Tian et al. *EPJ Quantum Technology* (2025) 12:76
https://doi.org/10.1140/epjqt/s40507-025-00383-w

EPJ Quantum Technology
a SpringerOpen Journal

EPJ.org

RESEARCH Open Access

Transformer-based quantum error decoding enhanced by QGANs: towards scalable surface code correction algorithms

Cewen Tian[1], Zaixu Fan[1], Xiaoxuan Guo[1], Xinying Song[2] and Yanbing Tian[2*]

*Correspondence:
tianyanbing@qut.edu.cn
[2]School of Science, Qingdao University of Technology, Qingdao, China
Full list of author information is available at the end of the article

Abstract

To address qubits' high environmental sensitivity and reduce the significant error rates in current quantum devices, quantum error correction stands as one of the most dependable approaches. The topological surface code, renowned for its unique qubit lattice structure, is widely considered a pivotal tool for enabling fault-tolerant quantum computation. Through redundancy introduced across multiple qubits, the surface code safeguards quantum information and identifies errors via state changes captured by syndrome qubits. However, simultaneous errors in data and syndrome qubits substantially escalate decoding complexity. Quantum Generative Adversarial Networks (QGANs) have emerged as promising deep learning frameworks, effectively harnessing quantum advantages for practical tasks such as image processing and data optimization. Consequently, a topological code trainer for quantum-classical hybrid GANs is proposed as an auxiliary model to enhance error correction in machine learning-based decoders, demonstrating significantly improved training accuracy compared to the traditional Minimum Weight Perfect Matching (MWPM) algorithm, which achieves an accuracy of 65%. Numerical experiments reveal that the decoder achieves a fidelity threshold of P = 0.1978, substantially surpassing the traditional algorithm's threshold of P = 0.1024. To enhance decoding efficiency, a Transformer decoder is integrated, incorporating syndrome error outputs trained via QGANs into its framework. By leveraging its self-attention mechanism, the Transformer effectively captures long-range qubit dependencies at a global scale, enabling high-fidelity error correction over larger dimensions. Numerical validation of the surface code error threshold demonstrates an 8.5% threshold with a correction success rate exceeding 94%, whereas the local MWPM decoder achieves only 55% and fails to support large-scale computation at a 4% threshold.

Keywords: Quantum error correction; QGAN decoder; Reinforcement learning

1 Introduction

Many quantum physical systems, such as superconducting qubits [1], ion trap quantum computing [2], and photonic quantum systems [3], are widely regarded as promising platforms for the future development of quantum computers. Quantum bits are highly susceptible to noise and quantum decoherence, which can lead to computational errors dur-

Springer

ing the calculation process. One promising approach to overcoming these challenges is the implementation of quantum error correction (QEC), a critical technology that reduces errors by encoding multiple physical qubits to protect the information of logical qubits. In QEC, logical qubits are used to store the actual quantum information, while data qubits provide redundancy encoding for the logical qubits, with error detection assisted by parity-check qubits. Syndrome extraction is the core process for error detection, and periodic syndrome extraction, through repeated measurements, locates errors as observable syndrome states, allowing errors to accumulate and manifest over multiple cycles. Classical decoders analyze these syndromes to infer errors and correct them. Since errors in quantum systems can overlap and influence each other, decoders typically require broader correction subspaces to accurately correct errors, enabling practical computation with low error rates.

The rotated surface code [4] is considered one of the most promising quantum error-correcting codes (QECCs), where the encoding capacity does not depend on the size of the code itself, but rather on the number of topological qubits embedded in the torus. This encoding strategy not only effectively protects the quantum bit information but also ensures that error localization exhibits high locality. Topological quantum error correction leverages the geometric locality of check operators, using it as the foundation for symbolic operations. By mapping multiple physical qubits to a single logical qubit and arranging them on a grid for validation, this method relies only on minimal information from neighboring grids for verification. This topology-based error-correcting code, particularly in applications such as surface codes and color codes, can increase the code distance by expanding the grid size, thereby enhancing the stability and noise resistance of qubits [5]. Quantum topological codes exhibit high degeneracy, meaning the same error correction operator can be corrected by multiple strategies, requiring the decoder to precisely select the optimal error correction path. Designing high-fidelity and efficient decoders for quantum topological codes has long been a research focus in this field. For example, the minimum-weight perfect matching (MWPM) algorithm [6] offers an effective decoding strategy by identifying the shortest error correction path within the correction operator. However, despite the improvements in fault tolerance performance provided by MWPM, further research is required to enhance the fidelity of the decoder. In recent years, deep learning has demonstrated significant potential in this field, offering new methods and theoretical support for optimizing quantum topological code decoders. However, MWPM faces bottlenecks in decoding speed for large-scale quantum systems, especially under high-noise environments, where its performance degrades. To address this, the Union-Find (UF) method [7] has emerged as a more efficient alternative, capable of rapidly identifying and repairing error chains, significantly improving decoding efficiency and becoming another focal point in research. However, despite the superior speed of the UF method, it tends to produce suboptimal solutions when dealing with complex topological structures or long-distance qubit error correction, resulting in lower decoding fidelity compared to MWPM.

Over the past decade, we have witnessed significant success of machine learning in various complex tasks, such as classification, regression, and generative modeling [8–10]. Recently, a considerable amount of research has focused on applying machine learning to quantum error correction, with neural networks, as a deep learning model, being used in decoding quantum topological codes. For example, convolutional neural net-

works (CNNs) are used to learn quantum noise patterns [11]. By continuously optimizing weights, CNNs can effectively handle complex error patterns and improve decoding accuracy. However, training such network models requires large amounts of labeled data, and generating sufficient high-quality training data in practical quantum systems is a challenge. The application of traditional generative adversarial networks (GANs) in quantum error correction has also gradually gained attention [12]. Despite the dual advantages of generative and adversarial learning offered by GANs, they are prone to mode collapse during training, preventing them from fully capturing all noise distributions. Additionally, the traditional GAN structure struggles to efficiently handle high-dimensional quantum data, limiting its application in quantum decoding. To overcome the limitations of traditional machine learning methods, quantum generative adversarial networks (QGANs) have emerged as a promising decoder design approach, demonstrating tremendous potential. Unlike classical machine learning methods, quantum GANs operate directly in quantum state space, offering greater expressive power and the ability to extract deep features from complex quantum noise distributions, thereby generating higher fidelity error correction strategies. Quantum GANs not only improve efficiency but also enhance the decoder's generalization ability across diverse noise models through quantum state adversarial training. With the development of quantum computing hardware, quantum GANs are expected to become a core technology for improving the decoding performance of quantum topological codes, addressing bottleneck issues in existing classical methods, and advancing quantum error correction to new heights.

In this paper, we pioneered the use of a hybrid quantum-classical generative adversarial network (QGAN) as a pre-training model for error correction, assuming that the quantum generator partially serves as a black-box model capable of generating error-free or near-ideal quantum states. The "black-box" assumption here means that we are not concerned with the specific details of the internal operation of the quantum circuit, nor do we consider the noise or errors that it may introduce, and we focus on its function as a training error syndrome in the error correction process, with the overall architecture training shown in Fig. 1. The training of this decoder is based on optimal decoding datasets with code distances of d = 3 and d = 5, and the error correction strategies generated by the generator during training demonstrate a high success rate. Meanwhile, to accelerate decoding efficiency, we choose a Transformer decoder. The syndrome error outputs trained by GAN are input into the Transformer model, where its self-attention mechanism captures long-range dependencies between qubits globally, enabling high-fidelity error correction at larger scales. Experimental results show that under the same noise rate conditions, the decoder processed by quantum GAN and Transformer demonstrates higher success rates and fault tolerance thresholds compared to the MWPM algorithm and traditional GAN algorithm. Particularly, at the error correction threshold of the quantum surface code, the QGAN decoder exhibits superior fault tolerance. The hybrid decoder based on Quantum GAN and Transformer will provide new possibilities for achieving fault-tolerant quantum computing.

The main content of this paper is outlined as follows. In the next section, we will briefly introduce the background of Quantum GAN, rotational surface codes, and quantum error correction. Section 3 provides a detailed description of the structure and underlying principles of the QGAN algorithm for training the error correction model. Section 4 will discuss the details and principles of the Transformer model. Section 5 presents the exper-

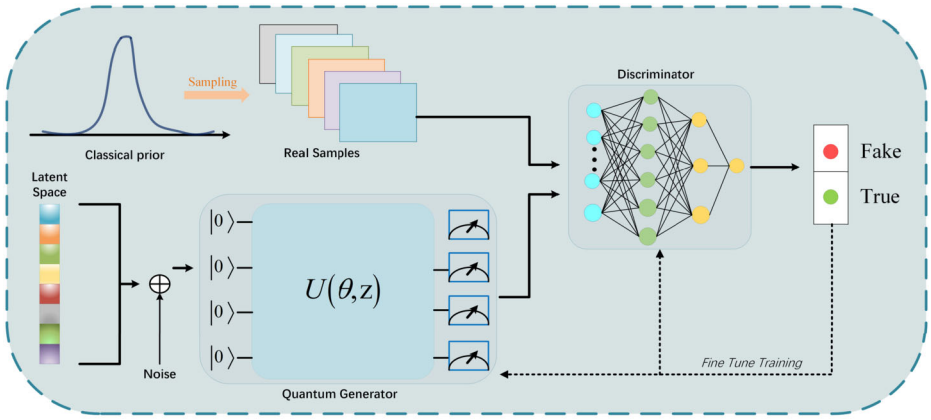

Figure 1 Architecture of the QGAN error correction training model. The lower left corner is the input of potential vectors into the PQC circuit network encoder to obtain a new feature vector $\omega 1$. The discriminator into the classical network divides the data into real data and fake data in the upper left corner. It is also constantly compared with the real samples of the input and returns to the PQC circuit for training and optimization to obtain the final feature vector $\omega 2$

imental results of the hybrid error correction scheme and compares them with traditional algorithms. Finally, a summary and outlook of the research findings are provided.

2 Theoretical background

2.1 Surface code

The 2D plane mapping of the quantum surface code on a torus [13] is a class of quantum topological codes that maintain spatial properties under continuous deformations. This topological quantum error correction code not only simplifies physical implementation but also possesses the advantages of stabilizer codes, enabling the effective protection of target qubits through the introduction of auxiliary qubits. Adjacent parity-check qubits detect errors in data qubits through the stabilizer circuit, which can identify and distinguish X-type, Y-type, or Z-type errors. The self-duality of surface codes and their 1/2 rotational degree of freedom allow their stabilizer operators [14] to be periodically embedded into the grid, forming an efficient periodic structure. Let the number of vertices, edges, and faces of the lattice be denoted as N_V, N_E, and N_F, respectively. According to the structure of the surface code, the code has $N_V + N_F - 2$ independent generators. Based on stabilizer theory, the number of encoded qubits "k" can be expressed as:

$$k = N_E - (N_V + N_F - 2) = 2 - \chi = 2g \tag{1}$$

Here, χ represents the Euler characteristic, and g denotes the number of closed loops. For a surface code with code distance "d", we have $N_V = N_F = d^2$, thus the number of qubits is $2d^2$ and the number of stabilizers is $2d^2 - 2$. We define the lattice operator and vertex operator as follows:

$$S_m = \prod_{j \in m} \sigma_j^x, T_n = \prod_{j \in n} \sigma_j^z \tag{2}$$

For any edge $j \in m$, the code ensures that $S_m \mid \psi\rangle = \mid \psi\rangle$, which holds for all m; Similarly, $T_n \mid \psi\rangle = \mid \psi\rangle$, which holds for all n. The stability of the topological code is maintained

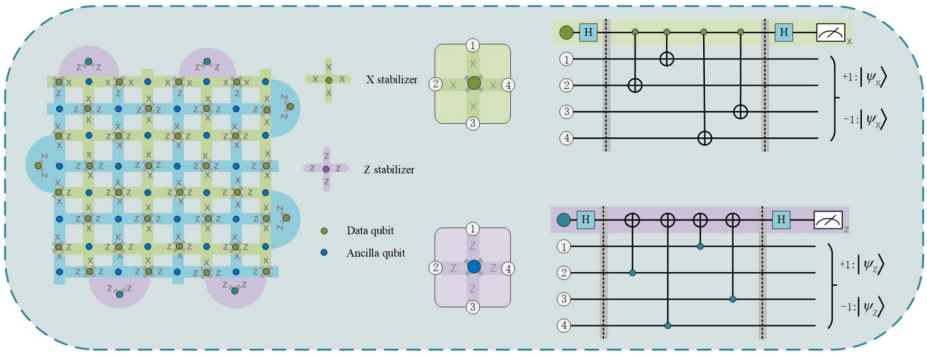

Figure 2 Basic structure of a rotating surface code. The *X*-stabilizer is used to detect *Z*-errors and the *Z*-stabilizer is used to detect *X*-errors. On the right is a single-cycle surface code quantum circuit that includes *X*-ancilla quantum bits and *Z*-ancilla quantum bits

through stabilizer generators and local cohomology [15], ensuring that particles can transition from the excited state to the ground state. The topological code space consists of a single qubit in a two-dimensional Hilbert space, and its operations are defined as follows:

$$\Pi_n = \left\{ e^{i\theta} G_1 \otimes \cdots \otimes G_n \mid \forall k \in \{1,\ldots,n\}, G_k \in \mathcal{G}, \theta \in \left\{ 0, \frac{\pi}{2}, \pi, \frac{3\pi}{2} \right\} \right\} \tag{3}$$

The eigenspace with eigenvalue +1 constitutes the surface code space. Assuming that the dimension of the code space is 2^m, p quantum bits can be encoded in this space. Among these quantum bits, the smallest stabilizer T can independently describe p independent topologically stable generators $\{h_1,\ldots,h_{m-p} \mid \forall j \in \{1,\ldots,m-p\}, h_i \in T\}$. Since the surface code possesses a local topological homotopy, it acts on the spin states of the $2D$ neighboring lattice, generating an analog Hamiltonian quantity of:

$$\mathcal{H}_{SC} = -K \sum_{i=1}^{N} S_i - K \sum_{j=1}^{M} T_j, K > 0 \tag{4}$$

The rotated surface code is a simplified variant of the surface code, as shown in Fig. 2. Unlike the classical surface code, the rotated surface code optimizes the encoding structure by introducing angular rotations to the grid, thus improving encoding density and error resilience. Its core advantage lies in its efficient use of hardware resources, particularly in significantly reducing the demand for qubits and gates. This makes it easier to implement on existing quantum hardware platforms and gives it good scalability potential. Furthermore, the rotated surface code retains the stabilizer architecture of the traditional surface code, with the decoder correcting errors on the data qubits by analyzing the stabilizers $\{C_i\}$ from auxiliary measurements. [16] With the further development of the rotated surface code and the integration of efficient decoder algorithms, it is expected to play a significant role in achieving more efficient error correction schemes.

2.2 Quantum generative adversarial network(QGAN)

Quantum Generative Adversarial Networks (QGAN) are a novel unsupervised learning model that combines the classical GAN framework with quantum computing [17]. The

QGAN employed in this paper consists of a quantum generator G and a classical neural network discriminator D, with the generator leveraging the unique properties of quantum superposition and entanglement. The two components are parameterized by θ_G and θ_D, respectively, and engage in a game of maximization and minimization. Specifically, G's task is to obtain a noise vector z from the noise source and generate data x' that closely mimics real data x to deceive D, while D's task is to distinguish whether the received data is real or generated by G. The training process of G is to maximize D's error probability, while D's training process minimizes the error probability of distinguishing between the data, alternating updates of θ_G and θ_D until the network reaches optimality. Ideally, in the game between the two, when the data x' generated by G perfectly matches the real data x, and D cannot determine the data's origin, optimality is reached, which corresponds to the Nash equilibrium [18].

The generator G acquires a low-dimensional noise vector z sampled from some distribution P_Z (e.g., a Gaussian distribution [19]) and then maps it into a high-dimensional space to generate the output $G_{(n)}$), with the goal of matching the learned distribution P_G with the true distribution P_{data}. The discriminator D takes its input from either the real data x or the output $G_{(n)}$), and D outputs the probabilistic outcome y of its belief that x originates from P_Z, and classifies the input as originating from the real data if y is greater than one-half of it, and otherwise classifies it as originating from the generator G. The optimization objective of D is to maximize the probabilistic outcome y, while the optimization objective of G is to generate spurious samples to fool D. The training process of the GAN is represented as a process of playing with each other.

$$minmax\mathbb{E}_{\mathbf{x}\sim P_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim P_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))] \tag{5}$$

2.3 ML based decoder

In recent years, machine learning (ML) decoders have made significant advancements in the field of quantum error correction, particularly in neural networks (NN) and reinforcement learning (RL). Early work in the neural network domain saw the introduction of the Boltzmann machine for decoding topological codes [20]. This was later followed by the application of multi-layer perceptrons [21] and long short-term memory networks (LSTM) [22] in the decoding of three-dimensional codes, further promoting the use of ML decoders. Progress in the reinforcement learning domain is marked by framing the decoding problem as a reinforcement learning environment [23], optimizing for specific errors and greatly enhancing the robustness of decoding. To address the scalability challenges in decoding, researchers have proposed several solutions, including deep convolutional neural networks [24] and ResNet [25]. In this context, the Transformer architecture, one of the most groundbreaking neural networks in recent years, has shown enormous potential in quantum error correction tasks due to its ability to process sequences without requiring sequential handling. The Transformer model, through its self-attention mechanism, can process large amounts of data in parallel and effectively capture both global and local correlations, making it highly adaptable to large-scale error patterns. The Transformer-based decoding architecture adopts a two-level local-global design, where a local decoder first corrects local errors, and a global decoder then integrates global information to achieve a more efficient and precise decoding process. Unlike traditional neural networks, Transformers significantly enhance parallel computation capabilities, reduce decoding time,

and their scalability enables them to handle larger quantum error correction tasks. With the development of hardware accelerators, quantum error correction schemes based on Transformers are expected to achieve significant practical results.

3 Methods

3.1 Error correction

In quantum error correction, the first step is to measure the stabilizers of the surface code to obtain the syndrome. [26] Within the surface code framework, stabilizers are categorized into two types: one for detecting phase errors (Z errors) through vertex operators A_v, and the other for detecting bit-flip errors (X errors) through plaquette operators B_f. [27] When an error is detected, the stabilizer's eigenvalue changes from +1 to -1, creating symmetric defects at the lattice vertices. In an $A \times A$ lattice, there are A^2-1 independent stabilizer operators, meaning each independent syndrome can be derived from these stabilizer measurements. Theoretically, an isolated -1 eigenvalue is impossible, as errors must occur in pairs, and stabilizer measurements typically generate two symmetric defects locally. The probability of obtaining a given correction operator is:

$$P(\partial\lambda, \partial^*\mu) = \sum_{\epsilon \in X_\epsilon} \sum_{\zeta \in z_\zeta} P(\lambda + \epsilon, \mu + \zeta) \tag{6}$$

In the surface code, syndromes exhibit degeneracy, meaning that different error patterns can produce the same syndrome. To correct errors, correction qubits must be selected and defects in the syndromes eliminated pairwise. If these correction qubits form a trivial loop [28] with the error qubits, the surface code's logical state will revert to its initial state, and the error will be successfully corrected. However, if the correction qubits form a nontrivial loop, i.e., one that loops around the lattice's topological boundary, although defects are eliminated, the logical state of the quantum code is altered, leading to a correction failure. In this case, although local errors are eliminated, the system's global state has changed. An optimal decoder selects the path with the fewest correction qubits to minimize the risk of forming a nontrivial loop. The success probability of this optimal path is:

$$\mathcal{P}_{opt} = \sum_{\sigma, \sigma^*} \max_{\epsilon, \epsilon^* \in \mathcal{E}} \mathcal{P}(\lambda + \epsilon, \mu + \epsilon^*) \tag{7}$$

In the limit of large-scale lattices, when the noise is below a specific critical threshold, the probability of successful correction, p_{opt}, approaches 1. For incoherent noise models [29], as the system size d tends to infinity, the critical threshold $\mathcal{P}_b$ is determined by mapping the system to the random Ising model [30]. Below this threshold, the correction strategy leads to the formation of trivial loops. Figure 3 illustrates the error correction process of the quantum rotational surface code, clearly depicting the relationship between quantum bits and stabilizers, and providing examples of both successful and failed corrections caused by a string of erroneous quantum bits.

A corresponding error correction strategy must be devised for a string of erroneous quantum bits. Error correction in the quantum rotational surface code exhibits degeneracy, meaning that for a given error syndrome, multiple valid correction schemes exist.

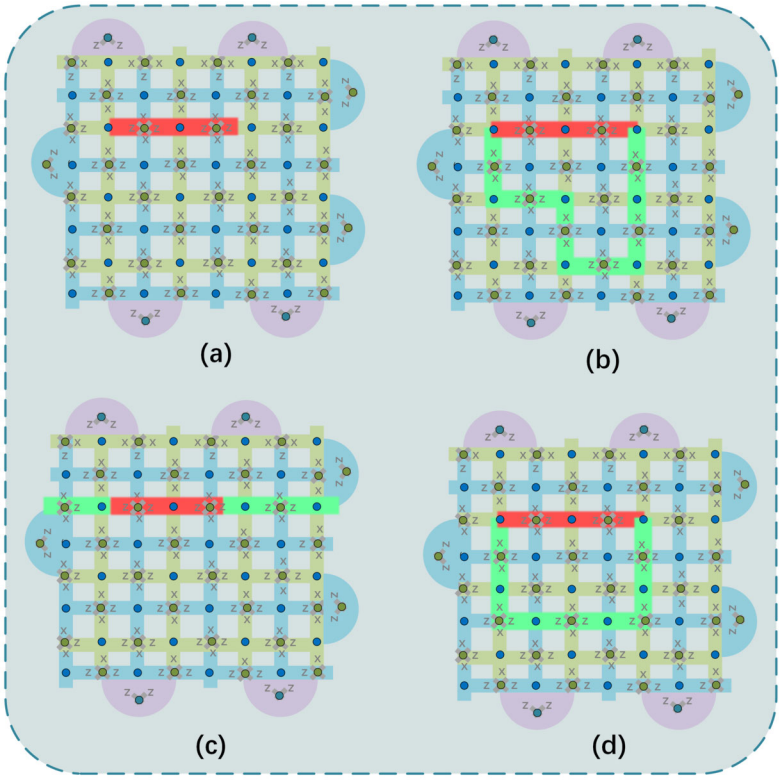

Figure 3 Rotating surface code error type. Red data quantum bits indicate that an error has occurred. Green data quantum bits mean they are selected as error correcting quantum bits. (a) Surface code of the erroneous data quantum bit. The stabilizer eigenvalue around the erroneous quantum bit becomes −1. (b) Successful quantum surface code for error correction. This strategy forms a mundane cycle, but is not optimal (c) Failed quantum surface code error correction. The error correction chain and the error chain form a non-trivial loop (d) Successful quantum surface code error correction. The error correction chain and the error chain form a mundane loop

This differs from other quantum error correction codes, which typically provide fixed correction methods for each error type. In the rotational surface code, error correction is considered successful as long as the erroneous quantum bits form a trivial loop with the correcting quantum bits; conversely, if a nontrivial loop is formed, error correction fails. Therefore, the goal of the decoder is to generate a correction path such that the quantum bits along the path form a trivial loop with the erroneous quantum bits.

When bit-flip errors occur, a pair of X syndromes will be generated on the X quantum bits. Once the error is correctly corrected, the corresponding syndromes will disappear. However, when a Y error occurs, both X and Z syndromes are generated simultaneously, making error correction irreversible and reducing the success rate of correction. To improve the success rate of error correction, we should avoid Y errors as well as situations where both X and Z syndromes are encountered simultaneously, as errors in such cases are more difficult to correct.

To address this issue and simultaneously find an optimal correction path, we introduce a quantum generative adversarial network (QGAN) model based on a quantum generator in the context of a non-polarized noise model, training it to generate defect-matching results [31] that assist in effective classification of syndromes. This approach optimizes the

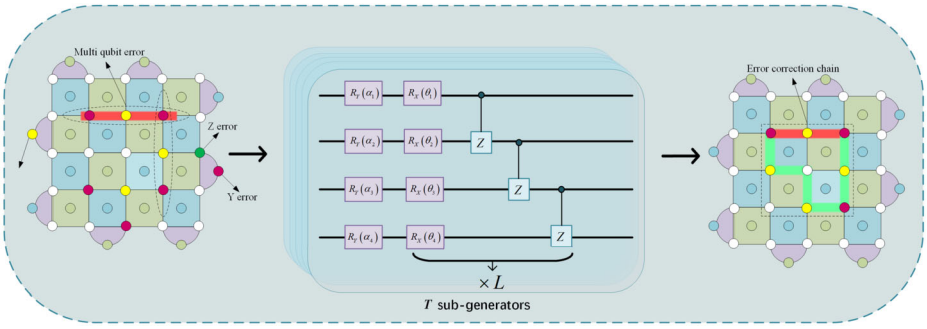

Figure 4 Generator structure of the GAN decoder. On the left, in red, is the surface code with erroneous bit quantum bits, which is fed into the quantum generator for training, resulting in a corrected surface code that forms a mundane loop

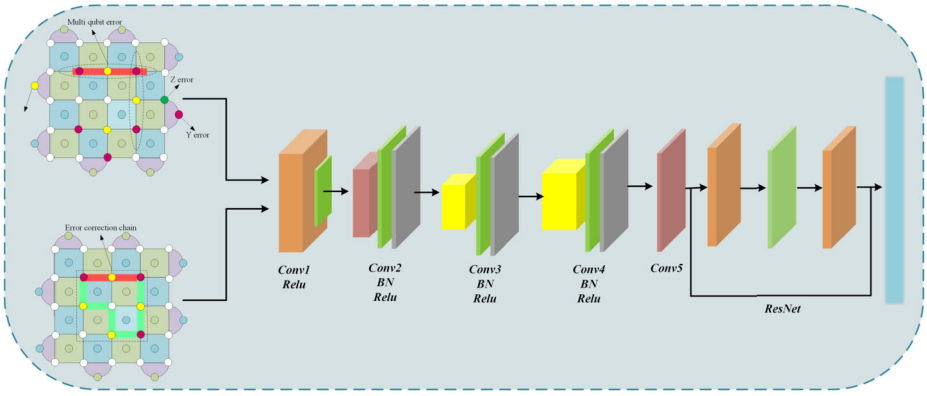

Figure 5 Structure of the classical discriminator decoder. Orange and red blocks indicate the different convolutional layers, green blocks indicate the ReLU function, gray blocks indicate the BN layer, and blue rectangles indicate the FC layer

syndrome recognition process, thereby reducing correction failures caused by complex error combinations and improving the overall efficiency of quantum error correction.

3.2 QGAN decoding model

In this section, we introduce the quantum generative adversarial network framework in quantum error correction (See Appendix C for details). Consisting of a quantum generator and a discriminator of a classical neural network, its training framework is shown in Fig. 4, 5. Unlike the classical GAN, the variational quantum generator does not need to map the a priori noise distribution into a high-dimensional space, but instead obtains different results through the randomness of the quantum measurements. The QGAN decoder is continuously adversarial-trained to make the generator produce correction paths with mundane loops, and the eigenvalues of the rotated surface code are taken as inputs, which are passed in the form of matrices into the discriminator, which is trained to recognize the authenticity of the inputs and updates the parameters. The quantum generator iteratively optimizes training and learning through variational quantum circuits (see Appendix A for details), generates forged data to confuse the discriminator, and updates the discriminator parameters, repeating the process until a Nash equilibrium point is reached.

3.2.1 Quantum generator structure

The quantum generator component is a network of multiple sub-generators. Each of these sub-generators is a variational quantum circuit (PQC) that optimizes the entire model by training iteratively on each sub-generator individually. A quantum circuit is essentially a series of operations on a series of quantum gates applied to quantum bits. Single-bit quantum gates [32], such as the Bubblegate (R_x, R_y, R_z) produce a rotation angle transformation on a quantum bit. And multi-bit quantum gates [33], such as the CNOT gate, are applied to produce entanglement on two quantum bits.

The generator concatenates the eigenvalue matrices of the rotating surface code along the channel dimensions, with N-dimensional eigenvectors $L = (l_1, l_2, \ldots, l_N)$. Immediately after, the components of the feature vector L are input to the R_y layer rotational coding that is parameterized in the sub-generator. From the input $|0\rangle^{\hat{\otimes} n}$ to the generator, we obtain the state by encoding the circuit:

$$|L\rangle = R_Y^1(l_1)R_Y^2(l_2)\ldots R_Y^N(l_N)|0\rangle^{\otimes N} \tag{8}$$

Here, $R_Y^N(l_N)$ represents a rotation gate parameterized by angles and applied to quantum bits. Subsequently, the vectors pass through the rotational layers of an efficient ANSATZ structure, each comprising parameterized $R(\lambda, \theta, \phi)$ gates, followed by a set of CNOT gates applied to neighboring qubits to generate entanglement. $R(\lambda, \theta, \phi)$ can be expressed as $R_z(\lambda)R_Y(\theta)R_z(\phi)$, designed to map a quantum bit to an arbitrary quantum state on the Bloch sphere [34]using unitary transformations. A variational quantum circuit consists of X, representing fixed gates for entangling quantum bits, and θ, representing gates with tunable parameters, where the latter is optimized via training. Thus, the entire circuit V can be represented as an optimized function V = U(X, θ) through unitary transformations [35]. One advantage of variational quantum circuits is their ability to be trained and optimized using existing machine learning techniques, such as the cross-entropy loss function [36] and the Adam optimizer [37].

All sub-generators share an identical circuit structure, comprising a five-qubit circuit, with the configuration of each sub-generator illustrated in Fig. 4. The circuit encodes the eigenvalues of the input surface code into angles using R_x and R_y gates, while CZ gates generate entanglement, employing a unitary transformation $U(\lambda, \theta, \phi)$ such that the quantum state of the i-th sub-generator evolves through the unitary operation as:

$$|\psi_i\rangle = U_L(\lambda_i, \theta_i, \phi_i)|L\rangle \tag{9}$$

Subsequently, a measurement operation is applied to the Ancilla qubits, yielding the resultant state of the output data qubits. This process aims to enhance the nonlinearity of the generator circuit. In general, the measurement operator $M = (|0\rangle\langle 0|)^{\otimes A}$ is chosen for projection measurement, resulting in the quantum state $|\psi_U\rangle$ after tracing out the Ancilla qubits being:

$$|\psi_U\rangle = \mathrm{Tr}_A\left(\frac{(|0\rangle\langle 0|)^{\otimes A} \otimes \mathbb{I}|\psi_i\rangle\langle\psi_i|}{\langle\psi_i|(|0\rangle\langle 0|)^{\otimes A} \otimes \mathbb{I}|\psi_i\rangle}\right) \tag{10}$$

From the formula, it is evident that both the numerator and denominator contain $|\psi_i\rangle$, indicating that the resultant quantum state is a nonlinear transformation of $|L\rangle$. Finally,

we measure the probability of each resultant quantum state to obtain the output of each sub-generator.

3.2.2 Discriminator structure

The discriminator used is a classical deep learning neural network, designed to assist in training the quantum generator, as illustrated in Fig. 5. The classical discriminator is larger in scale than the quantum generator, as it needs to compete with multiple quantum generators in adversarial training. Similar to traditional discriminators, it extracts deep features from both generated and real data through convolutional layers, interspersed with residual blocks [38] for feature extraction over changes in data dimensions, and includes multiple linear layers with *LeakyReLU* activation functions [39]:

$$\mathrm{LReLU}(x) = \begin{cases} x, & \text{if} x > 0 \\ \alpha \cdot x, & \text{if} x \leq 0 \end{cases} \tag{11}$$

Here, α is a constant used to handle negative values. In addition to the activation function, a batch normalization (BN) layer [40] is added after each convolutional layer to standardize the data, thereby stabilizing the training process. The final single output allows it to function as a classifier, distinguishing real data from generated data:

$$y_i = \gamma \cdot \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta \tag{12}$$

Here, x_i represents the i-th input sample, μ_B and σ_B^2 are the mean and variance of the mini-batch, ϵ is a constant to prevent division by zero, and γ and β are learnable parameters, which represent scaling and shifting operations applied to the normalized input data, ensuring stable and rapid convergence of the network. After convolution and batch normalization, the output is produced through a fully connected layer, with the final layer utilizing a *Sigmoid* activation function to increase the network's nonlinearity, enabling it to learn more complex structures. The entire process is represented as:

$$y = \sigma(Wx + b) \tag{13}$$

In the equation, W represents the weight matrix, and b represents the bias vector. The details of the network parameters are shown in Table 1 in Appendix C.

3.3 Transformer decoding model

Building on the error syndromes generated by the QGAN, the Transformer model further optimizes the decoding process [41]. The Transformer processes all input data elements in parallel using the self-attention mechanism, demonstrating high efficiency, particularly in the handling of multi-round measurement data. This overcomes the limitations of sequential processing in traditional decoding algorithms and meets the requirement of surface code decoding for multi-round measurements. Unlike recurrent neural networks (RNNs) [42], the Transformer does not rely on sequentially propagated hidden states, making it more robust in handling long sequence data and effectively avoiding the

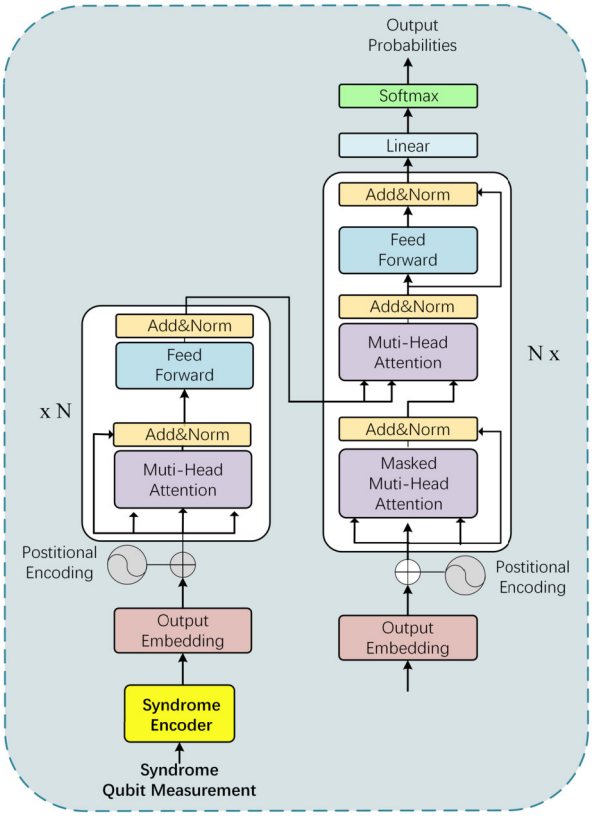

Figure 6 Transformer model architecture. This figure shows the architecture of the Transformer model, including the encoder and decoder. The encoder consists of N identical layers each containing two sub-layers: a multi-head self-attention mechanism and a fully connected feed-forward network(See Appendix C for details). The decoder also consists of N identical layers, but with the addition of sublayers for masking the multi-head attention to prevent positional attention to subsequent positions of the data quantum bits encoded to produce physical error predictions

vanishing gradient problem. Errors in quantum computing have a cumulative effect, requiring the decoder to capture error correlations over long time spans. The Transformer's global self-attention mechanism [43] can simultaneously access the full input sequence, exhibiting superior fault tolerance under complex error syndrome sequences.

The Transformer encoder consists of multiple layers, each containing a multi-head self-attention mechanism (MHSA) and a feed-forward neural network (FFN), as shown in the model architecture in Fig. 6. The multi-head self-attention mechanism enables the model to globally interact with any syndrome feature in a three-dimensional grid, enhancing its sensitivity in detecting error correlations among multiple qubits:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{v} \tag{14}$$

Here, $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ represent the query, key, and value vectors, respectively, and d_k denotes the dimension of the key. This mechanism effectively captures widely distributed error patterns, rather than being limited to a local range. The feedforward network layer projects the embedded feature information into a high-dimensional space to extract more complex and deeper error features. After the high-dimensional projection, the feedfor-

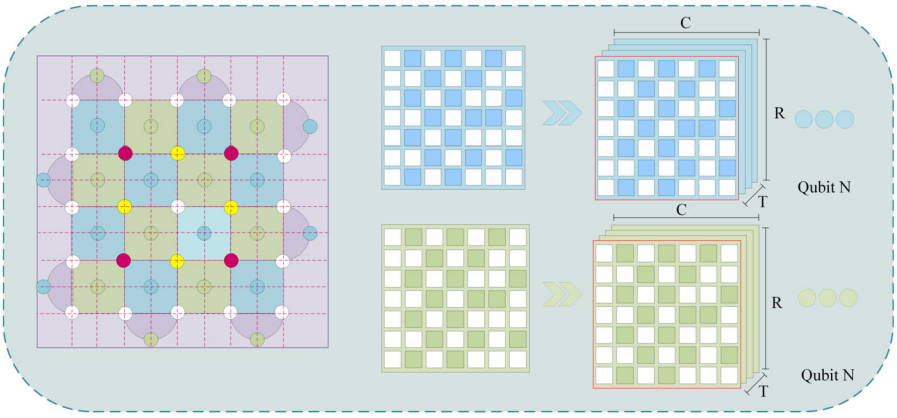

Figure 7 The architecture of the transformer decoder model. First, the input syndromes are encoded into a cubic grid of dimension $\mathbf{d}+1$, which is then fed into the Transformer encoder that includes self-attention and FFN layers. Next, the transformer decoder processes the $\mathbf{d}$-dimensional positional encoding of the data qubits to generate the predictions for physical errors

ward network uses a nonlinear activation function to compress the high-dimensional features back to a lower-dimensional space, restoring the original input dimensions. This not only enhances the model's generalization ability but also effectively improves the prediction accuracy of complex errors, ultimately enhancing the overall performance of the quantum error correction system. Finally, we use a Transformer decoder to predict physical errors based on the syndrome measurement results of each round, with the final output prediction converted into the probability of error occurrence via a '*Sigmoid*' function:

$$P(\text{error}) = \sigma(\mathbf{W}_o \cdot \mathbf{h}_i + \mathbf{b}_o) \tag{15}$$

Here, σ represents the activation function, $\mathbf{W}_o$ is the weight matrix, $\mathbf{h}_i$ is the input feature vector, and $\mathbf{b}_o$ is the bias vector. Once the decoder completes the prediction of all errors, these predictions will be used to remove the syndromes and obtain the global parity of the errors [44]. The process of syndrome removal involves flipping the syndromes associated with the errors again, with the final output being the global parity prediction of the surface code syndromes. During the decoding process, the positional information of the quantum bits is embedded as vector inputs. By analyzing the three-dimensional grid data, the decoder can determine which quantum bits have errors. Each decoding layer includes a self-attention mechanism and a cross-attention mechanism between the encoder and the decoder. The cross-attention allows the decoder to access the output information from the encoder and combine it with the processed syndrome data to predict errors, referencing all prior measurements and features. This mechanism ensures that the decoder can access and integrate all prior syndrome information.

3.4 Details of model decoding

The transformer architecture includes an embedding module, a recurrent core, and a readout network. It is used in surface code decoding due to the correlation between the surface code's grid structure and its vertices. The syndrome sequence from qubit measurements feeds into the recurrent core, and the readout network classifies these syndromes into labels. Surface code decoding involves multiple rounds of measurements, with each round

producing a new syndrome sequence. Unlike traditional decoders, ML decoders like transformers process all input data in parallel, making them more efficient for multi-round data and avoiding the vanishing gradient problem seen in RNNs.

To prepare the input data for the neural network, we map the surface code onto a square lattice, as shown in Fig. 7. This lattice representation ensures that each syndrome qubit is positioned at the intersection of grid lines. For a surface code with code distance d, we transform it into a lattice of size (d + 1) × (d + 1) by adding virtual nodes. These virtual nodes do not contribute to the decoding process and are always set to 0 during decoding. The lattice contains $d^2 - 1$ stabilizers, and the virtual nodes help ensure that the syndrome qubits are placed at the correct locations on the lattice. Error syndromes only affect the stabilizer values, and the number of measurement rounds generally equals the code distance d. However, an extra round is added to account for physical mechanisms, bringing the total number of rounds to d + 1. By combining the data from these rounds, each syndrome qubit's features for each round are placed into a (d + 1)-dimensional cubic grid. Each position in the grid is represented as a feature vector of length six. To distinguish between d and d syndrome qubits, they are represented in separate channels, ensuring accurate mapping of their positions and measurement results. This encoding ensures that the network receives precise information for decoding.

$$V_i^{(t)} = \left[V_{X,i}^{(t)}, V_{Z,i}^{(t)}, D_X^{(t)}, D_Z^{(t)}, T_1^{(t)}, T_2^{(t)} \right] \tag{16}$$

Here, $T_1^{(t)}$ and $T_2^{(t)}$ represent the start and end points of the measurement rounds. The first two channels are used to encode the positions of the X syndrome qubits and Z syndrome qubits, respectively. Since syndrome information can change across measurement rounds, the next two channels are used to capture this variability, reflecting the state of the syndrome qubits at different times. To help the network adapt to different measurement rounds, the temporal boundaries are clearly defined using the fifth and sixth channels. In the first round, these channels are set to 1, and in all subsequent rounds, they are set to 0. This ensures that the network can identify the start and end points of the measurement sequence. In the final stage of quantum error correction, errors are corrected by measuring the data qubits in a specific basis.

4 Simulation analysis

When a qubit error occurs, a quantum generative adversarial network (QGAN) is first trained to generate a correction path. This path ensures that the qubits on the path and the error qubits form a trivial loop, maximizing the likelihood of successful error correction. The generated parity-check matrix is subsequently processed by a Transformer network model, which decodes it to produce the final syndrome.

4.1 Training model

The experiment was conducted using Python 3 with the PyTorch and PennyLane libraries. PyTorch is a widely adopted and efficient machine learning library, while PennyLane, a popular quantum machine learning framework, integrates seamlessly with mainstream machine learning libraries, facilitating enhanced interaction between quantum computing and classical machine learning.

To evaluate the performance of quantum generators and enable a direct comparison with the classical QGpatch [45]model, we adopted its training parameters and employed the ADAM optimizer. The learning rate for the generator was set to 0.25 and that for the discriminator to 0.06, based on the results of multiple training iterations. The outcomes of sub-generators were integrated using the measurement results of Ancilla qubits in each generator. Due to resource limitations, we selected a batch size of 20 for each training iteration, updating the generator's parameters throughout the training process. The generator underwent 500 iterations in total.

In traditional neural network algorithms, gradient descent methods (e.g., stochastic gradient descent, SGD) [46] are typically employed to update the generator parameters and compute gradients of the loss function. By applying the chain rule in the backpropagation algorithm, the partial derivatives of the loss function are calculated step by step. However, this method is not suitable for the complex gradient scenarios in quantum circuits, where the gradient information in variational quantum circuits (VQCs) is typically treated as a "black box" Therefore, the parameter-shift rule [47] is employed to compute the partial derivatives of quantum circuits by altering the parameters of the PQC twice. The parameter-shift formula is expressed as:

$$\frac{\partial L}{\partial \theta} \approx \frac{L(\theta+\delta) - L(\theta-\delta)}{c} \tag{17}$$

Here, δ denotes a small offset added to the selected parameter of the circuit. This offset is finite, with its value restricted to a maximum of $\pi/2$. The values of δ and c are determined by the type of quantum gate. The gradient is estimated by comparing the changes in the loss function values before and after applying the shift. In this manner, the gradient information of the PQC is derived within the same quantum circuit. Assuming the generator has n total parameters, the gradient of the j-th parameter of the i-th sub-generator with respect to loss function $f(\theta)$ is expressed using the parameter-shift rule as follows:

$$\begin{aligned}\frac{\partial f(\theta,\lambda)}{\partial \theta_{i,j}} = \frac{1}{2}\langle f(\left[\theta_{1,1},\ldots,\theta_{i,j}+\frac{\pi}{2},\ldots,\theta_n\right],\lambda)\rangle \\ - \langle f(\left[\theta_{1,1},\ldots,\theta_{i,j}-\frac{\pi}{2},\ldots,\theta_n\right],\lambda)\rangle\end{aligned} \tag{18}$$

By continuously optimizing and training the quantum GAN model, a trivial correction loop path was achieved with an accuracy of 99.8%. Simultaneously, during the search for the optimal ancillary path, determining the optimal code distance between erroneous qubits is essential. Figure 8 and Figure 9 presents the training outcomes of the quantum GAN decoder, showing the generator and discriminator losses (L_G and L_D) after 700 iterations with a code distance of d = 3 and 3000 iterations with a code distance of d = 5. Furthermore, we introduced an approximation using a discrete Gaussian distribution [48] described in Appendix C.

After determining the optimal error correction path, a new feature vector is generated and subsequently decoded (i.e., syndrome selection). Thus, training a stabilizer to match the optimal measurement state with high probability and output successfully corrected syndromes is crucial. A Transformer network decoder is employed for training. Specifically, the Transformer model features a six-layer encoder-decoder architecture, with an embedding dimension of 64. The multi-head attention mechanism incorporates

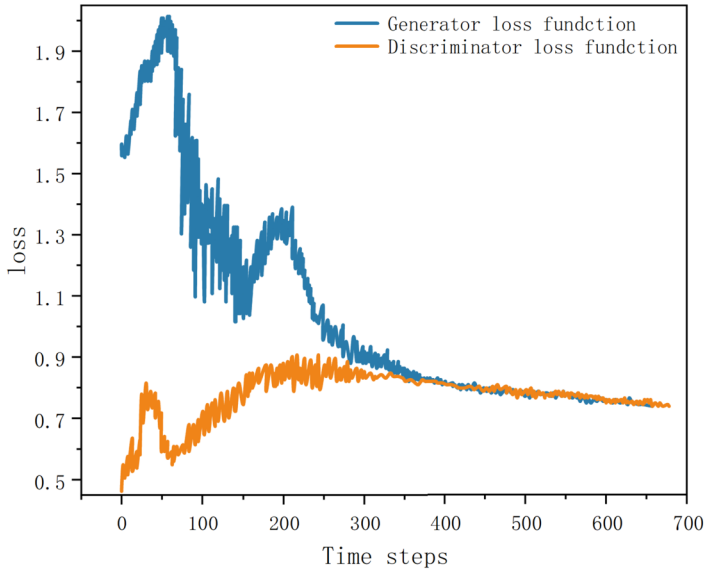

Figure 8 Training results of the QGAN decoder. The graphs show the loss of the generator and the discriminator. Used for d = 3 is considered fully trained after 700 training iterations

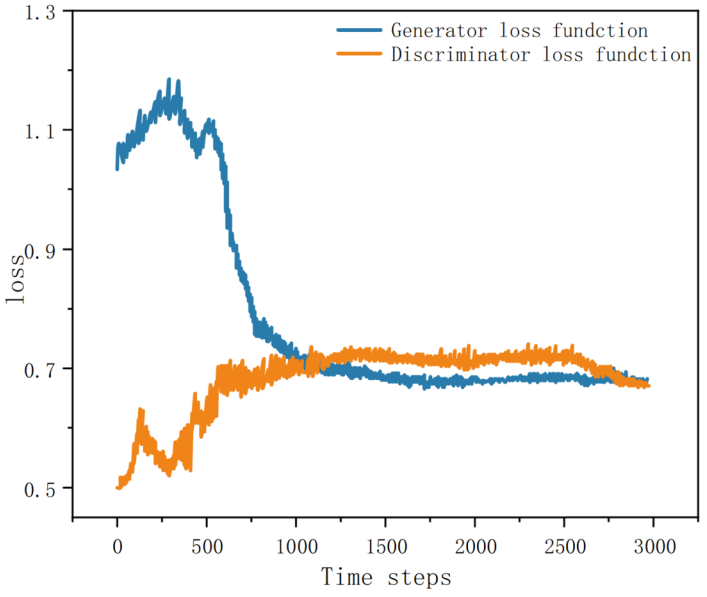

Figure 9 Training results of the QGAN decoder. The graphs show the loss of the generator and the discriminator. Used for d = 5 is considered fully trained after 3000 training iterations

four heads, while the feedforward network's hidden layer dimension is set to 256. The training dataset comprises 10^5 samples, with a base error rate of 1.2%. In each batch, the model executes forward propagation to compute the output and loss, followed by back-propagation to update the weights. Each parameter update uses a step size of 0.1%, a dropout rate of 0.1, and L2 regularization with a coefficient of 0.0002 to effectively mitigate overfitting and improve generalization. Furthermore, the Adam optimizer and binary

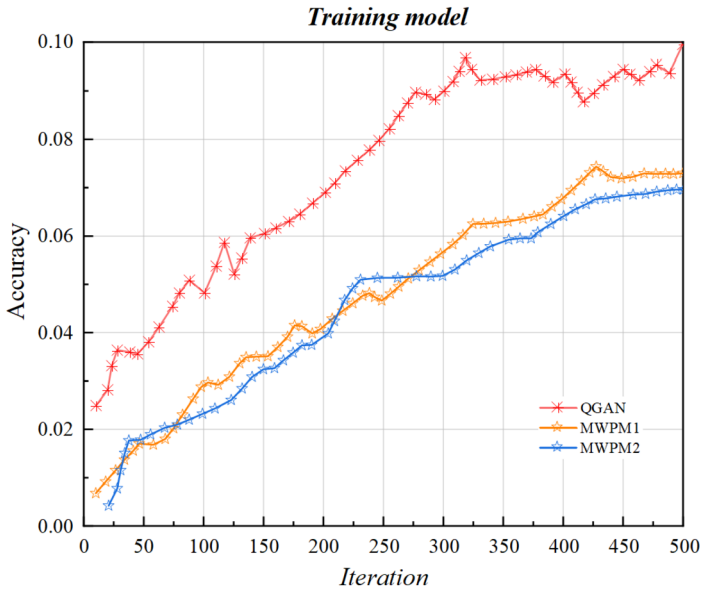

Figure 10 The figure shows a comparison of the iterative change in model training accuracy. The red star solid line indicates that the decoder model reaches 92.4% for the training accuracy model under the QGAN network structure. The orange circular solid line and the blue circular solid line are the training accuracies of 55.6% and 66.7% for the two different states of the MWPM

cross-entropy loss function are employed to ensure a smoother training process and accelerated convergence. The hyperparameters for the model training process are summarized in Table 2 in Appendix C.

During the training process, we first train the data with a code distance of d = 3, stopping when the decoding performance approaches the threshold. Subsequently, we increase the code distance and continue training on data with the same error rate, stopping again when the new threshold is approached. Finally, when the code distance is further increased, we observe that for the same error rate dataset, the prediction results begin to diverge from the threshold, indicating that a high-precision prediction model sufficiently close to the threshold has been trained. This is shown in Fig. 10, which illustrates the iteration of training and prediction accuracy.

4.2 Analysis of results

We tested the error correction capabilities of the Quantum GAN and the traditional local MWPM model on systems with code distances of d = 3,5,7,9, and the results obtained after training under specific noise conditions are shown in Fig. 11. At lower error rates, the error correction success rates for different code distances all exhibited a high level of performance. The logical error rates vary with different code distances, with larger code distances leading to higher error correction success rates. However, as the flip error rate increases, the logical error rate gradually approaches saturation. We define the threshold as the orange solid line in the figure, which represents the error rate at which error correction can achieve a logical error rate below the threshold. It can be observed that the error correction threshold for the local MWPM model is around 4.3%, whereas the threshold for the trained Quantum GAN reaches approximately 7.5%, highlighting the superiority

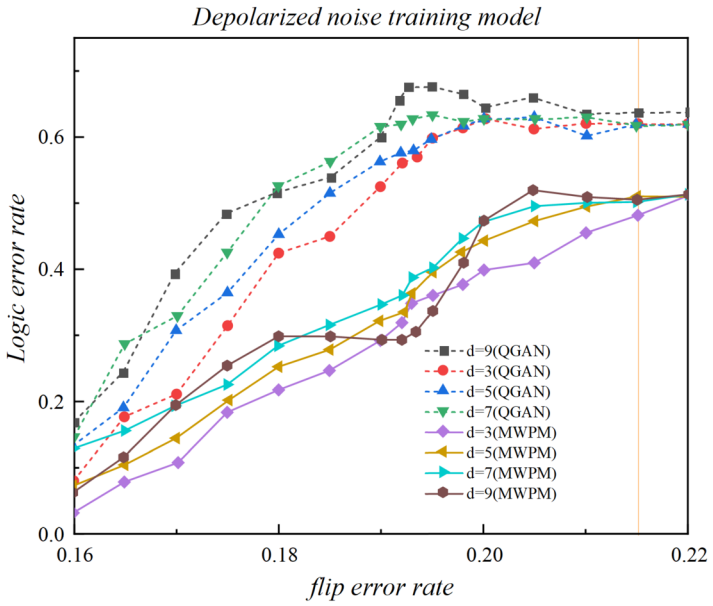

Figure 11 The error correction accuracies of the model using QGAN structure with surface code distances of 3, 5, 7 and 9 are green triangular dashed, blue triangular dashed and red circular dashed, respectively. The logical error rate changes from 0.14, 0.174, and 0.21 to 0.589 as the flip error rate increases, and the error correction strength of the MWPM changes to about 0.4 as the flip error rate increases for the purple-brown circular solid line and blue-yellow triangular solid line. The triangular solid lines and circular solid lines are much lower than the model with QGAN added to it

of Quantum GAN, which can perform effective error correction even at higher logical error rates.

Additionally, after training the Quantum GAN, we achieved a logical fidelity of 99.875% on the surface code decoder. Figure 12 illustrates the variation in logical fidelity of the Quantum GAN and local MWPM models at different code distances under varying physical error rates. As shown in the figure, in the initial stage, the logical fidelity for smaller code distances is lower than that for larger code distances. As the error rate gradually increases, the error correction success rate decreases until it reaches a threshold p (represented by the red solid points in the figure), which is a critical error rate used as an indicator of the decoding algorithm's performance. When the error rate is below the threshold p, the logical fidelity increases further with the increase in code distance. However, when the error rate exceeds the threshold p, the logical fidelity decreases with increasing code distance. This indicates that the change in error correction success rate due to code distance is influenced by the threshold. Overall, the Quantum GAN model outperforms the traditional MWPM algorithm in terms of logical fidelity performance, with a fidelity threshold of P = 0.1978 for the Quantum GAN decoder, compared to P = 0.1024 for the local MWPM model.

The experimental results show that the local MWPM algorithm, in the context of traditional error correction, fails to fully account for the strong correlations between the vertices and lattice points in the surface code topology, leading to its limitations in error correction. In contrast, the trained Quantum GAN is highly sensitive to classification, effectively exploiting the strong correlation characteristics to significantly improve the error correction threshold and logical fidelity under physical error rates. By integrating Trans-

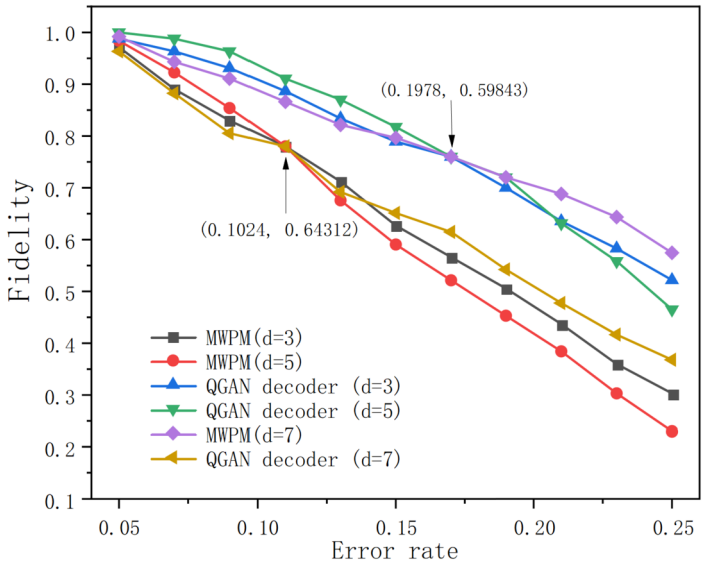

Figure 12 The figure represents the logical fidelity of the training model as a function of the physical error rate, where the horizontal axis represents the physical error rate and the vertical axis represents the logical fidelity. The MWPM and QGAN model thresholds are 0.1024 and 0.1978 at code distances $d = 3$, 5, and 7, which are indicated by orange-red-gray solid lines and blue-green-purple solid lines, respectively

former network decoding, which combines all input syndromes, the accuracy of surface code error correction can be significantly enhanced.

5 Discussion

In this work, we employ a Quantum-Classical Hybrid GAN model to train and obtain an effective correction path for the rotated surface code, using a Transformer network decoder to fully integrate global information, thereby significantly enhancing decoding performance. Moreover, under a phenomenological noise model, the QGAN as a surface code predictor achieves a decoding accuracy of 99.875% and an error correction threshold of 7.5%, compared to 65% accuracy for the local MWPM algorithm, marking a significant breakthrough. Furthermore, the Quantum GAN decoder model achieves a fidelity threshold of P = 0.1978, compared to the logical fidelity of P = 0.1024 for the traditional MWPM algorithm, further improving the fault tolerance of quantum error correction and demonstrating the immense potential of quantum machine learning in the field of quantum error correction. One direction for future research is to explore the integration of QGAN with other topological quantum error correction codes (e.g., color codes and surface codes), while optimizing the efficiency and accuracy of quantum decoders to achieve higher fidelity in a broader range of quantum computing applications. Although our study shows that QGAN can achieve threshold decoding for surface codes, there remains a gap compared to the theoretical optimal threshold. With the ongoing development of quantum machine learning, Quantum GAN-based decoders are poised to become a reliable tool for quantum error correction, and hybrid quantum error correction schemes using Quantum GAN and Transformer decoding algorithms open new avenues in the field of error correction.

In this paper, the quantum circuit is treated as a black box, assuming that it does not introduce errors, which enables the implementation of error correction in quantum com-

puting. This assumption implies that we do not directly address potential noise and errors within the QGAN, but instead rely on external error detection and correction for the generated results. However, the current QGAN models may still be affected by quantum noise at this stage. The realization of this assumption and its application to error correction in practical quantum computing may require integrating quantum machine learning, error correction techniques, and innovative methods in quantum generative models, warranting further research and exploration in this area. With advancements in quantum hardware, further analysis and validation of QGAN's application on practical quantum computers, particularly in the domain of quantum error correction, may be possible in the future.

Appendix A: A Parametric Quantum Circuit (PQC)

The variational quantum circuit (PQC) is usually organized as a quantum neural network, which is a circuit consisting of a series of parameterized quantum gates, where the parameters of the quantum gates in the circuit are controlled by θ, and these parameters can be adjusted by classical optimization algorithms in order to minimize the objective function. The PQC generally consists of an initial state, quantum gates with the parameter θ, and a measurement operation. The initial state is $|0\rangle^{\otimes n}$, and this initial quantum state will be used as the input of the PQC. The initial quantum state can be transformed into a new state by continuously adjusting the parameter θ of the quantum gate through the optimization algorithm, until the convergence condition is satisfied. Finally, the expectation value of the quantum circuit, obtained through the measurement operation, can be expressed as:

$$E(\theta) = \langle 0 \mid U^{\dagger}(\theta)\hat{M}U(\theta) \mid 0\rangle \tag{19}$$

where U represents the youngest positive transformation of the gate, M represents the measurement operation of the expectation value, and θ represents the parameter set of a series of quantum gates. The optimization objective of a quantum circuit is often to solve a specific task by finding the optimal set of gate parameters θ such that the expectation value $E(\theta)$ is minimized or a specific objective condition is satisfied. The optimization process of PQC is shown in Fig. 13.

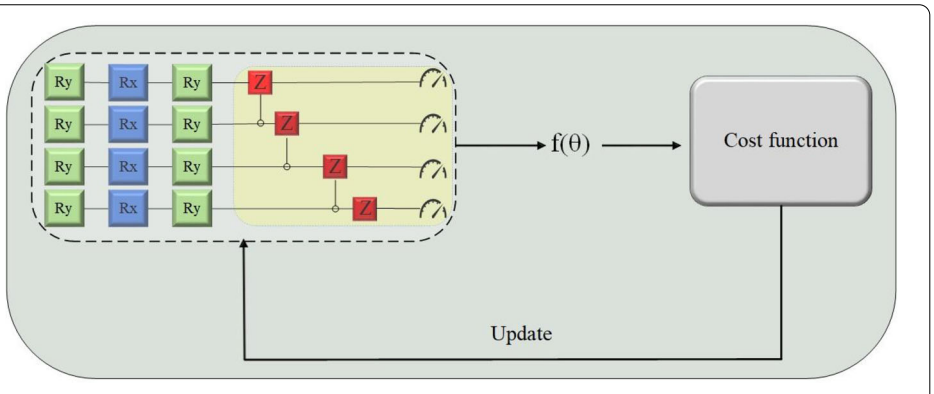

Figure 13 Optimization process of PQC

Appendix B: Gate-model quantum neural networks

In recent years, the advancement of quantum computing has made gate-model quantum neural networks (Gate-QNNs) a viable research direction, particularly in the context of near-term quantum computing architectures. [49, 50]Gate-based quantum computers implement quantum state transformations using quantum gates, where each quantum gate corresponds to a unitary operation. By applying a sequence of quantum gates to an input state and performing measurements, a class of neural networks based on the gate model can be constructed.

In Gate-QNNs, neurons are composed of a set of quantum gates and parameterized weights that connect them, with the core of the network being a quantum circuit formed by unitary operations. The input to the network typically consists of a computational basis state and auxiliary qubits, which serve as readout states during the measurement phase. The training process aims to adjust the parameters of the quantum gates such that the predicted labels closely match the true labels, minimizing the loss function. This optimization process is analogous to classical deep learning training but leverages quantum-specific mechanisms such as coherence and entanglement.

A $\mathrm{QNN_{QG}}$ refers to a quantum neural network (QNN) designed for a gate-model quantum computer, utilizing a specific quantum gate structure QG, as shown in the gate architecture in Fig. 14. This network incorporates quantum connections between unitary operations and classical channels for transmitting classical side information. Within a $\mathrm{QNN_{QG}}$, quantum information flows exclusively in a forward direction from input to output, whereas classical side information has the flexibility to move both forward and backward throughout the network. Additionally, no historical data regarding past execution sequences of the structure is accessible within a $\mathrm{QNN_{QG}}$.

A $\mathrm{QNN_{QG}}$ is formulated by a collection of L unitary gates, such that an i-th, $i = 1, \ldots, L$ unitary gate $U_i(\theta_i)$ is

$$U_i(\theta_i) = \exp(-i\theta_i P) \tag{20}$$

where P is a generalized Pauli operator formulated by a tensor product of Pauli operators $[X, Y, Z]$, while θ_i is referred to as the gate parameter associated with $U_i(\theta_i)$.

In $\mathrm{QNN_{QG}}$, a given unitary gate $U_i(\theta_i)$ sequentially acts on the output of the previous unitary gate $U_{i-1}(\theta_{i-1})$, without any nonlinearities [14]. The classical side information of $\mathrm{QNN_{QG}}$ is used in calculations related to error derivation and gradient computations, such that side information can propagate arbitrarily in the network structure.

The sequential application of the L unitaries formulates a unitary operator $U(\overrightarrow{\theta})$ as

$$U(\overrightarrow{\theta}) = U_L(\theta_L)U_{L-1}(\theta_{L-1})\ldots U_1(\theta_1) \tag{21}$$

where $U_i(\theta_i)$ identifies an i-th unitary gate, and $\overrightarrow{\theta}$ is the gate parameter vector

$$\overrightarrow{\theta} = (\theta_1, \ldots, \theta_{L-1}, \theta_L)^T \tag{22}$$

At (2), the evolution of the system of $\mathrm{QNN_{QG}}$ for a particular input system $|\psi\rangle, \phi)$ is

$$|Y\rangle = U(\overrightarrow{\theta})|\psi\rangle|\phi\rangle = U(\overrightarrow{\theta})|z\rangle|1\rangle = U(\overrightarrow{\theta})|z, 1\rangle \tag{23}$$

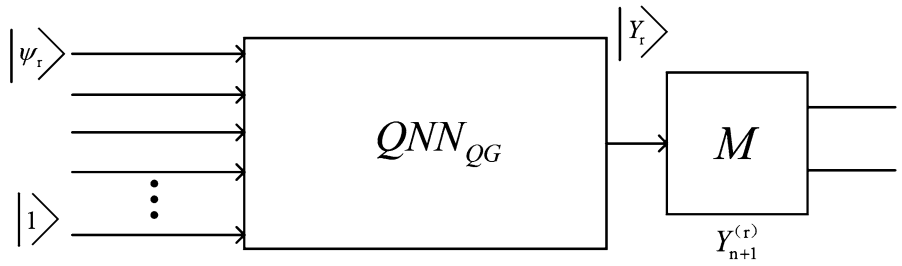

Figure 14 Schematic representation of a $\mathrm{QNN_{QG}}$ in the n-th measurement round

where $|Y\rangle$ is the $(n+1)$-length output quantum system, and $|\psi\rangle = |z\rangle$ is a computational basis state, where z is an n-length string

$$z = z_1 z_2 \ldots z_n \tag{24}$$

where each z_i represents a classical bit with values

$$z_i \in \{-1, 1\} \tag{25}$$

while the $(n+1)$-th quantum state is initialized as

$$|\varphi\rangle = |1\rangle \tag{26}$$

Furthermore, Gate-QNNs have attracted attention for their potential applications in quantum internet environments. By combining quantum computing with quantum communication, gate-model quantum neural networks are expected to play a crucial role in distributed quantum computing architectures. To deepen the understanding of these aspects, recent studies have explored the theoretical limitations of quantum information transfer across quantum channels [51], the integration of quantum learning for memory efficiency in near-term quantum devices [52]. These works provide valuable theoretical and experimental support for the practical development of $\mathrm{QNN_{QG}}$, particularly in the context of hybrid quantum-classical networks and distributed quantum computing scenarios.

Appendix C: Methodological details and parameter settings for model training

In the classical generative task, the generator samples random probabilities to perform quantum measurements of the ground state. Given that the initial input state is $|0\rangle^{\otimes n}$, only a rotational layer is required, primarily composed of R_x and R_y gates. In our simplified quantum generator, after encoding via R_y gates, we apply an additional R_x gate to each qubit, i.e., $U_R = \prod_{i=1}^{n} R_x(\theta_l^i)$. This reduces the number of parameters and resource usage significantly compared to more complex rotational layers.

To explore the effect of two-qubit gates, we tested CZ, CNOT, and ISWAP gates. Notably, CNOT performed worse than the other two. The best results were achieved with the CZ gate, yielding a fidelity of 0.946, suggesting better trainability of entanglement. Given

the lower hardware overhead, we propose using duplicated CZ gates in the model for entanglement.

All sub-generators in model follow the same structure, with parameterized quantum circuits (PQC) on five qubits. Each sub-generator's circuit, noisily encodes data into angles via R_y and R_x gates, with adjacent qubits entangled through CZ gates. After several layers, parametric weights are optimized, and measurements on ancilla qubits perform nonlinear transformations. This design reduces PQC depth, mitigating hardware errors, minimizing parameter counts.

From the input $|0\rangle\widehat{\otimes}n$ to the generator, we obtain the state by encoding the circuit:

$$|L\rangle = R_Y^1(l_1)R_Y^2(l_2)\ldots R_Y^N(l_N)|0\rangle^{\otimes N} \tag{27}$$

Where $R(\lambda,\theta,\phi)$ can be denoted as $R_z(\lambda)R_y(\theta)R_z(\phi)$, the aim is to change a quantum bit into an arbitrary quantum state on the Bloch sphere using the Mississippi transformation. This youngest-positive transformation we denote by a $U(\lambda,\theta,\phi)$, and the quantum state of the ith subgenerator after passing through the youngest-positive operation evolves as: $|0\rangle$

$$|\psi_i\rangle = U_L(\lambda_i,\theta_i,\phi_i)|L\rangle \tag{28}$$

In general, here we choose the measurement operator $M = (|0\rangle\langle 0|)^{\otimes A}$ for the projection measurements, so that the resultant quantum state $|\psi_v\rangle$ after tracking the Ancilla quantum bit is:

$$|\psi_U\rangle = \mathrm{Tr}_A\left(\frac{(|0\rangle\langle 0|)^{\otimes A}\otimes \mathrm{I}|\psi_i\rangle\langle\psi_i|}{\langle\psi_i|(|0\rangle\langle 0|)^{\otimes A}\otimes \mathrm{I}|\psi_i\rangle}\right) \tag{29}$$

For parameter updates, we utilize the gradient descent algorithm, traditionally used in neural networks, to derive gradients of the loss function. However, in quantum circuits, the gradient information is complex and often treated as a black box. To compute partial derivatives within variational quantum circuits, we apply the parameter shift rule, modifying the parameters of the PQC twice. The partial derivative is approximated by:

$$\frac{\partial L}{\partial\theta} \approx \frac{L(\theta+\delta)-L(\theta-\delta)}{c} \tag{30}$$

Here, δ represents a small offset (bounded by $\frac{\pi}{2}$) applied to circuit parameters, while c depends on the specific quantum gate. The change in loss function values before and after the shift allows us to estimate the gradient for the PQC within the same quantum circuit. Denoting the total parameters of our generator as n, we express the gradient of the calculated parameter j of the i-th sub-generator with respect to the loss function $f(\theta)$ accordingly.

$$\begin{aligned}\frac{\partial f(\theta,\lambda)}{\partial\theta_{i,j}} = \frac{1}{2}\langle f(\left[\theta_{1,1},\ldots,\theta_{i,j}+\frac{\pi}{2},\ldots,\theta_n\right],\lambda)\rangle \\ -\langle f(\left[\theta_{1,1},\ldots,\theta_{i,j}-\frac{\pi}{2},\ldots,\theta_n\right],\lambda)\rangle\end{aligned} \tag{31}$$

Table 1 Summary of Discriminator Parameters and Settings

Layer	Filter Size	Output Size
Conv1	5×5 / 2	64×64×64
Conv2	3×3 / 2	32×32×128
Conv3	3×3 / 2	16×16×256
Conv4	3×3 / 1	8×8×512
Conv5	1×1 / 1	8×8×256
Conv6	1×1 / 1	8×8×128
Resnet1	3×3 / 1	8×8×64
Resnet2	3×3 / 1	8×8×64
Resnet3	3×3 / 1	8×8×128
FC	-	1

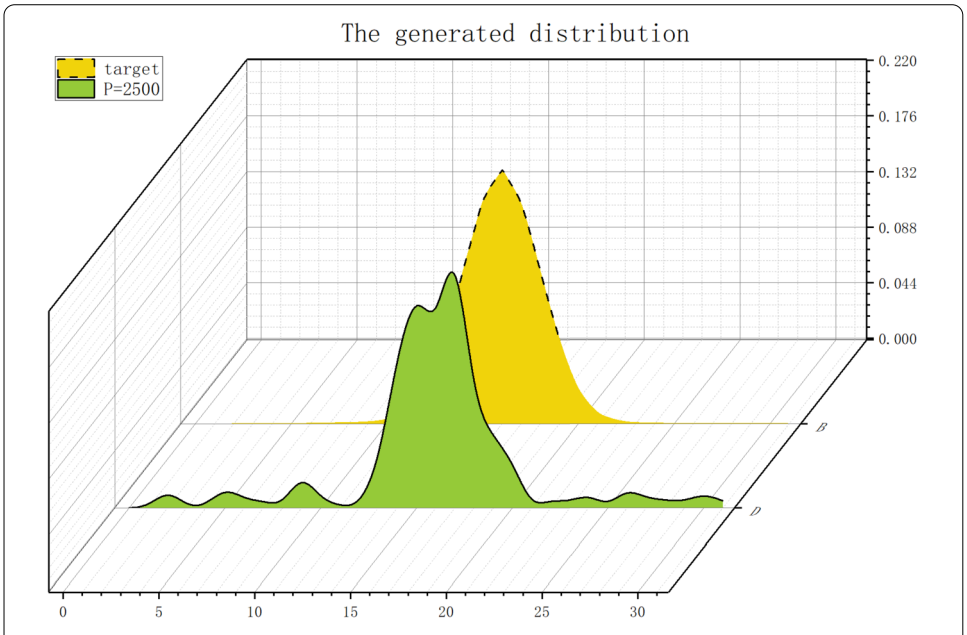

Figure 15 The figure shows the simulation results of approximating a discrete Gaussian distribution for observations taking P = 2500. The performance of the approximated discrete Gaussian distribution is shown, demonstrating how the distribution of the generated data evolves towards the closest true distribution

The approximation using a discrete Gaussian distribution defined as:

$$\pi(a;\nu,\tau) = \frac{\exp\left(-\frac{(a-\nu)^2}{2\tau^2}\right)}{Q} \tag{32}$$

Here, $a \in [0,31]$ denotes a discrete variable, Q serves as a normalization factor, and the objective is to express the quantum state $|\pi\rangle$ as a discrete Gaussian distribution, ensuring that the probability observed under the ground state $|b\rangle$ aligns closely with a Gaussian distribution. In the generator, the quantum state $\phi(\gamma)$ generated is expressed as:

$$|\phi(\gamma)\rangle = \prod_{m=1}^{J} V_m(\gamma)|0\rangle^{\otimes 5} \tag{33}$$

Here, V_m denotes the parameterized quantum circuit (PQC), which is optimized by adjusting parameters γ to approximate the target state. During training, the number of mea-

Table 2 Summary of Transformer Model Parameters and Settings

Parameter/Setting	Description
Transformer Layers	6 (for both encoder and decoder)
Embedding Dimension	64
Number of Attention Heads	4
Feedforward Network Dimension	256
Dropout Rate	0.1
Regularization Parameter (L2)	0.0002
Training Dataset Size	10^5 samples
Baseline Error Rate	1.2%
Learning Rate Step Size	0.1% per update
Batch Size	32
Optimization Algorithm	Adam optimizer
Loss Function	Binary cross-entropy
Training Details	
Forward Propagation	Computes output and loss
Backpropagation	Updates weights to minimize loss
Total Training Iterations	500
Implementation Libraries	PyTorch PennyLane

surements, P, is fixed at 2500, indicating that 2500 measurements are performed per training step to acquire the quantum state's probability distribution. Figure 15 demonstrates the simulation results for approximating the discrete Gaussian distribution, showcasing the evolution of the generated data distribution toward the true distribution. Subsequently, the Transformer network model was utilized for decoding, producing an output feature vector that represents the density matrix of 1 s and −1 s. Moreover, in the search for the optimal adjacency [53], determining the optimal code distance [54] between erroneous qubits is essential.

Acknowledgements
Project supported by the Natural Science Foundation of Shandong Province, China (Grant Nos. ZR2021MF049), Joint Fund of Natural Science Foundation of Shandong Province (Grant Nos. ZR2022LLZ012, ZR2021LLZ001), Key R&D Program of Shandong Province, China (Grant Nos. 2023CXGC010901).

Author contributions
Cewen Tian: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation,Writing Zaixu Fan: Investigation, Software, Validation, Formal analysis Rui Wang: Investigation, Data Curation, Resources, Software Xiaoxuan Guo: Visualization, Supervision, Formal analysis Yanbing Tian:Methodology, Formal analysis, Supervision, Project administration, Funding acquisition. Cewen Tian: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing Zaixu Fan: Investigation, Software, Validation, Formal analysis Rui Wang: Investigation, Data Curation, Resources, Software Xiaoxuan Guo: Visualization, Supervision, Formal analysis Yanbing Tian:Methodology, Formal analysis, Supervision, Project administration, Funding acquisition.

Data availability
No datasets were generated or analysed during the current study.

Code availability
Simulation code can be obtained from the author upon reasonable request, or refer: Open Source Code Provided:https://github.com/wen374/qgan-transform-decoder.

Declarations

Disclosures
The datasets generated during and analysed during the current study are available from the corresponding author on reasonable request.

Competing interests
The authors declare no competing interests.

Author details
[1]School of Information and Control Engineering, Qingdao University of Technology, Qingdao, China. [2]School of Science, Qingdao University of Technology, Qingdao, China.

Received: 24 December 2024 Accepted: 9 June 2025 Published online: 19 June 2025

References

1. Cao S, et al. Generation of genuine entanglement up to 51 superconducting qubits. Nature. 2023;619:738–42.
2. Jain S, et al. Penning micro-trap for quantum computing. Nature. 2024;627:510–4.
3. Gebhart V, et al. Learning quantum systems. Nat Rev Phys. 2023;5:141–56.
4. Katsuda M, Mitarai K, Fujii K. Simulation and performance analysis of quantum error correction with a rotated surface code under a realistic noise model. Phys Rev Res. 2024;6:013024.
5. Hindlycke C, Larsson JÅ. Single-qubit rotation algorithm with logarithmic Toffoli count and gate depth. Phys Rev Res. 2024;6:L042027.
6. iOlius AD, Martinez JE, Fuentes P, Crespo PM. Performance enhancement of surface codes via recursive minimum-weight perfect-match decoding. Phys Rev A. 2023;108:022401.
7. Chan T, Benjamin SC. Actis: a strictly local union–find decoder. Quantum. 2023;7:1183.
8. Burés J, Larrosa I. Organic reaction mechanism classification using machine learning. Nature. 2023;613:689–95.
9. Behera J, et al. Prediction based mean-value-at-risk portfolio optimization using machine learning regression algorithms for multi-national stock markets. Eng Appl Artif Intell. 2023;120:105843.
10. Gyongyosi L, Imre S. Training optimization for gate-model quantum neural networks. Sci Rep. 2019;9:12679.
11. Dalzell AM, Hunter-Jones N, Brandão FG. Random quantum circuits transform local noise into global white noise. Commun Math Phys. 2024;405:78.
12. Wang H, et al. Target-generating quantum error correction coding scheme based on generative confrontation network. Quantum Inf Process. 2022;21:280.
13. Zhu G, Jochym-O'Connor T, Dua A. Topological order, quantum codes, and quantum computation on fractal geometries PRX. Quantum. 2022;3:030338.
14. Sabo E, et al. Weight-reduced stabilizer codes with lower overhead PRX. Quantum. 2024;5:040302.
15. Lami G, Collura M. Unveiling the stabilizer group of a matrix product state. Phys Rev Lett. 2024;133:010602.
16. Gong L-H, et al. Novel semi-quantum private comparison protocol with Bell states. Laser Phys Lett. 2024;21:055209.
17. Cui Y, et al. Extending bandwidth sensitivity of Rydberg-atom-based microwave electrometry using an auxiliary microwave field. Phys Rev A. 2023;107:043102.
18. Ye M, et al. Distributed Nash equilibrium seeking in games with partial decision information: a survey. Proc IEEE. 2023;111:140–57.
19. Zhang Y, et al. On the properties of Kullback-Leibler divergence between multivariate Gaussian distributions. Adv Neural Inf Process Syst. 2024;**36**.
20. Bordoni S, Giagu S. Convolutional neural network based decoders for surface codes. Quantum Inf Process. 2023;22:151.
21. Moustafa EB, Elsheikh A. Predicting characteristics of dissimilar laser welded polymeric joints using a multi-layer perceptrons model coupled with Archimedes optimizer. Polymers. 2023;15:233.
22. Pathak D, Kashyap R. Neural Correlate-Based e-Learning Validation and Classification Using Convolutional and Long Short-Term Memory Networks. Trait Signal. 2023;**40**.
23. Qu Y-J, et al. Approximate error correction scheme for three-dimensional surface codes based reinforcement learning. Chin Phys B. 2023;32:100307.
24. Ashwini S, Arunkumar JR, Prabu RT, Singh NH, Singh NP. Diagnosis and multi-classification of lung diseases in CXR images using optimized deep convolutional neural network. Soft Comput. 2024;28:6219–33.
25. Targ S, Almeida D, Lyman K. 2016. Resnet in resnet: generalizing residual architectures. Preprint. Available at arXiv:1603.08029.
26. Siegel A, et al. Adaptive surface code for quantum error correction in the presence of temporary or permanent defects. Quantum. 2023;7:1065.
27. Dua A, Jochym-O'Connor T, Zhu G. Quantum error correction with fractal topological codes. Quantum. 2023;7:1122.
28. Rajput A, Roggero A, Wiebe N. Quantum error correction with gauge symmetries. npj Quantum Inf. 2023;**9**(41).
29. Tsang M. Quantum noise spectroscopy as an incoherent imaging problem. Phys Rev A. 2023;107:012611.
30. Roberts D, Clerk AA. Exact solution of the infinite-range dissipative transverse-field Ising model. Phys Rev Lett. 2023;131:190403.
31. Zhao RS, Ma HY, Cheng T, Wang S, Fan XK. Quantum generative adversarial networks based on a readout error mitigation method with fault tolerant mechanism. Chin Phys B. 2024;33:040304.
32. Deffner S. Energetic cost of Hamiltonian quantum gates. Europhys Lett. 2021;134:40002.
33. Klimov PV, et al. Optimizing quantum gates towards the scale of logical qubits. Nat Commun. 2024;15:2442.
34. Hu P, et al. Student understanding of the Bloch sphere. Eur J Phys. 2024;45:025705.
35. Yu Z, et al. Optimal quantum dataset for learning a unitary transformation. Phys Rev Appl. 2023;19:034017.
36. Mao A, Mohri M, Zhong Y. Cross-entropy loss functions: theoretical analysis and applications. In: International conference on machine learning, PMLR. 2023.
37. Reyad M, Sarhan AM, Arafa M. A modified Adam algorithm for deep neural network optimization. Neural Comput Appl. 2023;35:17095–112.
38. Luo X, Li Q. Human motion recognition based on feature fusion and residual networks. Sci Rep. 2024;14:29097.
39. Rane C, et al. Optimizing performance of feedforward and convolutional neural networks through dynamic activation functions. Evol Intell. 2024;17:4083–93.
40. Xu Y, et al. Bnet: batch normalization with enhanced linear transformation. IEEE Trans Pattern Anal Mach Intell. 2023;45:9225–32.
41. Aleissaee AA, et al. Transformers in remote sensing: a survey. Remote Sens. 2023;15:1860.
42. Fascianelli V, et al. Neural representational geometries reflect behavioral differences in monkeys and recurrent neural networks. Nat Commun. 2024;15:6479.
43. Shen X, et al. Local self-attention in transformer for visual question answering. Appl Intell. 2023;53:16706–23.
44. Wang H, et al. 2023. Transformer-QEC: Quantum Error Correction Code Decoding with Transferable Transformers. Preprint. Available at arXiv:2311.16082.

45. Huang H-L, et al. Experimental quantum generative adversarial networks for image generation. Phys Rev Appl. 2021;16:024051.
46. Tian Y, Zhang Y, Zhang H. Recent advances in stochastic gradient descent in deep learning. Mathematics. 2023;11:682.
47. Izmaylov AF, Lang RA, Yen T-C. Analytic gradients in variational quantum algorithms: Algebraic extensions of the parameter-shift rule to general unitary transformations. Phys Rev A. **104**:062443.
48. Delon J, Desolneux A, Salmona A. Gromov–Wasserstein distances between Gaussian distributions. J Appl Probab. 2022;59:1178–98.
49. Gyongyosi L, Imre S. Circuit depth reduction for gate-model quantum computers. Sci Rep. 2020;10:11229.
50. Gyongyosi L, Imre S. Scalable distributed gate-model quantum computers. Sci Rep. 2021;11:5172.
51. Gyongyosi L, Imre S, Viet Nguyen H. A survey on quantum channel capacities. IEEE Commun Surv Tutor. 2018;20:1149–205.
52. Gyongyosi L, Sandor Imre S. Optimizing high-efficiency quantum memory with quantum machine learning for near-term quantum devices. Sci Rep. 2020;10:135.
53. Chen C, Wang J, Xu Q, Wang J, Li K. Mixed platoon control of automated and human-driven vehicles at a signalized intersection: dynamical analysis and optimal control. Transp Res, Part C, Emerg Technol. 2021;127:103138.
54. Krinner S, et al. Realizing repeated quantum error correction in a distance-three surface code. Nature. 2022;605:669–74.

Publisher's note