Joo *EPJ Quantum Technology* (2026) 13:52
https://doi.org/10.1140/epjqt/s40507-026-00492-0

EPJ.org

EPJ Quantum Technology

RESEARCH Open Access

# Decoding the surface code with a spatio-temporal transformer

Robert Joo[1*]

*Correspondence: sjoo@exeter.edu
[1]Phillips Exeter Academy, 20 Main St., Exeter, NH, 03833, USA

**Abstract**

Quantum error correction is a major area of research for building fault-tolerant quantum computers. Recently, machine learning has emerged as a compelling approach to quantum error decoding because of its flexibility and high performance compared to classical methods. In this work, we introduce a spatio-temporal transformer with graph Laplacian positional encodings and factorized latent attention to efficiently deal with the topological structure of the code and the temporal correlations across measurement rounds. We perform experiments with simulated data of the surface code for small distances and demonstrate the potential for our model.

**Keywords:** Quantum error correction; Surface code; Geometric deep learning; Neural decoders

## 1 Introduction

Quantum computing has been attracting more interest over the years as the next computational paradigm. Harnessing the power of superposition and entanglement, quantum computers promise many future use cases in a wide domain of areas, from cryptography [47], database searching [27], combinatorial optimization [19], and molecular dynamics simulations [4] among many others. Arguably, even with the current version of quantum computers, experiments that advance the frontier of science have been performed [3].

However, despite the remarkable progress on the hardware front, Quantum Error Correction (QEC) is indispensable for realizing large-scale, fault-tolerant quantum computation. For quantum computers to have real-world usage in running algorithms like Shor's Algorithm, we need to ideally reach an error rate of about one error per ten billion operations [25]. Unfortunately, physical qubits are prone to decoherence, gate errors, and measurement errors, all of which can accumulate over time and change the state of the qubit. Similar to classical error correction, quantum error correction schemes utilize multiple physical qubits to represent one logical qubit, encoding the information repetitively [45].

Although many error correction schemes exist, much attention has been devoted in the research community to the surface code. Recent physical experiments have demonstrated their promise and scalability [1, 36, 54]. By arranging qubits on a 2D lattice and repeatedly



Springer

measuring its stabilizer operators, one can detect the presence of errors without direct measurement. The result is a set of syndromes that indicate where errors have likely occurred. The decoding task translates to interpreting these syndromes and proposing a correction that, with high probability, restores the original logical state.

While classical algorithms such as the Minimum-Weight Perfect Matching (MWPM) method have seen some success, they face various challenges, with some taking too long for larger distances and others being too inaccurate [20]. In contrast, with the rise of the field of artificial intelligence, machine learning methods are being increasingly applied to quantum decoding, promising improvements in speed, accuracy, and offering flexibility to take into account additional data and model non-ideal situations.

The decoder landscape is crowded with convolutional models [5, 8, 23, 41], reinforcement learners [2, 40], energy-based approaches [51], and sequence models based on transformers [13, 49, 53]. In this work we introduce a *spatio-temporal transformer* that treats the measurement record as a graph embedded in space-time. Two design choices are key: a *factorized attention* scheme that alternates temporal and spatial mixing and a *supra-Laplacian positional encoding* that injects topological and temporal structure into the token geometry. We train and test our model on the surface code in the memory-decoding setting, showing that it matches or exceeds the performance of classical decoders while demonstrating competitive parameter efficiency compared to other machine-learning-based approaches.

## 2 Preliminary background

### 2.1 The stabilizer formalism

In a two-state quantum computer, each qubit $|\psi\rangle$ is a superposition of $|0\rangle$ and $|1\rangle$, represented as $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, where $|\alpha|^2 + |\beta|^2 = 1$. While classical bit noise reduces down to bit flips, quantum noise must be described by means of Pauli channels. The goal in quantum error correction is to represent $k$ logical qubits with $n$ physical but informationally redundant qubits.

Before any measurements are made, the state of $n$ physical qubits is described by a complex vector $|\psi\rangle$ within a $2^n$-dimensional Hilbert space. In the stabilizer formalism, we have $n-k$ independent generators, each of which projects $|\psi\rangle$ onto one of the $2^k$ subspaces of dimension $2^{n-k}$. To use the language of group theory, the stabilizer code is constructed using an Abelian subgroup $S \subset G_n$ with a minimal representation in terms of $n-k$ linearly independent generators:

$$\{g_1, \ldots, g_{n-k} \mid g_i \in S\}.$$

The simultaneous +1-eigenspace of these operators constitutes the codespace,

$$\mathcal{T}(S) = \left\{|\bar{\psi}\rangle \in \mathcal{H}_2^{\otimes n} \mid M\,|\bar{\psi}\rangle = |\bar{\psi}\rangle\,,\ \forall M \in S\right\}.$$

The *distance* $d$ is the minimum weight of such a logical operator, and the code is summarized as $[\![n, k, d]\!]$.

Over the years, many researchers have come up with new error correction schemes based on this stabilizer formalism. Examples include the $[[5, 1, 3]]$ code, the smallest distance-three construction saturating the quantum Hamming bound [6, 37]; Steane's

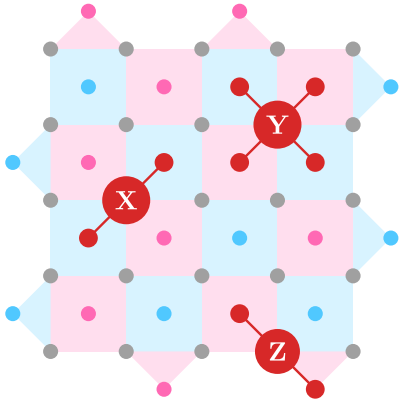


**Figure 1** Distance-five surface code with a representative syndrome pattern. Data qubits (gray) are arranged on the vertices of a square lattice. Blue and pink plaquettes correspond to $Z$- and $X$-type stabilizer checks, respectively, while the red markers denote example Pauli errors ($X$, $Y$, and $Z$) acting on specific data qubits. In the surface code, an $X$ error anticommutes with neighboring $Z$-type stabilizers, flipping their measurement outcomes and producing detectable syndrome changes, whereas it commutes with $X$-type stabilizers and thus leaves their outcomes unchanged. The stabilizer violations visible in adjacent plaquettes therefore indicate the resulting syndrome measurements

CSS $[[7,1,3]]$ code, whose dual classical structure enables transversal Clifford operations [12, 48]; and Shor's pioneering $[[9,1,3]]$ concatenated scheme that introduced fault-tolerant syndrome extraction [26, 46]. Recent hypergraph-product and other homological LDPC (Low Density Parity Check) codes maintain constant stabilizer weight yet achieve distance scaling as $n^{1/2}$, signaling a promising route toward finite-rate, high-threshold quantum memories [11, 22, 50].

### 2.2 Rotated surface codes

The surface code is a $[[O(d^2),1,d]]$ stabilizer code based on Kitaev's topological stabilizer framework (toric code) [35]. It has been one of the most promising candidates for error correction due to its locality and resilience to errors. Among various types of surface codes, the rotated surface code is especially significant because of its simpler boundary conditions and efficient use of physical qubits.

In a rotated surface code, physical qubits (data qubits) are arranged on the vertices of a rotated $d \times d$ square lattice embedded on a two-dimensional surface. Stabilizers in this setup are defined based on measurement of operators acting locally on the qubits at the vertices of each face. Each square corresponds to a correlated measurement of the stabilizer operator $S_\alpha = \sigma_\alpha^a \otimes \sigma_\alpha^b \otimes \sigma_\alpha^c \otimes \sigma_\alpha^d$ where $\alpha$ alternates between $X$-checks and $Z$-checks in a checkerboard pattern. It is easy to see that all measurements commute as the $X$-checks and $Z$-checks alternate and these checks share two corners or none. The physical implementation of the surface code means there would be ancilla qubits at the center of each face and at certain boundary points, bringing the total number of qubits to $d^2 + (d^2 - 1)$. An example distance-five surface code with a sample error syndrome is illustrated in Fig. 1.

Since the measurement operators commute, repeated parity check measurements do not affect or entangle the encoded state within this space. As time $t$ progresses, errors will occur and the syndrome evolves, producing a discrete trajectory of syndrome configurations $s(t)$. Since our goal is to preserve the encoded logical state rather than the physical state of each qubit, it suffices to determine whether an error has occurred or not. Rather than applying corrections after every measurement cycle, a more efficient strategy is to perform the parity checks continuously but defer the actual recovery operation for several rounds. This reduces the frequency of active error correction while maintaining full information about the accumulated syndromes. Thus, the task of our decoder boils down

to outputting whether or not a logical error occurred given $r$ rounds of measurements of the detectors.

Thanks to the surge of experimental demonstrations of the surface code on real quantum devices ranging from superconducting qubit platforms [1, 36, 54] to trapped-ion and neutral atom arrays, this error correction architecture is among the most studied and experimentally validated. As a result, faster and more efficient ways to decode the surface code have been researched by many.

### 2.3 Surface code decoders

Classical decoders for the surface code have historically been more extensively studied: the long-standing *minimum-weight perfect-matching* (MWPM) algorithm [29, 31, 43], nearly linear-time *Union–Find* variants [16, 30], hierarchical *renormalization-group* (RG) schemes [17], iterative *belief-propagation* message passing on sparse factor graphs [14, 42], and likelihood-exact but memory-hungry *tensor-network contraction* methods [9, 44]. However, these approaches share two key shortcomings: those that achieve near-optimal thresholds incur prohibitive runtime as the code distance grows, while those designed for speed sacrifice performance. Perhaps most importantly, these deterministic algorithms do not have the flexibility to take into account leakage or hardware-specific information that we get from running quantum computers.

In recent years, machine learning (ML) has emerged as a compelling alternative to these algorithms [2, 5, 8, 13, 23, 38–41, 49, 51, 53]. ML decoders can learn and adapt to different noise models directly from data, making them naturally more robust. Early efforts to decode surface code have mainly relied on convolutional neural networks [5, 8, 23, 41] that demonstrated strong performance by exploiting the lattice symmetries of surface code. More recently, transformer-based architectures have gained attention. Works such as [13, 53] adapt the transformer framework to quantum error correction, while [5] introduces a hybrid recurring convolutional transformer that merges convolutional inductive bias with global self-attention. Graph-based approaches have also emerged as a natural representation of the code's structure, with [38] demonstrating the promise of graph machine learning for data-driven decoding.

## 3 Model architecture

We present our transformer model that treats the surface code's stabilizer syndrome as a spatio-temporal graph. At a high level, our design is heavily influenced by ideas from graph machine learning [10, 33] and video transformers [7, 52]. We utilized spectral/supra-Laplacian positional features as described in [18, 32] and separate attention for space and time as introduced by [7].

### 3.1 Positional encoding

The Supra-Laplacian encoding we employed is largely motivated by interpreting all the measurements across time and space of the surface code as one big spatio-temporal graph. We first build a spatial graph where neighboring parity-check operators are connected by weighted edges. Distinct weights are assigned to $(XX, ZZ)$ pairs and $(XZ)$ pairs to reflect their differing physical correlations and the number of data qubits they share as shown in Fig. 2. This is a natural way of conceptualizing the surface code, as neighboring checks share data qubits. We can then also add time edges between consecutive rounds of the same detection qubit, thereby forming a spatio-temporal graph as illustrated in Fig. 3.

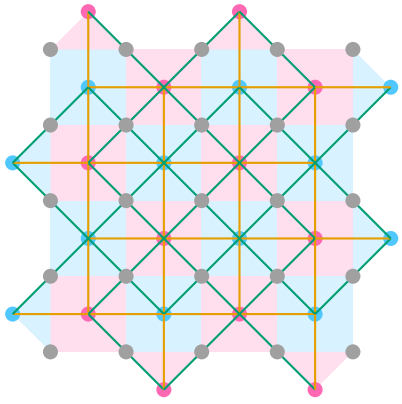

**Figure 2** Spatial graph of the surface code showing weighted edges between neighboring stabilizers. Orange edges (——) represent *XZ* couplings that share two data qubits, while green edges (——) denote *XX* or *ZZ* couplings that share one data qubit

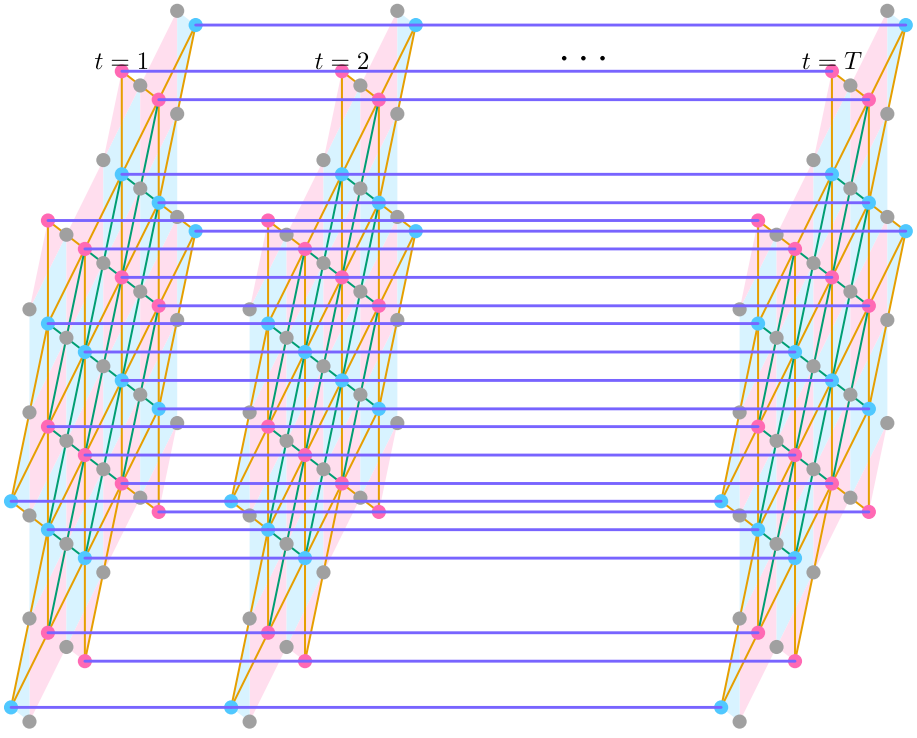


**Figure 3** Spatio-temporal graph representation of the surface code. Each 2D lattice slice represents a single round of stabilizer measurements, with orange and green edges denoting spatial couplings and purple edges representing temporal connections between identical detectors across rounds

Consider the weighted symmetric adjacency matrix $\mathbf{W} \in \mathbb{R}^{L\times L}$, where $L$ is the total number of detector nodes across all rounds. Temporal edges are assigned between identical nodes measured in consecutive rounds. Within each round, spatial couplings reflect the geometry of the code. We distinguished direct stabilizer neighbors from the diagonally offset stabilizers as well as from the temporal edges by assigning a different weight for each type of edge. Letting $\mathbf{A} \in \mathbb{R}^{n\times n}$ denote the spatial adjacency of one round and $r$ the

number of rounds, we obtain the global adjacency matrix

$$\mathbf{W} = \alpha \times \begin{bmatrix} \mathbf{A} & \mathbf{I} & 0 & \cdots & 0 \\ \mathbf{I} & \mathbf{A} & \mathbf{I} & \cdots & 0 \\ 0 & \mathbf{I} & \mathbf{A} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \mathbf{I} \\ 0 & 0 & \cdots & \mathbf{I} & \mathbf{A} \end{bmatrix},$$

where $\mathbf{I} \in \mathbb{R}^{n \times n}$ is the identity matrix representing temporal connections across consecutive rounds and $\alpha$ is some scaling parameter. From this adjacency matrix the degree matrix is defined as $\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_L)$, where $d_i = \sum_j \mathbf{W}_{ij}$. The symmetric normalized Laplacian is then

$$\mathbf{L}_{\mathrm{sym}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}.$$

The spectral features of $\mathbf{L}_{\mathrm{sym}}$ are well known in spectral graph theory [15] to capture important information about graphs. The eigenvectors of the graph Laplacian form a natural basis that captures coherent, multi-scale structure over the graph topology. We compute its non-trivial eigenvalues $\lambda_1 \leq \cdots \leq \lambda_k$ and their eigenvectors $\phi_1, \ldots, \phi_k$. The value of $\phi_j$ at node $u$ encodes the embedding of that detector. In practice, these eigenvectors are stacked into a feature matrix $\mathbf{U} \in \mathbb{R}^{L \times k}$, normalized column-wise and sign-resolved so that $\phi_j$ is consistent across runs.

We then project the raw spectral features $\mathbf{U}$ onto the transformer embedding space. Letting $d_{\mathrm{model}}$ be the model dimension, the projection is a learnable linear map $\mathrm{proj} : \mathbb{R}^k \to \mathbb{R}^{d_{\mathrm{model}}}$, initialized so that the first $\min(k, d_{\mathrm{model}})$ output coordinates are identity-mapped, while any excess dimensions ($d_{\mathrm{model}} > k$) are filled with random Kaiming-uniform weights [28].

### 3.2 Factorized latent attention

Self-attention over a full sequence scales quadratically with sequence length. For decoding with $\nu$ detectors observed across $r$ rounds, this would mean $O((\nu r)^2)$ operations, which can be impractical for real-time decoding where inference must complete within millisecond-scale coherence times. In addition, being fully connected may not be computationally efficient as certain qubits separated by time and space need not be communicating with each other. To address this, we factorize attention into spatial and temporal components. This decomposition substantially reduces the computational cost while retaining much of its expressivity.

Let $\mathbf{H}^{(\ell-1)} \in \mathbb{R}^{(\nu r) \times d_{\mathrm{model}}}$ denote the input to layer $\ell$, reshaped into $(r, \nu, d_{\mathrm{model}})$, where $r$ is the number of temporal rounds and $\nu$ the number of detector measurements per round. First, for each temporal slice, we perform attention across all $\nu$ detectors for every time step independently in parallel. Following spatial mixing, the tensor is permuted so that now we deal with each detector's measurements across time independently and in parallel. Temporal attention then operates along this axis across the $r$ rounds of each detector. Lastly, we have a Multilayer Perceptron at the end that takes care of global mixing. Each sublayer is wrapped in a pre-normalized residual block: RMS normalization [55] is applied prior

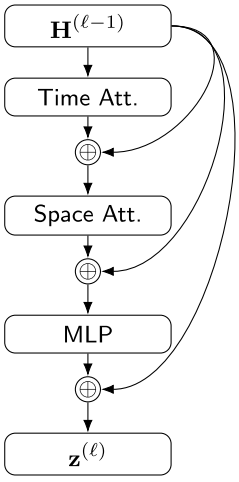


**Figure 4** Illustration of the space-time attention block. Figure taken from [7]

to the transformation, and the output is added back to the sublayer input via a residual connection. An illustration is given in Fig. 4. Mathematically, this can be written as:

$$\mathbf{h}_{\tau,i}^{(\ell-1)} \in \mathbb{R}^{d_{\text{model}}}, \qquad \tau = 1,\dots,r,\ \ i = 1,\dots,v,$$

$$\tilde{\mathbf{h}}_{\tau,i} = \mathbf{h}_{\tau,i}^{(\ell-1)} + \text{Attn}_S\Big(\text{RMSNorm}\big(\{\mathbf{h}_{\tau,j}^{(\ell-1)}\}_{j=1}^{v}\big)\Big)_i,$$

$$\hat{\mathbf{h}}_{\tau,i} = \tilde{\mathbf{h}}_{\tau,i} + \text{Attn}_T\Big(\text{RMSNorm}\big(\{\tilde{\mathbf{h}}_{\tau',i}\}_{\tau'=1}^{r}\big)\Big)_\tau,$$

$$\mathbf{h}_{\tau,i}^{(\ell)} = \hat{\mathbf{h}}_{\tau,i} + \text{MLP}\big(\text{RMSNorm}(\hat{\mathbf{h}}_{\tau,i})\big).$$

This structure reduces total complexity from $O(v^2r^2)$ to $O(v^2r + vr^2)$, while still achieving global information flow through alternating spatial and temporal interactions. To further optimize this model, we can also implement masks for both spatial and temporal attention, which would bring the computational complexity down even further. For temporal attention, we can use a local temporal sliding-window mask where each detector can attend only to the previous $w$ rounds. In the spatial dimension, attention can be restricted to detectors within a fixed geometric neighborhood. Although such masking limits direct interactions, stacking multiple layers allows information to propagate across the full space-time lattice when needed.

## 4 Experimentation and results

### 4.1 Experimental methodology

We trained and tested our model described in the previous section with the data simulated by the Stim library [24]. All experiments were conducted on a single NVIDIA GH200 system equipped with one H100 GPU (96 GB HBM), paired with an ARM64 host CPU (64 vCPUs), 432 GiB of system memory, and 4 TiB of local SSD storage. For our spatio-temporal transformer, we used a detector embedding dimension of 240, a spatial attention radius of 4 lattice units, and a temporal attention window of 3 rounds (additional details in Sect. A).

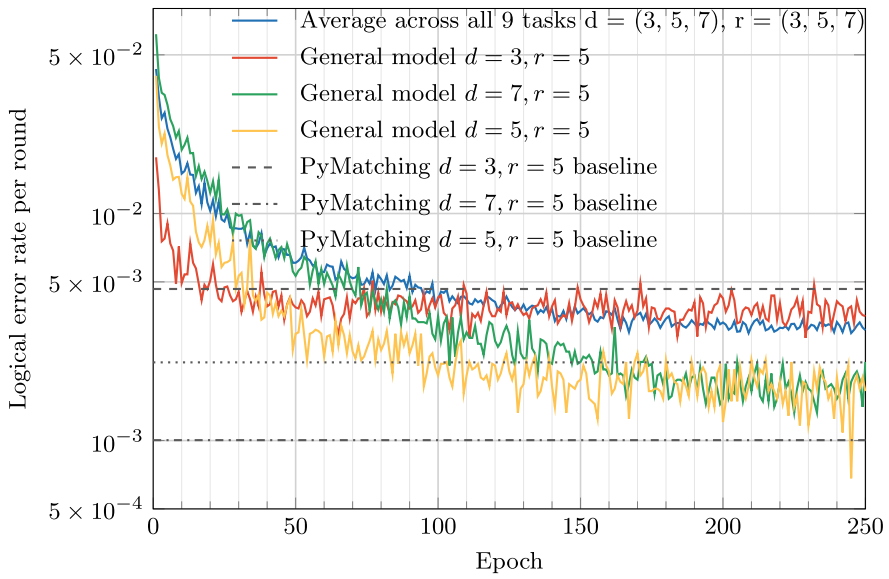


**Figure 5** General Model Training Dynamics at p = 0.005

We first trained a general model on synthetic surface-code syndrome sequences generated with Stim across distances {3, 5, 7}, rounds {3, 5, 7}, and physical error rates {0.002, 0.005, 0.008}. We used circuit-level depolarizing noise, where each Clifford gate is followed by depolarization with probability $p$, each qubit undergoes depolarization before each round with probability $p$, and each measurement outcome is flipped with probability $p$. Each epoch used one million samples uniformly distributed over the 27 configurations. Training ran for 250 epochs using binary cross-entropy as the loss function with the Adam optimizer [34] (learning rate $10^{-4}$ with cosine decay, no weight decay; batch size 512). Validation employed freshly generated data per configuration, and the final checkpoint was selected by the highest validation accuracy averaged across configurations.

We then took the general model and fine-tuned it on various fixed distance-rounds-probability configurations. Starting from the general model, we fine-tuned for roughly 20 epochs (50 epochs for $d = 7$), using the same loss function and AdamW optimizer but with a constant learning rate of $2 \times 10^{-5}$. For $d = 7$, because we are in the extreme accuracy regime, we used a batch size of 2048.

We also compared with other machine learning models: graph neural networks (GNN), fully connected transformers, and 3D convolutional neural networks. All models are approximately 4M parameters and further details are given in Sect. A. We trained each architecture on $d = 5$, $r = 5$, $p = 0.005$ under depolarizing noise, then fine-tuned these models (10 epochs, $10^6$ samples) under three alternative noise models: (1) biased noise with Z-bias ratio $\eta = 10$ (Z errors occur 10 times more frequently than X or Y errors), (2) coherent noise with over-rotation angle $\theta = 0.1$ rad, and (3) spectator qubit noise with correlated error probability $p_{\text{spec}} = 0.01$ on next-nearest-neighbor qubit pairs.

## 4.2 Results

In training the general model, despite limited data per configuration, the model learned a strong shared representation: for $d \in \{3, 5\}$ it generally matched or exceeded PyMatching [29], while performance at $d = 7$ fell narrowly behind the baseline. Attempts to

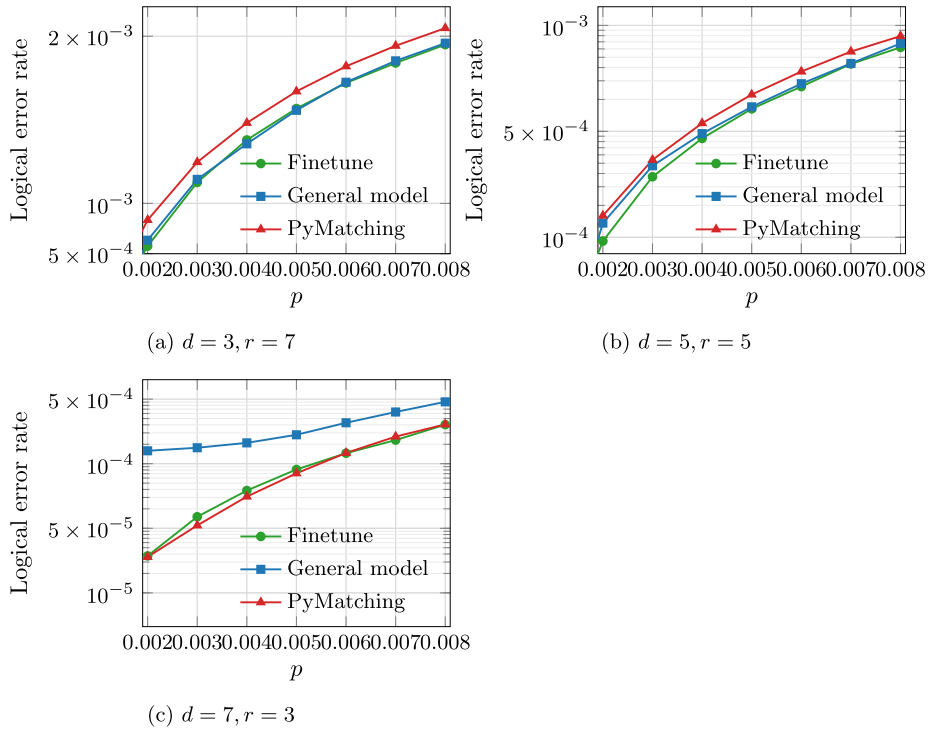


**Figure 6** Logical error rate comparison across different decoding strategies at selected physical error probabilities

reweight the dataset toward larger $d$ and $r$ marginally improved those cases but degraded smaller codes (notably $d = 3$). The overall training dynamics are shown in Fig. 5, where the logical error rate (LER) trajectories over epochs are plotted for the overall model as well as performance for $r = 5$, $d = 3, 5, 7$. The model converged rapidly for smaller distances but plateaued higher for larger codes, indicating that data scarcity and noise scaling limited performance at higher distances where more complexity is demanded.

Fine-tuning the general model for each $(d, r, p)$ configuration resulted in even stronger models. Fig. 6 summarizes these results for $(d, r) = (3, 7), (5, 5), (7, 3)$, $p = 0.002, 0.003, \ldots,$ 0.008. For the smaller distances, fine-tuning resulted in marginal improvement over an already strong general model. For $d = 7$, fine-tuning was essential to reach performance on par with PyMatching. More compute and bigger models seem to be needed as we scale up to bigger and bigger distances to deal with increasing complexity. Because attention in the spatial direction is restricted by a fixed-radius spatial mask, scaling to larger code distances likely requires either more layers or larger receptive fields.

Fig. 7 compares the performance of different machine learning architectures against PyMatching on $d = 5$, $r = 5$, $p = 0.005$ across four noise models. All neural decoders outperform PyMatching under every noise condition tested, with particularly strong gains under biased and coherent noise where the error structure deviates from the neat assumptions underlying minimum-weight perfect matching. Among neural architectures, the 3D CNN achieves the lowest logical error rates, likely benefiting from its inductive symmetry bias. Our spatio-temporal transformer performs comparably, ranking second across most noise models despite using attention mechanisms that do not explicitly encode trans-

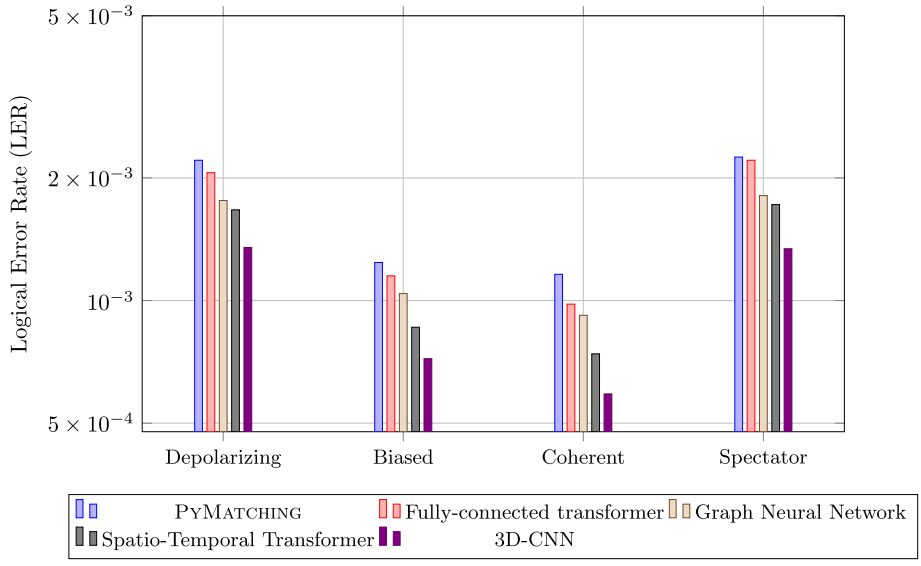


**Figure 7** Logical error rate comparison across different noise models and decoders (d = 5, r = 5, p = 0.005)

lational equivariance. The Graph Neural Network and fully connected transformer trail slightly behind. These results suggest that while the 3D CNN performed best among the models tested, transformer-based models remain competitive and offer much flexibility for adapting to diverse noise environments.

## 5 Discussion and outlook

Our results show that spatio-temporal transformers can serve as efficient high-performing decoders for the surface code. Despite using a relatively compact model, our model matches or exceeds several existing decoding baselines. Our study, however, remains limited to an idealized setting with simplified noise assumptions, and further testing with real-world data seems necessary. Because our model makes no assumptions about symmetry, we believe it will be more robust to localized, clustered, or correlated error patterns commonly observed in quantum computers today.

Future work should extend these results to realistic hardware noise models and variable stabilizer layouts. Two directions appear particularly promising: first, a systematic comparison between recurrent and transformer-based architectures under identical conditions; and second, incorporating known code symmetries directly into the network via equivariant or weight-sharing attention mechanisms. Moreover, while we only tested our model on the surface code, our method could generalize beyond to other formalisms such as color codes or subsystem codes. For codes with irregular geometries or heterogeneous qubit connectivity, position-specific attention mechanisms may offer advantages over architectures that assume uniform spatial structure.

## Appendix A: Model details

For the sake of fairness, we ensured that every model we trained was roughly of comparable size to each other. The parameters of every model we used are summarized in Fig. 8.

**Table A1**: Spatio-temporal Transformer

| Parameter | Value |
|---|---|
| $d_{\text{model}}$ | 240 |
| $n_{\text{layers}}$ | 4 |
| Spatial heads | 8 |
| Temporal heads | 4 |
| FFN dim | 1024 |
| Dropout | 0.1 |
| Temporal window | 3 |
| Spatial Radius | 4 |
| **Parameters** | **4,150,346** |

**Table A2**: 3D Convolutional Neural Network

| Stage | Layer | Spec |
|---|---|---|
| Conv | Conv3d | $1 \to 64$, $k$=5 |
| | Conv3d | $64 \to 128$, $k$=3 |
| | Conv3d | $128 \to 256$, $k$=3 |
| | Conv3d | $256 \to 256$, $k$=3 |
| | Conv3d | $256 \to 128$, $k$=3 |
| | Conv3d | $128 \to 64$, $k$=3 |
| Pooling | AdaptiveAvgPool3d | |
| MLP | Linear | $64 \to 256 \to 1$ |
| **Parameters** | | **4,007,105** |

**Table A3**: Graph Neural Network

| Parameter | Value |
|---|---|
| GraphConv layers | 6 |
| GraphConv widths | 640 |
| MLP head | [512, 256] |
| Node features | 5 |
| Dropout | 0.1 |
| Global pooling | Mean over nodes |
| **Parameters** | **4,573,697** |

**Table A4**: Fully connected Transformer

| Parameter | Value |
|---|---|
| $d_{\text{model}}$ | 272 |
| $n_{\text{layers}}$ | 5 |
| $n_{\text{heads}}$ | 8 |
| FFN dim | 1152 |
| Dropout | 0.1 |
| Positional encoding | Sinusoidal |
| **Parameters** | **4,885,713** |

**Figure 8** Architectural hyperparameters and total trainable parameter counts for all decoder baselines evaluated in this work.

## Appendix B: Latency analysis

Decoding latency is a key practical consideration when selecting a decoder. All timings here should be taken with a grain of salt. They fluctuate depending on hardware (CPU/GPU model, clocks, memory bandwidth), software stack (PyTorch/CUDA versions), and implementation details (e.g., fused attention kernels, graph optimizations). Our benchmarks use straightforward implementations with no custom kernels and are performed on an H100. In principle, both neural and classical decoders could be substantially accelerated with optimized kernels and/or dedicated hardware (e.g., FPGAs/ASICs).

Fig. 9 compares the runtime of a fully connected transformer with our spatio-temporal transformer, matched in embedding dimension and number of layers. For the spatio-temporal model, the measured decoding time at small numbers of rounds is dominated by fixed implementation overheads (e.g., kernel launches and memory allocation), but these costs are quickly amortized as the number of rounds increases. In contrast, the fully connected transformer initially exhibits similar amortization, but beyond a modest number of rounds the quadratic cost of global attention dominates, causing the per-round latency to increase with $r$.

Fig. 10 shows that while PyMatching is faster per round than our spatio-temporal decoder in the regimes tested, its per-round latency increases with code distance and physical error rate while our decoder's per-round latency is comparatively stable. The com-

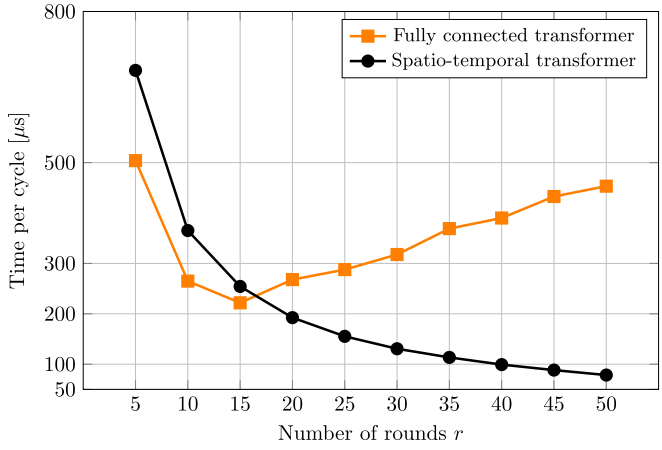


**Figure 9** Comparison of per-cycle decoding latency for a fully connected transformer and our spatio-temporal transformer at $d = 11$ (batch = 1)

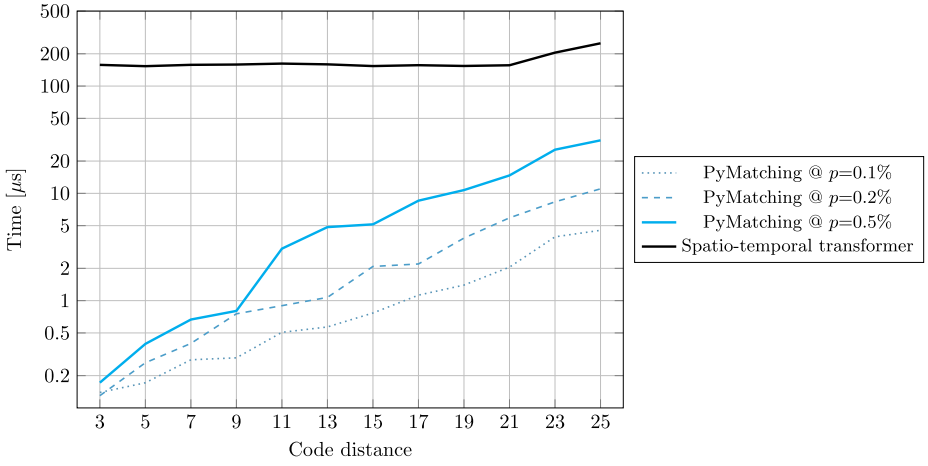


**Figure 10** Per-cycle decoding latency versus code distance for our spatio-temporal transformer and PyMatching at $r = 25$. Our model's latency is largely insensitive to the physical error rate, while PyMatching becomes slower at higher physical error rates as increased syndrome density leads to more work per decode

parisons in Fig. 10 should be interpreted as indicative scaling trends rather than a strict apples-to-apples contest. PyMatching is a highly optimized CPU-oriented implementation, whereas our decoder runs on GPUs that are not as optimized.

## Appendix C: Model ablations

### C.1 Positional encoding variants

In addition to the Supra-Laplacian positional encoding described in the main text, we implemented and evaluated several alternative schemes for comparison. First, we extended sinusoidal positional encoding to three dimensions (3D). Each detection event in the surface code is uniquely identified by $(x, y, t)$, where $(x, y)$ denotes the qubit lattice position and $t$ the round in which the detection occurred. We took the sine and cosine functions of those values at various frequencies to obtain our embedding. We also experimented

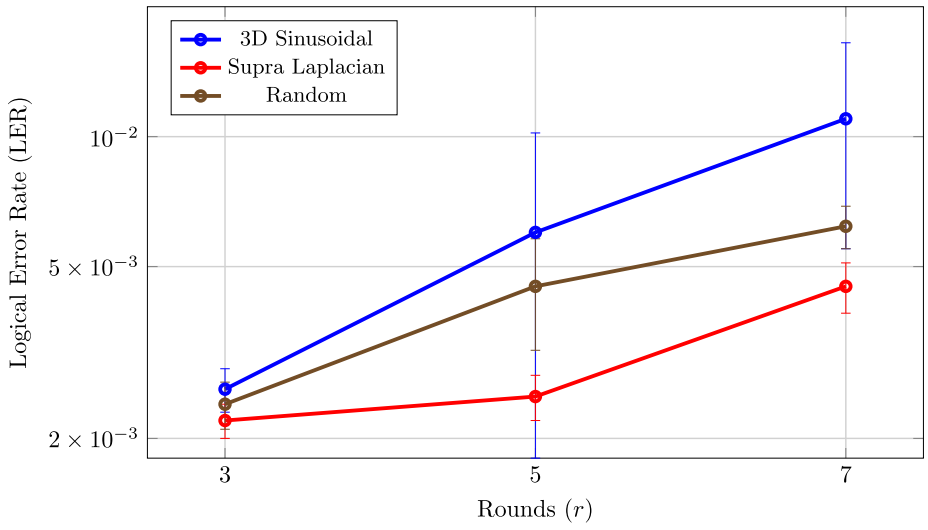


**Figure 11** Logical error rate comparison of decoder with different positional encodings ($d = 5, p = 0.005$)

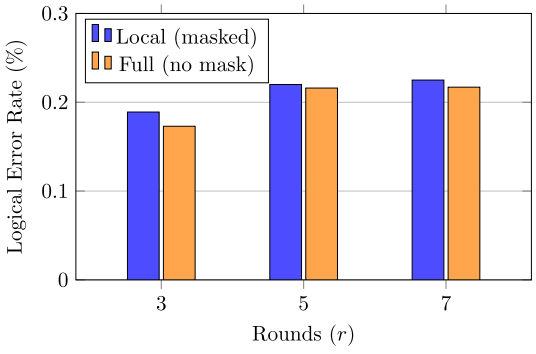


**Figure 12** Logical error rate of masked temporal and spatial attention vs. full temporal and spatial attention ($d = 5, p = 0.005$)

with random positional encodings, where each detection was assigned a stationary random vector in $\mathbb{R}^{d_{\text{model}}}$.

We conducted three trial runs for $d \in \{3,5,7\}$ and $r \in \{3,5,7\}$ using a fully connected transformer with 6 layers and $d_{\text{model}} = 256$. We present results for $d = 5$ below which are representative of $d = 3$ and $d = 7$ as well.

As Fig. 11 demonstrates, our Supra-Laplacian positional encoding performs far better than sinusoidal encoding and random encoding. Interestingly, for larger distances and rounds, sinusoidal encodings degrade more severely, while random encodings remain robust likely due to the near orthogonality of random vectors.

## C.2 Attention masks

To evaluate the contribution of the locality-inducing attention masks in our spatio-temporal transformer architecture, we conducted an ablation study comparing the masked model against a variant with full attention in both spatial and temporal dimensions. Both models were trained from scratch for 30 epochs on 1M samples per epoch at $d = 5$ and $p = 0.005$ across $r \in \{3,5,7\}$. As shown in Fig. 12, removing the spatial locality

mask (radius $r_s = 4$) and temporal sliding window (size $w = 3$) yields only marginal improvements in validation accuracy: +0.047% at $r = 3$, +0.017% at $r = 5$, and +0.054% at $r = 7$. These minimal improvements indicate that global attention provides only marginal benefits, suggesting that increasing embedding dimensionality may be a more effective use of model capacity.

**Author contributions**
This work was conducted by the sole author.

**Funding information**
Self-funded study.

**Data Availability**
The code implementing our decoder is available at https://github.com/sapient-sapiens/ST-Decoder.

## Declarations

**Ethics approval and consent to participate**
Not applicable; the study only uses simulated data and did not require institutional review.

**Competing interests**
The author declares no competing interests.

Received: 12 November 2025 Accepted: 4 March 2026 Published online: 23 March 2026

**References**

1. Google Quantum AI. Suppressing quantum errors by scaling a surface code logical qubit. Nature. 2023;614:676–81. https://doi.org/10.1038/s41586-022-05434-1.
2. Andreasson P, Johansson J, Liljestrand S, Granath M. Quantum error correction for the toric code using deep reinforcement learning. Quantum. 2019;3:183. https://doi.org/10.22331/q-2019-09-02-183.
3. Arute F, et al. Quantum supremacy using a programmable superconducting processor. Nature. 2019;574:505–10. https://doi.org/10.1038/s41586-019-1666-5.
4. Aspuru-Guzik A, Dutoi AD, Love PJ, Head-Gordon M. Simulated quantum computation of molecular energies. Science. 2005;309(5741):1704–7. https://doi.org/10.1126/science.1113479.
5. Bausch J, Senior AW, Heras FJH, et al. Learning high-accuracy error decoding for quantum processors. Nature. 2024;635:834–40. https://doi.org/10.1038/s41586-024-08148-8.
6. Bennett CH, DiVincenzo DP, Smolin JA, Wootters WK. Mixed-state entanglement and quantum error correction. Phys Rev A. 1996;54(5):3824–51. https://doi.org/10.1103/PhysRevA.54.3824.
7. Bertasius G, Wang H, Torresani L. Is space-time attention all you need for video understanding? In: Proceedings of the 38th international conference on machine learning. Proceedings of machine learning research. vol. 139. PMLR; 2021. p. 813–824. https://proceedings.mlr.press/v139/bertasius21a.html.
8. Bordoni S, Giagu S. Convolutional neural network based decoders for surface codes. Quantum Inf Process. 2023;22(3):151. https://doi.org/10.1007/s11128-023-03908-1.
9. Bravyi S, Suchara M, Vargo A. Efficient algorithms for maximum likelihood decoding in the surface code. arXiv preprint. arXiv:1405.4883 (2014).
10. Bronstein MM, Bruna J, LeCun Y, Szlam A, Vandergheynst P. Geometric deep learning: going beyond Euclidean data. IEEE Signal Process Mag. 2017;34(4):18–42. https://doi.org/10.1109/MSP.2017.2693418.
11. Breuckmann NP, Eberhardt JN. Quantum low-density parity-check codes. PRX Quantum. 2021;2(4):040101. https://doi.org/10.1103/PRXQuantum.2.040101.
12. Calderbank AR, Shor PW. Good quantum error-correcting codes exist. Phys Rev A. 1996;54(2):1098–105. https://doi.org/10.1103/PhysRevA.54.1098.
13. Cao H, Pan F, Wang Y, Zhang P. qecgpt: decoding quantum error-correcting codes with generative pre-trained transformers. arXiv preprint. arXiv:2307.09025 (2023).
14. Chen J, Yi Z, Liang Z, Wang X. Improved belief propagation decoding algorithms for surface codes. arXiv preprint. arXiv:2407.11523 (2024).
15. Chung FRK. Spectral Graph Theory. CBMS regional conference series in mathematics. vol. 92. American Mathematical Society; 1997.
16. Delfosse N, Nickerson NH. Almost-linear time decoding algorithm for topological codes. Quantum. 2021;5:595. https://doi.org/10.22331/q-2021-12-02-595.
17. Duclos-Cianci G, Poulin D. A renormalization group decoding algorithm for topological quantum codes. arXiv preprint. arXiv:1006.1362 (2010).
18. Dwivedi VP, Bresson X. A generalization of transformer networks to graphs. arXiv preprint. arXiv:2012.09699 (2020). https://doi.org/10.48550/arXiv.2012.09699.
19. Farhi E, Goldstone J, Gutmann S. A quantum approximate optimization algorithm. arXiv preprint. arXiv:1411.4028 (2014).
20. Fowler AG. Minimum weight perfect matching of fault-tolerant topological quantum error correction in average $O(1)$ parallel time. Quantum Inf Comput. 2015;15(1–2):145–58.
21. Fowler AG, Mariantoni M, Martinis JM, Cleland AN. Surface codes: towards practical large-scale quantum computation. Phys Rev A. 2012;86(3):032324. https://doi.org/10.1103/PhysRevA.86.032324.

22. Freedman M, Meyer DA, Luo F. $\mathbb{Z}_2$–systolic freedom and quantum codes. In: Mathematics of quantum computation. 2002. p. 287–320.
23. Gicev S, Hollenberg LC, Usman M. Fully convolutional 3d neural network decoders for surface codes with syndrome circuit noise. arXiv preprint. arXiv:2506.16113 (2025).
24. Gidney C. Stim: a fast stabilizer circuit simulator. Quantum. 2021;5:497. https://doi.org/10.22331/q-2021-07-06-497.
25. Gidney C, Ekerå M. How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits. Quantum. 2021;5:433. https://doi.org/10.22331/q-2021-04-15-433.
26. Gottesman D. Stabilizer codes and quantum error correction. arXiv preprint. arXiv:quant-ph/9705052 (1997).
27. Grover LK. A fast quantum mechanical algorithm for database search. In: Proceedings of the twenty-eighth annual ACM symposium on theory of computing (STOC '96). 1996. p. 212–9. https://doi.org/10.1145/237814.237866.
28. He K, Zhang X, Ren S, Sun J. Delving deep into rectifiers: surpassing human-level performance on ImageNet classification. In: Proceedings of the IEEE international conference on computer vision (ICCV). 2015. p. 1026–34. https://doi.org/10.1109/ICCV.2015.123.
29. Higgott O, Gidney C. Sparse Blossom: correcting a million errors per core second with minimum-weight matching. Quantum. 2025;9:1600. https://doi.org/10.22331/q-2025-01-20-1600.
30. Huang S, Newman M, Brown KR. Fault-tolerant weighted union-find decoding on the toric code. Phys Rev A. 2020;102(1):012419. https://doi.org/10.1103/PhysRevA.102.012419.
31. deMarti iOlius A, Etxezarreta Martinez J, Fuentes P, Crespo PM. Performance enhancement of surface codes via recursive mwpm decoding. Phys Rev A. 2023;108(2):022401. https://doi.org/10.1103/PhysRevA.108.022401.
32. Karmim Y, Lafon M, Fournier-S'niehotta R, Thome N. Supra-Laplacian encoding for transformer on dynamic graphs. Adv Neural Inf Process Syst. 2024;37:17215–46.
33. Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. In: 5th International Conference on Learning Representations (ICLR) (2017). https://doi.org/10.48550/arXiv.1609.02907.
34. Kingma DP, Ba J. Adam: a method for stochastic optimization. In: 3rd International Conference on Learning Representations (ICLR) (2015). https://doi.org/10.48550/arXiv.1412.6980.
35. Kitaev AY. Fault-tolerant quantum computation by anyons. Ann Phys. 2003;303(1):2–30. https://doi.org/10.1016/S0003-4916(02)00018-0.
36. Krinner S, Lacroix N, Remm A, et al. Realizing repeated quantum error correction in a distance-three surface code. Nature. 2022;605:669–74. https://doi.org/10.1038/s41586-022-04566-8.
37. Laflamme R, Miquel C, Paz JP, Zurek WH. Perfect quantum error correcting code. Phys Rev Lett. 1996;77(1):198–201. https://doi.org/10.1103/PhysRevLett.77.198.
38. Lange M, Havström P, Srivastava B, Bengtsson I, Bergentall V, Hammar K, Heuts O, Nieuwenburg E, Granath M. Data-driven decoding of quantum error correcting codes using graph neural networks. Phys Rev Res. 2025;7(2):023181. https://doi.org/10.1103/PhysRevResearch.7.023181.
39. Meinerz K, Park C-Y, Trebst S. Scalable neural decoder for topological surface codes. Phys Rev Lett. 2022;128(8):080505. https://doi.org/10.1103/PhysRevLett.128.080505.
40. Nautrup HP, Delfosse N, Dunjko V, Briegel HJ, Friis N. Optimizing quantum error correction codes with reinforcement learning. Quantum. 2019;3:215. https://doi.org/10.22331/q-2019-12-16-215.
41. Ni X. Neural network decoders for large-distance 2D toric codes. Quantum. 2020;4:310. https://doi.org/10.22331/q-2020-08-24-310.
42. Old J, Rispler M. Generalized belief propagation algorithms for decoding of surface codes. Quantum. 2023;7:1037. https://doi.org/10.22331/q-2023-06-07-1037.
43. Paler A, Fowler AG. Pipelined correlated minimum weight perfect matching of the surface code. Quantum. 2023;7:1205. https://doi.org/10.22331/q-2023-12-12-1205.
44. Piveteau C, Chubb CT, Renes JM. Tensor-Network Decoding Beyond 2D. PRX Quantum. 2024;5(4):040303. https://doi.org/10.1103/PRXQuantum.5.040303.
45. Shor PW. Scheme for reducing decoherence in quantum computer memory. Phys Rev A. 1995;52:R2493–6. https://doi.org/10.1103/PhysRevA.52.R2493.
46. Shor PW. Fault-tolerant quantum computation. In: Proceedings of the 37th conference on foundations of computer science (FOCS '96). 1996. p. 56–65. https://doi.org/10.1109/SFCS.1996.548464.
47. Shor PW. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J Comput. 1997;26(5):1484–509. https://doi.org/10.1137/S0097539795293172.
48. Steane AM. Error correcting codes in quantum theory. Phys Rev Lett. 1996;77(5):793–7. https://doi.org/10.1103/PhysRevLett.77.793.
49. Tian C, Fan Z, Guo X, Song X, Tian Y. Transformer-based quantum error decoding enhanced by QGANs: towards scalable surface code correction algorithms. EPJ Quantum Technol. 2025;12(1):76. https://doi.org/10.1140/epjqt/s40507-025-00383-w.
50. Tillich J-P, Zémor G. Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength. IEEE Trans Inf Theory. 2014;60(2):1193–202. https://doi.org/10.1109/TIT.2013.2292061.
51. Torlai G, Melko RG. Neural decoder for topological codes. Phys Rev Lett. 2017;119(3):030501. https://doi.org/10.1103/PhysRevLett.119.030501.
52. Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. In: Advances in neural information processing systems. vol. 30. 2017.
53. Wang H, Liu P, Shao K, Li D, Gu J, Pan DZ, Ding Y, Han S. Transformer-qec: quantum error correction code decoding with transferable transformers. arXiv preprint. arXiv:2311.16082 (2023).
54. Zhao Y, Jin F, Pan X, et al. Realization of an error-correcting surface code with superconducting qubits. Phys Rev Lett. 2022;129(3):030501. https://doi.org/10.1103/PhysRevLett.129.030501.
55. Zhang B, Sennrich R. Root mean square layer normalization. In: Advances in neural information processing systems. vol. 32. 2019.


## Publisher's note