

Approximate error correction scheme for three-dimensional surface codes based reinforcement learning

Ying-Jie Qu(曲英杰), Zhao Chen(陈钊), Wei-Jie Wang(王伟杰), and Hong-Yang Ma(马鸿洋)
Citation: Chin. Phys. B, 2023, 32 (**10**): 100307. DOI: 10.1088/1674-1056/acd8a9
Journal homepage: http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

What follows is a list of articles you may be interested in

Thermodynamic limit of the *XXZ* central spin model with an arbitrary central magnetic field

Fa-Kai Wen(温发楷) and Kun Hao(郝昆)
Chin. Phys. B, 2023, 32 (**9**): 090307. DOI: 10.1088/1674-1056/acb426

Commensurate and incommensurate Haldane phases for a spin-1 bilinear-biquadratic model

Yan-Wei Dai(代艳伟), Ai-Min Chen(陈爱民), Xi-Jing Liu(刘希婧), and Yao-Heng Su(苏耀恒)
Chin. Phys. B, 2023, 32 (**9**): 090304. DOI: 10.1088/1674-1056/acac0c

Degenerate polarization entangled photon source based on a single Ti-diffusion lithium niobate waveguide in a polarization Sagnac interferometer

Yu Sun(孙宇), Chang-Wei Sun(孙昌伟), Wei Zhou(周唯), Ran Yang(杨然), Jia-Chen Duan(端家晨), Yan-Xiao Gong(龚彦晓), Ping Xu(徐平), and Shi-Ning Zhu(祝世宁)
Chin. Phys. B, 2023, 32 (**8**): 080308. DOI: 10.1088/1674-1056/acd8b3

Algorithm for evaluating distance-based entanglement measures

Yixuan Hu(胡奕轩), Ye-Chao Liu(刘烨超), and Jiangwei Shang(尚江伟)
Chin. Phys. B, 2023, 32 (**8**): 080307. DOI: 10.1088/1674-1056/acd5c5

Improving source-in-the-middle continuous-variable quantum key distribution using a heralded hybrid linear amplifier

Lei-Xin Wu(伍磊鑫), Yan-Yan Feng(冯艳艳), and Jian Zhou(周健)
Chin. Phys. B, 2023, 32 (**7**): 070310. DOI: 10.1088/1674-1056/ac9b39

Approximate error correction scheme for three-dimensional surface codes based reinforcement learning

Ying-Jie Qu(曲英杰)[1], Zhao Chen(陈钊)[2], Wei-Jie Wang(王伟杰)[1], and Hong-Yang Ma(马鸿洋)[1,†]

[1] *School of Sciences, Qingdao University of Technology, Qingdao* 266033, *China*
[2] *School of Information and Control Engineering, Qingdao University of Technology, Qingdao* 266033, *China*

(Received 8 February 2023; revised manuscript received 16 May 2023; accepted manuscript online 25 May 2023)

Quantum error correction technology is an important method to eliminate errors during the operation of quantum computers. In order to solve the problem of influence of errors on physical qubits, we propose an approximate error correction scheme that performs dimension mapping operations on surface codes. This error correction scheme utilizes the topological properties of error correction codes to map the surface code dimension to three dimensions. Compared to previous error correction schemes, the present three-dimensional surface code exhibits good scalability due to its higher redundancy and more efficient error correction capabilities. By reducing the number of ancilla qubits required for error correction, this approach achieves savings in measurement space and reduces resource consumption costs. In order to improve the decoding efficiency and solve the problem of the correlation between the surface code stabilizer and the 3D space after dimension mapping, we employ a reinforcement learning (RL) decoder based on deep Q-learning, which enables faster identification of the optimal syndrome and achieves better thresholds through conditional optimization. Compared to the minimum weight perfect matching decoding, the threshold of the RL trained model reaches 0.78%, which is 56% higher and enables large-scale fault-tolerant quantum computation.

Keywords: fault-tolerant quantum computing, surface code, approximate error correction, reinforcement learning

PACS: 03.67.–a, 87.64.Aa, 03.67.Pp **DOI:** 10.1088/1674-1056/acd8a9

1. Introduction

Quantum error correction (QEC),[1–4] which has gained popularity in recent years, is now regarded as the procedure in quantum computing that requires the most time and resources. However, given the current strategies for quantum computing, QEC is an effective means of reliable quantum computing and storage as well as protecting quantum information from loss. The requirement for QEC arises from the quantum systems' inescapable connection to their surroundings, which causes qubits to change state (decoherence).[5,6] To mitigate the effects of quantum decoherence, it mainly controls error propagation and maintains a low error rate through active error correction and fault tolerance mechanisms, thereby achieving good local stability.[7–10] Topological properties of subsystems have become an important resource for building better and more robust quantum error correction codes. Due to the relatively low overhead and locality requirements of surface codes, as well as the availability of practical strategies for implementing the necessary logic gates, topological quantum codes[11–13] particularly only need local operations to diagnose and correct errors. This makes them a particularly promising candidate for large-scale fault-tolerant quantum computing.[14,15]

Typically, naturally occurring physical systems are classified as having approximate or exact symmetries, which can be used to classify matter in equilibrium. The approximations of significant implications are also present in quantum error correction codes. For example, certain energy subspaces are known to form approximations of quantum error correction codes in the context of time-translation-invariant many-body systems. Additionally, error correction codes with suitable approximate properties can be used to protect information from noise.[16] In general, we can say that a quantum operation will be covariant with respect to the group G if it commutes with the group action. When a quantum error correction code is covariant with respect to G, its encoding map is called the G-covariant operation, and we can achieve approximate error correction[17–20] by studying the properties of the recovery operation. A crucial step in quantum error correction is decoding.[21,22] Decoding is a process of identifying and correcting errors in the quantum system due to factors such as the environment and noise to ensure that the quantum state is error-free. This process is typically implemented classically in quantum error correction schemes. It takes to input a set of stabilizer measurements (syndromes) and returns the syndrome operators. If the product of the syndrome and the original error is equivalent to a stabilizer,[23–25] the correction will be successful. If the decoding process takes longer duration than the budgeted error correction time, errors will accumulate, eventually reaching an uncorrectable error state. To address rapid decoding difficulties, machine learning techniques have been employed in various quantum physics domains, and

†Corresponding author. E-mail: hongyang_ma@aliyun.com
© 2023 Chinese Physical Society and IOP Publishing Ltd

different types of neural networks[26–29] have also been studied during this time.

In this paper, we investigate the accuracy of quantum error correction codes concerning continuous approximate covariance. Taking the surface code as an example, we study the properties of approximate quantum error correction of the surface code by combining its properties with the approximation of the quantum system. It is worth noting that to address the spatial correlation problem of surface codes,[30,31] this paper focuses on realizing the jumping of the surface code from the 2D–3D dimension,[32,33] and error detection is performed through the measurement of stabilizer operators, and subsequently, the surface code is reduced back to a 2D lattice for decoding using the reinforcement learning (RL) decoder. This methodology offers the advantage of high scalability for surface codes in 3D space while reducing the complexity and resource costs by saving on stabilizer measurements.

This paper is organized is as follows. In Section 2, we provide a brief introduction to the background knowledge of approximate quantum error correction and surface codes. In Section 3, we design an error correction algorithm for approximate surface codes. We outline the decoding strategy in Section 4 and perform training and simulation analysis in Section 5, followed by our conclusions in Section 6.

2. Background

We begin by briefly explaining the basics of surface codes and RL. However, the framework and method proposed in this work are not limited to surface codes, and we can also apply to other stabilizer codes. This article takes the surface code as an example to make a simplified demonstration, which is also for experimental correlation. This article mainly introduces the application of surface code.

2.1. Surface code

Quantum error correction codes generally have topological properties, which are codes defined on various lattices, and typical topological codes are complex surface codes.[34,35] Surface codes are a class of 2D quantum error correction codes that can have periodic boundaries, while toric codes are a subclass of surface codes that are defined on a torus with periodic boundaries. Place qubits on a 2D square lattice, with each vertex corresponding to a qubit. The codespace can then be defined by the parity operator applied to the nearest four qubits on the square lattice of $L \times L$. Lattice and vertex operators can form abelian groups or stabilizer groups. Since qubits can only interact with their nearest neighbors, stabilizer generators with local properties are required.

We first consider a 2D surface code. The surface code, introduced by Kitaev, defined on the square lattice of a torus with a qubit on each side. All stabilizers are commutative and have eigenvalues of ± 1. The logical operators are operators that preserve the code space, the logical X (Z) operator is expressed as $X_{\mathrm{L}} = \bigotimes_{i \in v} X_i$, $Z_{\mathrm{L}} = \bigotimes_{i \in f} Z_i$. They are contiguous strings of single-vertex X (Z) operators that connect the top and bottom (left and right) boundaries of the lattice, and the code is defined as the ground space of the Hamiltonian

$$H = -\sum_{v \in V} X_v - \sum_{f \in F} Z_f. \tag{1}$$

Here data qubits are associated with edges, corresponding to each vertex on the lattice. There is an operator X_v associated with each vertex v of the lattice and a patch operator Z_f associated with each face f. X_v is the product of Pauli-X matrices acting on the edges associated with v, i.e., $X_v = \prod_{e \in v} X_e$, and $Z_f = \prod_{e \in f} Z_e$ is the product f of Pauli-Z acting on all edges of the face. The code space is defined as the simultaneous "+1" eigenstates of these operators X_v and Z_f. These operators and any product of them are called stabilizers[23–25] of the code and form a stabilizer group S.

2.2. Reinforcement learning

Reinforcement learning (RL) is a framework that can be articulated precisely with the old adage learning through experience.[36] In RL, the environmental framework for controlling physical systems and the selection of agents to apply system control define the control problem, as shown in Fig. 1. The agent operates on the environment and performs a series of actions to solve certain problems.[37,38] Each time step t can be represented by a state $s_t \in S$, where S is the state space, showing how the environment is represented. The feedback loop between the agent and the environment is called a Markov decision process (MDP).[39]

Generally speaking, we describe the agent's π (or policy) as a mapping from states to particular probabilities, that is, $\pi(a \mid s)$ is the environment in the state $A_t = a$, the probability that the agent chooses $S_t = s$ is formalized. By using a measure of cumulative reward, the value of any given state depends not only on the immediate reward from the particular policy stated in the following, but also on the expected reward in the future. The optimal Q value for convergence[36] can be determined if the agent is trained with an unlimited number of times, and the best strategy is simple to find the given best Q-function. The best course of action in a given state s can be easily determined by selecting the action $a = \mathrm{argmax}_{a'}[q_*(s, a')]$. Deep Q-learning (DQL) uses deep convolutional networks,[40,41] when encountering an unknown state, DQL compares its global features with those from experience.[42]

We use DQL to train the agent to decode irrelevant bits on approximate 3D surface codes or phase flips caused by ambient noise. The training process only terminates after a certain number of events or when the loss function of the convolutional neural network stops decreasing. We also use the training techniques of duel deep Q-learning[43,44] to ensure the stable operation of the training process.

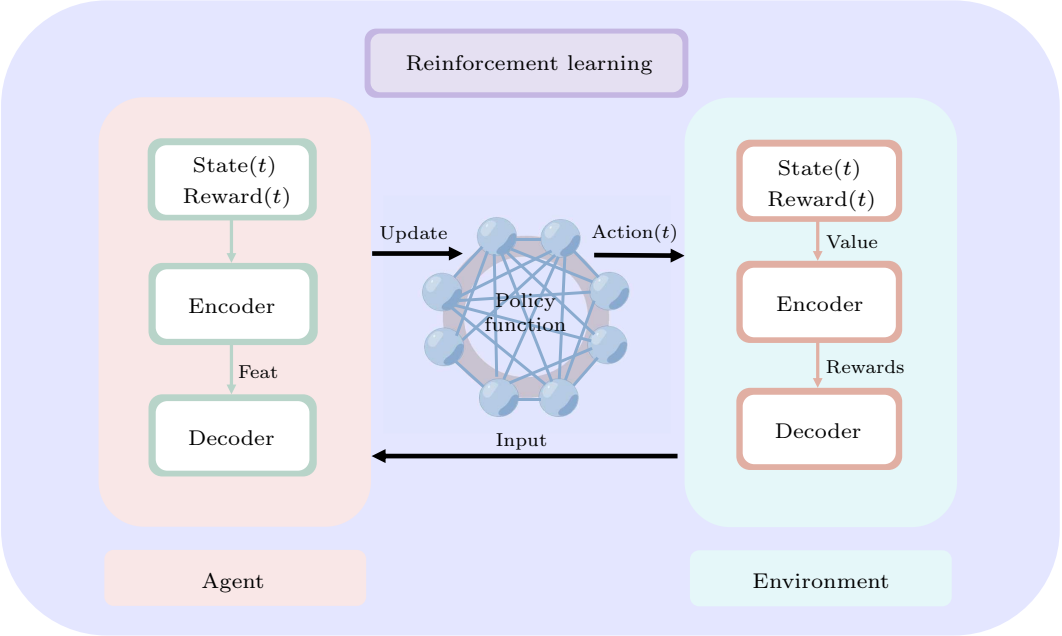


Fig. 1. The agent's objective is to maximize its total reward value throughout the decision period as an illustration of the signals passing between the agent and the environment for subsequent turn-based durations.

3. Algorithmic process

Single decoding is not possible for 2D topology codes, so measurement errors must be counteracted by repeating stabilizer measurements multiple times and preparing $O(d)$ repetitions in a distance d code for error tolerance.[32] To improve the strong resistance to the dullness of encoded qubits, we take advantage of the scalability property of topological error correction codes to encode qubits into a 3D cubic lattice. Although such encoding can be a candidate for quantum memory, it does not enable self-correction. Thus the key issue of how to effectively correct errors on 3D codes should be addressed.

3.1. Approximate surface code

As mentioned in the introduction section, in the case of time-shift-invariant many-body systems, it is known that certain energy subspaces can form approximate quantum error correction codes that are preserved under time evolution. More specifically, the mapping embeds a logical code space in the tensor product of n physical subsystems. The unitary transformation acting on the code space can be used to realize the tensor product of n unitary transformations, which acts on each subsystem. We study the approximate error correction of quantum error correction codes with respect to symmetry covariance, when errors occur due to the loss of one or more subsystems out of n, we can identify which subsystems are lost and approximate error correction can be achieved by studying the properties of recovery operations.

We define a completely positive trace-preserving mapping X which assigns each logical state that exists on one or more logical systems L to the corresponding state on a physical system A consisting of n subsystems $A = A_1 \otimes A_2 \otimes \cdots \otimes A_n$. We assume that errors occur randomly in some subsystem A_i and that i is known, so that the recovery mapping may depend on i. Consider the code that maps any purely logical state $|x\rangle_{\mathrm{L}}$ to the physical state $\{|\psi_x\rangle_A\}$, we call the latter the code word, and all the code words $\{|\psi_x\rangle_A\}$ constitute the total code space. In general, we consider codes that are isometric, but in some special situations, a more general coding mapping is also taken into account.[18]

To investigate the approximate error correction performance of the code, we quantify the approximate error correction by using the distance metric between states and channels. The fidelity between quantized quantum states using trace distances is implemented to quantify the proximity of the quantum channel $\mathcal{K}$ to the constant channel. These two standard measures are the maximum mixed input state entanglement fidelity and the worst-case entanglement fidelity, respectively,

$$F_{\mathrm{e}}^2(\mathcal{K}) = \langle\hat{\phi}|(\mathcal{K}\otimes \mathrm{i}d)(|\hat{\phi}\rangle\langle\hat{\phi}|)|\hat{\phi}\rangle, \tag{2}$$

$$F_{\mathrm{worst}}^2(\mathcal{K}) = \min_{|\phi\rangle}\langle\phi|(\mathcal{K}\otimes \mathrm{i}d)(|\phi\rangle\langle\phi|)|\phi\rangle, \tag{3}$$

where d is the subsystem dimension, for a code that admits universal transversal logical gates. The input state in the definition of F_{e} is $|\hat{\phi}\rangle_{LR} = \sum_{k=0}^{d_{\mathrm{L}}-1}|k\rangle\otimes|k\rangle/\sqrt{d_{\mathrm{L}}}$, i.e., the maximum entangled state of the systems L and R, and the systems R and L have the same dimension, which is denoted by d_{L}. The optimization range in the definition of F_{worst} contains all dichotomous states of L and R.

As shown in Fig. 2, to construct the approximate error correction model for the surface codes, we further define the approximate surface code by Eqs. (2) and (3) and represent the

way in which the approximate surface code interacts with the quantum channel using the following equations:

$$F_{\mathrm{e}}^{2}(\mathcal{K}) = \langle\hat{\phi}_s|(\mathcal{K}\otimes 2\mathrm{i})(|\hat{\phi}\rangle_s\langle\hat{\phi}|_s)|\hat{\phi}\rangle_s, \tag{4}$$

$$F_{\mathrm{worst}}^{2}(\mathcal{K}) = \min_{|\phi\rangle_s}\langle\phi|_s(\mathcal{K}\otimes 2\mathrm{i})(|\phi\rangle_s\langle\phi|_s)|\phi\rangle_s. \tag{5}$$

The input state defined in F_{e} is $|\hat{\phi}\rangle_s = (|X\rangle\otimes|X\rangle + |Z\rangle\otimes|Z\rangle)/2$. We obtain the cubic lattice after mapping the 2D surface code dimension to the approximate 3D code, and further characterize the strong noise immunity of the 3D lattice qubits.

In a 2D surface code, both data qubits and ancilla qubits are situated on the planar lattice. However, in the approximate 2D surface code, data qubits are placed at the center of the lattice while the ancilla qubits are assigned to the edges of the lattice. To move the data qubit to the center of the lattice, we need to first measure the four surrounding stabilizers, and then verify each data qubit with its corresponding four stabilizers. Finally, based on the measurement results of the corresponding Pauli Z gate operation to complete the movement. This allocation scheme transforms the surface code into an approximate surface code and allows for standard surface code operations on the dual double lattice. Therefore, by performing a flat lattice transformation on the original surface code, an approximate surface code can be obtained, and standard surface code operations can be performed on the dual double lattice.

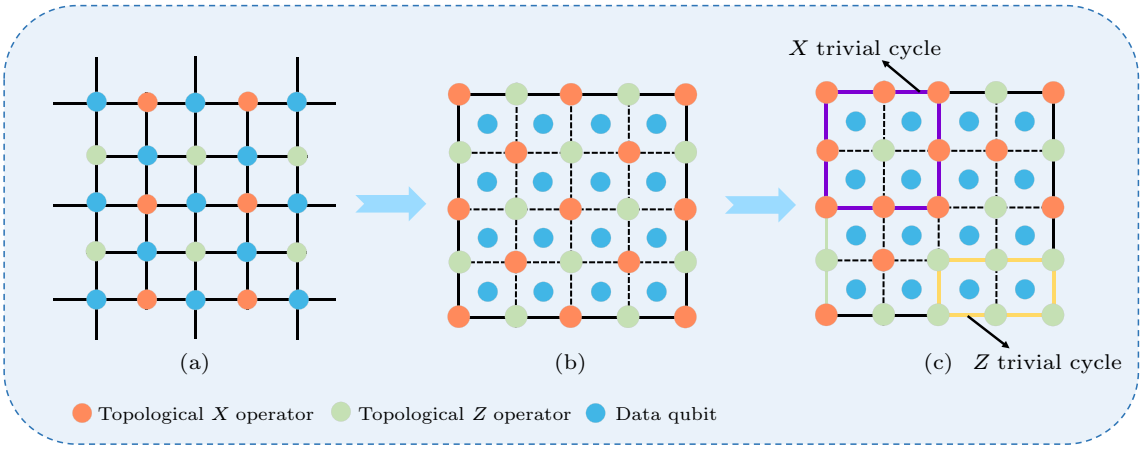


Fig. 2. Generic components of surface code. (a) A 2D surface code, the auxiliary qubits that measure the stabilizers are orange and green display, data qubits are shown in blue. (b) Approximate the 2D surface code, the dotted line represents the dual double lattice behind the flat lattice. (c) The topological operators X and Z each form a chain of topological corrections on the dual lattice (purple and yellow in the figure).

3.2. Dimension mapping

The ability to exchange 3D and 2D surface codes (a process called dimensional mapping[32,33]) is at the heart of Brown programs. We consider transforming the approximate 2D surface code into the 3D surface code by dimension mapping. Compared to the 2D code, the 3D surface code exhibits greater scalability due to its three-dimensional lattice structure, which enables error correction to be performed by measuring the parity of stabilizer operators acting on each face, thereby demonstrating a higher degree of redundancy. This increased redundancy allows for a higher threshold error rate in the 3D surface code, meaning that it can tolerate a greater level of noise before errors become uncorrectable. Moreover, the 3D surface code allows for more efficient error correction, as it only requires the measurement of stabilizers relevant to the current dimension after mapping, reducing the space required for stabilizer measurements and the number of ancilla qubits needed for error correction, thereby reducing resource overhead during the error correction process.

Converting approximate 2D surface codes to 3D codes still encodes only one qubit, so we must employ additional methods to ensure that the final logical state of the 3D codes is the same as the initial logical state of the 2D codes. Meanwhile, we apply X stabilizer and logical gate operations to the code state at each time step in specific situations, ensuring that the dimension mapping only involves measurements of Z stabilizers. As X stabilizers may not commute with logical gate applications, the 2D code is not necessarily in the eigenstate of X stabilizers during the execution of operations.

As shown in Fig. 3, considering the realization of the dimensional mapping operation from the approximate 2D surface code to the 3D code, we start with two approximate 2D surface codes and entangle them through the measurement of the intermediate stabilizer to form an approximate 3D surface code perspective picture. At the same time, the stabilizer measurement is commutative, so we do not need to do anything to ensure an accurate conversion of logical Z from 2D code to 3D code. In order to successfully transfer the approximate 2D code to the 3D code, we must apply the matched logical X to the X (gray dashed line) of the 3D code exactly.

Dimension mapping will have several situations as shown in Fig. 3: (i) Schematic diagram of the conversion of two approximate 2D surface codes to 3D codes. (ii) There is a partial X error in the front code, the loop connecting to the front

and side border constitutes the red syndrome, correcting these loops and moving them to the front code boundary completes the 2D code state to 3D transition. (iii) Two errors detected on the stabilizers of the front code (gray solid line) and the back code (gray dotted line) made it impossible for us to track and find the inner and outer space of the cycle. This in turn leads to the inability to transfer the state of the 2D code to the 3D code space with overall accuracy.

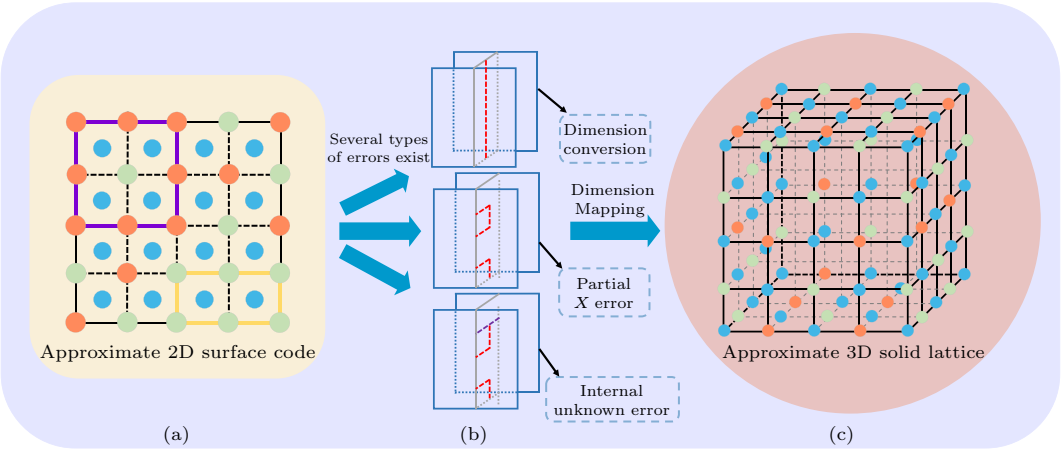


Fig. 3. (a) Approximate 2D surface code in the presence of topologically corrected chains. (b) Several situations exist in the dimension mapping process. (c) Approximate 3D code after dimension mapping.

A suitable dimension mapping is mainly divided into the following steps:

- Starting from a 2D surface code of state $|\bar{\psi}\rangle$ mapped to a 3D surface code $\mathcal{S}$ and its boundary $\partial\mathcal{S}$. The choice of these codes must ensure that the Z stabilizer for the 2D code commutes with the X stabilizer for the 2D and 3D codes.
- Prepare all qubits in $|+\rangle$ that belong to $\mathcal{S}\backslash\partial\mathcal{S}$.
- Measure the Z stabilizer operators for 3D code, not for 2D code.
- Perform different error correction methods according to the type of error that occurs at the code location. For example, if there are some X errors in the code, they will be corrected directly. If two errors are detected, the code cannot be transferred to the 3D code space as a whole accurately.
- The measurement is returned to the code space for error correction, which is not allowed to apply to any qubits in the original 2D code.

The Z stabilizer of the 2D code is combined with the qubit of $|+\rangle$ to ensure that the coding state of the measurement map is divided into two cases: One is the state of the code space in the 3D code, and the other is the state in the code space inferred by the correction from the stabilizer measurements. This correction to eliminate the X error distribution is what Brown calls a 3D code gauge.[45]

3.3. Error correction process

The approximate 2D surface code obtains a cubic lattice after dimension mapping, and the qubits in its internal lattice are locally stable (not prone to errors) from the law of conservation of physical energy, which further characterizes the strong noise immunity of the 3D lattice qubits. The qubit error correction under the 3D lattice only needs to consider the position information of the surrounding surface (check information of the six faces), and relies on the stability of the lattice operator and vertex operator to ensure the feasibility of error correction.

The stabilizer S is the Abelian subgroup of $\{1,-1\}\times\{I,X,Y,Z\}^n$, where $-I^n \notin S$. Assume that S has a set of $n-k$ independent generators $\{S_x\}_{x=1}^{n-k}$. For simplicity, consider $S_m \in \{I,X,Y,Z\}^n$, the binary $[[n,k,d]]$ stabilizer code defined by S is a 2^k dimension subspace in $\mathbb{C}^{2^n}$, and the parameter d is the code's the minimum distance, the elements in S are stabilizers. Any two Pauli operators with n qubits are either commuting or anticommuting. All stabilizers are interchangeable and have eigenvalues ±1. If Pauli error commutes with some stabilizers, measuring these stabilizers will return to eigenvalues -1, and commutation will return to eigenvalues 1. When the stabilizer measurement result is ±1, it will be mapped to $+1\rightarrow0$ and $-1\rightarrow1$, and the resulting $\{S_m\}_{m=1}^{n-k}$ is called the error syndrome (the measurement result of the stabilizer), and the error that the stabilizer can detect is represented by a nonzero error syndrome. Except that stabilizers have no impact on the code space, we do not need to account for stabilizer errors.

In order to improve the modeling of the dimensional jump switching process between two codes, we have simplified the intricate 3D cubic lattice into six separate 2D lattices, each with a size of 5×5. This approach leads to a reduction in the number of ancilla qubits necessary for error correction, as well as a decrease in the number of stabilizer measurements required. As a result, the complexity of surface code coding is significantly reduced, along with the resource overhead in

the error correction process. The error qubits are introduced and differentiated with various colors for the Pauli operators X, Y, and Z, after measuring the correctors and then performing approximate error correction, the specific error correction process is shown in Fig. 4.

In complex 3D lattice structures, this reduces the space required for the stabilizer measurements by only measuring the stabilizers associated with the current slice, rather than measuring all the stabilizers in the entire code. This dimensional mapping approach reduces the overall number of measurements required and can significantly reduce the resource overhead of the error correction process. We infer the results and correct them based on the structure of the stabilizer and guarantee the accuracy of the corrected errors and the logical state of the transmitted code. The approximate 3D surface code state is uniform in the total parity of the measured qubits on X, thus all four qubits may be in the same state, or two in $|+\rangle$ and two in $|-\rangle$. In the first case, we directly map the same logical state of the approximate 2D surface code. In the other case, it should be corrected. The reason is that the Z stabilizers that are projected to the rear when measured on the X stabilizers are randomly assigned, and these stabilizers leave traces that need to be corrected at the initial position of the code. We detect the traces left by the stabilizer in the back code and then apply Z to the qubits in the front code to correct the error.

In addition, when Z errors exist in the front code, it cannot be judged whether it needs to be corrected only by its measurement result. We identified the error (syndrome of 3D code after dimension mapping) by combining the measurements of the anterior code with the 2D code X stabilizers. For example, the parity of the front code violates $X_1X_2X_3X_4$, and we infer the measurement of the side X stabilizers based on the parity of the bottom qubits and the measurement of the back qubits.

When qubits error occurs, the corresponding syndrome is generated and disappears after choosing the correct position. Given the error and the stabilizer element, when the error is the same as the syndrome generated by the measurement, the measurement result of the stabilizer automatically selects the error correction operator, which is called the decoder. The job of the decoder is to find errors in the qubits of data from the error correction subset. Since vertex and lattice operators produce strong spatial correlations, we propose a reinforcement learning decoder based on a DQL algorithm to find the optimal correction chain and to achieve better thresholds by continuously optimizing the conditions.

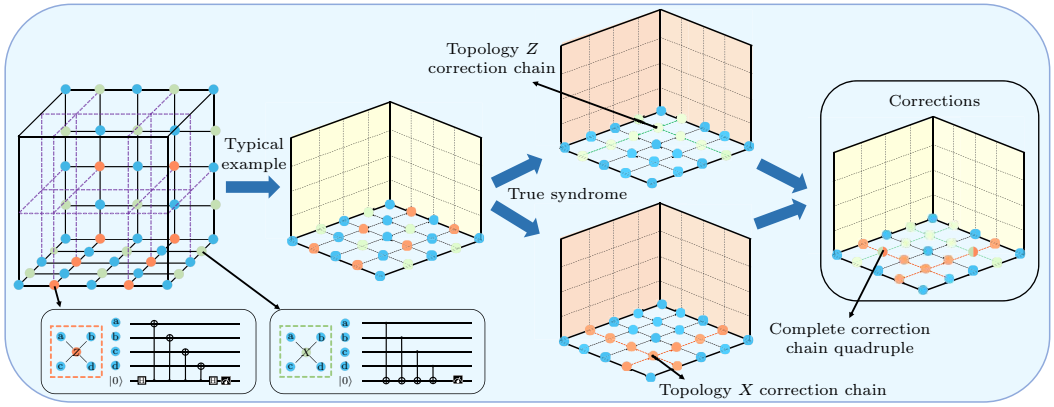


Fig. 4. Error correction diagram for the approximate 3D surface code. Take the bottom code in the three-dimensional cubic lattice as an example to measure the interior of the stabilizer, detect different error syndromes, and perform recovery operations to correct errors. The red and green circles at the bottom correspond to X and Z errors.

4. Decoding strategy

In fault-tolerant quantum computing based on surface codes, known protocols achieve Clifford gate operations through techniques such as lattice surgery, code deformation, and syndrome tracking. However, non-fault-tolerant Clifford gates, such as the T gate, can be fault-tolerantly executed through magic state distillation and gate teleportation. High-quality magic states are obtained through magic state distillation, which only requires Clifford gates and fault-tolerant magic states.[46,47] The primary objective in decoding idle qubits in quantum computing is to effectively suppress logical error rates by applying error correction schemes when the physical error rate of qubits is lower than a certain threshold, which is a crucial measure of fault-tolerant performance. This article introduces and compares two different error noise models for decoding, namely the minimum weight perfect matching (MWPM) decoder[27,42] and the RL decoder.[38,48–50]

4.1. MWPM decoder

To enhance the decoding performance of codes and reduce the overhead of qubits, we employed a universal decoder, the MWPM decoder. By reformulating the noise model into a mathematical model, we assume that the corrections selected in the previous time step were successful and that all stabilizers were in the $+1$ eigenstate. The measurement outcomes of the Z stabilizers directly lead to a random distribution of X errors on new qubits, which are connected to the top or side boundaries via a set of cycles. The decoder matches errors

present in certain stabilizer measurements to generate a set of effective syndromes (a set of lattices). To preserve quantum state information, the corrected qubits should have the same error pattern as the error qubits of stabilizers to avoid the occurrence of other logical errors.

Noise mainly arises from syndromes and stabilizer measurements, as the measurement results of stabilizers are not accurate enough, requiring d rounds of measurement of the cycles to be repeated. Therefore, we apply the MWPM decoder and the Dijkstra algorithm to approximate the matching of the 3D surface code, thereby increasing the probability of successful stabilizer detection. We construct the decoder mainly through the following steps: Firstly, assume that the approximate 3D surface codes and the quantum circuits are noiseless. Secondly, perform a round of measurement cycles without stabilizer noise. Finally, add additional noise environment to continue measuring during the measurement period under stabilizer noise. The noiseless measurement cycle is to ensure that the noise state can be restored to the original code space, thereby determining whether the error correction is successful.

When constructing an approximate 3D surface code, we use the MWPM decoding algorithm for error correction:

- Perform d rounds of stabilizer measurements under noise to approximate the initial state of the surface code and construct the correction graphs by measuring the stabilizers.
- Mark the vertex values of the previous round's stabilizer measurements. If the number of vertices is even, we mark the top vertex, if the number of vertices is odd, we mark the boundary vertex.
- Use the Dijkstra algorithm to find the minimum weight matching for the marked X-type and Z-type vertices, where each vertex is connected to show the optimal weight path.
- According to the law of conservation of physical energy, the qubits in the internal lattice are locally stable, so quantum error correction in the 3D lattice only needs to consider the check information of six faces. We calculate the display count of each X- and Z-type horizontal boundary. If the horizontal boundary is not marked, we perform X- and Z-type corrections.

The decoding problem of quantum surface codes is a complex combinatorial optimization problem. Compared to MWPM, RL decoders better adapt and handle the complex three-dimensional cubic lattice structure by learning and optimizing strategies. The training process of RL decoders may also be scalable for decoding larger quantum surface codes. On the other hand, MWPM decoders typically rely on predefined rules and heuristic algorithms, with lower adaptability and flexibility. Theoretically, RL decoders can handle various types of errors, including bit-flip errors and phase-flip errors, while MWPM decoders require modification or extension for error types other than flip errors. Moreover, when facing high error rates or complex error models, RL decoders can search for the optimal correction strategy through optimization strategies, achieving better decoding performance. RL decoders can also achieve higher efficiency in decoding by learning efficient decoding strategies such as parallel processing or local search. In the following section, we introduce the decoding work of reinforcement learning to better address the decoding problem of quantum surface codes.

4.2. Reinforcement learning decoder

This paper utilizes a decoder based on a neural network agent, which is optimized through RL to observe and gradually establish a recovery chain for the syndrome of the approximate 3D surface code. The agent employs the deep neural network (DNN) and the Q network to determine the actions and Q values of the syndromes. We typically consider discrete problems in which at each time step t, the environment is described by a state $S_t \in \mathcal{S}$, where $\mathcal{S}$ is the state space. Given the environment state, the agent selects an action $A_t \in \mathcal{A}$, where $\mathcal{A}$ is the action space. After the agent selects an action, the environment updates correspondingly, providing feedback to the agent in the form of a reward R_{t+1} and a new state S_{t+1}. Given an initial logical state $|\psi_0\rangle \in \mathcal{H}_{\rm sc}$, the agent's objective is to suppress errors as long as possible so that logical operations can succeed with high probability. The environment is formalized as a Markov decision process (MDP) under finite state and action spaces:

$$p(s', r \mid s, a) := \mathrm{pr}(S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a). \quad (6)$$

The action-value function (also known as the Q-function) for the policy π is defined as

$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] \\ &= \mathbb{E}_\pi\Big[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\Big]. \end{aligned} \quad (7)$$

At the time t, action a is taken and subsequently follows a policy π, where $\gamma \leqslant 1$ is the discount factor. The Q function conceptually resembles the state-value function, except that it provides values for state-action pairs. Additionally, we rank policies based on the value function, that is, $\pi > \pi' \Longleftrightarrow v_\pi(s) > v_{\pi'}(s) \ \ \forall s \in \mathcal{S}$. Meanwhile, we can define the optimal policy π^* in reverse:

$$q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \mid S_t = s, A_t = a]. \quad (8)$$

Given a state s, the optimal policy can be easily obtained by selecting action $a = \mathrm{argmax}\, a'[q_*(s,a')]$. The Q value $q_\pi(s,a)$ parameterized by the neural network is used as the output, and the network parameters are adjusted through the

stochastic gradient descent algorithm to minimize the error between the optimal Q value and the approximate Q value.

The decoding process feeds the algorithm as an input value, where the syndrome is the system state visible to the agent. The syndrome observed at each time step is a result of the cumulative effect of the agent's operations on the syndrome, which is initially randomized by a distribution of bit-flips. Once the system reaches a terminal state with a null syndrome, an odd number of non-trivial loops represent a failure in error correction. However, during the algorithm's usage, information about success and failure can only serve as a metric to evaluate the agent's performance during training. Regardless of whether the correction string requires logical operations, a reward of $r = -1$ is given at every time step until the terminal state is reached. Therefore, compared to the MWPM algorithm, the primary objective of this algorithm's agent is to eliminate syndromes with the fewest possible steps. The process of decoding by a well-trained agent is illustrated in Fig. 5.

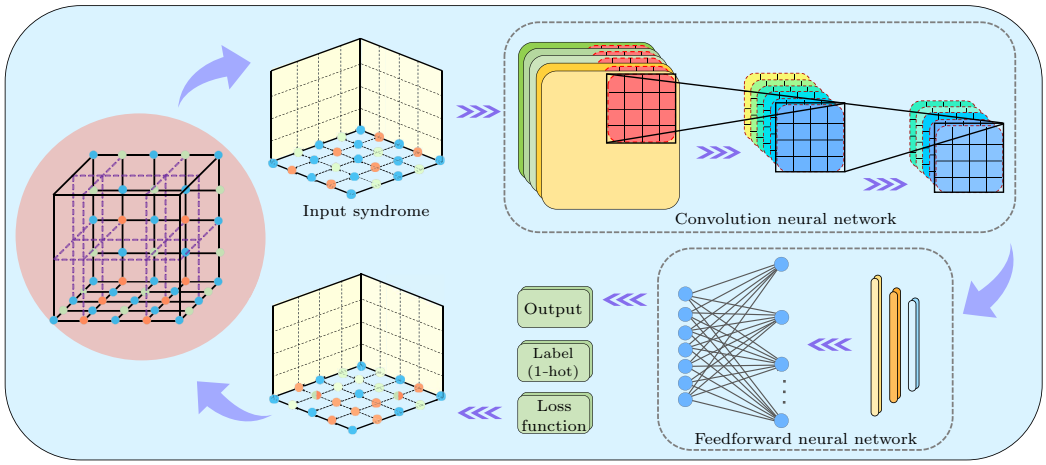


Fig. 5. Details of the deep Q decoding agent. Details of the deep Q decoding agent. The syndrome is encoded into a binary matrix and better fitted to the convolutional layer input, and the fully connected layer is used to complete the mapping from the input feature space to the label set, that is, to achieve the classification effect. Then, the training cost is reduced by training and optimizing the convolutional neural network structure, and the output eigenvalues are decoded by convolution operation. Finally, the feed-forward neural network is used for multi-layer connections to output multiple error chains that you want to correct.

The decoding process described here employs neural networks to distill information between syndromes and uses stepwise decoding to gradually reduce the syndromes to smaller subsets. Specifically, due to the periodic boundary conditions of the encoding, the syndrome can be represented around any plaquette. The Q network takes in a $d \times d$ matrix corresponding to the positions of vertex and plaquette errors. The agent can move any wrong option in any direction, corresponding to a bit flip on the various error physical qubits. The output is a triplet of Q values for X, Y, and Z operations on a specific qubit, and to obtain the complete set of action values for the syndrome, we shift and rotate the syndrome sequentially and locate the position of each qubit. Simultaneously, the complete relevant Q function of the syndrome is obtained by computing the Q function of each individual. After performing the selected action, a new syndrome is generated, and this process is repeated until no errors remain.

When decoding the approximate 3D surface code using reinforcement learning, the specific steps are as follows:

- We have the constituent parts that describe the rules, and when the agent takes action through these rules, the environment generates a tuple $[S_{t+1} = \{s_{\mathrm{sv},t}, h_t\}, r_t, t_t]$.
- The agent can move any wrong option in any direction (up, down, left, and right), corresponding to a bit flip on the various error physical qubits. The number of agent actions changes continuously with the number of errors.
- Use the neural network to represent the Q function, when errors exist in the system, they are separately sent to the Q network, and the Q function is parameterized by adjusting the weights and biases of the neural network, written as $Q(s, a, \theta)$, where θ is the set of network weights and biases.
- Take the Q value of the action in each error, and the new syndrome is sent to the algorithm after selecting the action and error using a greedy strategy. The process is repeated until there are no errors.

Given the fundamental factors of the environment, we define the action space A to consist of all Pauli X and Z flips on a single data qubit and special actions, while Pauli Y flips can be implemented by X and Z flips. In addition, if more qubits are included in the action space, it will have a certain impact on the complexity of the training process. Therefore, we set the agent to operate on single qubits, where in practice all single qubits accumulate incrementally between successive syndromic measurements, while multiple qubits can operate simultaneously and compute whether or not they are tracked. We also note that for a well-trained agent, the initial syndrome and unused error string distribution are typically combined, and most of

them can be appropriately corrected by the MWPM algorithm, while a small portion cannot be corrected. Since the agent only sees the given syndrome, it does not have the opportunity to learn and recognize other types of errors and therefore is not applicable to other types of training.

5. Simulation analysis

5.1. Training

The training process of the decoder agent is implemented through the deep Q network (DQN) algorithm. The agent uses the experience replay technique in the algorithm to store the constantly acquired experiences as transitions in a buffer, and then randomly samples small batches of transitions from the buffer for updating the Q network. By unifying the samples of small batches through random sampling, a portion of random samples is extracted to reduce the temporal correlation of the data and to improve the stability of the neural network training.

To better apply to the CNN, we embedded the syndromes collected in the input phase into a binary matrix and encoded and signed the information. Each layer of the convolutional neural network had 64 output filters, followed by a fully connected layer with 512 neurons. We utilize the DQN constructed by stacking a feed-forward neural network over multiple convolutional layers. The final layer of this network has activation points, with each activation point encoding an action. Additionally, a fully connected feed-forward layer follows the convolutional layers, with ReLU serving as the activation function for the hidden layer and softmax used as the activation function for the output layer. In order to accelerate the model training, the ResNet network layer is introduced as the underlying architecture, with data stacked with ResNet 7, 14, and 21 network layers, which ensures that a large number of stacks can be stacked without reducing the learning efficiency of the convolutional layer. Furthermore, we use two neural networks that are structurally equivalent, the regular Q network with parameters θ and the target Q network with parameters θ_t. For each parameter iteration, we clone the active Q network (the network used to select the best action at each state) to obtain the target Q network, and the target network is synchronized with the Q network at the set time interval.

We train the agent using the DQL algorithm until the parameters of the CNN are stable. The training sequence starts from the action phase, where the agent utilizes a greedy policy and adjusts the Q network parameters to calculate a new target, the agent employs an ε-greedy strategy, which implies that it suggests the action with the highest Q value with probability $(1-\varepsilon)$ and chooses the action with the highest Q value, or else takes a random action. By executing different actions, rewards are generated, and new observations are obtained from the resulting joint state, which is then stored as a complete transition tuple $T = (P, a, r, O)$ in a memory buffer.

The training sequence then enters the learning phase using the stochastic gradient descent algorithm. Firstly, given a batch size of N, a random sample of transition $\{T_i = (P_i, a_i, r_i, O_i)\}_{i=1}^{N}$ is drawn from the buffer and replaced, and the training target value of the Q network is defined as

$$y_i = r_i + \gamma \max_{P' \in O_i} Q(P_i, a_i, \theta), \tag{9}$$

where γ is the discount factor, and we adjust the θ parameters continuously to adjust the cumulative reward predicted by the target network. Secondly, the gradient descent algorithm is used to minimize the loss function and to reduce the difference between the target value of the sample and the Q network prediction until the Q network produces an accurate value for the Q function, and the network parameters are adjusted based on $-\nabla_\theta \sum_i (y_i - Q(P_i, a_i, \theta))^2$. Then, a new training sequence is started and the target network weights with parameters θ_t are synchronized with the θ of the Q network at some specific rate. Finally, the ResNet network architecture is utilized for multiple iterations and predictions in the convolutional neural network (CNN), uses the data set with the same error rate for predictive training, and stops training when it is close to the threshold, and the parameters of the fully connected network and Q network are synchronized.

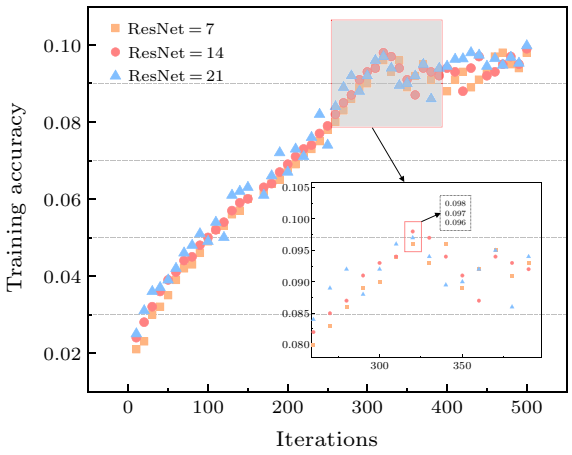


Fig. 6. The training iteration count as a function of both training error rate and accuracy. The horizontal axis denotes the number of training iterations, while the vertical axis represents training accuracy. Orange, red, and blue markers represent ResNet = 7, 14, and 21, respectively. To facilitate the visual inspection of data, zoom plots are employed.

The training process, as shown in Fig. 6, involves increasing the number of convolutional layers and adjusting the number of training iterations, which significantly improves the accuracy of the training. Increasing the number of layers of the network can obtain more precise training data. Before the number of iterations reaches 300, the ResNet network has an accuracy improvement of about 0.3% at each training interval compared to the ResNet = 7 network. However, after the number of iterations reaches 300 or more, the training accuracy starts to oscillate, the overfitting phenomenon appears, and the training accuracy improvement is small, fluctuating around 9.0%. After ensuring that the iteration depth reaches

500, the accuracy of all three networks can reach 9.6%, ensuring the generation of accurate predictions of the optimal error-correction chains under noisy conditions. After sufficient deep training, the Q value matrix of the Q network is fully replenished, and the performance approaches that of the optimal decoder, improving the decoding efficiency and accuracy significantly.

5.2. Error correction performance analysis

To quantify the decoding performance, the relationship between logical qubit error rate and physical qubit error rate must be analyzed. The physical error rate for which the decoder achieves approximately the same performance independent of the surface code distance is defined as the decoder threshold, which is often defined as p and is used as a single parameter to quantify the performance of the decoding algorithm. It is worth investing in larger distances for any physical error rate and decoding threshold. Since it is difficult to capture error correction for approximate surface code dimensions after mapping, this paper utilizes the MWPM and RL decoders to compute the threshold for restoring approximate 2D surface codes and draws the logical error rate in the physical error rate range of different code distances.

Through MWPM decoding, the error-correcting performance of regular surface codes and approximate surface codes at different code distances was obtained. As shown in Fig. 7(a), the orange line of code distance 5 indicates a slight increase in logical error rate with an increasing physical error rate, and the rate of increase is slow. The green line of code distance 11 shows a sharp increase in logical error rate, which reaches the highest threshold limit (the same horizontal coordinate is observed for each code distance). In general, the logical error rates exhibit a slight and slow increase with a threshold of 0.25%. As shown in Fig. 7(b), after approximate error correction, the logical error rate for the code distance of 11 increases significantly to 0.098. The data shows that the error probability for the code distance of 11 is the lowest, and its performance is relatively superior. However, the threshold strength achieved through MWPM decoding is still not high enough, at only 0.5%, which is below our expected result.

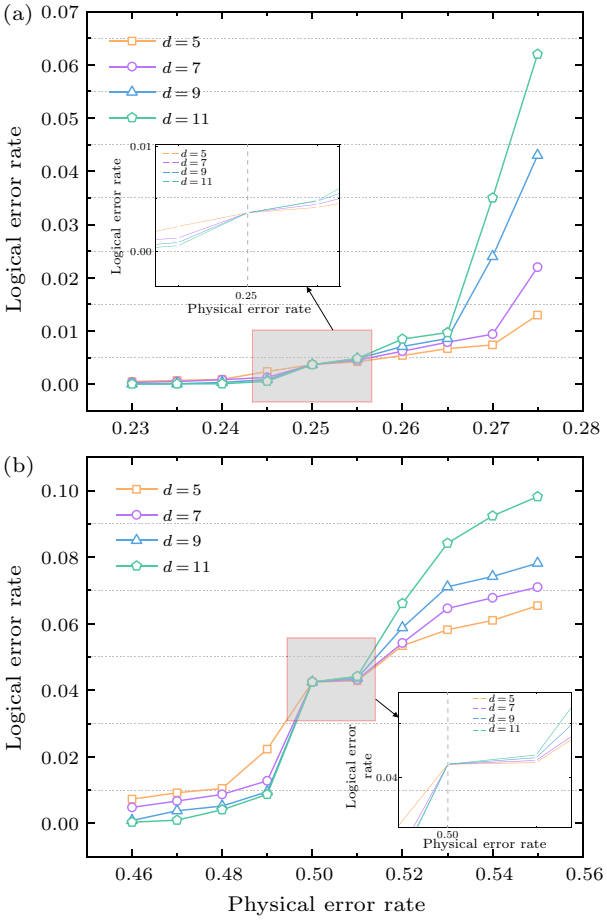


Fig. 7. The error correction performance of surface code with code distances of $d = 5$, 7, 9, and 11 decoded with MWPM. To facilitate comparison and analysis, different code distances are represented by different colors, namely orange, purple, blue, and green. (a) The error correction performance of MWPM decoding without approximation is shown. (b) The error correction performance of MWPM decoding after approximate quantum error correction at different d-values has a threshold of 0.5%.

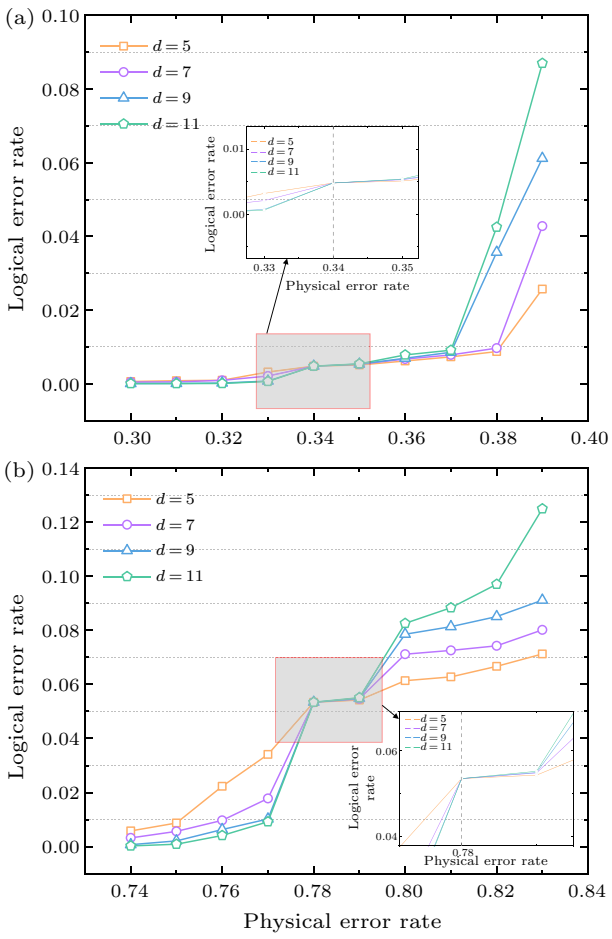


Fig. 8. The error correction performance of surface codes with the RL decoder evaluated for code distances $d = 5$, 7, 9, and 11. (a) The data is decoded by the RL decoder without performing approximate error correction. (b) The performance of the RL decoder after approximate error correction is demonstrated for different code distances, showing that the logical error rate increases significantly with the increase of physical error rate.

In order to further improve the threshold of the decoder, we use the trained deep Q network model to perform error correction in different code distance grids. As shown in Fig. 8(a), the logic error rate is relatively low in the orange line with a code distance of 5, while the logical error rate of code distance 11 increases significantly to 0.087. In addition, the RL decoding after approximate error correction also achieves significant improvements. As shown in Fig. 8(b), there is a noticeable increase in the logical error rate for the code distances of 5, 7, 9, 11, and the threshold reaches 0.78%.

For the display of the threshold results, MWPM did not achieve the good error correction performance we expected, and the error correction success rate was poor under the surface code dimension mapping model. RL models take the advantage of agent versus environment training, and the use of ResNet architecture shows good performance in error correction. Compared to the MWPM decoder, the threshold after approximate error correction has increased by 56%, which largely addresses the low threshold and poor error-correcting capability issues of surface codes. Furthermore, it is of great research interest to apply the RL model not only for threshold determination but also for constructing a universally distinguishable linear decoder.

6. Conclusions

In summary, we have investigated a reinforcement learning based error correction scheme for approximate surface codes under dimensional mapping operations, which is a novel error correction scheme. By exploiting the topological properties of error correction codes to map the surface code dimension to three dimensions, the three-dimensional lattice of topological codes with a higher degree of redundancy and more effective error correction exhibits excellent scalability. By reducing the space required for stabilizer measurements and the number of ancilla qubits needed for error correction, the approach achieves savings in measurement space and reduces resource consumption costs. In addition, to improve the decoding efficiency, we introduce a deep Q-learning based RL decoder, which made a major breakthrough in improvement of the error correction rate with a threshold of 0.78%, an improvement of 56% compared to the MWPM decoder. Of course, there are still many shortcomings in this work. For example, this paper only considers approximate error correction schemes for surface codes, and other quantum error correction codes such as color codes have not been considered and practiced in this paper. In terms of decoders, the development of generative adversarial network is currently relatively mature and is the focus of further research, which is also a preparation for further improving fault tolerance.

Acknowledgment

Project supported by the Natural Science Foundation of Shandong Province, China (Grant Nos. ZR2021MF049, ZR2022LLZ012, and ZR2021LLZ001).

References

[1] Xin T, Wang B X, Li K R, Xiang Y K, Shi J W, Wang T, Ruan D and Long G L 2018 *Chin. Phys. B* **27** 020308
[2] Zhou N R, Zhu K N and Zou X F 2019 *Ann. Phys.* **531** 1800520
[3] Ye T Y and Jiang L Z 2013 *Chin. Phys. Lett.* **30** 040305
[4] Yan D D, Fan X K, Chen Z Y and Ma H Y 2022 *Chin. Phys. B* **31** 010304
[5] Shor P W 1995 *Phys. Rev. A* **52** R2493
[6] Sun J and Lu S F 2020 *Chin. Phys. B* **29** 100303
[7] Wei R Y, Nie M, Yang G, Zhang M L, Sun A J and Pei C X 2019 *Acta Phys. Sin.* **68** 140302 (in Chinese)
[8] Barends R, Kelly J, Megrant A, *et al.* 2014 *Nature* **508** 500
[9] Takita M, Córcoles A D, Magesan E, Abdo B, Brink M, Cross A, Chow J M and Gambetta J M 2016 *Phys. Rev. Lett.* **117** 210505
[10] Linke N M, Gutierrez M, Landsman K A, Figgatt C, Debnath S, Brown K R and Monroe C 2016 *Sci. Adv.* **3** 1701074
[11] Nayak C, Simon S H, Stern A, Freedman M and Sarma S D 2008 *Rev. Mod. Phys.* **80** 1083
[12] Zarei M H 2015 *Phys. Rev. A* **91** 022319
[13] Brown B J, Loss D, Pachos J K, Self C N and Wootton J R 2016 *Rev. Mod. Phys.* **88** 045005
[14] Fowler A G, Mariantoni M, Martinis J M and Cleland A N 2012 *Phys. Rev. A* **86** 032324
[15] Nickerson N H, Li Y and Benjamin S C 2013 *Nat. Commun.* **4** 1756
[16] Hayden P, Nezami S, Popescu S and Salton G 2021 *PRX Quantum* **2** 010326
[17] Faist P, Nezami S, Albert V V, Salton G, Pastawski F, Hayden P and Preskill J 2020 *Phys. Rev. X* **10** 041018
[18] Cai W Z, Ma Y W, Wang W T, Zou C L and Sun L Y 2021 *Fundamen. Res.* **1** 50
[19] Kubica A and Demkowicz-Dobrzański R 2021 *Phys. Rev. Lett.* **126** 150503
[20] Geng M J, Chen Y, Xu T J and Ye T Y 2022 *EPJ Quantum Technol.* **9** 36
[21] Xue Y J, Wang H W, Tian Y B, Wang Y N, Wang Y X and Wang S M 2022 *Quantum Eng.* **9** 7643871
[22] Maskara N, Kubica A and Jochym-O'Connor T 2019 *Phys. Rev. A* **99** 052351
[23] Bravyi S, Browne D, Calpin P, Campbell E, Gosset D and Howard M 2019 *Quantum* **3** 181
[24] Qassim H, Pashayan H, Gosset D 2021 *Quantum* **5** 606
[25] Webster P, Vasmer M, Scruby T R and Bartlett S D 2022 *Phys. Rev. R* **4** 013092
[26] Ni X 2020 *Quantum* **4** 310
[27] Wang H W, Song Z Y, Wang Y N, Tian Y B and Ma H Y 2022 *Quantum Inf. Process.* **21** 280
[28] Varsamopoulos S, Bertels K and Almudever C G 2020 *Quantum Machine Intellig.* **2** 3
[29] Wang H W, Xue Y J, Qu Y J, Mu X Y and Ma H Y 2022 *npj Quantum Inf.* **8** 134
[30] Bombin H and Martin-Delgado M A 2007 *Phys. Rev. Lett.* **98** 160502
[31] Sheth M, Jafarzadeh S Z and Gheorghiu V 2020 *Phys. Rev. A* **101** 032338
[32] Bombín H 2016 *New J. Phys.* **18** 043038
[33] Scruby T R, Browne D E, Webster P and Vasmer M 2022 *Quantum* **6** 721
[34] Shao C P 2020 *Quantum Inf. Process.* **19** 102
[35] Baireuther P, O'Brien T E, Tarasinski B and Beenakker C W J 2018 *Quantum* **2** 48
[36] Arulkumaran K, Deisenroth M P, Brundage M and Bharath A A 2017 *IEEE Signal Process. Mag.* **34** 26
[37] Lin T X, Su Z, Xu Q C, Xing R and Fang D F 2020 *IEEE Acc.* **8** 69284
[38] Andreasson P, Johansson J, Liljestrand S and Granath M 2019 *Quantum* **3** 183
[39] Cheung W C, Simchi-Levi D and Zhu R 2020 *International Conference on Machine Learning* **119** 1843

[40] Chen S Y C, Yang C H H, Qi J, Chen P Y, Ma X L and Goao H S 2020 *IEEE Acc.* **8** 141007
[41] Fan X K, Liu G Z, Wang H W, Ma H Y, Li W and Wang S M 2022 *J. University of Electronic Science and Technology of China* **51** 642
[42] Mnih V, Kavukcuoglu K, Silver D, *et al.* 2015 *Nature* **518** 529
[43] Zhang Q C, Lin M, Yang L T, Chen Z K and Li P 2019 *IEEE Trans. Sustainable Comput.* **4** 132
[44] Zhang Q, Lin M, Yang L T, Chen Z K, Khan S U and Li P 2018 *IEEE Trans. Services Comput.* **12** 739
[45] Brown B J 2020 *Sci. Adv.* **6** eaay4929
[46] Chamberland C and Noh K 2020 *npj Quantum Information* **6** 91
[47] Chamberland C and Cross A W 2019 *Quantum* **3** 143
[48] Colomer L D, Skotiniotis M and Muñoz-Tapia R 2020 *Phys. Lett. A* **384** 126353
[49] Sweke R, Kesselring M S, van Nieuwenburg E P L and Eisert J 2020 *Machine Learning: Sci. Technol.* **2** 025005
[50] Nautrup H P, Delfosse N, Dunjko V, Briegel H J and Friis N 2019 *Quantum* **3** 215