Neural Decoders for Universal Quantum Algorithms

J. Pablo Bonilla Ataides,[*,†] Andi Gu,[*,‡] Susanne F. Yelin,[§] and Mikhail D. Lukin[‖]
Department of Physics, Harvard University, Cambridge, Massachusetts 02138, USA

(Received 10 November 2025; accepted 23 June 2026; published 4 August 2026)

Fault-tolerant quantum computing demands decoders that are fast, accurate, and adaptable to circuit structure and realistic noise. While machine learning decoders have demonstrated impressive performance for quantum memory, their use in algorithmic decoding—where logical gates create complex error correlations—remains limited. We introduce a modular attention-based neural decoder that learns gate-induced correlations and generalizes from training on random circuits to unseen multiqubit algorithmic workloads. Our decoders achieve fast inference and logical error rates comparable to most-likely-error decoders across varied circuit depths and qubit counts. Addressing realistic noise, we incorporate loss-resolving readout, yielding substantial gains when qubit loss is present. We further show that by tailoring the decoder to the structure of the algorithm and decoding only the relevant observables, we can simplify the decoder design without sacrificing accuracy. We validate our framework on multiple error correction codes—including surface codes and two-dimensional color codes—and demonstrate state-of-the-art performance under circuit-level noise. Finally, we show that the attention mechanism learns to track physically meaningful error correlations, consistent with the expected propagation pathways through entangling gates. Enabling experimental validation of deep-circuit fault-tolerant algorithms and architectures [Bluvstein *et al.*, Nature (London) **649**, 39 (2026)], these results establish neural decoders as practical, versatile, and high-performance tools for quantum computing.

DOI: 10.1103/vjn1-mbxl

Fault-tolerant quantum computation requires quantum error correction to suppress the effects of noise below the threshold needed for practical algorithms [1–3]. In stabilizer codes, continuous syndrome measurements extract error information, which a decoder must interpret to compute appropriate recovery operations. As quantum computing advances from proof-of-principle demonstrations to practical applications, the demands on decoders are becoming increasingly stringent: They must be fast enough for real-time operation, accurate enough to maintain subthreshold performance, and flexible enough to handle diverse algorithmic structures and realistic noise models.

Machine learning (ML) decoders have emerged as promising solutions, offering fast inference times and competitive performance compared to conventional alternatives [4–24]. These architectures combine hardware efficiency with neural network adaptability, being deployable on field programmable gate arrays (FPGAs) or application-specific integrated circuits (ASICs) for real-time decoding [25–27]; they further enable training directly on experimental data and fine-tuning simulated models [22,23,28,29].

Despite these advantages, the key focus of prior work was on the conceptually simpler problem of quantum memory—maintaining a logical state over time while performing active error correction. In this setting, qubits remain idle between syndrome measurements, and errors accumulate largely independently on stationary physical qubits. The transition to algorithmic decoding—correcting errors during the execution of logical quantum algorithms—presents fundamentally new challenges that are not addressed by existing ML decoders. During algorithm execution, entangling logical gates dynamically induce structured correlations between errors on different logical qubits. These correlations depend explicitly on the circuit being executed, evolve with circuit depth, and propagate across multiple qubits in a gate-dependent manner. Successfully decoding such circuits requires tracking how correlations are created, transformed, and shared across logical qubits throughout the computation—a qualitatively different task from decoding a stationary memory experiment. Architectures optimized for memory, including recent transformer-based and long short-term memory (LSTM)-based decoders [23,30], do not automatically solve this problem: Without explicit mechanisms to model gate-induced interqubit correlations, decoder performance degrades substantially in multiqubit algorithmic circuits. Moreover, practical quantum algorithms must operate under realistic noise models that include processes such as qubit loss and leakage. These features are either largely absent or difficult to incorporate into existing decoder frameworks.

Recent experimental progress has made these challenges increasingly urgent. Demonstrations of logical quantum algorithms on various platforms [28,31–38] have highlighted the need for improved decoding capabilities in order to meet

[*]These authors contributed equally to this work.
[†]Contact author: jbonillaataides@g.harvard.edu
[‡]Contact author: andigu@g.harvard.edu
[§]Contact author: syelin@g.harvard.edu
[‖]Contact author: lukin@physics.harvard.edu

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

the requirements for fault-tolerant computation. While specialized decoders exist for quantum memory [39–46] and there has been progress toward decoding algorithmic circuits [30,47–51], no existing framework provides the combination of speed, accuracy, and flexibility needed for practical algorithmic decoding.

Thus motivated, in this article, we develop neural decoders specifically tailored for universal quantum algorithms operating under realistic noise conditions. Our approach addresses algorithmic decoding challenges by using an attention-based modular neural architecture—where each logical gate is a distinct trainable module—that jointly tracks error correlations across all logical qubits, enabling fast inference with accuracy comparable to an exact MLE decoder, whose inference cost grows exponentially with system size. Because the architecture learns to model error propagation through individual gates, a single trained model can be automatically applied to new algorithmic circuits, without the need for retraining whenever the logical circuit changes. We further develop qubit loss-aware decoders that exploit loss-resolving readout for improved performance under qubit loss, and combine decoder-level robustness with algorithmic strategies like logical teleportation to prevent loss accumulation. By leveraging recent insights that decoding can be greatly simplified by targeting only the relevant observables of the circuit [49] and by exploiting constrained error propagation, we design specialized, efficient decoders for deep cluster-state circuits, enabling high-performance error correction in measurement-based quantum computing architectures. We additionally show that probing the attention weights reveals that the model learns to focus on correlations consistent with the expected propagation of errors through entangling gates, providing an internal consistency check that the architecture tracks physically meaningful structure. Collectively, these advances establish neural decoders as a robust, accurate, and practical foundation for fault-tolerant quantum algorithms under realistic circuit- and noise-level constraints.

Importantly, the neural decoder methods introduced and analyzed here have already been deployed in recent experimental tests of deep-circuit fault-tolerant algorithms and architectures, demonstrating superior performance on both quantum memory tasks with surface codes and deep-circuit algorithms involving complex encodings [29]. The present work systematically generalizes and extends these methods, uncovering the architectural features underpinning their efficient performance and demonstrating that these decoders can be scaled up and flexibly applied to a broad range of quantum computing tasks.

Before proceeding, we note that Zhou *et al*. [30] recently proposed an LSTM-based multicore circuit decoder (MCCD) for logical circuits and demonstrated decoding of random Clifford circuits on surface-code logical qubits. The modular LSTM design provides a natural way to incorporate circuit structure; however, it primarily uses local, forward-in-time information flow, so syndrome data from one qubit affects the other mainly at entangling gate boundaries, with limited ability for later syndrome evidence to refine the joint error belief. As a result, the MCCD decoder exhibits substantial degradation relative to MLE in circuits with entangling gates, and this degradation increases with circuit depth and code distance, indicating that the architecture does not adequately capture the interqubit correlations central to algorithmic decoding. In contrast, the attention-based architecture developed here provides continuous interqubit communication throughout the circuit, achieving near-MLE performance up to $d = 7$ and generalizing across unseen algorithms.

I. NEURAL DECODER ARCHITECTURE

Our neural decoder architecture [Fig. 1(a)] is designed around a central physical insight: Quantum error correction in algorithmic circuits requires tracking how errors propagate through both space and time, and how entangling gates create new correlations between logical qubits.

Spatially, errors spread between nearby physical qubits through gate operations and stabilizer measurements, and the structure of this spreading is determined by the geometry of the error-correcting code—a two-dimensional (2D) grid for surface codes and a triangular lattice for color codes. Temporally, errors accumulate and correlate as syndrome extraction rounds progress through the circuit. Crucially, in algorithmic circuits, entangling gates introduce a third dimension of structure: They dynamically create correlations between errors on different logical qubits, and these correlations evolve as the circuit proceeds.

These three physical requirements motivate the three core components of our architecture. First, spatial processing layers—implemented as convolutional neural networks (CNNs) for grid-based codes and graph neural networks for irregular topologies—model how detection events at one stabilizer influence error estimates at geometrically nearby stabilizers, respecting the code's native connectivity. Second, recurrent temporal updates allow the decoder to accumulate error information across syndrome rounds, capturing how error evidence builds over the depth of the circuit. Third, an attention mechanism tracks the dynamically evolving correlations between logical qubits induced by entangling gates. Unlike the spatial and temporal components, which operate within individual code blocks, the attention mechanism operates at the logical level, enabling each qubit's error estimate to be informed by syndrome evidence from all other qubits at every step of the circuit. This is essential because the syndrome evidence needed to disambiguate correlated errors often arrives in rounds subsequent to the gate that created the correlation.

The architecture is modular: Each logical gate type corresponds to a distinct trainable module, so a single trained model can be automatically composed to decode any circuit built from the supported gate set without retraining. The decoder takes as input the complete syndrome history from a logical circuit execution and outputs, for each logical qubit, the marginal probability that the qubit has experienced a logical error. We now describe the specific elements of this design.

A. Input and output format

The decoder takes as input a complete description of a logical quantum circuit execution:

(1) *Syndrome measurements*: For each stabilizer measurement round, we provide the measurement outcomes for all

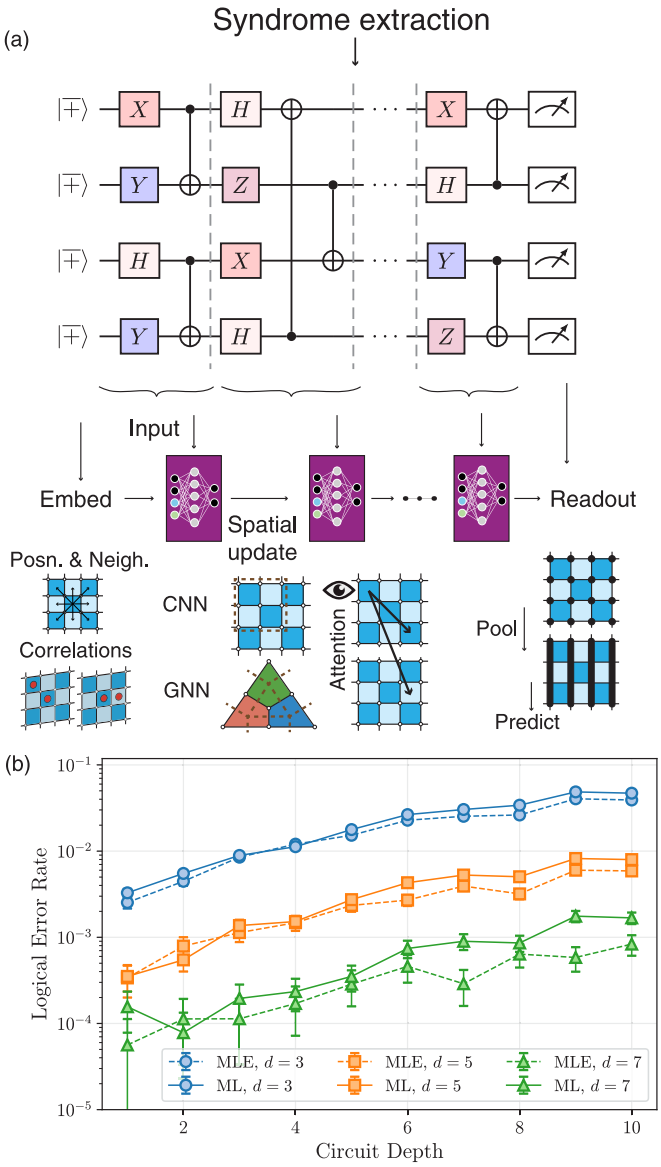


FIG. 1. Neural decoder architecture and performance. (a) Schematic of the neural decoder, which processes quantum error correction circuits to estimate logical errors on each qubit. The input comprises stabilizer measurement circuits with embedded gate and syndrome information. Each stabilizer is assigned a vector representation that evolves via recurrent and spatial updates through the circuit. Spatial processing is implemented using convolutional layers for surface codes and graph neural networks (GNNs) for color codes, with an attention block continuously tracking correlations between logical qubits. The final readout aggregates stabilizer embeddings through pooling to predict logical error probabilities. (b) Decoder performance as a function of circuit depth and code distance, comparing the ML decoder to an MLE decoder. The ML decoder was trained on random circuits and evaluated on independently sampled unseen circuit instances, demonstrating generalization across gate sequences, circuit depths, and code distances from a single trained model. Data are presented as mean values plus or minus standard deviations.

stabilizers across all logical qubits. This includes both X-type and Z-type stabilizer measurements, along with any flag measurements used in the error correction protocol.

(2) *Circuit structure*: We provide the sequence of logical operations (single-qubit gates, two-qubit gates) applied to each logical qubit, along with connectivity information indicating which pairs of logical qubits interact during two-qubit operations.

(3) *Code parameters*: The code distance and type (surface code or color code) for each logical qubit.

(4) *Noise parameters*: A compact representation of the noise model used in the simulation.

The decoder outputs, for each logical qubit, the marginal probability that it has experienced a logical error. Specifically, for circuits ending with computational basis measurements, this represents the probability of a logical X error (bit flip). For circuits ending with X-basis measurements, this represents the probability of a logical Z error (phase flip).

B. Internal representation

The decoder's internal representation must capture the evolving state of knowledge about errors at each point in the code as the circuit progresses. Because errors propagate both locally (through nearby stabilizers) and globally (across logical qubits via entangling gates), this representation must be rich enough to encode both types of information and flexible enough to update as new syndrome evidence arrives.

The core of our architecture is a learned *per-stabilizer embedding* that tracks error-relevant information for each stabilizer throughout the circuit execution. Each stabilizer—whether X type or Z type, and whether associated with syndrome extraction or final measurements—maintains a fixed-dimensional vector representation that evolves as the circuit progresses. These embeddings are designed to capture several types of information:

(1) *Local syndrome history*: The pattern of measurement outcomes for this stabilizer and its neighbors over time.

(2) *Geometric metadata*: The stabilizer's position within the error correction code, as well its connectivity to neighboring stabilizers.

(3) *Temporal correlations*: How errors affecting this stabilizer may have propagated from earlier time steps.

(4) *Interqubit correlations*: For multiqubit circuits, how this stabilizer's errors may be correlated with stabilizers on other logical qubits due to entangling operations.

The stabilizer embeddings are updated through two complementary mechanisms. First, a recurrent update incorporates new syndrome information as each layer of the circuit is processed, allowing the embedding to accumulate information over time. Second, spatial processing operations allow stabilizers to share information with their neighbors, capturing how errors propagate through the code's geometric structure. For surface codes, spatial processing is implemented using two-dimensional convolutions that reflect the grid structure of the code. For color codes, we use graph neural networks [52] that operate on the more complex connectivity pattern of the hexagonal lattice. The choice of spatial operation is thus dictated by the physics of the code: The decoder's information-sharing pathways mirror the geometric structure along which errors actually propagate. In both cases, the spatial operations are designed to model how detection events at one stabilizer should influence the error estimates at

nearby stabilizers, and these correlations are tracked internally through the use of attention.

C. Spatial readout and prediction

To produce logical error predictions, the decoder must aggregate information from all stabilizers associated with each logical qubit. This aggregation respects the structure of the logical operators: For computational basis measurements, we focus on information relevant to X-type logical errors, while for X-basis measurements, we focus on Z-type logical errors. The readout process first converts the per-stabilizer embeddings into per-data-qubit representations through spatial processing operations. For surface codes, this involves additional two-dimensional convolutions that map stabilizer information onto the data qubits they surround. For color codes, we use specialized graph convolutions that propagate information from stabilizer faces to the data qubits on their boundaries. Next, we perform *logical observable pooling*: We aggregate the data qubit representations along the support of the relevant logical operator. For example, when predicting X logical errors in a surface code, we pool information from data qubits along a column of the grid (the support of the Z logical operator, which anticommutes with X logical errors). For the surface code, we pool along each of the appropriate row/column representatives of the logical operator [see Fig. 1(a)], whereas for the color code pooling is conducted along a fixed logical representative. This pooling operation produces a single vector representation for each logical qubit. Finally, a fully connected network processes this aggregated representation to produce the final logical error probability. The network is trained to output well-calibrated probabilities that can be used both for error correction decisions and for confidence estimation in downstream applications [22,29].

D. Training methodology

A key physical requirement is that the decoder must generalize across arbitrary circuits composed from a fixed gate set, rather than memorizing the error statistics of any particular algorithm. The training methodology is designed to enforce this. While the decoder architecture is designed for end-to-end circuit processing, we employ several training techniques to improve data efficiency and generalization.

Intermediate supervision. Rather than only providing supervision at the end of each circuit, we exploit the physical structure of syndrome extraction to create additional training signal. Because transversal measurements can in principle be performed after each circuit layer, we simulate what would happen at each such point, creating n distinct training examples from a single n-layer circuit. This trick, also used in Ref. [22], dramatically increases the effective size of our training dataset.

On-the-fly data generation. We generate training circuits randomly during training, randomly sampling the code distance, number of logical qubits, circuit depth, and gate sequence. This ensures the model sees a diverse range of data during training.

Permutation invariance. To ensure the model can handle arbitrary numbers of logical qubits, we assign each logical qubit a randomly generated "fingerprint" embedding that is resampled for each training instance. The physical motivation is that the identity of a logical qubit should be irrelevant to the decoding—what matters is the circuit structure connecting it to other qubits. By randomizing the fingerprints, we prevent the model from associating particular error patterns with fixed qubit labels, forcing it instead to learn how correlations propagate as a function of the entangling gate structure of the circuit. Further details are provided in the Appendix.

E. Inference

After the neural decoders have been trained, they can be deployed for inference on multiqubit logical algorithms. In Fig. 1(b), we show the logical error rate as a function of circuit depth for the surface code with $d = 3, 5, 7$, comparing our neural decoder against an exact implementation of a decoder that identifies the MLE compatible with the observed syndrome. The MLE decoder achieves state-of-the-art logical error rates, but its computational cost increases exponentially with system size [47]. Remarkably, the trained neural decoder attains logical error rates approaching those of the MLE decoder. This highlights the flexibility and accuracy of our architecture: A single trained model generalizes across all code distances, depths, and circuit instances while maintaining state-of-the-art accuracies. The speed of inference is summarized in Fig. 5(a); decoding shot-by-shot yields a time per gate of approximately 27 ms, which can be reduced to around 100 μs by batching syndromes and utilizing GPU parallelism. Additional details of these simulations and timing benchmarks are provided in the Appendix.

II. LOSS DECODING

Realistic quantum hardware is subject to noise processes beyond simple Pauli errors. Qubit loss—in which a physical qubit is physically removed from the computation—is ubiquitous in platforms such as neutral atom arrays and represents a qualitatively different error mechanism. We now show that explicitly incorporating loss information into the decoder significantly enhances error correction performance, highlighting the flexibility of neural decoders to handle such realistic noise processes. As an example, inspired by the recent experiment of Ref. [29], we decode circuits in which a logical qubit, encoded with a [[15,1,3]] Reed-Muller (RM) code, undergoes alternating rounds of random transversal Clifford gates and logical teleportation, as illustrated in Fig. 2(a). Logical teleportation is implemented via a transversal measurement followed by a classically conditioned logical Clifford correction (or, equivalently, by an added transversal CNOT before measurement), and serves as a key algorithmic primitive to prevent the accumulation of loss [29]. At each teleportation step, lost qubits are detected using loss-resolving readout (LRR), ensuring that loss does not accrue over time. The transversal measurements further enable code checks to be reconstructed and corrections determined, removing the need for repeated rounds of active error correction.

During training, we simulate measurement outcomes from circuits in which each loss event is approximated by the application of a fully depolarizing channel—effectively modeling

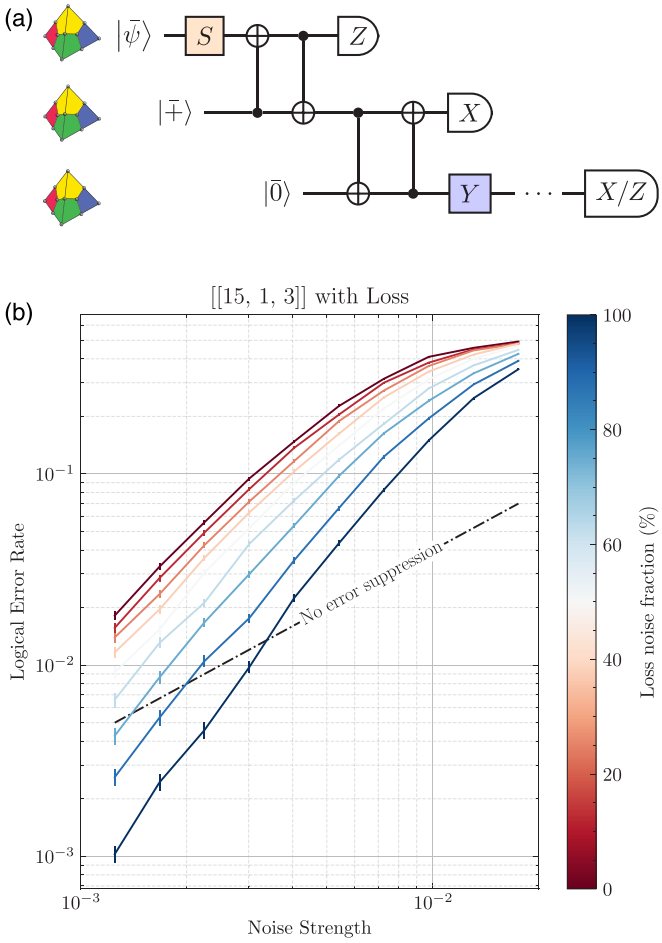


FIG. 2. Loss-aware decoding with neural decoders. (a) Circuit schematic for a logical qubit encoded in a [[15,1,3]] Reed-Muller code, undergoing alternating layers of random transversal Clifford gates and logical teleportation, subject to qubit loss events. Logical teleportation steps integrate loss-resolving readout, enabling the identification and correction of lost qubits. (b) Logical error rate as a function of noise strength for different loss noise fractions (a loss noise fraction of 100% indicates all noise in the system is comprised of loss, while a loss noise fraction of 0% indicates all noise is comprised of Pauli noise). As the loss fraction increases, the loss-aware decoder shows greater benefit, with steeper declines in logical error rate and crossings below the no error suppression line occurring at higher noise strengths.

the loss as a random Pauli error. This "erasure" approximation enables significantly faster data generation, and furthermore allows us to train the neural decoder using the same architecture described earlier, with minimal modification to the data format. Importantly, this approximation is well justified in our setting: Logical information in our circuits is frequently teleported, so there are few possible locations for a loss event to occur before detection and teleportation, limiting the impact of any unmodeled propagation of loss.

For inference, however, we simulate physical loss *exactly*: Lost qubits are removed from the circuit and any subsequent gates acting on them are omitted, so the deployed decoder is evaluated on data generated with a fully realistic noise model. We further assume access to LRR, allowing the decoder to distinguish among measurement outcomes 0, 1, and L (lost) [29,53]. The neural network is equipped with a distinct embedding for each possible outcome, ensuring that the model can effectively utilize loss information to improve logical error rates.

Figure 2(b) displays the logical error rate as a function of noise strength for different loss fractions. As the loss rate increases, the advantage of explicit loss-resolving readout becomes more apparent: The decoder achieves lower logical error rates and is able to maintain error suppression well below the "no error suppression" regime, even at higher noise levels. This is consistent with both theoretical and experimental observations indicating that while undetected loss errors can severally limit the performance of fault-tolerant quantum computing (FTQC), if detected and properly decoded they can result in increased error thresholds [29,53–57]. These results highlight the importance of accurately modeling loss in quantum circuits and demonstrate the capability of machine learning decoders to flexibly adapt to, and exploit, complex noise processes in realistic quantum hardware.

III. CONSTRAINING ATTENTION

Not all circuits require the full generality of all-to-all attention. When the physics of error propagation is sufficiently constrained, the decoder architecture can be simplified accordingly, yielding substantial gains in inference speed. This section thus also provides an empirical ablation of the attention mechanism: Where error correlations are locally bounded, a convolutional architecture without attention matches performance, showing that attention is an effective—not universally necessary—ingredient whose value depends on the correlation structure of the circuit. We now demonstrate this principle using recent insights into algorithmic decoding [49]. Specifically, we demonstrate that by tailoring the decoder to focus on the relevant observables of a quantum circuit—and by constraining the search for correlations to only the qubits associated with those observables—the architecture can be significantly simplified. This reduction in complexity diminishes the need for computationally expensive all-to-all attention mechanisms, resulting in substantial improvements in decoding speed.

Again motivated by recent experiments [29], we focus on one- and two-dimensional cluster-state circuits constructed from logical Steane codes [58], where logical qubits are initialized in the $|+\rangle$ state and entangled by nearest-neighbor controlled-Z (CZ) gates. In this setting, decoding can be simplified by restricting to *reliable logical Pauli products* [49]—in particular, the cluster-state logical stabilizers, which are products of logical $\overline{X}$ on each qubit and logical $\overline{Z}$ on its neighbors.

Backpropagating these logical stabilizers through the circuit shows that each stabilizer terminates as a logical $\overline{X}$ operator on the corresponding input $|\bar{+}\rangle$ state [see Fig. 3(a)]. Thus, predicting the value of each cluster-state stabilizer is equivalent to predicting the sign of the initial input state for that logical qubit. We set this sign of the initial input state as the objective function during training. To ensure balanced training, we randomize input states between $|\bar{+}\rangle$ and $|\bar{-}\rangle$.

The key feature enabling this simplification is the constrained propagation of errors: Z-type errors commute through

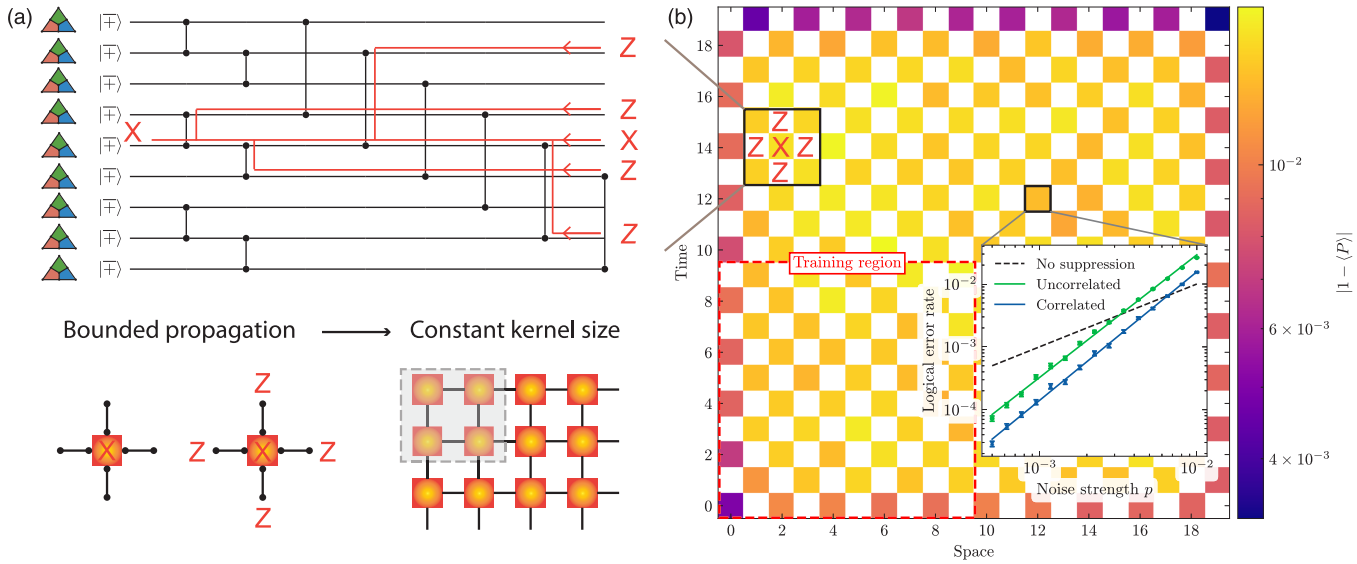


FIG. 3. Deep logical cluster circuits. (a) Schematic of 2D cluster-state circuits built from logical seven-qubit Steane codes. Error propagation is strictly local due to the commutation properties of the gates. (b) Decoder performance for cluster states of size 20×20 at noise strength $p = 0.01$, after training only on clusters up to size 10×10. The decoder predicts the *relevant observables* of the circuit—specifically, the cluster-state stabilizers, which are logical Pauli products comprising logical $\overline{X}$ on each qubit and logical $\overline{Z}$ on neighboring qubits. The results demonstrate generalization to larger systems and greater circuit depths. The inset compares a convolutional decoder exploiting local, observable-specific correlations ("correlated") vs an architecture that does not ("uncorrelated"), highlighting the substantial performance gain from decoding only those observables and qubits with nontrivial short-range error correlations.

the CZ gates, while X-type errors transform into XZ, with the additional Z passing to the neighboring qubit. This mechanism ensures that error correlations remain strictly short range in both space and time. Consequently, only the middle $\overline{X}$ qubit and its neighboring $\overline{Z}$ qubits are relevant for decoding each stabilizer observable.

Leveraging this locality, we replace the full attention-based architecture with a compact CNN decoder with a kernel size of three, sufficient to capture all relevant correlations [Fig. 3(a)]. Decoder performance is assessed by decoding the cluster-state logical stabilizers, corresponding to logical X on each qubit and logical Z on its neighbors. Figure 3(b) shows decoded stabilizer values for both the CNN architecture and, for comparison, an architecture lacking explicit access to local correlations. The CNN decoder, exploiting this structure, achieves superior performance. Furthermore, by measuring logical qubits in alternating X and Z bases, we recover half the logical stabilizers at any given time, with each decoded independently.

In summary, by identifying the relevant reliable observables of the circuit and focusing decoding only on those qubits that exhibit nontrivial correlations—guided by the framework of fast correlated decoding [49]—we are able to construct compact, efficient decoders specifically tailored for deep cluster circuits. This targeted strategy enables high accuracy with significantly reduced model complexity and much faster inference than general-purpose architectures. For instance, Fig. 5 demonstrates that by appropriately constraining attention, inference speed can increase by up to three orders of magnitude. Further details regarding inference time can be found in the Appendix. These findings emphasize that exploiting observable-specific correlations should be a central methodology when developing decoders for circuits with well-defined reliable observables.

IV. ATTENTION WEIGHT ANALYSIS

A natural question is whether the attention mechanism learns to track physically meaningful correlations, or instead relies on opaque statistical patterns. To investigate this, we examine the attention weights throughout the layers of an entangling gate circuit on logical qubits, as depicted in Fig. 4(a). After each layer of entangling operations, we extract and visualize the attention weights in the form of an adjacency matrix [Fig. 4(b)]. Initially, each qubit attends primarily to itself. After each entangling gate layer, the attention weights develop increased off-diagonal elements between precisely those qubits that were coupled by the gate. This pattern is consistent with the expected physical structure: Entangling gates are the operations that create interqubit error correlations, and the attention mechanism learns to focus on exactly these pathways.

This analysis serves as an internal consistency check: It confirms that the attention mechanism is performing its intended physical function—dynamically tracking gate-induced correlations—rather than acting as a generic sequence model. At the same time, we emphasize that the current analysis captures only coarse correlation structures, revealing which qubits the decoder considers correlated but not how it uses this information to make decoding decisions. Developing frameworks for extracting deeper, actionable insights—such as identifying dominant error channels, informing noise model construction, or guiding architectural choices—represents important future work, particularly as neural

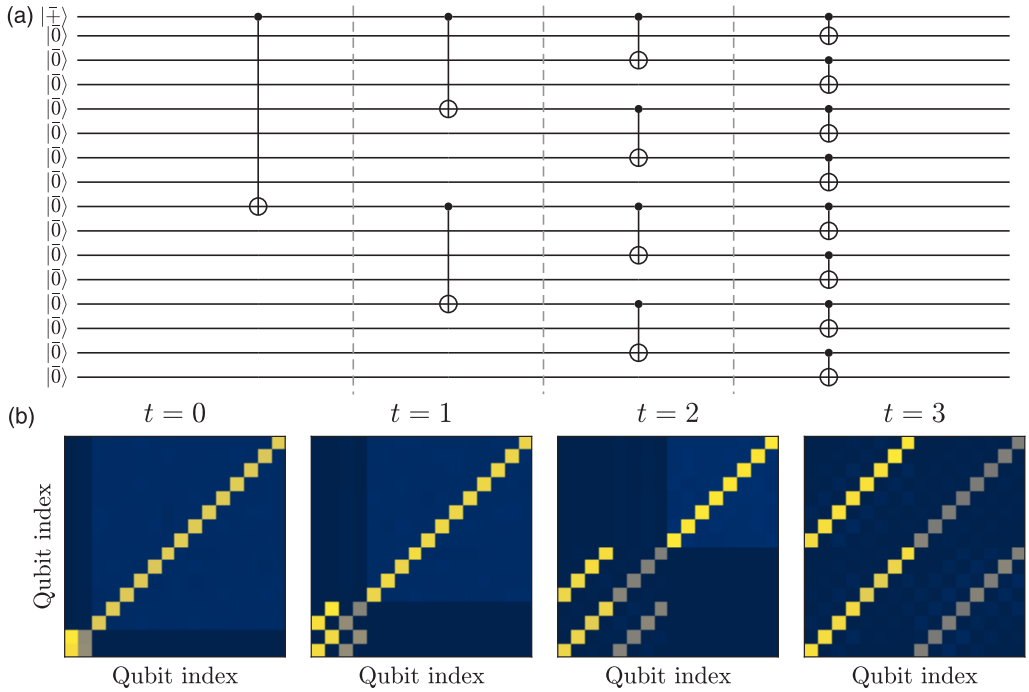


FIG. 4. Attention weight analysis. (a) Circuit diagram showing sequential layers of entangling gates applied to logical qubits. (b) Visualizations of the attention matrix at various stages in the circuit, with yellow indicating high attention weight. Initially, each qubit attends primarily to itself. After each layer of entangling gates, the attention mechanism dynamically tracks the development of correlations between qubits involved in each gate, as indicated by increased off-diagonal elements. This demonstrates that the attention mechanism learns to focus on pathways along which errors can propagate via entangling operations, consistent with the expected propagation of errors through entangling gates.

decoders are deployed in increasingly complex experimental settings.

V. DISCUSSION AND OUTLOOK

We have demonstrated that machine learning decoders constitute a powerful approach for interpreting complex error patterns in quantum circuits. Their practicality depends critically on generating representative training data. For Clifford circuits, such data can be efficiently sampled via classical simulation. The challenge becomes more acute with the inclusion of non-Clifford gates since full quantum state simulation can become intractable. Importantly, our framework circumvents this bottleneck by focusing solely on error propagation and resultant detector outcomes, rather than simulating full quantum states. By carefully tracking how errors propagate through non-Clifford gates, we can continue to efficiently generate necessary syndrome data. This capability substantially broadens the applicability of machine-learning-based decoding, bringing even classically nonsimulable circuits within reach. Additionally, our analysis of attention weights (Fig. 4) confirms that the decoder learns to track correlations along physically expected pathways—the entangling gates in the circuit. While the current analysis is limited to coarse correlation structure rather than detailed mechanistic understanding, it provides a useful validation that the model has internalized the relevant physics of error propagation. Extending this analysis toward actionable insights for decoder design and noise characterization is an important direction for future work.

We note that while memory decoding provides an important baseline for ML decoders, it does not test the architectural capacity central to this work. In memory experiments, logical qubits are stationary and errors accumulate independently—The dynamically evolving gate-induced correlations that drive our architecture design are entirely absent. Indeed, for memory tasks, the full attention mechanism would be unnecessary: A simpler convolutional architecture, analogous to the constrained decoder we developed for cluster-state circuits (Fig. 3), would suffice. The performance demonstrated in Fig. 1(b)—across varying circuit depths, code distances up to $d = 7$, and circuits unseen during training—constitutes a more stringent and relevant test of the architectural contributions than memory benchmarks.

Despite these advances, scalability remains an open problem. As code distance increases, the number of training samples required for reliable performance has been found to grow exponentially [22], making training at large distances prohibitively expensive. Addressing this will require the development of more sample-efficient training strategies, such as transfer learning, data augmentation, or architectural innovations aimed at enhancing generalization. We also note that our timing analysis (Fig. 5) shows that general attention-based decoders can achieve $\sim$100 µs inference per gate when batched, while more specialized architectures can reach submicrosecond speeds. Furthermore, these times can be improved through deployment on dedicated hardware such as FPGAs, making real-time decoding within reach for various quantum computing platforms.

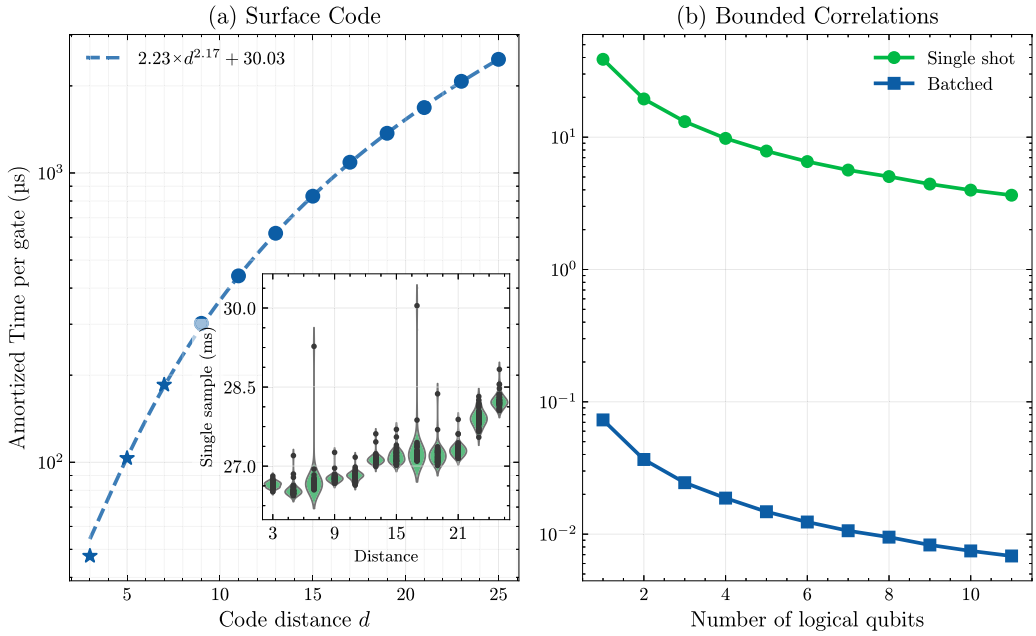


FIG. 5. Neural decoder inference time for random Clifford circuits. The main panel (a) shows the averaged (amortized) inference time per sample as a function of code distance, using batched evaluation on a NVIDIA H100 GPU. The stars indicate distances for which we have evaluated logical error rates (see Fig. 1). The inset displays single-sample inference times. For batched inference, scaling is dominated by attention operations but is mitigated in practice by GPU optimizations, resulting in near-quadratic scaling with code distance. However, single-shot performance is almost independent of code distance, as a result of the high degree of parallelism on GPUs. On the other hand, in panel (b), we show the decoding time per qubit, per gate (as a function of the number of logical qubits), for the decoder architecture used to decode the cluster-state circuit with Steane codes. Importantly, this architecture does not use attention, and employs only convolutions to model error correlations. Not only is this architecture significantly faster, the natural parallelism of the convolution mechanism means the total decoding time depends very weakly on the number of logical qubits (hence the decrease in amortized decoding time as a function of the number of logical qubits). For the "batched" timing on both plots, the batch size is 256.

It is also instructive to compare our approach with the recently published LSTM-based decoder of Ref. [30]. Both works share the goal of modular, ML-based decoding for logical circuits. However, the two architectures differ in a crucial respect: the mechanism for tracking interqubit correlations. In Ref. [30], correlations are modeled by concatenating hidden states at entangling gates, with information flowing only forward through the circuit. Our attention-based architecture instead provides all-to-all communication between logical qubit embeddings at every syndrome round, enabling error beliefs to be continuously updated as new evidence accumulates. This difference is reflected in performance: Reference [30] reports that for $d = 5$ circuits with entangling gates, the MCCD decoder can fall well below MLE performance at moderate circuit depths, whereas our decoder achieves near-MLE accuracy across all tested conditions including $d = 7$. On inference speed, our attention-based decoder achieves amortized times of approximately 100 µs per gate for small distance (Fig. 5), competitive with the ∼500 µs per gate reported for MCCD, while our constrained convolutional architecture for cluster-state circuits reaches submicrosecond speeds. We note, however, that a direct speed comparison is most meaningful between decoders operating in the same accuracy regime. The widening gap between MCCD and MLE with increasing depth and distance, reported in Ref. [30], indicates a qualitatively different operating point from our decoder, which tracks MLE with a small and approximately constant offset across tested regimes. The two frameworks also differ in the scope of circuits they address. MCCD is trained and evaluated on mirror-symmetric random Clifford circuits, in which the second half of the circuit inverts the first, ensuring the noiseless output returns to the input state—a structure used both to define the training signal and to generate ground-truth labels. Our decoder is trained on general random Clifford circuits and deployed on circuits implementing genuine quantum algorithms with arbitrary inputs and outputs, the regime relevant to most practical applications [29]. These observations suggest that explicit, continuous interqubit communication—rather than one-time hidden-state concatenation—is an effective architectural ingredient for near-optimal decoding of multiqubit logical circuits. Our cluster-state results (Fig. 3) further illustrate that when error propagation is constrained, simpler convolutional architectures without attention suffice, so the choice of interqubit communication mechanism should be matched to the correlation structure induced by the circuit. Alternative architectures—for example, graph-based message-passing models tailored to the gate structure [59], or hybrid designs combining local convolutions with sparse global connections—could in principle achieve similar goals, and a systematic comparison of such alternatives is a valuable direction for future work. We also note that our framework encompasses a broader range of codes (surface, color, Reed-Muller), incorporates realistic noise features such as qubit loss, and has been validated experimentally [29].

Another important direction is the extension to more general classes of codes. Although our current work focused on surface, color, and Reed-Muller codes, the modular design of our neural decoders—notably, the use of graph neural networks for modeling code connectivity—readily accommodates more complex codes, including topological and low-density parity-check (qLDPC) codes [60–63]. While most progress in qLDPC decoding has focused on simpler memory tasks and noise models [64–69], extending fast, circuit-level decoding to these architectures represents an exciting avenue for future work, requiring new solutions for syndrome processing and error correlation tracking in more intricate code geometries.

Looking ahead, as experiments move toward increasingly complex logical circuits and more realistic noise regimes, the need for fast, accurate, and adaptable decoders will become ever more pronounced. With demonstrated performance on algorithmic circuits and universal gate sets our neural decoder framework offers a strong foundation for this next stage of fault-tolerant quantum computation.

ACKNOWLEDGMENTS

We acknowledge helpful discussions with Dolev Bluvstein, Sasha Geim, Madelyn Cain, Dom Kufel, Nishad Maskara, Gefen Baranes, Varun Menon, and Nazli Ugur Koyluoglu. We particularly thank Hengyun (Harry) Zhou for providing detailed feedback on our manuscript and results. A.G. thanks Tejasvi Kothapalli for introducing him to the `schedulefree` optimizer [70] used in training the convolutional models for decoding 2D cluster-state circuits. J.P.B.A. and A.G. acknowledge support from the Unitary Foundation. A.G. acknowledges support from the IBM fellowship. We acknowledge support from the National Science Foundation (PHY-2012023 and CCF-2313084) and through the CUA Physics Frontiers Center (PHY-2317134) and NVQL (PHY-2410716) and from DOE through the QUACQ collaboration (DE-SC0025572), Quantum Systems Accelerator Center (DE-AC02-05CH11231), IARPA and the Army Research Office, under the Entangled Logical Qubits program (W911NF-23-2-0219), and the DARPA MeasQuIT program (HR0011-24-9-0359).

DATA AVAILABILITY

An implementation of the neural decoder framework described in this work is publicly available in Ref. [71].

APPENDIX

1. Architecture

The neural decoder implements a multistage architecture that processes syndrome measurements through three conceptually distinct phases: embedding, spatial-temporal processing, and prediction. Unlike traditional decoders that operate on fixed syndrome patterns, our architecture dynamically tracks error correlations as they evolve through the quantum circuit, adapting its internal representations based on both the error correction code structure and the quantum algorithm being executed.

Core processing pipeline. The decoder maintains evolving vector representations for each stabilizer measurement location. These representations are updated through alternating spatial and temporal processing layers: Spatial layers aggregate information across the code's topology (using convolutions for grid codes like surface codes, or graph neural networks for irregular topologies like color codes), while temporal layers track how errors propagate through circuit layers using gated recurrent mechanisms. This alternating structure allows the model to learn complex spatiotemporal error patterns that arise from coherent errors and correlated noise.

Input representation and preprocessing. Raw syndrome measurements are preprocessed through code-specific transformers that preserve topological structure. For surface codes, measurements are arranged on a 2D grid; for color codes, the triangular lattice structure is maintained; and for quantum RM codes, specialized graph representations capture the code's connectivity. The preprocessor also constructs dynamic connectivity graphs that change at each time step to reflect transversal gates—When an entangling gate connects two logical qubits, edges are added between their respective syndrome embeddings, allowing error correlations to propagate.

Embedding system. Each stabilizer measurement is embedded into a high-dimensional vector space (for the decoders in this work, the dimension is 384) by combining multiple information sources: the binary detector value, spatial position within the code, the type of gate applied, noise parameters, temporal position, and (optionally) loss information for erasure events. These diverse features are combined through learnable weighted sums and projected to form the initial stabilizer representations.

Logical qubit context. A key innovation is the logical qubit embedding system that tracks correlations between logical qubits. Each logical qubit maintains a "fingerprint" embedding that evolves based on the gates applied to it. When entangling gates create correlations between logical qubits, these embeddings are updated through graph neural networks operating on the algorithm's connectivity graph, followed by attention mechanisms and bidirectional recurrent layers. These logical-level embeddings are integrated into the spatial processing, allowing the decoder to distinguish between local physical errors and correlated logical errors.

Spatial-temporal processing. The core of the decoder alternates between spatial and temporal updates. Spatial processing uses a combination of local operations (convolutions or graph neural networks depending on code topology) and global attention across all stabilizers. The attention mechanism can dynamically focus on relevant error patterns, with logical qubit embeddings modulating which correlations to track. Temporal processing uses gated updates similar to gated recurrent units (GRUs) but optimized for the discrete time steps of syndrome extraction rounds. Each processing block includes residual connections and normalization for training stability.

Measurement-basis-aware readout. The readout module implements sophisticated prediction heads tailored to the measurement basis. It employs dual attention mechanisms with separate parameter sets for X-basis and Z-basis measurements, together with learnable query tokens assigned to each logical observable. The pooled representation is subsequently

processed by a multilayer classifier composed of successive linear transformations interleaved with gaussian error linear unit (GELU) activations, culminating in a scalar prediction for each logical error probability.

The aggregate parameter count for each of these components together is 13.6 million parameters.

2. Training methodology

Loss definition. The job of the neural network decoder is to predict logical flips that have occurred along the readout basis for each logical qubit. The value of this logical flip can be derived by tracking Pauli errors that occur at each step of the circuit, extracting the cumulative Pauli error right before the transversal readout, and calculating the relevant parity. For instance, if the Pauli error afflicting the three qubits along the logical observable for a $d = 3$ surface code were ZYI, and the readout basis was $\bar{X}$, the logical flip would be 0 (since the number of Z-type errors in this error configuration is 2). Crucially, this method of calculating the logical flip can be used without reference to the actual value of the measurement outcome. For algorithms such as simple memories, this is completely equivalent to checking the parity of the final transversal measurement values along the appropriate logical observable. However, for circuits where the transversal measurement value is nondeterministic (even in the absence of noise)—which is the case for most circuits with magic-state inputs—the transversal measurement values cannot be used to derive the logical flip, and one must resort to calculating the logical flip using the relevant Pauli errors. With this definition of logical flip, the decoder can be treated as a simple binary classifier, and the training loss is defined simply as a binary cross entropy loss, using the derived logical flip as the true label.

Data generation pipeline. Training data are generated online through a simulation pipeline. Random quantum algorithms are sampled with uniform distributions over gate types, circuit depths, numbers of logical qubits, and code parameters. A quantum circuit simulator (Stim) generates syndrome measurements under circuit-level noise, with noise parameters sampled from continuous distributions. The simulation includes realistic effects such as gate errors, measurement errors, and optional qubit loss.

Optimization and learning rate scheduling. We employ AdamW optimization with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$, without weight decay. The learning rate follows a cosine annealing schedule with 1000-step linear warmup:

$$\eta(t) = \begin{cases} \eta_0 \times t/T_0, & t < T_0, \\ \eta_{\min} + \frac{\eta_0 - \eta_{\min}}{2}\left(1 + \cos\left(\frac{\pi \times (t - T_0)}{T_{\max} - T_0}\right)\right), & t \geqslant T_0, \end{cases} \tag{A1}$$

where $\eta_0 = 10^{-4}$, $\eta_{\min} = 10^{-5}$, $T_0 = 1000$, and $T_{\max} = 10^6$ steps.

Training strategies for improved generalization. A key training trick is layered supervision: For a circuit with n layers, the simulator constructs n separate simulation snapshots. The ith snapshot contains syndrome measurements for the cumulative circuit up to layer i followed by transversal readout. The decoder processes syndrome measurements from all n snapshots and produces predictions at each layer, with the loss computed across all predictions. This provides n times more training signal per circuit compared to end-to-end supervision, dramatically improving data efficiency and helping the model learn how errors accumulate through circuit layers. A similar trick is used by Bausch *et al.* [22]. Mixed-precision training with half-precision accelerates training while maintaining numerical stability.

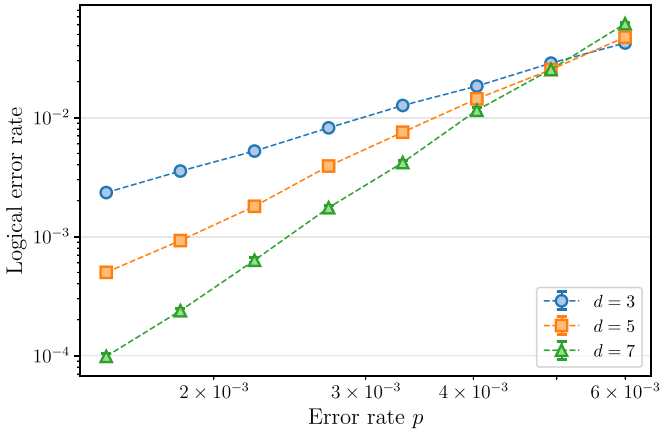


FIG. 6. Memory performance of 2D color code decoder. Logical error rate as a function of physical error rate p for 2D color codes of distance $d = 3, 5, 7$, decoded using a graph neural network architecture. The observed threshold of approximately 0.5% matches state-of-the-art results for circuit-level noise. Data are presented as mean values plus or minus standard deviations.

In total, we trained five distinct models corresponding to the figures presented in the main text. The general attention-based decoder for surface codes was used in Fig. 1, and was trained with a batch size of 128 for 450 000 steps. The loss-aware decoder for the [[15,1,3]] Reed-Muller code (Fig. 2) was trained with a batch size of 192 for 200 000 steps. For cluster-state circuits (Fig. 3), both the correlated and uncorrelated decoders were trained with a batch size of 256 for 100 000 steps. Finally, a graph neural network decoder was trained for 2D color codes with flag-based extraction (Fig. 6), using a batch size of 256 for 250 000 steps. Each model was trained independently with identical optimization settings, differing only in the circuit families and code parameters used for data generation.

3. Run-time analysis

We profile inference time to assess scalability. Figure 5 shows inference time for the neural decoder on random Clifford algorithms. Batch-mode inference on NVIDIA H100 GPUs reveals that while theoretical complexity is $O(d^4)$ due to all-to-all attention, empirical scaling is approximately $O(d^2)$ due to GPU optimization. Single-sample inference shows minimal distance dependence due to parallelism. For specialized architectures like the cluster-state decoder (Fig. 3), replacing attention with fixed-kernel convolutions yields sub-microsecond inference per qubit, with total decoding time nearly independent of logical qubit count. We note that the single-sample inference times can be significantly improved,

and brought closer to the amortized run times, by deploying the models on dedicated hardware, such as FPGAs.

4. Other codes

The general algorithmic decoder framework can be extended to a wide range of quantum codes by adapting the spatial update layer [see Fig. 1(a)]. Convolutions are naturally suited for surface codes and other square-lattice variants, given their grid-based spatial structure. For other codes, such as topological color codes, applying standard CNNs is less suitable. For example, forcing a triangular lattice to fit a square grid would necessitate extra padding, complicating the update steps.

A natural and effective generalization is to use graph neural networks. By constructing a stabilizer connectivity graph—where stabilizers are linked if their support overlaps—the GNN can perform updates analogous to convolutions, but directly on the code's true geometry.

We apply a GNN-based decoder to the 2D color code family, using flag-based syndrome extraction as in Ref. [72]. As shown in Fig. 6, the decoder achieves a memory circuit threshold near 0.5%, on par with state-of-the-art circuit-noise decoding for color codes [73,74].

This stabilizer-graph-driven spatial update approach via GNNs is highly versatile—It can be extended to other classes of codes, including high-rate quantum LDPC codes [75]. A comprehensive investigation of this generalized strategy for broader code families is left for future work.

5. Numerical simulation details

We now provide additional details regarding the numerical simulations described in the main text.

Figure 1(b) shows results for the general algorithm decoder applied to random transversal Clifford circuits on four surface-code logical qubits. The qubits are first initialized in the logical $|\overline{+}\rangle$ state using a noiseless unitary circuit. Subsequent circuit layers feature noisy transversal single-qubit Clifford gates, randomly drawn from the set I, X, Y, Z, H on each qubit. Since the Hadamard gate is not transversal in the surface code—it swaps the logical $\overline{X}$ and $\overline{Z}$ operators—we implement it by applying Hadamard gates to all physical qubits, followed by a noiseless lattice rotation to align the logical operators accordingly.

Each layer is followed by a round of two-qubit CNOT gates between randomly selected pairs of logical qubits, applied in random orientations. Although our model architecture includes support for SWAP gate modules, the reported results are restricted to circuits with conventional CNOT gates. After the entangling gate layer, we perform syndrome extraction by measuring the stabilizer checks of the surface code using the standard fault-tolerant pattern.

After the initial noiseless preparation, noise is applied to all physical qubits and gates according to a circuit-level depolarizing noise model. In particular, if we let p characterize the noise strength of the two-qubit gate, so that each of the 15 errors $\{IX, IY, \ldots, ZZ\}$ occur with the same probability $p/15$, then single-qubit gates fail with probability $p/10$, so that each of the three Pauli errors occur with one-third of that probability, and both resets and measurements fail with probability $p/10$. The results in Fig. 1(b) correspond to a fixed physical error rate of $p = 0.3\%$.

For inference, we test decoder performance on circuits of varying depth. For each circuit depth, we sample 20 different random Clifford circuits with the protocol described above. For each sampled circuit, we generate multiple noise realizations (shots) and apply both the trained ML decoder and an optimal MLE decoder, which is implemented with gurobi [47].

To evaluate decoding accuracy, we focus on logical observables associated with the stabilizer state generated by the circuit. Since simulation of Y-basis measurement in the surface code is more complex, our analysis focuses on observables composed solely of I, X, or Z logical operators. While the ML decoder outputs logical flip probabilities for all logical qubits, the comparison with the MLE decoder is performed by evaluating the parity of a specified target stabilizer after decoding.

The logical error rates presented as a function of circuit depth reflect the fraction of shots on which the decoded parity differed from the true value for the target observable. Reported logical error rates are normalized on a per-round basis.

Figure 2(b) shows decoding performance for a logical algorithm in which a qubit encoded in the [[15,1,3]] RM code undergoes repeated layers of transversal single-qubit Clifford gates, Steane-style error correction, and logical teleportation. Ancilla blocks used for syndrome extraction and teleportation are also initialized as [[15,1,3]] RM codes to ensure consistency in error protection throughout the circuit.

During training, we sample circuits of randomly chosen depths, where each layer of single-qubit Clifford gates is selected uniformly from the subset of gates transversal on the RM code: I, X, Y, Z, S. We prepare the RM state noiselessly using a unitary preparation circuit. All subsequent operations of the logical algorithm are noisy. The error model assigns the total error budget for each operation between Pauli errors and loss errors, according to the "loss noise fraction" parameter shown in Fig. 2.

Because the RM code is comparatively small, we model its structure with a dense neural network architecture, embedding code-specific features directly into the decoder input. This same approach generalizes to other small quantum codes such as the [[16,6,4]] Tesseract code [29].

[1] D. Gottesman, Stabilizer codes and quantum error correction, Ph.D. thesis, California Institute of Technology, 1997.

[2] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Topological quantum memory, J. Math. Phys. **43**, 4452 (2002).

[3] A. Y. Kitaev, Fault-tolerant quantum computation by anyons, Ann. Phys. **303**, 2 (2003).

[4] S. Krastanov and L. Jiang, Deep neural network probabilistic decoder for stabilizer codes, Sci. Rep. **7**, 11003 (2017).

[5] S. Varsamopoulos, B. Criger, and K. Bertels, Decoding small surface codes with feedforward neural networks, Quantum Sci. Technol. **3**, 015004 (2018).
[6] G. Torlai and R. G. Melko, Neural decoder for topological codes, Phys. Rev. Lett. **119**, 030501 (2017).
[7] P. Baireuther, T. E. O'Brien, B. Tarasinski, and C. W. J. Beenakker, Machine-learning-assisted correction of correlated qubit errors in a topological code, Quantum **2**, 48 (2018).
[8] C. Chamberland and P. Ronagh, Deep neural decoders for near term fault-tolerant experiments, Quantum Sci. Technol. **3**, 044002 (2018).
[9] P. Baireuther, M. D. Caio, B. Criger, C. W. J. Beenakker, and T. E. O'Brien, Neural network decoder for topological color codes with circuit level noise, New J. Phys. **21**, 013003 (2019).
[10] N. Maskara, A. Kubica, and T. Jochym-O'Connor, Advantages of versatile neural-network decoding for topological codes, Phys. Rev. A **99**, 052351 (2019).
[11] P. Andreasson, J. Johansson, S. Liljestrand, and M. Granath, Quantum error correction for the toric code using deep reinforcement learning, Quantum **3**, 183 (2019).
[12] R. Sweke, M. S. Kesselring, E. P. L. van Nieuwenburg, and J. Eisert, Reinforcement learning decoders for fault-tolerant quantum computation, Mach. Learn.: Sci. Technol. **2**, 025005 (2021).
[13] S. Varsamopoulos, K. Bertels, and C. G. Almudever, Decoding surface code with a distributed neural network–based decoder, Quantum Mach. Intell. **2**, 3 (2020).
[14] L. D. Colomer, M. Skotiniotis, and R. Muñoz-Tapia, Reinforcement learning for optimal error correction of toric codes, Phys. Lett. A **384**, 126353 (2020).
[15] X. Ni, Neural network decoders for large-distance 2D toric codes, Quantum **4**, 310 (2020).
[16] D. Fitzek, M. Eliasson, A. F. Kockum, and M. Granath, Deep Q-learning decoder for depolarizing noise on the toric code, Phys. Rev. Res. **2**, 023230 (2020).
[17] T. Wagner, H. Kampermann, and D. Bruß, Symmetries for a high-level neural decoder on the toric code, Phys. Rev. A **102**, 042411 (2020).
[18] K. Meinerz, C.-Y. Park, and S. Trebst, Scalable neural decoder for topological surface codes, Phys. Rev. Lett. **128**, 080505 (2022).
[19] H. Wang, P. Liu, K. Shao, D. Li, J. Gu, D. Z. Pan, Y. Ding, and S. Han, Transformer-QEC: Quantum error correction code decoding with transferable transformers, arXiv:2311.16082.
[20] S. Gicev, L. C. L. Hollenberg, and M. Usman, A scalable and fast artificial neural network syndrome decoder for surface codes, Quantum **7**, 1058 (2023).
[21] C. Chamberland, L. Goncalves, P. Sivarajah, E. Peterson, and S. Grimberg, Techniques for combining fast local decoders with global decoders under circuit-level noise, Quantum Sci. Technol. **8**, 045011 (2023).
[22] J. Bausch, A. W. Senior, F. J. H. Heras, T. Edlich, A. Davies, M. Newman, C. Jones, K. Satzinger, M. Y. Niu, S. Blackwell, G. Holland, D. Kafri, J. Atalaya, C. Gidney, D. Hassabis, S. Boixo, H. Neven, and P. Kohli, Learning high-accuracy error decoding for quantum processors, Nature (London) **635**, 834 (2024).
[23] M. Lange, P. Havström, B. Srivastava, I. Bengtsson, V. Bergentall, K. Hammar, O. Heuts, E. van Nieuwenburg, and M. Granath, Data-driven decoding of quantum error correcting codes using graph neural networks, Phys. Rev. Res. **7**, 023181 (2025).
[24] B. M. Varbanov, M. Serra-Peralta, D. Byfield, and B. M. Terhal, Neural network decoder for near-term surface-code experiments, Phys. Rev. Res. **7**, 013029 (2025).
[25] R. W. J. Overwater, M. Babaie, and F. Sebastiano, Neural-network decoders for quantum error correction using surface codes: A space exploration of the hardware cost-performance tradeoffs, IEEE Trans. Quantum Eng. **3**, 3101719 (2022).
[26] A. Boutros, A. Arora, and V. Betz, Field-programmable gate array architecture for deep learning: Survey and future directions, Proc. IEEE **113**, 613 (2025).
[27] C. N. Coelho, A. Kuusela, S. Li, H. Zhuang, J. Ngadiuba, T. K. Aarrestad, V. Loncar, M. Pierini, A. A. Pol, and S. Summers, Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors, Nat. Mach. Intell. **3**, 675 (2021).
[28] Google Quantum AI, Quantum error correction below the surface code threshold, Nature (London) **638**, 920 (2025).
[29] D. Bluvstein, A. A. Geim, S. H. Li, S. J. Evered, J. P. Bonilla Ataides, G. Baranes, A. Gu, T. Manovitz, M. Xu, M. Kalinowski, *et al.*, A fault-tolerant neutral-atom architecture for universal quantum computation, Nature (London) **649**, 39 (2026).
[30] Y. Zhou, C. Wan, Y. Xu, J. P. Zhou, K. Q. Weinberger, and E.-A. Kim, Learning to decode logical circuits, Nat. Comput. Sci. **5**, 1158 (2025).
[31] L. Egan, D. M. Debroy, C. Noel, A. Risinger, D. Zhu, D. Biswas, M. Newman, M. Li, K. R. Brown, M. Cetina, *et al.*, Fault-tolerant control of an error-corrected qubit, Nature (London) **598**, 281 (2021).
[32] C. Ryan-Anderson, J. G. Bohnet, K. Lee, D. Gresh, A. Hankin, J. P. Gaebler, D. Francois, A. Chernoguzov, D. Lucchetti, N. C. Brown, T. M. Gatterman, S. K. Halit, K. Gilmore, J. Gerber, B. Neyenhuis, D. Hayes, and R. P. Stutz, Realization of real-time fault-tolerant quantum error correction, Phys. Rev. X **11**, 041058 (2021).
[33] Y. Zhao, Y. Ye, H.-L. Huang, Y. Zhang, D. Wu, H. Guan, Q. Zhu, Z. Wei, T. He, S. Cao, *et al.*, Realization of an error-correcting surface code with superconducting qubits, Phys. Rev. Lett. **129**, 030501 (2022).
[34] S. Krinner, N. Lacroix, A. Remm, A. Di Paolo, E. Genois, C. Leroux, C. Hellings, S. Lazar, F. Swiadek, J. Herrmann, G. J. Norris, C. K. Andersen, M. Müller, A. Blais, C. Eichler, and A. Wallraff, Realizing repeated quantum error correction in a distance-three surface code, Nature (London) **605**, 669 (2022).
[35] Google Quantum AI, Suppressing quantum errors by scaling a surface code logical qubit, Nature (London) **614**, 676 (2023).
[36] D. Bluvstein, S. J. Evered, A. A. Geim, S. H. Li, H. Zhou, T. Manovitz, S. Ebadi, M. Cain, M. Kalinowski, D. Hangleiter, *et al.*, Logical quantum processor based on reconfigurable atom arrays, Nature (London) **626**, 58 (2024).
[37] V. V. Sivak, A. Eickbusch, B. Royer, S. Singh, I. Tsioutsios, S. Ganjam, A. Miano, B. L. Brock, A. Z. Ding, L. Frunzio, S. M. Girvin, R. J. Schoelkopf, and M. H. Devoret, Real-time quantum error correction beyond break-even, Nature (London) **616**, 50 (2023).
[38] R. S. Gupta, N. Sundaresan, T. Alexander, C. J. Wood, S. T. Merkel, M. B. Healy, M. Hillenbrand, T. Jochym-O'Connor, J. R. Wootton, T. J. Yoder, A. W. Cross, M. Takita, and B. J.

Brown, Encoding a magic state with beyond break-even fidelity, Nature (London) **625**, 259 (2024).

[39] A. G. Fowler, Optimal complexity correction of correlated errors in the surface code, arXiv:1310.0863.

[40] S. Bravyi, M. Suchara, and A. Vargo, Efficient algorithms for maximum likelihood decoding in the surface code, Phys. Rev. A **90**, 032326 (2014).

[41] S. Huang, M. Newman, and K. R. Brown, Fault-tolerant weighted union-find decoding on the toric code, Phys. Rev. A **102**, 012419 (2020).

[42] N. Delfosse and N. H. Nickerson, Almost-linear time decoding algorithm for topological codes, Quantum **5**, 595 (2021).

[43] O. Higgott, T. C. Bohdanowicz, A. Kubica, S. T. Flammia, and E. T. Campbell, Improved decoding of circuit noise and fragile boundaries of tailored surface codes, Phys. Rev. X **13**, 031007 (2023).

[44] N. Sundaresan, T. J. Yoder, Y. Kim, M. Li, E. H. Chen, G. Harper, T. Thorbeck, A. W. Cross, A. D. Córcoles, and M. Takita, Demonstrating multi-round subsystem quantum error correction using matching and maximum likelihood decoders, Nat. Commun. **14**, 2852 (2023).

[45] O. Higgott and C. Gidney, Sparse Blossom: Correcting a million errors per core second with minimum-weight matching, Quantum **9**, 1600 (2025).

[46] B. Barber, K. M. Barnes, T. Bialas, O. Buğdaycı, E. T. Campbell, N. I. Gillespie, K. Johar, R. Rajan, A. W. Richardson, L. Skoric, C. Topal, M. L. Turner, and A. B. Ziad, A real-time, scalable, fast and resource-efficient decoder for a quantum computer, Nat. Electron. **8**, 84 (2025).

[47] M. Cain, C. Zhao, H. Zhou, N. Meister, J. P. B. Ataides, A. Jaffe, D. Bluvstein, and M. D. Lukin, Correlated decoding of logical algorithms with transversal gates, Phys. Rev. Lett. **133**, 240602 (2024).

[48] H. Zhou, C. Zhao, M. Cain, D. Bluvstein, N. Maskara, C. Duckering, H.-Y. Hu, S.-T. Wang, A. Kubica, and M. D. Lukin, Low-overhead transversal fault tolerance for universal quantum computation, Nature (London) **646**, 303 (2025).

[49] M. Cain, D. Bluvstein, C. Zhao, S. Gu, N. Maskara, M. Kalinowski, A. A. Geim, A. Kubica, M. D. Lukin, and H. Zhou, Fast correlated decoding of transversal logical algorithms, arXiv:2505.13587.

[50] M. Serra-Peralta, M. H. Shaw, and B. M. Terhal, Decoding across transversal Clifford gates in the surface code, PRX Quantum **7**, 010335 (2026).

[51] M. L. Turner, E. T. Campbell, O. Crawford, N. I. Gillespie, and J. Camps, Scalable decoding protocols for fast transversal logic in the surface code, PRX Quantum **7**, 010320 (2026).

[52] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, Modeling relational data with graph convolutional networks, *The Semantic Web (ESWC 2018)*, edited by A. Gangemi, *et al.*, Lecture Notes in Computer Science, Vol. 10843 (Springer, Cham, 2018), pp. 593–607.

[53] G. Baranes, M. Cain, J. P. Bonilla Ataides, D. Bluvstein, J. Sinclair, V. Vuletic, H. Zhou, and M. D. Lukin, Leveraging qubit loss detection in fault-tolerant quantum algorithms, Phys. Rev. X **16**, 011002 (2026).

[54] Y. Wu, S. Kolkowitz, S. Puri, and J. D. Thompson, Erasure conversion for fault-tolerant quantum computing in alkaline earth Rydberg atom arrays, Nat. Commun. **13**, 4657 (2022).

[55] S. Ma, G. Liu, P. Peng, B. Zhang, S. Jandura, J. Claes, A. P. Burgers, G. Pupillo, S. Puri, and J. D. Thompson, High-fidelity gates and mid-circuit erasure conversion in an atomic qubit, Nature (London) **622**, 279 (2023).

[56] P. Scholl, A. L. Shaw, R. B.-S. Tsai, R. Finkelstein, J. Choi, and M. Endres, Erasure conversion in a high-fidelity Rydberg quantum simulator, Nature (London) **622**, 273 (2023).

[57] A. Kubica, A. Haim, Y. Vaknin, H. Levine, F. Brandão, and A. Retzker, Erasure qubits: Overcoming the T_1 limit in superconducting circuits, Phys. Rev. X **13**, 041022 (2023).

[58] A. Steane, Multiple-particle interference and quantum error correction, Proc. R. Soc. London, Ser. A **452**, 2551 (1996).

[59] A. Gu, J. P. B. Ataides, M. D. Lukin, and S. F. Yelin, Scalable neural decoders for practical fault-tolerant quantum computation, arXiv:2604.08358.

[60] N. P. Breuckmann and J. N. Eberhardt, Quantum low-density parity-check codes, PRX Quantum **2**, 040101 (2021).

[61] Q. Xu, J. P. B. Ataides, C. A. Pattison, N. Raveendran, D. Bluvstein, J. Wurtz, B. Vasić, M. D. Lukin, L. Jiang, and H. Zhou, Constant-overhead fault-tolerant quantum computation with reconfigurable atom arrays, Nat. Phys. **20**, 1084 (2024).

[62] Q. Xu, H. Zhou, G. Zheng, D. Bluvstein, J. P. B. Ataides, M. D. Lukin, and L. Jiang, Fast and parallelizable logical computation with homological product codes, Phys. Rev. X **15**, 021065 (2025).

[63] J. P. Bonilla Ataides, H. Zhou, Q. Xu, G. Baranes, B. Li, M. D. Lukin, and L. Jiang, Constant-overhead fault-tolerant Bell-pair distillation using high-rate codes, Phys. Rev. Lett. **135**, 130804 (2025).

[64] Y.-H. Liu and D. Poulin, Neural belief-propagation decoders for quantum error-correcting codes, Phys. Rev. Lett. **122**, 200501 (2019).

[65] A. S. Maan and A. Paler, Machine learning message-passing for the scalable decoding of QLDPC codes, npj Quantum Inf. **11**, 78 (2025).

[66] V. Ninkovic, O. Kundacina, D. Vukobratovic, C. Häger, and A. Graell i Amat, Decoding quantum LDPC codes using graph neural networks, arXiv:2408.05170.

[67] A. Gong, S. Cammerer, and J. M. Renes, Graph neural networks for enhanced decoding of quantum LDPC codes, in *2024 IEEE International Symposium on Information Theory (ISIT)* (IEEE, Piscataway, NJ, 2024), pp. 2700–2705.

[68] G. Hu, W. Ouyang, C.-Y. Lu, C. Lin, and H.-S. Zhong, Efficient and universal neural-network decoder for stabilizer-based quantum error correction, arXiv:2502.19971.

[69] J. Blue, H. Avlani, Z. He, L. Ziyin, and I. L. Chuang, Machine learning decoding of circuit-level noise for bivariate bicycle codes, arXiv:2504.13043.

[70] A. Defazio, X. A. Yang, H. Mehta, K. Mishchenko, A. Khaled, and A. Cutkosky, The road less scheduled, in *Advances in Neural Information Processing Systems 37, Vancouver, Canada* (Curran Associates, Inc., 2024), pp. 9974–10007.

[71] J. P. B. Ataides, A. Gu, S. Yelin, and M. Lukin, Neural decoders for universal quantum algorithms, Zenodo, 2026, https://doi.org/10.5281/zenodo.19625192.

[72] C. Chamberland, A. Kubica, T. J. Yoder, and G. Zhu, Triangular color codes on trivalent graphs with flag qubits, New J. Phys. **22**, 023019 (2020).

[73] C. Gidney and C. Jones, New circuits and an open source decoder for the color code, arXiv:2312.08813.

[74] S. Koutsioumpas, T. Noszko, H. Sayginel, M. Webster, and J. Roffe, Colour codes reach surface code performance using vibe decoding, arXiv:2508.15743.

[75] S. Bravyi, A. W. Cross, J. M. Gambetta, D. Maslov, P. Rall, and T. J. Yoder, High-threshold and low-overhead fault-tolerant quantum memory, Nature (London) **627**, 778 (2024).