

Recurrent neural network decoding of rotated surface codes based on distributed strategy

Fan Li(李帆), Ao-Qing Li(李熬庆), Qi-Di Gan(甘启迪), and Hong-Yang Ma(马鸿洋)

Citation: Chin. Phys. B, 2024, 33 (**4**): 040307. DOI: 10.1088/1674-1056/ad2bef
Journal homepage: http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

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

Performance of entanglement-assisted quantum codes with noisy ebits over asymmetric and memory channels

Ji-Hao Fan(樊继豪), Pei-Wen Xia(夏沛文), Di-Kang Dai(戴迪康), and Yi-Xiao Chen(陈一骁)
Chin. Phys. B, 2023, 32 (**12**): 120304. DOI: 10.1088/1674-1056/acf492

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)
Chin. Phys. B, 2023, 32 (**10**): 100307. DOI: 10.1088/1674-1056/acd8a9

An overview of quantum error mitigation formulas

Dayue Qin(秦大粤), Xiaosi Xu(徐晓思), and Ying Li(李颖)
Chin. Phys. B, 2022, 31 (**9**): 090306. DOI: 10.1088/1674-1056/ac7b1e

Determination of quantum toric error correction code threshold using convolutional neural network decoders

Hao-Wen Wang(王浩文), Yun-Jia Xue(薛韵佳), Yu-Lin Ma(马玉林), Nan Hua(华南), and Hong-Yang Ma(马鸿洋)
Chin. Phys. B, 2022, 31 (**1**): 010303. DOI: 10.1088/1674-1056/ac11e3

Quantum computation and error correction based on continuous variable cluster states

Shuhong Hao(郝树宏), Xiaowei Deng(邓晓玮), Yang Liu(刘阳), Xiaolong Su(苏晓龙), Changde Xie(谢常德), and Kunchi Peng(彭堃墀)
Chin. Phys. B, 2021, 30 (**6**): 060312. DOI: 10.1088/1674-1056/abeb0a

---

Recurrent neural network decoding of rotated surface codes based on distributed strategy

Fan Li(李帆)[1], Ao-Qing Li(李熬庆)[1], Qi-Di Gan(甘启迪)[2], and Hong-Yang Ma(马鸿洋)[2,†]

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

(Received 6 December 2023; revised manuscript received 5 February 2024; accepted manuscript online 22 February 2024)

Quantum error correction is a crucial technology for realizing quantum computers. These computers achieve fault-tolerant quantum computing by detecting and correcting errors using decoding algorithms. Quantum error correction using neural network-based machine learning methods is a promising approach that is adapted to physical systems without the need to build noise models. In this paper, we use a distributed decoding strategy, which effectively alleviates the problem of exponential growth of the training set required for neural networks as the code distance of quantum error-correcting codes increases. Our decoding algorithm is based on renormalization group decoding and recurrent neural network decoder. The recurrent neural network is trained through the ResNet architecture to improve its decoding accuracy. Then we test the decoding performance of our distributed strategy decoder, recurrent neural network decoder, and the classic minimum weight perfect matching (MWPM) decoder for rotated surface codes with different code distances under the circuit noise model, the thresholds of these three decoders are about 0.0052, 0.0051, and 0.0049, respectively. Our results demonstrate that the distributed strategy decoder outperforms the other two decoders, achieving approximately a 5% improvement in decoding efficiency compared to the MWPM decoder and approximately a 2% improvement compared to the recurrent neural network decoder.

Keywords: quantum error correction, rotated surface code, recurrent neural network, distributed strategy

PACS: 03.67.Pp, 03.67.–a, 87.64.Aa **DOI:** 10.1088/1674-1056/ad2bef

1. Introduction

Quantum computers[1,2] can effectively solve problems that are either intractable or require significant time to solve on classical computers. They can execute quantum algorithms[3,4] that offer advantages over classical algorithms. Noise is a major obstacle to achieving fault-tolerant quantum computing.[5] Quantum devices are very sensitive to noise. Quantum error correction can reduce the impact of noise on quantum systems, which is a solution to this problem. Quantum states are always affected by the environment or noise, resulting in quantum decoherence and errors. Quantum error correction[6–8] can correct these errors and restore the quantum state to its original state. Quantum error correction includes encoding and decoding processes.[9,10] Encoding is used to enhance protection against errors by encoding the logical information of physical qubits. Decoding is used to identify the type and location of errors and correct them. However, the time budget for decoding is limited, determined by the time of a single round of error correction,[11] and must be processed quickly and accurately to ensure that the quantum state is error-free.

Quantum error correction codes include surface codes,[12,13] color codes,[14,15] bosonic codes,[16] etc. Surface code is a representative method to achieve error correction due to its high tolerance for errors.[17] It is a topological code. Topological codes are formed by piecing together repeated elements. The way they are constructed makes the structure of surface codes relatively simple and easy to understand. Surface codes are applicable to the two-dimensional nearest neighbor limit of many current quantum technologies, and their error threshold is approximately 1%.[18] Rotated surface code[19] is a variant of surface code, which is an encoding scheme that is improved and optimized based on surface code.

Various decoding algorithms have been proposed over the years, such as maximum-likelihood algorithm,[20] renormalization group (RG) algorithm,[21] minimum weight perfect matching (MWPM),[22,23] etc. The recently proposed sliding window decoder, known as the sandwich decoder,[24] successfully utilizes its parallelism to address the issue of backlog that arises with the code scales up when decoding surface codes. In recent years, due to the development of machine learning, neural networks are used as an alternative to classical decoders to identify errors,[25,26] and they can improve decoding speed. Among them, various neural networks have different decoding methods, and various neural networks have also been proposed,[27–30] such as feedforward neural networks (FFNN), recurrent neural networks (RNN), and convolutional neural networks (CNN). Decoding based on neural network[31–33] shortens the time required for decoding and can achieve an effect similar to that of classical decoding algorithms. However, as the code distance increases, the number of samples required for neural network training increases exponentially.[34] Com-

†Corresponding author. E-mail: hongyang_ma@aliyun.com

© 2024 Chinese Physical Society and IOP Publishing Ltd

pared with traditional decoding, it is more difficult to apply to larger quantum systems. If the number of training samples is insufficient, then it will be difficult to achieve decoding performance similar to traditional decoders.[35]

This paper is organized as follows. We first give a brief introduction to surface codes and rotated surface codes in Section 2. Then, the decoding ideas of this article and the construction of the distributed strategy decoder are explained in Section 3. The simulation results are discussed and analyzed in Section 4, and finally the conclusion of this paper is summarized in Section 5.

2. Background

A quantum error-correcting code is defined by three parameters $[n, k, d]$, where n represents the number of physical qubits, k represents the number of logical qubits allowed by the code, and d represents the code distance.[36] The code distance is a measure of code strength and represents the minimum number of physical qubits required to change the state of a logical qubit.[37] The number of errors that can be corrected between quantum error-correcting codes is determined by the code distance. If the physical error rate is lower than the error threshold, then the logical error rate can be suppressed by increasing the code distance. The code distance represents the length of the smallest logical operator, that is, the length of the smallest undetectable error chain. The relation between code distance and the weight of errors (w.o.e) that can be corrected successfully is represented according to the following equation:

$$\text{w.o.e} = \left\lfloor \frac{d-1}{2} \right\rfloor. \tag{1}$$

Stabilizer codes[38] make it possible to detect information about errors by measuring specific Pauli stabilizer generators without revealing the state of the encoded information. Stabilizer codes consist of Pauli groups. The surface code is a topological stabilizer code.[39,40] The Pauli group is defined by the following formula:

$$G = \{I, X, Y, Z\}, \tag{2}$$

where $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, $Y = \begin{bmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{bmatrix}$, $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$, $Y = \mathrm{i}XZ$. The n-bit Pauli group, which is composed of four Pauli operators, represents the set of all possible Pauli operations on n qubits. It is defined as the tensor product of n Pauli operators

$$G_n = \{P_1 \otimes P_2 \otimes P_3 \otimes \cdots \otimes P_n;\ P_i \in G,\ \forall i\}. \tag{3}$$

Logical X and Z operators are anticommute with each other, but commute with all stabilizers. Errors in the data qubits can be known by measuring the stabilizers. That is, if the error is anticommuted with the stabilizer, the result of the stabilizer syndrome measurement is -1.

Surface codes, first proposed by Kitaev,[41] are one of the most popular quantum error-correcting codes. They are an error-correcting code that defines a family of qubits in a two-dimensional nearest neighbor structure. Surface codes are made up of two types of qubits, ancilla and data, which differ only in their function in the code. The ancilla qubits are measured to detect the presence of errors, while the data qubits are used to perform calculations. Implementing a surface code with distance d has d data qubits per row and column. Its qubit layout is shown in Fig. 1(a). Each lattice encodes a logical qubit. Errors in qubits are divided into bit errors and phase flip errors. Surface codes perform detection and error correction on physical qubits using X and Z stabilizers. The X-type stabilizer is used to detect Z errors (phase flip errors), and the Z-type stabilizer is used to detect X errors (bit errors).[42] The circuits for measuring the X and Z stabilizers are shown in Fig. 1(b).

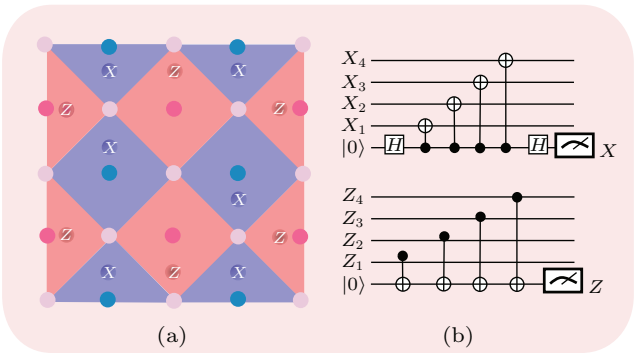


Fig. 1. (a) Qubit layouts of the surface code. Data qubits are located at the grey dots and the ancilla qubits are located at the blue dots, which are used to measure the stabilizers. Blue plaquettes represent X stabilizers and red plaquettes represent Z stabilizer. (b) Up: circuit for X-type stabilizer measurement. Down: circuit for Z-type stabilizer measurement.

The smallest surface codes known to be constructed are called rotated surface codes.[43] The number of physical qubits required for a rotated surface code with a code distance d to encode a logical qubit is $2d^2 - 1$, which is approximately half of that required for surface codes.[44] Therefore, 17 and 49 physical qubits are required for encoding one logical qubit by rotated surface codes with code distances of 3 and 5, respectively. The rotated surface code with a code distance of 3 consists of 9 data qubits at the vertices of the grid and 8 ancilla qubits inside the grid. Each ancilla qubit can interact with its adjacent 4/2 data qubits. The surface code rotates the lattice at a certain angle to form a rotated surface code. Compared with standard surface codes, rotated surface codes reduce errors and provide better error tolerance by rotating to eliminate errors that may occur in different directions. Errors occurring in the code can be detected when taking syndrome measurements.

For example, a single X error in data qubit DQ_2 in Fig. 2 is detected when measuring stabilizers $DQ_1DQ_2DQ_4DQ_5$. The correction operation of the rotated surface code involves several steps. First, the physical qubit is measured using X and Z stabilizers. Based on the measurement results, the location of the physical qubit where the error occurred and the type of error can be determined. Finally, the identified error is corrected.

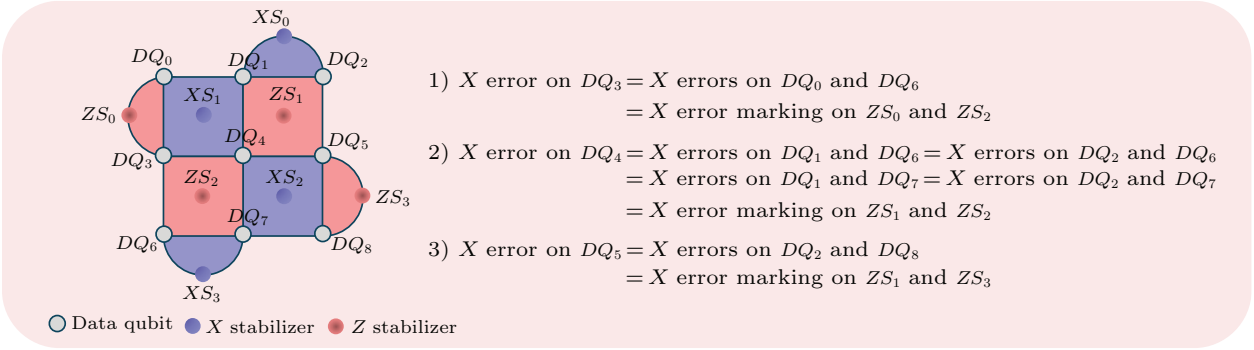


Fig. 2. The left part represents rotated surface code with code distance of 3. The right part represents part of the single X error and the two X errors that produce the same Z syndrome in this $d = 3$ rotated surface code.

Table 1 summarizes the list of stabilizers for this $d = 3$ rotated surface code. In this table, X_i and Z_i represent the Pauli operators acting on the i-th data qubit.

Table 1. List of stabilizers of the $d = 3$ rotated surface code.

Index	X stabilizer	Z stabilizer
1	X_1X_2	Z_0Z_3
2	$X_0X_1X_3X_4$	$Z_1Z_2Z_4Z_5$
3	$X_4X_5X_7X_8$	$Z_3Z_4Z_6Z_7$
4	X_6X_7	Z_5Z_8

3. Decoding the rotated surface code

3.1. Decoding strategy

The main criterion for a good decoder is its decoding performance. We need a decoder that can run at high speed to save time consumption. Therefore, the neural network decoder[45,46] is used because the neural network decoder can greatly reduce its running time after training, thereby reducing time loss. However, to train the algorithm with a neural network decoder, the demand for qubits increases exponentially as the training continues. Therefore, we use a distributed strategy decoder to reduce the number of qubits required to train neural network.

Applying the idea of distributed decoding to neural network decoders is similar to the method of renormalization group decoding.[47,48] Our decoding idea is to decompose the quantum error correction code to obtain small parts of the code, and all the small parts together constitute the complete code. By decoding these small parts one by one to obtain the decoding information of each part, and then integrating the decoding results of all parts, we can get all the information of the code. As shown in Fig. 3(a), the result of decoding the four green squares on the left is equivalent to the result of decoding the brown square on the right.

Renormalization group decoding strategy is represented by Fig. 3(b). Renormalization group decoding is self-adaptive and parallelizable, and it can decode efficiently under high noise. Stepwise tuning of the probability distribution for recovery from noise by progressively dividing the lattice into smaller parts. The calculation first divides the lattice into small blocks and calculates the error probability of the qubits within them, then it divides into larger blocks and calculates the error probability of all the qubits included in the block, and repeats this process until only the last block is left itself, then the block contains all the qubits in the code.

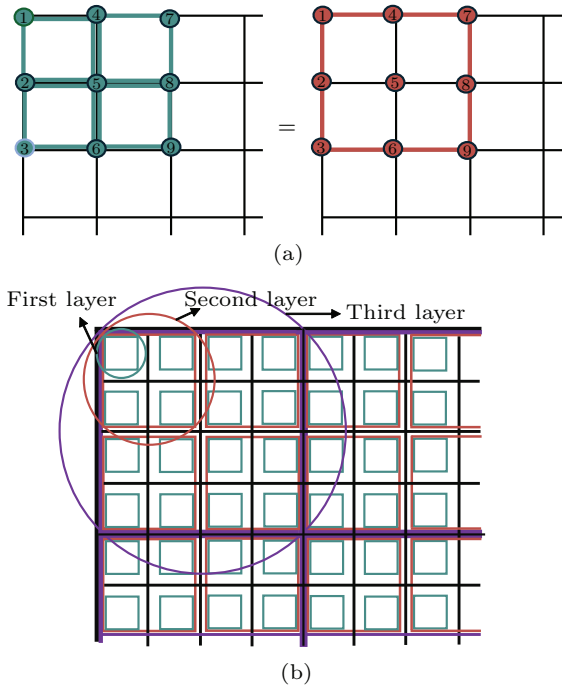


Fig. 3. (a) Our decoding strategy. Different colored squares represent different sized portions of the quantum error correction code. (b) Block segmentation of the code. The smallest level is represented by the green blocks, the next level is represented by the brown blocks, the following level is represented by the purple blocks, etc.

We apply this approach to rotated surface codes. Since parity checks cannot be broken down into constant-sized blocks in such a way that each parity check corresponds to an entire block, we propose overlapping blocks to ensure that the entire parity check of the code is contained in the block. The error rate is calculated by the belief propagation method of the renormalization group. Belief propagation[49] is used to calculate the message passing algorithm of the graph. Messages are passed through the nodes in the graph to update the value of variables in it. Variables are affected by the measurement of other variables.

Let M_q represent the marginal distribution of qubit q, which is obtained from the output of the previous renormalization group. The output message of a block at time t becomes the input message of its adjacent block at time $t+1$

$$m^{t+1,\mathrm{in}}_{(B_1,q)}(M_q) = m^{t,\mathrm{out}}_{(B,q)}(M_q). \tag{4}$$

We use a neural network decoder for distributed decoding based on the concept of renormalization group decoder. The iterative nature of the renormalization group decoder allows it to gradually improve the error correction process through training. By gradually enlarging and analyzing larger qubit blocks, different errors in each qubit can be captured and corrected, thereby effectively suppressing noise.

The main difference between the distributed strategy decoder and the renormalization group decoder is that it has only one cascade. Instead of moving from small blocks to larger blocks until the last block is left, it divides the qubit lattice into identical sized blocks that overlap each other so that each block contains the parity of the code. It then gets the error information from each block, integrates the information in all blocks, and finally gets the error information of the whole lattice. Instead of computing the qubit information and forwarding it to the next cascade, we need to think about ways to put it all together.

The advanced decoder mainly consists of two modules that run in parallel to achieve fast and efficient decoding performance. These two modules can both be neural networks, or one may be a neural network and the other may be a non-neural network. For the first module, our distributed strategy decoder adopts a simple decoder.[50] Simple decoders can be constructed in different ways, and the decoding performance of different constructions is also different. In order to maintain high-speed decoding, the simple decoder adopts the simplest design, and less consideration is given to its error correction quality. For the second module, we use a recurrent neural network to further correct and improve the decoding results of the simple decoder.

3.2. Decoding algorithm

3.2.1. Simple decoder

The simple decoder provides naive decoding for the entire lattice, and each detection event generates a chain that connects boundaries of the same type. The simple decoder's decoding is provided on the data qubits covered by this chain. As Fig. 4 shows, the corrections proposed by the simple decoder will be, XS_7 proposes corrections to Z_6, Z_{12} and ZS_7 propose corrections to X_{18} and X_{22}.

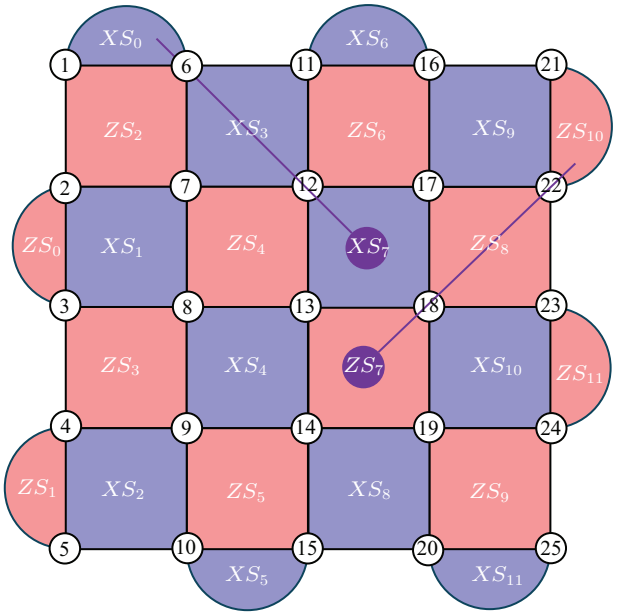


Fig. 4. A description of the simple decoder operation for the rotated surface code with code distance 5. Purple dots represent detection events, and purple lines represent data qubits that will be corrected by detection events.

The simple decoder performs error syndrome across the entire lattice, providing a set of corrections for the lattice. It is a quick process though because each detection event is independent of the others and they can run in parallel. However, the simple decoder has a simple decoding process and cannot provide high decoding accuracy in order to remain fast in decoding.

Therefore, we introduce a neural network for further decoding and improve the performance of our decoder.

3.2.2. Neural network

The neural network that we use is a recurrent neural network. It can recursively process the information of qubits in the rotated surface code, and its memory feature helps to solve the problem of information connection before and after each round of processing, and helps to finally get an ideal result.

The reason we use a recurrent neural network is that it can be efficiently decoded without knowing the number of iterations required for decoding. We use a recurrent neural network architecture with long-short term memory layers,[51] which consists of two long-short term memory layers with $N_L^{(1)}$ and $N_L^{(2)}$ with different internal state sizes and a layer with $N_L^{(\mathrm{E})}$ neurons consisting of a fully connected evaluation layer. In Fig. 5, when the long-short term memory layer updates the state of a qubit, it is updated every cycle and the information is saved for several cycles.

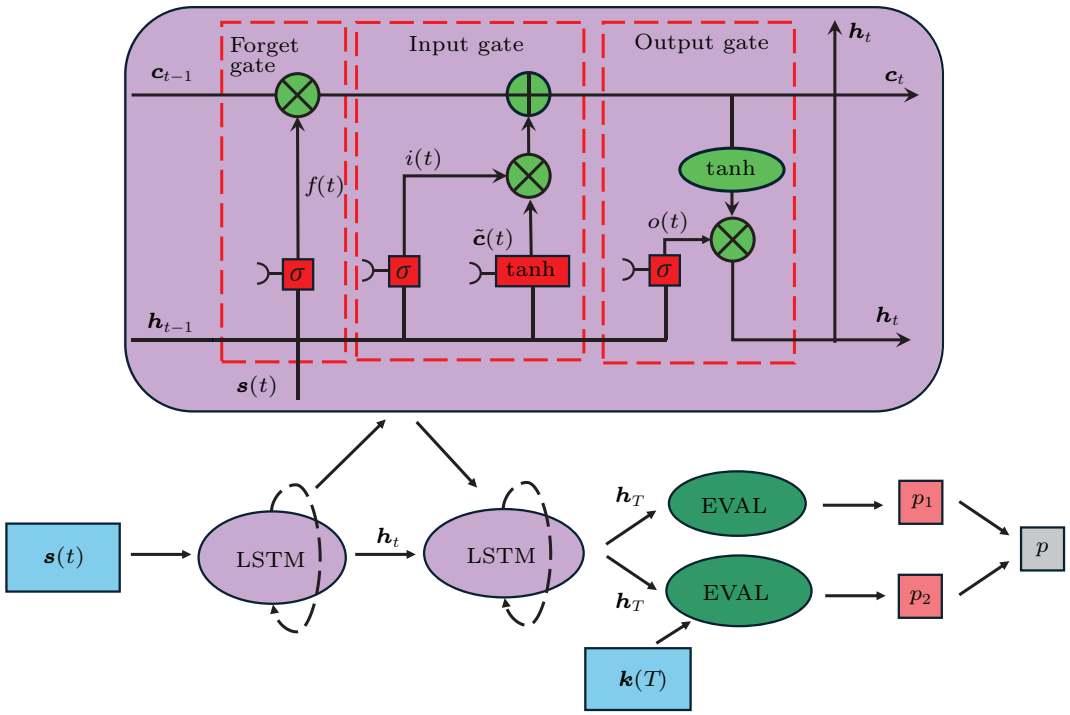


Fig. 5. Architecture of recurrent neural networks. The upper head is obtained only by the syndrome increment $\boldsymbol{s}$ from the ancilla qubits, while the lower head is obtained by the syndrome increment $\boldsymbol{k}$ from the final measurement of the data qubits. Eval means the fully connected evaluation layers. While blocks denote input and output data, solid arrows denote data flow of the system, and dashed arrows denote the internal memory flow of the LSTM layers.

The forget gate in the long-short term memory layer determines what information in the qubit will be discarded

$$f_t = \sigma(W_{f\boldsymbol{h}}\boldsymbol{h}_{t-1} + W_{f\boldsymbol{s}}\boldsymbol{s}(t) + b_f), \tag{5}$$

where W represents the weight, b represents the bias, $\boldsymbol{s}(t)$ represents the input of the LSTM layer, and f_t represents the forget gate information. When f_t is 1, the information is retained. When f_t is 0, all information is discarded. The input gate determines the information to be stored in the qubit, following the rule

$$i_t = \sigma(W_{i\boldsymbol{h}}\boldsymbol{h}_{t-1} + W_{i\boldsymbol{s}}\boldsymbol{s}(t) + b_i), \tag{6}$$

where i_t represents the input gate information. The output gate determines the information that can be output based on the qubit state

$$o_t = \sigma(W_{o\boldsymbol{h}}\boldsymbol{h}_{t-1} + W_{o\boldsymbol{s}}\boldsymbol{s}_t + b_o), \tag{7}$$

where o_t represents the output gate information. In this long-short term layer architecture, long time memory and short time memory can be formulated as

$$\tilde{c}_t = \tanh(W_{\tilde{c}\boldsymbol{h}}\boldsymbol{h}_{t-1} + W_{\tilde{c}\boldsymbol{s}}\boldsymbol{s}(t) + b_{\tilde{c}}), \tag{8}$$

$$\boldsymbol{c}_t = f_t \cdot \boldsymbol{c}_{t-1} + i_t \cdot \tilde{c}_t, \tag{9}$$

$$\boldsymbol{h}_t = o_t \cdot \tanh(\boldsymbol{c}_t), \tag{10}$$

where the operator '$\cdot$' represents the point-wise multiplication of two vectors, $\boldsymbol{c}_t$ and $\boldsymbol{h}_t$ represent long-term memory and short-term memory, respectively.

Suppose there is a logical qubit, which is an arbitrary logical state $|\varphi_L\rangle$. The result measured r in the Z stabilizer is $r \in \{-1, 1\}$. During the measurement process, the phase information is lost. We need information about the parity of bit flips in stabilizers. If the bit flip parity is odd, correct the error by inverting r. The purpose of decoding is to estimate the probability p of a logical qubit having an odd number of bit flips.

The measurement of the ancilla qubit in this experiment is represented by the syndrome increment $\boldsymbol{s}(t)$, and the readout of the data qubit can be used to calculate the final syndrome increment vector $\boldsymbol{k}(T)$. When parity prediction is required at time T, the final syndrome increment $\boldsymbol{k}(T)$ is obtained from the final syndrome k measured by the data qubit and the final syndrome readout $\boldsymbol{s}(T)$ of the ancilla qubit

$$\boldsymbol{k}(T) = k - \boldsymbol{s}(T) \mod 2. \tag{11}$$

We initialize the logical qubit $|\varphi_L\rangle = |0\rangle$ and then compare with the final measured logical state to get the true bit flip parity $p_{\text{true}} \in \{0, 1\}$.

The decoder needs to be time-shift invariant, and it must process the decoded data the same in each round. The final syndrome of the recurrent network breaks the time translation invariance at the end of the computation. The decoder needs to consider the possibility of the error signal being propagated to subsequent cycles over time. But this is different from the final syndrome: because it is obtained directly from the measurement of the data qubit, any error in the data qubit will not

propagate forward in time, and the ancilla qubit has no effect on the time translation invariance in terms of the error propagation.

To achieve time-translation invariance of the decoder, we train it in parallel using a single network with two heads. The upper head estimates the probability of bit flip error parity p_1 based on the syndrome increment. The data available to the decoder after T rounds is the T syndrome $\boldsymbol{s}(t)$ and the final syndrome increment $\boldsymbol{k}(T)$ computed from the final data qubit readout. The decoder must ultimately output a single data bit, called the final parity correction, to determine the final logic state correction.

The recurrent neural network consists of two long-short term memory layers with $N_L^{(1)}$ and $N_L^{(2)}$ with different internal state sizes and a layer with $N_L^{(\mathrm{E})}$ neurons fully connected evaluation layer. The ancilla qubit measurements of the experiment are represented by the syndrome measurement vector $\boldsymbol{s}(t)$. By reading the data qubits at the end of the trial, the final syndrome increment vector $\boldsymbol{k}(T)$ can be calculated. When it is necessary to predict the parity of the network at time T, the information of the recurrent network is passed to the evaluation layer, and the final syndrome incremental vector $\boldsymbol{s}(t)$ measured by the ancilla qubit. The lower head adds the information of the final syndrome increment $\boldsymbol{k}(T)$ to estimate the probability p_2 that the measurement of the final data qubits changes the final parity.

During the training phase, the bit flip probabilities p_1 and $p_2 \in \{0,1\}$ of the upper and lower heads are compared with the true parity probability p_{true}. We minimize the loss function by adjusting the weight of the network connection to make p_1 and p_2 close to p_{true}. We use TensorFlowLibrary to perform this process. If the probability of the lower head output is $p < 0.5$, the probability of predicting a bit flip error is even, otherwise it is odd.

3.3. Training dataset

To train the recurrent neural network, we create a training dataset. Our training dataset consists of the error syndrome of logical qubits and their logical states obtained by the simple decoder after running a rotated surface code. The size of the training dataset varies with code distance.

We can see that the rotated surface code with a code distance of 5 is composed of 4 overlapping rotated surface codes with a code distance of 3 in Fig. 6.

It can be seen from Fig. 6 that each parity check involves at most two blocks. The resulting error syndrome for the entire lattice is divided into error syndrome parts for each small block. We average the logical error rate for each small block and find the probability of each small block's logical state. The four probabilities of the logical state of each small block are used as the input of the recurrent neural network, and the recurrent neural network will output the probability of the logical state of the entire lattice when outputting.

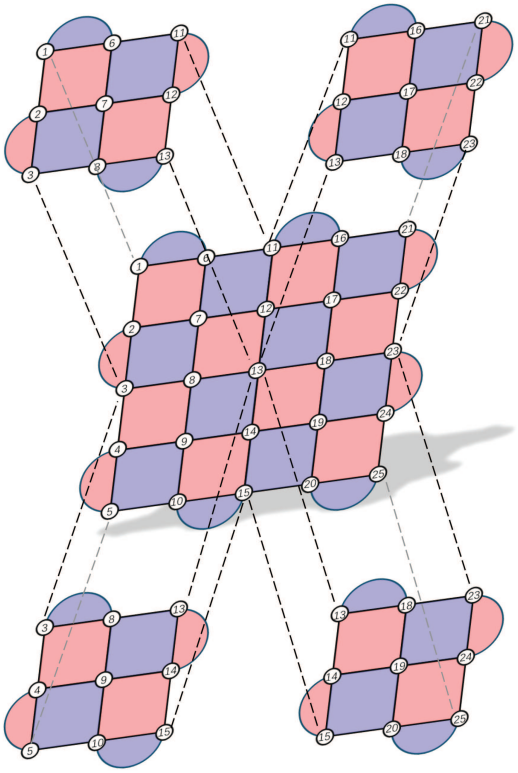


Fig. 6. Divide a $d = 5$ rotated surface code into four overlapping $d = 3$ rotated surface codes.

The recurrent neural network is trained to effectively detect logical errors resulting from error syndromes generated by simple decoder. Based on the output of the recurrent neural network, additional corrections are applied in the form of logical operators to eliminate any potential logical errors that simple decoder may have made.

The problem faced by all neural network-based decoders is the exponential growth of training samples required for effective decoding. We modified the decoding algorithm to increase the number of important training samples without increasing the size of the training dataset. Comparing the outputs of simple decoder and recurrent neural network decoder, if the results are consistent, then it is considered that simple decoder has successfully corrected error syndromes. Conversely, if the results are inconsistent, then it is considered that simple decoder has failed to correct error syndromes. Subsequently, by removing error syndromes that were properly corrected by simple decoder from the training dataset, the dataset can be further expanded, thereby generating more relevant error syndromes.

The depolarizing error model is a classic noise model. In the depolarizing error model, X, Y, Z errors have the same probability $p_{\mathrm{error}}/3$, and the probability of no error is $(1 - p_{\mathrm{error}})$, which only acts on data qubits. The circuit noise model applies depolarizing noise to the data qubits and the ancilla qubits. Each single-qubit gate is assumed to be perfect

and has a probability of depolarizing noise of $p_{\text{error}}/3$. Each two-qubit gate is also assumed to be perfect and has a two-bit depolarizing map, where the two-qubit Pauli has probabilities $p_{\text{error}}/15$. The probability of no error is $(1-p_{\text{error}})$.

4. Numerical results

In order to speed up the training process of the recurrent neural network model and enable it to be successfully used on a wider range of datasets, we introduce the ResNet layer as the underlying architecture of the network, ensuring that the shortcut output is added to the output of the stacked layer through the residual block, we use ResNet 7 and 14 for data training. Using ResNet can effectively alleviate the gradient disappearance problem in our network and improve the performance of the network.

The training process is shown in Fig. 7, we can observe that as the number of iterations increases, ResNet 14 achieves slightly better training accuracy than ResNet 7. Additionally, both ResNet 7 and ResNet 14 demonstrate significant improvements in training accuracy compared to the training curve without ResNet. When the number of training times reaches 300, overfitting phenomenon is prone to occur. When the number of iterations exceeds 500, the training accuracy of ResNet 7 and 14 will reach more than 9.9%. The size of the training dataset varies with code distance. In the composition of the training dataset, we use 90% of the data for training, and the remaining 10% are used as the test set to test the learning efficiency of the network. We use the distributed strategy decoder on this training set and perform a performance comparison with two different decoders. In this section, we analyze the performance of our proposed decoder by decoding rotated surface codes with different code distances.

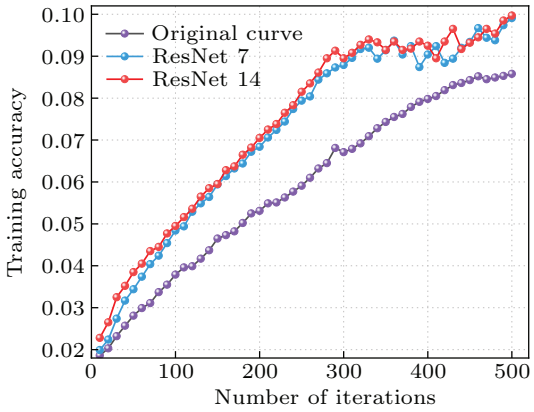


Fig. 7. Compare the training accuracy of the training curve without ResNet and the training curve using ResNet 7, 14 through the recurrent neural network.

We chose to divide the code into overlapping blocks with code distance of 3. Because rotated surface code with code distance of 3 is the smallest rotated surface code, it consists of only 256 error syndromes. This is a complete training dataset that is easy to obtain, and the error syndrome of any large code distance rotated surface code can be composed of it without having to study all possible error syndrome of large lattices, which greatly simplifies the training difficulty. The only disadvantage of using overlapping blocks with a code distance of 3 is that the logic state caused by some error syndrome is very fuzzy. Suppose an error syndrome involves three adjacent physical qubits. In a rotated surface code, the combined error of these three qubits may correspond to a variety of possible error conditions, such as one bit-flip error, two bit-flip errors, or three bit-flip errors. Since the specific type of error cannot be directly determined, uncertainty arises when correcting the error, resulting in ambiguity in the logical state. The frequency of such fuzzy errors is very low, so errors caused by this shortcoming rarely occur.

Codes composed of overlapping blocks are important to our study. Therefore, we used a rotated surface code consisting of rotated surface code with code distance of 3 as overlapping blocks in our experiments. For a fair comparison, we use the same dataset to train both decoders, so the decoding performance between them is comparable.

As shown in Fig. 8, we decoded rotated surface code with the code distance of 3, 5, 7, and 9 using MWPM decoder, recurrent neural network decoder, and our distributed strategy decoder. This let us know that code distance is not the only factor that affects the success of error correction because with the increase of bit flip errors, the logical error rate will get saturated. The decoding performance of rotated surface codes composed of four overlapping blocks ($d = 3$ rotated surface code) at different code distances is compared with normal rotated surface codes using recurrent neural network and classical MWPM decoder. Figure 8 shows the comparison of the physical error rate and logical error rate of the three decoders for rotated surface codes with different code distances. It can be seen that the logical error rate increases with the increase of physical error rate. The decoder threshold is the physical error rate, which indicates that a logical qubit outperforms a physical qubit. The threshold value of MWPM decoder is about 0.0049 by using ResNet network as training model to decode $d = 3, 5, 7, 9$ rotated surface codes. Using the recurrent neural network decoder to decode rotated surface codes with different code distances, the threshold value is 0.0051, and it is 0.0052 for our distributed strategy decoder. The MWPM decoder threshold is slightly lower than our scheme. The results show that rotated surface codes composed of overlapping methods have similar performance to normal rotated surface codes, they only have slight differences. We can see that recurrent neural network is better than MWPM decoder because it needs less time to decode rotated surface code, while it gets a better decoding performance. Our distributed strategy decoder has a better performance than recurrent neural network decoder and MWPM decoder. In terms of decoding efficiency, we get about 2% and 5% improvement, respectively.

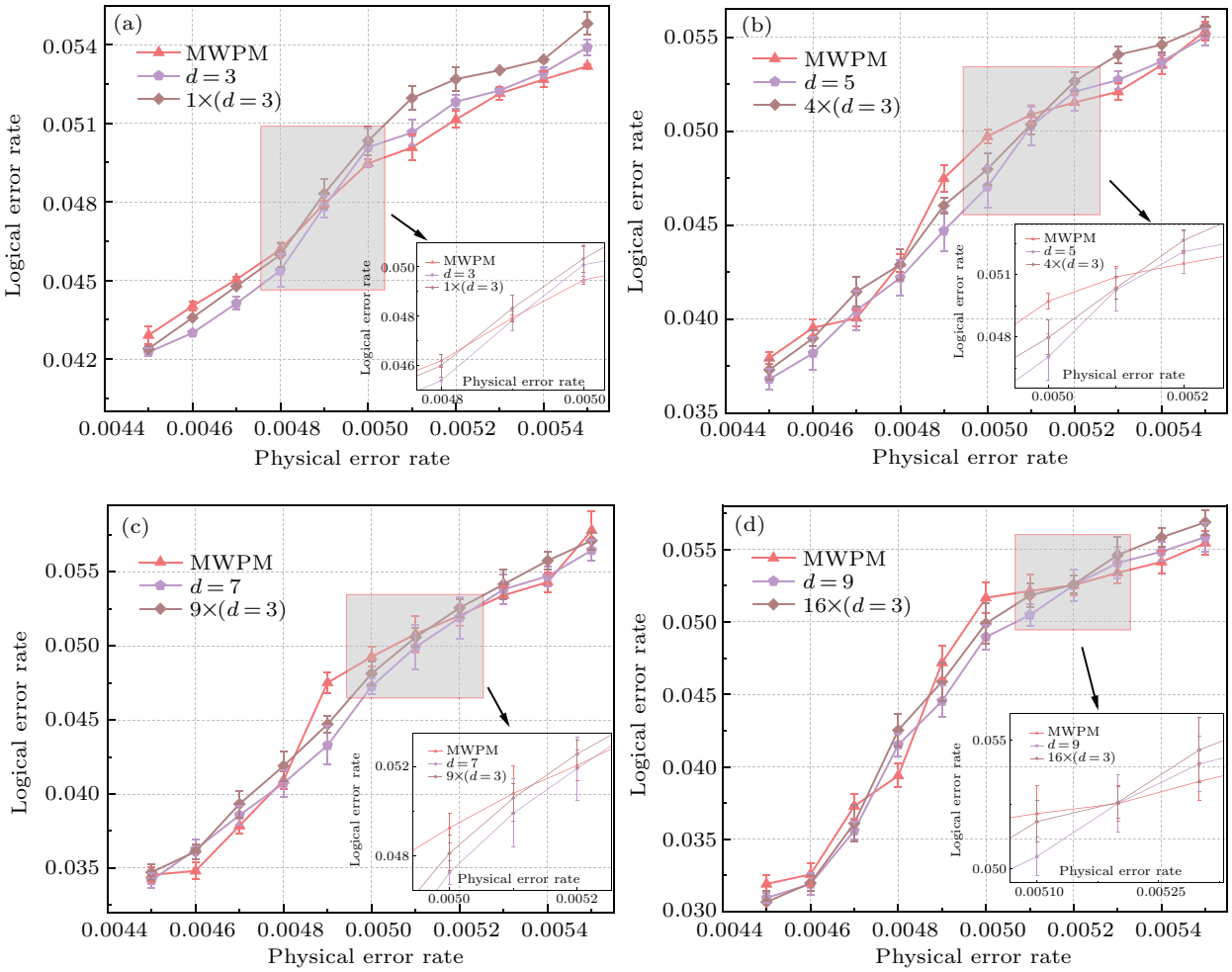


Fig. 8. The $d = 3$, 5, 7 and 9 rotated surface codes are decoded using MWPM, recurrent neural network and distributed strategy decoder. The red line represents the performance of the MWPM decoder, the purple line represents the performance of the recurrent neural network decoder, and the brown line represents the performance of our distributed strategy decoder.

5. Conclusion

In this work, we use a distributed approach to study rotated surface codes. The large code distance rotated surface code formed by overlapping small code distance rotated surface codes using the concatenated code idea has almost the same properties as the normal rotated surface code. The exponential growth of the training set size required for the neural network decoder in relation to the linear growth of the code distance is a thorny problem. In this work, the distributed strategy decoder we proposed based on the renormalization group method can effectively alleviate this problem, making it possible to study error-correcting codes with larger code distances. The near-parallel operation of our simple decoder and recurrent neural network also makes the decoding time very short. The network is trained through the ResNet architecture to improve its decoding precision and accuracy, we use 7-layer and 14-layer residual networks to obtain a better result. In our research, the decoding efficiency of the distributed strategy decoder is 5% higher than that of the MWPM decoder and 2% higher than that of the recurrent neural network decoder. We get that the threshold of distributed strategy decoder is 0.0052, and the thresholds of the MWPM decoder and the recurrent neural network decoder are 0.0049 and 0.0051.respectively. Future research can apply the distributed decoding strategy to other types of topological quantum error-correcting codes, such as cat code, bosonic code, etc. The neural network used can also be replaced for better error correction.

Acknowledgement

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

References

[1] Steane A 1998 *Reports on Progress in Physics* **61** 117
[2] Chuang I L and Yamamoto Y 1995 *Phys. Rev. A* **52** 3489
[3] Montanaro A 2016 *npj Quantum Information* **2** 1
[4] Wiebe N, Braun D and Lloyd S 2012 *Phys. Rev. Lett.* **109** 050505
[5] Steane A 1998 *Reports on Progress in Physics* **61** 117
[6] Devitt S J, Munro W J and Nemoto K 2013 *Reports on Progress in Physics* **76** 076001
[7] Cory D G, Price M D, Maas W, Knill E, Laflamme R, Zurek W H, Havel T F and Somaroo S S 1998 *Phys. Rev. Lett.* **81** 2152
[8] Terhal B M 2015 *Rev. Mod. Phys.* **87** 307

[9] Roffe J 2019 *Contemporary Physics* **60** 226
[10] Varsamopoulos S, Bertels K and Almudever C G 2019 *IEEE Transactions on Computers* **69** 300
[11] O'Brien T E, Tarasinski B and DiCarlo L 2017 *npj Quantum Information* **3** 39
[12] Yoder T J and Kim I H 2017 *Quantum* **1** 2
[13] Qu Y J, Chen Z, Wang W J and Ma H Y 2023 *Chin. Phys. B* **32** 10
[14] Chamberland C, Kubica A, Yoder T J and Zhu G 2020 *New J. Phys.* **22** 023019
[15] Brown B J, Nickerson N H and Browne D E 2016 *Nat. Commun.* **7** 12302
[16] Cai W, Ma Y, Wang W, Zou C L and Sun L 2021 *Fundamental Research* **1** 50
[17] Horsman D, Fowler A G, Devitt S and Van Meter R 2012 *New J. Phys.* **14** 123011
[18] Fowler A G, Mariantoni M, Martinis J M and Cleland A N 2012 *Phys. Rev. A* **86** 032324
[19] Tomita Y and Svore K M 2014 *Phys. Rev. A* **90** 062320
[20] Bravyi S, Suchara M and Vargo A 2014 *Phys. Rev. A* **90** 032326
[21] Duclos-Cianci G and Poulin D 2010 *Phys. Rev. Lett.* **104** 050504
[22] Dennis E, Kitaev A, Landahl A and Preskill J 2002 *Journal of Mathematical Physics* **43** 4452
[23] Wang H, Song Z, Wang Y, Tian Y and Ma H 2022 *Quantum Information Processing* **21** 280
[24] Tan X, Zhang F, Chao R, Shi Y and Chen J 2023 *PRX Quantum* **4** 040344
[25] Baireuther P, O'Brien T E, Tarasinski B and Beenakker C W J 2018 *Quantum* **2** 48
[26] Ji N, Chen Z, Qu Y, Bao R, Yang X and Wang S 2023 *Frontiers in Physics* **11** 1164567
[27] Varsamopoulos S, Criger B and Bertels K 2017 *Quantum Science and Technology* **3** 015004
[28] Chamberland C and Ronagh P 2018 *Quantum Science and Technology* **3** 044002
[29] Torlai G and Melko R G 2017 *Phys. Rev. Lett.* **119** 030501
[30] Cao Q, Wang H W, Qu Y J, Xue Y J and Wang S M 2022 *Quantum Engineering* **2022** 6245336
[31] Baireuther P, Caio M D, Criger B, Beenakker C W J and O'Brien T E 2019 *New J. Phys.* **21** 013003
[32] Li A, Li F, Gan Q and Ma H 2023 *Applied Sciences* **13** 9689
[33] Wang H, Xue Y, Qu Y, Mu X and Ma H 2022 *npj Quantum Information* **8** 134
[34] Varsamopoulos S, Bertels K and Almudever C G 2019 *IEEE Transactions on Computers* **69** 300
[35] Varsamopoulos S, Bertels K and Almudever C G 2020 *Quantum Machine Intelligence* **2** 1
[36] Ryan-Anderson C, Bohnet J G, Lee K, *et al.* 2021 *Phys. Rev. X* **11** 041058
[37] Terhal B M 2015 *Rev. Mod. Phys.* **87** 307
[38] Cleve R 1997 *Phys. Rev. A* **55** 4054
[39] Bombin H, Duclos-Cianci G and Poulin D 2012 *New J. Phys.* **14** 073048
[40] Xue Y J, Wang H W, Tian Y B, Wang Y N, Wang Y X and Wang S M 2022 *Quantum Engineering* **2022** 7643871
[41] Kitaev A Y 2003 *Annals of Physics* **303** 2
[42] Chamberland C, Zhu G, Yoder T J, Hertzberg J B and Cross A W 2020 *Phys. Rev. X* **10** 011022
[43] Bravyi S B and Kitaev A Y 1998 arXiv:9811052 [quant-ph]
[44] Katsuda M, Mitarai K and Fujii K 2022 arXiv:2204.11404 [quant-ph]
[45] Krastanov S and Jiang L 2017 *Scientific Reports* **7** 11003
[46] Overwater R W J, Babaie M and Sebastiano F 2022 *IEEE Transactions on Quantum Engineering* **3** 1
[47] Duclos-Cianci G and Poulin D 2013 arXiv:1304.6100 [quant-ph]
[48] Duivenvoorden K, Breuckmann N P, and Terhal B M 2018 *IEEE Transactions on Information Theory* **65** 2545
[49] Hastings M B 2007 *Phys. Rev. B* **76** 201102
[50] Wootton J 2015 *Entropy* **17** 1946
[51] Graves A 2012 *Supervised Sequence Labelling with Recurrent Neural Networks* **385** 37