trained on.
- Jitter-PRIME, a confidence-guided perturbed-prior ensemble that retains the measured gain of always-on ensembling on held-out BB codes while using about 15% of its BP-OSD call budget.
- Experiments showing that a single model trained on BB codes of length at most 144 generalizes without retraining to BB codes of length up to 3000 and to toric codes. On these unseen codes Jitter-PRIME lowers the logical error rate (LER) over BP-OSD with an order-6 combination sweep (BP-OSD-CS-6) by 6.6–57.8$\times$ at $p = 0.10$ and by 21.8$\times$ and 15.7$\times$ at $p = 0.12$ for $n = 1500$ and $n = 3000$, improves on Astra at every evaluated point, and lowers circuit-level LER by up to 15$\times$. On toric codes it improves on minimum-weight perfect matching (MWPM) by 2.3–10.7$\times$ while matching a per-code neural decoder, which fine-tuning then surpasses. Ablations isolate the gate and the predictor's components.

## 2 Related Work

**Codes and classical decoders.** qLDPC codes (Breuckmann & Eberhardt, 2021b) combine sparse checks with higher rates than topological codes (Dennis et al., 2002; Fowler et al., 2012), and product and lifted constructions reach good families (Tillich & Zemor, 2014; Panteleev & Kalachev, 2021; Breuckmann & Eberhardt, 2021a; Hastings et al., 2021; Leverrier & Zémor, 2022; Panteleev & Kalachev, 2022), BB codes (Bravyi et al., 2024) offer practical block sizes (Wang & Mueller, 2026). Matching (Higgott, 2022; Wu & Zhong, 2023) and union-find (Delfosse & Nickerson, 2021) exploit errors flipping at most two checks, which weight-six BB graphs violate, and tensor networks trade speed for accuracy (Bravyi et al., 2014; Chubb, 2021), leaving BP-OSD the standard BB decoder (Sahay et al., 2026; Tan et al., 2026).

**BP on quantum codes.** BP is effective on classical low-density parity-check codes (Gallager, 1962), but short cycles and degeneracy can trap it in symmetric fixed points (Poulin & Chung, 2008; David, 2004). Remedies keep BP as the solver and add OSD repair (Fossorier & Lin, 1995; Panteleev & Kalachev, 2021; Roffe et al., 2020) or other post-processing and inference changes (Hillmann et al., 2025; Yao et al., 2024; Kuo & Lai, 2022; Du Crest et al., 2022; Wolanski & Barber, 2025; Gong et al., 2024; Müller et al., 2025; Yin et al., 2024), PRIME instead learns the initial channel probabilities.

**Learned decoders.** Neural decoders span Boltzmann machines, classifiers, recurrent networks and reinforcement learning (Torlai & Melko, 2017; Varsamopoulos et al., 2017; Chamberland & Ronagh, 2018; Baireuther et al., 2018; Andreasson et al., 2019; Sweke et al., 2020), large surface-code systems (Meinerz et al., 2022; Bausch et al., 2024), per-code Transformers (Choukroun & Wolf, 2024; Zenati & Nachmani, 2026), GNNs (Lange et al., 2025; Bhave et al., 2026), and neural BP (Nachmani et al., 2016; Liu & Poulin, 2019). Astra (Maan & Paler, 2025) is our main learned BB baseline, neural MWPM (NMWPM) (Peled et al., 2026) learns syndrome-dependent matching weights, and circuit-level BB decoders (Blue et al., 2026) retrain per setting. Most train per code or size, a limitation foundation-style work has begun to address (Yan et al., 2026), whereas PRIME trains one set of GNN weights on small BB codes and applies it unchanged to larger ones. Other methods improve solver inputs: correlated matching (Fowler, 2013) and belief-matching (Higgott et al., 2023) adapt matching weights to the syndrome, and per-edge priors can be calibrated offline (Sivak et al.,

2024), PRIME learns this adaptation for BP-OSD across BB geometries. Ensembles of perturbed or symmetry-related decoders have also been explored (Shutty et al., 2024; Koutsioumpas et al., 2025), Jitter-PRIME instead perturbs a learned prior and routes with a learned gate.

# 3 BACKGROUND

## 3.1 QUANTUM ERRORS AND SYNDROMES

A Calderbank–Shor–Steane (CSS) code has $[\![n, k, d]\!]$ parameters: $n$ physical qubits, $k$ logical qubits, and minimum nontrivial logical weight $d$. Errors have components $e^x, e^z \in \mathbb{F}_2^n$ over the binary field: Pauli $X$ flips the first, $Z$ the second, and $Y$ both. Check matrices $H_X \in \mathbb{F}_2^{m_x \times n}$, $H_Z \in \mathbb{F}_2^{m_z \times n}$ contain $m_x, m_z$ checks and satisfy $H_X H_Z^\top = 0$, ensuring commuting measurements ($\top$ denotes transpose). Their syndromes obey

$$s^x = H_X e^z, \qquad s^z = H_Z e^x. \tag{1}$$

Binary sums and products are modulo two. The pairing is crossed because $X$-type checks detect $Z$ errors and conversely, allowing separate binary decoding of the two components.

## 3.2 DEGENERACY AND THE DECODING PROBLEM

The syndrome $(s^x, s^z)$ fixes violated constraints but not a unique error. Errors sharing a syndrome differ by a zero-syndrome operator: a stabilizer preserves the encoded information, while a nontrivial logical operator changes its logical class. A correction succeeds when it shares the sampled error's class. Choosing the most probable class exactly is intractable, so practical decoders approximate the posterior and seek a likely representative.

## 3.3 BIVARIATE BICYCLE CODES

A BB code (Bravyi et al., 2024) has $n = 2\ell m$ qubits on two copies of an $\ell$-by-$m$ periodic lattice, where $\ell, m$ are its side lengths. It builds both check matrices from cyclic shifts drawn from two binary polynomials. The resulting Tanner graph is sparse and periodic, but it carries many short cycles and highly degenerate error patterns, which is what makes BB codes hard for iterative decoders.

## 3.4 CHANNEL PRIORS, BP, AND OSD

The channel prior $\pi = (\pi_1, \ldots, \pi_n)$ gives each qubit's assumed error probability before message passing. The solver uses its log-likelihood ratio (LLR), $\Lambda_i = \log[(1-\pi_i)/\pi_i]$, whose sign gives the preferred bit value and magnitude measures confidence. Under independent depolarizing noise of rate $p$, each of $X, Y, Z$ occurs with probability $p/3$, so each binary component has marginal $2p/3$. Standard BP-OSD initializes all qubits with this constant before using the syndrome.

BP decodes each component separately: $(H, s) = (H_Z, s^z)$ for $X$ and $(H_X, s^x)$ for $Z$, seeking $H\hat{e} = s$. The Tanner graph joins qubit $i$ to check $c$ wherever $H_{ci} = 1$. With $q_{i\to c}$ the LLR message from qubit $i$ to check $c$, $r_{c\to i}$ the reverse message, and $N(\cdot)$ a graph neighborhood, one sum-product iteration is

$$q_{i\to c} = \Lambda_i + \sum_{c' \in N(i)\setminus c} r_{c'\to i}, \qquad r_{c\to i} = 2\tanh^{-1}\Big[(-1)^{s_c} \prod_{j \in N(c)\setminus i} \tanh\big(q_{j\to c}/2\big)\Big], \tag{2}$$

and the posterior LLR $\Lambda_i + \sum_{c\in N(i)} r_{c\to i}$ orders the qubits by reliability. The prior therefore enters every message and shapes that ordering. These updates are exact on trees, but qLDPC Tanner graphs contain many cycles, so BP may fail to converge or grow confident in a wrong pattern. Our experiments use scaled min-sum, which approximates the check update using the syndrome sign, the product of message signs, and 0.75 times their smallest magnitude. When BP fails to converge, OSD uses its reliability ordering to select independent columns of $H$ and solve $H\hat{e} = s$ (Fossorier & Lin, 1995). The library's combination sweep, CS-$q_{\mathrm{OSD}}$, tests all single-position patterns in the non-pivot set and all pairs within its first $q_{\mathrm{OSD}}$ ordered positions. We use CS-6, which sets this subset size to six, and retain the candidate with the lowest solver cost. A better prior can redirect both stages without changing their updates or candidate rule. PRIME uses this opening.

# 4 METHOD

## 4.1 OVERVIEW

PRIME combines a learned prior predictor with BP-OSD. The GNN reads both syndromes and predicts two error probabilities per qubit. BP-OSD uses these priors to infer a syndrome-consistent correction: the network primes the solver. A decoding instance is the tuple $(H_X, H_Z, s^x, s^z, p)$, holding the two check matrices, their measured syndromes, and the physical depolarizing rate. PRIME processes it in four stages:

1. **Preprocessing.** Encode the instance as one unified Tanner graph $\mathcal{G}$.
2. **Prior prediction.** Run the GNN on $\mathcal{G}$, obtaining the priors $\pi^x, \pi^z \in (0,1)^n$.
3. **Decoding.** Give those priors to BP-OSD in place of the constant $2p/3$, returning a syndrome-consistent correction $(\hat{e}^x, \hat{e}^z)$.
4. **Selective ensembling.** Score that correction with a learned gate. Low-risk instances return it unchanged, and the riskiest fraction $r$ goes to Jitter-PRIME, which decodes again under perturbed priors and pools the results by logical class.

Only the prior predictor and gate are learned, BP updates, OSD repair, and the logical-validity test follow the baseline. Figure 1 shows the pipeline.

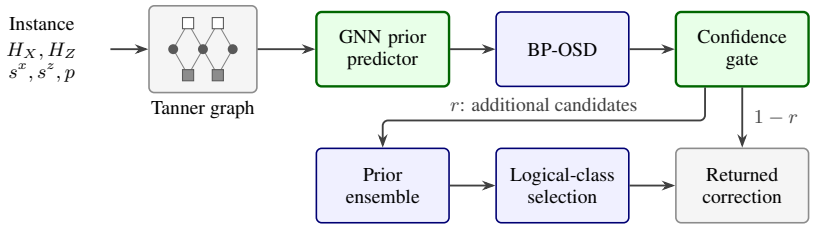


Figure 1: PRIME and selective Jitter-PRIME. Qubits (circles) connect $X$- and $Z$-checks (open and filled squares); the GNN supplies priors to two crossed BP-OSD calls, and the gate returns fraction $1-r$ directly and routes $r$ through ensembling and class selection. Green boxes are learned, blue classical.

The network estimates $\Pr[e_i^b = 1 \mid s^x, s^z]$ for $b \in \{x, z\}$. Both syndromes expose the correlation a $Y$ fault induces between components, factorized outputs fit the solver's channel interface (Section 3.4).

## 4.2 UNIFIED TANNER GRAPH

The predictor joins the two component Tanner graphs at their shared qubit nodes:

$$\mathcal{G} = \big(\mathcal{V}_Q \cup \mathcal{V}_X \cup \mathcal{V}_Z,\ \mathcal{E}_X \cup \mathcal{E}_Z\big), \tag{3}$$

where $\mathcal{V}_Q, \mathcal{V}_X, \mathcal{V}_Z$ hold the qubits, the $X$-checks and the $Z$-checks, and an edge joins qubit $i$ to check $c$ exactly when the corresponding check-matrix entry is one. Evidence from both syndrome types therefore meets at a qubit, including the coupling caused by $Y$ errors. All nodes share six features: four encode the two periodic lattice coordinates as sines and cosines, two hold $[p, p]$ for qubits, $[s_c^x, 0]$ for $X$-checks, or $[s_c^z, 1]$ for $Z$-checks. A type-specific affine map $P_\tau$, $\tau \in \{Q, X, Z\}$, maps node $v$'s features to $h_v^{(0)} \in \mathbb{R}^{d_h}$, with hidden width $d_h$, learned scalars rescale the check types. These graph states enter the predictor.

## 4.3 NEURAL PRIOR PREDICTOR

The predictor runs $T$ rounds of message passing on $\mathcal{G}$, sharing its weights across rounds. Round $t$ updates the checks first. For a check $c$ of type $\tau \in \{X, Z\}$ with neighboring qubits $N(c)$, learned affine maps $A_\tau$ and $B_\tau$ give

$$m_{i\to c}^{(t)} = A_\tau\big([h_i^{(t)} \| h_c^{(t)}]\big), \qquad h_c^{(t+1)} = h_c^{(t)} + \mathrm{ReLU}\Big(B_\tau\big(\mathrm{LN}\big(\textstyle\sum_{i\in N(c)} m_{i\to c}^{(t)}\big)\big)\Big), \tag{4}$$

where $\|$ is concatenation and LN is layer normalization. Including $h_c^{(t)}$ makes each message depend on its receiving check. Checks send affine projections of their updated states back to qubits, which sum their $X$- and $Z$-check inputs. A gated recurrent unit (GRU) (Cho et al., 2014) incorporates this sum into the recurrent state, layer normalization gives $h_i^{(t+1)}$. Local message passing cannot reach distant qubits in a fixed number of rounds. After the penultimate round, a linear-attention layer (Katharopoulos et al., 2020) exchanges information across all qubits at cost linear in $n$, the final round is local. A shared head maps each qubit state to two logits $z_i^x, z_i^z$, and the priors are $\pi_i^x = \sigma(z_i^x)$ and $\pi_i^z = \sigma(z_i^z)$, where $\sigma(z) = 1/(1 + e^{-z})$. Every learned map acts on a node or edge, allowing the same weights to generalize across code sizes.

### 4.4 BP-OSD Decoding

We replace the constant $2p/3$ with predicted priors clipped to $[\epsilon, 1 - \epsilon]$, keeping the LLR finite. The channel interface converts them internally, with the crossed pairing of Eq. equation 1:

$$\hat{e}^x = \mathrm{BPOSD}(H_Z, s^z, \pi^x), \qquad \hat{e}^z = \mathrm{BPOSD}(H_X, s^x, \pi^z). \tag{5}$$

The syndrome-consistent pair $(\hat{e}^x, \hat{e}^z)$ is the PRIME single decode. With matched settings, constant priors $\pi_i^x = \pi_i^z = 2p/3$ reproduce the baseline exactly. Predictions are decoder inputs, not an exact joint posterior over logical classes.

### 4.5 Selective Prior Ensemble

A syndrome-consistent correction succeeds if it differs from the true error by a stabilizer, a nontrivial logical difference changes the encoded information. BP-OSD returns one correction without knowing which case holds. A perturbed prior can lead it to another logical class, providing an alternative when the initial decode fails. Jitter-PRIME therefore decides which instances need extra decoding, generates candidate corrections, and chooses among their logical classes. The gate first scores the completed decode using syndrome density, prior uncertainty, correction weight and likelihood, and BP convergence. We standardize $d_\phi$ features per component and concatenate them into $\phi \in \mathbb{R}^{2d_\phi}$. Logistic regression gives the risk score

$$g(\phi) = \sigma\big(w_g^\top \phi + b_g\big), \qquad w_g \in \mathbb{R}^{2d_\phi}, \quad b_g \in \mathbb{R}, \tag{6}$$

fitted by class-balanced logistic regression. The highest-scoring fraction $r$ receives extra decodes, the rest keep the initial correction. Features use only decoder diagnostics, not true-error information.

Each additional candidate correction is obtained by perturbing the prior used for the initial decode. Writing $\pi$ for that prior, candidate $a = 1, \ldots, K$ uses

$$\pi_i^{(a)} = \sigma\left(\log \frac{\pi_i}{1 - \pi_i} + \sigma_i \xi_i^{(a)}\right), \qquad \xi_i^{(a)} \overset{\text{iid}}{\sim} \mathcal{N}(0, 1). \tag{7}$$

Adding the noise to the log-odds rather than to the probability keeps every perturbed prior strictly inside $(0, 1)$. The standard deviation $\sigma_i$ controls how far we perturb qubit $i$'s prior in log-odds space. It grows with the binary entropy of $\pi_i$, is scaled by $\bar{\sigma}$, and is capped at $\sigma_{\max}$. This concentrates exploration on uncertain predictions while applying smaller perturbations to confident ones. The original decode is kept alongside the new ones as candidate $a = 0$, and the single network evaluation is shared across all perturbed priors, so the additional cost is confined to the classical solver. Candidates are aggregated into logical equivalence classes, which we call class pooling, using the signature $\kappa(\hat{e}) = L_\perp(\hat{e} \oplus \hat{e}^{(0)})$, where $L_\perp \in \mathbb{F}_2^{k \times n}$ is the opposite-type logical-operator matrix and $\oplus$ denotes binary XOR. A zero signature means the candidate shares the original's logical class, other signatures identify the alternatives. After removing duplicates, candidates with the same signature form $\mathcal{D}_\kappa$, scored under the physical noise model as

$$S_\kappa = \log \sum_{\hat{e} \in \mathcal{D}_\kappa} \exp\{-\lambda_p w_H(\hat{e})\}, \qquad \lambda_p = \log \frac{1 - 2p/3}{2p/3}, \tag{8}$$

where $w_H$ counts flipped qubits and $\lambda_p$ is the LLR of Section 3.4 at the native prior $2p/3$. Pooling accumulates support across distinct representatives: several light corrections outweigh a single heavy one. We return the lightest member of the winning class, separately for each component.

An ordinary instance needs two solver calls and a routed one $2(K + 1)$, giving $2 + 2Kr$ calls per instance on average versus $2(K + 1)$ for always-on ensembling. A small $r$ limits the extra cost.

### 4.6 CIRCUIT-LEVEL CONFIGURATION

Circuit-level decoding uses a detector error model (DEM): a binary detector matrix $D$, a logical observable matrix $O$, and native mechanism priors $\pi^{\mathrm{dem}}$, where column $j$ records which detectors and observables mechanism $j$ flips. BP-OSD solves $D\hat{f} = s_D$ for a fault vector $\hat{f}$ given the measured detector syndrome $s_D$, and returns the observable flips $O\hat{f}$.

Jitter-PRIME perturbs the native logits of $\pi^{\mathrm{dem}}$ at fixed scales instead of entropy-scaling a learned prior, another arm raises priors for a random subset of mechanisms. Candidates are selected by native fault likelihood rather than logical-class pooling. The score $\sum_j \hat{f}_j \Lambda_j^{\mathrm{dem}}$ is the negative log-likelihood up to a constant, where $\Lambda_j^{\mathrm{dem}} = \log\left[(1-\pi_j^{\mathrm{dem}})/\pi_j^{\mathrm{dem}}\right]$. The lowest-scoring candidate is returned.

### 4.7 LOSS FUNCTION

The two learned components are fitted in turn, and neither is trained through the solver. We sample errors on the training codes, compute their syndromes, and fit the predictor on the sampled binary labels. Writing $\mathrm{BCE}(\pi, e)$ for mean binary cross-entropy and $\mathcal{H}(\pi)$ for mean binary entropy, the objective is

$$\mathcal{L} = \tfrac{1}{2} \sum_{b \in \{x,z\}} \left[ \mathrm{BCE}(\pi^b, e^b) + \beta \mathcal{H}(\pi^b) \right], \tag{9}$$

where the small positive $\beta$ mildly sharpens the predictions. No gradient passes through BP-OSD.

We then hold the predictor fixed and fit the gate of Eq. equation 6 by class-balanced logistic regression on features $\phi$ and single-decode failure labels from decoded samples. Both learned components remain fixed on every evaluated code.

## 5 EXPERIMENTS AND RESULTS

### 5.1 SETUP

**Training data.** The GNN trains on the BB codes $[\![36, 8, 4]\!]$ (lattices $3 \times 6$ and $6 \times 3$), $[\![72, 12, 6]\!]$, and $[\![144, 12, 12]\!]$, and is evaluated without retraining on $[\![288, 12, 18]\!]$, $[\![360, 12, 24]\!]$, $[\![756, 16, 34]\!]$, the BB codes with $n = 1500$ and $n = 3000$, and the toric codes with $L = 6, 8, 10, 12, 24$. Code-capacity samples use depolarizing noise: each qubit independently suffers an $X$, $Y$, or $Z$ error with total probability $p$, and syndromes are measured without error. Training draws fresh errors for every sample, with $p$ drawn from $\{0.01, 0.02, \ldots, 0.10, 0.12, 0.15\}$, for a total of 1.92 million samples. The gate is fitted on 60,000 decoded samples from the $n = 288$ code, and toric fine-tuning uses 0.96 million samples drawn jointly from the toric codes with $L = 6, 8, 10$, on which the fine-tuned model is then evaluated. BP-OSD-CS-6 and our decoders are evaluated on the same error samples.
**Hyperparameters.** The predictor has 342,469 parameters, hidden width $d_h = 128$, and $T = 4$ message-passing rounds. Adam trains it for 200 epochs of 300 minibatches of 32 samples (60,000 steps), with the learning rate decreasing from $3 \times 10^{-4}$ to $10^{-5}$ and entropy weight $\beta = 0.01$, toric fine-tuning runs 100 further epochs starting at $10^{-4}$. PRIME's BP stops at convergence or after 30 iterations. The BB ensemble draws $K = 16$ perturbed priors per routed sample, with average log-odds perturbation $\bar{\sigma} = 0.5$ and per-qubit cap $\sigma_{\max} = 2$, the gate routes the fraction $r = 0.10$ of samples with the highest predicted risk.
**Baselines.** The main baseline is BP-OSD-CS-6 with the native prior $2p/3$ and a 100-iteration BP cap. Both it and PRIME use serial min-sum BP with check messages scaled by 0.75 to offset min-sum's overconfidence. We also compare Astra, a learned BB decoder that we train on $n = 288$ (best checkpoint at epoch 39), and, on toric codes, MWPM and published NMWPM/QECCT values.
**Hardware and training time.** The predictor trains in about one hour on a single RTX 4090 GPU. Decoding-time measurements run BP on CPU and the GNN on an RTX 3090.

### 5.2 GENERALIZATION ACROSS THE BB FAMILY

Figure 2 evaluates PRIME on BB codes larger than any it was trained on, using the same GNN weights on every code and no retraining. At $p = 0.10$, Jitter-PRIME reduces LER over BP-OSD-CS-

6 by 6.6×, 9.0×, and 57.8× for $n = 288$, 360, and 756, a further 2.0×, 2.2×, and 6.3× over PRIME. PRIME transfers to these larger codes because its prior depends only on local structure. Every code in the family has weight-six checks, and the GNN applies the same node update everywhere, so after $T = 4$ rounds each qubit's prediction depends on a neighborhood whose size and degrees do not change with $n$. Within that neighborhood the prior has two properties that the constant $2p/3$ lacks. First, it is *localized*: it raises the error probability of qubits next to unsatisfied checks and lowers it elsewhere, so BP has fewer candidate explanations to separate and OSD ranks qubits from more accurate reliabilities. Second, it is *correlated across $X$ and $Z$*: the GNN reads both syndromes together, so a $Y$ error that flips checks of both types can raise both components of the prior on that qubit, whereas standard BP-OSD decodes the two components independently and discards this information. Jitter-PRIME adds to this by addressing the qubits on which the prior is still uncertain. A single decode often fails because a few such qubits send OSD to the wrong logical class. Scaling the perturbation with the entropy of each prediction explores those qubits while leaving confident ones intact, and pooling by logical class exploits degeneracy: the correct class is usually supported by several distinct light corrections, whereas a wrong one rarely is. The further gain over PRIME is largest on $n = 756$, which we attribute to its larger number of uncertain qubits and equivalent corrections, giving the perturbations more distinct candidates to find and pooling more support to accumulate. The advantage grows at lower noise. At $p = 0.06$ with $n = 288$, PRIME reduces LER by 15× and Jitter-PRIME by 47×, at $p = 0.08$ with $n = 756$, by 24× and by more than 300×. PRIME's LER falls as $p^{13.5}$, $p^{14.0}$, and $p^{20.0}$ on the three codes, faster than BP-OSD-CS-6 ($p^{11.4}$ for $n = 288$, $p^{16.0}$ for $n = 756$). A steeper slope means that the remaining failures require heavier errors. At low $p$, many BP-OSD failures come from light errors that the fixed prior misdecodes, the learned prior corrects many of these, so fewer failures remain, and each needs more faults to occur, which becomes rapidly less likely as $p$ falls.

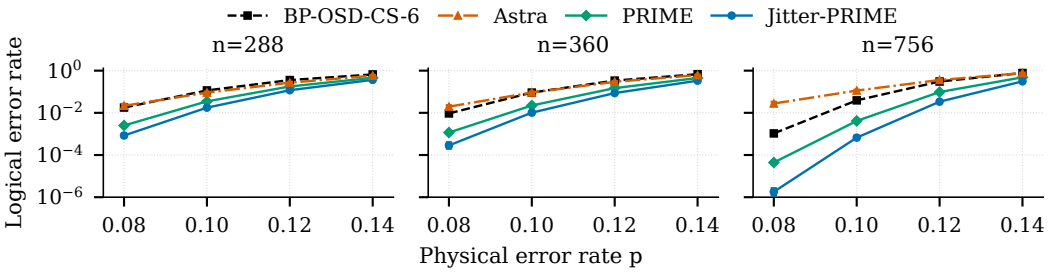


Figure 2: Decoding larger BB codes without retraining. All panels use the same GNN weights, trained only on the BB codes with $n \leq 144$. Jitter-PRIME uses $K = 16$. Error bars show 95% Wilson intervals.

PRIME and Jitter-PRIME improve on Astra at every point in the figure, although Astra is trained on $n = 288$. The gain comes from the prior rather than from additional search: at $p = 0.10$, PRIME with OSD-0, which omits the combination sweep, still reduces LER by 1.8–3.1× relative to BP-OSD-CS-10 on the three codes. The predictor also generalizes to $n = 1500$ and $n = 3000$, where at $p = 0.12$ Jitter-PRIME with $K = 8$ reduces LER over BP-OSD-CS-6 by 21.8× and 15.7×.

### 5.3 Circuit-Level Decoding

Jitter-PRIME reduces circuit-level BB LER by 1.5–15× over BP-OSD-CS-6 (Figure 3). We use Stim (Gidney, 2021) DEMs from the syndrome-extraction circuits of Bravyi et al. (2024), with six rounds on $[\![72, 12, 6]\!]$ and twelve on $[\![144, 12, 12]\!]$. Here $p$ is the circuit fault rate, and LER counts failure of any of the $k$ tracked Z-memory logical observables. The decoder uses the circuit-level configuration of Section 4.6. For $n = 144$ at $p = 0.003$, Jitter-PRIME corrects 28 of the 30 BP-OSD failures, a 15× reduction. Across the grid, the candidate pool contains a logically correct correction for 92%–100% of BP-OSD failures. A circuit-level gate identifies 79% of BP-OSD failures at 10% routing, versus 30% for syndrome-weight ranking and 19% for BP-nonconvergence ranking.

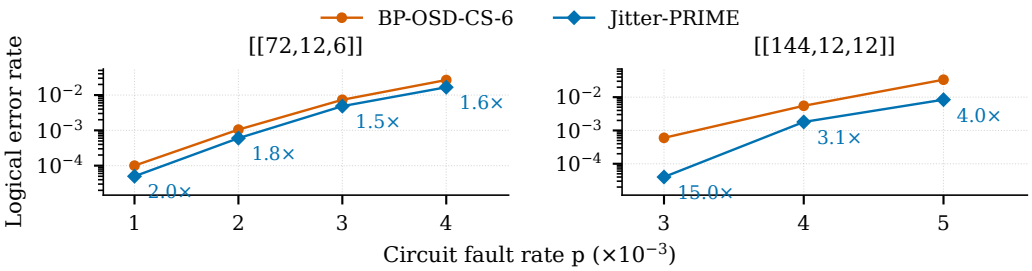


Figure 3: Circuit-level Jitter-PRIME on two BB codes. The ensemble perturbs native DEM priors and selects by fault likelihood, labels give the LER reduction over BP-OSD-CS-6.

### 5.4 Generalization to Toric Codes

Toric codes have $[\![2L^2, 2, L]\!]$ parameters, where $L$ is lattice size, and their weight-four checks and two logical qubits differ from the BB training codes. Applied directly at $L = 6, 8, 10$, the BB predictor matches the per-code NMWPM decoder to within a factor 0.92–1.09, and at $p = 0.10$ it improves on MWPM by 2.3–3.7× on these sizes and by 10.7× at $L = 24$. Fine-tuning these BB-trained weights jointly on $L = 6, 8, 10$ improves the model further (Figure 4): its LER is below both the published NMWPM and QECCT values at all 22 points shown. Toric results use the single decode.

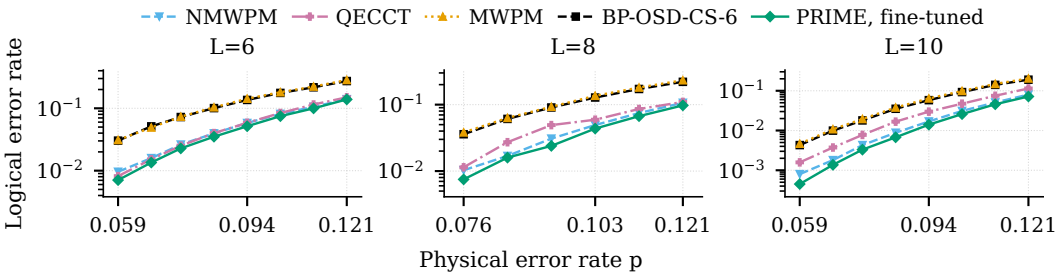


Figure 4: PRIME jointly fine-tuned on toric sizes $L = 6, 8, 10$, against BP-OSD-CS-6, MWPM, and published per-code neural decoders.

### 5.5 Decoding Cost

At $p = 0.10$, PRIME is faster than BP-OSD-CS-6 on every code in Figure 5: 0.91, 1.1, and 2.07 ms per sample for $n = 288$, 360, and 756, against 1.18, 1.6, and 3.99 ms, including the GNN. On $n = 756$, the GNN takes 0.80 ms and the solver 1.27 ms. Gated Jitter-PRIME takes 5.08 ms and 14.30 ms on $n = 288$ and $n = 756$, against 11.07 ms and 30.19 ms for always-on ensembling and 21.3 ms and 57.2 ms for Astra on the same host. The gate reduces solver calls from 34 to 5.2 per sample and latency by 2.1–2.4×, latency falls less than the call count because routed samples are the slowest to decode, 115 ms against 0.68 ms on $n = 756$.

## 6 Ablation Study

We validate the design of Jitter-PRIME by removing one component at a time under code-capacity depolarizing noise (Table 1), at two levels. In the decoding pipeline, we remove the gate, the ensemble, and the learned prior in turn on the held-out $n = 360$ and $n = 756$ codes at $p = 0.10$, recording the BP-OSD calls each variant needs. In the predictor, we retrain the GNN without one architectural component and evaluate it on the held-out $n = 288$ code, both on its own and as the

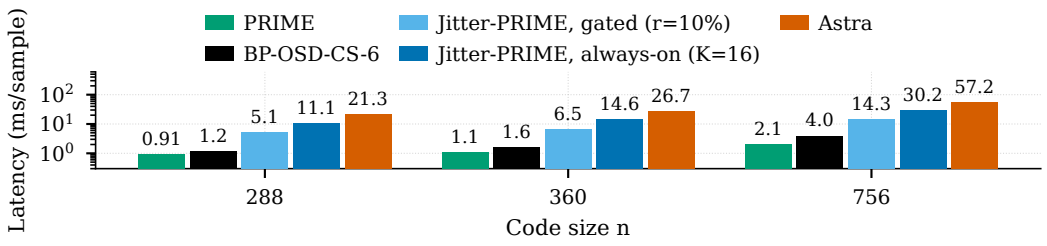


Figure 5: Mean latency per sample at $p = 0.10$, logarithmic axis. BP runs on CPU and the GNN on an RTX 3090, Jitter-PRIME uses $K = 16$, gated at $r = 0.10$.

prior for BP-OSD, which separates what a component contributes to the network's predictions from what it contributes to the decoder.

Table 1: Ablation from the full decoder. Each row below the first removes one component. Left table: decoding pipeline at $p = 0.10$. *Calls* is the mean number of BP-OSD calls per sample, and the $n = 360$ and $n = 756$ columns give the logical error rate in units of $10^{-3}$. Jitter-PRIME and the random-routing row send 10% of samples to the $K = 16$ ensemble. Right table: GNN predictors retrained without one component, evaluated on $n = 288$. Each entry is the logical error rate relative to the full predictor, averaged over $p = 0.060$–$0.070$, for the GNN read out on its own (*GNN*) and as the prior for BP-OSD (*PRIME*).

| Decoder, $p = 0.10$ | Calls | $n = 360$ | $n = 756$ |
|---|---|---|---|
| Jitter-PRIME | 5.2 | 8.9 | 0.60 |
| − gate (always on) | 34 | 9.0 | 0.60 |
| − gate (random 10%) | 5.2 | 21.7 | 4.38 |
| − ensemble (PRIME) | 2 | 23.1 | 4.80 |
| − prior (BP-OSD-CS-6) | 2 | 91.3 | 39.0 |

| Predictor, $n = 288$ | GNN | PRIME |
|---|---|---|
| Full predictor | 1.00 | 1.00 |
| − type scaling | 1.15 | 0.94 |
| − check conditioning | 1.17 | 1.08 |
| − GRU | 1.28 | 1.05 |
| − linear attention | 1.38 | 1.07 |

**Decoding pipeline.** At $p = 0.10$, the learned gate retains the full always-on accuracy on $n = 360$ and $n = 756$ with 5.2 instead of 34 solver calls per sample, whereas routing the same 10% at random keeps only a tenth of the ensemble gain. Removing the ensemble raises LER $2.6\times$ and $8.0\times$, and removing the learned prior raises it by a further $3.9\times$ and $8.1\times$, to BP-OSD-CS-6. Class pooling (Eq. 8) corrects 3%–5% more failures than selecting the lightest candidate on the same pools.
**Predictor components.** Retraining the predictor on the identical recipe without linear attention, the GRU, check-conditioned messages, or type scaling and evaluating on $n = 288$ at $p = 0.060$–$0.070$, every component improves the GNN's own predictions, read out without BP-OSD, by $1.38\times$, $1.28\times$, $1.17\times$, and $1.15\times$ respectively. Through BP-OSD, every variant stays 9–$14\times$ better than BP-OSD-CS-6 and within 8% of the full model, comparable to the variation between identically configured training runs, so BP-OSD absorbs architectural differences in the prior. What limits accuracy is the prior itself: an oracle prior derived from the true error decodes every sample correctly for $n = 72$ at $p = 0.05$.

## 7 Conclusion

PRIME learns syndrome-conditioned priors that improve BP-OSD and generalize across BB sizes and to toric codes, Jitter-PRIME strengthens these results and extends them to circuit-level noise, with a learned gate concentrating ensemble calls on the difficult samples.

## Reproducibility statement

The code construction, splits, model, training, solver settings, and evaluation protocol are specified in the main text and Appendices A–E. Figure and table scripts read the underlying result files, so every reported result can be regenerated.

## AI use statement

We used generative AI coding assistants to help implement and review code, edit the manuscript, prepare figures and tables, check notation, analyze saved evaluation outputs, and discuss experimental and ablation design. Training and evaluation samples were generated by the specified noise simulators. The authors are responsible for verifying the code, numerical claims, experimental interpretation, and final manuscript, including all AI-assisted material.

## References


Philip Andreasson, Joel Johansson, Simon Liljestrand, and Mats Granath. Quantum error correction for the toric code using deep reinforcement learning. *Quantum*, 3:183, 2019.

Paul Baireuther, Thomas E O'Brien, Brian Tarasinski, and Carlo WJ Beenakker. Machine-learning-assisted correction of correlated qubit errors in a topological code. *Quantum*, 2:48, 2018.

Johannes Bausch, Andrew W Senior, Francisco JH Heras, Thomas Edlich, Alex Davies, Michael Newman, Cody Jones, Kevin Satzinger, Murphy Yuezhen Niu, Sam Blackwell, et al. Learning high-accuracy error decoding for quantum processors. *Nature*, 635(8040):834–840, 2024.

Ameya S Bhave, Navnil Choudhury, and Kanad Basu. Hypernq: A hypergraph neural network decoder for quantum ldpc codes. In *ICC 2026-IEEE International Conference on Communications*, pp. 1–7. IEEE, 2026.

John Blue, Harshil Avlani, Zhiyang He, Liu Ziyin, and Isaac L Chuang. Machine learning decoding of circuit-level noise for bivariate bicycle codes. *Quantum*, 10:2149, 2026.

Sergey Bravyi, Martin Suchara, and Alexander Vargo. Efficient algorithms for maximum likelihood decoding in the surface code. *Physical Review A*, 90(3):032326, 2014.

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, 2021a.

Nikolas P Breuckmann and Jens Niklas Eberhardt. Quantum low-density parity-check codes. *PRX quantum*, 2(4):040101, 2021b.

Christopher Chamberland and Pooya Ronagh. Deep neural decoders for near term fault-tolerant experiments. *Quantum Science and Technology*, 3(4):044002, 2018.

Kyunghyun Cho, Bart Van Merriënboer, Çağlar Gulçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1724–1734, 2014.

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

Christopher T Chubb. General tensor network decoding of 2d pauli codes. *arXiv preprint arXiv:2101.04125*, 2021.

MACKAY David. Sparse graph codes for quantum error-correction. *IEEE Transactions on Information Theory*, 2004.

Nicolas Delfosse and Naomi H Nickerson. Almost-linear time decoding algorithm for topological codes. *Quantum*, 5:595, 2021.

Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. Topological quantum memory. *Journal of Mathematical Physics*, 43(9):4452–4505, 2002.

Julien Du Crest, Mehdi Mhalla, and Valentin Savin. Stabilizer inactivation for message-passing decoding of quantum ldpc codes. In *2022 IEEE Information Theory Workshop (ITW)*, pp. 488–493. IEEE, 2022.

Marc PC Fossorier and Shu Lin. Soft-decision decoding of linear block codes based on ordered statistics. *IEEE Transactions on information Theory*, 41(5):1379–1396, 1995.

Austin G Fowler. Optimal complexity correction of correlated errors in the surface code. *arXiv preprint arXiv:1310.0863*, 2013.

Austin G Fowler, Matteo Mariantoni, John M Martinis, and Andrew N Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A—Atomic, Molecular, and Optical Physics*, 86(3):032324, 2012.

Robert Gallager. Low-density parity-check codes. *IRE Transactions on information theory*, 8(1): 21–28, 1962.

Craig Gidney. Stim: a fast stabilizer circuit simulator. *Quantum*, 5:497, 2021.

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*, 2024.

Matthew B Hastings, Jeongwan Haah, and Ryan O'Donnell. Fiber bundle codes: breaking the n 1/2 polylog (n) barrier for quantum ldpc codes. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1276–1288, 2021.

Oscar Higgott. Pymatching: A python package for decoding quantum codes with minimum-weight perfect matching. *ACM Transactions on Quantum Computing*, 3(3):1–16, 2022.

Oscar Higgott, Thomas C Bohdanowicz, Aleksander Kubica, Steven T Flammia, and Earl T Campbell. Improved decoding of circuit noise and fragile boundaries of tailored surface codes. *Physical Review X*, 13(3):031007, 2023.

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.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

A Yu Kitaev. Fault-tolerant quantum computation by anyons. *Annals of physics*, 303(1):2–30, 2003.

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

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.

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

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

Kai Meinerz, Chae-Yeun Park, and Simon Trebst. Scalable neural decoder for topological surface codes. *Physical Review Letters*, 128(8):080505, 2022.

Tristan Müller, Thomas Alexander, Michael E. Beverland, Markus Bühler, Blake R. Johnson, Thilo Maurer, and Drew Vandeth. Improved belief propagation is sufficient for real-time decoding of quantum memory, 2025. URL https://arxiv.org/abs/2506.01779.

Eliya Nachmani, Yair Beery, and David Burshtein. Learning to decode linear codes using deep learning, 2016. URL https://arxiv.org/abs/1607.04793.

Pavel Panteleev and Gleb Kalachev. Degenerate quantum ldpc codes with good finite length performance. *Quantum*, 5:585, November 2021. ISSN 2521-327X. doi: 10.22331/q-2021-11-22-585. URL http://dx.doi.org/10.22331/q-2021-11-22-585.

Pavel Panteleev and Gleb Kalachev. Asymptotically good quantum and locally testable classical ldpc codes, 2022. URL https://arxiv.org/abs/2111.03654.

Yotam Peled, David Zenati, and Eliya Nachmani. Neural minimum weight perfect matching for quantum error codes, 2026. URL https://arxiv.org/abs/2601.00242.

David Poulin and Yeojin Chung. On the iterative decoding of sparse quantum codes, 2008. URL https://arxiv.org/abs/0801.1241.

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), December 2020. ISSN 2643-1564. doi: 10.1103/physrevresearch.2.043423. URL http://dx.doi.org/10.1103/PhysRevResearch.2.043423.

Kaavya Sahay, Dominic J. Williamson, and Benjamin J. Brown. A matching decoder for bivariate bicycle codes, 2026. URL https://arxiv.org/abs/2602.22770.

Noah Shutty, Michael Newman, and Benjamin Villalonga. Efficient near-optimal decoding of the surface code through ensembling, 2024. URL https://arxiv.org/abs/2401.12434.

Volodymyr Sivak, Michael Newman, and Paul Klimov. Optimization of decoder priors for accurate quantum error correction. *Physical Review Letters*, 133(15), October 2024. ISSN 1079-7114. doi: 10.1103/physrevlett.133.150603. URL http://dx.doi.org/10.1103/physrevlett.133.150603.

Ryan Sweke, Markus S Kesselring, Evert P L van Nieuwenburg, and Jens Eisert. Reinforcement learning decoders for fault-tolerant quantum computation. *Machine Learning: Science and Technology*, 2(2):025005, December 2020. ISSN 2632-2153. doi: 10.1088/2632-2153/abc609. URL http://dx.doi.org/10.1088/2632-2153/abc609.

Shi Jie Samuel Tan, Ian Gill, Eric Huang, Pengyu Liu, Chen Zhao, Hossein Dehghani, Aleksander Kubica, Hengyun Zhou, and Arpit Dua. Generalized matching decoders for 2d topological translationally-invariant codes, 2026. URL https://arxiv.org/abs/2603.05402.

Jean-Pierre Tillich and Gilles Zemor. 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, February 2014. ISSN 1557-9654. doi: 10.1109/tit.2013.2292061. URL http://dx.doi.org/10.1109/TIT.2013.2292061.

Giacomo Torlai and Roger G. Melko. Neural decoder for topological codes. *Physical Review Letters*, 119(3), July 2017. ISSN 1079-7114. doi: 10.1103/physrevlett.119.030501. URL http://dx.doi.org/10.1103/PhysRevLett.119.030501.

Savvas Varsamopoulos, Ben Criger, and Koen Bertels. Decoding small surface codes with feedforward neural networks. *Quantum Science and Technology*, 3(1):015004, November 2017. ISSN 2058-9565. doi: 10.1088/2058-9565/aa955a. URL http://dx.doi.org/10.1088/2058-9565/aa955a.

Ming Wang and Frank Mueller. Coprime bivariate bicycle codes and their layouts on cold atoms. *Quantum*, 10:2009, February 2026. ISSN 2521-327X. doi: 10.22331/q-2026-02-23-2009. URL http://dx.doi.org/10.22331/q-2026-02-23-2009.

Stasiu Wolanski and Ben Barber. Ambiguity clustering: an accurate and efficient decoder for qldpc codes, 2025. URL https://arxiv.org/abs/2406.14527.

Yue Wu and Lin Zhong. Fusion blossom: Fast mwpm decoders for qec, 2023. URL https://arxiv.org/abs/2305.08307.

Ge Yan, Shanchuan Li, Shiyi Xiao, Pengyue Ma, Hanyan Cao, Feng Pan, and Yuxuan Du. Efficient foundation decoders for fault-tolerant quantum computing, 2026. URL https://arxiv.org/abs/2606.27119.

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, 2024. URL https://arxiv.org/abs/2312.10950.

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, 2024. URL https://arxiv.org/abs/2412.02885.

David Zenati and Eliya Nachmani. Saq: Stabilizer-aware quantum error correction decoder, 2026. URL https://arxiv.org/abs/2512.08914.

# A Codes, Noise, and Decoding Conventions

## A.1 BB constructions and data splits

A BB code is built from binary polynomials $a(x, y)$ and $b(x, y)$ with $x^\ell = y^m = 1$, where $\ell, m$ are the periodic lattice dimensions. Replacing monomials by cyclic-shift matrices gives commuting binary matrices $A, B \in \mathbb{F}_2^{\ell m \times \ell m}$. The checks are

$$H_X = [A\ B], \qquad H_Z = [B^\top\ A^\top], \qquad n = 2\ell m. \tag{10}$$

Thus $H_X H_Z^\top = 0$. Sparsity bounds check weight but does not by itself imply geometric locality. Table 2 specifies the geometries. The two data-qubit blocks share corresponding lattice coordinates.

None of the evaluation codes enters predictor training. The distances 24 and 34 of the 360- and 756-qubit codes, and 200 and 302 for the two largest constructions, are upper bounds found by search.

Table 2: Code construction and role. The 36-qubit row contains two geometries. The two largest codes have $k = 8$ and $k = 16$, respectively.

| Role | $n$ | $(\ell, m)$ | $a(x, y)$ | $b(x, y)$ |
|---|---|---|---|---|
| Training | 36 | $(3, 6), (6, 3)$ | $x^3 + y + y^2$ | $y^3 + x + x^2$ |
| Training | 72 | $(6, 6)$ | $x^3 + y + y^2$ | $y^3 + x + x^2$ |
| Training | 144 | $(12, 6)$ | $x^3 + y + y^2$ | $y^3 + x + x^2$ |
| Evaluation | 288 | $(12, 12)$ | $x^3 + y^2 + y^7$ | $y^3 + x + x^2$ |
| Evaluation | 360 | $(30, 6)$ | $x^9 + y + y^2$ | $y^3 + x^{25} + x^{26}$ |
| Evaluation | 756 | $(21, 18)$ | $x^3 + y^{10} + y^{17}$ | $y^5 + x^3 + x^{19}$ |
| Evaluation | 1500 | $(30, 25)$ | $x^{22} + y^{19} + y^{18}$ | $y^{19} + x^{15} + x^4$ |
| Evaluation | 3000 | $(75, 20)$ | $x^2 + y^{10} + y^{18}$ | $y^{18} + x^{13} + x^{12}$ |

### A.2 Logical equivalence and solver interface

An X residual $e^x \oplus \hat{e}^x$ is tested with syndrome matrix $H_Z$ and Z-type logical operators $L_Z$. A Z residual uses $H_X$ and $L_X$. The matrices $L_X, L_Z \in \mathbb{F}_2^{k\times n}$ have one row per logical operator of the stated type. A successful full correction has zero residual syndrome and zero logical signature for both components.

For a syndrome $s = (s^x, s^z)$, let $\mathcal{C}(s)$ contain its compatible logical classes. An ideal maximum-likelihood class decoder selects

$$\hat{C}(s) = \arg\max_{C\in\mathcal{C}(s)} \sum_{e\in C} P(e \mid s). \tag{11}$$

Here $e$ is a complete Pauli error and $C$ one compatible class. Our class score sums only over discovered, distinct candidates, with per-component native-channel weights. It does not evaluate the full joint posterior.

The LDPC BP-OSD implementation accepts per-variable channel probabilities. We use serial min-sum BP with scale 0.75 and OSD with an order-6 combination sweep. PRIME uses a 30-iteration cap, and the native baseline and circuit-level runs use 100. Match these settings when testing constant-prior equivalence. OSD constructs a valid correction by reliability-ordered binary elimination and combination-sweep reprocessing.

### A.3 Circuit-level noise

The circuit follows the Bravyi BB extraction schedule with seven CNOT layers per syndrome round. The implementation applies two-qubit depolarizing noise after CNOTs, single-qubit depolarizing noise at idle data locations, and preparation and measurement flips, controlled by $p$. It prepares a Z-memory experiment and tracks all $k$ Z-logical observables. The reported DEM uses Z-check detector comparisons between rounds and a final boundary from data measurements. X-check measurements are performed but are not added as a second detector family in these runs.

Let $M_D$ be the number of detectors and $N_f$ the number of mechanisms. Stim constructs $D \in \mathbb{F}_2^{M_D\times N_f}$, $O \in \mathbb{F}_2^{k\times N_f}$, and $q \in (0,1)^{N_f}$, without decomposing the DEM into graphlike errors. Its sampler draws a fault vector $f$, syndrome $s_D = Df$, and observable flips $Of$. Correctness means $O(\hat{f} \oplus f) = 0$ for all tracked observables. The 72-qubit, six-round model has 2,232 mechanisms, and the 144-qubit, twelve-round model has 8,784.

## B Architecture and Training

### B.1 Node features and projection

For periodic coordinates $(u_1, u_2)$, define

$$\mathrm{PE}(u_1,u_2) = \left[\sin\frac{2\pi u_1}{\ell}, \cos\frac{2\pi u_1}{\ell}, \sin\frac{2\pi u_2}{m}, \cos\frac{2\pi u_2}{m}\right] \in \mathbb{R}^4. \tag{12}$$

Writing $\mathrm{PE}_v$ for the encoding at node $v$, the features are

$$f_i^Q = [p, p, \mathrm{PE}_i], \qquad f_c^X = [s_c^x, 0, \mathrm{PE}_c], \qquad f_c^Z = [s_c^z, 1, \mathrm{PE}_c] \in \mathbb{R}^6. \tag{13}$$

The matrix has $n+m_x+m_z$ rows. Duplicated qubit entries reserve two channel slots, and the check flag distinguishes syndrome types. For node type $\tau \in \{Q, X, Z\}$, $P_\tau$ applies a 6-to-$d_h$ affine map, layer normalization, and ReLU. For each check type $\tau \in \{X, Z\}$, a learned scalar $\alpha_\tau$ is clamped to $[0.3, 0.7]$. Check states are scaled by $\alpha_\tau + \rho(1-\alpha_\tau)$, with $\rho = 0.1$. This is the type-specific symmetry-scaling component, inspired by SymBreak (Yin et al., 2024).

### B.2 Local updates

In Eq. equation 4, $A_\tau : \mathbb{R}^{2d_h} \to \mathbb{R}^{d_h}$ and $B_\tau : \mathbb{R}^{d_h} \to \mathbb{R}^{d_h}$ are affine maps including biases. For the return path, let $C_X, C_Z : \mathbb{R}^{d_h} \to \mathbb{R}^{d_h}$ be learned affine maps, and $N_X(i), N_Z(i)$ the neighboring

checks of qubit $i$. Its received message is

$$u_i^{(t)} = \sum_{c \in N_X(i)} C_X(h_c^{(t+1)}) + \sum_{c \in N_Z(i)} C_Z(h_c^{(t+1)}). \tag{14}$$

The recurrent state $h_{i,\mathrm{rec}}^{(t)} \in \mathbb{R}^{d_h}$ starts at zero and obeys

$$h_{i,\mathrm{rec}}^{(t+1)} = \mathrm{GRU}(u_i^{(t)}, h_{i,\mathrm{rec}}^{(t)}), \qquad h_i^{(t+1)} = \mathrm{LN}(h_{i,\mathrm{rec}}^{(t+1)}). \tag{15}$$

Local maps and GRU weights are shared across rounds. The input projection supplies $h_i^{(0)}$ to the first message round but does not initialize the recurrent state.

### B.3 Linear attention and output

After the third local round, collect qubit states as rows of $M \in \mathbb{R}^{n \times d_h}$. Learned matrices $W_q, W_k, W_v \in \mathbb{R}^{d_h \times d_h}$ give $Q = MW_q$, $K_A = MW_k$, and $V = MW_v$. The subscript on $K_A$ distinguishes attention keys from ensemble size. With positive feature map $\varphi(x) = \mathrm{ELU}(x) + 1$, compute

$$U = \frac{\varphi(Q)[\varphi(K_A)^\top V]}{\varphi(Q)[\varphi(K_A)^\top \mathbf{1}] + \epsilon_A} \in \mathbb{R}^{n \times d_h}, \qquad \epsilon_A = 10^{-6}. \tag{16}$$

Here $\mathbf{1} \in \mathbb{R}^n$ is the all-ones vector. Division uses one denominator per row. Evaluating bracketed products first avoids an $n \times n$ matrix, giving cost $O(nd_h^2)$, linear in $n$ at fixed width. Learned output matrix $W_o \in \mathbb{R}^{d_h \times d_h}$, bias $b_o \in \mathbb{R}^{d_h}$, and clamped gate $\gamma \in [0, 1]$ give

$$M \leftarrow M + \gamma(UW_o + \mathbf{1}b_o^\top). \tag{17}$$

This consistently uses row states. Attention is evaluated separately per graph, then the fourth local round follows. The output head applies layer normalization, an affine $d_h$-to-$d_h/2$ map, ReLU, and an affine $d_h/2$-to-2 map.

### B.4 Training details

For component prediction $\pi$ and sampled binary error $e$,

$$\mathrm{BCE}(\pi, e) = -n^{-1} \sum_i [e_i \log \pi_i + (1 - e_i) \log(1 - \pi_i)], \tag{18}$$

$$\mathcal{H}(\pi) = -n^{-1} \sum_i [\pi_i \log \pi_i + (1 - \pi_i) \log(1 - \pi_i)]. \tag{19}$$

We use $\beta = 0.01$. The positive entropy term favors confident probabilities and changes the pure cross-entropy optimum. We use these as learned decoder inputs rather than claim exact calibrated marginals.

Training cycles through four geometries with fresh errors. Each sample draws $p$ from $\{0.01, 0.02, \ldots, 0.10, 0.12, 0.15\}$. Training graphs receive the batch-mean rate, and evaluation uses the exact rate. Adam runs for 200 epochs of 300 batches of 32 samples, learning rate decreasing from $3 \times 10^{-4}$ to $10^{-5}$, weight decay $10^{-4}$, and gradient clipping at 1.0, with seed 42. Training takes about one hour on a single RTX 4090 GPU. Checkpoint selection uses PRIME LER at $p = 0.05$ on 2,000 samples from the training $[\![72, 12, 6]\!]$ code and selects epoch 200. Toric fine-tuning starts from these weights and runs 100 epochs of the same batch schedule on $L = 6, 8, 10$, with learning rate decreasing from $10^{-4}$ to $10^{-5}$ and checkpoint selection at $p = 0.10$ on $L = 10$. GRU, attention every third round, type scaling, and check-conditioned messages are enabled.

## C Ensemble and Routing Details

### C.1 Perturbation scale and selection

For the entropy calculation, clip the network output to $\tilde{\pi}_i \in [10^{-6}, 1 - 10^{-6}]$. Define $\eta_i = -\tilde{\pi}_i \log \tilde{\pi}_i - (1 - \tilde{\pi}_i) \log(1 - \tilde{\pi}_i)$ and its positive mean $\bar{\eta} = n^{-1} \sum_i \eta_i$. The scale is

$$\sigma_i = \min\left(\bar{\sigma} \frac{\eta_i}{\bar{\eta}}, \sigma_{\max}\right), \tag{20}$$

so clipping prevents a zero denominator. The single decode clips its priors with $\epsilon = 10^{-4}$, and each candidate prior is separately clipped to the same range at the BP-OSD interface. The BB default is $\bar{\sigma} = 0.5$, $\sigma_{\max} = 2$, $K = 16$. The sigmoid is written $\sigma$, reserving $\sigma$ for scales and $\tau$ for node type.

The original candidate is retained and duplicate corrections are removed before scoring. Use $L_\perp = L_Z$ for X candidates and $L_\perp = L_X$ for Z candidates. Under the uniform component marginal $2p/3$, log probability is $-\lambda_p w_H$ plus a constant. Selection applies separately to the components.

### C.2 Gate features and fitting

For one component let $m_s$ be syndrome length, $w = \sum_i \hat{e}_i$ correction weight, $\mu_w = \sum_i \pi_i$ expected weight, and $v_w = \sum_i \pi_i(1-\pi_i)$. The $d_\phi = 11$ features per component are syndrome weight divided by $m_s$, $\mu_w/n$, $w/n$, $(w-\mu_w)/\sqrt{v_w + 10^{-9}}$, correction negative log-likelihood divided by $n$, mean entropy, fraction of priors strictly between 0.1 and 0.9, mean entropy of the ten most uncertain qubits, BP convergence, BP iterations divided by the cap, and OSD triggering. Unavailable diagnostics use $-1$.

Concatenate the two components and standardize using the means and standard deviations of the fitting data, adding $10^{-9}$ to each standard deviation. Class-balanced logistic regression uses regularization parameter $C = 1$ and at most 2,000 optimizer iterations. It fits 60,000 decoded samples from the $n = 288$ code at $p = 0.08, 0.10$. Rank a batch of $N$ samples by score and route the top $\mathrm{round}(rN)$, using stable input order for ties.

### C.3 Circuit-level ensemble

There are four arms of twelve alternatives each. Three perturb the native mechanism logits with Gaussian scales 0.5, 1.0, and 2.0. The fourth raises the priors of a random set of 16 mechanisms to 0.3. All candidates use BP-OSD-CS-6 and a 100-iteration cap. The baseline candidate is included. The native likelihood selector resolves ties in favor of the baseline and returns $O\hat{f}$. The 48-alternative count is specific to circuit-level noise, distinct from $K = 16$ for BB code-capacity noise.

## D Additional Results

### D.1 Evaluation conventions

Each LER is the fraction of failed decodes over independently sampled errors. Error bars are two-sided 95% Wilson intervals. Samples drawn from independent random streams at the same physical error rate are pooled before computing the rate. Published NMWPM and QECCT values are the authors' reported numbers.

### D.2 Additional BB codes

Figure 6 shows the two largest constructions of Table 2 at $p = 0.12$, where PRIME reduces LER over BP-OSD-CS-6 by $5.4\times$ and $4.9\times$ and the ensemble by $21.8\times$ and $15.7\times$. On the training codes $n = 72$ and $n = 144$, PRIME reduces LER over BP-OSD-CS-6 by up to $3.9\times$ and $6.9\times$, and the ensemble does not reduce LER beyond PRIME: these codes have too few qubits for the perturbations to produce distinct candidates.

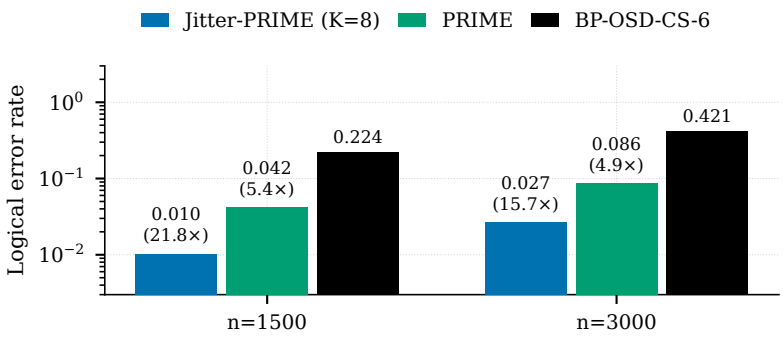


Figure 6: LER at $p = 0.12$ on the $n = 1500$ and $n = 3000$ BB constructions. Labels give the reduction over BP-OSD-CS-6.

### D.3 Circuit-level measurements

Let $N$ denote the number of samples, $F_B$ the BP-OSD failures, $R_S$ the failures the selector corrects, and $\delta$ the fraction of BP-OSD successes that the ensemble turns into failures, measured on a subset of successes. The reported LER is

$$\widehat{\mathrm{LER}}_{\mathrm{CL}} = \frac{F_B - R_S + (N - F_B)\,\delta}{N}. \tag{21}$$

A correction is logically correct when it predicts all tracked logical observables. The circuit-level gate is evaluated out of fold at $p = 0.005$ on $n = 144$. Its recall is the fraction of BP-OSD failures among the 10% of samples it routes.

### D.4 Toric codes and published baselines

Toric codes use $a = 1 + x$, $b = 1 + y$, and $\ell = m = L$. Direct generalization evaluates $L = 6, 8, 10, 12, 24$, and fine-tuning uses one model jointly trained on $L = 6, 8, 10$. NMWPM and QECCT curves use numerical values provided by the authors of NMWPM v2 (Peled et al., 2026). The direct-generalization comparison interpolates the published values in log space, and the fine-tuned comparison uses the published grid rates. The ensemble increases toric LER at every evaluated size, so this family is reported with PRIME's single decode.

## E Ablation Protocols

### E.1 Decoding pipeline

All rows of the left block of Table 1 use the same single corrections and the same ensemble candidates, on evaluation samples drawn independently of those in Figure 2, and only the routing rule differs. The gate is fitted on $n = 288$ samples at $p = 0.08$ and 0.10. Uniform random routing includes each sample with probability $r$, so its LER is $(1 - r)\,\mathrm{LER}_S + r\,\mathrm{LER}_E$ for single-decode and ensemble rates $\mathrm{LER}_S$ and $\mathrm{LER}_E$. The BP-OSD-CS-6 row uses the rate of Figure 2 at $p = 0.10$.

### E.2 Predictor components

Each variant disables one component in the configuration of Appendix B.4 and is retrained with the same schedule, seed, and checkpoint-selection rule. All variants are evaluated on identical error samples. The GNN-alone readout thresholds each predicted probability at 0.5, and PRIME passes the same predictions to BP-OSD. Table 3 gives each rate. A second training run of the full configuration differs from the reported predictor by 12%–17% in PRIME LER, which sets the resolution of the PRIME column.

Table 3: Predictor component ablation on $n = 288$ at each physical error rate.

| | GNN alone | | | PRIME ($\times 10^{-4}$) | | |
|---|---|---|---|---|---|---|
| Predictor | 0.060 | 0.065 | 0.070 | 0.060 | 0.065 | 0.070 |
| Full predictor | 0.172 | 0.238 | 0.316 | 0.68 | 1.91 | 4.93 |
| − type scaling | 0.202 | 0.273 | 0.355 | 0.63 | 1.80 | 4.77 |
| − check conditioning | 0.207 | 0.279 | 0.360 | 0.75 | 2.16 | 4.93 |
| − GRU | 0.228 | 0.304 | 0.389 | 0.70 | 2.14 | 5.00 |
| − linear attention | 0.248 | 0.328 | 0.414 | 0.69 | 2.01 | 5.63 |

The weights are shared across message-passing rounds, so the number of rounds can be changed at evaluation without retraining. Table 4 shows that the predictor is most accurate at its trained depth: it is a learned unrolling of fixed length rather than an iterative solver that improves with more rounds.

Table 4: GNN-alone LER when the number of message-passing rounds is changed at evaluation, for two independently trained predictors with $T = 4$, on $n = 72$ at $p = 0.05$.

| Rounds $T$ | 1 | 2 | 3 | 4 | 6 | 8 |
|---|---|---|---|---|---|---|
| PRIME predictor | 0.491 | 0.234 | 0.088 | 0.048 | 0.225 | 0.190 |
| Predictor trained on $n = 72$ | 0.520 | 0.261 | 0.098 | 0.042 | 0.143 | 0.697 |

### E.3 Circuit-level perturbation arms

Table 5 compares correct-candidate coverage on the same circuit-level failures. Each arm has twelve alternatives and their union has 48. On $n = 144$, coverage depends strongly on the arm: at $p = 0.005$ the $\sigma = 2.0$ arm covers 5% of failures against 89% for $\sigma = 0.5$. This concentration shows that the pool exploits structure in the DEM rather than chance, which would spread coverage evenly across arms.

Table 5: Circuit-level candidate coverage: the fraction of BP-OSD failures for which each 12-candidate arm contains a logically correct correction. Combined is the union over all 48 candidates.

| $n$ | $p$ | $\sigma = 0.5$ | $\sigma = 1$ | $\sigma = 2$ | Boost | Combined |
|---|---|---|---|---|---|---|
| 72 | 0.002 | 76% | 76% | 81% | 62% | 95% |
| 72 | 0.003 | 76% | 84% | 76% | 69% | 94% |
| 72 | 0.004 | 72% | 79% | 60% | 69% | 92% |
| 144 | 0.003 | 97% | 100% | 93% | 90% | 100% |
| 144 | 0.004 | 95% | 96% | 37% | 85% | 98% |
| 144 | 0.005 | 89% | 83% | 5% | 81% | 96% |

### E.4 Candidate selection

We compare minimum-weight selection with logical-class pooling on identical saved candidates from the $n = 288$ code. These pools use native priors, 64 perturbations plus the BP-OSD correction, and a 100-iteration BP cap, so the comparison isolates the selection rule. Class pooling corrects more failures in both sectors and at both rates (Table 6).

Table 6: Candidate selection on $n = 288$: the fraction of BP-OSD failures corrected by each rule on the same 64-candidate native-prior pools.

| Sector, $p$ | Min. weight | Class pooling |
|---|---|---|
| X, 0.07 | 81.3% | 83.9% |
| Z, 0.07 | 83.6% | 87.9% |
| X, 0.1 | 60.3% | 63.4% |