applied sciences

IMPACT FACTOR 2.5

CITESCORE 5.5

Article

Resource-Efficient Decoding of Topological Color Codes via Neural-Guided Union-Find Optimization

Minghao Fu, Cewen Tian, Zaixu Fan and Hongyang Ma

Topic

Quantum Information and Quantum Computing, 2nd Volume

Edited by

Dr. Durdu Guney and Dr. David Petrosyan

MDPI

https://doi.org/10.3390/app15168937

applied sciences

MDPI

Article

Resource-Efficient Decoding of Topological Color Codes via Neural-Guided Union-Find Optimization

Minghao Fu [1], Cewen Tian [1], Zaixu Fan [1] and Hongyang Ma [2,*]

[1] School of Information and Control Engineering, Qingdao University of Technology, Qingdao 266033, China
[2] School of Science, Qingdao University of Technology, Qingdao 266033, China
* Correspondence: mahongyang@qut.edu.cn

Abstract

Quantum error correction (QEC) is crucial for achieving reliable quantum computation. Among topological QEC codes, color codes can correct bit-flip and phase-flip errors simultaneously, enabling efficient resource utilization. However, existing decoders such as the Union–Find (UF) algorithm exhibit limited accuracy under high noise levels. We propose a hybrid decoding framework that augments a modified UF algorithm—enhanced with a secondary growth strategy—with a lightweight recurrent neural network (RNN). The RNN refines the error chains identified by UF, improving resolution without significantly increasing computational overhead. The simulation results show that our method achieves notable accuracy gains over baseline UF decoding, particularly in high-error regimes, while preserving the near-linear runtime scaling and low memory footprint of UF. At higher physical error rates, RNN-based path optimization improves UF decoding accuracy by approximately 4.7%. The decoding threshold of the color code reaches 0.1365, representing an increase of about 2% compared to UF without RNN optimization. With its simple data structure and low space complexity, the proposed method is well suited for low-latency, resource-constrained quantum computing environments.

Keywords: quantum error correction; color code; union-find decoder; recurrent neural network; decoding threshold; fault-tolerant quantum computing

Academic Editors: Durdu Guney and David Petrosyan

Received: 2 July 2025
Revised: 8 August 2025
Accepted: 11 August 2025
Published: 13 August 2025

Citation: Fu, M.; Tian, C.; Fan, Z.; Ma, H. Resource-Efficient Decoding of Topological Color Codes via Neural-Guided Union-Find Optimization. *Appl. Sci.* **2025**, *15*, 8937. https://doi.org/10.3390/app15168937

1. Introduction

Quantum error-correcting codes (QECCs) are encoding schemes designed to protect quantum information from noise-induced errors. By redundantly encoding quantum information across multiple qubits, this redundancy enables error detection and correction without destroying the underlying quantum state. Ancillary qubits are employed to measure the error states of quantum systems, thereby acquiring error information while preserving the encoded quantum state. Subsequent comprehensive measurement operations determine the presence and nature of errors, allowing corresponding corrective actions to be applied and thus achieving effective error detection and mitigation. Such encoding schemes play a critical role in quantum computing [1–3].

QECCs encompass a diverse range of code types, including Shor codes, Steane codes, surface codes, and five-qubit codes. The color codes investigated in this paper, along with surface codes, belong to the class of geometric structure-based quantum error-correcting codes. Their primary advantage lies in the ability to simultaneously correct multiple types of quantum errors, including bit flips (X errors), phase flips (Z errors), and their combined forms (Y errors). This ability provides strong robustness against multi-error patterns. The

main differences between color codes and surface codes lie in their physical implementation and structural design: surface codes typically employ two-dimensional square or hexagonal lattices, where X-stabilizer and Z-stabilizer information is stored in two distinct types of faces (or vertices), with qubits and stabilizers arranged in a checkerboard configuration. In contrast, color codes are often implemented using two- or three-dimensional colored lattices. A typical structure uses face coloring—usually a three-color scheme—to ensure that adjacent faces have different colors. Their high symmetry allows for the unified implementation of X and Z operations [4–6]. Although surface codes generally exhibit higher error thresholds, their substantial redundancy requirements lead to significant resource consumption. In comparison, while color codes have slightly lower threshold stability, they can simultaneously correct both phase and amplitude errors and handle certain complex error patterns. Moreover, their stabilizer arrangement makes their resource requirements more compatible with conventional quantum computing environments.

For surface codes, numerous decoders have been developed due to their relative simplicity, such as the Minimum Weight Perfect Matching (MWPM) method, the Renormalization Group, Markov Chain Monte Carlo, and various approaches utilizing belief propagation, neural networks, or hierarchical designs [7–9]. In contrast, decoding strategies for color codes remain relatively limited. This paper employs the Union-Find (UF) approach to detect non-trivial measurement points in color codes, using four-round growth and quadratic growth methods to identify the most probable error paths. Additionally, a Recurrent Neural Network (RNN) is employed to perform deep refinement of the error paths identified by UF, thereby achieving the efficient decoding of color codes.

Compared with the Hard-Decision Renormalization Group (HDRG) decoder proposed by Anwar, which employs hierarchical clustering and merging strategies to decode topological quantum codes [10], this paper adopts a distinct approach to decoding color codes. Anwar's method applies local rules to segment error syndromes and progressively merge clusters, which may introduce additional complexity and result in suboptimal decoding performance under certain noise conditions. By contrast, this work integrates the UF-based defect detection framework with a Recurrent Neural Network (RNN) to more accurately predict error chains. The UF structure efficiently identifies connected error paths, while the RNN improves decoding accuracy by learning complex error correlations beyond local recursions.

Recent advances in quantum error correction have explored various machine learning-based decoders, including Convolutional Neural Networks (CNNs), Transformer-based models, and advanced RNN variants such as bidirectional or gated architectures. CNN decoders excel at capturing spatial patterns within local syndrome regions but often struggle with long-range correlations in topological codes such as color codes. Transformer decoders offer strong global context modeling through self-attention but require significant computational resources and large datasets, limiting their scalability in resource-constrained quantum hardware environments. Previous RNN-based decoders, including Gated Recurrent Units (GRUs) and bidirectional LSTMs, have shown promise in modeling sequential syndrome evolution but often incur higher inference latency or increased parameter complexity.

The novelty of this work lies in introducing a secondary-growth strategy into the Union-Find (UF) decoding process and coupling it with a lightweight two-layer LSTM-based recurrent neural network (RNN) for path-level error refinement in color codes. Unlike conventional UF decoders, which often misidentify correlated error chains in high-noise regimes, the proposed Neural-Guided Union-Find (NGUF) decoder leverages temporal learning to selectively optimize only the ambiguous paths suggested by UF clustering. This targeted enhancement preserves the near-linear time complexity and low space overhead

of classical UF decoding while significantly improving decoding accuracy and modestly increasing the decoding threshold. Overall, the contributions of this work include the following: (i) proposing a hybrid UF–RNN decoding framework with a novel secondary-growth mechanism tailored to the structural properties of color codes, and (ii) achieving a favorable balance between decoding accuracy and computational efficiency, making it suitable for real-time and resource-constrained quantum computing.

The structure of this paper is organized as follows. Section 2 systematically overviews color codes, the Union-Find (UF) algorithm, and related background knowledge. Section 3 elaborates on the construction logic and core principles of the joint decoding framework combining the UF algorithm and Recurrent Neural Network (RNN) designed specifically for color codes. Section 4 presents an analysis and discussion based on the simulation results, compares the joint UF–RNN decoding of color codes with standalone UF decoding, and examines their respective advantages, disadvantages, and underlying causes. Finally, Section 5 summarizes the conclusions of this paper.

2. Background

2.1. Color Code

As a type of topological quantum error-correcting code, color codes represent an instance of three-dimensional and higher-dimensional topological quantum codes, exhibiting favorable error correction capabilities and symmetry properties. Despite similarities with surface codes, their geometric and topological characteristics endow color codes with unique performance advantages. Color codes are typically defined on a three-colored planar graph or lattice, where the faces of the color code are divided into three colors—usually red, green, and blue—to ensure that any two faces sharing an edge have distinct colors [11,12]. Qubits are placed on the vertices of the lattice, with X-type and Z-type stabilizer generators associated with each face. Each unit cell has two relevant projectors, defined as follows:

$$S_p^X = \frac{1}{2}\left(1 + \prod_{i \in p} X(i)\right), S_p^Z = \frac{1}{2}\left(1 + \prod_{i \in p} Z(i)\right) \quad (1)$$

Figure 1a presents a two-dimensional lattice structure with either four faces or eight faces, where each vertex corresponds to a qubit. The two dashed lines of each face in the figure represent the X stabilizer and Z stabilizer (a specific example is shown in the region outlined by the red box, which contains a total of 16 qubits and is associated with 8 stabilizers) [13]. If no errors occur, each stabilizer in the lattice is in the +1 eigenstate.

Figure 1b presents a typical error example in color codes. In color codes, three colors (red, blue, and green) are assigned to different faces, ensuring that adjacent faces have distinct colors, and each color category of faces contains both X-type and Z-type stabilizers. When a qubit experiences an X or Z error, if no X or Z errors occur in the adjacent qubits sharing a face with it, the X or Z stabilizer eigenvalues of the three faces it belongs to (corresponding to the three colors) will flip to -1, as shown in the example at the lower right of Figure 1b. If other qubits with X or Z errors exist in the adjacent shared faces of this qubit, the stabilizer eigenvalues of the three faces will flip based on the parity of the number of error qubits, as shown in the example in the lower left of Figure 1b. When a large number of error qubits exist within the same region, these errors interact with each other, eventually forming an extended error region (as shown in the example in the upper region of Figure 1b) [14–16]. In other words, for color codes, a single qubit error affects a larger area of the stabilizers than in surface codes [17]. Therefore, when generating error clusters, we must not only consider the range of single-step growth but also secondary growth—a deliberate second expansion phase triggered when residual correlations remain between clusters after the first growth. This step allows for the detection and linking of

error regions that are spatially separated but statistically correlated, which is particularly important in color codes, where a single qubit error can influence multiple stabilizers across different faces.

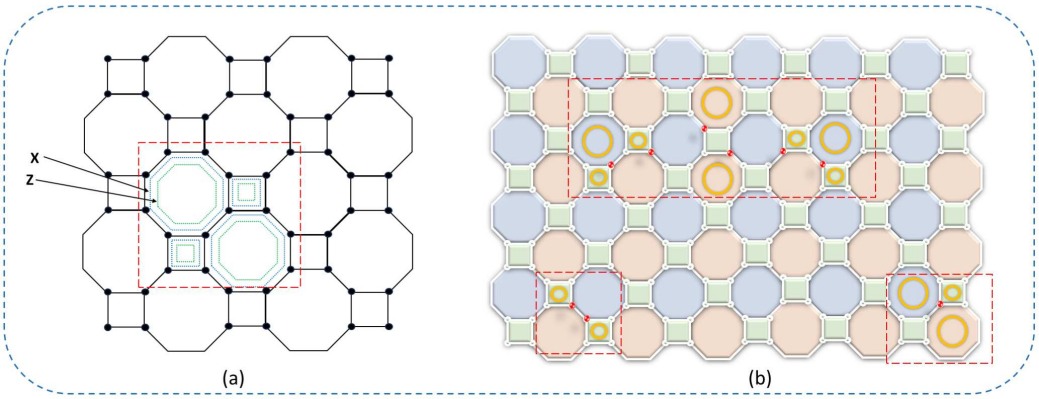

Figure 1. Color-coded markers and error types. (**a**) 4-8-8 structure color code layout. (**b**) Example of errors and syndromes, with the yellow circle marking the qubit of interest, the red box the affected stabilizer region, and the red dot the flipped stabilizer eigenvalue.

2.2. Union-Find Algorithm

Conceptually, the limitation of the Union-Find (UF) algorithm lies in spatial clustering; syndromes can only be found at the edges of the cluster. During decoding, UF maintains the total size of all clusters small to ensure the correctness of the correction [18,19]. The algorithm includes the following stages:

- An error cluster expands uniformly in all directions at the same speed from each defect.
- Spanning trees grow within each cluster. Parity checks are performed when determining whether a cluster is active.
- Peeling the edges of the error cluster involves removing them from the tree, and these edges are either included in or excluded from the correction based on the syndrome.

The UF decoder typically operates on two independent graph structures: the X-check graph and the Z-check graph. During the decoding process, check nodes must first be determined: if the measurement result of a stabilizer element corresponds to a trivial measurement (i.e., eigenvalue +1), the corresponding node is defined as an even check node; conversely, if the measurement result of a stabilizer element corresponds to a non-trivial measurement (i.e., eigenvalue -1), the corresponding node is defined as an odd check node. If cluster C has an odd defect count and does not touch the boundary, then cluster C is active [20–23], i.e.,

$$(|S \cap V_C| \text{ odd}) \wedge \left(V_{\text{boundary}} \cap V_C = \varnothing\right) \tag{2}$$

The core mechanism of the UF decoder is based on the growth of error clusters: all odd check nodes are initially considered as independent error clusters, with each error cluster containing only a single odd check node in the initial state. Subsequently, growth operations are performed on each error cluster. Since there are typically multiple independent error clusters in the color code system, when two error clusters come into contact during the growth process, the cluster merging mechanism is triggered [24–26]. Mathematically, if clusters A and B merge along edge e, the resulting cluster is

$$C = \left(V_A \cup V_B, E_A \cup E_B \cup \{e\}\right) \tag{3}$$

When a cluster grows to the virtual boundary, or the parity check of the merged cluster becomes even, or one of the merged clusters reaches the virtual growth boundary, the

cluster will be frozen, i.e., cease growth [27–29]. Once the growth process of the Union-Find is completed, a peeling operation is performed on the grown tree structure formed in this stage. For the resulting peeled tree, its topological structure fully retains the distribution characteristics of error qubits, and it is ultimately defined as an error path containing error qubits [30–32].

3. Hybrid Decoding Architecture Design

3.1. Cluster Growth Dynamics in 2D Color Codes

The growth process in the proposed decoder adopts a two-stage strategy. The first stage, primary growth, begins with each detected non-trivial measurement point expanding uniformly in all directions, thereby forming the initial error clusters. This is followed by secondary growth, a targeted expansion applied to clusters that remain statistically correlated after the primary stage. This additional step is designed to capture latent error connections that might otherwise be overlooked, thereby reducing the risk of under-connecting correlated errors—an inherent limitation of conventional Union–Find decoders, particularly under high-noise conditions.

An illustrative example is shown in the top panel of Figure 2, which depicts the stabilizer verification for the X-check graph in a color code. Yellow circles indicate non-trivial measurement points. The decoding process begins by identifying these points and initiating outward growth from each—starting from data qubits adjacent to the non-trivial checks and extending to neighboring check qubits. Growth from all non-trivial measurement points occurs simultaneously, and after the first growth cycle, distinct error clusters emerge, i.e.,

$$\text{access}\, v := \{u \in V : uv \in E \wedge uv.\text{growth} = 1\} \tag{4}$$

Steps 1, 2, and 3 in Figure 2 provide an example of the stabilizer verification process. Mathematically, a cluster is a connected subgraph of G. A cluster is active when using only the edges within the cluster and without correcting its defects, i.e.,

$$\nexists \mathbb{C} \subseteq E_C : \sigma(\mathbb{C}) = \mathbb{S} \cap V_C \tag{5}$$

Based on the experimental observations in Step 1 of Figure 2, the error clusters located in the lower-left and lower-right regions terminate their growth upon reaching the virtual boundary conditions. In contrast, the three initial error clusters positioned at the top proceed to a second growth phase, driven by the risk of statistical correlation. This situation arises when the minimum error-chain length required to connect two root node sets falls below a defined threshold [33]. The threshold is determined to ensure that no residual correlation remains between the two spanning trees during Union–Find decoding, and is calculated as follows:

$$t_{\min} = \min\left(\frac{d_{\min}}{2}, \frac{-\ln \epsilon}{\ln(1/p)}\right) \tag{6}$$

As shown in Step 2 of Figure 2, the second growth phase causes the three initial error clusters to merge into a single connected cluster via expansion paths, where orange segments indicate the cluster boundaries formed during the first growth stage and purple segments denote the newly generated expansion paths from the second stage. Upon termination of the growth process, only the effective paths that connect distinct error clusters are preserved, while redundant connections generated by multiple growth cycles are pruned. After the second expansion stage, the merged upper cluster continues to grow until it reaches the virtual boundary, thereby satisfying the stopping condition [34,35]. As illustrated in Step 3 of Figure 2, the red-marked segments represent the effective connection

paths retained after multiple rounds of growth. Notably, once clusters are merged, each large cluster corresponds to a correlated set of errors. Due to the structural properties of the color code lattice (e.g., the high connectivity of hexagonal lattices), multiple potential connection paths may exist within the same cluster. At this stage, the minimum-weight path—typically defined by path length or the inverse of the associated error probability—is selected, and all redundant paths are discarded. This marks the completion of the correction operator verification process.

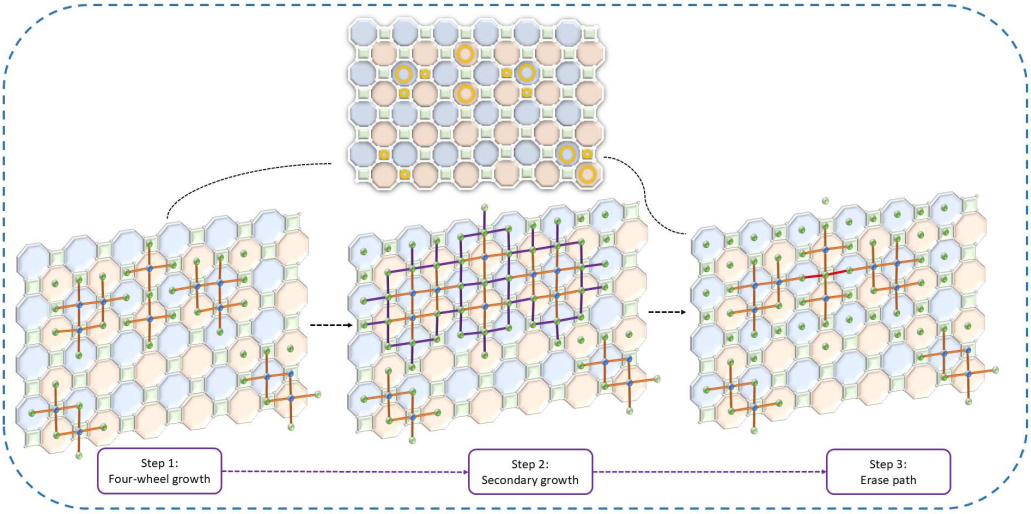

Figure 2. Union-Find decoding process for topological color codes: syndrome points undergo four growth rounds (including secondary growth); after cluster fusion, redundant spanning trees are removed, yielding error chains consistent with the syndrome topology.

3.2. Cluster Stripping Dynamics in 2D Color Codes

In the implementation process of the peeling decoder, the definition of leaf nodes in spanning trees must first be clarified: spanning trees, as undirected, connected, and acyclic graph structures, have leaf nodes defined as vertices with a degree of 1 (i.e., terminal vertices connected by only one edge) [36–38]. The specific decoding process is as follows: select any leaf node in the spanning tree as the initial root node (if multiple spanning trees exist, choose one for processing). Starting from the root node, leaf nodes are peeled layer by layer, with two scenarios categorized based on the measurement states of the leaf node.

Non-Trivial Measurement Leaf Node: If the current leaf node is a non-trivial measurement point, record the information of its adjacent edge (the two qubits connected by this edge are potential Pauli error candidate positions). Additionally, flip the measurement states of this non-trivial measurement point and its parent node (i.e., converting the non-trivial measurement point to a trivial measurement point, and the original trivial measurement point of the parent node to a non-trivial measurement point).

Trivial Measurement Leaf Node: If the current leaf node is a trivial measurement point, directly perform the peeling operation to remove the connecting edge between the leaf node and its parent node.

Repeat the above peeling operations until the spanning tree is completely peeled down to only a single vertex, completing the decoding process. For non-terminal nodes, we calculate the expected value as follows:

$$V(s_1) = \sum_{s_2} P(s_2|s_1)(r(s_1, s_2) + V(s_2)) \tag{7}$$

$P(s_2|s_1)$ needs to be specified. If s1 is a stripping process, then $P(s_2|s_1)$ is determined by the stripping model. If s1 is in the process of growth, then $P(s_2|s_1)$ is determined by choosing the best place to grow, i.e.,

$$P(s_2|s_1) = \begin{cases} 1 & \text{if} s_2 = arg\max_s r(s_1, s) + V(s) \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

As shown in Figure 3, the peeling process for syndrome verification is illustrated. The left diagram depicts the peeling process of spanning trees, with these trees visualized using distinct colors: green edges denote those associated with leaves, while brown edges represent the trunk of the tree. Additionally, arrows indicate the peeling direction from the tree root to the tree leaves, and the error path post-peeling is presented in the right diagram.

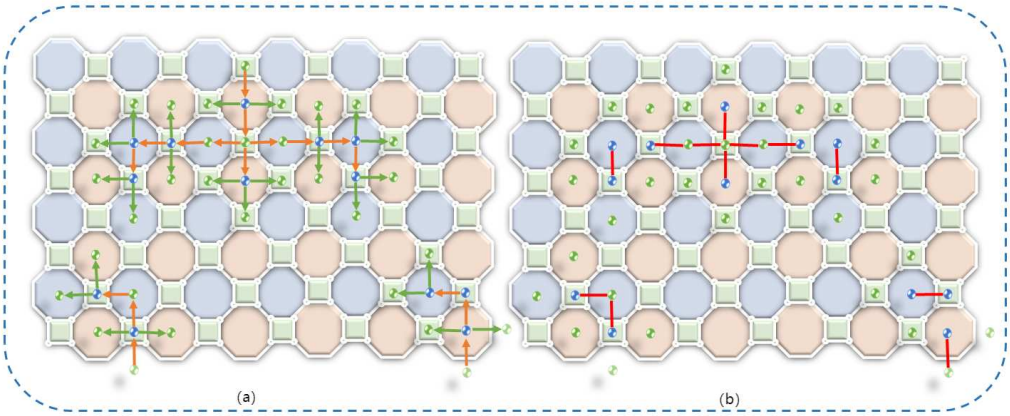

Figure 3. Spanning tree peeling in Union-Find decoding of color codes. (**a**) Peeling direction from root to leaves (green arrows) and associated error propagation paths (red arrows). (**b**) Final error paths after peeling completion (red arrows).

3.3. RNN-Enhanced Error Pattern Recognition

In Section 3.1, we discussed the application of the Union-Find (UF) algorithm to predict error paths in color codes, which involves peeling from the leaf nodes of the spanning tree, processing the measurement results, and identifying potential paths of Pauli errors. However, relying solely on the UF decoder has significant limitations: its merging performance is highly dependent on the definition of "edges" (e.g., the weights of nearest-neighbor edges), which may lead to failure in accurate error correction when the error density exceeds half the code distance. In contrast, neural networks can dynamically generate superior edge weights or connection rules for UF by learning from error data; detailed steps will be presented in Section 3.4 and the complete error correction process is illustrated in Figure 4.

We utilize a Recurrent Neural Network (RNN), which effectively processes qubit information in color codes. Its memory properties help address the issue of information continuity between processing rounds, thereby bringing us closer to the ideal output. The rationale for choosing RNN lies in its ability to efficiently decode without prior knowledge of the number of iterations. In this work, we adopt an RNN architecture incorporating long short-term memory (LSTM) layers. LSTM is a specialized RNN model: unlike conventional RNNs, where the repeating neural network module has a very simple structure, LSTM enhances this structure by replacing the single neural layer with four interacting components. LSTM primarily consists of three distinct gating mechanisms: the forget gate, input gate, and output gate. These gates regulate the retention and transmission of information within

the LSTM, ultimately influencing the cell state and output signals [39–41], as illustrated in Figure 5.

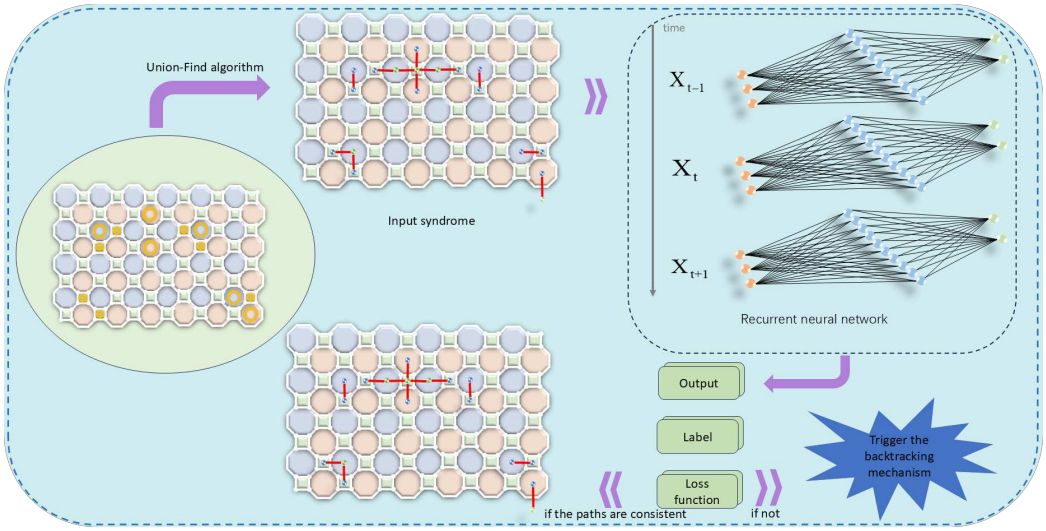

Figure 4. A hybrid quantum error correction protocol for topological color codes. The red line represents the feedback path from the neural network to the UF decoder.

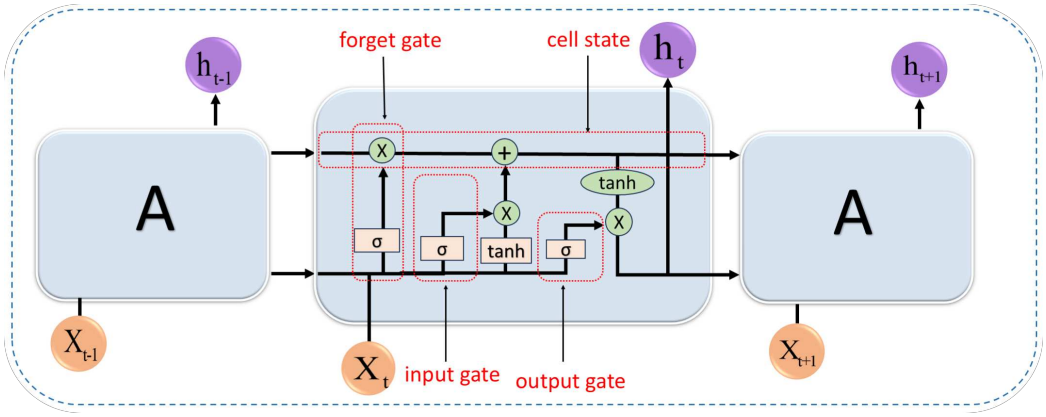

Figure 5. Recurrent neural network architecture for long and short term memory layer.

In the first step of LSTM, the system needs to determine which information to discard from the cell state, a decision made by the "forget gate" structure. Specifically, the forget gate takes the output from the previous time step (i.e., the error probability of qubits containing Pauli errors) and the input at the current time step (i.e., the error path) as inputs. After processing through an activation function (here, a sigmoid non-linear mapping), it outputs the forget gate value $f(t)$ [42]. Since the output $f(t)$ of the sigmoid function has the range [0, 1], this value directly characterizes the forgetting probability of the hidden cell state from the previous layer. Mathematically, this can be expressed as follows:

$$f^{(t)} = \sigma(W_f h^{(t-1)} + U_f x^{(t)} + b_f) \tag{9}$$

Next, the input gate determines what new information to store in the cell state; in this experiment, this new information specifically refers to newly added error path information. The operation of the input gate involves two components: the first component generates the output i(t) through a sigmoid activation function, and the second component generates the output a(t) through a tanh activation function; ultimately, the update to the cell state

is achieved through the product of the two [43]. Mathematically, this can be expressed as follows:

$$i^{(t)} = \sigma(W_i h^{(t-1)} + U_i x^{(t)} + b_i) \tag{10}$$

$$a^{(t)} = tanh(W_a h^{(t-1)} + U_a x^{(t)} + b_a) \tag{11}$$

Before entering the output gate phase, it is necessary to compute the new cell state of the LSTM (the memory state after incorporating new error path information) [44]. At this stage, the results of the forget gate and input gate collectively act on the cell state C(t),which is composed of two parts: the first part is the product of the previous cell state C(t−1) and the forget gate output f(t), and the second part is the product of the input gate outputs i(t) and a(t). That is,

$$C^{(t)} = C^{(t-1)} \odot f^{(t)} + i^{(t)} \odot a^{(t)} \tag{12}$$

Finally, the output value of the LSTM needs to be determined. This output value is based on the current cell state: first, the sigmoid layer determines which part of the cell state should be outputted; subsequently, the cell state undergoes tanh processing (mapped to the $[-1, 1]$ range), and the processed value is multiplied by the output of the sigmoid gate. The final output is this product result [45]. That is,

$$o^{(t)} = \sigma(W_o h^{(t-1)} + U_o x^{(t)} + b_o) \tag{13}$$

$$h^{(t)} = o^{(t)} \odot tanh(C^{(t)}) \tag{14}$$

3.4. Training Dataset and Learning Methodology

To improve decoding precision beyond classical Union-Find (UF) decoding, we developed a Recurrent Neural Network (RNN)-based module that infers fine-grained qubit-level error probabilities along candidate paths extracted from the UF decoder. This module adopts a two-layer Long Short-Term Memory (LSTM) architecture, with each layer comprising 128 hidden units, effectively capturing sequential syndrome dependencies and enabling the memory of long-range correlations. A dropout rate of 0.2 is applied between layers to mitigate overfitting. The final hidden state is passed through a fully connected linear output layer with a sigmoid activation to produce error probabilities for each qubit in the range $[0, 1]$.

The model was trained using the Adam optimizer with hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and an initial learning rate of 0.001. We employed cosine annealing to decay the learning rate by 10% every 10 epochs. Training proceeds for a maximum of 200 epochs or is terminated early if the validation loss plateaus for 20 consecutive epochs. A batch size of 40 is used for stable and efficient gradient descent.

Training labels are generated using a depolarizing noise model applied to various color code lattices. Each decoding instance simulates a complete error injection and syndrome extraction cycle, from which we can extract the ground-truth Pauli error positions. The qubits along UF-generated candidate paths are then assigned real-valued error probability labels as regression targets. Crucially, model correctness is not defined by alignment with the UF decoder output but rather by consistency with the underlying physical error configuration.

The training and evaluation datasets span color codes with distances $d = 5$ to $d = 11$, comprising approximately 8000 training samples and 2000 held-out validation and test samples. Each sample encodes a candidate path formed by linking non-trivial syndrome nodes via UF clustering, with input features constructed as spatial vector sequences. These samples cover a broad spectrum of error densities and cluster complexities, from sparse isolated faults to dense ambiguous regions that require deeper RNN inference.

All experiments were conducted on a high-performance computing environment comprising an Intel® Xeon® Silver 4210R CPU (Intel Corporation, Santa Clara, CA, USA), 256 GB of RAM, and an NVIDIA® RTX A6000 GPU with 48 GB VRAM (NVIDIA Corporation, Santa Clara, CA, USA).

We used PyTorch 2.0.1 with CUDA 11.8 on Python 3.10 and Ubuntu 20.04. Each complete training cycle typically takes 4–6 h, depending on code distance and path topology. Although our implementation and models are not yet publicly released, we intend to share them in a future open-source repository to promote transparency and reproducibility.

The proposed RNN decoder enhances the ability of classical decoders by dynamically refining error path predictions, particularly in overlapping or complex clusters where UF alone may fail. By learning from realistic noise statistics, the LSTM-based model significantly improves error localization and contributes to the robustness of the hybrid decoding architecture.

3.5. Sampling Strategy and RNN Decoder Architecture

In this study, we employed a depolarizing error model to generate the training dataset for training the RNN, aiming to enhance the prediction accuracy of qubit errors. In this experiment, physical errors were simulated by incorporating depolarizing noise channels into the quantum circuit. The construction of this error synthesis path and the determination of the true state of data qubits ensure that the training data can authentically reflect the error distribution of qubits during operation. For dataset sampling, we collected qubit error data by simulating multiple quantum error correction cycles during training dataset generation. In each cycle, we used the UF decoder to perform preliminary decoding on the color code, then ran and recorded the logical qubits with errors and their error paths to generate candidate error bit paths. However, the error locations provided by the UF decoder are not entirely accurate; thus, we further predicted these paths using the RNN. The overall decoding workflow, integrating the Union-Find and RNN modules, is illustrated in Figure 6, which outlines the sequential stages from syndrome extraction and UF-based clustering to RNN-guided path refinement and final error correction.

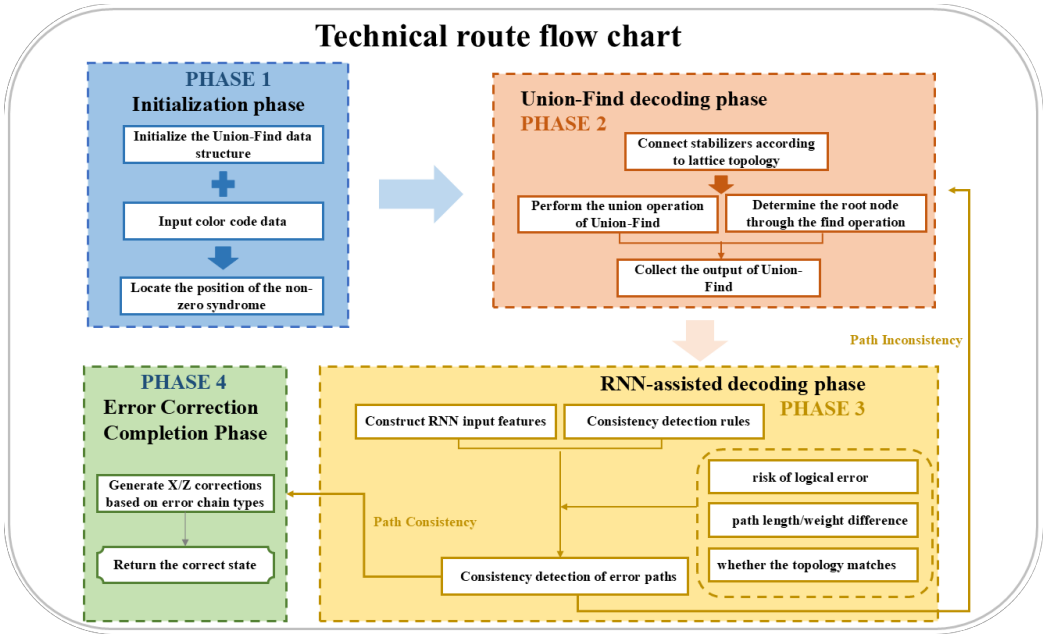

Figure 6. Hybrid decoding of color codes: syndromes are processed by UF to identify candidate error chains, refined by an RNN, and corrected to recover the logical state.

To construct the dataset, we generated corresponding stabilizer and syndrome data for color code structures with different code distances. Figure 7 illustrates the stabilizer arrangement and path input for a 5 × 5 color code. Each error path is composed of stabilizer syndromes, and we constructed these into training sample paths to be input based on the output of the UF decoder. For example, when an 'X' error is detected, the UF decoder marks non-trivial measurement points, and we then connect adjacent non-trivial measurement points along the path to form the vector information input to the RNN. For more complex error clusters, we extended the path to the quadratic growth region to enable the neural network to more accurately predict the state of erroneous data qubits. When the error cluster structure becomes more complex, we further expanded this to three layers to introduce a multi-layer growth strategy. In each growth phase, we added connecting paths to cover broader potential error regions, ensuring all interconnected non-trivial measurement points were included within the path. Through multi-layer growth, the RNN can capture global error information in complex paths, enhancing the comprehensiveness and accuracy of its predictions.

In the RNN training dataset, we generated a label for each error path, where the label represents the error probability distribution of each data qubit along the path. Each sample in the training dataset includes an error cluster path (formed by connecting non-trivial measurement points) as input, and the error probability of each data qubit (with the error probability value pp being a real number between 0 and 1) as the output label. During the training process, specific training metrics and parameters were set. The initial training epoch was set to 200 or until the termination condition was met. In each epoch, the model performed a complete pass over all training samples, and the difference between the predicted error probability and the true label was measured using the mean squared error (MSE) loss function:

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{15}$$

where y_i is the true error probability, $\hat{y}_i$ is the predicted error probability of the RNN, and N is the total number of data samples. The model adjusts its weights based on the loss value. To evaluate the training effect, we set an accuracy threshold of 0.95. That is, when the RNN model predicts the error-bit position, the match rate with the real error position needs to reach 0.95 or higher. We set the batch size to 40 to stabilize the gradient-descent process and speed up training. The model weights were updated using small-batch random gradient descent. The initial learning rate was 0.001, and it was reduced by 10% every 10 cycles using cosine annealing, ensuring stable convergence, i.e.,

$$\eta_t = \eta_{min}^i + \frac{1}{2}(\eta_{max}^i - \eta_{min}^i)(1 + cos(\frac{T_{cur}}{T_i}\pi)) \tag{16}$$

The training termination conditions were defined as follows: the model achieves a prediction accuracy of over 95% and a loss below 0.01 across multiple validation sets, or the validation loss stops decreasing for 20 consecutive epochs (Early Stopping strategy). After training, the model's predictions were compared with the output of the UF decoder. When the error path positions predicted by the RNN align with those decoded by the UF decoder, the model is considered to have effectively predicted the error locations. If the error paths from the two models are inconsistent, a backtracking mechanism is triggered: if the misjudgment arises from the premature merging of a high-weight edge, the merging operation of that edge is reversed, and the corresponding two clusters are re-split into independent ones. After reversal, the neural network generates new merging priorities to re-execute the decoding process. Following re-execution, the neural network checks the validity of the results again. If the conditions are met, backtracking terminates; if

not, the UF backtracking mechanism continues to be triggered. Considering the real-time nature of information transmission, excessive backtracking may cause the decoding time to exceed the coherence time, leading to quantum state decoherence and offsetting the error correction effect. Thus, termination rounds for unmet conditions are set. The total time for three backtracking iterations is approximately $3n\alpha(n)$, which remains controllable within 1μs during decoding. Therefore, three backtracking iterations were selected in the experiments. Through these settings, the RNN model can accurately predict error paths, thereby enhancing its capabilities in predicting and correcting qubit errors.

To improve reproducibility, we provide the overall pseudocode of the hybrid decoder integrating Union-Find (UF) and Recurrent Neural Network (RNN), as shown in Algorithm 1:

Algorithm 1: UF-RNN Hybrid Decoder for Color Codes

Input: Syndrome set $\mathcal{S}$, trained RNN model $\mathcal{M}$, color code graph G
Output: Predicted Pauli error configuration $\hat{E}$

Initialize each odd-check node $o_i \in \mathcal{S}$ as an individual cluster C_i;
for *each growth round* $r = 1$ *to* 4 **do**
    Grow each active cluster C_i by one layer;
    if *two clusters are adjacent and not frozen* **then**
        Merge them into a larger cluster C_k;
Construct a spanning tree T_k from each merged cluster C_k;
Peel T_k to obtain candidate paths $\mathcal{P}$ (leaf-to-root peeling);
for *each path* $p_i \in \mathcal{P}$ **do**
    Transform p_i into a sequential vector $x^{(t)}$;
    Predict $\hat{y}_i \leftarrow \mathcal{M}(x^{(t)})$ (per-qubit error probability);
    Mark qubit as erroneous if $\hat{y}_i > \theta$ (threshold, e.g., 0.5);
Fuse UF-generated paths and RNN predictions into $\hat{E}$;
if *Mismatch between* $\hat{E}$ *and ground-truth path* **then**
    for *backtracking round* $b = 1$ *to* 3 **do**
        Backtrack last merge step causing misalignment;
        Re-run UF growth, peeling, and RNN prediction;
        if *New* $\hat{E}$ *matches ground-truth* **then**
            Break;
Return Final corrected error configuration $\hat{E}$;

This modular approach ensures low time complexity during UF cluster construction ($O(n\alpha(n))$) and fast inference during RNN path refinement. The RNN module only activates when error clusters form long-range paths or ambiguous merges.

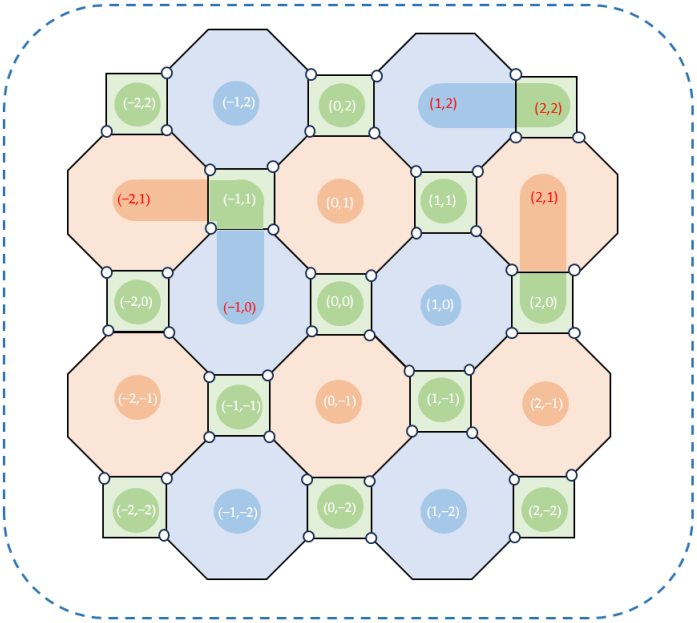

Figure 7. Stabilizer arrangement and path input for a 5×5 color code. Error syndromes are connected into vectors via Union-Find decoding and fed into the RNN, with additional points added to complete discontinuous paths.

4. Numerical Result Analysis

4.1. Training Loss Analysis of the RNN

To address the limitations of the Union-Find decoder, which has reduced prediction accuracy under high error rates, we introduced a recurrent neural network (RNN) architecture based on long short-term memory (LSTM) units, which enhances the prediction accuracy of Pauli error positions. As shown in Figure 8, the training loss curves of the RNN decoder across different code distances exhibit a distinct three-phase convergence pattern. In the initial phase (epochs 1–50), the loss drops sharply, indicating that the model rapidly learns the sequential correlations between non-trivial syndrome points. During the middle training phase (epochs 50–150), the loss gradually decreases as the model refines its internal representations through the LSTM memory mechanism. After epoch 150, the loss stabilizes, suggesting that the model has converged and stabilized its predictive capabilities. Additionally, we observe that for shorter code distances (e.g., $d = 5$), the final converged loss values consistently exceed those of longer code distances (e.g., $d = 17$). This discrepancy arises because larger code distances correspond to larger color code grids, more detection points, and more complex combinations of error paths. Consequently, in the early training stages (first dozens of epochs), larger code distances exhibit higher loss values as the model needs to learn more complex features. As training progresses, the model adapts, and the loss decreases. However, larger code distances benefit from higher-quality labels, leading to lower losses in later stages compared to smaller code distances. In summary, the RNN decoder proposed in this paper demonstrates stable convergence and a strong generalization capability, effectively correcting decoding errors caused by the limitations of the Union-Find decoder.

To quantitatively assess the performance of different decoders under varying physical error rates and code distances, we conducted extensive Monte Carlo simulations. The results are summarized in Table 1, where each entry reports the mean decoding accuracy over 10,000 independent trials, along with the corresponding standard deviation to indicate statistical confidence and the results' variability. We benchmark the classical Union-Find

(UF) decoder, a CNN-based decoder, and the proposed NGUF hybrid decoder. As the physical error rate p increases, decoding accuracy naturally declines across all methods; however, the NGUF consistently outperforms the baselines under all tested conditions.

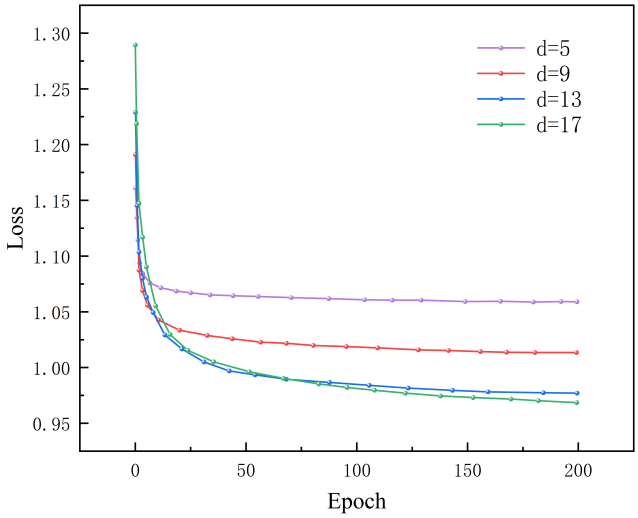

Figure 8. Loss function curves of RNN under code distances λ = 5, 9, 13, 17. The horizontal axis represents the number of training iterations, and the vertical axis represents the loss function.

Table 1. Decoding accuracy (%) with standard deviation under varying physical error rates p, across different code distances. Each result is averaged over 10,000 simulations.

Distance	p	UF Decoder	CNN Decoder	NGUF Decoder
5	0.05	99.42 ± 0.21	99.51 ± 0.18	99.55 ± 0.17
	0.10	98.68 ± 0.32	98.92 ± 0.29	98.73 ± 0.24
	0.15	95.33 ± 0.55	96.45 ± 0.48	97.12 ± 0.42
9	0.05	97.80 ± 0.27	98.55 ± 0.23	98.91 ± 0.19
	0.10	94.21 ± 0.51	96.88 ± 0.43	97.53 ± 0.37
	0.15	89.14 ± 0.69	93.02 ± 0.60	94.35 ± 0.54
13	0.05	95.12 ± 0.34	96.40 ± 0.28	96.91 ± 0.26
	0.10	90.03 ± 0.67	94.11 ± 0.59	95.62 ± 0.45
	0.15	84.23 ± 0.74	90.88 ± 0.63	92.46 ± 0.58
17	0.05	92.23 ± 0.39	94.85 ± 0.31	95.50 ± 0.28
	0.10	85.42 ± 0.73	91.29 ± 0.68	93.44 ± 0.50
	0.15	76.87 ± 0.82	86.04 ± 0.74	89.62 ± 0.61

Notably, under moderate to high noise regimes (e.g., $p = 0.1$) and larger code distances (e.g., $d = 13$, $d = 17$), NGUF achieves accuracy improvements of up to 4.7% over the UF decoder, with consistently lower standard deviations. This indicates that the observed gains are statistically significant. For example, at $d = 13$ and $p = 0.1$, the NGUF decoder achieves $95.62\% \pm 0.45\%$ accuracy compared to $90.03\% \pm 0.67\%$ with UF decoding. The non-overlapping confidence intervals reinforce the statistical robustness of this improvement.

Furthermore, to evaluate the fault tolerance capability of each decoder, we compute the decoding threshold—defined as the physical error rate at which logical error rates become independent of code distance. This is typically identified by observing the intersection point of logical error curves for different code distances. In our study, the threshold for standalone UF decoding is approximately 0.134, while the proposed NGUF framework elevates this to 0.1365. This improvement reflects the NGUF decoder's enhanced ability to correct high-weight error chains, confirming its practical advantage for scalable, fault-tolerant quantum computing.

4.2. Comparison of Decoding Accuracy

The color code exhibits a dual characteristic of high efficiency and instability, and the method proposed in this paper can effectively mitigate its instability. To visually validate the improvement effect of the method, comparative experiments with surface codes were conducted to verify the effectiveness of the proposed approach. As shown in Figure 9, performance comparison experiments are presented under the influence of depolarizing noise, involving three scenarios: decoding surface codes with the UF decoder alone, decoding color codes with the UF decoder alone, and decoding color codes with the combined UF decoder and RNN. The figure compares the decoding accuracy of color codes and surface codes across different code distances (5, 11, 13, 17). As shown in the figure, under relatively low physical error rates, the decoding accuracy of color codes using the UF decoder alone is relatively high, reaching 98.68%. After path optimization via RNN, this accuracy increases to 98.73%. Although still slightly inferior to surface codes, the gap among the three scenarios is minimal. This is primarily attributed to the fact that under low physical error rates, the color code does not form error clusters, reducing the likelihood of errors in determining error positions through error paths. Furthermore, as the code distance increases, stabilizers are distributed more broadly across the 2D plane, further reducing mutual interference, decreasing the formation of error chains, and significantly improving decoding accuracy. When the physical error rate is relatively high, the figure shows that the decoding accuracy of color codes using the UF decoder alone decreases significantly. This is because, compared to surface codes, stabilizers in color codes are arranged more compactly. Once error chains form, the probability of errors in error path determination increases, leading to a notable decline in the performance of the UF decoder alone for color code decoding. Therefore, we introduced RNN to assist the UF decoder in optimizing error paths. The results demonstrate that the decoding accuracy of color codes with RNN path optimization increases by approximately 4.7%, becoming comparable to that of surface codes, while its space occupancy rate remains lower than that of surface codes.

To evaluate the practical efficiency and scalability of the proposed NGUF decoder, we compare the average decoding latency across increasing code distances against baseline methods. As shown in Figure 10, we measure the decoding time required for a single error correction cycle under a fixed depolarizing noise rate of $p = 0.1$, averaged over 10,000 decoding instances for each code distance. All measurements were conducted on an Intel Xeon Silver 4210R CPU using single-threaded inference to simulate low-latency environments. The CNN decoder, although achieving reasonable accuracy, shows a steep increase in runtime due to its heavy computational overhead and large convolutional receptive field. In contrast, the NGUF decoder, while integrating a lightweight RNN module, incurs only a modest runtime increase compared to the classical UF method. This demonstrates the time complexity advantage of the NGUF decoder: by restricting RNN inference only to paths proposed by Union-Find clustering, the model achieves higher decoding accuracy while maintaining near-linear runtime scaling. These results confirm the suitability of our hybrid framework for resource-constrained or real-time fault-tolerant quantum computing scenarios.

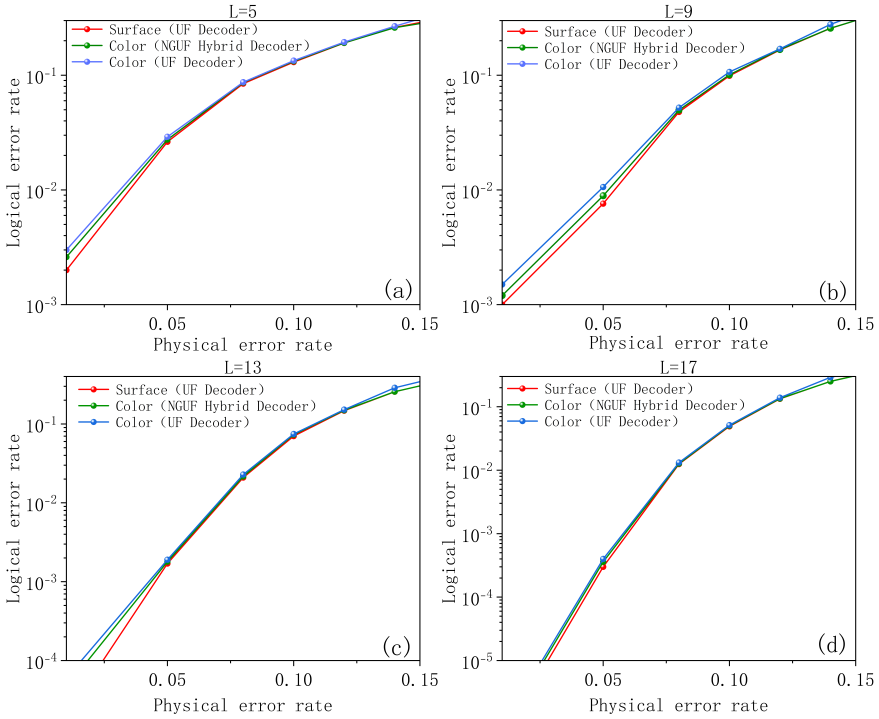

Figure 9. Decoding accuracy of color code and surface code under different decoding methods. (**a**) Decoding accuracy at a code distance of 5. (**b**) Decoding accuracy at a code distance of 9. (**c**) Decoding accuracy at a code distance of 13. (**d**) Decoding accuracy at a code distance of 17.

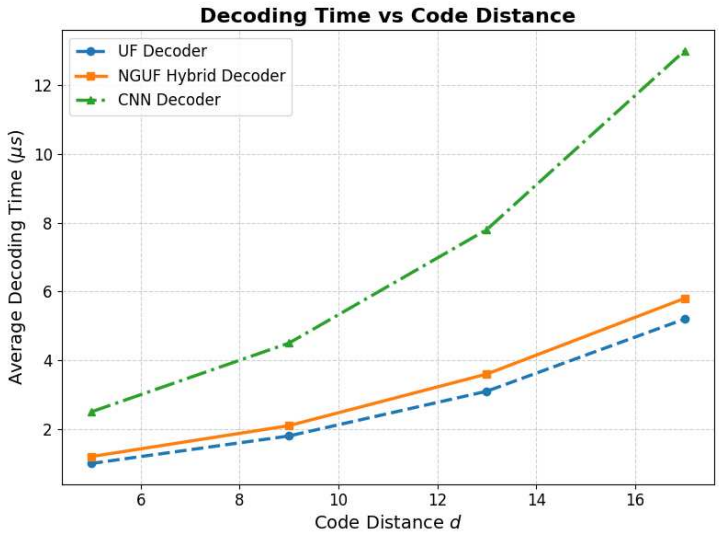

Figure 10. Average decoding time per sample versus code distance (d), comparing Union-Find (UF), CNN-based decoding, and the proposed NGUF hybrid decoder. The NGUF decoder achieves significantly better decoding efficiency than CNN while maintaining a near-linear runtime comparable to UF.

4.3. Analysis of Color Code Decoding Threshold

Building on the subgrid projection method proposed in Ref. [46]—which embeds color code grid points into the subgrid of surface codes—existing studies have demonstrated that the Union-Find (UF) algorithm can effectively perform topological decoding for color codes. Under this framework, Ref. [47] constructed a decoding protocol for (6,6,6)-type color codes, achieving a decoding threshold of 0.084. This work achieves the following

innovative breakthroughs for (4,8,8)-type color codes by improving the cluster growth mechanism of the UF algorithm: a secondary growth criterion is introduced during the error chain expansion phase, dynamically adjusting the correlation between cluster boundaries and stabilizers to enhance the identification of topological correlations between error chains. The numerical simulation results show that, compared with the growth algorithm in Ref. [48], this method enhances robustness against complex error patterns under the same code distance. Additionally, by integrating a Recurrent Neural Network (RNN) model while improving the cluster growth mechanism of the UF algorithm, the decoding threshold is further elevated. As shown in Figure 11, the physical error rates and logical error rates of the joint decoder for color codes with different code distances are presented. It can be observed that the logical error rate increases with the physical error rate. The decoding threshold refers to the physical error rate: the threshold of the UF decoder alone for color codes with different code distances is approximately 0.134, while the threshold of the joint UF-RNN decoder is 0.1365. This is because the RNN introduced in this work is designed to assist the UF algorithm in determining the growth of error clusters. Consequently, the RNN influences the error paths of color codes, thereby affecting their decoding efficiency under high error rates and exerting a certain impact on the threshold error rate of color codes. The results indicate that compared to using the UF decoder alone, the hybrid UF-RNN decoding scheme not only improves decoding accuracy under high physical error rates but also slightly elevates the decoding threshold. The reason for this is that under high physical error rates, the minimum weight ratio method alone struggles to accurately determine multi-qubit error paths; however, the introduction of RNN effectively addresses this issue.

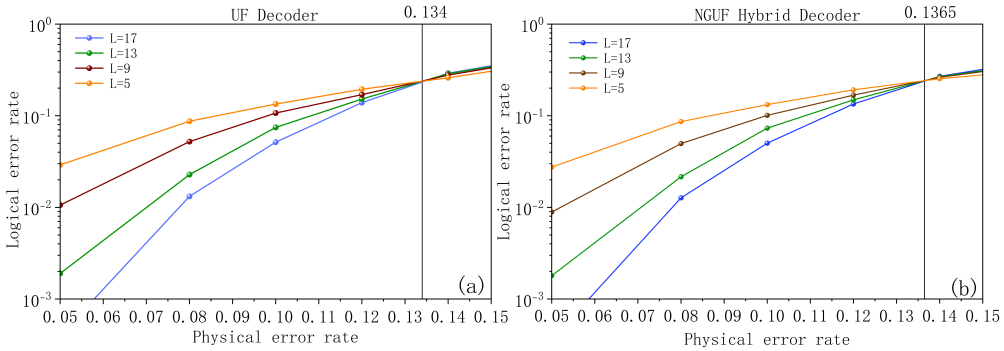

Figure 11. Decodingthresholds of color codes under different decoding methods: (**a**) UF decoding threshold of color codes (**b**) UF combined with RNN decoding threshold of color codes.

5. Conclusions

A hybrid decoding framework was proposed that integrates the Union-Find (UF) algorithm with a Recurrent Neural Network (RNN) to improve the decoding performance of color codes. The experimental results demonstrate that at low physical error rates, standalone UF decoding achieves competitive accuracy with a significantly reduced resource overhead compared to surface codes. Under high-error regimes, the Neural-Guided Union-Find (NGUF) decoder provides a 4.7% improvement in decoding accuracy and increases the threshold from 0.134 to 0.1365, outperforming traditional UF decoding and approaching the performance of surface code decoders. These findings indicate that the hybrid approach effectively addresses the decoding challenges associated with the densely connected structure of color codes, such as error chain propagation and syndrome ambiguity. Beyond accuracy gains, the NGUF framework offers practical advantages for fault-tolerant quantum computing. Its lightweight RNN enables real-time inference with minimal overhead, allowing for its deployment on resource-constrained hardware. The

UF backbone ensures scalability and parallelism, while the RNN refines uncertain regions without reducing throughput.

Nevertheless, our approach has limitations. Reliance on pre-trained RNNs entails a trade-off between generalization and accuracy, as parameters are tuned to specific noise models. Significant deviations in noise conditions may require retraining or fine-tuning, reducing adaptability in dynamic environments. Although latency is modest compared to more complex neural decoders, it may still be non-negligible for ultra-low-latency or high-throughput scenarios. These factors motivate further optimization of both the neural module through lightweight architectures, pruning, and quantization, and use of the classical UF component to minimize delay without sacrificing accuracy.

Future research will focus on broadening the applicability of the NGUF framework to encompass more general error models, including correlated, biased, and time-varying noise, as well as extending its support to diverse code families beyond two-dimensional color codes, such as higher-dimensional topological codes and subsystem codes. Another important direction is the exploration of hardware-efficient RNN deployment using techniques such as model pruning, quantization, and specialized accelerators, enabling low-latency inference on embedded or cryogenic control hardware. In parallel, the integration of adaptive learning strategies capable of online parameter adjustment in response to evolving noise characteristics will be crucial for maintaining robustness in dynamic environments. Collectively, these developments could pave the way for next-generation hybrid neural–classical decoders that combine scalability, resource efficiency, and adaptability, accelerating the practical realization of fault-tolerant quantum computing.

Author Contributions: Writing—original draft, M.F.; data curation, M.F.; formal analysis, C.T. and Z.F.; Simulation Computation, M.F.; methodology, M.F. and H.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Natural Science Foundation of Shandong Province (NSFSD, Jinan, China; Grant No. ZR2021MF049), the Joint Fund of the Natural Science Foundation of Shandong Province (NSFSD, Jinan, China; Grant Nos. ZR2022LLZ012 and ZR2021LLZ001), the Natural Science Foundation of Shandong Province (NSFSD, Jinan, China; Grant No. ZR2020QA078), and the Key R&D Program of Shandong Province (Jinan, China; Grant No. 2023CXGC010901).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Devitt, S.J.; Munro, W.J.; Nemoto, K. Quantum error correction for beginners. *Rep. Prog. Phys.* **2013**, *76*, 076001. [CrossRef] [PubMed]
2. Roffe, J. Quantum error correction: An introductory guide. *Contemp. Phys.* **2019**, *60*, 226–245. [CrossRef]
3. Terhal, B.M. Quantum error correction for quantum memories. *Rev. Mod. Phys.* **2015**, *87*, 307. [CrossRef]
4. Krinner, S.; Lacroix, N.; Remm, A.; Di Paolo, A.; Genois, E. Realization of quantum error correction. *Nature* **2004**, *432*, 602–605. [CrossRef]
5. Cory, D.G.; Price, M.D.; Maas, W.; Knill, E.; Laflamme, R. Experimental Quantum Error Correction. *Phys. Rev. Lett.* **1998**, *81*, 2152. [CrossRef]
6. Knill, E.; Laflamme, R. Theory of quantum error-correcting codes. *Phys. Rev. A* **1997**, *55*, 900. [CrossRef]
7. Aoki, T.; Takahashi, G.; Kajiya, T.; Yoshikawa, J.I.; Braunstein, S.L.; Van Loock, P.; Furusawa, A. Quantum error correction beyond qubits. *Nat. Phys.* **2009**, *5*, 541–546. [CrossRef]
8. Dür, W.; Skotiniotis, M.; Froewis, F.; Kraus, B. Improved Quantum Metrology Using Quantum Error Correction. *Phys. Rev. Lett.* **2014**, *112*, 080801. [CrossRef]

9. Kubica, A.; Yoshida, B.; Pastawski, F. Unfolding the color code. *New J. Phys.* **2015**, *17*, 083026. [CrossRef]
10. Anwar, H.; Brown, B.J.; Campbell, E.T.; Browne, D.E. Fast decoders for qudit topological codes. *New J. Phys.* **2014**, *16*, 063038.
11. Brown, B.J.; Nickerson, N.H.; Browne, D.E. Fault-tolerant error correction with the gauge color code. *Nat. Commun.* **2016**, *7*, 12302. [CrossRef]
12. Wang, Y.; Simsek, S.; Gatterman, T.M.; Gerber, J.A.; Gilmore, K.; Gresh, D.; Criger, B. Fault-tolerant one-bit addition with the smallest interesting color code. *Sci. Adv.* **2024**, *10*, 9024. [CrossRef]
13. Kesselring, M.S.; Pastawski, F.; Eisert, J.; Brown, B.J. The boundaries and twist defects of the color code and their applications to topological quantum computation. *Quantum* **2018**, *2*, 101. [CrossRef]
14. Yoshida, B. Topological color code and symmetry-protected topological phases. *Phys. Rev. B* **2015**, *91*, 245131. [CrossRef]
15. Reichardt, B.W. Fault-tolerant quantum error correction for Steane's seven-qubit color code with few or no extra qubits. *Quantum Sci. Technol.* **2020**, *6*, 015007. [CrossRef]
16. San Miguel, J.F.; Williamson, D.J.; Brown, B.J. A cellular automaton decoder for a noise-bias tailored color code. *Quantum* **2023**, *7*, 940. [CrossRef]
17. Baireuther, P.; Caio, M.D.; Criger, B.; Beenakker, C.W.; O'Brien, T.E. Neural network decoder for topological color codes with circuit level noise. *New J. Phys.* **2019**, *21*, 013003. [CrossRef]
18. Patwary, M.M.A.; Blair, J.; Manne, F. Experiments on Union-Find Algorithms for the Disjoint-Set Data Structure. In *International Symposium on Experimental Algorithms*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 411–423.
19. Manne, F.; Patwary, M.M.A. A Scalable Parallel Union-Find Algorithm for Distributed Memory Computers. In *Parallel Processing and Applied Mathematics*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6067.
20. Cybenko, G.; Allen, T.G.; Polito, J.E. Practical parallel Union-Find algorithms for transitive closure and clustering. *Int. J. Parallel Program.* **1988**, *17*, 403–423. [CrossRef]
21. Wilkinson, M.H.; Roerdink, J.B. Fast Morphological Attribute Operations Using Tarjan's Union-Find Algorithm. In *Mathematical Morphology and Its Applications to Image and Signal Processing*; Springer: Boston, MA, USA, 2000; pp. 311–320.
22. Griffiths, S.J.; Browne, D.E. Union-find quantum decoding without union-find. *Phys. Rev. Res.* **2024**, *6*, 013154. [CrossRef]
23. Charguéraud, A.; Pottier, F. Verifying the Correctness and Amortized Complexity of a Union-Find Implementation in Separation Logic with Time Credits. *J. Autom. Reason.* **2019**, *62*, 331–365. [CrossRef]
24. Mehlhorn, K.; Näher, S.; Alt, H. A Lower Bound on the Complexity of the Union-Split-Find Problem. *SIAM J. Comput.* **1988**, *17*, 1093–1102. [CrossRef]
25. Vittal, S.; Das, P.; Qureshi, M. Astrea: Accurate quantum error-decoding via practical minimum-weight perfect-matching. In Proceedings of the 50th Annual International Symposium on Computer Architecture, Orlando, FL, USA, 17–21 June 2023; Volume 2, pp. 1–16.
26. Asathulla, M.K.; Khanna, S.; Lahn, N.; Raghvendra, S. A faster algorithm for minimum-cost bipartite perfect matching in planar graphs. *ACM Trans. Algorithms (TALG)* **2019**, *16*, 1–30.
27. Delfosse, N.; Londe, V.; Beverland, M.E. Toward a Union-Find Decoder for Quantum LDPC Codes. *IEEE Trans. Inf. Theory* **2022**, *68*, 3187–3199. [CrossRef]
28. Cook, W.; Rohe, A. Computing minimum-weight perfect matchings. *INFORMS J. Comput.* **1999**, *11*, 138–148. [CrossRef]
29. Wu, Y.; Liyanage, N.; Zhong, L. An interpretation of Union-Find Decoder on Weighted Graphs. *arXiv* **1999**, arXiv:2211.03288.
30. Liyanage, N.; Wu, Y.; Tagare, S.; Zhong, L. FPGA-Based Distributed Union-Find Decoder for Surface Codes. *IEEE Trans. Quantum Eng.* **2024** , *5*, 3103318. [CrossRef]
31. Chan, T.; Benjamin, S.C. Actis: A strictly local union–find decoder. *Quantum* **2023**, *7*, 1183. [CrossRef]
32. Wang, H.W.; Xue, Y.J.; Ma, Y.L.; Hua, N.; Ma, H.Y. Determination of quantum toric error correction code threshold using convolutional neural network decoders. *Chin. Phys. B* **2022**, *31*, 010303. [CrossRef]
33. Löbl, M.C.; Chen, S.X.; Paesani, S.; Sørensen, A.S. Breadth-first graph traversal union-find decoder. *arXiv* **2024**, arXiv:2407.15988.
34. Tian, Y.; Zheng, Y.; Wang, X. High-performance and efficient decoding of surface codes: An iterative reweighted union-find approach. In Proceedings of the Ninth International Symposium on Advances in Electrical, Electronics, and Computer Engineering, Changchun, China, 1–3 March 2024; Vulume 13291, pp. 1298–1303.
35. Das, P.; Pattison, C.A.; Manne, S.; Carmean, D.; Svore, K.; Qureshi, M.; Delfosse, N. A Scalable Decoder Micro-architecture for Fault-Tolerant Quantum Computing. *arXiv* **2020**, arXiv:2001.06598.
36. Wolanski, S.; Barber, B. Ambiguity Clustering: An accurate and efficient decoder for qLDPC codes. *arXiv* **2024**, arXiv:2406.14527. [CrossRef]
37. Connolly, N.; Londe, V.; Leverrier, A.; Delfosse, N. Fast erasure decoder for hypergraph product codes. *Quantum* **2024**, *8*, 1450. [CrossRef]
38. Egorov, E.; Bondesan, R.; Welling, M. The END: An Equivariant Neural Decoder for Quantum Error Correction. *arXiv* **2023**, arXiv:2304.07362. [CrossRef]

39. Sherstinsky, A. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Phys. D Nonlinear Phenom.* **2020**, *404*, 132306. [CrossRef]
40. Yin, W.; Kann, K.; Yu, M.; Schütze, H. Comparative Study of CNN and RNN for Natural Language Processing. *arXiv* **2017**, arXiv:1702.01923. [CrossRef]
41. Dhruv, P.; Naskar, S. Image Classification Using Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN): A Review. In *Machine Learning and Information Processing*; Springer: Singapore, 2024; pp. 367–381.
42. Greff, K.; Srivastava, R.K.; Koutník, J.; Steunebrink, B.R.; Schmidhuber, J. LSTM: A Search Space Odyssey. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *28*, 2222–2232. [CrossRef] [PubMed]
43. Li, F.; Li, A.Q.; Gan, Q.D.; Ma, H.Y. Recurrent neural network decoding of rotated surface codes based on distributed strategy. *Chin. Phys. B* **2024**, *33*, 040307. [CrossRef]
44. Siami-Namini, S.; Tavakoli, N.; Namin, A.S. The Performance of LSTM and BiLSTM in Forecasting Time Series. In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 9–12 December 2019; pp. 3285–3292.
45. Gers, F.A.; Schmidhuber, J.; Cummins, F. Learning to Forget: Continual Prediction with LSTM. *Neural Comput.* **2000**, *12*, 2451–2471. [CrossRef]
46. Delfosse, N. Decoding color codes by projection onto surface codes. *Phys. Rev. A* **2014**, *89*, 012317. [CrossRef]
47. Delfosse, N.; Nickerson, N.H. Almost-linear time decoding algorithm for topological codes. *Quantum* **2021**, *5*, 595. [CrossRef]
48. Google Quantum AI and Collaborators. Quantum error correction below the surface code threshold. *Nature* **2025**, *638*, 920–926. [CrossRef]