GROWING DYNAMIC DECODING GRAPH FOR HIGH-ACCURACY QUANTUM ERROR CORRECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Quantum error correction requires high-fidelity error priors to improve the decoding accuracy. Existing approaches mainly focus on estimating the probabilities or weights of decoding priors, while largely neglecting their graph topology, which limits the error correction performance even when the edge weights are well estimated. This motivates us to ask whether the decoding graph itself can be automatically constructed from syndrome samples and a given decoder. To this end, we introduce the **PrunerNet** that applies physics-guided sparse self-attention over the hyperedges in the detector error model. Each hyperedge maintains a learnable embedding to determine its retention probability, and attention is restricted to pairs of hyperedges that share stabilizer check components, capturing physically meaningful correlations in the noise model. By growing dynamic decoding graphs with reinforcement learning and under the direct supervision of the logical error rates, we achieve high-precision decoding priors without human-engineered calibration, enabling high-fidelity quantum error correction. Benchmarking on error correction circuits of the surface code demonstrates the effectiveness of learning dynamic decoding graph structures compared with leading methods that optimize prior probabilities alone. Across correlated-error and randomly reordered-circuit benchmarks and relative to fixed-topology calibration, our method averagely reduces logical error rates by 3.8% and 13.4%, respectively. We have also developed **Auto-DEM**, a modular compilation framework for detector error model, which enables joint optimization of the weights and topology of arbitrary decoding graphs, paving the way towards fault-tolerant quantum computing. All codes and evaluations are available at the Anonymous Github, and will be open-sourced upon publication.

1 INTRODUCTION

Quantum computing has the potential to tackle difficult problems beyond classical computational capacity (Shor, 1999; Grover, 1996; Lloyd, 1996; Peruzzo et al., 2014; Babbush et al., 2026), but noise in physical qubits remains a major obstacle. Fortunately, quantum error correction (QEC) (Shor, 1995; Gottesman, 1997) offers a promising route by encoding logical qubits into redundant physical qubits and performing computation at the logical level. In this way, the effect of physical noise can be suppressed by encoding rather than directly eliminated at the hardware level. More importantly, the threshold theorem (Aharonov & Ben-Or, 1997; Kitaev, 1997; Knill et al., 1998) guarantees that, below a physical error-rate threshold, increasing the code distance can in principle suppress logical errors to arbitrarily low levels, providing a route to fault-tolerant quantum computation (FTQC).

The performance of QEC depends on both encoding and decoding. While better quantum error-correcting codes (Dennis et al., 2002; Bravyi et al., 2023) and circuit-level implementations improve the intrinsic protection of logical information, an inaccurate or mismatched decoder can still fail even when the code has sufficient error-correction capability, making high-fidelity decoding (Fowler, 2013; Higgott et al., 2023; Bausch et al., 2024; Higgott & Gidney, 2025; Beni et al., 2025) an essential component of scalable FTQC. The decoding prior, which characterizes physical errors throughout the quantum circuit and is typically modeled as a hypergraph that guides the decoder, known as the detector error model (DEM) (Gidney, 2021), is therefore critical for QEC.

Decoding priors are typically calibrated on manually constructed decoding graphs. However, a fixed topology built from the human-designed DEM compilation rules may miss hardware error mecha-

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.

nisms such as crosstalk (Zhao et al., 2022; Wesdorp et al., 2026), leakage (McEwen et al., 2021), and long-range interactions (McEwen et al., 2022; Google Quantum AI and Collaborators, 2025). Furthermore, hardware-specific constraints and compiler optimization objectives can result in different stabilizer measurement schedules (Yin et al., 2025; Liu et al., 2026; Zhang et al., 2026; Viszlai et al., 2026), altering circuit-level error propagation and giving rise to heterogeneous error topologies in the resulting DEMs. More fundamentally, the DEM representation is not uniquely determined by the circuit. As noted by McEwen et al. (2023), choosing appropriate detectors and detecting regions from an unannotated circuit is also non-trivial; different choices can expose different error structures to the decoder and affect logical accuracy even under the same circuit and hardware noise.

Unlike methods that calibrate weights on a fixed decoding graph, we optimize the DEM hypergraph topology while retaining efficient weight estimation. Specifically, starting from the standard DEM, we construct a learnable set of candidate hyperedges and dynamically optimize their selection through reinforcement learning directly guided by the logical error rate (LER). The main contributions of this work are as follows:

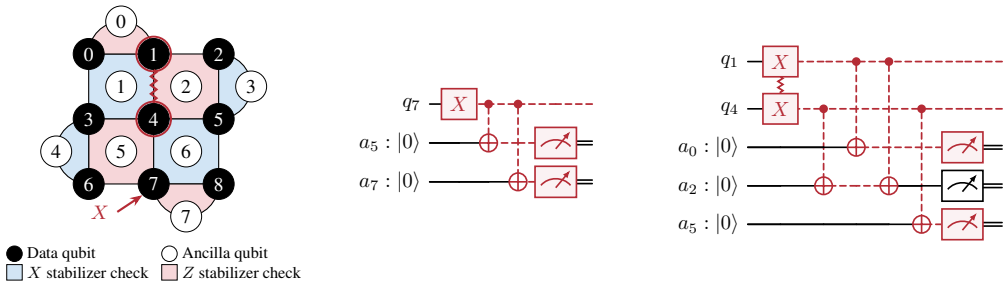
- We design **PrunerNet** to model the retention probability of each hyperedge in the decoding prior, with physics-guided attention that explicitly captures adjacency among hyperedges. Together with graph extension and splitting, it enables dynamic hyperedge selection while maintaining decoder-compatible graph structures throughout optimization.
- We propose an end-to-end reinforcement learning pipeline that directly optimizes the DEM topology under LER supervision. On circuits with correlated errors and randomized stabilizer ordering, it consistently outperforms conventional approaches that optimize edge weights alone.
- We develop **Auto-DEM**, a modular and flexible DEM compilation framework that enables joint optimization of the weights and topology of decoding graphs for quantum error correction. This unified pipeline for DEM construction and optimization facilitates both algorithmic research and experimental evaluation of near-term QEC systems.

2 Related Work

Previous works on optimizing decoding priors for QEC have extensively studied edge-weight estimation on fixed decoding hypergraphs, leading to increasingly accurate and efficient methods.

Syndrome-statistical-based methods. A natural direction is to exploit statistical information from syndrome samples to optimize the hyperedge weights. Such conventional numerical methods include Correlation-based estimation (Chen et al., 2022; Google Quantum AI, 2021; Acharya et al., 2022) and its dual, Parity-based estimation implemented via the Walsh–Hadamard transform (Blume-Kohout & Young, 2025; Arms et al., 2025), both are widely used in current hardware experiments and demonstrations (Acharya et al., 2022; Google Quantum AI and Collaborators, 2025; He et al., 2025). Despite their computational efficiency, the finite-order statistics employed by these methods, together with the intrinsic degeneracy of QEC (Dennis et al., 2002), generally limit them to approximate rather than globally optimal estimation of the underlying error probabilities. More recently, maximum-likelihood estimation has also been developed by Cao et al. (2026), which directly optimizes noise parameters by maximizing the global syndrome likelihood and further improves the logical decoding performance, with exact likelihood evaluation enabled tensor-network contraction. However, the computational cost of tensor-network contraction grows rapidly with the code size, making the approach computationally demanding and challenging to scale.

Logical-error-rate-supervised optimization. Moving beyond syndrome-based estimation, Google Quantum AI first introduced an end-to-end learning paradigm (Sivak et al., 2024) that directly optimizes decoding priors against the ultimate QEC objective, i.e., the LER, using reinforcement learning. This motivates our work by demonstrating the potential of directly supervising optimization with logical decoding performance, rather than relying on intermediate syndrome statistics. However, their optimization remains restricted to the edge weights of a predefined decoding graph, leaving the graph topology fixed throughout training. Notably, neither the code nor the training data are publicly available, making a direct and fair benchmarking infeasible.



(a) Rotated surface code (b) Single-qubit error (c) Correlated error

Figure 1: Detection of single-qubit and correlated errors in a code distance $d = 3$ rotated surface code. (a) Surface code patch. The red arrow at q_7 and the red zigzag link between q_1 and q_4 locate the two separate error examples in (b) and (c), respectively. (b) An X_7 error flips the Z stabilizer checks at a_5 and a_7. (c) A correlated X_1X_4 error flips the outcomes at a_0 and a_5.

3 Background

3.1 Quantum Error Correction

Stabilizer codes provide a widely used framework for QEC (Gottesman, 1997). We introduce their decoding problem through a parity-check matrix, as in classical linear codes. In the binary symplectic form (BSF), a Pauli error is represented by a binary vector $\boldsymbol{e} = (\boldsymbol{e}_X|\boldsymbol{e}_Z)$. Incorporating the symplectic product into the parity-check matrix H, the observed syndrome $\boldsymbol{s}$ satisfies

$$H\boldsymbol{e} = \boldsymbol{s} \pmod 2. \tag{1}$$

Given an observed syndrome, a decoder aims to identify the most likely compatible error:

$$\hat{\boldsymbol{e}} = \arg \max_{\boldsymbol{e}:\, H\boldsymbol{e}=\boldsymbol{s}} P(\boldsymbol{e}), \tag{2}$$

where $P(\mathbf{e})$ defines the decoding prior over possible errors.

For practical circuit-level noise models (Dennis et al., 2002), this formulation naturally generalizes from *stabilizer syndromes* to *detector events* (Gidney, 2021). A detector is a parity of syndrome measurements whose outcome can be flipped by circuit-level faults (Gidney, 2021; McEwen et al., 2023). Denoting the observed detector events by $\boldsymbol{x}$, circuit-level error detection can be expressed as

$$M\boldsymbol{e} = \boldsymbol{x} \pmod 2, \tag{3}$$

where each column of the detection matrix M specifies the detectors flipped by an elementary circuit-level error mechanism. This formulation generalizes the parity-check on data-qubit errors only in Equation 1 to spatial-temporal errors in circuit-level noise.

Together with their probabilities, these error mechanisms define the aforementioned *detector error model* (DEM). Each error mechanism induces a hyperedge connecting the detectors it flips, forming a weighted decoding hypergraph. The DEM thus serves as the decoding prior for distinguishing among error mechanisms consistent with the observed detector events. Accordingly, circuit-level decoding can be viewed as

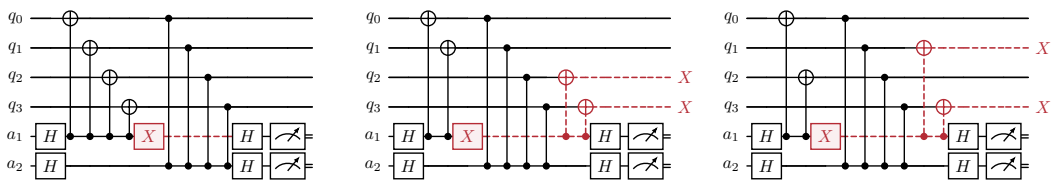
$$\hat{\boldsymbol{e}} = \arg \max_{\boldsymbol{e}:\, M\boldsymbol{e}=\boldsymbol{x}} P(\boldsymbol{e} \mid \mathrm{DEM}). \tag{4}$$

In this representation, the connectivity encoded by M determines the topology of the decoding hypergraph, while the probabilities of individual error mechanisms determine its hyperedge weights.

3.2 Errors Beyond Standard Noise Model

In practice, however, quantum errors can exhibit structures that go beyond the conventional topology assumptions derived from standard error correction circuits and prescribed noise models.

The first example is correlated errors, as shown in Figure 1. Under the conventional assumption, errors occur only on individual qubits or between qubits that directly entangled through multi-qubit



(a) Standard (b) Equivalent ordering I (c) Equivalent ordering II

Figure 2: Different fault propagation in equivalent stabilizer measurement circuits. All three circuits implement the same stabilizer measurements of $X_0X_1X_2X_3$ and $Z_0Z_1Z_2Z_3$. The red box marks an X error inserted before the first CZ gate. Dashed red lines indicate error propagation, and red X labels mark the resulting different data-qubit errors: (a) I, (b) X_2X_3, and (c) X_1X_3.

gates, as illustrate in Figure 1b. In practice, however, correlated errors can introduce error mechanisms beyond this local topology. For instance, spatially nearby superconducting qubits may experience correlated errors due to crosstalk or shared physical interactions even without direct gate operations. As illustrated in Figure 1c, correlated X errors on q_1 and q_4 produce a detector pattern absent from the standard DEM. Without the corresponding hyperedge in the decoding prior, the decoder may misidentify the error mechanism, increasing the LER.

Another example is the stabilizer circuits with random check order, as illustrated in Figure 2. Although they preserve the ideal stabilizer checks, different orderings can produce different fault-propagation patterns, making the standard DEM unsuitable for high-accuracy decoding.

These examples motivate learning the decoding graph topology itself to better capture such error mechanisms and improve logical decoding accuracy.

4 METHODS

The overview of our framework, namely **Auto-DEM**, which allows joint optimization of a dynamic decoding graph topology and the corresponding weight estimation, is illustrated in Figure 3. In the following sections, we introduce the three key components of **Auto-DEM**: (1) the decoding graph compiler, (2) the neural network model, and (3) the reinforcement learning strategy.

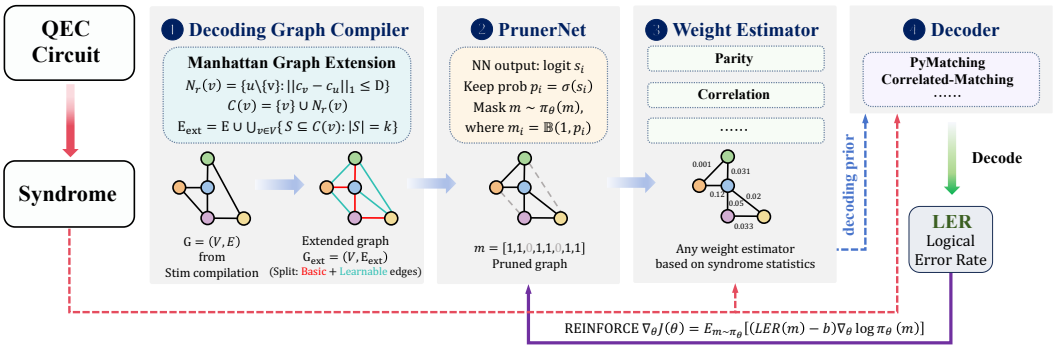


Figure 3: Overview of **Auto-DEM**. The decoding graph compiler extends the nominal decoding graph with fixed basic edges and learnable candidates. **PrunerNet** samples a binary mask to select the graph topology, after which the retained edges are weighted and evaluated by a conventional decoder. The resulting LER directly guides reinforcement learning of the decoding prior.

4.1 DECODING GRAPH COMPILER

In this section, we introduce how we construct the extended and learnable decoding graph from the standard DEM compilation.

Manhattan graph extension. To allow more possibilities, we expand Stim's compiled DEM's hyperedge set to accommodate additional error correlations. Because physical interactions are predominantly local, we use Manhattan distance to restrict new candidates to nearby detectors in space and time.

Let $G = (V, E)$ denote the original DEM hypergraph, with detector vertices V and hyperedges E. Each vertex $v \in V$ has coordinates $\mathbf{c}_v = (x_v, y_v, t_v) \in \mathbb{R}^3$, where (x_v, y_v) gives the physical location of its ancilla stabilizer qubit and t_v the syndrome extraction round. Under a Manhattan-distance threshold D, the local neighborhood of v is

$$\mathcal{N}_D(v) = \{u \in V \setminus \{v\} \mid \|\mathbf{c}_v - \mathbf{c}_u\|_1 \le D\}. \tag{5}$$

For each vertex v, we form a candidate set from v and its neighbors. For a chosen hyperedge order k, we add all size-k subsets of this set to the original hyperedge set:

$$\mathcal{C}(v) = \{v\} \cup \mathcal{N}_D(v), \qquad E_{ext} = E \cup \bigcup_{v \in V} \{S \subseteq \mathcal{C}(v) \mid |S| = k\}. \tag{6}$$

This locality-constrained construction captures additional error correlations among nearby detectors without enumerating all k-vertex subsets of V. Finally, we can get the extended graph from the original hyperedge vertices and the extended hyperedge set, namely $G_{ext} = (V, E_{ext})$.

Graph splitting. The extended graph G_{ext} provides the candidate hyperedges for topology optimization. However, directly optimizing all hyperedges may break graph connectivity and lead to decoding failures. To ensure valid decoding throughout optimization, we adopt the decomposition illustrated in Figure 3. Specifically, we use the hypergraph from the phenomenological noise model (Dennis et al., 2002) as a connected spanning subgraph, termed the *basic graph*, by including all horizontal and vertical edges from the original decoding graph G. The remaining hyperedges in G_{ext} constitute the *learnable graph*, which is dynamically grown and pruned during optimization.

4.2 Transformer Edge Pruner

To represent hyperedge retention probabilities, we propose **PrunerNet**, which predicts a retention score for each hyperedge in the *learnable graph*. As illustrated in Figure 4, it has L transformer layers with H attention heads and an embedding dimension of d (Vaswani et al., 2017).

Since the architecture relies solely on the hypergraph connectivity and does not require syndrome information or logical states as input, the pruning decision is made purely from the graph structure. Each hyperedge is represented by a learnable embedding $h_i \in \mathbb{R}^d$. All hyperedge embeddings are jointly processed by the Transformer layers to capture structural dependencies among hyperedges.

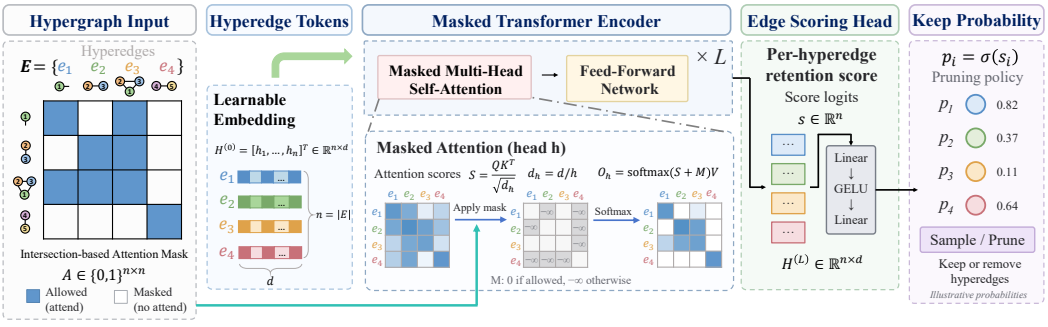


Figure 4: Architecture of the **PrunerNet**. The model maintains a learnable embedding for each hyperedge in the *learnable graph* and uses a transformer encoder with an intersection-based attention mask, allowing self-attention only between adjacent hyperedges. A lightweight MLP head then predicts a score for each hyperedge, which is converted to a retention probability for sampling.

We restrict the self-attention mechanism using an intersection-based attention mask as illustrated in Figure 4. The resulting embeddings from the transformer layers are then passed through a lightweight scoring head consisting of two linear layers with a `GELU` activation, returning a scalar logit s_i representing the importance of hyperedge e_i. Higher scores indicate that the hyperedge is more likely to be retained by the pruning policy, which will be discussed later.

4.3 Reinforcement Learning Strategy

With the extended decoding graph and **PrunerNet** in place, we optimize the pruning policy $\pi_\theta(\mathbf{m})$ under the direct supervision of the decoded LER. As illustrated in Figure 3, weight estimation, DEM construction, and decoding form a black-box pipeline. This allows different estimator–decoder combinations without changing the learning objective or downstream components.

Stochastic policy with connectivity-constrained sampling. During training, the pruner induces an independent Bernoulli policy over the learnable edges N_l

$$\pi_\theta(\mathbf{m}) = \prod_{i=1}^{N_\mathrm{l}} p_i^{m_i}(1-p_i)^{1-m_i}, \qquad p_i = \sigma(s_i/\tau), \tag{7}$$

where s_i is the pruning score of learnable hyperedge e_i and τ is a sampling temperature. Basic edges are always retained and do not enter this product. In the implementation, we use its log probabilities for policy updates. To avoid overly aggressive pruning, we reject and resample masks whose total retention ratio falls below a prescribed minimum above the basic-edge fraction. If repeated resampling still fails to meet this threshold, training will be terminated.

Logical error rate as the learning signal. For each sampled mask $\mathbf{m}$, the kept hyperedges are first assigned weights by the weight estimator, the resulting DEM is then constructed, and the given decoder is applied to a randomly selected subset of N_reward training shots, yielding the LER $L(\mathbf{m})$. Since the LER can vary by orders of magnitude over the course of training, we adopt a logarithmic rescaling similar to that used by Sivak et al. (2024)

$$\tilde{L}(\mathbf{m}) = 20\log_{10} L(\mathbf{m}), \tag{8}$$

which preserves the ordering of candidate masks while keeping the learning signal on a comparable scale across noise regimes: relative improvements at low LER are no longer overwhelmed by absolute fluctuations at high LER. At each epoch, a batch of masks is sampled from the current policy and evaluated in parallel, and $\tilde{L}(\mathbf{m})$ serves as the per-mask cost in the policy-gradient update.

Policy-gradient objective. We use an exponential moving average (EMA) of the batch-mean rescaled cost as the baseline: $b_t = 0.9b_{t-1} + 0.1\overline{L}_t$, where $\overline{L}_t$ is the mean of $\tilde{L}(\mathbf{m})$ over the masks sampled at epoch t. The baseline is initialized to the first batch mean and treated as a constant during backpropagation. Using $b = b_t$, the REINFORCE (Williams, 1992) update is

$$\nabla_\theta J(\theta) = \mathbb{E}_{\mathbf{m}\sim\pi_\theta}\left[\big(\tilde{L}(\mathbf{m}) - b\big)\nabla_\theta \log \pi_\theta(\mathbf{m})\right] - \lambda_t \nabla_\theta \mathcal{H}(\pi_\theta), \tag{9}$$

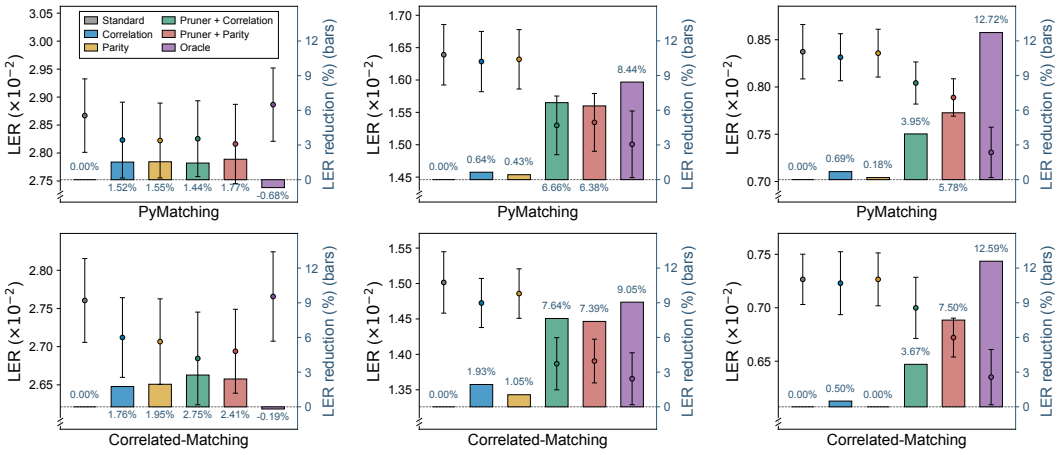
where $\mathcal{H}(\pi_\theta)$ is the entropy of the Bernoulli policy. Additional details on entropy coefficient λ_t, initialization and optimization are provided in the Appendix A.2.

5 Evaluation

We evaluate **Auto-DEM** under two forms of decoding-prior mismatch: circuits with correlated errors and circuits with random stabilizer measurement order. We first verify that **PrunerNet** can learn a final DEM typology with decoding LER comparable to the circuit-compiled reference under the SI1000 noise model (Bausch et al., 2024; Google Quantum AI and Collaborators, 2025), as detailed in Appendix A.1.

5.1 Experimental Setup

Over the course of this work, we utilize Stim (Gidney, 2021) to simulate all the quantum error correction circuits. We begin from the standard SI1000 noise model and build up circuits with correlated errors and circuits with random stabilizer measurement order. Although our framework is, in principle, applicable to arbitrary quantum error correcting codes, we choose the surface code (Dennis et al., 2002; Fowler et al., 2012; Acharya et al., 2022; Google Quantum AI and Collaborators, 2025) as the experimental code, as it is one of the most promising routes towards FTQC. We use the surface code with $d \in \{3, 5, 7\}$, $r = 3$ syndrome-extraction rounds, and the SI1000-based physical error parameter $p = 0.003$ in the following experiments. These code distances are representative of



(a) $d = 3$. (b) $d = 5$. (c) $d = 7$.

Figure 5: LER under the correlated-error model. Points (left axes) show mean LER $\pm$ one standard deviation over 10 independent 100k-shots repetitions. Bars (right axes) show the reduction ratio relative to Standard under the same decoder, with negative values indicating degradation. Standard and Oracle are compiled without and with the additional channel, respectively.

current experimental demonstrations (Google Quantum AI and Collaborators, 2025; He et al., 2025) while keeping the computational cost manageable. During the training and evaluation process, we choose the Correlation- and Parity-based methods to achieve efficient DEM weight estimation. We also select two relatively fast matching-based decoders, PyMatching (Higgott & Gidney, 2025) and Correlated-Matching (Fowler, 2013), as the decoding backends.

5.2 Effectiveness in correcting correlated errors

We introduce additional correlated errors absent from the nominal DEM. Motivated by parasitic exchange interactions between superconducting qubits (Zhao et al., 2022) and residual ZZ coupling during idling (Goto, 2022), we add a local two-qubit channel to the standard surface-code memory circuit. To isolate a storage-window mechanism in this benchmark, we assume that, during each ancilla measurement-and-reset interval, every nearest-neighbor pair on the data-qubit lattice independently undergoes an exchange rotation $U = \exp[-i\theta(XX+YY)/2]$. Pauli twirling this rotation yields (Geller & Zhou, 2013)

$$P(XX) = P(YY) = a(1-a), \qquad P(ZZ) = a^2, \qquad a = 1 - \sqrt{1-q}, \tag{10}$$

where q is the total probability of nonidentity. This introduces joint data-qubit faults whose detector supports need not belong to the nominal DEM.

For the additional correlated channel, we set $q = 0.2p = 6 \times 10^{-4}$ per pair per storage interval. The coefficient 0.2 is motivated by the ratio between CZ-crosstalk and baseline CZ-error probabilities reported in Google's error budget (Google Quantum AI and Collaborators, 2025).

Correlation and Parity recalibrate only the weights of the fixed Standard DEM compiled by Stim before the additional correlated-noise channel is introduced. Our method also adapts the Standard DEM topology through graph extension and **PrunerNet**. The Oracle DEM, compiled from the augmented circuit, serves as a ground-truth reference for the achievable decoding performance and is not used in any training. Figure 5 shows similar trends under PyMatching and Correlated-Matching. At $d = 3$, methods perform similarly, and Oracle slightly underperforms Standard, indicating that a DEM compiled directly from the circuit used to generate the samples does not necessarily provide a lower bound on the LER. At $d = 5$ and $d = 7$, both weight estimators combined with **PrunerNet** outperform their fixed-topology counterparts. Across all distances and both decoders, **PrunerNet** with Correlation and Parity reduces the LER relative to Standard by 4.35% and 5.20%, respectively, whereas their fixed-topology versions achieve reductions of only 1.17% and 0.86%.

Table 1: Comparison on the logical error rate on the random-ordered stabilizer circuit.

Distance	Decoder	Standard	Correlation	Parity	Pruner		Oracle
					Correlation	Parity	
3	Pymatching	0.0243(12)	0.0242(13)	0.0241(15)	0.0189(12)	0.0195(12)	0.0193(2)
3	Corr-Matching	0.0234(15)	0.0234(15)	0.0240(16)	0.0188(11)	0.0189(11)	0.0176(5)
5	Pymatching	0.0129(12)	0.0119(11)	0.0117(11)	0.0107(12)	0.0110(9)	0.0101(3)
5	Corr-Matching	0.0126(10)	0.0111(10)	0.0112(9)	0.0099(10)	0.0104(9)	0.0081(2)
7	Pymatching	0.0095(10)	0.0088(9)	0.0087(9)	0.0076(7)	0.0074(6)	0.0068(2)
7	Corr-Matching	0.0078(9)	0.0069(9)	0.0070(9)	0.0063(8)	0.0065(10)	0.0049(2)

Note: Cell colors indicate relative LER compared with the DEM compiled from the standard circuit under the same code distance and decoder. Lighter color indicates lower LER. The mean and standard deviation are evaluated over 10 repetitions of 100k independent shots.

5.3 Robustness Against Noise from Random-Ordered Stabilizer Circuits

We also evaluate the robustness against noise from random-ordered stabilizer circuits of our framework. We randomize the order of stabilizer checks while keeping the correctness using the SMT solver (Yin et al., 2025; Zhang et al., 2026). The reordered circuit follows the same noise channels as the standard SI1000 circuit, while introducing richer error-propagation mechanisms, thus the DEM compiled from the standard circuit may no longer match the sampled circuit. Similar to Section 5.2, all evaluated methods start from the standard DEM compiled from the standard circuit, while the Oracle DEM is compiled from the randomized circuit solely as a ground-truth reference for evaluation and is never exposed to any method during training or optimization.

Table 1 summarizes the decoding performance. Across the six combinations of code distance and decoder, **PrunerNet** with Correlation and Parity further reduces the LER by an average of 14.1% and 12.6%, respectively, relative to the corresponding fixed-topology estimators. Relative to the Standard DEM, the average reductions are 19.9% and 18.3%. These consistent gains in logical accuracy across code distances and decoders demonstrate the benefit of adapting the graph topology alongside its weights when the implemented stabilizer schedule differs from the nominal one.

6 Ablation Study

We first test whether graph extension without pruning improves decoding accuracy, then ablate the pruner capacity, graph-extension configuration, entropy regularization, and reward sample size N_{reward}. We use the $d = 5$ SI1000 noise model, as detailed in Appendix A.1. The shared default configuration uses a model dimension of 64, 4 heads, 2 layers, initial entropy coefficient $\lambda_0 = 1.0$, $N_{\text{reward}} = 50$k, and 200k shots for each of training, validation, and testing. Final LERs are reported as the mean and standard deviation over 10 repetitions of 100k shots.

Random extension. Figure 6 examines whether graph extension alone improves decoding. Starting from the basic graph, we randomly add learnable edges and construct DEMs using the Parity method. The LER generally increases with the number of added edges, highlighting the importance of topology selection rather than graph extension alone.

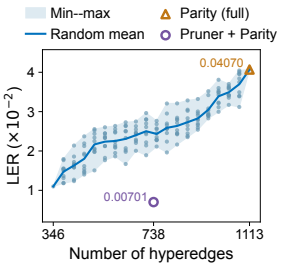
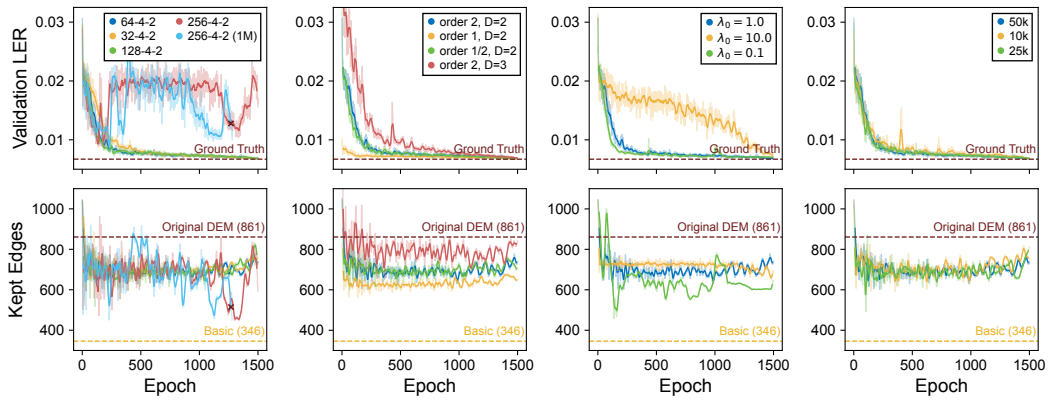


Figure 6: LER under Parity calibration as learnable edges are randomly added to the fixed basic graph.

Pruner capacity. Figure 7a compares pruners with different model dimensions. We vary the model dimension over $\{32, 64, 128, 256\}$ with four heads and two layers, denoting each configuration as dimension-heads-layers. Larger models converge faster initially, but 256-4-2 becomes unstable. We hypothesized that training with more samples might stabilize this higher-capacity model; however, increasing training samples to 1M shots did not prevent the instability. The 32-4-2 model degrades the final LER to $0.00727(16)$, compared with $0.00703(15)$ for default 64-4-2. The default model therefore provides the best balance between convergence speed, stability, and accuracy, consistent with Appendix A.3.3.



(a) Pruner model dimension. (b) Extension order. (c) Initial λ_0. (d) Reward sample size.

Figure 7: Ablations of (a) pruner capacity, (b) graph-extension configuration, (c) the initial entropy coefficient, and (d) reward sample size. Red crosses indicate termination of the 256-4-2 runs when the retained-edge ratio falls below the prescribed minimum required by the sampling constraint described in Section 4.3.

Graph extension. Figure 7b compares the default extension (Manhattan distance $D = 2$, order 2) with order-1, mixed order-1/2, and $D = 3$ order-2 variants. Larger candidate sets retain more edges without improving accuracy, suggesting that unnecessarily enlarging the policy search space can hinder optimization. Throughout this work, we also restrict candidates to graph-like hyperedges of order at most two for compatibility with matching-based decoders.

Entropy coefficient. Figure 7c shows that an intermediate entropy coefficient performs best. Excessive exploration with $\lambda_0 = 10.0$ delays convergence and yields an LER of $0.00766(18)$, while premature policy sharpening with $\lambda_0 = 0.1$ yields $0.00732(16)$, compared with $0.00703(15)$ for the default $\lambda_0 = 1.0$. Similar retained-edge counts indicate that entropy mainly affects which edges are selected rather than how many are retained.

Reward sample size. Figure 7d shows that all three reward sample sizes lead to convergence with similar retained-edge counts. Evaluations indicate that reducing N_{reward} from the default 50k to 10k degrades the final LER to $0.00736(22)$, while 25k achieves a comparable $0.00697(12)$. These results suggest that 25k shots suffice in this setting, while reducing the sample size to 10k may compromise decoding accuracy. In general, N_{reward} should be chosen according to the typical LER scale of the target QEC setting, which varies with factors such as noise level and code distance.

7 Conclusion

In this work, we introduced **Auto-DEM**, a framework that treats the detector error model as an optimizable decoding prior rather than a fixed output of circuit compilation. Auto-DEM jointly optimizes the topology and weights of the decoding graph by combining physically constrained graph extension, topology learning, and modular weight estimation, with the logical error rate serving as the end-to-end optimization objective. Across different forms of mismatch between the nominal noise model and the underlying error processes, our results show that optimizing the decoding graph can capture error structures that are important for decoding, thus achieving high decoding accuracy.

More broadly, these results suggest that the decoding graph need not be tied to a predefined circuit-level noise model. Instead, its topology can itself be learned and adapted according to the ultimate objective of quantum error correction: suppressing logical errors. This perspective provides a path toward decoding priors that adapt to error mechanisms beyond standard noise assumptions and to evolving device behavior. While direct optimization through logical error rates remains computationally and sample intensive, improving the efficiency and scalability of this approach may enable adaptive decoding graphs to be learned directly from experimental quantum hardware.

AI USE STATEMENT

In this work, we used generative AI tools to assist with grammatical checking and language polishing, and to help standardize the code structure of our repository. We did not use generative AI tools to generate manuscript sections or experimental results. All AI-assisted outputs were reviewed and verified by the authors. We take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

The Anonymous Github repository accompanying this submission contains the complete implementation of **Auto-DEM**, including the decoding-graph compiler, the neural network model, the correlation- and Parity-based weight estimators, and the training and evaluation pipeline. It also provides the data generation scripts for the simulated surface code experiments, the exact configuration files for all reported runs, and scripts for reproducing the figures in evaluation.

REFERENCES

Rajeev Acharya, I. Aleiner, Richard Allen, Trond I. Andersen, Markus Ansmann, Frank Arute, Kunal Arya, Abraham T. Asfaw, Juan Atalaya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, João Marcos Vensi Basso, Andreas Bengtsson, Sergio Boixo, Gina Bortoli, Alexandre Bourassa, Jenna Bovaird, Leon Brill, Mick Broughton, Bob B Buckley, David A. Buell, Tim Burger, Brian Burkett, Nicholas Bushnell, Yu Chen, Zijun Chen, Benjamin Chiaro, J. Zachery Cogan, Roberto Collins, P. N. Conner, William Courtney, Alexander L. Crook, B Curtin, Dripto M. Debroy, A Del Toro Barba, Sean Demura, Andrew Dunsworth, Daniel Eppens, Catherine Erickson, Lara Faoro, Edward Farhi, Reza Fatemi, Leslie Flores Burgos, Ebrahim Forati, Austin G. Fowler, Brooks Foxen, William Giang, Craig Gidney, Dar Gilboa, Marissa Giustina, Alejandro Grajales Dau, Jonathan A. Gross, S. Habegger, Michael C. Hamilton, Matthew P. Harrigan, Sean D. Harrington, Oscar Higgott, Jeremy P. Hilton, Michael J. Hoffmann, Sabrina Hong, Trent Huang, Ashley Huff, William J. Huggins, L. B. Ioffe, Sergei V. Isakov, Justin Iveland, Evan Jeffrey, Zhang Jiang, Cody Jones, Pavol Juhás, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Tanuj Khattar, Mostafa Khezri, M'aria Kieferov'a, Seon Kim, Alexei Kitaev, Paul V. Klimov, Andrey R. Klots, Alexander N. Korotkov, Fedor Kostritsa, John Mark Kreikebaum, David Landhuis, Pavel Laptev, Kim Ming Lau, Lily Laws, Joonho Lee, Kenny Lee, Brian J. Lester, Alexander T Lill, Wayne Liu, Aditya Locharla, Erik Lucero, Fionn D. Malone, Jeffrey Marshall, Orion Martin, Jarrod R. McClean, Trevor McCourt, Matthew J. McEwen, Anthony Megrant, Bernardo Meurer Costa, Xiao Mi, Kevin C. Miao, Masoud Mohseni, Shirin Montazeri, Alexis Morvan, Emily Mount, Wojciech Mruczkiewicz, Ofer Naaman, Matthew Neeley, Charles J. Neill, Ani Nersisyan, Hartmut Neven, Michael Newman, Jiun How Ng, Anthony Nguyen, Murray L. Nguyen, Murphy Yuezhen Niu, Thomas E. O'Brien, Alexander Opremcak, J. Platt, Andre Petukhov, Rebecca Potter, Leonid P. Pryadko, Chris Quintana, Pedram Roushan, Nicholas C. Rubin, Negar Saei, Daniel Thomas Sank, K Sankaragomathi, Kevin J. Satzinger, Henry F. Schurkus, C. Schuster, Michael Shearn, Aaron Shorter, Vladimir Shvarts, Jindra Skruzny, Vadim N. Smelyanskiy, William C. Smith, George Sterling, Doug Strain, Yuan Su, Marco Szalay, Alfredo Torres, Guifre Vidal, Benjamin Villalonga, Catherine Vollgraff Heidweiller, Theodore White, Chen Xing, Z. Jamie Yao, Ping Yeh, Juhwan Yoo, Grayson Young, Adam Zalcman, Yaxing Zhang, and Ningfeng Zhu. Suppressing quantum errors by scaling a surface code logical qubit. *Nature*, 614:676 – 681, 2022.

Dorit Aharonov and Michael Ben-Or. Fault-tolerant quantum computation with constant error. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pp. 176–188, 1997.

Kregg Elliot Arms, Martin James McHugh, Joseph Edward Nyhan, William Frederick Reus, and James Loudon Ulrich. Estimating detector error models on google's willow. *arXiv preprint arXiv:2512.10814*, 2025.

Ryan Babbush, Adam Zalcman, Craig Gidney, Michael Broughton, Tanuj Khattar, Hartmut Neven, Thiago Bergamaschi, Justin Drake, and Dan Boneh. Securing elliptic curve cryptocurrencies against quantum vulnerabilities: Resource estimates and mitigations. *PRX Quantum*, 7(3): 031001, 2026.

Johannes Bausch, Andrew W. Senior, Francisco J. H. Heras, Thomas Edlich, Alex Davies, Michael Newman, Cody Jones, Kevin J. Satzinger, Murphy Yuezhen Niu, Sam Blackwell, George Holland, Dvir Kafri, Juan Atalaya, Craig Gidney, Demis Hassabis, Sergio Boixo, Hartmut Neven, and Pushmeet Kohli. Learning high-accuracy error decoding for quantum processors. *Nature*, 635: 834 – 840, 2024.

Laleh Aghababaie Beni, Oscar Higgott, and Noah Shutty. Tesseract: A search-based decoder for quantum error correction. *arXiv preprint arXiv:2503.10988*, 2025.

Robin Blume-Kohout and Kevin Young. Estimating detector error models from syndrome data. *arXiv preprint*, 2025.

Sergey Bravyi, Andrew W. Cross, Jay M. Gambetta, Dmitrii L. Maslov, Patrick Rall, and Theodore J. Yoder. High-threshold and low-overhead fault-tolerant quantum memory. *Nature*, 627:778 – 782, 2023.

Hanyan Cao, Dongyang Feng, Cheng Ye, and Feng Pan. Differentiable maximum likelihood noise estimation for quantum error correction. *arXiv preprint arXiv:2602.19722*, 2026.

Edward H Chen, Theodore J Yoder, Youngseok Kim, Neereja Sundaresan, Srikanth Srinivasan, Muyuan Li, Antonio D Córcoles, Andrew W Cross, and Maika Takita. Calibrated decoders for experimental quantum error correction. *Physical Review Letters*, 128(11):110504, 2022.

Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. Topological quantum memory. *Journal of Mathematical Physics*, 43(9):4452–4505, September 2002. ISSN 1089-7658.

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

Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3), September 2012. ISSN 1094-1622.

Michael R. Geller and Zhongyuan Zhou. Efficient error models for fault-tolerant architectures and the pauli twirling approximation. *Physical Review A*, 88(1), 2013. ISSN 1094-1622.

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

Google Quantum AI. Exponential suppression of bit or phase errors with cyclic error correction. *Nature*, 595(7867):383–387, 2021.

Google Quantum AI and Collaborators. Quantum error correction below the surface code threshold. *Nature*, 638(8052):920–926, 2025.

Hayato Goto. Double-transmon coupler: Fast two-qubit gate with no residual coupling for highly detuned superconducting qubits. *Physical review applied*, 18(3):034038, 2022.

Daniel Gottesman. *Stabilizer Codes and Quantum Error Correction*. PhD thesis, California Institute of Technology, Pasadena, CA, 1997.

Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 212–219, 1996.

Tan He, Weiping Lin, Rui Wang, Yuan Li, Jiahao Bei, Jianbin Cai, Sirui Cao, Danning Chen, Kefu Chen, Xiawei Chen, et al. Experimental quantum error correction below the surface code threshold via all-microwave leakage suppression. *Physical Review Letters*, 135(26):260601, 2025.

Oscar Higgott and Craig Gidney. Sparse blossom: correcting a million errors per core second with minimum-weight matching. *Quantum*, 9:1600, 2025.

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.

A Yu Kitaev. Quantum error correction with imperfect gates. In *Quantum communication, computing, and measurement*, pp. 181–188. 1997.

Emanuel Knill, Raymond Laflamme, and Wojciech H. Zurek. Resilient quantum computation. *Science*, 279:342–345, 1998.

Yuhao Liu, Shuohao Ping, Junyu Zhou, Ethan Decker, Justin Kalloor, Mathias Weiden, Kean Chen, Yunong Shi, Ali Javadi-Abhari, Costin Iancu, et al. Alphasyndrome: Tackling the syndrome measurement circuit scheduling problem for qec codes. In *Proceedings of the 31st ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 77–93, 2026.

Seth Lloyd. Universal quantum simulators. *Science*, 273(5278):1073–1078, 1996.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Matt McEwen, Dvir Kafri, Z Chen, Juan Atalaya, KJ Satzinger, Chris Quintana, Paul Victor Klimov, Daniel Sank, C Gidney, AG Fowler, et al. Removing leakage-induced correlated errors in superconducting quantum error correction. *Nature communications*, 12(1):1761, 2021.

Matt McEwen, Lara Faoro, Kunal Arya, Andrew Dunsworth, Trent Huang, Seon Kim, Brian Burkett, Austin Fowler, Frank Arute, Joseph C Bardin, et al. Resolving catastrophic error bursts from cosmic rays in large arrays of superconducting qubits. *Nature Physics*, 18(1):107–111, 2022.

Matt McEwen, Dave Bacon, and Craig Gidney. Relaxing hardware requirements for surface code circuits using time-dynamics. *Quantum*, 7:1172, 2023.

Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5(1):4213, 2014.

Peter W Shor. Scheme for reducing decoherence in quantum computer memory. *Physical review A*, 52(4):R2493, 1995.

Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303–332, 1999.

Volodymyr Sivak, Michael Newman, and Paul Klimov. Optimization of decoder priors for accurate quantum error correction. *Physical Review Letters*, 133(15):150603, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Joshua Viszlai, Satvik Maurya, Swamit Tannu, Margaret Martonosi, and Frederic T Chong. Prophunt: Automated optimization of quantum syndrome measurement circuits. In *Proceedings of the 31st ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 1476–1491, 2026.

Jaap J Wesdorp, Eric Hyyppä, Joona Andersson, Janos Adam, Rohit Beriwal, Ville Bergholm, Saga Dahl, Simone Diego Fasciati, Alejandro Gomez Friero, Zheming Gao, et al. Mitigating crosstalk errors for simultaneous single-qubit gates on a superconducting quantum processor. *arXiv preprint arXiv:2603.11018*, 2026.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Keyi Yin, Hezi Zhang, Xiang Fang, Yunong Shi, Travis S Humble, Ang Li, and Yufei Ding. Qeccsynth: A layout synthesizer for quantum error correction codes on sparse architectures. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pp. 876–890, 2025.

Kai Zhang, Dingchao Gao, Zhaohui Yang, Runshi Zhou, Fangming Liu, Zhengfeng Ji, and Jianxin Chen. Optimal compilation of syndrome extraction circuits for general quantum ldpc codes. In *2026 Design, Automation & Test in Europe Conference (DATE)*, pp. 1–7. IEEE, 2026.

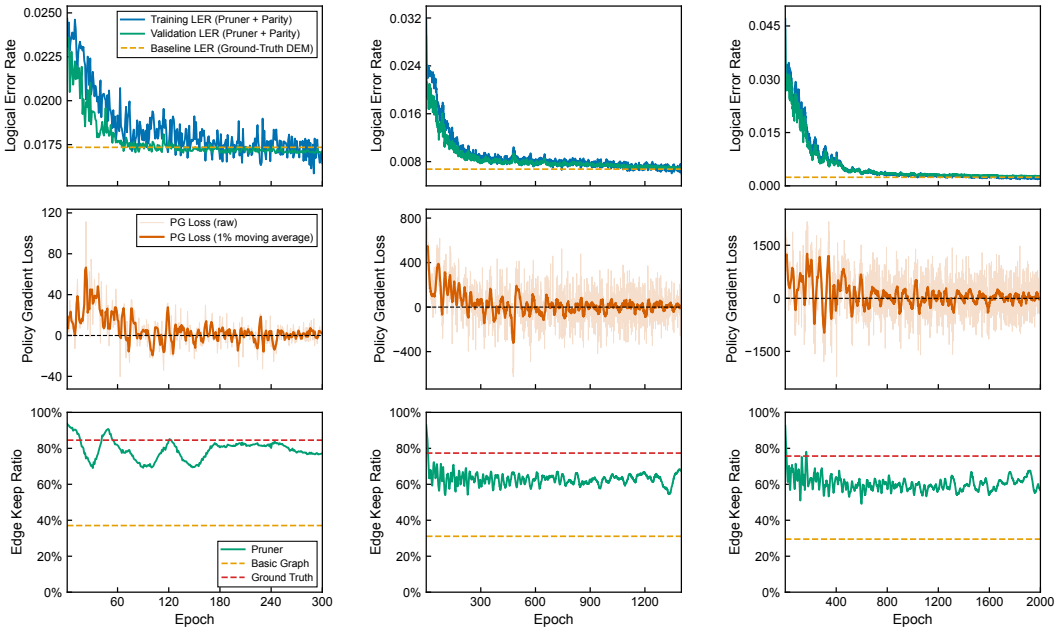
Peng Zhao, Kehuan Linghu, Zhiyuan Li, Peng Xu, Ruixia Wang, Guangming Xue, Yirong Jin, and Haifeng Yu. Quantum crosstalk analysis for simultaneous gate operations on superconducting qubits. *PRX quantum*, 3(2):020301, 2022.

A Appendix

A.1 Decoding Performance under the SI1000 Noise Model

To evaluate topology optimization under a known noise model, we extend the DEM compiled from the standard surface-code circuit under the SI1000 noise model (Bausch et al., 2024; Google Quantum AI and Collaborators, 2025) and train **PrunerNet** with the Parity-based weight estimator. We evaluate $d = 3, 5, 7$ with $r = 3$ syndrome-extraction rounds, comparing the learned DEM with the circuit-compiled reference in terms of logical error rate and hyperedge count.

Figure 8 shows the logical error rate, policy-gradient loss, and retained-edge ratio during training. Although convergence takes longer as d increases, **PrunerNet** matches the logical accuracy of the circuit-compiled DEM while retaining 10%–20% fewer edges. These results show that matching the decoding performance of the circuit-compiled DEM does not require reproducing its exact topology.



(a) $d = 3$. (b) $d = 5$. (c) $d = 7$.

Figure 8: Training dynamics on the SI1000 benchmark for code distances $d = 3, 5, 7$ and physical error rate $p = 0.003$. The policy-gradient loss is smoothed over a window of 1% of the training epochs.

A.2 Training Details

Entropy annealing. Let λ_0 denote the entropy coefficient specified in the training configuration and N_l the number of learnable edges. Because the policy entropy aggregates contributions from all learnable Bernoulli variables, the effective initial coefficient is normalized as

$$\lambda_0^{\text{eff}} = \frac{\lambda_0}{\sqrt{N_l}}, \tag{11}$$

which keeps the entropy gradient on a comparable relative scale to the policy-gradient term across code distances and graph-extension configurations. We then anneal this coefficient to prevent the Bernoulli edge-selection policy from collapsing prematurely while still allowing it to converge towards a near-deterministic policy. Let $\rho_t = t/T \in [0, 1]$ denote the normalized training progress at epoch t over T epochs. The coefficient follows the piecewise schedule

$$\lambda_t = \lambda_0^{\text{eff}} \begin{cases} 1, & 0 \le \rho_t < 0.3, \\ 1 - 0.99\dfrac{\rho_t - 0.3}{0.7}, & 0.3 \le \rho_t \le 1. \end{cases} \tag{12}$$

Thus, the full effective entropy regularization is retained during the first 30% of training, after which it is linearly reduced to $0.01\lambda_0^{\text{eff}}$. This schedule encourages broad structural exploration early in training and progressively shifts the pruning policy towards exploitation.

Initialization and optimization. The scoring head is initialized so that each learnable edge has an initial retention probability close to one, allowing training to start from a nearly intact graph and progressively prune redundant edges. We use AdamW (Loshchilov & Hutter, 2017) with a cosine learning-rate schedule and clip the gradient norm to 1.0. After each epoch, the current DEM is evaluated on a fixed held-out validation set, and the checkpoint with the lowest validation LER is restored after training. This checkpoint is used to construct the deterministic final DEM for all subsequent evaluations.

Experimental configurations. Table 2 collects the shared training defaults and experiment-specific settings for the SI1000 noise model, correlated-error circuits, and random-ordered circuits. Here, B counts candidate graph masks per policy update, whereas N_{reward} counts training shots used to evaluate each candidate's LER; neither is the size of the fixed validation set. Weight-only baselines retain the nominal topology and perform a single weight-estimation pass, so pruner architecture, epoch budgets, and policy hyperparameters apply only to the topology-learning runs. Ablation-specific variations are described in Section 6.

A.3 DEM Graph Analysis

To better understand what the pruner learns, we analyze the structure of the learned DEM along three complementary aspects: the edge retention pattern (Appendix A.3.1), the weight distribution of the selected edges (Appendix A.3.2), and the influence of the pruner capacity (Appendix A.3.3). We focus on the SI1000 noise model with $d = 5$ and $r = 3$, detailed in Appendix A.1.

A.3.1 Edge Retention Pattern

We compare DEM hyperedges by their complete detector supports, i.e., the sets of detectors jointly flipped by the corresponding error mechanisms. For visualization, we represent higher-order hyperedges using one possible decomposition into components involving at most two detectors; only two-detector components are drawn. This decomposition is not unique, and all hyperedge counts and comparisons still use complete supports.

Overall sparsity. At $d = 5$, the circuit-compiled ground-truth DEM contains 861 hyperedges. Graph extension adds 252 candidates, bringing the total edge number to 1113. On this common candidate set, the unchanged ground-truth DEM contains 861 hyperedges (77% of total 1113 edges), whereas the pruner retains 738 (66%). A similar keep ratio of 65% is observed at $d = 7$. Thus, the learned DEM does not simply reproduce the circuit-compiled topology: it uses fewer hyperedges while attaining comparable decoding accuracy on the SI1000 noise model.

Structured rewriting. Figure 9 places the candidate additions and learned changes alongside the full circuit-compiled DEM. All 173 removed ground-truth hyperedges span multiple rounds and have orders between two and four. The solid segments visible in Figure 9c do not contradict this observation: they are visual components of higher-order cross-round hyperedges rather than independent intra-round mechanisms. In contrast, all 50 hyperedges unique to the learned DEM are local order-2 candidates introduced by graph extension, including both intra-round and inter-round connections. The pruner thus preserves the original intra-round mechanisms while selectively replacing part of the cross-round structure with local pairwise candidates.

Table 2: Experimental settings and model hyperparameters.

Parameter	Shared across experiments		
Embedding dimension	64		
Attention heads, H	4		
Transformer layers, L	2		
Dropout probability	0.05		
Initial retention ratio[a]	90%		
Sampling temperature, τ	1.0		
Optimizer	AdamW		
Weight decay	10^{-5}		
Learning rate	$10^{-3} \to 3 \times 10^{-4}$ (cosine decay)		
Gradient-norm clipping	1.0		
EMA baseline decay	0.9		
Entropy schedule	Constant, then linear decay[b]		
Physical noise parameter, p	0.003		
Code distances, d	3, 5, 7		
Syndrome-extraction rounds, r	3		
Training shots	200,000		
Extension orders, k	{2}		
Parameter	**SI1000 noise model**	**Random ordering**	**Correlated errors**
Candidate masks per update, B	(8, 16, 16)	(8, 16, 16)	16
Shots per mask, N_{reward}	50,000	50,000	25,000
Training epochs, T	(300, 1400, 2000)	(200, 1400, 1800)	(1000, 1500, 2000)
Base entropy coefficient, λ_0	(0.1, 1, 1)	(0.1, 1, 1)	1
Manhattan threshold, D	2	3	2
Crosstalk noise probability, q	—	—	6×10^{-4}
Weight estimators	Parity	Correlation; Parity	Correlation; Parity
Decoders[c]	Correlated matching	PyMatching; correlated matching	PyMatching; correlated matching

Tuples list values for $d = 3, 5, 7$, respectively; single values apply to all three distances. Each training epoch comprises one policy update.

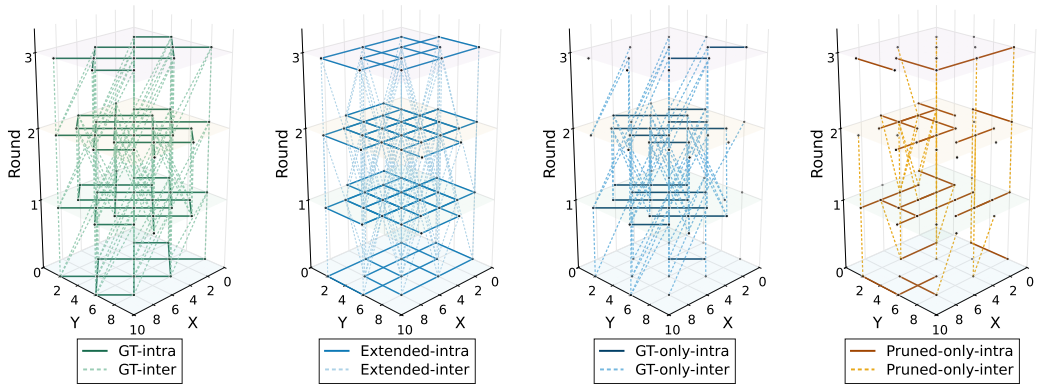
[a] The initial retention probability applies to learnable edges; basic edges are always retained.
[b] The effective initial entropy coefficient $\lambda_0^{\text{eff}} = \lambda_0/\sqrt{N_l}$, where N_l is the number of learnable edges.
[c] The decoder lists apply to both training and evaluation.

This preference for removing cross-round mechanisms has two possible explanations. Some may be redundant once the remaining structure is reweighted, since the learned DEM attains comparable decoding accuracy. Alternatively, high-order cross-round mechanisms may be harder to credit under noisy LER supervision and could be pruned prematurely. Distinguishing these effects requires further study and motivates edge-type-aware initialization, for example assigning stronger initial retention probabilities to cross-round hyperedges.

A.3.2 Weight Distribution of Selected Edges

Redundancy among weak edges. Figure 10 compares the estimated weight distributions of the Parity-only DEM and the learned DEMs. Relative to the Parity-only DEM, the learned ones remove predominantly weak edges: in the $[2\times10^{-4}, 5\times10^{-4})$ and $[5\times10^{-4}, 10^{-3})$ bins, counts drop from 220 and 202 to 110–177 and 97–156, respectively, while the distributions remain similar above 2×10^{-3}. This confirms that the extended DEM contains substantial redundancy among weak correlations, which the pruner identifies and removes.

Nevertheless, weight magnitude alone does not determine retention: the learned DEMs still retain 290–493 edges with weights below 10^{-3}. Under the LER-guided objective, an edge is retained as long as it contributes to the decoding decision, so that weak but structurally important edges are



(a) Ground-truth hyperedges (b) Added candidate hyperedges (c) Removed ground-truth hyperedges (d) Retained candidate hyperedges

Figure 9: Decoding-graph topology for the surface code with $d = 5$ and $r = 3$: (a) the full circuit-compiled DEM, (b) 252 candidate hyperedges introduced by graph extension, (c) 173 ground-truth hyperedges removed by the pruner, and (d) 50 candidate hyperedges retained only in the learned DEM. For visualization, high-order hyperedges are shown in graph-like edges according to Stim's decomposition; solid and dashed lines denote intra- and inter-round components, respectively.

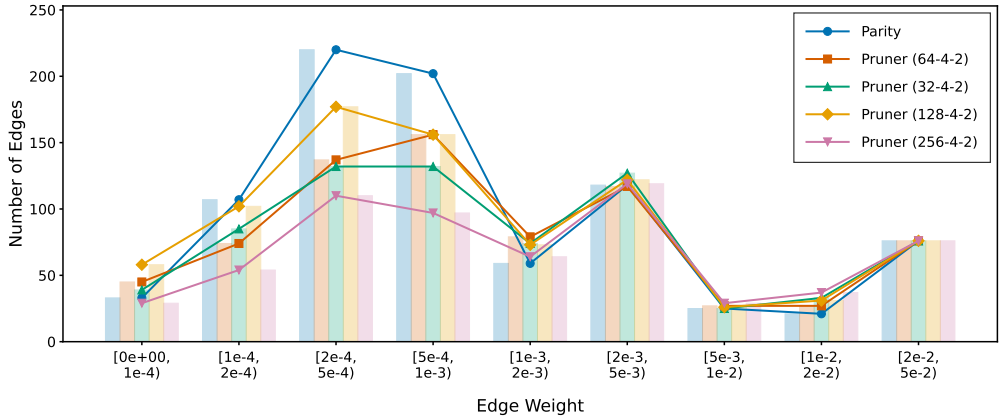


Figure 10: Distribution of the estimated edge weights for the Parity-only DEM and the DEMs learned by pruners with different model dimensions ($d = 5$, $r = 3$). The learned DEMs concentrate their removals in the low-weight regime, while all distributions nearly coincide above 2×10^{-3}.

preserved. The learned selection is therefore distinct from a simple weight threshold, which would discard exactly these edges.

A.3.3 Model Size Observation

To examine how model capacity affects pruning, we compare the four configurations in Figure 10. The final DEMs contain 723, 738, 821, and 615 edges for 32-4-2, 64-4-2, 128-4-2, and 256-4-2, respectively, compared with 861 edges in the Parity-only DEM. Thus, increasing the model dimension does not simply produce a progressively sparser graph: 128-4-2 retains more edges than either smaller model, while 256-4-2 retains the fewest. This difference is concentrated among weak edges. In the $[2 \times 10^{-4}, 5 \times 10^{-4})$ bin, the four learned DEMs contain 132, 137, 177, and 110 edges, respectively. The 128-4-2 distribution is therefore closer to Parity-only in this range, whereas 256-4-2 removes substantially more of these edges.

These differences in sparsity do not translate directly into decoding performance. Despite retaining fewer edges than 128-4-2, the 64-4-2 configuration achieves the best balance of final LER and

training stability in Section 6; the much sparser 256-4-2 model becomes unstable during training. A weight histogram records how many edges fall into each interval, but not which detectors they connect or how the retained mechanisms interact during decoding. Consequently, a distribution closer to Parity-only need not yield a better decoder, nor does more aggressive removal of low-weight edges necessarily improve LER. Model capacity must therefore be evaluated through both the selected graph and its decoding performance, rather than through edge counts or marginal weight distributions alone.