Neural network decoder for near-term surface-code experiments

Boris M. Varbanov [1,*] Marc Serra-Peralta [1,2] David Byfield [3] and Barbara M. Terhal [1,2]
[1]QuTech, Delft University of Technology, P.O. Box 5046, 2600 GA Delft, The Netherlands
[2]Delft Institute of Applied Mathematics, Technische Universiteit Delft, 2628 CD Delft, The Netherlands
[3]Riverlane, Cambridge CB2 3BZ, United Kingdom

(Received 23 October 2023; revised 19 February 2024; accepted 18 October 2024; published 8 January 2025)

Neural network decoders can achieve a lower logical error rate compared to conventional decoders, like minimum-weight perfect matching, when decoding the surface code. Furthermore, these decoders require no prior information about the physical error rates, making them highly adaptable. In this study, we investigate the performance of such a decoder using both simulated and experimental data obtained from a transmon-qubit processor, focusing on small-distance surface codes. We first show that the neural network typically outperforms the matching decoder due to better handling of errors leading to multiple correlated syndrome defects, such as Y errors. When applied to the experimental data of Google Quantum AI [R. Acharya *et al.*, Nature (London) **614**, 676 (2023)], the neural network decoder achieves logical error rates approximately 25% lower than minimum-weight perfect matching, approaching the performance of a maximum-likelihood decoder. To demonstrate the flexibility of this decoder, we incorporate the soft information available in the analog readout of transmon qubits and evaluate the performance of this decoder in simulation using a symmetric Gaussian-noise model. Considering the soft information leads to an approximately 10% lower logical error rate, depending on the probability of a measurement error. The good logical performance, flexibility, and computational efficiency make neural network decoders well-suited for near-term demonstrations of quantum memories.

DOI: 10.1103/PhysRevResearch.7.013029

I. INTRODUCTION

Quantum computers are anticipated to outperform classical computers in solving specific problems, such as integer factorization [1] and quantum simulation [2]. However, for a quantum computer to perform any meaningful computation, it has to be able to execute millions of operations, requiring error rates per operation lower than 10^{-10} [3,4]. Despite a valiant experimental effort aimed at enhancing operational performance, state-of-the-art processors typically exhibit error rates per operation around 10^{-3} [5–14], which is far from what is needed to perform any useful computation.

Fortunately, quantum error correction (QEC) provides a means to reduce the error rates, albeit at the cost of additional overhead in the required physical qubits [15–18]. Two-dimensional stabilizer codes [19], such as the surface codes [20], have emerged as a prominent approach to realizing fault-tolerant computation due to their modest connectivity requirements and high tolerance to errors [21–23]. These codes encode the logical information into an array of physical qubits, referred to as data qubits. Ancilla qubits are used to repeatedly measure parities of sets of neighboring data qubits. Changes between consecutive measurement outcomes, which are typically referred to as syndrome defects, indicate that errors have occurred. A classical decoder processes this information and aims at inferring the most likely correction.

The increased number of available qubits [24–27] and the higher fidelities of physical operations [5–14,28–33] in modern processors have enabled several experiments employing small-distance codes to demonstrate the capacity to detect and correct errors [26,27,34–46]. In a recent milestone experiment, the error rate per QEC round of a surface-code logical qubit was reduced by increasing the code distance [26], demonstrating the fundamental suppression achieved by QEC.

The performance of the decoder directly influences the performance of a QEC code. Minimum-weight perfect matching (MWPM) is a good decoding algorithm for the surface code, which is computationally efficient and, therefore, scalable [21,47–50]. Its good performance is ensured under the assumption that the errors occurring in the experiment can be modeled as independent X and Z errors [21]. This leads to the MWPM decoder performing worse than decoders based on belief propagation [51–54] or a (more computationally expensive) approximate maximum-likelihood decoder based on tensor-network (TN) contraction [55,56]. A more practical concern is that a decoder relies on a physical error model to accurately infer the most likely correction. Typically, this requires constructing an approximate model and a series of benchmarking experiments to extract the physical error rates. While there are methods to estimate the physical error rates based on the measured defects [26,39,57,58], they typically ignore nonconventional errors such as crosstalk or leakage. The presence of these errors can impact both the accuracy with which the physical error rates are estimated from the data and the performance of the decoder itself [58].

*Contact author: Boris.Mihailov.Varbanov@USherbrooke.ca

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

An alternative approach to decoding is based on using neural networks (NNs) to infer the most likely correction given a set of measured defects [59–79]. These decoders do not require any prior information about the error model and therefore alleviate the need to construct any error model, making them highly adaptable. This flexibility comes at the cost of requiring a significant amount of data for training the network and optimizing the hyper-parameters to ensure that the optimal performance of the decoder is reached during training. Despite the potential issues during the training, it has been shown that they can match and generally exceed the performance of MWPM decoders, in several cases achieving near-optimal performance [62,64]. Depending on the NN architecture employed, these decoders can be scalable and run in real time [66,75–78]. While decoders based on recurrent NNs are more computationally expensive, they enable the decoding of experiments performing a variable number of stabilizer measurement rounds [62,64,69], making them well-suited for decoding near-term memory [62] and stability experiments [80].

Most of the NN decoders proposed in the literature so far have only been benchmarked on simulated data, leaving open the question of what their performance will be when applied to experimental data. Reference [81] explored the performance of a graph neural network decoder on data from the repetition code experiment done in [26]. Reference [82] developed a transformer-based recurrent NN decoder and applied it to the surface code experiments that were also done in [26], achieving a lower logical error rate than the TN decoder and demonstrating that the performance of such a decoder can be further improved by considering the information about leakage outside of the computational subspace of transmon qubits and the continuous information available in the measurement outcomes of these qubits [83,84].

In this work, we assess the performance of a recurrent neural network decoder using both simulated and experimental data. Our work goes beyond [62] and previous NN decoding works in applying and partially training a NN decoder for the first time on data from a surface-code experiment [26], thus capturing realistic performance and showing the versatility of NN decoders. In addition, we go beyond [62] in training the NN decoder for a distance-7 surface code and extracting the exponential error suppression factor Λ [39], defined in Eq. (1), on simulated data using a circuit-level noise model. Thirdly, we show that our NN decoder can be trained with (simulated) soft measurement data and get a performance enhancement.

We begin by simulating the performance of a $d = 3$ surface code using a circuit-level noise model to show that the NN decoder outperforms MWPM by learning to deal with Y errors, as previous studies have suggested [62].

Next, we investigate the performance of the NN decoder when applied to data from a recent surface code experiment [26]. Due to the limited volume of available experimental data (see Sec. III B), we train the NN decoder on simulated data generated using an error model based on the measured physical error rates. However, we evaluate the decoder's performance on simulated and experimental data. The NN decoder significantly outperforms MWPM when decoding simulated data, and furthermore achieves a lower logical error rate for the $d = 5$ code than the constituent $d = 3$ codes. When evaluated on experimental data, the NN decoder achieves a performance approaching that of a tensor-network decoder, which approximates a maximum-likelihood decoder. However, contrary to the finding in [26], the logical error rate observed in the $d = 5$ experiment is higher than the average of each of the $d = 3$ experiments, which we attribute to either a suboptimal choice of hyper-parameters or the mismatch between the simulated data that the decoder was trained on and the experimental data.

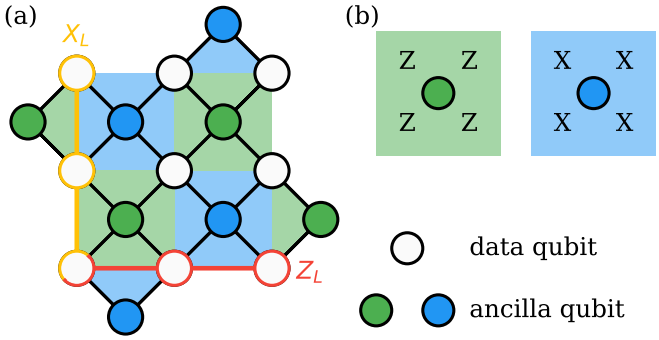


FIG. 1. (a) Schematic of a distance $d = 3$ surface-code logical qubit, where nine data qubits (white circles) store the logical information. Eight ancilla qubits (blue and green circles) are used to measure the Z-type (green plaquettes) and X-type (blue plaquettes) stabilizers of the code. Examples are shown of the X_L (yellow) and Z_L (red) logical operators of the code. (b) Illustration of the Z-type plaquette (left, green) and X-type (right, blue) plaquette corresponding to the $ZZZZ$ and $XXXX$ stabilizer operators measured by each ancilla qubit.

To further explore the performance of NNs, we consider the continuous information available in the measurement outcomes of transmon qubits [83,84], typically referred to as soft information [85]. By calculating the defect probabilities given the soft outcomes and providing them to the neural network during training and evaluation, we demonstrate that the soft NN decoder can achieve an approximately 10% lower logical error rate compared to the hard NN decoder that does not consider this soft information if the measurement error probability is sufficiently high.

II. BACKGROUND

A. The surface code

A (rotated) surface code encodes a single logical qubit into a two-dimensional array of $n = d \times d$ physical qubits, referred to as data qubits, where d is the distance of the code. The logical state of the qubit is determined by the stabilizers of the code, which are the weight-4 or weight-2 X-type (blue plaquettes) or Z-type (green plaquettes) Pauli operators; see Fig. 1. In addition to the stabilizers, the code is given by a pair of anticommuting logical operators, X_L and Z_L, which commute with the code stabilizers. The stabilizers are typically measured indirectly with the help of $n - 1$ ancilla qubits. To perform this measurement, each ancilla coherently interacts with its neighboring data qubits in a specific order [86], after which the ancilla qubit is measured and reset. The stabilizer measurement outcomes are typically referred to as the syndromes and hold information about the errors that have occurred. The full circuits used to perform these

measurements are shown in Fig. 8. In particular, we use the circuits used in [26], which feature several echo gates used for dynamical decoupling in the experiment, used to mitigate the dephasing experienced by the qubits due to low-frequency flux noise; see Appendix A 1 for additional details.

To characterize the performance of the code, we perform a series of logical memory experiments. In each experiment, the physical qubits are prepared in an eigenstate of the X_L (Z_L) logical operator, after which $N-1$ rounds of stabilizer measurements are executed. The experiment is concluded by reading out each data qubit in the X (Z) basis, which also performs a logical X_L (Z_L) measurement. The goal of each experiment is to maintain the logical state for as many QEC rounds as possible by using error correction; see Appendix A 1 for more details. We refer to each individual such experiment as a *shot*.

The information about errors is contained in the stabilizer measurement outcome $m_{r,a}$ of ancilla a at round r. The data-qubit measurement outcomes obtained at the end of each experiment can be used to infer a final set of outcomes $m_{r=N,a}$ for either the X-type or Z-type stabilizers. The defects $d_{r,a} = m_{r,a} \oplus m_{r-1,a}$ isolate the changes in $m_{r,a}$ such that an error is signaled by an observation of one or more $d_{r,a} = 1$. The choice of initial state and the dynamical decoupling gates can also flip some of the measured $m_{r,a}$, which is accounted for when calculating $d_{r,a}$. A decoder processes the observed $d_{r,a}$ to infer a correction for the measured logical observable. By repeating each experiment many times, we extract the probability of a logical error $p_L(r)$ at QEC round r, from which we calculate the logical fidelity $F_L(r) = 1 - 2p_L(r)$, which decays exponentially with the number of executed QEC rounds. We model this decay as $F_L(r) = (1-2\varepsilon_L)^{r-r_0}$, where ε_L is the logical error rate per QEC round and r_0 is a fitting constant. When fitting the decay of $F_L(r)$ to extract ε_L, we start the fit at $r = 3$ to avoid any time-boundary effects that might impact this estimate.

B. Error models

To explore the performance of the NN decoder, we perform simulations using circuit-level Pauli-noise models. For most of our simulations, we consider a depolarizing circuit-level noise, which is defined as follows:

(i) After each single-qubit gate or idling period, with a probability $p/3$, we apply an error drawn from $\{X, Y, Z\}$.

(ii) After each two-qubit gate, with a probability $p/15$, we apply an error drawn from $\{I, X, Y, Z\}^{\otimes 2} \setminus \{II\}$.

(iii) With a probability p, we apply an X error before each measurement.

(iv) With a probability p, we apply an X error after each reset operation or after the qubits are first prepared at the start of an experiment.

In some of our simulations, we consider noise models that are biased to have a higher or a lower probability of applying Y errors. To construct this model, we define a Y-bias factor η and modify the standard depolarizing circuit-level noise model, as follows:

(i) After each single-qubit gate or idling period, there is a probability $\eta p/(\eta+2)$ to apply a Y error and a probability $p/(\eta+2)$ to apply an X or a Z error.

(ii) After each two-qubit gate, there is a probability $\eta p/(7\eta+8)$ of applying an error drawn from $\mathcal{P}_B = \{IY, XY, YI, YX, YY, YZ, ZY\}$ and a probability $p/(7\eta+8)$ of applying an error drawn from $\{I, X, Y, Z\}^{\otimes 2} \setminus (\mathcal{P}_B \cup \{II\})$.

This biased error model is a generalization of the depolarizing model. In particular, choosing $\eta = 1$ makes this noise model equivalent to the depolarizing one. On the other hand, when $\eta = 0$, the model leads to only X or Z errors applied after operations. In the other limiting case, as $\eta \to \infty$, the model applies only Y errors after idling periods and gates. Given that the error probability is the same across all operations of the same type, we will refer to these error models as uniform circuit-level noise models.

Finally, we also perform simulations of the recent experiment conducted by Google Quantum AI, using the error model which they provided together with the experimental data [26]. This is once again a circuit-level Pauli-noise model similar to the ones presented above, but the probability of a depolarizing error after each operation is based on the measured physical error rates. We will refer to this model as the experimental circuit-level noise model.

We use *stim* [87] to perform the stabilizer simulations. We have written a wrapper package that helps with constructing the circuit for each experiment, which is available in [88]. We use *pymatching* [49] for the MWPM decoding. The weights used in the MWPM decoder are directly extracted from the sampled circuit using the built-in integration between *stim* and *pymatching*.

C. Neural network architecture

Here, we describe the NN architecture that we employ in this work, which follows nearly exactly the one proposed in [62,64]. Many NN decoders studied previously are based on feed-forward or convolutional NN architecture. These decoders can generally decode experiments running a fixed number of QEC rounds. Decoders based on recurrent NN architectures, on the other hand, can learn the temporal correlations between the data (in our case, the correlations between the defects in different QEC rounds generally resulting from ancilla-qubit or measurement errors), allowing them to directly process experiments performing a variable number of QEC rounds. We have used the *TensorFlow* library [89] to implement the NN architecture, with the source code of the decoder available in [90], the parameters used for each training are listed in Table I, while the scripts that perform the training are available upon request.

The NN architecture takes as input the defects $d_{a,r}$ with $r = 1, 2, \ldots, N$. The decoder solves a binary classification problem and determines whether a correction of the logical observable is required based on the observed defects. In practice, the architecture is based on a two-headed network that makes two predictions p_{main} and p_{aux}, which are used to improve the training of the network; see Fig. 2. To train a decoder, a series of memory experiments are performed. Since the logical qubit is prepared in a known logical state and measured at the end of each experiment, it is possible to extract the actual value $p_{\text{true}} \in \{0, 1\}$ of whether a correction is required or not. In particular, the cost function I that the

TABLE I. The hyperparameters used for training the NN decoders. Different parameters are used for simulations based on the uniform circuit-level noise model and the experimental circuit-level noise, which models the experiments done in [26]. The internal state size of the network layers N_L is chosen to scale with the code distance d. The QEC round parameters $[i, j, k]$ for each data set refer to performing experiments starting with i QEC rounds and going up to j rounds in steps of k. The total number of shots used for training is given, which is equally divided over the QEC rounds and prepared states (not shown in the table). The learning rate, batch size, and dropout rate are the hyperparameters we tune to help the network to train.

Distance	Shots	Rounds	Dim. N_L	Learning rate	Batch size	Dropout rate
			Experimental circuit-level noise			
3	2×10^7	[1, 25, 2]	64	5×10^{-4}	64	5%
5	6×10^7	[1, 25, 2]	253	5×10^{-4}	256	5%
			Uniform circuit-level noise			
3	10^7	[1, 37, 4]	64	10^{-3}	256	20%
5	10^7	[1, 37, 4]	96	10^{-3}	256	20%
7	10^7	[1, 37, 4]	128	10^{-3}	256	20%

network attempts to minimize during training is the weighted sum of the binary cross-entropies between each prediction and p_{true}, expressed as

$$I = H(p_{\text{main}}, p_{\text{true}}) + w_a H(p_{\text{aux}}, p_{\text{true}}),$$

where w_a is a weight that is typically chosen as $w_a = 0.5$ in our runs, while

$$H(p_i, p_j) = -p_i \log p_j - (1 - p_i) \log(1 - p_j)$$

is the binary cross-entropy function. The choice behind this loss function is elaborated below.

Figure 2 schematically illustrates the architecture of the recurrent network. The recurrent body of the neural network consists of two stacked long short-term memory (LSTM) layers [91,92]. Each LSTM layer is defined by a pair of internal memory states: a short-term memory, referred to as the hidden state, and a long-term memory, referred to as the cell state. Here, we use the same internal states size N_L for both LSTM layers, with $N_L = 64, 96, 128$ for surface codes of distance $d = 3, 5, 7$, unless otherwise specified. The LSTM layers receive the defects for each QEC round as input, calculated

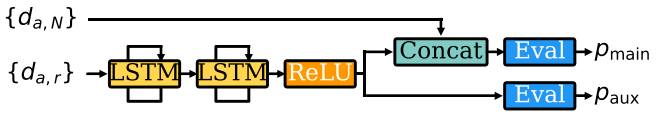


FIG. 2. Schematic of the recurrent NN architecture used in this work, following the design proposed in [64]. The inputs to the network are the set of defects $\{d_{a,r}\}$, which are calculated from the measurement outcomes of each ancilla qubit a at QEC round $r = 1, 2, \ldots, N-1$, and the final defects $\{d_{a,N}\}$, which are inferred from data qubit measurements. The time-invariant input $\{d_{a,r}\}$ is provided to the recurrent part of the network, consisting of two stacked LSTM layers (yellow rectangles) and a ReLU activation layer (orange rectangle). The recurrent output is then passed to the two heads of the decoder, which consist of an evaluation layer (blue rectangle) that predicts a probability of a logical error. The lower head takes as input only the recurrent output and outputs a probability p_{aux}. The upper head, on the other hand, combines (teal rectangle) the recurrent output with $\{d_{a,N}\}$ and outputs a probability p_{main}. Arrows indicate the flow of information through the network.

from both the X-type and the Z-type stabilizer measurement outcomes. The first LSTM layer outputs a hidden state for each QEC round, which is then provided as input to the second LSTM layer, which outputs only its final hidden state. A rectified linear unit (ReLU) activation function [89] is applied to the output of the second LSTM layer before being passed along to each of the two heads of the network.

The heads of the network are feed-forward evaluation networks [89] consisting of a single hidden layer of size N_L using the ReLU activation function and an output layer using the sigmoid activation function, which maps the hidden layer output to a probability used for binary classification. The output of the recurrent part of the network is directly passed to the lower head of the network, which uses this information to predict a probability p_{aux} of a logical error. The upper head also considers the defects inferred from the data qubit measurements, which are combined with the recurrent output and provided as input. Therefore, unlike the lower head, the upper one uses the full information about the errors that have occurred when making its prediction p_{main} of whether a logical error occurred. Both p_{main} and p_{aux} are used when training the network, which promotes the neural network to place greater importance on the defects obtained from the ancilla-qubit measurements and helps it to more easily generalize to handle longer input sequences. However, only p_{main} is used when evaluating the performance of the decoder. We provide additional details about the training procedure in Appendix A 2 and list the hyperparameters of the network in Table I.

III. RESULTS

A. Performance on circuit-level noise simulations

We first demonstrate that the NN decoder can achieve a lower logical error rate than the MWPM decoder by learning error correlations between the defects, which are otherwise ignored by the MWPM decoder. We consider the Y-biased circuit-level noise model described previously, parametrized by the bias η towards Y errors and a probability $p = 0.001$ of inserting an error after each operation. We use this noise model to simulate the performance of a

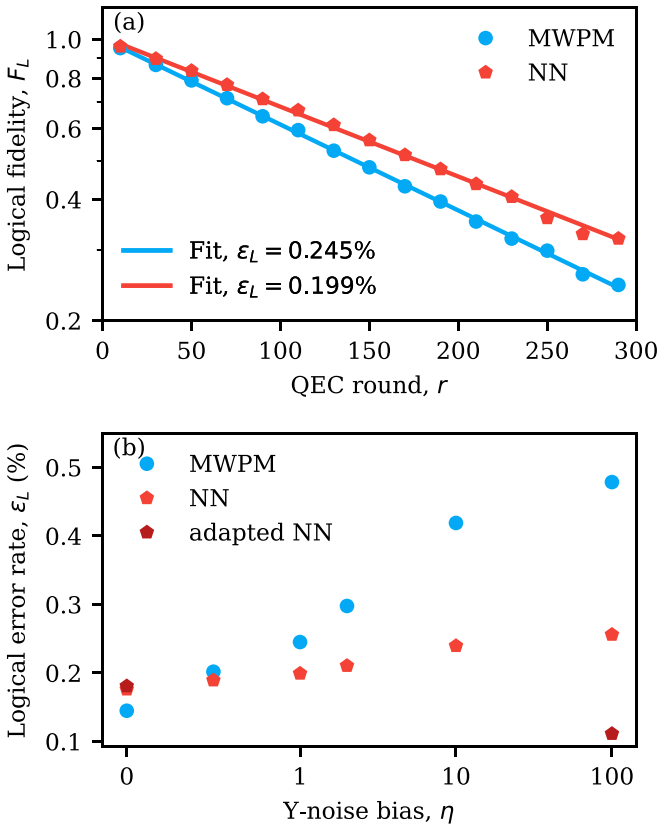


FIG. 3. (a) Logical fidelity F_L as a function of the number of QEC rounds r for the MWPM (blue) and the NN decoders (red) using a uniform circuit-level depolarizing noise model. Each data point is averaged over 4×10^4 shots. Solid lines show the fits to the data used to extract the logical error rate per round ε_L. (b) The logical error rate ε_L as a function of the bias η towards Y errors for the MWPM decoder (blue) and a NN decoder trained on simulated data using depolarizing noise (red), corresponding to $\eta = 1$. The performance of an adapted NN decoder at a bias of $\eta = 0$ or $\eta = 100$ is shown in dark red. Each point is extracted from a fit of the decay of the logical fidelity over 300 QEC rounds. The error bars are smaller than the marker sizes.

$d = 3$ surface-code quantum memory experiment in the Z-basis, initially preparing either $|0\rangle^{\otimes n}$ or $|1\rangle^{\otimes n}$. To train the NN decoder, we generated data sets of $r = 1, 5, \ldots, 37$ QEC rounds, sampling 5×10^5 shots for each round and initial state. When evaluating the decoder's performance, we simulate the code performance over $r = 10, 30, \ldots, 290$ QEC rounds and sample 2×10^4 shots instead.

To benchmark the logical performance, we calculate the logical fidelity F_L at the end of each experiment. Averaging F_L over each initial state, we fit the exponential decay of F_L with the number of QEC rounds to extract the logical error rate per round ε_L. Figure 3 shows that the NN decoder maintains a constant ε_L when evaluated on data sets going up to 300 QEC rounds, demonstrating the ability of the decoder to generalize to significantly longer sequences than those used for training. On the other hand, the NN decoder achieves about 20% lower ε_L compared to the MWPM decoder. We then evaluate the trained NN decoder on simulated data using $\eta \in \{0, 0.5, 1, 2, 10, 100\}$ and keep all other parameters the same without training any new neural networks, with the resulting error rates shown in Fig. 3(b). At $\eta = 0$, corresponding to an error model leading to X and Z errors, the NN decoder displays a higher ε_L than the MWPM decoder. For $\eta \geqslant 0.5$, the NN decoder instead demonstrates a lower logical error, with the relative reduction increasing with the bias. This demonstrates that the NN decoder can achieve a lower logical error rate by learning the correlations between the defects caused by Y errors, consistent with the results presented in [62]. The NN decoder can achieve an even lower logical error rate at a bias of $\eta = 100$ by being trained on a data set generated using this bias (referred to as the adapted NN decoder in Fig. 3). On the other hand, training a model for $\eta = 0$ does not lead to any improvement in ε_L of the NN decoder, showing that the MWPM decoder is more optimal in this setting.

B. Performance on experimental data

Next, we evaluate the performance of the NN decoder on experimental data available from the recent experiment executed by Google Quantum AI [26], where a 72-qubit quantum processor was used to implement a $d = 5$ surface code as well as the four $d = 3$ surface codes which use a subset of the qubits of the larger code. The stabilizer measurement circuits used in that experiment are the same as those shown in Fig. 8. For each distance-d surface code, the data qubits are prepared in several random bitstrings, followed by $r = 25$ rounds of stabilizer measurement, followed by a logical measurement, with experiments performed in both the X-basis and Z-basis. The experiment demonstrated that the $d = 5$ surface code achieves a lower ε_L compared to the average of the four constituent $d = 3$ patches when using a tensor-network (TN) decoder, an approximation to a maximum-likelihood decoder.

We find that training a NN decoder to achieve good logical performance requires a large number of shots (approximately 10^7 in total or more) obtained from experiments preparing different initial states and running a different number of rounds. As the amount of experimental data is too small to train the NN decoder (the total number of shots being 6.5×10^5), we instead opt to simulate the experiments using the Pauli error model based on the measured error rates of each operation, available in [26]. Keeping the same number of rounds and prepared state, we generate a total of 2×10^7 shots for training the decoder for each $d = 3$ experiment and 6×10^7 to train the decoder for the $d = 5$ experiment; see Table I. While we train the network on simulated data, we still evaluate the decoder performance on both simulated and experimental data, with the results shown in Figs. 4(a) and 4(b), respectively. Both the training and evaluation data consist of $r = 1, 3, \ldots, 25$ rounds of QEC and consider the same initial states. When evaluating the NN decoder on simulated data, we observe that the $d = 5$ code achieves a lower ε_L compared to the average of the $d = 3$ codes; see Fig. 4(a). Evaluating the decoder on the experimental data leads to an approximately 15% (40%) higher ε_L for the $d = 3$ ($d = 5$) code, demonstrating that the approximate error model used in simulation fails to fully capture the errors in the experiment. Furthermore, we observe that the $d = 5$ has a higher ε_L instead, see Fig. 4(a), contrary to what was demonstrated in [26] using a tensor-network decoder.

To put the performance of the NN decoder in perspective, in Fig. 5 we compare the logical performance of the NN

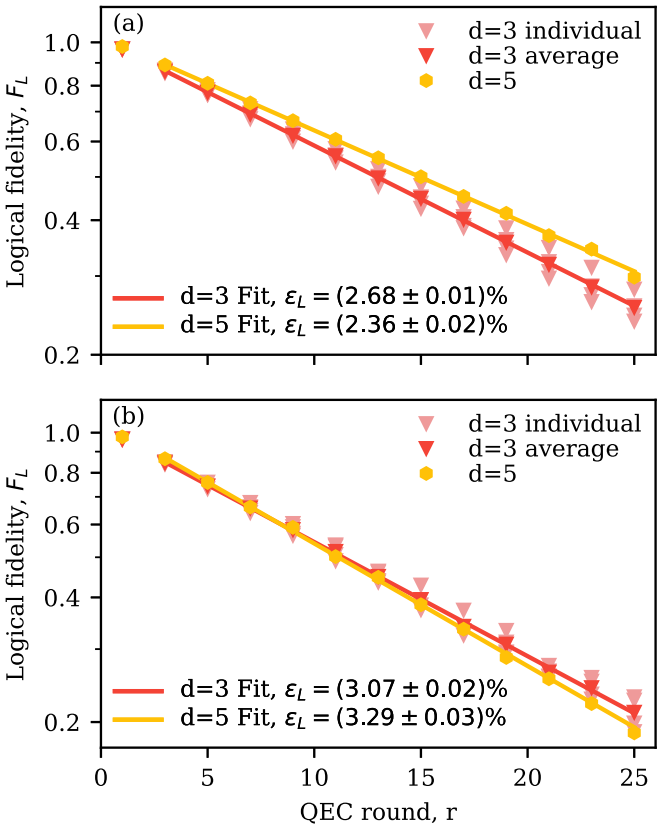


FIG. 4. Logical fidelity F_L as a function of the number of QEC rounds r for the NN decoder evaluated on simulated data [shown in (a)] and on experimental data [shown in (b)]. The average performance of the $d = 3$ surface code (red triangle), which is the average of the performance of each of the four constituent codes (bright red triangles), is compared to the $d = 5$ code (orange hexagons). Each data point is averaged over 5×10^4 shots for both experiment and simulation. Solid lines show the fits to the data used to extract the logical error rate per round ε_L. The error bars are smaller than the marker sizes.

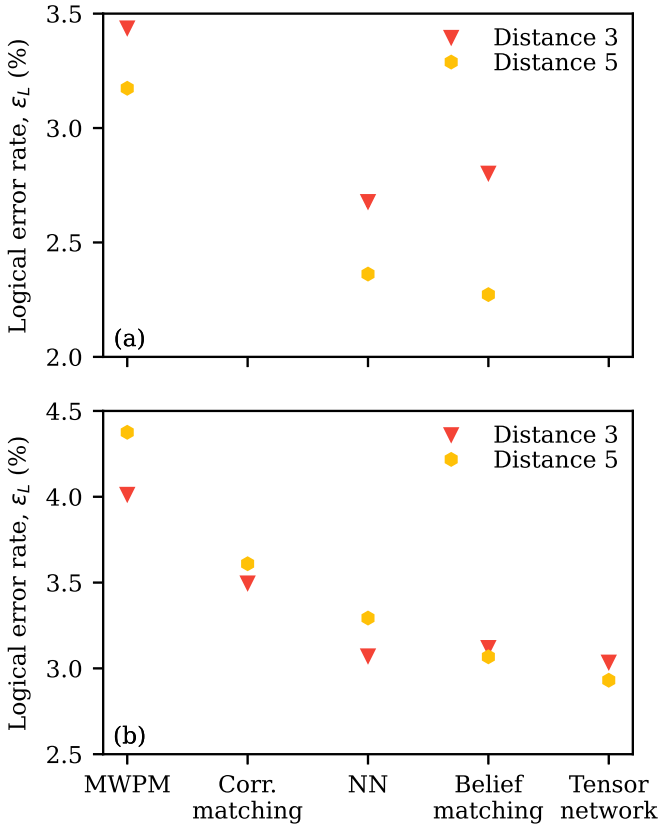


FIG. 5. The logical error rate per round ε_L for the $d = 3$ (red triangle) and $d = 5$ (orange hexagon) for several decoder implementations applied to either simulated data [shown in (a)] or experimental data [shown in (b)]. These correspond (from left to right) to minimum-weight perfect matching (MWPM), a correlated modification of MWPM (Corr. MWPM) [93], our neural network (NN) decoder, belief matching (BM) [53], and a tensor network (TN) decoder, which approximates maximum-likelihood decoding. We did not run the corr. MWPM or TN decoder on the simulated data so fewer data points appear in (a). All logical error rates on the experimental data, except for the NN decoder, are taken from [26]. The error bars are smaller than the marker sizes.

decoder to the performance of several other decoders that were also implemented in [26]. We perform this comparison both on simulated [see Fig. 5(a)] and experimental [see Fig. 5(b)] data. We find that the NN decoder consistently outperforms the standard MWPM decoder in either case. On the experimental data set, the NN decoder performs equivalent to the TN decoder when decoding the $d = 3$ surface codes. However, when decoding the $d = 5$ surface code experiment, the NN decoder displays a higher ε_L than the TN decoder and the computationally efficient belief-matching (BM) decoder [53]. When evaluated on simulated data, the NN and BM decoders exhibit similar error rates, with the NN decoder again demonstrating better performance when decoding the $d = 3$ code but worse when dealing with the $d = 5$ code. The BM decoder we use for the simulated data is described in [54] and uses the belief propagation implemented in [94]. The higher error rate of the NN decoder for the $d = 5$ code in both simulation and experiment can be related to the difficulty of optimizing the performance of the substantially larger NN model used (see Table I for the model hyperparameters). However, the discrepancy in the experiment can also be attributed to a mismatch between the simulated data used for training (based on an approximate error model) and the experimental data used for evaluation. Compared to the $d = 3$ surface code data, the accumulation of qubit leakage can cause the $d = 5$ performance to degrade faster over the QEC rounds [26]. We expect that training on experimental data and a better hyperparameter optimization will enable a NN performance comparable to state-of-the-art decoders like BM and TN while offering additional flexibility to the details of the noise model. Compared to the TN decoder, both NN and BM can achieve similar logical performance while remaining significantly faster, and if their implementation is optimized, they can potentially be used to decode experiments in real time.

C. Logical error rate suppression

An exponential suppression of the logical error rate, assuming that the physical error rates are below 'threshold',

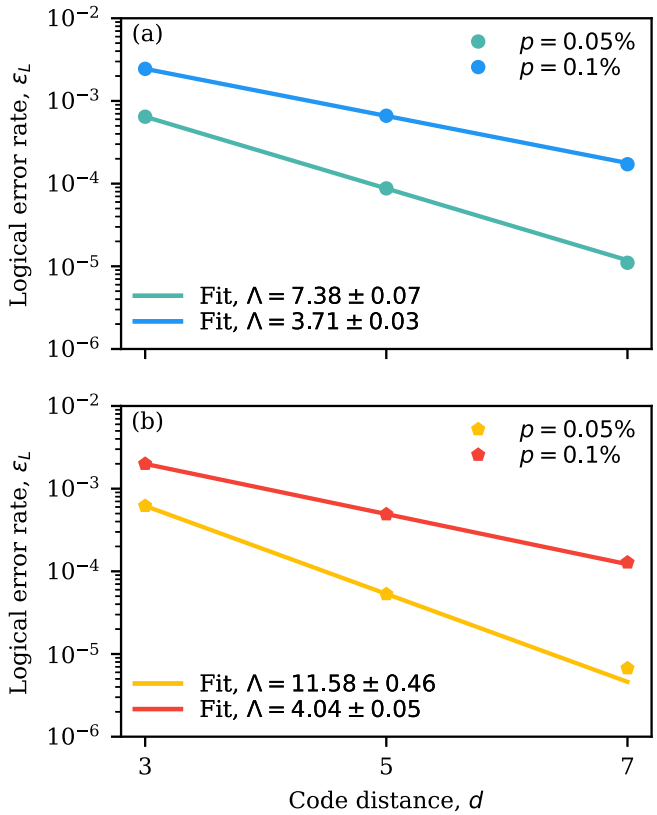


FIG. 6. The logical error rate per round ε_L for surface codes of distance $d = 3, 5, 7$ for an MWPM decoder, shown in (a), and our NN decoder, shown in (b). This is evaluated on data sets using a uniform depolarizing circuit-level noise model with error probabilities of $p = 0.1\%$ (blue for the MWPM, red for the NN decoder) and $p = 0.05\%$ (teal for the MWPM, orange for the NN decoder). Solid lines show the fits to the data used to extract the logical error suppression factor Λ. Each data point is extracted from a fit to the F_L as a function of QEC rounds. The logical fidelities are extracted over 10^5 shots. The error bars are smaller than the marker sizes.

is vital for realizing a fault-tolerant quantum computer. We explore the error suppression achieved when using the NN decoder. We characterize the logical performance of $d = 3, 5, 7$ surface codes simulated using a uniform depolarizing circuit-level noise model with an error probability of $p = 0.1\%$, close to the state-of-the-art physical error rates achieved in the experiment. To train the NN decoder, we use data generated using this error probability. We find that also training using a higher probability of $p = 0.2\%$ leads to a significantly lower logical error rate for the $d = 7$ code. Furthermore, we evaluate the performance of the NN decoder on data simulated using $p = 0.05\%$, which is an example of the physical error rate needed to achieve practical subthreshold scaling of the error rate. For each distance d and error probability p, we perform simulations of memory experiments in the Z-basis with varying numbers of QEC rounds, going up to 600 rounds for the $d = 7$ code with an error rate of $p = 0.05\%$ to extract the logical error per round ε_L. The logical error rates obtained when using an MWPM decoder are shown in Fig. 6(a), while those achieved by the NN decoder are shown in Fig. 6(b). If the physical error rate is below threshold, ε_L is expected to decay exponentially with the code distance d, following

$$\varepsilon_L(d) = C/\Lambda^{(d+1)/2}, \tag{1}$$

where Λ is the suppression factor and C is a fitting constant [39]. The data show an apparent exponential suppression of the error rates by either decoder for the considered error rates, which we fit to extract the suppression factor Λ, shown in Fig. 6. In either case, the NN decoder achieves better logical performance compared to the MWPM decoder. While for $p = 0.1\%$ the NN decoder achieves an approximately 10% higher Λ, for $p = 0.05\%$ the more accurate NN decoder leads to an approximately 60% higher suppression factor instead. The higher suppression factors Λ obtained from using better decoders significantly reduce the code distance required to achieve algorithmically relevant logical error rates. For example, for an error rate of $p = 0.05\%$, realizing $\varepsilon_L \approx 10^{-10}$ would require a $d = 19$ surface code when using the MWPM decoder and $d = 15$ when using the NN decoder, corresponding to roughly 40% less physical qubits required. However, whether the NN can continue to exhibit similar performance when decoding higher distance codes remains to be demonstrated.

D. Decoding with soft information

Measurements of physical qubits generally produce a continuous signal that is subsequently converted into declared binary outcomes by classical processing and thresholding. For example, transmon qubits are dispersively coupled to a dedicated readout resonator, which itself is connected to a readout feedline. Readout is performed by applying a microwave pulse to the feedline, populating the readout resonator. Due to a state-dependent shift of the resonator frequency, the outgoing signal is phase-shifted depending on whether the qubit is in the state $|0\rangle$ or $|1\rangle$. This leads to a change in the real and imaginary components of the outgoing signal, which is experimentally measured. This two-dimensional output can be transformed into a single continuous real variable and converted to a binary outcome by applying some threshold calibrated using a separate experiment [32,83,84].

While binary variables are convenient to work with and store, continuous measurement outcomes hold much more information about the state of the qubit, referred to as soft information. It has been demonstrated that an MWPM-based decoder, which considers the soft information of the individual measurements when decoding, offers higher thresholds and lower logical error rates than a hard decoder, which only considers the binary outcomes [85]. To demonstrate the flexibility of machine-learning decoders, we consider providing the soft information available from readout when training and evaluating the NN decoder.

In our simulations, measurements project the qubit into either $|0\rangle$ or $|1\rangle$. A measurement outcome $m_{r,q} = i$ of qubit q at round r corresponds to the ancilla qubit being in $|i\rangle$ directly after the measurement. Given $m_{r,q} = i$, we model the soft outcome $\tilde{m}_{r,q} \in \mathbb{R}$ to follow a Gaussian distribution $\mathcal{N}_i$ with mean μ_i and standard deviation σ. The soft outcome $\tilde{m}_{r,q}$ can then be converted to a binary outcome $\bar{m}_{r,q}$ by introducing a

threshold t, such that

$$\bar{m}_{r,a} = \begin{cases} 0 & \text{if } \tilde{m}_{r,a} \leqslant t, \\ 1 & \text{otherwise.} \end{cases}$$

For the symmetric Gaussian distributions that we consider, this process leads to an assignment error probability $P(\bar{m}_{r,q} = 0 \mid m_{r,q} = 1) = P(\bar{m}_{r,q} = 1 \mid m_{r,q} = 0) = p_m$. This assignment error is *added* to the errors considered in our circuit-level noise models, specifically the X error before each measurement that happens with a probability p. The assignment error probability can be related to the signal-to-noise ratio $\mathrm{SNR} = |\mu_0 - \mu_1|/2\sigma$ as $p_m = \frac{1}{2}\mathrm{erfc}(\frac{\mathrm{SNR}}{\sqrt{2}})$. We fix $\mu_0 = -1$ and $\mu_1 = 1$ such that a given probability p_m fixes the standard deviation σ of the two distributions.

The most straightforward approach to incorporating the soft information into the NN decoder is to directly provide the soft measurement outcomes $\tilde{m}_{r,q}$ as input during training and evaluation. However, we find that doing this leads to an overall poor logical performance. Instead, we estimate the probability of a defect $P(d_{r,a} = 1 \mid \tilde{m}_{r,a}, \tilde{m}_{r-1,a})$, given the soft measurement outcomes of an ancilla qubit a in consecutive QEC rounds. Given a soft outcome $\tilde{m}_{r,q}$, the probability of the measured qubit "having being in the state" $|i\rangle$ can be expressed as

$$P(i \mid \tilde{m}_{r,q}) = \frac{P(\tilde{m}_{r,q} \mid i)P(i)}{\sum_{j\in\{1,2\}} P(\tilde{m}_{r,q} \mid j)P(j)}.$$

The soft outcomes follow a Gaussian distribution, that is, $P(\tilde{m}_{r,q} \mid i) = \mathcal{N}_i(\tilde{m}_{r,q})$. Finally, we make the simplifying assumption that the prior state probabilities $P(i) = P(j) = \frac{1}{2}$, such that

$$P(i \mid \tilde{m}_{r,q}) = \frac{\mathcal{N}_i(\tilde{m}_{r,q})}{\sum_{j\in\{1,2\}} \mathcal{N}_j(\tilde{m}_{r,q})}.$$

The probability of observing a defect can then be expressed as

$$P(d_{r,a} = 1 \mid \tilde{m}_{r,a}, \tilde{m}_{r-1,a}) = 1 - \sum_{i\in\{0,1\}} P(i \mid \tilde{m}_{r,a})P(i \mid \tilde{m}_{r-1,a}).$$

The expression for the defect probability inferred from using the soft (final) data qubit measurement outcomes can be derived similarly.

To explore the performance of the soft NN decoder, we simulate the $d = 3$ surface-code memory experiment using a circuit-level noise model with an error rate per operation of $p = 0.1\%$. We consider two separate assignment error probabilities p_m^a and p_m^d for ancilla qubit and data qubit measurements. We motivate this choice by the fact that data qubits remain idling while the ancilla qubits are being measured. A shorter measurement time can reduce the decoherence experienced by the data qubits but will typically lead to a higher p_m^a. The data qubit measurements at the end of the experiment, on the other hand, can be optimized to minimize p_m^d. Therefore, we focus on how a soft decoder can help with decoding when p_m^a is higher, similar to the discussion in [85]. We train the NN decoder using data sets of $r = 1, 5, \ldots, 37$ QEC rounds, sampling 5×10^5 shots for each round and initial logical state. When evaluating the performance, we simulate $r = 10, 30, \ldots, 150$ QEC rounds, sampling 5×10^4 shots instead.

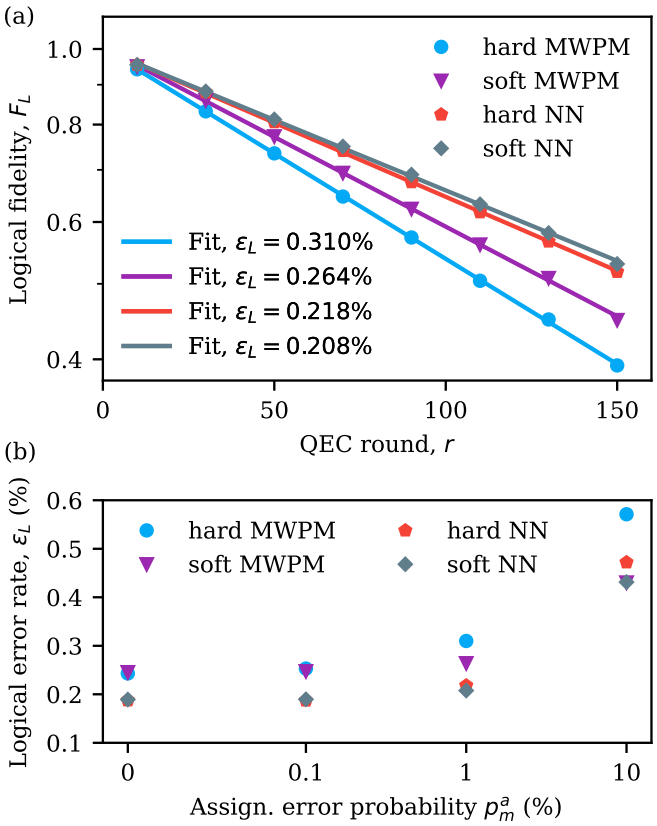


FIG. 7. (a) The logical fidelity F_L as a function of the number of QEC rounds r for a hard and a soft version of the MWPM decoder (blue circles and purple triangles, respectively) and the NN decoder (red pentagons and gray diamonds, respectively). The soft decoders use the soft information by using the probability of observing defects in the case of the soft NN decoder or the likelihood of an assignment error [85] in the case of the soft MWPM decoder. The hard decoders use the defects obtained from the hard measurement outcomes. This performance is estimated on simulated data using a uniform depolarizing circuit-level noise model with an error probability $p = 0.1\%$. The soft outcome distributions are such that ancilla and data qubits have a probability of assignment errors of $p_m^a = 1\%$ and $p_m^d = 0.1\%$, respectively. Solid lines show the fits to the data used to extract the logical error rate per round ε_L. Each data point is averaged over 10^5 shots. (b) The extracted logical error rate ε_L for each of the four decoders as a function of the ancilla qubit assignment error probability p_m^a, keeping $p_m^d = 0.1\%$ and $p = 0.1\%$. The error bars are smaller than the marker sizes.

The results for $p_m^a = 1\%$ are shown in Fig. 7(a). The hard NN decoder achieves an approximately 20% lower logical error rate than the hard MWPM decoder, consistent with the results shown in Fig. 3. Furthermore, the soft NN decoder achieves an approximately 5% lower error rate compared to the hard NN decoder, demonstrating the ability of the decoder to adapt to the provided soft information. Finally, we also compare the performance of these decoders to the soft MWPM decoder proposed in [85]. This decoder encodes the soft information in the weights of the matching graph using the likelihood of an assignment

error $L_{r,a} = \mathcal{N}_{\neg i}(\tilde{m}_{r,a})/\mathcal{N}_i(\tilde{m}_{r,a})$ given a soft outcome $\tilde{m}_{r,a}$ that leads to a hard outcome of $\bar{m}_{r,a} = i$. We observe that using the soft MWPM decoder reduces the logical error rate by approximately 15% relative to the hard MWPM decoder, indicating that the soft NN decoder is not optimally using the available soft information. In Fig. 7(b) the logical error rate ε_L of the three decoders is shown for $p_m^a \in \{0, 0.1\%, 1\%, 10\%\}$, where both NN decoders are trained at the corresponding p_m^a. For low p_m^a, the performance of the soft NN decoder is essentially equivalent to the hard NN decoder, with a moderate reduction in ε_L achieved for $p_m^a \geqslant 1\%$. In particular, for $p_m^a = 10\%$ the soft NN decoder achieves a 10% lower logical error rate compared to the hard NN decoder. We observe that the performance of the soft MWPM decoder becomes closer to that of the soft NN decoder as p_m^a increases, demonstrating that the probability of defects is likely not the optimal way to provide the soft information to the decoder. Another downside of this representation is that for a high assignment error probability $p_m^a \geqslant 20\%$, the probability of observing a defect is close to 50%, which also impacts the training and leads the soft NN decoder to exhibit a higher logical error rate compared to the hard one (not shown in Fig. 7). Finding a more optimal representation of the soft information that can be provided to the NN decoder and optimizing its performance remain open questions.

IV. DISCUSSION

We now discuss in more detail the performance of the NN decoder on the experimental data. Unfortunately, we only use simulated data to train the NN decoder throughout this work. These simulations use approximate Pauli-noise models that account for the most significant error mechanisms in the experiment, such as decoherence and readout errors. However, they do not include several important error sources present in the actual experiments, such as leakage, crosstalk, and stray interactions. The exclusion of these error mechanisms leads to the Pauli-noise models underpredicting the logical error rate compared to the rates observed in the experiment, as observed in Fig. 4. Furthermore, it was shown that the $d = 5$ code is more sensitive to errors such as leakage and crosstalk, which can lead to a more significant deviation relative to simulations of the $d = 3$ codes [26]. Despite using these approximate models for training, when evaluating the NN decoder on experimental data, we observe that it outperforms MWPM and can achieve logical error rates comparable to those obtained using maximum-likelihood decoding, which is approximated by the TN decoder. The TN decoder requires information about the error probabilities, what defects they lead to, and their corresponding corrections, which can be encoded into a hypergraph, where the nodes correspond to defects and the hyperedges represent errors. Importantly, this hypergraph also does not explicitly include hyperedges corresponding to nonconventional errors, such as leakage or crosstalk. We expect that training on experimental data and optimizing the hyperparameters of the network will enable it to match the performance of the TN decoder closely and potentially exceed it by learning about errors not included in the hypergraph.

Despite the large volume of training data required to achieve good performance, we do not expect that generating sufficient experimental data for training will be an issue. Assuming that the QEC round duration is 1 μs and that it takes 200 ns to reset all qubits between subsequent runs, we estimate that it would take approximately three minutes to generate the data sets with 10^7 shots running $r = 1, 5, \ldots, 37$ rounds of QEC that were used for training the $d = 3, 5, 7$ surface codes; see Table I.

The soft NN decoder used in this work achieves only a moderate performance increase compared to the hard NN decoder. Furthermore, it uses the available soft information less optimally than the soft MWPM decoder. An alternative approach to incorporating the soft information into the decoder is to estimate the likelihood of assignment errors $L_{r,a}$ used by the soft MWPM decoder and to provide them as input to the NN decoder together with the (hard) defects $d_{r,a}$ that were measured. In addition to the representation of the input data, it is an open question whether using a soft NN decoder will be useful in practice, where assignment error rates are typically low. Specifically, it would be interesting to see if using a soft NN decoder will enable using a shorter measurement time that might lead to a higher assignment error rate but maximize the logical performance overall, as discussed in [85]. The symmetric Gaussian distributions of the continuous measurement outcomes we consider here are only very simple approximations of the distributions seen in experiments, and in our modeling we could adapt these. In particular, the relaxation that the qubit experiences during the readout leads to an asymmetry between the distributions and a generally higher probability of an assignment error when the qubit was prepared in $|1\rangle$. Furthermore, the continuous outcomes observed in the experiment can also contain information about leakage [28,95,96] or correlations with other measurements. Therefore, it will be essential to investigate and optimize the performance of the soft decoders using experimental data (see the performance of a modified version of our NN decoder and the use of soft information in a next superconducting qubit experiment [97]).

Finally, we outline some possible directions for future research necessary to use these decoders for decoding large-distance experiments. Decoders based on feedforward and convolutional architectures have been shown to achieve low-latency decoding, making them a possible candidate for being used in real time [75–78]. On the other hand, recurrent networks generally have a larger number of parameters and carry out more complex operations when processing the data. However, recurrent NN decoders have been shown to achieve higher accuracy and be more easily trainable than other architectures, especially when considering realistic noise models [69]. Therefore, whether hardware implementations of recurrent NN decoders can be used for real-time decoding is an open question. In addition to the latency, the scalability of NN decoders is an open question. Decoding higher-distance codes will require larger neural networks and larger training data sets, which will most likely be more challenging to train, given that approaches based on machine learning generally struggle when the dimension of the input becomes very large. Practically, one might be interested in whether the NN decoder can be trained and used to decode some finite code distance, which is expected to lead to algorithmically relevant logical error rates given the processor's performance. Alternatively, there

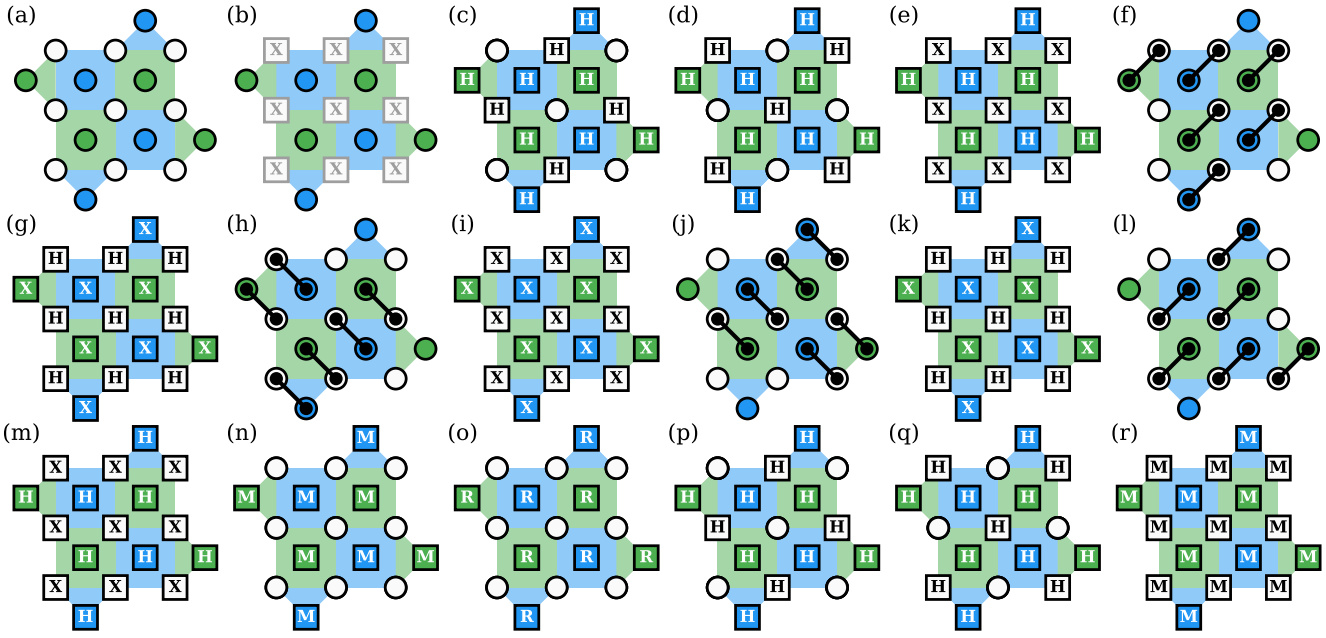


FIG. 8. Schematic of the circuits used in the quantum memory experiments for a $d = 3$ surface code. Parts (a)–(d) are used to initialize the logical state at the start of each experiment. The qubits are first prepared in the ground state (a), after which a set of conditional X gates (gray) are used to prepare the data qubits in a bit-string state (b). Afterward, a set of H (Hadamard) gates transform this into an eigenstate of the X-type (c) or Z-type (d) stabilizers. Parts (e)–(o) show the circuits used to measure the stabilizers. The ancilla qubits are first placed in a superposition by a set of H gates [(c) or (d) in the first round, (e) otherwise]. The parity of the neighboring data qubits is then mapped using four CZ gates [(f), (h), (j), and (l)]. The order of the gates used to measure the X- and Z-type stabilizers is chosen to avoid any "hook" errors propagating to a logical error. Two layers of H gates are applied to the data qubits [(g) and (k)] to measure the parity in the X-basis. In the middle of this sequence, X gates are applied to all qubits [(i)] for dynamical decoupling. Finally, the ancilla qubits are rotated back [(m)] using a set of H gates, measured [(n), denoted by M] and reset [(o), denoted by R]. Several X gates are applied to the data qubit throughout this sequence for dynamical decoupling. In the final round, all data qubits are measured (p)–(r), which is also a logical measurement. Some of the data qubits are rotated depending on whether the experiment is done in the X [(p)] or Z [(q)] logical basis. This step replaces (m) in the final round. Afterward, all qubits are measured simultaneously [(r)], replacing (n) in the final round. Data qubits are denoted with white circles, while ancilla qubits are illustrated as blue and green circles. For the definition of the plaquettes, see Fig. 1. The circuits we run follow the ones used in [26].

exist approaches that enable scalable NN decoders. These are typically based on convolutional neural networks that learn to infer and correct the physical errors that have occurred while a secondary global decoder handles any possibly remaining errors [75,77], but a purely convolutional NN method has been explored as well [66]. The recurrent NN decoder used in this work is not scalable, and adapting it to work with larger code distances and using it to decode through logical operations is another open research venue.

The data and software that support the plots presented in this figure are available in [98]. The raw simulated data and the scripts used for training and decoding these data are available upon reasonable request.

ACKNOWLEDGMENTS

We are grateful to E. Campbell for insightful discussions and for comments on the manuscript. We also thank L. Caune for implementing the belief-matching decoder that we have used in this work. B.M.V. and B.M.T. are supported by QuTech NWO funding 2020-2024 – Part I "Fundamental Research" with Project No. 601.QT.001-1. B.M.T and M.S.-P. thank the OpenSuperQPlus100 Project (No. 101113946) of the EU Flagship on Quantum Technology (No. HORIZON-CL4-QUANTUM-01-SGA) for support.

APPENDIX

1. Quantum memory experiments

To characterize the logical performance of a surface code, we look at its ability to maintain an initial logical state as a function of the number of QEC rounds, commonly referred to as a quantum memory experiment. The circuits used to perform these experiments are illustrated in Fig. 8 and follow the ones used in the recent $d = 5$ surface code experiment done by Google Quantum AI [26]. Removing some of the Hadamard gates when compiling the stabilizer measurement circuits leads to each ancilla qubit measuring the $ZXXZ$ operator instead of the standard $XXXX$ and $ZZZZ$ stabilizers of the surface code. Implementing this $ZXXZ$ variant of the surface code symmetrizes the logical error rates between experiments done in the logical X-basis or Z-basis [26]. Despite this modification, we use notations associated with the traditional stabilizers measured by the surface code.

Each experiment begins by preparing a given logical state, performed by the circuits in Figs. 8(a)–8(d). The data qubits are first initialized in the ground state and then prepared in

either $|0\rangle$ or $|1\rangle$ by a layer of conditional X gates. A subset of the data qubits is then rotated and transforms the initial state into an eigenstate of the X- or Z-type stabilizers. The parity of the initial bistring state determines whether $|0\rangle_L$ or $|1\rangle_L$ ($|+\rangle_L$ or $|-\rangle_L$) is prepared if the experiment is done in the Z-basis (X-basis). In simulation, we prepare either $|0\rangle^{\otimes n}$ or $|1\rangle^{\otimes n}$ when using uniform circuit-level noise models. In the experiment, several random bitstring states are used in order to symmetrize the impact of amplitude damping [26].

The prepared logical state is then maintained over a total of $r \in \{1, 2, \ldots, N-1\}$ QEC rounds, with the circuit given by Figs. 8(e)–8(o). The first QEC round then projects this initial state into a simultaneous eigenstate of both the X- or Z-type stabilizers. Each cycle involves a series of four interactions between each ancilla qubit and its neighboring data qubits, which map the X or Z parity onto the state of the ancilla qubit. The order in which these two-qubit operations are executed is carefully chosen to minimize the impact of errors occurring during the execution of the circuit [86]. At the end of each QEC round, all of the ancilla qubits are measured and reset. The stabilizer measurement circuits also contain several echo (X) gates on either the data or ancilla qubits, shown in Figs. 8(e)–8(m), which dynamically decouple the qubits in the experiment [26]. These echo gates are used to mitigate the dephasing experienced by the qubits due to the low-frequency flux noise. Naturally, these gates do not improve the logical performance extracted from simulations using the approximate Pauli-error models that we consider here. Instead, these gates are included to account for the fact that these operations are implemented with a certain error rate. In the final QEC round, the data qubits rotated during the state preparation are rotated back and measured in the Z-basis together with the ancilla qubits, illustrated in Figs. 8(p)–8(r). The data qubit measurement outcomes are then used to calculate the value of the X_L or Z_L logical observable as well as to infer a final set of X- or Z-type stabilizer measurement outcomes.

2. Decoder training and evaluation

Here we provide additional details about how we train the NN decoder and the hyperparameters we use. We use the Adam optimizer typically with a learning rate of 10^{-3} or 5×10^{-4} for training. In addition, we apply dropout after the hidden layer of the feedforward network of each head and, in some cases, after the second LSTM layer with a dropout rate of either 20% or 5% to avoid overfitting and assist with the generalization of the network. We use a batch size of 256 or 64, which we found to lead to a smoother minimization of the loss. After each training epoch, we evaluate the loss of the network on a separate data set that considers the same number of QEC rounds and prepared states as the training data set but samples fewer shots for each experiment. After each epoch, we save the networks' weights if a lower loss has been achieved. Furthermore, we use early stopping to end the training if the loss has not decreased over the last 20 epochs to reduce the time it takes to train each model. We have observed that not using early-stopping and leaving the training to continue does not typically lead the network to reach a lower loss eventually. For some data sets, we lower the learning rate after the initial training has stopped early and train the network once more to achieve better performance. The hyperparameters we have used for training each network and the parameters of the training data sets used are presented in Table I.

The NN architecture we employ in this work uses two stacked LSTM layers to process the recurrent input [64]. We observe poor logical performance for a $d = 3$ surface code when using only a single LSTM layer. On the other hand, we see no significant improvement in the logical error rate when using four layers instead, motivating the choice to use only two. This network architecture also performs well when decoding $d = 5$ and 7 surface code experiments. However, we expect that a deeper recurrent network might improve the logical error rates when decoding larger-distance codes or when training on and decoding experimental data. We have also practically observed that training the NN decoder for larger distances is more challenging, especially if the physical error rates are small. Training the neural network on a data set with a higher physical error rate (in addition to data using the same error rate as the evaluation data set) can also improve the performance of the decoder, as we also discussed in Sec. III C.

The training of our neural networks was performed on the DelftBlue supercomputer [99] and was carried out on an NVIDIA Tesla V100S GPU. Once trained, the decoder takes approximately 0.7 s per QEC round for a $d = 3$ surface code (corresponding to an internal state size of $N_L = 64$) using a batch size of 50 000 shots on an Intel(R) Core(TM) i7-8850H CPU @ 2.60GHz. For a $d = 5$ surface code ($N_L = 96$), it takes about 0.8 s per round, while for a $d = 7$ surface code ($N_L = 128$), it takes about 1.1 s per round, using the same batch size of 50 000 shots. We note that using smaller batch sizes leads to a higher overall runtime due to parallelism when the network processes the inputs. Therefore, larger batch sizes are preferable as long as they fit into the memory. Each runtime was extracted by decoding simulated data sets running $r = 10, 30, \ldots, 290$ rounds of QEC and averaging the runtime per QEC round over all the data sets.

[1] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM J. Comput. **26**, 1484 (1997).

[2] S. Lloyd, Universal quantum simulators, Science **273**, 1073 (1996).

[3] M. Reiher, N. Wiebe, K. M. Svore, D. Wecker, and M. Troyer, Elucidating reaction mechanisms on quantum computers, Proc. Natl. Acad. Sci. USA **114**, 7555 (2017).

[4] C. Gidney and M. Ekerå, How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits, Quantum **5**, 433 (2021).

[5] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. C. White, J. Mutus, A. G. Fowler, B. Campbell *et al.*, Superconducting quantum circuits at the surface code threshold for fault tolerance, Nature (London) **508**, 500 (2014).

[6] M. A. Rol, C. C. Bultink, T. E. O'Brien, S. R. de Jong, L. S. Theis, X. Fu, F. Luthi, R. F. L. Vermeulen, J. C. de Sterke,

A. Bruno, D. Deurloo, R. N. Schouten, F. K. Wilhelm, and L. DiCarlo, Restless tuneup of high-fidelity qubit gates, Phys. Rev. Appl. **7**, 041001(R) (2017).

[7] R. Barends, C. M. Quintana, A. G. Petukhov, Y. Chen, D. Kafri, K. Kechedzhi, R. Collins, O. Naaman, S. Boixo, F. Arute *et al.*, Diabatic gates for frequency-tunable superconducting qubits, Phys. Rev. Lett. **123**, 210501 (2019).

[8] M. A. Rol, F. Battistel, F. K. Malinowski, C. C. Bultink, B. M. Tarasinski, R. Vollmer, N. Haider, N. Muthusubramanian, A. Bruno, B. M. Terhal, and L. DiCarlo, Fast, high-fidelity conditional-phase gate exploiting leakage interference in weakly anharmonic superconducting qubits, Phys. Rev. Lett. **123**, 120502 (2019).

[9] V. Negîrneac, H. Ali, N. Muthusubramanian, F. Battistel, R. Sagastizabal, M. S. Moreira, J. F. Marques, W. J. Vlothuizen, M. Beekman, C. Zachariadis *et al.*, High-fidelity controlled-*Z* gate with maximal intermediate leakage operating at the speed limit in a superconducting quantum processor, Phys. Rev. Lett. **126**, 220502 (2021).

[10] B. Foxen *et al.* (Google AI Quantum), Demonstrating a continuous set of two-qubit gates for near-term quantum algorithms, Phys. Rev. Lett. **125**, 120504 (2020).

[11] P. Jurcevic, A. Javadi-Abhari, L. S. Bishop, I. Lauer, D. F. Bogorin, M. Brink, L. Capelluto, O. Günlük, T. Itoko, N. Kanazawa *et al.*, Demonstration of quantum volume 64 on a superconducting quantum computing system, Quantum Sci. Technol. **6**, 025020 (2021).

[12] T. P. Harty, D. T. C. Allcock, C. J. Ballance, L. Guidoni, H. A. Janacek, N. M. Linke, D. N. Stacey, and D. M. Lucas, High-fidelity preparation, gates, memory, and readout of a trapped-ion quantum bit, Phys. Rev. Lett. **113**, 220501 (2014).

[13] S. S. Hong, A. T. Papageorge, P. Sivarajah, G. Crossman, N. Didier, A. M. Polloreno, E. A. Sete, S. W. Turkowski, M. P. da Silva, and B. R. Johnson, Demonstration of a parametrically activated entangling gate protected from flux noise, Phys. Rev. A **101**, 012302 (2020).

[14] W. Huang, C. H. Yang, K. W. Chan, T. Tanttu, B. Hensen, R. C. C. Leon, M. A. Fogarty, J. C. C. Hwang, F. E. Hudson, K. M. Itoh *et al.*, Fidelity benchmarks for two-qubit gates in silicon, Nature (London) **569**, 532 (2019).

[15] P. W. Shor, Scheme for reducing decoherence in quantum computer memory, Phys. Rev. A **52**, R2493 (1995).

[16] E. Knill, R. Laflamme, and W. H. Zurek, Resilient quantum computation, Science **279**, 342 (1998).

[17] D. Aharonov and M. Ben-Or, Fault-tolerant quantum computation with constant error rate, SIAM J. Comput. **38**, 1207 (2008).

[18] D. Gottesman, Fault-tolerant quantum computation with constant overhead, Quantum Inf. Comput. **14**, 1338 (2014).

[19] D. Gottesman, Stabilizer codes and quantum error correction, Ph.D. thesis, California Institute of Technology (1997).

[20] A. Y. Kitaev, Fault-tolerant quantum computation by anyons, Ann. Phys. **303**, 2 (2003).

[21] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Topological quantum memory, J. Math. Phys. **43**, 4452 (2002).

[22] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Surface codes: Towards practical large-scale quantum computation, Phys. Rev. A **86**, 032324 (2012).

[23] R. Raussendorf and J. Harrington, Fault-tolerant quantum computation with high threshold in two dimensions, Phys. Rev. Lett. **98**, 190504 (2007).

[24] A. D. Córcoles, A. Kandala, A. Javadi-Abhari, D. T. McClure, A. W. Cross, K. Temme, P. D. Nation, M. Steffen, and J. M. Gambetta, Challenges and opportunities of near-term quantum computing systems, Proc. IEEE **108**, 1338 (2020).

[25] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell *et al.*, Quantum supremacy using a programmable superconducting processor, Nature (London) **574**, 505 (2019).

[26] R. Acharya, I. Aleiner, R. Allen, T. I. Andersen, M. Ansmann, F. Arute, K. Arya, A. Asfaw, J. Atalaya, R. Babbush *et al.*, Suppressing quantum errors by scaling a surface code logical qubit, Nature (London) **614**, 676 (2023).

[27] N. Sundaresan, T. J. Yoder, Y. Kim, M. Li, E. H. Chen, G. Harper, T. Thorbeck, A. W. Cross, A. D. Córcoles, and M. Takita, Demonstrating multi-round subsystem quantum error correction using matching and maximum likelihood decoders, Nat. Commun. **14**, 2852 (2023).

[28] J. Heinsoo, C. K. Andersen, A. Remm, S. Krinner, T. Walter, Y. Salathé, S. Gasparinetti, J.-C. Besse, A. Potočnik, A. Wallraff, and C. Eichler, Rapid high-fidelity multiplexed readout of superconducting qubits, Phys. Rev. Appl. **10**, 034040 (2018).

[29] J. F. Marques, H. Ali, B. M. Varbanov, M. Finkel, H. M. Veen, S. L. M. van der Meer, S. Valles-Sanclemente, N. Muthusubramanian, M. Beekman, N. Haider *et al.*, All-microwave leakage reduction units for quantum error correction with superconducting transmon qubits, Phys. Rev. Lett. **130**, 250602 (2023).

[30] M. McEwen, D. Kafri, Z. Chen, J. Atalaya, K. J. Satzinger, C. Quintana, P. V. Klimov, D. Sank, C. Gidney, A. G. Fowler *et al.*, Removing leakage-induced correlated errors in superconducting quantum error correction, Nat. Commun. **12**, 1761 (2021).

[31] K. C. Miao, M. McEwen, J. Atalaya, D. Kafri, L. P. Pryadko, A. Bengtsson, A. Opremcak, K. J. Satzinger, Z. Chen, P. V. Klimov *et al.*, Overcoming leakage in quantum error correction, Nat. Phys. **19**, 1780 (2023).

[32] E. Jeffrey, D. Sank, J. Y. Mutus, T. C. White, J. Kelly, R. Barends, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth *et al.*, Fast accurate state measurement with superconducting qubits, Phys. Rev. Lett. **112**, 190504 (2014).

[33] C. C. Bultink, M. A. Rol, T. E. O'Brien, X. Fu, B. C. S. Dikken, C. Dickel, R. F. L. Vermeulen, J. C. de Sterke, A. Bruno, R. N. Schouten, and L. DiCarlo, Active resonator reset in the nonlinear dispersive regime of circuit QED, Phys. Rev. Appl. **6**, 034008 (2016).

[34] J. Kelly, R. Barends, A. G. Fowler, A. Megrant, E. Jeffrey, T. White, D. Sank, J. Mutus, B. Campbell, Y. Chen *et al.*, State preservation by repetitive error detection in a superconducting quantum circuit, Nature (London) **519**, 66 (2015).

[35] L. Egan, D. M. Debroy, C. Noel, A. Risinger, D. Zhu, D. Biswas, M. Newman, M. Li, K. R. Brown, M. Cetina, and C. Monroe, Fault-tolerant control of an error-corrected qubit, Nature (London) **598**, 281 (2021).

[36] M. H. Abobeih, Y. Wang, J. Randall, S. J. H. Loenen, C. E. Bradley, M. Markham, D. J. Twitchen, B. M. Terhal, and T. H. Taminiau, Fault-tolerant operation of a logical qubit in a diamond quantum processor, Nature (London) **606**, 884 (2022).

[37] C. Ryan-Anderson, J. G. Bohnet, K. Lee, D. Gresh, A. Hankin, J. P. Gaebler, D. Francois, A. Chernoguzov, D. Lucchetti, N. C. Brown *et al.*, Realization of real-time fault-tolerant quantum error correction, Phys. Rev. X **11**, 041058 (2021).

[38] J. F. Marques, B. M. Varbanov, M. S. Moreira, H. Ali, N. Muthusubramanian, C. Zachariadis, F. Battistel, M. Beekman, N. Haider, W. Vlothuizen *et al.*, Logical-qubit operations in an error-detecting surface code, Nat. Phys. **18**, 80 (2022).

[39] Z. Chen, K. J. Satzinger, J. Atalaya, A. N. Korotkov, A. Dunsworth, D. Sank, C. Quintana, M. McEwen, R. Barends, P. V. Klimov *et al.*, Exponential suppression of bit or phase errors with cyclic error correction, Nature (London) **595**, 383 (2021).

[40] C. K. Andersen, A. Remm, S. Lazar, S. Krinner, N. Lacroix, G. J. Norris, M. Gabureac, C. Eichler, and A. Wallraff, Repeated quantum error detection in a surface code, Nat. Phys. **16**, 875 (2020).

[41] S. Krinner, N. Lacroix, A. Remm, A. Di Paolo, E. Genois, C. Leroux, C. Hellings, S. Lazar, F. Swiadek, J. Herrmann *et al.*, Realizing repeated quantum error correction in a distance-three surface code, Nature (London) **605**, 669 (2022).

[42] Y. Zhao, Y. Ye, H.-L. Huang, Y. Zhang, D. Wu, H. Guan, Q. Zhu, Z. Wei, T. He, S. Cao *et al.*, Realization of an error-correcting surface code with superconducting qubits, Phys. Rev. Lett. **129**, 030501 (2022).

[43] N. Ofek, A. Petrenko, R. Heeres, P. Reinhold, Z. Leghtas, B. Vlastakis, Y. Liu, L. Frunzio, S. M. Girvin, L. Jiang *et al.*, Extending the lifetime of a quantum bit with error correction in superconducting circuits, Nature (London) **536**, 441 (2016).

[44] A. Grimm, N. E. Frattini, S. Puri, S. O. Mundhada, S. Touzard, M. Mirrahimi, S. M. Girvin, S. Shankar, and M. H. Devoret, Stabilization and operation of a Kerr-cat qubit, Nature (London) **584**, 205 (2020).

[45] P. Campagne-Ibarcq, A. Eickbusch, S. Touzard, E. Zalys-Geller, N. E. Frattini, V. V. Sivak, P. Reinhold, S. Puri, S. Shankar, R. J. Schoelkopf *et al.*, Quantum error correction of a qubit encoded in grid states of an oscillator, Nature (London) **584**, 368 (2020).

[46] V. V. Sivak, A. Eickbusch, B. Royer, S. Singh, I. Tsioutsios, S. Ganjam, A. Miano, B. L. Brock, A. Z. Ding, L. Frunzio *et al.*, Real-time quantum error correction beyond break-even, Nature (London) **616**, 50 (2023).

[47] A. G. Fowler, A. C. Whiteside, and L. C. L. Hollenberg, Towards practical classical processing for the surface code, Phys. Rev. Lett. **108**, 180501 (2012).

[48] A. G. Fowler, Minimum weight perfect matching of fault-tolerant topological quantum error correction in average O(1) parallel time, Quantum Inf. Comput. **15**, 145 (2015).

[49] O. Higgott and C. Gidney, Sparse blossom: correcting a million errors per core second with minimum-weight matching, arXiv:2303.15933.

[50] Y. Wu and L. Zhong, Fusion blossom: Fast MWPM decoders for QEC, in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)* (Bellevue, WA, USA, 2023), pp. 928–938.

[51] J. Roffe, D. R. White, S. Burton, and E. Campbell, Decoding across the quantum low-density parity-check code landscape, Phys. Rev. Res. **2**, 043423 (2020).

[52] B. Criger and I. Ashraf, Multi-path summation for decoding 2D topological codes, Quantum **2**, 102 (2018).

[53] O. Higgott, T. C. Bohdanowicz, A. Kubica, S. T. Flammia, and E. T. Campbell, Improved decoding of circuit noise and fragile boundaries of tailored surface codes, Phys. Rev. X **13**, 031007 (2023).

[54] L. Caune, J. Camps, B. Reid, and E. Campbell, Belief propagation as a partial decoder, arXiv:2306.17142.

[55] S. Bravyi, M. Suchara, and A. Vargo, Efficient algorithms for maximum likelihood decoding in the surface code, Phys. Rev. A **90**, 032326 (2014).

[56] C. T. Chubb and S. T. Flammia, Statistical mechanical models for quantum codes with correlated noise, Ann. l'Inst. Henri Poincare D **8**, 269 (2021).

[57] S. Spitz, B. M. Tarasinski, C. Beenakker, and T. O'Brien, Adaptive weight estimator for quantum error correction in a time-dependent environment, Adv. Quantum Technol. **1**, 1800012 (2018).

[58] E. H. Chen, T. J. Yoder, Y. Kim, N. Sundaresan, S. Srinivasan, M. Li, A. D. Córcoles, A. W. Cross, and M. Takita, Calibrated decoders for experimental quantum error correction, Phys. Rev. Lett. **128**, 110504 (2022).

[59] G. Torlai and R. G. Melko, Neural decoder for topological codes, Phys. Rev. Lett. **119**, 030501 (2017).

[60] S. Krastanov and L. Jiang, Deep neural network probabilistic decoder for stabilizer codes, Sci. Rep. **7**, 11003 (2017).

[61] S. Varsamopoulos, B. Criger, and K. Bertels, Decoding small surface codes with feedforward neural networks, Quantum Sci. Technol. **3**, 015004 (2018).

[62] P. Baireuther, T. E. O'Brien, B. Tarasinski, and C. W. J. Beenakker, Machine-learning-assisted correction of correlated qubit errors in a topological code, Quantum **2**, 48 (2018).

[63] C. Chamberland and P. Ronagh, Deep neural decoders for near term fault-tolerant experiments, Quantum Sci. Technol. **3**, 044002 (2018).

[64] P. Baireuther, M. D. Caio, B. Criger, C. W. J. Beenakker, and T. E. O'Brien, Neural network decoder for topological color codes with circuit level noise, New J. Phys. **21**, 013003 (2019).

[65] P. Andreasson, J. Johansson, S. Liljestrand, and M. Granath, Quantum error correction for the toric code using deep reinforcement learning, Quantum **3**, 183 (2019).

[66] X. Ni, Neural network decoders for large-distance 2D toric codes, Quantum **4**, 310 (2020).

[67] T. Wagner, H. Kampermann, and D. Bruß, Symmetries for a high-level neural decoder on the toric code, Phys. Rev. A **102**, 042411 (2020).

[68] M. Sheth, S. Z. Jafarzadeh, and V. Gheorghiu, Neural ensemble decoding for topological quantum error-correcting codes, Phys. Rev. A **101**, 032338 (2020).

[69] S. Varsamopoulos, K. Bertels, and C. Almudever, Comparing neural network based decoders for the surface code, IEEE Trans. Comput. **69**, 300 (2020).

[70] S. Varsamopoulos, K. Bertels, and C. G. Almudever, Decoding surface code with a distributed neural network–based decoder, Quantum Machine Intell. **2**, 3 (2020).

[71] D. Fitzek, M. Eliasson, A. F. Kockum, and M. Granath, Deep Q-learning decoder for depolarizing noise on the toric code, Phys. Rev. Res. **2**, 023230 (2020).

[72] R. Sweke, M. S. Kesselring, E. P. L. van Nieuwenburg, and J. Eisert, Reinforcement learning decoders for fault-tolerant quantum computation, Machine Learning: Sci. Technol. **2**, 025005 (2021).

[73] K. Meinerz, C.-Y. Park, and S. Trebst, Scalable neural decoder for topological surface codes, Phys. Rev. Lett. **128**, 080505 (2022).

[74] Y. Ueno, M. Kondo, M. Tanaka, Y. Suzuki, and Y. Tabuchi, NEO-QEC: Neural network enhanced online superconducting decoder for surface codes, arXiv:2208.05758.

[75] C. Chamberland, L. Goncalves, P. Sivarajah, E. Peterson, and S. Grimberg, Techniques for combining fast local decoders with global decoders under circuit-level noise, Quantum Sci. Technol. **8**, 045011 (2023).

[76] R. W. J. Overwater, M. Babaie, and F. Sebastiano, Neural-network decoders for quantum error correction using surface codes: A space exploration of the hardware cost-performance tradeoffs, IEEE Trans. Quantum Eng. **3**, 1 (2022).

[77] S. Gicev, L. C. L. Hollenberg, and M. Usman, A scalable and fast artificial neural network syndrome decoder for surface codes, Quantum **7**, 1058 (2023).

[78] M. Zhang, X. Ren, G. Xi, Z. Zhang, Q. Yu, F. Liu, H. Zhang, S. Zhang, and Y.-C. Zheng, A scalable, fast and programmable neural decoder for fault-tolerant quantum computation using surface codes, arXiv:2305.15767.

[79] E. Egorov, R. Bondesan, and M. Welling, The END: An equivariant neural decoder for quantum error correction, arXiv:2304.07362.

[80] C. Gidney, Stability experiments: the overlooked dual of memory experiments, Quantum **6**, 786 (2022).

[81] M. Lange, P. Havström, B. Srivastava, V. Bergentall, K. Hammar, O. Heuts, E. van Nieuwenburg, and M. Granath, Data-driven decoding of quantum error correcting codes using graph neural networks, arXiv:2307.01241.

[82] J. Bausch, A. W. Senior, F. J. H. Heras, T. Edlich, A. Davies, M. Newman, C. Jones, K. Satzinger, M. Y. Niu, S. Blackwell *et al.*, Learning high-accuracy error decoding for quantum processors, Nature (London) **635**, 834 (2024).

[83] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, A quantum engineer's guide to superconducting qubits, App. Phys. Rev. **6**, 021318 (2019).

[84] A. Blais, A. L. Grimsmo, S. M. Girvin, and A. Wallraff, Circuit quantum electrodynamics, Rev. Mod. Phys. **93**, 025005 (2021).

[85] C. A. Pattison, M. E. Beverland, M. P. da Silva, and N. Delfosse, Improved quantum error correction using soft information, arXiv:2107.13589.

[86] Y. Tomita and K. M. Svore, Low-distance surface codes under realistic quantum noise, Phys. Rev. A **90**, 062320 (2014).

[87] C. Gidney, Stim: a fast stabilizer circuit simulator, Quantum **5**, 497 (2021).

[88] B. M. Varbanov and M. Serra-Peralta, surface-sim (2023).

[89] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, TensorFlow: a system for large-scale machine learning, in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16 (USENIX Association, Savannah, GA, USA, 2016), pp. 265–283.

[90] B. M. Varbanov and M. Serra-Peralta, qrennd (2023).

[91] S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Comput. **9**, 1735 (1997).

[92] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, Nature (London) **521**, 436 (2015).

[93] A. G. Fowler, Optimal complexity correction of correlated errors in the surface code, arXiv:1310.0863.

[94] J. Roffe, LDPC: Python tools for low density parity check codes (2022).

[95] D. Sank, Z. Chen, M. Khezri, J. Kelly, R. Barends, B. Campbell, Y. Chen, B. Chiaro, A. Dunsworth, A. Fowler *et al.*, Measurement-induced state transitions in a superconducting qubit: Beyond the rotating wave approximation, Phys. Rev. Lett. **117**, 190503 (2016).

[96] M. Khezri, A. Opremcak, Z. Chen, K. C. Miao, M. McEwen, A. Bengtsson, T. White, O. Naaman, D. Sank, A. N. Korotkov *et al.*, Measurement-induced state transitions in a superconducting qubit: Within the rotating-wave approximation, Phys. Rev. Appl. **20**, 054008 (2023).

[97] H. Ali, J. Marques, O. Crawford, J. Majaniemi, M. Serra-Peralta, D. Byfield, B. Varbanov, B. M. Terhal, L. DiCarlo, and E. T. Campbell, Reducing the error rate of a superconducting logical qubit using analog readout information, Phys. Rev. Appl. **22**, 044031 (2024).

[98] B. M. Varbanov, M. Serra-Peralta, D. Byfield, and B. M. Terhal, Data supporting "Neural network decoder for near-term surface-code experiments", Zenodo (2023), doi:10.5281/zenodo.8108286.

[99] Delft High Performance Computing Centre (DHPC), DelftBlue Supercomputer (Phase 1), https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1 (2022).