SYNQBENCH: HARDWARE-GROUNDED SYNTHETIC BENCHMARKS FOR NEURAL QUANTUM ERROR CORRECTION DECODERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural decoders for quantum error correction (QEC) rely heavily on synthetic data for both training and evaluation. However, the field currently lacks standardized, hardware-grounded benchmarks. Consequently, existing studies commonly depend on custom datasets with simplified noise assumptions, compromising the data quality and limiting the hardware relevance of resulting conclusions. In this work, we introduce SynQBench, a hardware-grounded synthetic benchmark for neural QEC decoders. SynQBench integrates device-dependent physical profiles and empirical calibrations into a circuit-aware noise model, enabling the scalable generation of faithful, labeled syndrome data. We validate the statistical fidelity of SynQBench against Google's Willow processor measurements, achieving a $6.9\times$ reduction in the mean absolute error of detector-event rates relative to the widely used SI1000 noise model, alongside substantially closer agreement with hardware spatial and temporal correlations. Furthermore, we demonstrate dual downstream value of SynQBench across the decoder development lifecycle: training on SynQBench reduces hardware logical error rates by 30.4 to 42.0% across four decoder architectures, while evaluating on SynQBench narrows the sim-to-real gap in logical error rate by $14.0\times$ across 16 trained decoders. Together, these results position SynQBench as a standardized, hardware-grounded foundation for reproducible neural QEC decoder development and assessment.

1 INTRODUCTION

Quantum error correction (QEC) forms the foundation of fault-tolerant quantum computing (FTQC) by protecting logical information from physical noise (Gottesman, 1998; Terhal, 2015). By encoding logical information across multiple physical qubits and repeatedly measuring stabilizer checks, QEC detects errors without directly measuring the encoded quantum state. Recent demonstrations of logical error suppression on superconducting processors underscore its growing practical importance (Google Quantum AI, 2023; Google Quantum AI and Collaborators, 2025). Realizing this protection requires classical decoders to interpret noisy syndrome histories and infer a recovery that preserves the logical information (Dennis et al., 2002), making decoding accuracy central to the reliability of logical computation.

Within this landscape, neural decoders have emerged as a highly promising approach. By learning error correlations directly from data, these models flexibly capture complex, high-order noise patterns that conventional decoding approximations struggle to resolve, yielding substantial accuracy gains (Bausch et al., 2024; Hu et al., 2025; Gu et al., 2026). Consequently, the effectiveness of neural decoders relies heavily on the statistical fidelity and diversity of the data used in their development. Because experimental datasets currently remain constrained in volume and coverage across code configurations, synthetic noise data becomes indispensable for both model training and systematic evaluation (Google Quantum AI, 2023; Bausch et al., 2024). This reliance endows synthetic data with two distinct but deeply interconnected roles: it dictates the physical error distributions and correlations exposed to the model during training, while also defining the conditions to assess decoder performance during evaluation.

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.

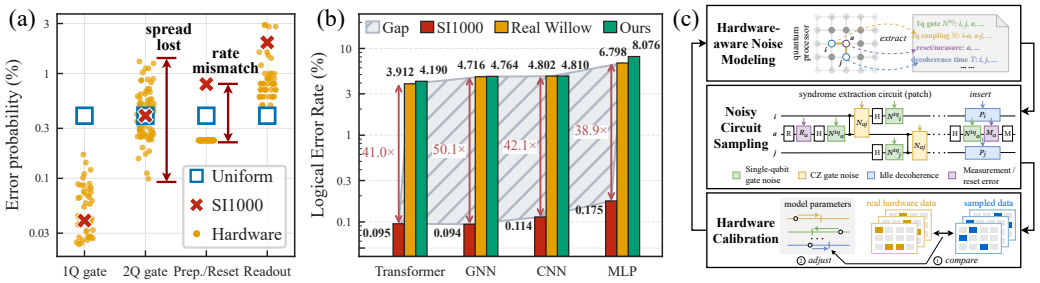


Figure 1: Motivation and our solution. **(a)** Simplified noise models misrepresent hardware error rates and local variability. **(b)** Sim-to-real gap for fixed pretrained decoders evaluated on synthetic SI1000, real Willow and SynQBench (log scale). Shading highlights the existing sim-to-real gap, while ours matches the hardware much more closely. **(c)** Overview of the SynQBench pipeline.

Despite this foundational role of synthetic data, the development of shared, hardware-grounded benchmarks has lagged significantly behind the rapid pace of architectural innovation in neural decoders (Maurya et al., 2025). Current literature therefore commonly evaluates models under custom synthetic regimes operating on simplified noise assumptions (Meinerz et al., 2022), including the widely used SI1000 model (Gidney et al., 2021). By enforcing uniform rates or fixed operational ratios, these models overlook local hardware variability and misrepresent operation-specific error probabilities (Figure 1(a)), compromising training data quality by omitting realistic error diversity. This discrepancy can also produce a substantial sim-to-real gap during evaluation, with decoders exhibiting much higher logical error rates on experimental data than on idealistic synthetic benchmarks (Figure 1(b)), diminishing the real-world validity of resulting conclusions. These limitations underscore a pressing need for a standardized benchmark grounded in empirical hardware evidence to enable realistic decoder training, rigorous evaluation, and direct cross-study comparison.

In this work, we introduce **SynQBench**, a hardware-grounded synthetic benchmark for neural QEC decoders. By combining hardware-aware noise modeling, noisy circuit sampling, and calibration against experimental syndrome statistics, SynQBench systematically generates labeled syndrome data that capture spatial noise heterogeneity and the temporal structure of syndrome-extraction circuits (Figure 1(c)). We validate SynQBench by demonstrating its statistical fidelity to physical hardware alongside its practical utility in downstream neural decoder training and evaluation. Together, these studies establish a rigorous empirical foundation for leveraging SynQBench in neural decoder development and assessment. Particularly, this paper makes the following contributions:

- **Systematic Hardware-Grounded Data Generation.** We present a unified generation pipeline that integrates a circuit-aware noise model calibrated against experimental syndrome statistics. This workflow generates standardized, labeled datasets, establishing a common foundation for reproducible neural QEC decoder training and benchmarking.
- **Experimentally Validated Noise Fidelity.** We validate SynQBench against Willow hardware measurements, demonstrating that it reduces detector-event-rate mean absolute error by $6.9\times$ over SI1000 while substantially tightening agreement in spatial and temporal correlations, yielding improved fidelity to both local noise heterogeneity and hardware syndrome dependencies.
- **Downstream Value in Training and Evaluation.** We demonstrate that SynQBench provides dual benefits for neural decoding: training decoders on SynQBench significantly reduces logical error rates on the Willow processor by 30.4 to 42.0% relative to SI1000, while evaluating on SynQBench narrows the sim-to-real domain gap by $14.0\times$, yielding performance estimates that closely mirror physical device behavior.

2 Background

2.1 Quantum Error Correction Decoding

An $[[n, k, d]]$ stabilizer code embeds k logical qubits into n physical qubits, where code distance d is the minimum weight of an undetectable logical Pauli operator (Gottesman, 1997). Commuting par-

ity checks, called stabilizers, extract error syndromes without directly measuring and collapsing the encoded logical state (Ristè et al., 2015). For example, a surface code of distance d has parameters $[[d^2, 1, d]]$, encoding one logical qubit into d^2 physical data qubits.

A quantum memory experiment initializes a logical state, executes repeated stabilizer measurement rounds, and terminates with a data-qubit readout. The resulting detection-event history is a binary vector $\mathbf{s} \in \{0,1\}^D$, where D is the number of detector events and each entry indicates a parity change between consecutive rounds or a boundary event relative to the noiseless expectations. Each execution, or shot, is paired with a target $\mathbf{y} \in \{0,1\}^k$ whose entries record flips of the k measured logical observables.

A decoder predicts the corresponding logical correction $\hat{\mathbf{y}} = f(\mathbf{s})$. We count a shot as a logical failure if the prediction disagrees with the target in at least one logical observable. Over N evaluation shots, the shot-level logical error rate (LER) is therefore defined as

$$\widehat{\mathrm{LER}}(f) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}[f(\mathbf{s}_i) \neq \mathbf{y}_i], \tag{1}$$

2.2 Noise Modeling and Benchmark Data

Noise modeling provides a circuit-level description of physical errors during QEC by specifying how errors occur during gates, initialization, measurement, and idle operations (Gidney et al., 2021). Neural decoder studies often adopt simplified parameterizations in which these error processes are governed by a single physical error rate, either uniformly or through fixed ratios across operation types (Varsamopoulos et al., 2020; Bausch et al., 2024). SI1000, for example, assigns operation-dependent error rates inspired by superconducting hardware (Gidney et al., 2021). While such models offer controlled environments for decoder development, their stylized assumptions fail to reproduce the statistics of a particular device, creating potential discrepancies between simulated and experimental noise distributions (Manor et al., 2026). These circuit-level models can be sampled efficiently with tools such as Stim to generate the large syndrome datasets required for neural decoder training and evaluation (Gidney, 2021).

Experimental data from superconducting processors such as Sycamore and Willow provide direct evidence of hardware noise, but their limited volume makes them insufficient as the sole source of large-scale training data (Google Quantum AI, 2023; Google Quantum AI and Collaborators, 2025). Prior work therefore combines synthetic data with experimental observations, either through hardware-informed simulation or fine-tuning on device data (Bausch et al., 2024; Yan et al., 2026). These developments motivate extending the use of experimental data beyond individual decoder studies: calibrating a reusable circuit-level noise model from finite hardware observations enables the generation of substantially larger synthetic datasets, which provide a common, hardware-grounded distribution for neural decoder training and evaluation.

3 SynQBench: construction and benchmark protocol

3.1 Hardware-Aware Circuit-Level Noise Model

Generating realistic synthetic syndrome data requires modeling noise at each circuit location rather than matching only an average physical error rate. Hardware errors differ across qubits and couplers, and the noise at each circuit location depends on the operation and its duration. Homogeneous circuit-level models replace these differences with one or a few global parameters, losing device-specific patterns that affect detector statistics. Following the PAEMS framework (He et al., 2026), we constructed the noise channel for each scheduled circuit location from the hardware profile of a real quantum processor. The channel depends on the operation, the qubits or coupler involved, the operation duration, and the corresponding hardware parameters. Together, these fine-grained noise channels form our hardware-aware circuit-level noise model, which captures spatial variation and schedule-dependent noise.

To parameterize these location-specific channels, we use three groups of hardware information. Qubit-level parameters include coherence times, single-qubit gate errors and durations, and readout durations. Coupler-level parameters describe the error rates and durations of two-qubit gates on each

physical connection. The hardware profile also provides state-preparation, measurement, and reset error probabilities for each physical qubit. Together with the scheduled circuit, these parameters determine the noise assigned to each gate, idle, measurement, and reset location. Complete parameter definitions are provided in Appendix A.1.

At each circuit location, we convert the relevant hardware parameters into a stochastic Pauli channel. For gate and idle locations, the coherence times and operation duration define a duration-dependent Pauli channel $\mathcal{P}_\ell(\tau_\ell)$. At a gate location, we combine this channel with the ideal operation $\mathcal{U}_\ell$ and a residual depolarizing channel $\mathcal{D}_\ell(r_\ell)$:

$$\mathcal{E}_\ell = \mathcal{D}_\ell(r_\ell) \circ \mathcal{P}_\ell(\tau_\ell) \circ \mathcal{U}_\ell. \tag{2}$$

The residual error is chosen so that the combined channel matches the gate fidelity reported in the hardware profile without counting coherence loss twice. Preparation, measurement, and reset errors are applied as Pauli flips with their qubit-specific error probabilities. The complete channel definitions are provided in Appendix A.2.

The hardware profile therefore provides a physically grounded, device-informed initialization of the noise model, which we denote by **SynQBench-base**. To account for differences between profile-based predictions and observed device behavior, we next refine a selected subset of noise parameters using hardware syndrome statistics.

3.2 Statistics-Guided Noise-Parameter Calibration

The SynQBench-base model provides a physically grounded description of device noise. However, device conditions evolve over time, and the influence of hardware effects such as leakage and crosstalk can depend on the operating context. We therefore calibrate a small subset of noise parameters using hardware syndrome statistics, while keeping the model structure and all other hardware parameters fixed. We denote the resulting calibrated model by **SynQBench**. Calibration involves three components: the target statistics, calibrated parameter groups, and the update rule.

The target statistics. To compare synthetic and hardware syndrome behavior, we use four complementary families of detector statistics computed over the bulk rounds. These include the detector-event rate r_s of each stabilizer, the temporal correlation τ_s of the same stabilizer across consecutive rounds, the spatial correlation σ_{ab} between neighboring stabilizers in the same round, and the directed spacetime correlation $\kappa_{a\to b}$ between neighboring stabilizers in consecutive rounds. We normalize each synthetic–hardware difference by its estimated hardware uncertainty and form two robust objectives: a cross-stabilizer objective over (σ, κ) and a single-stabilizer objective over (τ, r). Complete definitions of the statistics and objectives are provided in Appendix B.1.

The calibrated parameter groups. We calibrate three groups of noise parameters according to which target statistics they primarily affect:

$$\theta_{\mathrm{CZ}} \to (\sigma, \kappa), \qquad \theta_{\mathrm{D}} \to \sigma, \qquad \theta_{\mathrm{T}} \to (\tau, r). \tag{3}$$

Here, θ_{CZ}, θ_{D}, and θ_{T} scale selected CZ-gate, data-idle, and ancilla measurement and reset error parameters, respectively. The spacetime statistic κ distinguishes CZ-gate effects from data-idle effects, while (τ, r) constrains measurement and reset errors. Further details are provided in Appendix B.2.

The update rule. Guided by these statistical roles, we calibrate the parameter groups in alternating cross-stabilizer and single-stabilizer blocks. Within each block, we rescale only a subset of parameters with the largest estimated effects on the current objective. This keeps the local optimization low-dimensional and focuses the simulation budget on the dominant synthetic–hardware mismatch. Cross-stabilizer blocks update θ_{CZ} and θ_{D} against the (σ, κ) objective, while single-stabilizer blocks update θ_{T} against the (τ, r) objective. The procedure repeatedly performs a sequence of cross-stabilizer blocks followed by a single-stabilizer block. Each block uses a robust Gauss–Newton step on the calibration split to propose a parameter update. To control overfitting, we reevaluate each proposal on a disjoint acceptance split and retain it only if the improvement persists. Complete optimization and acceptance rules are provided in Appendix B.3.

3.3 Dataset Generation Workflow

After calibration, SynQBench integrates the calibrated hardware-aware noise model into a scheduled quantum error-correction circuit represented in Stim (Gidney, 2021). At each gate, idle, measure-

ment, and reset location, we insert the corresponding location-specific Pauli channel to construct a noisy circuit. We then use Stim to sample this circuit efficiently, recording the detector events and logical-observable outcome of each shot. The collected samples form a reproducible benchmark for training and evaluating different decoders under the same circuit and noise conditions.

4 Statistical Fidelity to Hardware

4.1 Data and Evaluation Metrics

In this section, we evaluate how closely the synthetic syndrome data reproduce the statistical behavior observed on hardware. Our analysis proceeds at three progressively more comprehensive levels. At Level I, we examine detector-event rates to assess first-order agreement in the overall error scale and location-dependent heterogeneity. At Level II, we evaluate spatial, temporal, and directed space-time correlations to determine whether the synthetic data capture structured second-order dependencies between detector events. At Level III, we extend the evaluation beyond these structured statistics to assess agreement with hardware across the full detector correlation matrix. At each level, we use two comparisons with distinct purposes. Comparing Uniform and SI1000 against SynQBench-base isolates the value of hardware-aware parameterization, while comparing SynQBench-base against SynQBench quantifies the additional value of statistics-guided noise-parameter calibration.

For the hardware reference, we use the publicly released Willow dataset (Google Quantum AI and Collaborators, 2024), focusing on the distance-5, 10-round, Z-basis surface-code memory experiment. All reported comparisons in this section use a held-out set of 10,000 hardware shots and the same number of independently generated shots per synthetic model; the complete data-partition and evaluation protocol is provided in Appendix C. The SI1000 baseline is the noisy SI1000 model released by Google. For the Uniform baseline, we set a single global noise parameter to $p = 0.001$, matching the global parameter of SI1000. This ensures that differences between the two baselines arise from how noise is assigned across circuit operations rather than from the nominal value of p.

4.2 Multi-Level Statistical Agreement

Level I: Detector-event rate scale and heterogeneity. Detector-event rates (DERs) provide the first level of validation because they measure how often errors become visible at each circuit location. We assess absolute agreement using the mean absolute error (MAE) and pattern agreement using the Pearson correlation coefficient r across stabilizer–round cells; precise definitions are provided in Appendix C.1. Figure 2 examines these properties from the stabilizer-level profile to the full set of round-resolved rates.

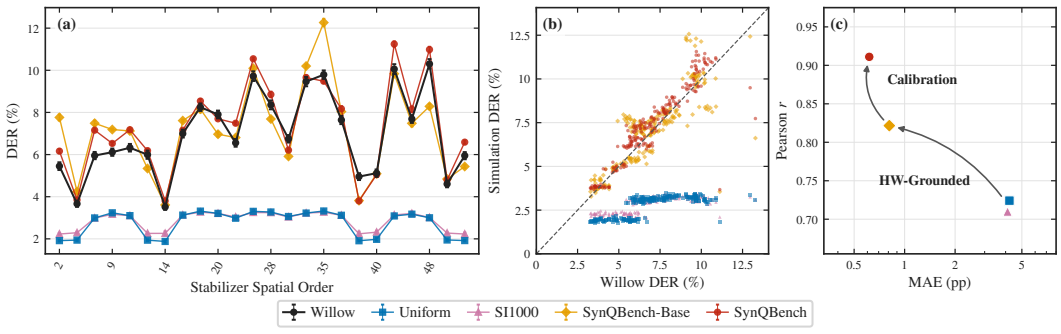


Figure 2: Detector-event rate agreement between synthetic models and Willow. **(a)** Detector-event rate averaged over bulk rounds for each stabilizer, ordered by physical location. **(b)** Per-stabilizer, per-round synthetic rates plotted against their hardware counterparts; the diagonal denotes perfect agreement. **(c)** Summary in the MAE–Pearson fidelity plane, where lower MAE and higher r indicate better agreement; HW abbreviates hardware.

Figure 2(a) shows that Uniform and SI1000 not only underestimate the overall DER scale but also compress the marked peaks and valleys observed across stabilizers. In contrast, SynQBench-base already tracks most of the hardware profile, showing that location-specific device parameters recover

much of this heterogeneity. Panels (b) and (c) extend the comparison to individual stabilizer–round cells: hardware-aware initialization substantially improves both MAE and Pearson r, and calibration improves both metrics further. Because uniformly rescaling all DERs leaves Pearson r unchanged, the additional upward movement in panel (c) shows that calibration refines the relative DER pattern rather than merely shifting the global noise scale.

Level II: Structured correlation agreement. We evaluate the spatial, temporal, and directed space-time correlation families defined in Section 3.2 using the mean absolute residual (MAR) between synthetic and hardware features. The metric is specified in Appendix C.2. Table 1 reports the MARs for all models and the reduction factors associated with the two stages of SynQBench construction.

Table 1: MARs between synthetic and Willow correlation features. Lower MAR indicates closer agreement. HW gain (hardware-aware gain) is measured from Uniform to SynQBench-base, and Calib. gain (calibration gain) from SynQBench-base to SynQBench.

	Uniform	SI1000	SynQBench-base	**SynQBench**	HW gain	Calib. gain
Spatial	1.10×10^{-2}	1.15×10^{-2}	4.60×10^{-3}	$\mathbf{1.59 \times 10^{-3}}$	$2.4\times$	$2.9\times$
Temporal	7.56×10^{-3}	4.29×10^{-3}	5.80×10^{-3}	$\mathbf{6.57 \times 10^{-4}}$	$1.3\times$	$8.8\times$
Spacetime	1.93×10^{-3}	2.51×10^{-3}	1.25×10^{-3}	$\mathbf{1.10 \times 10^{-3}}$	$1.5\times$	$1.1\times$

Relative to Uniform, SynQBench-base reduces the spatial, temporal, and spacetime MARs by $2.4\times$, $1.3\times$, and $1.5\times$, respectively. The two simplified baselines retain some circuit-induced structure because they use the same scheduled circuit, but their shared or operation-level noise prescriptions do not reproduce its location-dependent strength. Assigning measured parameters to individual qubits, couplers, and operations therefore recovers a substantial portion of the correlation structure before any statistics-guided updates. Calibration then provides large additional gains: relative to SynQBench-base, SynQBench reduces the spatial and temporal MARs by $2.9\times$ and $8.8\times$, respectively, while reducing the spacetime MAR by $1.1\times$. We next test whether these family-level improvements extend to the broader detector correlation matrix.

Level III: Matrix-level correlation agreement. We extend the comparison from structured correlation families to individual detector pairs. We apply round-window averaging to the detector correlation matrix. This window-averaged representation preserves the stabilizer and within-window round structure while providing a higher-dimensional consistency check. The windowing procedure, finite-shot masking, and matrix-level comparison metrics are detailed in Appendix C.3.

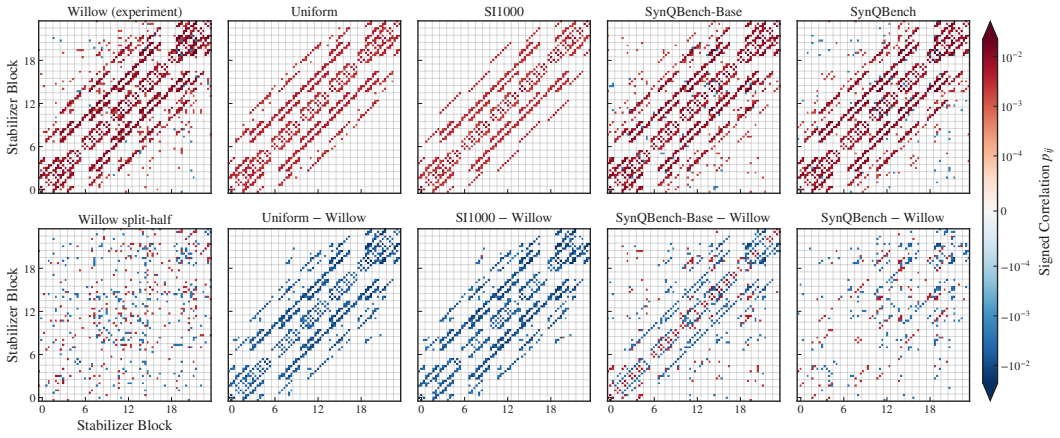


Figure 3: Window-averaged full detector correlation matrices and residuals. The top row compares Willow with the four synthetic models; the bottom row shows a Willow split-half reference and each model's residual relative to Willow. Entries with magnitude below the 3σ finite-shot noise floor are omitted from both the plots and metric calculations.

Figure 3 shows that the Willow matrix has a distinctive arrangement of repeated bands and off-diagonal blocks. These patterns reflect the fine-grained structure of the spatial, temporal, and di-

rected spacetime correlations examined at Level II. Because all four synthetic models use the same error-correction circuit and gate schedule as Willow, the broad band-and-block geometry is visible in every synthetic matrix. Uniform and SI1000 place many bands and blocks in the correct locations, but the correlation magnitudes within these structures are consistently lower than in Willow. SynQBench-base restores substantially more of the correlation magnitude, showing that location-specific qubit, coupler, and operation parameters are needed in addition to the circuit schedule. Calibration further improves both pattern and magnitude across the retained matrix entries: Pearson r increases from 0.807 for SynQBench-base to 0.881 for SynQBench, while the residual RMSE decreases from 1.81×10^{-3} to 1.45×10^{-3}. These simultaneous improvements show that the gains extend to the broader organization of detector correlations.

Summary. Across the three levels, hardware-aware initialization provides the primary recovery of detector-rate heterogeneity and circuit-induced correlation magnitude, while statistics-guided calibration further refines local rates and substantially improves spatial and temporal dependence. The matrix-level results show that these gains extend beyond the family-level objectives to the broader detector correlation structure.

5 Downstream value for training and evaluation

5.1 Experimental Protocol

To assess how statistical fidelity translates into downstream value, we benchmark Transformer, GNN, CNN, and MLP decoders using the four synthetic noise models evaluated in Section 4. Our experimental pipeline structures this assessment across three phases: *Data Generation*, *Training*, and *Evaluation*. For each noise model, we generate 10^7 synthetic shots for training alongside independent validation and test sets of 10^6 shots each. To guarantee controlled comparisons, training and checkpoint-selection protocols are held constant within each architecture family across all noise distributions. Architecture and configuration details are provided in Appendix D.2 and D.3.

During training, early stopping and checkpoint selection rely exclusively on synthetic validation loss (minimum binary cross-entropy). In parallel, we monitor the hardware LER after each epoch using all $50{,}000$ physical Willow shots, which track progress without influencing model selection. In the evaluation phase, selected checkpoints are benchmarked across all four synthetic test sets and the Willow hardware data, alongside a MWPM (PyMatching) baseline configured with a fixed, RL-optimized prior error model. Using shot-level LER (Eq. 1), we assess downstream training utility via target hardware accuracy and evaluation fidelity via the sim-to-real LER gap.

5.2 Training Utility and Evaluation Fidelity

Training on SynQBench improves decoding on real hardware. Figure 4 compares learning trajectories across the four decoder architectures, with performance on Willow shown in the top row. Across all architectures, at checkpoints selected by synthetic validation, training on SynQBench consistently improves hardware performance, reducing Willow LER by 28.1 to 47.1% relative to Uniform noise and 30.4 to 42.0% relative to SI1000. All four decoders trained on SynQBench also outperform the fixed MWPM reference, which yields 4.174% LER on the same hardware shots. Within our benchmark family, calibration further improves performance for Transformer, GNN, and CNN, while MLP performs slightly better with the base variant.

This advantage also holds when training exposure is matched. At the latest epoch shared by all four training distributions within each architecture, SynQBench reduces Willow LER by 27.3 to 41.3% relative to the better of the two standard references, with the number of processed training examples held equal. Its training benefit extends to Uniform and SI1000 test data: decoders trained on SynQBench even outperform those trained directly on the corresponding test distribution in all eight architecture–target comparisons, reducing LER by 8.2 to 36.3% and 12.0 to 30.5%, respectively. Detailed selected-checkpoint results and cross-distribution LERs are provided in Appendix D.4.

Checkpoint selection on SynQBench yields near-best observed performance. Figure 4 enables a retrospective comparison of each selected checkpoint's Willow LER with the minimum observed within the same run. For SynQBench, selected Willow LERs exceed their monitored training minima by merely 0.030, 0.026, 0.032, and 0.046 percentage points for Transformer, GNN, CNN, and

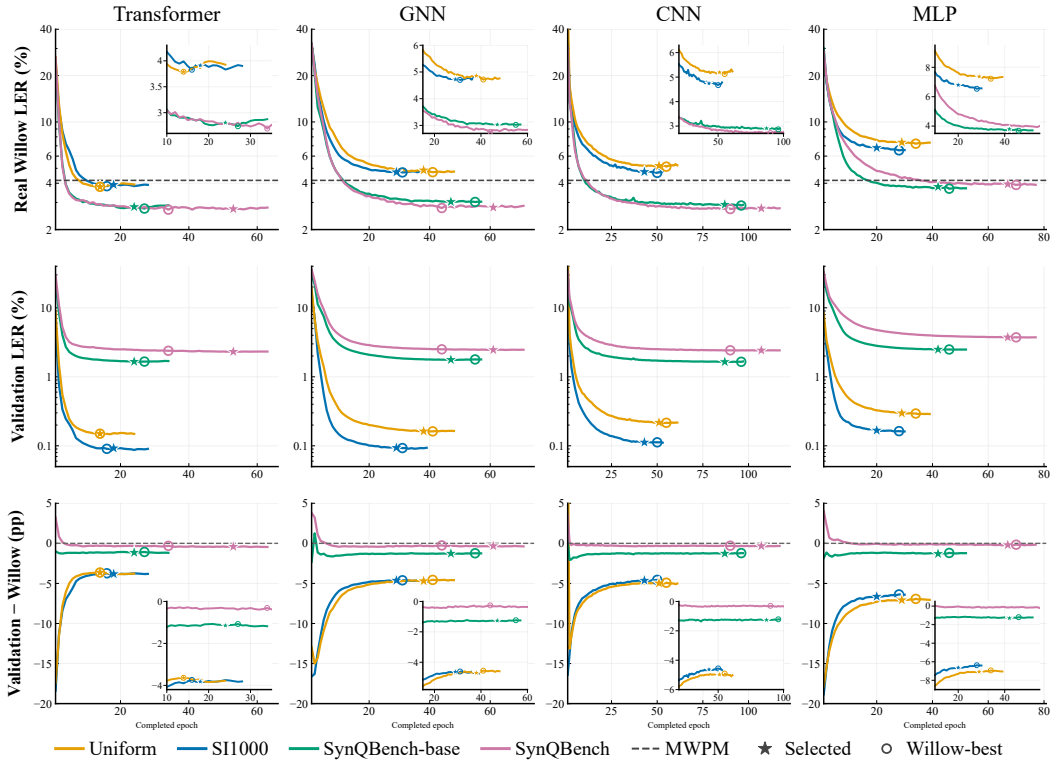
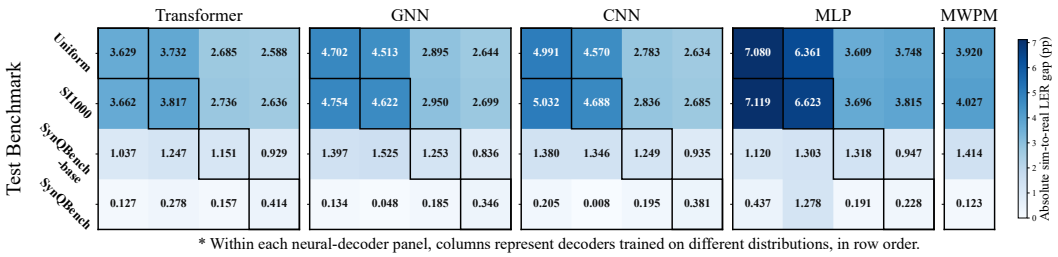


Figure 4: Training trajectories across four decoder architectures and four synthetic noise distributions. Columns correspond to Transformer, GNN, CNN, and MLP. LER axes are logarithmic, and insets provide linear-scale views of later training. The top row reports LER on the common 50,000-shot Willow monitor, where the dashed line denotes the fixed MWPM reference with the Willow RL-optimized prior DEM. The middle row reports LER on independent, one-million-shot validation sets drawn from each training distribution. The bottom row shows validation LER minus Willow LER in percentage points. Negative gaps indicate optimistic synthetic estimates. Stars mark checkpoints selected by minimum synthetic validation loss, while hollow circles mark the lowest observed Willow LER within each run.

MLP, respectively. Though selection occurs 17 to 19 epochs after observed minima for three architectures, all four selected models remain within 1.2% of their optimal LERs.

Validation on SynQBench closely reflects hardware performance during training. The validation trajectories in the middle row of Figure 4 reveal a different ordering from the hardware results. Uniform and SI1000 yield the lowest LERs on their respective validation sets, although their trained decoders perform worse on Willow. The bottom row makes this mismatch explicit by plotting the synthetic validation LER minus Willow LER, with negative values indicating optimistic synthetic estimates. At the selected checkpoints, Uniform and SI1000 underestimate Willow LER by 3.64 to 7.07 percentage points. The corresponding discrepancies decrease to 1.15 to 1.30 percentage points for SynQBench-base and further to 0.20 to 0.41 percentage points for SynQBench. Thus, the higher validation LERs under SynQBench more closely reflect the error rates encountered on hardware, connecting its training benefits with its value for realistic decoder evaluation.

Evaluation on SynQBench faithfully estimates performance across decoders. To examine whether this agreement extends beyond each decoder's training distribution, Figure 5 compares absolute sim-to-real gaps for every selected checkpoint across all four synthetic distributions. SynQBench provides the closest estimate of Willow LER for all 16 neural decoders. Across these models, the mean absolute gap decreases from 3.95 percentage points for Uniform and 4.02 for SI1000 to 1.19 for SynQBench-base and 0.288 for SynQBench. These results correspond to $13.7\times$ and $14.0\times$ reductions relative to Uniform and SI1000, respectively, and a $4.1\times$ reduction relative



* Within each neural-decoder panel, columns represent decoders trained on different distributions, in row order.

Figure 5: Sim-to-real evaluation gaps for different synthetic test benchmarks (rows) across a fixed decoder set. Cells show absolute synthetic–Willow LER differences in percentage points on a shared color scale; lighter rows indicate closer agreement. Within each architecture, columns identify separately trained decoders, with training distributions ordered as the rows. Diagonal cells evaluate decoders on their training distribution; off-diagonal cells evaluate transfer across distributions. MWPM uses a fixed Willow RL-optimized prior DEM.

to SynQBench-base. The MWPM column further demonstrates agreement for a fixed algorithmic decoder, where evaluation on SynQBench reproduces its hardware LER within 0.123 percentage points, compared with gaps of 3.920 for Uniform and 4.027 for SI1000.

Overall, SynQBench provides both effective training data and a common evaluation benchmark, enhancing decoder performance while yielding faithful estimates of real-world logical error rates.

6 Discussion and Conclusion

Applicability. The circuit-level formulation of SynQBench provides a basis for extending the benchmark beyond the Willow experiments studied here. Adaptation to other superconducting processors, such as Sycamore and IBM devices, requires mapping their hardware profiles to the target syndrome-extraction circuits and calibrating against available syndrome data. Platforms with different physical mechanisms, including neutral atoms, also share circuit-level abstractions for gate, preparation, measurement, and idle errors. These common abstractions offer opportunities to reuse components of our modeling framework while incorporating platform-specific effects. Appendix E summarizes representative hardware profiles across these platforms and their data sources. Similarly, although we adopt the surface code because of its extensive theoretical and experimental foundations, the circuit-based workflow can be adapted to other stabilizer-code families with compatible syndrome-extraction circuits.

Limitation. This extensibility rests on a deliberate tradeoff between physical detail and scalable benchmark generation. Our model approximates relaxation and dephasing through effective Pauli channels, while detailed treatment of decay dynamics, leakage, and crosstalk would require translating physical processes below the operation level into suitable circuit-level representations. Developing and validating these representations requires device-specific characterization and can complicate the use of efficient simulation tools for large-scale data generation. We therefore prioritize a shared modeling framework that captures hardware-relevant syndrome statistics while retaining efficient sampling and portability across devices. Our empirical validation currently covers distance-5, 10-round, Z-basis surface-code memory experiments on Willow. Future evaluations across broader hardware and code configurations will help establish how widely the observed benefits extend.

Conclusion. We introduced SynQBench, a hardware-grounded synthetic benchmark that combines device-informed noise modeling, statistics-guided calibration, and scalable data generation. Our experiments demonstrate improved agreement with hardware syndrome statistics, enhanced neural decoder performance on real hardware data, and smaller gaps between synthetic and experimental evaluation. Together, these results position SynQBench as a standardized benchmark that supports effective training and faithful evaluation, providing a common foundation for reproducible neural QEC decoder development.

AI USE STATEMENT

In this work, we used generative AI tools to improve the clarity and readability of the manuscript and to support data analysis and software infrastructure development. The authors reviewed the AI-assisted text and analyses, and verified the AI-assisted infrastructure code through manual review, testing, and comparison with existing implementations. We take full responsibility for the final manuscript and associated research artifacts, including the accuracy and integrity of the reported results and conclusions.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide a supplementary artifact for reproducing the results reported in this work. We also plan to release the SynQBench implementation as open-source software. The benchmark construction and protocol are described in Section 3, while Appendix D provides detailed descriptions of the neural decoder architectures and relevant configurations used in our experiments.

REFERENCES

Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5):052328, 2004. doi: 10.1103/PhysRevA.70.052328.

Johannes Bausch, Andrew W. Senior, Francisco J. H. Heras, Thomas Edlich, Alex Davies, Michael Newman, Cody Jones, Kevin 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 (8040):834–840, November 2024. ISSN 1476-4687. doi: 10.1038/s41586-024-08148-8. URL http://dx.doi.org/10.1038/s41586-024-08148-8.

Cirq Developers. Cirq. GitHub repository, 2021. URL https://github.com/quantumlib/Cirq. cirq-google v1.3.0; Weber calibration snapshot dated 2021-11-03, file weber_2021_11_03_calibration.json.

Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. Topological quantum memory. *Journal of Mathematical Physics*, 43(9):4452–4505, 2002. doi: 10.1063/1.1499754.

Dongxin Gao, Daojin Fan, et al. Establishing a new benchmark in quantum computational advantage with 105-qubit Zuchongzhi 3.0 processor. *Phys. Rev. Lett.*, 134:090601, 2025. doi: 10.1103/PhysRevLett.134.090601.

Michael R. Geller and Zhongyuan Zhou. Efficient error models for fault-tolerant architectures and the pauli twirling approximation. *Physical Review A*, 88(1):012314, 2013. doi: 10.1103/PhysRevA.88.012314.

Craig Gidney. Stim: a fast stabilizer circuit simulator. *Quantum*, 5:497, July 2021. doi: 10.22331/q-2021-07-06-497. URL https://quantum-journal.org/papers/q-2021-07-06-497/.

Craig Gidney, Michael Newman, Austin Fowler, and Michael Broughton. A fault-tolerant honeycomb memory. *Quantum*, 5:605, December 2021. doi: 10.22331/q-2021-12-20-605. URL https://quantum-journal.org/papers/q-2021-12-20-605/.

Google Quantum AI. Suppressing quantum errors by scaling a surface code logical qubit. *Nature*, 614(7949):676–681, February 2023. doi: 10.1038/s41586-022-05434-1. URL https://www.nature.com/articles/s41586-022-05434-1.

Google Quantum AI and Collaborators. Data for "quantum error correction below the surface code threshold", August 2024. URL https://zenodo.org/records/13273331. Dataset.

Google Quantum AI and Collaborators. Quantum error correction below the surface code threshold. *Nature*, 638(8052):920–926, February 2025. doi: 10.1038/s41586-024-08449-y. URL https://www.nature.com/articles/s41586-024-08449-y.

Daniel Gottesman. Stabilizer codes and quantum error correction, 1997. URL https://arxiv.org/abs/quant-ph/9705052.

Daniel Gottesman. Theory of fault-tolerant quantum computation. *Physical Review A*, 57(1):127–137, 1998. doi: 10.1103/PhysRevA.57.127.

Andi Gu, J. Pablo Bonilla Ataides, Mikhail D. Lukin, and Susanne F. Yelin. Scalable neural decoders for practical fault-tolerant quantum computation, 2026. URL https://arxiv.org/abs/2604.08358.

S. He, Y. Cui, B. Liu, K. Guo, and C. Wang. PAEMS: Precise and adaptive error model for superconducting quantum processors, 2026.

Gengyuan Hu, Wanli Ouyang, Chao-Yang Lu, Chen Lin, and Han-Sen Zhong. Efficient and universal neural-network decoder for stabilizer-based quantum error correction, 2025. URL https://arxiv.org/abs/2502.19971.

IBM Quantum. Qiskit IBM Runtime. GitHub repository, 2025. URL https://github.com/Qiskit/qiskit-ibm-runtime. Version 0.43.1; FakeFez calibration snapshot dated 2025-02-26, file props_fez.json.

Shay J. Manor, Leila S. Erhili, and Yassine Jebbouri. A sim-to-real study of surface-code decoder benchmarking, 2026. URL https://arxiv.org/abs/2609.04557.

Satvik Maurya, Joshua Viszlai, Nithin Raveendran, Poulami Das, and Swamit Tannu. decoderbench: Benchmarking decoders for quantum error correction. In *2025 IEEE International Symposium on Workload Characterization (IISWC)*, pp. 286–295, 2025. doi: 10.1109/IISWC66894.2025.00032.

Kai Meinerz, Chae-Yeun Park, and Simon Trebst. Scalable neural decoder for topological surface codes. *Physical Review Letters*, 128:080505, 2022. doi: 10.1103/PhysRevLett.128.080505.

Quantinuum. Quantinuum hardware specifications. GitHub repository, 2025. URL https://github.com/CQCL/quantinuum-hardware-specifications.

A. G. Radnaev, W. C. Chung, D. C. Cole, et al. Universal neutral-atom quantum computer with individual optical addressing and nondestructive readout. *PRX Quantum*, 6:030334, 2025. doi: 10.1103/66s8-jj18.

Ben W. Reichardt, Adam Paetznick, David Aasen, et al. Fault-tolerant quantum computation with a neutral atom processor. arXiv preprint arXiv:2411.11822, 2024. URL https://arxiv.org/abs/2411.11822.

Diego Ristè, Stefano Poletto, M.-Z. Huang, Alessandro Bruno, Visa Vesterinen, O.-P. Saira, and Leonardo DiCarlo. Detecting bit-flip errors in a logical qubit using stabilizer measurements. *Nature Communications*, 6:6983, 2015. doi: 10.1038/ncomms7983.

Barbara M. Terhal. Quantum error correction for quantum memories. *Reviews of Modern Physics*, 87(2):307–346, 2015. doi: 10.1103/RevModPhys.87.307.

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.

Savvas Varsamopoulos, Koen Bertels, and Carmen G. Almudever. Comparing neural network based decoders for the surface code. *IEEE Transactions on Computers*, 69(2):300–311, 2020. doi: 10.1109/TC.2019.2948612.

Ge Yan, Shanchuan Li, and Yuxuan Du. Rethink the role of neural decoders in quantum error correction, 2026. URL https://arxiv.org/abs/2605.12046.

A Detailed Hardware-Aware Noise Model

A.1 Parameters in the Hardware Profile

For each physical qubit i, the hardware-aware noise model takes the relaxation and dephasing times $T_{1,i}$ and $T_{2,i}$, the single-qubit gate error ϵ_i^{1q} (equivalently, the fidelity $F_i^{1q} = 1 - \epsilon_i^{1q}$), the single-qubit gate duration τ_i^{1q}, and the readout duration τ_i^{ro}. For each physical connection (i, j), the model takes a coupler-specific two-qubit gate error ϵ_{ij}^{2q} and gate duration τ_{ij}^{2q}, allowing different physical couplers to exhibit different error behavior in the simulation. Finally, the hardware profile also provides state-preparation, measurement, and reset error probabilities for each physical qubit, denoted by p_i^{prep}, p_i^{meas}, and p_i^{reset}, respectively. Table 2 summarizes the complete set of hardware parameters used by the model.

Table 2: Hardware parameters used by the hardware-aware circuit-level noise model.

Category	Symbol	Description	Applied at
Qubit	$T_{1,i}$	Relaxation time of qubit i	Gate and idle
Qubit	$T_{2,i}$	Dephasing time of qubit i	Gate and idle
Qubit	ϵ_i^{1q}	Single-qubit gate error; $F_i^{1q} = 1 - \epsilon_i^{1q}$	Single-qubit gate
Qubit	τ_i^{1q}	Single-qubit gate duration	Single-qubit gate
Qubit	τ_i^{ro}	Readout duration	Readout
Coupler	ϵ_{ij}^{2q}	Two-qubit gate error on connection (i, j)	Two-qubit gate
Coupler	τ_{ij}^{2q}	Two-qubit gate duration on connection (i, j)	Two-qubit gate
SPAM	p_i^{prep}	Preparation error probability for qubit i	Preparation
SPAM	p_i^{meas}	Measurement error probability for qubit i	Measurement
SPAM	p_i^{reset}	Reset error probability for qubit i	Reset

A.2 Construction of Location-Specific Pauli Channels

Relaxation and dephasing. Given the relaxation and dephasing times T_1 and T_2, the decoherence accumulated during an operation or idle period of duration τ can be described by amplitude- and phase-damping channels. To enable stabilizer-based classical simulation, we use their Pauli-twirled approximation (Aaronson & Gottesman, 2004; Geller & Zhou, 2013). To enforce the physical constraint $T_2 \leq 2T_1$, we define

$$\widetilde{T}_{2,i} = \min(T_{2,i}, 2T_{1,i}) \,. \tag{4}$$

For an interval of duration τ, the corresponding Pauli error probabilities are

$$p_{X,i}(\tau) = p_{Y,i}(\tau) = \frac{1 - \exp(-\tau/T_{1,i})}{4}, \qquad p_{Z,i}(\tau) = \frac{1 - \exp(-\tau/\widetilde{T}_{2,i})}{2} - p_{X,i}(\tau). \tag{5}$$

These probabilities define the single-qubit channel

$$\mathcal{P}_i(\tau)(\rho) = (1 - p_{X,i} - p_{Y,i} - p_{Z,i})\,\rho + p_{X,i} X\rho X + p_{Y,i} Y\rho Y + p_{Z,i} Z\rho Z, \tag{6}$$

where the dependence of the probabilities on τ is omitted on the right-hand side for readability.

Single-qubit gates. For a single-qubit operation g on qubit i, the noisy operation is represented as

$$\mathcal{E}_{i,g}^{1q} = \mathcal{D}^{(1)}(r_{i,g}) \circ \mathcal{P}_i\left(\tau_{i,g}^{1q}\right) \circ \mathcal{U}_g, \tag{7}$$

where $\mathcal{U}_g$ is the ideal gate and $\mathcal{D}^{(1)}(r_{i,g})$ is a residual single-qubit depolarizing channel. The residual probability is derived from the difference between the gate fidelity reported in the hardware profile and the fidelity already explained by relaxation and dephasing:

$$r_\ell = \max\left(0, \frac{n_\ell \left(F_\ell^{coh} - F_\ell^{hw}\right)}{n_\ell F_\ell^{coh} - 1}\right), \tag{8}$$

where n_ℓ is the local Hilbert-space dimension, with $n_\ell = 2$ for a single-qubit gate. Here, F_ℓ^{coh} is the fidelity implied by the T_1–T_2 channel and F_ℓ^{hw} is the gate fidelity reported in the hardware profile. This separation prevents the same coherence loss from being counted independently as both decoherence and gate error.

Two-qubit gates. For a two-qubit operation g on the physical connection (i, j), the model similarly applies

$$\mathcal{E}^{2q}_{ij,g} = \mathcal{D}^{(2)}(r_{ij,g}) \circ \left[\mathcal{P}_i\left(\tau^{2q}_{ij}\right) \otimes \mathcal{P}_j\left(\tau^{2q}_{ij}\right)\right] \circ \mathcal{U}_{ij,g}. \tag{9}$$

The residual two-qubit depolarizing probability is obtained from Eq. (8) with $n_\ell = 4$, the product of the two coherence fidelities, and the two-qubit gate fidelity reported for the corresponding physical connection in the hardware profile.

Measurement, preparation, and reset. Measurement noise is represented as a basis-dependent Pauli error immediately before an ideal measurement. Equivalently, if m_i is the ideal measurement outcome,

$$\widetilde{m}_i = m_i \oplus b_i, \qquad b_i \sim \text{Bernoulli}(p^{\text{meas}}_i)\,. \tag{10}$$

An X error is applied before a Z-basis measurement, and a Z error is applied before an X-basis measurement. Preparation and reset errors are inserted after the corresponding ideal operations as state flips with probabilities p^{prep}_i and p^{reset}_i, respectively.

B Detailed Statistics-Guided Calibration

B.1 Detector Statistics and Calibration Objectives

Detector events and pairwise error probabilities. Let $x_{s,t} \in \{0, 1\}$ denote the detector event of stabilizer s in round t. We compute all statistics over the bulk-round set $\mathcal{T}$, excluding the first round and the final readout layer. For two detector events x_i and x_j, we define the pairwise error probability as

$$p(x_i, x_j) = \frac{1}{2} - \frac{1}{2}\sqrt{1 - \frac{4\left(\langle x_i x_j\rangle - \langle x_i\rangle\langle x_j\rangle\right)}{1 - 2\langle x_i\rangle - 2\langle x_j\rangle + 4\langle x_i x_j\rangle}}, \tag{11}$$

where the angle brackets denote empirical averages over shots. This quantity estimates the probability that an independent error mechanism flips both detectors.

Four families of detector statistics. Let $\mathcal{P}$ denote the set of stabilizer pairs that share a data qubit. We define

$$\begin{aligned} r_s &= \langle x_{s,t}\rangle_{t\in\mathcal{T}}\,, & \tau_s &= \langle p(x_{s,t}, x_{s,t+1})\rangle_t\,, \\ \sigma_{ab} &= \langle p(x_{a,t}, x_{b,t})\rangle_t\,, & \kappa_{a\to b} &= \langle p(x_{a,t}, x_{b,t+1})\rangle_t\,, \qquad (a, b) \in \mathcal{P}. \end{aligned} \tag{12}$$

These four families are the detector-event rate r_s of each stabilizer, the temporal correlation τ_s of the same stabilizer across consecutive rounds, the spatial correlation σ_{ab} between neighboring stabilizers in the same round, and the directed spacetime correlation $\kappa_{a\to b}$ between neighboring stabilizers in consecutive rounds.

Uncertainty-normalized residuals. For each statistic S_k, we express the synthetic–hardware mismatch in units of the hardware uncertainty:

$$z_k(\boldsymbol{\theta}) = \frac{S^{\text{syn}}_k(\boldsymbol{\theta}) - \overline{S}^{\text{hw}}_k}{\varsigma_k}, \qquad \varsigma_k = \sqrt{\text{SE}^2_k + \epsilon^2}. \tag{13}$$

Here, $\overline{S}^{\text{hw}}_k$ is the mean estimated from the calibration split of the hardware syndrome data, SE_k is its standard error estimated from disjoint shards of that split, and $\epsilon = 5 \times 10^{-4}$ is a noise floor. Using the data partitions described in Appendix C, the calibration split is divided into ten disjoint shards of 3,000 shots, while the acceptance and held-out evaluation splits are each divided into five shards of 2,000 shots. For a split with S shards, we compute

$$\text{SE}_k = \frac{\text{sd}\left(\widehat{S}^{(1)}_k, \ldots, \widehat{S}^{(S)}_k; \text{ddof} = 1\right)}{\sqrt{S}}. \tag{14}$$

For detector-event rates, we additionally impose the scale floor $\varsigma_k \leftarrow \max(\varsigma_k, 0.002)$.

Robust calibration objectives. For $\delta = 2$, the Huber loss is

$$H_\delta(z) = \begin{cases} \frac{1}{2}z^2, & |z| \le \delta, \\ \delta\left(|z| - \frac{1}{2}\delta\right), & |z| > \delta. \end{cases} \tag{15}$$

Let $\overline{H}_\delta(z_q)$ denote the arithmetic mean of this loss over all residuals in statistic family q. We define the cross-stabilizer and single-stabilizer objectives as

$$\mathcal{L}_{\text{cross}} = w\,\overline{H}_\delta(z_\sigma) + (1-w)\,\overline{H}_\delta(z_\kappa), \tag{16}$$

$$\mathcal{L}_{\text{single}} = 0.85\,\overline{H}_\delta(z_\tau) + 0.15\,\overline{H}_\delta(z_r). \tag{17}$$

The dynamic choice of w is described in Section B.3.

B.2 Calibrated Parameter Groups

We calibrate three groups of multiplicative noise parameters. Their structure is chosen to keep the search low-dimensional while exposing noise mechanisms with different effects on the detector statistics.

CZ-gate errors. For each physical CZ edge e, θ_{CZ} scales its computational error $\epsilon_e = 1 - F_e$. A CZ error may act on the ancilla qubit, the data qubit, or both. An error on a data qubit can flip neighboring stabilizers in the same round or the next round. An error on an ancilla qubit can directly flip its measurement outcome or propagate to data qubits through the remaining CZ gates. Consequently, CZ-gate errors primarily contribute to the spatial and directed spacetime statistics (σ, κ), with secondary effects on temporal correlations.

Data-qubit idle noise. The parameters θ_{D} multiply the idle channels of individual data qubits at the end of each round and during ancilla measurement and reset. These errors can flip stabilizers containing the affected data qubit within the same round, contributing to σ but not to κ. This group allows the calibration to change same-round correlations without forcing the cross-round correlations to move in the same direction.

Measurement and reset errors. The parameters θ_{T} scale the measurement error m_q and reset error ρ_q of each ancilla qubit. The parameterization includes one multiplier for each stabilizer class (X/Z and boundary/interior), together with individual multipliers for stabilizers with the largest temporal residuals. For a selected subset, a contrast parameter c scales the two error sources as $m_q \leftarrow m_q e^c$ and $\rho_q \leftarrow \rho_q e^{-c}$. Measurement and reset errors can flip the same stabilizer in two consecutive rounds, thereby changing (τ, r) without directly creating correlations between different stabilizers.

B.3 Blockwise Calibration and Acceptance

Alternating generations. We alternate between cross-stabilizer and single-stabilizer updates. Cross-stabilizer updates consist of CZ and joint generations, both targeting $\mathcal{L}_{\text{cross}}$, whereas single-stabilizer updates are implemented as time generations targeting $\mathcal{L}_{\text{single}}$. A CZ generation updates the four physical CZ edges whose nearby detector pairs contribute most to $\mathcal{L}_{\text{cross}}$. A joint generation updates these CZ edges together with the data-qubit multipliers θ_{D}, allowing the spatial and spacetime losses to move in different directions. A time generation updates θ_{T} and is constrained not to increase $\mathcal{L}_{\text{cross}}$ by more than 1%. CZ and joint generations are scheduled adaptively according to their recent loss reduction per simulation, and a time generation is inserted after every 20–60 CZ generations.

Dynamic cross-stabilizer weight. At the start of every generation, we simulate the current parameters once using the optimization seeds and evaluate the calibration-split losses $L_{\text{sp}} = \overline{H}_\delta(z_\sigma)$ and $L_{\text{st}} = \overline{H}_\delta(z_\kappa)$, where L_{st} averages over all spacetime entries with $|\Delta t| = 1$ without separating later and earlier entries. We then set

$$w \leftarrow \text{clip}\left(\frac{L_{\text{sp}}}{L_{\text{sp}} + L_{\text{st}}}, 0.15, 0.6\right), \tag{18}$$

falling back to $w_0 = 0.5$ when the denominator is zero. The weight is updated once per generation and then held fixed during that generation's linearized solve, line search, and accept/reject tests.

Robust Gauss–Newton update. Each generation updates a small parameter block. Because the parameters are multiplicative, we work in log space and write $\boldsymbol{\delta} = \Delta \log \boldsymbol{\theta}$. We estimate the Jacobian J of the block's statistics using central finite differences and solve

$$\boldsymbol{\delta}^\star = \underset{\|\boldsymbol{\delta}\|_\infty \le b}{\arg\min} \left\| W^{1/2}\left(\widetilde{J}\boldsymbol{\delta} - \widetilde{z}\right)\right\|_2^2 + \lambda\|\boldsymbol{\delta}\|_2^2. \tag{19}$$

Here, $\widetilde{J}$ and $\widetilde{z}$ are the Jacobian and negative residual scaled by the uncertainty and objective weights, W contains the Huber weights obtained by iteratively reweighted least squares, and the bound b defines an ℓ_∞ trust region. We use $\lambda = 0.75$ and evaluate the line-search candidates $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} \odot \exp(\alpha\boldsymbol{\delta}^\star)$ for $\alpha \in \{1/8, 1/4, 1/2, 1\}$.

Proposal acceptance and stopping. The line search first selects a proposal on the calibration split. The best proposal is then reevaluated on the disjoint acceptance split using two independent simulation seeds and is retained only if the improvement persists for both seeds. In a joint generation, if no line-search step is accepted but at least one candidate improves the joint loss, we solve once more with $w' = \max(0.15, w/2)$; this retry is not used for CZ generations. We define an optimization sweep as one pass through the scheduled cross-stabilizer updates followed by a time generation. Calibration stops after two consecutive sweeps with no accepted update.

C Evaluation Metric Definitions

The Willow dataset contains $50{,}000$ hardware shots, which we partition into $30{,}000$ calibration shots, $10{,}000$ acceptance shots, and $10{,}000$ held-out evaluation shots. The held-out evaluation split is not used for parameter updates, update acceptance, or any other model-construction decision. All reported hardware comparisons in Section 4 use only these $10{,}000$ held-out shots, and each synthetic model is evaluated using the same number of independently generated shots. Downstream decoder monitoring and evaluation in Section 5 use all $50{,}000$ Willow shots, as detailed in Appendix D.4.

C.1 Detector-Event Rate Metrics

Let $D_{s,t}^{(n)} \in \{0, 1\}$ denote the detector event observed at stabilizer s in measurement round t for shot n. For an evaluation set of N shots, the detector-event rate (DER) of stabilizer–round cell (s, t) is

$$\widehat{f}_{s,t} = \frac{1}{N}\sum_{n=1}^{N} D_{s,t}^{(n)}. \tag{20}$$

For the stabilizer profile in Figure 2(a), we average over the bulk-round set $\mathcal{T}$:

$$\overline{f}_s = \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}} \widehat{f}_{s,t}. \tag{21}$$

Let $\mathcal{C}$ denote the set of stabilizer–round cells used in the held-out comparison, and let $\widehat{f}_c^{\mathrm{syn}}$ and $\widehat{f}_c^{\mathrm{hw}}$ be the corresponding synthetic and hardware DERs. We report the mean absolute error in percentage points as

$$\mathrm{MAE} = \frac{100}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} \left|\widehat{f}_c^{\mathrm{syn}} - \widehat{f}_c^{\mathrm{hw}}\right|. \tag{22}$$

Pattern agreement is measured by the Pearson correlation coefficient

$$r = \frac{\sum_{c\in\mathcal{C}}(\widehat{f}_c^{\mathrm{syn}} - \overline{f}^{\mathrm{syn}})(\widehat{f}_c^{\mathrm{hw}} - \overline{f}^{\mathrm{hw}})}{\sqrt{\sum_{c\in\mathcal{C}}(\widehat{f}_c^{\mathrm{syn}} - \overline{f}^{\mathrm{syn}})^2}\sqrt{\sum_{c\in\mathcal{C}}(\widehat{f}_c^{\mathrm{hw}} - \overline{f}^{\mathrm{hw}})^2}}, \tag{23}$$

where $\overline{f}^{\mathrm{syn}}$ and $\overline{f}^{\mathrm{hw}}$ are the means over $\mathcal{C}$. MAE measures absolute rate mismatch, whereas Pearson r measures agreement in the relative DER pattern across cells.

C.2 Structured-Correlation Residuals

For each correlation family $k \in \{\text{space}, \text{time}, \text{spacetime}\}$, let $\mathcal{F}_k$ denote the common set of features evaluated for both the synthetic data and the held-out hardware data. Using the pairwise detector statistics defined in Appendix B.1, we summarize the mismatch for family k by the mean absolute residual

$$\mathrm{MAR}_k = \frac{1}{|\mathcal{F}_k|} \sum_{j \in \mathcal{F}_k} \left| c_j^{\mathrm{syn}} - c_j^{\mathrm{hw}} \right|, \tag{24}$$

where c_j^{syn} and c_j^{hw} are the synthetic and hardware values of correlation feature j, respectively. The same feature set is used for all model comparisons within each family.

C.3 Full and Window-Averaged Correlation Matrices

Full matrix and indexing. The nine bulk rounds and 24 stabilizers define 216 detector coordinates. We order them with stabilizer as the major index and round as the minor index, and form the signed detector correlation matrix $C \in \mathbb{R}^{216\times 216}$ with $C_{ij} = p(x_i, x_j)$, where $p(x_i, x_j)$ is defined in Appendix B.1.

Round-window averaging. Let the time-ordered bulk rounds be $t_0, \ldots, t_8$. We partition them into three nonoverlapping segments $\mathcal{W}_q = \{t_{3q}, t_{3q+1}, t_{3q+2}\}$ for $q \in \{0, 1, 2\}$. More generally, a trailing segment containing fewer than three rounds is discarded. For each segment, we construct an index array by first traversing the 24 stabilizers and then the three rounds within that segment. Selecting these indices on both axes produces an aligned submatrix $C^{(q)} \in \mathbb{R}^{72\times 72}$. Thus, entry i in every segment represents stabilizer $\lfloor i/3 \rfloor$ and within-segment round $i \bmod 3$.

We average the three submatrices elementwise while excluding NaN values. With $F_{ab}^{(q)} = \mathbb{1}[C_{ab}^{(q)} \text{ is finite}]$, the displayed matrix is

$$\overline{C}_{ab} = \frac{\sum_{q=0}^{2} F_{ab}^{(q)} C_{ab}^{(q)}}{\sum_{q=0}^{2} F_{ab}^{(q)}}, \tag{25}$$

whenever the denominator is nonzero, and is NaN otherwise. We reset the main diagonal to NaN after averaging. The associated 72 detector coordinates are (x_s, y_s, ℓ), where (x_s, y_s) is the coordinate of stabilizer s and $\ell \in \{0, 1, 2\}$ is the round within a segment. These coordinates are used to re-estimate the finite-shot noise floor for the averaged matrix.

Masking and comparison metrics. Let $\mathcal{M}$ denote the common set of entries retained after applying the 3σ finite-shot noise mask to the window-averaged representation. The same mask is used for every synthetic–hardware comparison. We vectorize the retained entries as $\mathbf{c}^{\mathrm{syn}} = \mathrm{vec}_{\mathcal{M}}(\overline{C}^{\mathrm{syn}})$ and $\mathbf{c}^{\mathrm{hw}} = \mathrm{vec}_{\mathcal{M}}(\overline{C}^{\mathrm{hw}})$.

Pattern agreement is measured by the Pearson correlation

$$r = \mathrm{corr}\left(\mathbf{c}^{\mathrm{syn}}, \mathbf{c}^{\mathrm{hw}}\right), \tag{26}$$

Absolute matrix mismatch is measured by the MAE and RMSE,

$$\mathrm{MAE} = \frac{1}{|\mathcal{M}|} \sum_{(i,j)\in\mathcal{M}} \left| \overline{C}_{ij}^{\mathrm{syn}} - \overline{C}_{ij}^{\mathrm{hw}} \right|, \tag{27}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{|\mathcal{M}|} \sum_{(i,j)\in\mathcal{M}} \left(\overline{C}_{ij}^{\mathrm{syn}} - \overline{C}_{ij}^{\mathrm{hw}} \right)^2}. \tag{28}$$

We also report the mean absolute correlation magnitude of each matrix,

$$\mathrm{Mean}\,|p| = \frac{1}{|\mathcal{M}|} \sum_{(i,j)\in\mathcal{M}} \left| \overline{C}_{ij} \right|. \tag{29}$$

Table 3: Matrix-level agreement with the held-out Willow reference after round-window averaging. Lower MAE and RMSE and higher Pearson r indicate closer agreement. Mean $|p|$ reports the average absolute correlation magnitude of the corresponding model or hardware reference.

Model	Pearson $r \uparrow$	MAE $\downarrow$	RMSE $\downarrow$	Mean $\lvert p \rvert$
SI1000	0.738	1.13×10^{-3}	2.39×10^{-3}	4.18×10^{-4}
Uniform	0.813	1.11×10^{-3}	2.35×10^{-3}	4.20×10^{-4}
SynQBench-base	0.807	1.17×10^{-3}	1.81×10^{-3}	1.23×10^{-3}
SynQBench	**0.881**	$\mathbf{1.08 \times 10^{-3}}$	$\mathbf{1.45 \times 10^{-3}}$	$\mathbf{1.27 \times 10^{-3}}$
Willow split-half (noise reference)	–	7.20×10^{-4}	9.41×10^{-4}	1.27×10^{-3}

For the hardware-only sampling reference in Figure 3, we divide the held-out hardware shots into two disjoint halves, apply the same round-window averaging procedure to each half, and plot their difference. This split-half residual estimates the matrix variation expected from finite sampling and is not used in model construction or calibration.

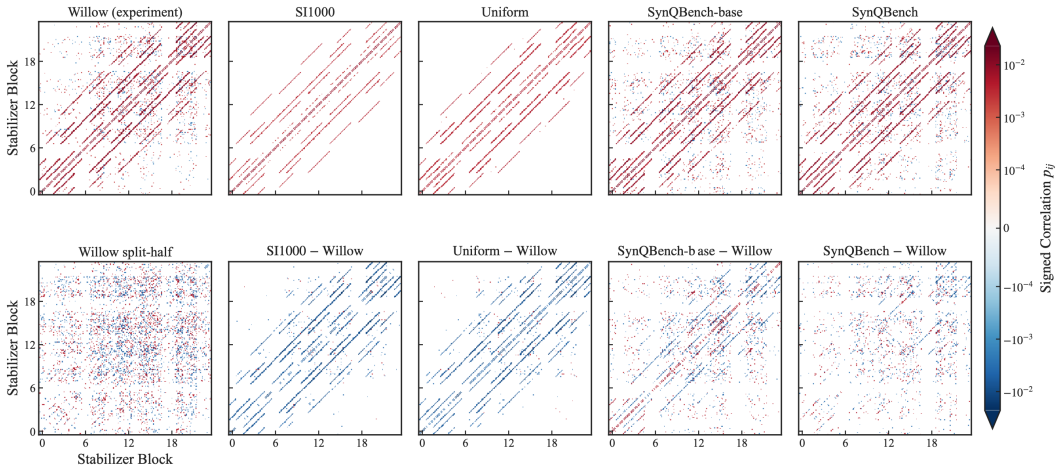


Figure 6: Unaveraged full 216×216 signed detector correlation matrices and residuals on the held-out Willow evaluation split. Detector indices are ordered by stabilizer and then by bulk round. The main-text comparison in Figure 3 is obtained by applying the three-segment averaging procedure described above.

D Decoder and Training Details

D.1 Decoding Task and Notation

We use the decoding task defined in Sec. 2. For a single shot, the detector-event record and logical target are $\mathbf{s} \in \{0,1\}^D$ and $\mathbf{y} \in \{0,1\}^k$, where D is the number of detector events and k is the number of measured logical observables. For neural-network training, a batch of B shots is represented as

$$\mathbf{S} \in \{0,1\}^{B \times D}, \qquad \mathbf{Y} \in \{0,1\}^{B \times k}. \tag{30}$$

All decoder-specific input representations are constructed deterministically from $\mathbf{S}$ together with the circuit or a validated code layout, and are independent of the target labels $\mathbf{Y}$. We additionally use T for the number of temporal detector slices, N_c for the number of check or stabilizer tracks, N_q for the number of data-qubit nodes, and $H \times W$ for the detector-lattice dimensions.

Each neural decoder maps an input batch to one logit per measured logical observable, $\mathbf{Z} = f_\theta(\mathbf{S}) \in \mathbb{R}^{B \times k}$, where θ denotes the trainable parameters. Hard logical predictions are obtained as $\hat{Y}_{ij} = \mathbb{1}[Z_{ij} \geq 0]$, where $\mathbb{1}[\cdot]$ is the indicator function. We train the decoder by minimizing the mean

Table 4: Transformer architecture.

Stage	Layer	Dimension or setting
Input	Detector transport	$B \times T \times N_{\max}$
Embedding	Linear + identity embedding	$1 \to 256$
Refinement	Residual MLP	2 layers, dimension 256
Temporal	GRUCell	$256 \to 256$
Backbone	Transformer block	3 blocks
Attention	Multi-head self-attention	4 heads, key/value dimension 32
Feed-forward	Gated FFN	$256 \to 1536 \to 256$
Projection	Geometry mapping	$256 \to 32$
Pooling	Logical-support pooling	$B \times k \times 32$
Readout	Residual MLP	4 layers, inner dimension 64
Output	Linear	$32 \to 1$

binary cross-entropy (BCE) loss, where $\sigma(\cdot)$ denotes the sigmoid function:

$$\mathcal{L}_{\mathrm{BCE}} = -\frac{1}{Bk}\sum_{i=1}^{B}\sum_{j=1}^{k}\left[Y_{ij}\log\sigma(Z_{ij}) + (1-Y_{ij})\log(1-\sigma(Z_{ij}))\right]. \tag{31}$$

At evaluation time, the resulting hard predictions are scored using the shot-level logical error rate defined in Eq. (1), with a shot counted as a logical failure whenever at least one measured logical observable is predicted incorrectly.

D.2 Neural Decoder Architectures

Neural QEC decoders differ in the inductive biases through which they represent and aggregate syndrome information. These biases determine how geometric locality, parity-check connectivity, and correlations between distant detectors enter the decoding process. We consider four representative architectural families, characterized by their principal operations for modeling spatial relationships.

Transformer. The Transformer decoder maps the detector record into recurrent stabilizer tracks, following the decoding framework introduced in AlphaQubit by Bausch et al. (2024). Its padded transport tensor has shape

$$\mathbf{S} \in \{0,1\}^{B\times T\times N_{\max}}, \tag{32}$$

and an activity mask maps each temporal row to stable check identities. The binary value at each active check is projected to a 256-dimensional representation, augmented with a learned check-identity embedding, and refined by two residual MLP layers.

The recurrent state contains one learned 256-dimensional initial vector per check. At each temporal row, a shared GRUCell updates the active checks. The updated states then pass through three Transformer blocks. Each block contains pre-normalized four-head self-attention with key and value dimension 32 per head, giving a total attention projection dimension of 128. The attention sublayer is followed by a pre-normalized gated feed-forward network,

$$\mathrm{GatedFFN}(\mathbf{x}) = W_o[\mathrm{GELU}(\mathbf{g})\odot\mathbf{c}], \qquad (\mathbf{g},\mathbf{c}) = \mathrm{split}(W_i\mathbf{x}),$$

where

$$W_i : \mathbb{R}^{256} \to \mathbb{R}^{1536}, \qquad W_o : \mathbb{R}^{768} \to \mathbb{R}^{256}.$$

Both sublayers use residual connections. A learned scalar initialized to 0.1 gates the Transformer-block contribution to the recurrent state.

The final check representations are mapped to data-qubit representations according to the circuit-derived geometry. Logical supports then aggregate these representations into one feature vector per logical observable, followed by residual readout layers and a final linear projection to the corresponding logical logit.

Graph Neural Network (GNN). The GNN decoder constructs a Tanner graph containing check nodes, data-qubit nodes, check-to-data incidence edges, and data-to-logical support edges, following

Table 5: GNN architecture.

Stage	Layer	Dimension or setting
Input	Embedding + position embedding	$2 \to 64$
Graph encoding	Tanner message passing	3 layers, dimension 64
Cycle projection	RMSNorm + Linear	$64 \to 128$
Cycle recurrence	GRUCell	$128 \to 128$
Cycle mixing	Graph mixer	2 layers, dimension 128
Terminal projection	RMSNorm + Linear	$64 \to 64$
Terminal recurrence	Projection + GRUCell	$128 \to 64$
Terminal mixing	Graph mixer	12 layers, dimension 64
Logical aggregation	Support product + SwiGLU	Dimension 64
Output	RMSNorm + MLP	$64 \to 128 \to 1$

the graph-based representation of GraphQEC (Hu et al., 2025). Detector events are partitioned into initial, repeated-cycle, and terminal slices, denoted by S_{init}, S_{cycle}, and S_{readout}, respectively.

Each binary check value is embedded into a 64-dimensional representation and combined with a learned node-position embedding. The graph encoder performs three bidirectional Tanner message-passing layers with untied parameters. For a directed edge from source u to target v, the message network receives

$$[\mathbf{h}_u,\ \mathbf{h}_v,\ \mathbf{h}_u \odot \mathbf{h}_v,\ |\mathbf{h}_u - \mathbf{h}_v|]\,.$$

Check-to-data and data-to-check message networks use dimensions $4d \to 2d \to d$, while residual node-update networks use $2d \to 4d \to d$. Messages are degree-normalized at the target node, and the residual updates use SiLU activations and LayerNorm.

The cycle encoder maps the 64-dimensional graph features to 128-dimensional data-node states. A GRUCell updates one recurrent state per data node over the initial and repeated-cycle slices. Two graph-mixer layers then exchange information between data nodes that share a Tanner check, with a learned scalar initialized to 0.1 gating the mixer contribution.

The terminal slice is encoded independently with dimension 64. The recurrent cycle state is projected from 128 to 64 dimensions, updated by a second GRUCell, and processed by twelve data-node graph-mixer layers, with a second scalar gate initialized to 0.1.

For logical observable $j = 1, \ldots, k$, the final data-node states are aggregated over the circuit-derived logical support $\mathcal{S}_j$ using an elementwise product,

$$\mathbf{g}_j = \prod_{q \in \mathcal{S}_j} \tanh(\mathbf{h}_q).$$

A residual SwiGLU layer is then followed by an RMS-normalized two-layer readout MLP with hidden dimension 128, producing one logit per logical observable.

Convolutional Neural Network (CNN). The CNN decoder maps the binary detector record to a validated detector lattice. Each lattice site represents padding, no event, or event, producing an integer tensor

$$\mathbf{S} \in \{0, 1, 2\}^{B \times T \times P \times H \times W}, \tag{33}$$

where P indexes check planes. A three-entry embedding maps each lattice site to a 64-dimensional representation.

The backbone consists of six factorized residual blocks. Each block first projects the 64-dimensional representation to dimension 32, applies the spatial convolution independently to each temporal slice, and then applies a centered width-three temporal convolution independently at each spatial site. A final pointwise projection restores the representation to dimension 64. Batch normalization and SiLU activations are applied within each block.

Table 6: CNN architecture.

Stage	Layer	Dimension or setting
Input	Detector lattice	$B \times T \times P \times H \times W$
Embedding	Token embedding	$3 \to 64$
Backbone $\times 6$	Pointwise projection	$64 \to 32$
Backbone $\times 6$	Spatial convolution	3×3, dimension 32
Backbone $\times 6$	Temporal convolution	Kernel 3, dimension 32, centered
Backbone $\times 6$	Pointwise projection	$32 \to 64$
Pooling	Logical-support aggregation	$B \times k \times T \times 64$
Temporal readout	GRU	$64 \to 64$, one layer
Output	Linear	$64 \to 1$

Table 7: MLP architecture.

Stage	Layer	Dimension or setting
Input	Flatten	$B \times D$
Projection	Linear + LayerNorm + GELU + Dropout	$D \to 512$, dropout 0.1
Backbone $\times 6$	Residual MLP	$512 \to 512 \to 512$
Readout	Linear	$512 \to k$

The residual update is

$$\mathbf{X}_{\ell+1} = \left(\mathbf{X}_\ell + \frac{F_\ell(\mathbf{X}_\ell)}{\sqrt{12}} \right) \odot \mathbf{M}, \tag{34}$$

where $\mathbf{M}$ masks invalid lattice sites. The spatial operator is a checkerboard-aware 3×3 two-site convolution defined by the validated detector geometry (Gu et al., 2026).

Circuit-derived detector-to-data incidence and logical supports aggregate the backbone representations into

$$\mathbf{H} \in \mathbb{R}^{B \times k \times T \times 64}. \tag{35}$$

For each logical observable, a one-layer GRU with input and hidden dimension 64 processes the temporal sequence. A final linear layer maps the recurrent state to the corresponding logical logit.

Multilayer Perceptron (MLP). The MLP decoder is a structure-free reference model that consumes the flattened binary detector-event vector directly, without an explicit spatial or graph representation (Varsamopoulos et al., 2017). The input is projected from dimension D to 512, followed by LayerNorm, GELU, and dropout with probability 0.1.

The resulting 512-dimensional representation passes through six residual MLP blocks. Each block contains two 512-dimensional linear layers with LayerNorm, GELU, and dropout, followed by an identity residual connection. A final linear layer maps the hidden representation to k logical logits.

D.3 Experimental Configuration

Table 8 summarizes the training hyperparameters. All models are trained with AdamW using binary cross-entropy with logits. The learning rate follows a linear warmup and cosine-decay schedule. Gradient clipping and exponential moving-average weights are used throughout training.

Table 9 summarizes the evaluation protocol. Decoder checkpoints are selected exclusively using the matched-noise synthetic validation set. Hardware monitoring and final decoder evaluation use all 50,000 Willow shots, including the noise-model calibration and acceptance subsets; these hardware scores are not used for decoder checkpoint selection. Final checkpoints are also evaluated on independent synthetic test sets from all four noise distributions. Logical error rate (LER) is reported as the primary metric with 95% Wilson confidence intervals. MWPM serves as the reference decoder, with the Willow RL-optimized detector error model (DEM) used as the primary real-data baseline.

Table 8: Training hyperparameters for the SynQBench benchmark.

Hyperparameter	**Value**
Optimizer	AdamW
Learning rate	10^{-4}
LR schedule	Cosine decay
Warmup epochs	5
Decay end epoch	150
Minimum LR	10^{-5}
Batch size	768×4 (effective 3,072)
Epochs	200 (maximum)
Early stopping patience	10
Gradient clipping	1.0
Loss function	BCEWithLogitsLoss
Weight decay	2×10^{-5}
EMA decay	0.999
Precision	BF16

Table 9: Evaluation protocol for the SynQBench benchmark.

Configuration	**Value**
Code	Surface code, $d = 5$, 10 rounds, Z basis
Validation set	10^6 matched-noise simulated shots
Test set	10^6 held-out simulated shots per noise distribution
Cross-noise evaluation	Four simulated noise distributions
Real-data evaluation	5×10^4 Willow shots
Checkpoint selection	Minimum validation loss
Evaluation frequency	Once per epoch
Primary metric	Logical error rate (LER)
Uncertainty	95% Wilson interval
Decision threshold	Zero logit
Reference decoder	MWPM
Real-data reference	Willow RL-optimized DEM

D.4 Detailed Downstream Results

Tables 10–13 report the LERs of all 16 synthetically selected neural checkpoints across the four synthetic test distributions and Willow. The Willow monitor values support the training comparisons in Section 5, while standalone evaluations underlie the sim-to-real gaps in Figure 5. These values can differ slightly because of inference execution settings. Both evaluations use the same 50,000 Willow shots, including those used for simulator calibration. Table 14 reports the corresponding evaluations for the fixed MWPM decoder with the Willow RL-optimized prior DEM.

Table 10: Selected-checkpoint LER (%) for Transformer. Bold indicates the lowest in each row.

	Training distribution			
Evaluation distribution	Uniform	SI1000	SynQBench-base	SynQBench
Selected epoch	14	18	24	53
Uniform	0.1432	0.1804	**0.1230**	0.1315
SI1000	0.1104	0.0953	**0.0716**	0.0837
SynQBench-base	2.7353	2.6651	**1.6573**	1.7913
SynQBench	3.8990	4.1903	2.6510	**2.3061**
Willow (standalone)	3.7720	3.9120	2.8080	**2.7200**
Willow (training monitor)	3.7900	3.9120	2.8080	**2.7240**

Table 11: Selected-checkpoint LER (%) for GNN. Bold indicates the lowest in each row.

	Training distribution			
Evaluation distribution	Uniform	SI1000	SynQBench-base	SynQBench
Selected epoch	38	29	47	61
Uniform	0.1583	0.2030	**0.1268**	0.1375
SI1000	0.1057	0.0941	**0.0724**	0.0828
SynQBench-base	3.4634	3.1912	**1.7690**	1.9457
SynQBench	4.7264	4.7644	2.8367	**2.4361**
Willow (standalone)	4.8600	4.7160	3.0220	**2.7820**
Willow (training monitor)	4.8600	4.7220	3.0300	**2.7860**

Table 12: Selected-checkpoint LER (%) for CNN. Bold indicates the lowest in each row.

	Training distribution			
Evaluation distribution	Uniform	SI1000	SynQBench-base	SynQBench
Selected epoch	51	43	87	107
Uniform	0.2045	0.2317	**0.1214**	0.1302
SI1000	0.1636	0.1140	**0.0681**	0.0792
SynQBench-base	3.8165	3.4559	**1.6545**	1.8295
SynQBench	4.9911	4.8096	2.7092	**2.3831**
Willow (standalone)	5.1960	4.8020	2.9040	**2.7640**
Willow (training monitor)	5.1940	4.7440	2.9080	**2.7500**

Table 13: Selected-checkpoint LER (%) for MLP. Bold indicates the lowest in each row.

	Training distribution			
Evaluation distribution	Uniform	SI1000	SynQBench-base	SynQBench
Selected epoch	29	20	42	67
Uniform	0.2918	0.4367	**0.1851**	0.1999
SI1000	0.2531	0.1747	**0.0984**	0.1329
SynQBench-base	6.2515	5.4953	**2.4764**	3.0010
SynQBench	7.8092	8.0760	3.9851	**3.7195**
Willow (standalone)	7.3720	6.7980	**3.7940**	3.9480
Willow (training monitor)	7.3720	6.7980	**3.7940**	3.9440

Table 14: Evaluation of the fixed PyMatching MWPM decoder with the Willow RL-optimized prior DEM. The absolute gap is measured against its Willow LER and the smallest is marked bold.

Test distribution	Shots	Failures	LER (%)	Absolute gap (pp)
Uniform	1,000,000	2,537	0.2537	3.9203
SI1000	1,000,000	1,469	0.1469	4.0271
SynQBench-base	1,000,000	27,598	2.7598	1.4142
SynQBench	1,000,000	42,968	4.2968	**0.1228**
Willow	50,000	2,087	4.1740	—

E Cross-Platform Hardware Profiles

Figure 7 summarizes representative single-qubit, two-qubit, and SPAM error profiles for Willow (Google Quantum AI and Collaborators, 2024), Sycamore (Cirq Developers, 2021), IBM Heron r2 (IBM Quantum, 2025), Zuchongzhi 3.0 (Gao et al., 2025), Atom Computing (Reichardt et al., 2024), Infleqtion (Radnaev et al., 2025), and Quantinuum H2-2 (Quantinuum, 2025). We retain per-qubit and per-coupler values when available and otherwise use the reported device-level averages.

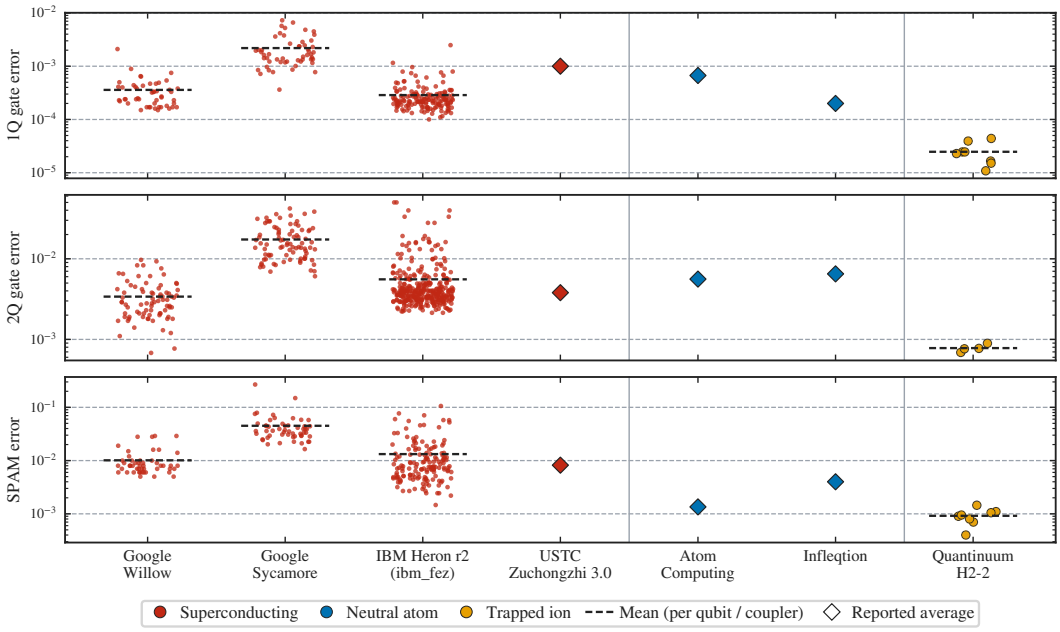


Figure 7: Representative hardware-error profiles across quantum-computing platforms. Points show location-resolved values, dashed lines their means, and diamonds reported device-level averages.