General framework for constructing fast and near-optimal machine-learning-based decoder of the topological stabilizer codes

Amarsanaa Davaasuren,[1,*] Yasunari Suzuki,[2,3,†] Keisuke Fujii,[3,4,‡] and Masato Koashi[1,5,§]

[1]*Department of Applied Physics, Graduate School of Engineering, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan*
[2]*NTT Secure Platform Laboratories, NTT Corporation, Musashino 180-8585, Japan*
[3]*JST, PRESTO, 4-1-8 Honcho, Kawaguchi, Saitama, 332-0012, Japan*
[4]*Graduate School of Engineering Science, Osaka University, 1-3 Machikaneyama, Toyonaka, Osaka 560-8531, Japan*
[5]*Photon Science Center, Graduate School of Engineering, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan*

(Received 30 August 2019; revised 21 April 2020; accepted 9 July 2020; published 11 September 2020)

Quantum error correction is an essential technique for constructing a scalable quantum computer. In order to implement quantum error correction with near-term quantum devices, a fast and near-optimal decoding method is required. A decoder based on machine learning is considered one of the most viable solutions for this purpose since its prediction is fast once training has been done, and it is applicable to any quantum error-correcting code and any noise model. So far, various formulations of the decoding problem as the task of machine learning have been proposed. Here we discuss general constructions of machine-learning-based decoders. We find several conditions to achieve near-optimal performance and propose a criterion which should be optimized when the size of a training data set is limited. We also discuss preferable constructions of neural networks and propose a decoder using spatial structures of topological codes using a convolutional neural network. We numerically show that our method can improve the performance of machine-learning-based decoders in various topological codes and noise models.

DOI: 10.1103/PhysRevResearch.2.033399

I. INTRODUCTION

In order to build a scalable quantum computer, quantum error correction (QEC) [1–3] is a vital technique for achieving reliable computation. According to the theory of QEC, if the noise strength is smaller than a certain threshold value, we can protect logical qubits encoded in physical qubits from the noise. Supported by extensive experimental efforts, the noise level of the quantum operations on arrays of qubits is now approaching and meets the threshold value. Therefore, a demonstration of QEC in a fully fault-tolerant setting is considered to be a milestone for near-term quantum devices [4–6]. Topological codes [7–9] are a family of quantum error-correcting codes inspired by a topological nature in condensed matter physics [7]. Since topological codes such as surface codes [8,10,11] have both high experimental feasibility and high performance [12–15], they are considered the most promising candidates of quantum error-correcting codes.

In QEC, information on occurrence of physical errors is measured as a syndrome value. A suitable recovery operation is estimated from the syndrome so that the original state of the logical qubits is decoded with a high success probability. Unfortunately, constructing an optimal decoder is computationally hard in general. Thus, massive efforts have been made for developing efficient and near-optimal decoders. One approach is to use the most likely physical errors that are consistent with the observed syndrome value as a recovery operation. This scheme is called the *minimum-distance (MD) decoder*. Though this decoding method is not necessarily optimal, it shows almost optimal performance [13–15]. In the case of the surface codes, if we can assume that bit-flip (Pauli X) and phase-flip (Pauli Z) errors are uncorrelated, we can construct an efficient MD decoder using minimum-weight perfect matching. However, if bit-flip and phase-flip errors are correlated or if we use other codes, even MD decoding is not efficiently implementable [16]. Some of these problems can be avoided using geometrically local features of the topological codes. For example, as for color codes [17], we can perform decoding by projecting a color code to a surface code [18]. Another approach is to use the renormalization group method [19], which is applicable to any topological codes including the surface and color codes. While these approaches have been improved, there is an unavoidable trade-off between the performance and time efficiency of the decoder. For the first experimental realization of QEC on near-term devices, more efficient and near-optimal decoders are required.

In this article, we discuss a general construction of machine-learning-based decoders. Recently, the technology of machine learning has been applied to various theoretical and experimental researches of quantum physics, such as

*amarsanaa137@qc.rcast.u-tokyo.ac.jp
†yasunari.suzuki.gz@hco.ntt.co.jp
‡fujii@qc.ee.es.osaka-u.ac.jp
§koashi@qi.t.u-tokyo.ac.jp

classification of readout signals in experiments [20], simulation of a quantum system [21], classification of the phase of matter [22], data compression of the quantum state [23], and decoding in QEC [24–28]. In the machine-learning-based decoder, we construct a prediction model which outputs a recovery operator from a given syndrome value. The prediction model is trained with many correct pairs of syndrome values and correct recovery operations before prediction. While the training task may take a long time, it is required only once before many prediction runs, and each prediction is expected to be performed fast. Thus, the machine-learning-based decoder is one of the best solutions for demonstrating experimental QEC in near-term quantum devices.

As a prediction model, an artificial neural network is believed to have large representation power and is suitable for constructing a machine-learning-based decoder. Recently, the performances of machine-learning-based decoders with various neural networks have been numerically studied, such as a restricted Boltzmann machine [24], multilayer perceptron [25], recurrent neural network [26], and deep neural network [27]. The machine-learning-based decoder using a neural network is called a *neural decoder* [24]. All these existing methods numerically showed that the performance of the neural decoder is superior to the known efficient decoders when a sufficiently large amount of the training data set is supplied. However, the following three points have yet to be understood. The first one is how the decoding problem should be translated to the task of machine learning in order to obtain faster learning and better prediction. So far, each of the previous studies introduces its own construction of the data set and neural network with little consideration on this point. Second, the spatial feature of the topological codes has not been considered in the construction of the neural decoder, except for a very recent study [28] that was carried out independently of this work. While it is expected that the performance of the neural decoder is improved by explicitly considering the spatial arrangement of the syndrome, the spatial information has not been given to the neural network explicitly. Finally, the applicability of the neural decoder to various topological codes is not known. The neural decoder is benchmarked only with surface codes [24–28]. Therefore, it has not been known whether the neural decoder is applicable to other codes, such as color codes.

We have addressed all of these points in this paper. First, we discuss how the decoding problem should be formulated as a task of machine learning. We propose a general framework for constructing a neural decoder, the *linear prediction framework*, to elucidate the factors that determine the performance of the decoders. We propose a criterion called *normalized sensitivity* which should be optimized for constructing a near-optimal neural decoder. Then we propose the specific construction of a training data set which minimizes the normalized sensitivity. We call these constructions *uniform data construction*. We also propose the use of construction of neural networks which explicitly utilize the spatial structure of the topological codes. We show that the performance of the neural decoder is improved with these techniques, and it shows better performance than that of a decoder using minimum-weight perfect matching with a 10^6 data set at a distance $d = 11$ in the surface code under a depolarizing noise. Though its performance is slightly worse than that of a minimum-distance decoder, the decoding time of the neural decoder is faster than it. We show that the neural decoder is also applicable to the color codes, for which we cannot utilize minimum-weight perfect matching decoder directly. The performance of the neural decoder for the color codes also reaches that of the MD decoder at small distances. Thus, the neural decoder can be used as a fast, versatile, and high-performance decoder for decoding topological stabilizer codes.

A. Organization of the article

In Sec. II we overview preliminary topics. We review a QEC scheme in the case of stabilizer codes. We explain specific constructions of the topological codes and the surface and color codes. We also review the basics of the supervised machine learning with neural networks in this section. In Sec. III we address the question of how the neural decoder should be constructed. We propose a general framework, the linear prediction framework, in this section. We introduce a quantity called the *normalized sensitivity* and argue that it serves as a criterion for better performance of decoders for topological stabilizer codes. We also propose uniform data construction, which consists of specific instructions to optimize the normalized sensitivity for surface codes and color codes. We numerically confirm that the performance of the neural decoder is improved with this construction in the case of the surface and color codes. In Sec. IV we propose a network construction which explicitly utilizes the spatial information of the topological codes. We confirm that this construction also improves the performance of the neural decoder. Finally, we summarize this paper in Sec V.

II. PRELIMINARY INFORMATION

In this section, we review the basic concepts and introduce notations used in this paper. We first review a scheme of QEC. We also introduce well-known topological codes and decoders. Since we employ technical notations, we show a concrete example in Appendix A for clear understanding. The scheme of supervised machine learning with neural network and its terminologies are also explained in this section.

A. Quantum error correction

We consider the case where k logical qubits are encoded in n physical qubits. We assume that any noise can be represented as a probabilistic Pauli operation on the n physical qubits. We denote Pauli operators on a single qubit as $\{I, X, Y, Z\}$ and the Pauli operator A on the ith physical qubit as A_i. When we consider operations on the n physical qubits, we ignore the global phase of the state and operator. Then we can represent any physical error as $E \in \{I, X, Y, Z\}^{\otimes n}$. A weight $w(E)$ is defined for a Pauli operator E on the n physical qubits as the number of the physical qubits to which the Pauli operator E is nontrivially applied.

In the framework of stabilizer codes [29], the code is defined by 2^{n-k} stabilizer operators $\mathcal{L}_I$ generated by $n-k$ Pauli operators $\mathcal{L}_I := \langle\{S_i\}\rangle$ $(1 \leqslant i \leqslant n-k)$, where $S_i \in \pm\{I, X, Y, Z\}^{\otimes n}$, $-I \notin \langle\{S_i\}\rangle$, and they commute with each other. The logical space of the code is defined as the subspace which has eigenvalue $+1$ for all the stabilizer operators, i.e., $S_i |\psi\rangle = |\psi\rangle$ for all i. We denote the normalizer of the stabilizer operators as

$$\begin{aligned}\mathcal{L} &:= \{P | P \in \{I, X, Y, Z\}^{\otimes n}, P\mathcal{L}_I = \mathcal{L}_I P\} \\ &= \{P | P \in \{I, X, Y, Z\}^{\otimes n}, PL_I = L_I P, \forall L_I \in \mathcal{L}_I\}. \quad (1)\end{aligned}$$

We call elements in $\mathcal{L} \setminus \mathcal{L}_I$ logical operators. Each stabilizer operator acts on the logical space trivially, and each logical operator acts on the logical space nontrivially. A distance d of the code is defined as $d := \min_{L\in\mathcal{L}\setminus\mathcal{L}_I} w(L)$. The code which encodes k logical qubits in n physical qubits with distance d is called a $[[n, k, d]]$ code.

The occurrence of a physical error is detected as the outcome of stabilizer measurement $\boldsymbol{s}$, where $\boldsymbol{s}^{\mathrm{T}} \in \{0, 1\}^{n-k}$ and the ith element s_i is the measurement outcome of the ith stabilizer operator S_i. We call $\boldsymbol{s}$ the syndrome vector. To recover the original state of the logical qubits, we estimate a recovery Pauli operator $\hat{T}(\boldsymbol{s}) \in \{I, X, Y, Z\}^{\otimes n}$ from the observed syndrome vector $\boldsymbol{s}$ so that the total operation including the physical error acts on the logical space trivially with a high probability. The mapping from the syndrome $\boldsymbol{s}$ to the recovery operator $\hat{T}(\boldsymbol{s})$ is called decoder $\hat{T}$. The logical error probability p_{L} is defined as the probability with which the total operation becomes logically nontrivial. Our purpose is to construct an efficient decoder $\hat{T}$ which minimizes the logical error probability p_{L}.

B. Binary representation of stabilizer code

It is convenient to translate the calculation in the stabilizer codes into a binary calculation in GF(2). In GF(2), addition $\oplus$ is performed with modulo 2. We relate the Pauli operators on the ith physical qubit to another representation,

$$I_i \mapsto \sigma_{00}^{(i)}, X_i \mapsto \sigma_{10}^{(i)}, Y_i \mapsto \sigma_{11}^{(i)}, Z_i \mapsto \sigma_{01}^{(i)}. \quad (2)$$

Then a Pauli operator P on the n physical qubits can be described as

$$P = \alpha \bigotimes_{i=1}^{n} \sigma_{v_i v_{n+i}}^{(i)}, \quad (3)$$

where $\alpha \in \{\pm 1, \pm i\}$ and $v_i \in \{0, 1\}$ $(1 \leqslant i \leqslant 2n)$. We define a binary mapping

$$b(P) := \boldsymbol{v}, \quad (4)$$

where $\boldsymbol{v} := (v_1, v_2, \ldots, v_{2n-1}, v_{2n}) \in \{0, 1\}^{2n}$ is a row vector, for the Pauli operator $P = \alpha \bigotimes_{i=1}^{n} \sigma_{v_i v_{n+i}}^{(i)}$. For arbitrary two Pauli operators P and P', $b(P) = b(P')$ means that the two Pauli operators are equivalent up to a global phase. The product of two Pauli operators P and P' is represented by the sum $b(PP') = b(P) \oplus b(P')$. With $2n \times 2n$ matrix $\Lambda = \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix}$, the commutation relation of two Pauli operators P and P' is given by $b(P)\Lambda b(P')^{\mathrm{T}}$, which is 0 if P and P' commute, and 1 if anticommute. We denote this commutation relation in terms of the binary representation $\boldsymbol{v}, \boldsymbol{v}' \in \{0, 1\}^{2n}$ as $c(\boldsymbol{v}, \boldsymbol{v}') := \boldsymbol{v}\Lambda\boldsymbol{v}'^{\mathrm{T}}$. The weight of the binary representation of a Pauli operator $w(\boldsymbol{v})$ is defined so as to be $w(b(P)) = w(P)$, which is equivalent to define the weight as the number of indices i $(1 \leqslant i \leqslant n)$ such that

$$v_i \oplus v_{i+n} \oplus v_i v_{i+n} = 1. \quad (5)$$

We use $h(\boldsymbol{v})$ for the Hamming weight of $\boldsymbol{v}$ as a binary string, namely, the number of indices i $(1 \leqslant i \leqslant 2n)$ such that $v_i = 1$. We denote the ith row vector of the matrix M as $(M)_i$. The length of the vector $\boldsymbol{v}$ is represented as $|\boldsymbol{v}|$. With this definition, the normalizer of the stabilizer operators $\mathcal{L}$ is defined as

$$b(\mathcal{L}) = \{\boldsymbol{v} | \boldsymbol{v} \in \{0, 1\}^{2n}, c(\boldsymbol{v}, \boldsymbol{v}') = 0, \forall \boldsymbol{v}' \in b(\mathcal{L}_I)\}. \quad (6)$$

Note that the stabilizer group can be defined with the normalizer $\mathcal{L}$ as

$$b(\mathcal{L}_I) = \{\boldsymbol{v} | \boldsymbol{v} \in \{0, 1\}^{2n}, c(\boldsymbol{v}, \boldsymbol{v}') = 0, \forall \boldsymbol{v}' \in b(\mathcal{L})\}. \quad (7)$$

With this formalism, QEC is translated as follows. The physical error E can be represented as a row binary vector $\boldsymbol{e} := b(E) \in \{0, 1\}^{2n}$ which occurs with a certain probability $p_{\boldsymbol{e}}$. The syndrome vector $\boldsymbol{s}$ is given by a column vector $\boldsymbol{s}(\boldsymbol{e}) := H_c \Lambda \boldsymbol{e}^{\mathrm{T}}$, where H_c is an $(n-k) \times 2n$ matrix of which the ith row vector $(H_c)_i$ is $b(S_i)$. The matrix H_c is called a check matrix. In binary representation, we denote a decoder as $\boldsymbol{r}$ which maps a given syndrome vector $\boldsymbol{s}^{\mathrm{T}} \in \{0, 1\}^{n-k}$ to a binary representation of a recovery operator $\boldsymbol{r}(\boldsymbol{s}) \in \{0, 1\}^{2n}$. It is convenient to define *pure error* $\boldsymbol{t}(\boldsymbol{s})$ [30] to represent various vectors succinctly. The pure error is a function which maps a syndrome vector $\boldsymbol{s}^{\mathrm{T}} \in \{0, 1\}^{n-k}$ to a vector $\boldsymbol{t}(\boldsymbol{s}) \in \{0, 1\}^{2n}$ and satisfies $\boldsymbol{t}(\boldsymbol{s}(\boldsymbol{e})) \oplus \boldsymbol{e} \in b(\mathcal{L})$ for an arbitrary $\boldsymbol{e} \in \{0, 1\}^{2n}$. We also introduce a $2k \times 2n$ generator matrix G such that the elements of $\mathcal{L}$ are uniquely represented as follows:

$$b(\mathcal{L}) = \{\boldsymbol{l}_0 \oplus \boldsymbol{w}G | \boldsymbol{l}_0 \in b(\mathcal{L}_I), \boldsymbol{w} \in \{0, 1\}^{2k}\}. \quad (8)$$

Note that the generator matrix G satisfies $H_c \Lambda G^{\mathrm{T}} = 0$. We define the cosets $\mathcal{L}_{\boldsymbol{w}}$ with $\boldsymbol{w} \in \{0, 1\}^{2k}$ as

$$\mathcal{L}_{\boldsymbol{w}} = \{\boldsymbol{l}_0 \oplus \boldsymbol{w}G | \boldsymbol{l}_0 \in \mathcal{L}_0\}. \quad (9)$$

Note that $\mathcal{L}_0 = b(\mathcal{L}_I)$. Given $\boldsymbol{t}(\boldsymbol{s})$ and G, an arbitrary physical error $\boldsymbol{e} \in \{0, 1\}^{2n}$ is decomposed uniquely in terms of $\boldsymbol{w}(\boldsymbol{e}) \in \{0, 1\}^{2k}$ as

$$\boldsymbol{e} = \boldsymbol{l}(\boldsymbol{e}) \oplus \boldsymbol{w}(\boldsymbol{e})G \oplus \boldsymbol{t}(\boldsymbol{s}(\boldsymbol{e})) \quad (10)$$

with $\boldsymbol{l}(\boldsymbol{e}) \in \mathcal{L}_0$. We call $\boldsymbol{w}(\boldsymbol{e})$ the class of $\boldsymbol{e}$.

A logical decoder with a recovery operation $\boldsymbol{r}(\boldsymbol{s})$ can correct an error $\boldsymbol{e}$ if and only if $\boldsymbol{e} \oplus \boldsymbol{r}(\boldsymbol{s}(\boldsymbol{e})) \in \mathcal{L}_0$. Under an error model $\{p_{\boldsymbol{e}}\}$, the logical error probability is given by

$$\begin{aligned}p_{\mathrm{L}} &= \Pr_{\boldsymbol{e}\sim\{p_{\boldsymbol{e}}\}}[\boldsymbol{e} \oplus \boldsymbol{r}(\boldsymbol{s}(\boldsymbol{e})) \notin \mathcal{L}_0\}] \\ &= \Pr_{\boldsymbol{e}\sim\{p_{\boldsymbol{e}}\}}[\boldsymbol{r}(\boldsymbol{s}(\boldsymbol{e})) \oplus \boldsymbol{w}(\boldsymbol{e})G \oplus \boldsymbol{t}(\boldsymbol{s}(\boldsymbol{e})) \notin \mathcal{L}_0]\}]. \quad (11)\end{aligned}$$

C. Optimal and near-optimal decoders

An optimal decoder is defined as the decoder which minimizes the logical error probability. Let us write the conditional probability of $\boldsymbol{w}(\boldsymbol{e}) \in \{0, 1\}^{2n}$ for a given syndrome vector $\boldsymbol{s}$ as

$$q_{\boldsymbol{s}}(\boldsymbol{w}) := \Pr_{\boldsymbol{e}\sim\{p_{\boldsymbol{e}}\}}[\boldsymbol{w}(\boldsymbol{e}) = \boldsymbol{w} | \boldsymbol{s}(\boldsymbol{e}) = \boldsymbol{s}]. \quad (12)$$

Since the decoder is provided only with $\boldsymbol{s}$ and distinct recovery operators are needed for correcting errors with different values of $\boldsymbol{w}(\boldsymbol{e})$, the maximum probability of successful correction

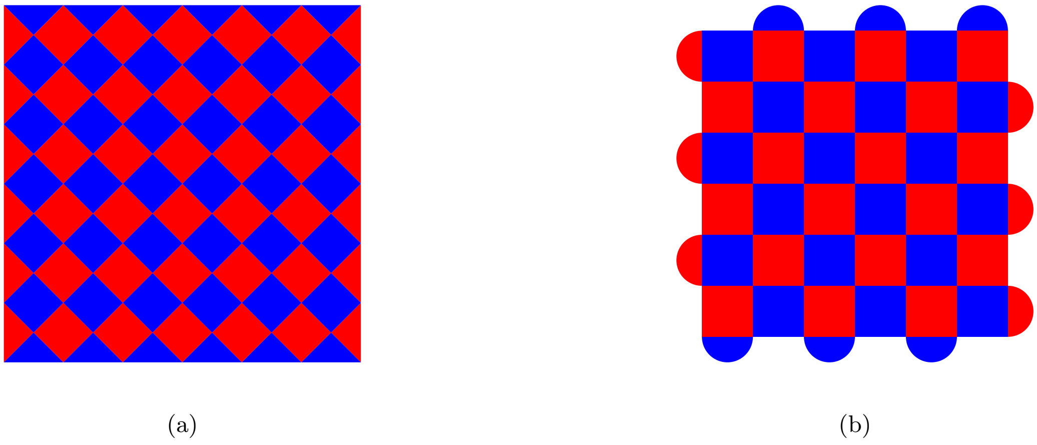


FIG. 1. The qubit allocation of the surface codes with (a) $[[2d^2 - 2d + 1, 1, d]]$ code and (b) $[[d^2, 1, d]]$ code. Each vertex corresponds to a physical qubit. Red and blue faces correspond to stabilizer measurements with X and Z Pauli operators, respectively.

given $\boldsymbol{s}$ is $\max_{\boldsymbol{w}\in\{0,1\}^{2k}} q_{\boldsymbol{s}(\boldsymbol{w})}$. We thus say a decoder is optimal if $\boldsymbol{r}(\boldsymbol{s})$ satisfies

$$\mathrm{Pr}_{\boldsymbol{e}\sim\{p_{\boldsymbol{e}}\}}[\boldsymbol{e} \oplus \boldsymbol{r}(\boldsymbol{s}) \in \mathcal{L}_0 | \boldsymbol{s}(\boldsymbol{e}) = \boldsymbol{s}] = \max_{\boldsymbol{w}\in\{0,1\}^{2k}} q_{\boldsymbol{s}}(\boldsymbol{w}) \quad (13)$$

for any $\boldsymbol{s}$ with

$$\mathrm{Pr}_{\boldsymbol{e}\sim\{p_{\boldsymbol{e}}\}}[\boldsymbol{s}(\boldsymbol{e}) = \boldsymbol{s}] > 0. \quad (14)$$

Though the definition of $\boldsymbol{w}(\boldsymbol{e})$ is dependent on the choice of $\boldsymbol{t}(\boldsymbol{s})$ and G, the optimality of a decoder $\boldsymbol{r}(\boldsymbol{s})$ is independent of the choice.

Another important definition of a near-optimal decoder is the minimum-distance (MD) decoder. An MD decoder chooses the most probable physical error $\boldsymbol{e}^*(\boldsymbol{s})$ which satisfies

$$p_{\boldsymbol{e}^*(\boldsymbol{s})} \geqslant p_{\boldsymbol{e}} \forall \boldsymbol{e} \in \{\boldsymbol{e} | \boldsymbol{s}(\boldsymbol{e}) = \boldsymbol{e}\} \quad (15)$$

as a recovery operation. Though the maximally likelihood physical error $\boldsymbol{e}^*(\boldsymbol{s})$ does not necessarily satisfy the condition (13), it is empirically known that the MD decoder achieves near-optimal performance.

It is known that the MD decoder can be constructed efficiently in limited cases of the code and the error model. For example, we can construct an efficient MD decoder for the surface code under independent bit-flip and phase-flip errors. In this case, we can reduce the decoding problem into minimum-weight perfect matching (MWPM), which can be efficiently solved with the blossom algorithm [31]. When bit-flip and phase-flip errors are correlated, we can still construct a decoder with MWPM by ignoring the correlation, resulting in an suboptimal decoder. We call such a decoder a MWPM decoder.

D. Topological code

We consider two types of the topological codes in this article: surface codes and color codes. The qubit allocation of the surface code is shown in Fig. 1. The $[[2d^2 - 2d + 1, 1, d]]$ code and the $[[d^2, 1, d]]$ code are shown in Fig. 1(a) and 1(b), respectively. In both figures, the physical qubits are located on the vertices of the colored faces. Each red face represents a stabilizer operator which is a product of Pauli X operators on the physical qubits of its vertices. Each blue face represents one with Pauli Z operators.

The color codes consist of the lattice, which has three-colored faces: red, green, and blue. Two types of codes, the [4,8,8]-color code and the [6,6,6]-color code, are shown in Figs. 2(a) and 2(b), respectively. The physical qubits are also located on each vertex of the faces. Each colored face represents a stabilizer operator, including nontrivial Pauli operators for its vertices. The [4,8,8]-color code is a $[[\frac{1}{2}d^2 + d - \frac{1}{2}, 1, d]]$ code, and the [6,6,6]-color code is a $[[\frac{3}{4}d^2 + \frac{1}{4}, 1, d]]$ code.

In both surface codes and color codes, $O(d^2)$ qubits are used for representing single logical qubit. For a clear understanding of our notations here, we show a simple example with a $d = 3$ surface code in Appendix D.

E. Supervised machine learning

Supervised machine learning [32] is a branch of artificial intelligence that requires a training data set $\{(\boldsymbol{x}_1, \boldsymbol{y}_1), \dots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$ which consists of feature data $\boldsymbol{x}_i$ and its corresponding label data $\boldsymbol{y}_i$. Its aim is to prepare a model that takes the feature data as input and outputs an inferred label for it. The model has a predetermined structure and *trainable* parameters $\boldsymbol{\theta}$.

Unlike a simple dictionary, the model is expected to infer a label even for an unseen feature data. This is achieved by optimizing the model parameters $\boldsymbol{\theta}$ for the training data set. This process is commonly called *training*. Specifically, during its training, the difference between the output of the model $\boldsymbol{y}'$ to a feature and the correct label $\boldsymbol{y}$ is evaluated with a real-valued loss function $L(\boldsymbol{y}, \boldsymbol{y}')$. The loss is minimized if and only if the prediction is exactly the same as the correct label. The training data are used to optimize the model parameters $\boldsymbol{\theta}$ to reduce the loss. This can be done with standard optimization methods such as stochastic gradient descent:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \nabla_{\boldsymbol{\theta}} L, \quad (16)$$

where $\gamma \in \mathbb{R}$ is a learning rate and L is calculated for a randomly chosen subset, called a *batch*, of the training data set. As we can see here, it is required that the loss function should be differentiable, such as L2 distance $||\boldsymbol{y} - \boldsymbol{y}'||_2^2$. Once trained, we can apply the model to an unseen feature data

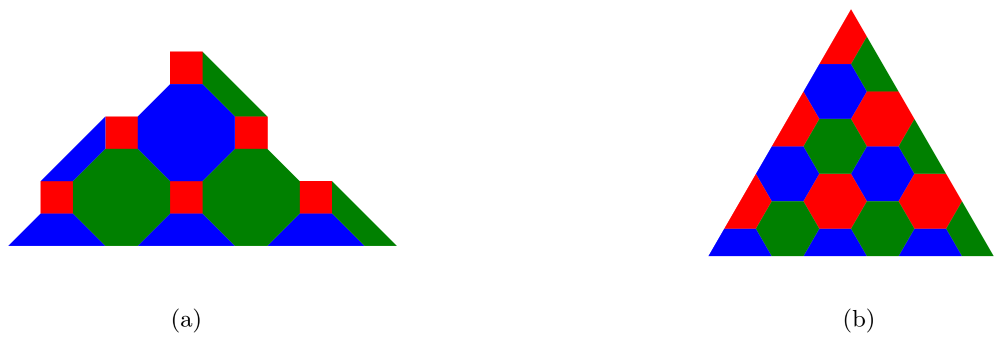


FIG. 2. The qubit allocation of the [4,8,8]-color code and the [6,6,6]-color code. Each vertex corresponds to a physical qubit, and each face corresponds to a stabilizer operator.

and obtain its predicted label with simple calculations of the network parameters and the input feature data.

An artificial neural network (ANN) [33] is a machine learning model inspired by neural structure found in nature. Here we assume that neurons are real-valued functions and a layer $\mathbf{h}$ is a vector of the neurons. Multilayer perceptron (MLP) is one of the simplest ANNs, which, as its name suggests, consists of multiple layers of neurons including the input and output layers. Here each neuron in a layer is connected to all neurons in the neighboring layers with trainable weights and biases, and yet completely independent of the other neurons in its own layer. Mathematically this can be described as

$$h_i^{(n)} = A\left(\sum_j W_{ij}^{(n,n-1)} h_j^{(n-1)} + b_i^{(n)}\right), \tag{17}$$

where A is a nonlinear activation function, $h_i^{(n)}$ is the ith neuron in the nth layer, $b_i^{(n)}$ is the bias added to the ith neuron in the nth layer, and $W_{ij}^{(n,n-1)}$ is the weight connecting the ith neuron in the nth layer to the jth neuron in the $(n-1)$th layer. We illustrate this in Fig. 3. Here the model parameters are the weights and biases. In this model the input information

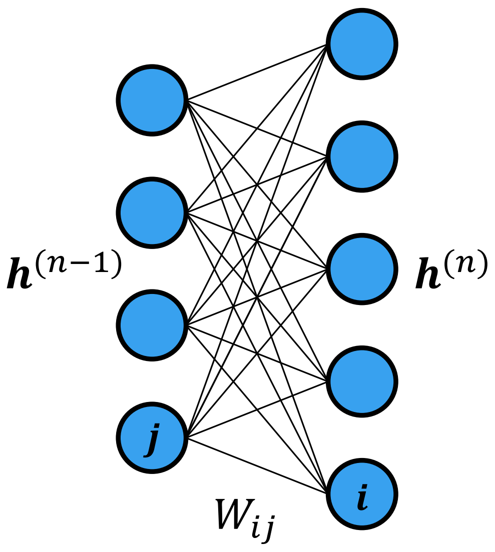


FIG. 3. Feed-forward network. The jth neuron in the $(n-1)$th layer is connected to the ith neuron in the n via weight W_{ij}.

propagates in forward direction, i.e., from the input nodes to the output nodes. At the output nodes, the loss value is calculated from the model output and the correct label. In order to update the model parameters, the gradient of the loss function $\nabla_{\boldsymbol{\theta}} L$ is evaluated with the back-propagation method. According to the universal approximation theorem [33], any continuous function can be approximated by an MLP model of a finite size, though its structure is simple and compact. Thus, we expect a neural decoder with an MLP model can achieve optimal performance under an appropriate training process.

On the other hand, as we numerically show in Secs. III D and IV C, we typically cannot achieve the exact optimal performance due to several reasons. Roughly stated, the reasons why the supervised learning underperforms an optimal one are as follows. First, the size of the sampled training data set is finite in practice. Second, even if we have an infinite data set, the representation power of an MPL model is limited by the memory of computing devices, and training results are suboptimal. Third, even when the first two problems were solved, the performance of the trained model is suboptimal since a time for the training process is finite. These three problems are ubiquitous and typically not avoidable in machine-learning tasks. Finally, if a problem reduction from quantum error correction to machine learning is wrong, i.e., inappropriate labels are employed in supervised machine learning, even the optimal prediction of machine learning cannot guarantee the optimal decoding. While the first three factors are matters of machine learning, the final factor is a problem about how we should relate decoding problems in quantum error correction to the task of supervised machine learning. Therefore we focus on the final point in this paper.

III. CONSTRUCTION OF TASKS OF MACHINE-LEARNING-BASED DECODERS

In general, achievable accuracy in machine learning with a given size of training data depends on the formulation of the prediction task. In order to construct a near-optimal neural decoder, it is vital to consider what is a preferable formulation of the prediction task. However, this point has not been discussed in a unified view in the existing methods [24–27]. In this section, we discuss how the decoding problem should be formulated as a task of machine learning in order to achieve near-optimal performance. To this end, we propose a general framework, which we call a *linear prediction framework*. The main concept of a linear prediction framework is to formalize the prediction target of the neural decoder as a binary vector linearly generated from a binary representation of physical error, which we call *diagnosis*. In this framework, we can analytically study the behavior of the neural decoder and can discuss the necessary conditions for achieving an optimal performance when unlimited training overhead is available. For example, a simple idea is to set a binary representation of physical error as a diagnosis. However, it later turns out that this setting cannot achieve optimal performance even if unlimited training overhead is employed. Setting a class of physical error $\boldsymbol{w}(\boldsymbol{e})$ introduced in Eq. (10) as a diagnosis is an example of achieving the optimal performance. Based on the discussion, we propose a criterion, *normalized sensitivity*, which should be optimized in defining the label for constructing a good decoder. We show specific constructions which minimize normalized sensitivity for the surface codes and the color codes, which we call *uniform data construction*. Then we numerically confirm that the performance of the neural decoder is improved with the construction. We also confirm that this construction is also applicable to the color codes.

A. Linear prediction framework

In order to discuss the behavior of the neural decoder in a unified view, we consider a neural decoder with the following two specifications. First, the neural decoder uses the syndrome vector $\boldsymbol{s}$ as the feature data to be fed to the trainable model. Second, the label data is a binary vector, and the correct label is linearly generated from the physical error vector $\boldsymbol{e}$ in GF(2). We call a linearly generated label vector $\boldsymbol{g}$ as a *diagnosis*, and a matrix H_g which generates the diagnosis $\boldsymbol{g} := H_g\Lambda\boldsymbol{e}^{\mathrm{T}}$ as a diagnosis matrix. We denote the length of the diagnosis vector $\boldsymbol{g}$ as L_g. The recovery operator $\boldsymbol{r}$ is calculated from the predicted diagnosis $\boldsymbol{g}$ and the syndrome $\boldsymbol{s}$. We use an assumed physical error distribution $\{p_{\boldsymbol{e}}\}$ only for generating a training data set $\{(\boldsymbol{s}_i, \boldsymbol{g}_i)\}$ and do not use it for constructing H_g or in the calculation of the recovery operator $\boldsymbol{r}$ from $\boldsymbol{g}$ and $\boldsymbol{s}$.

Though this framework restricts the label to be linearly generated from the physical error, this is general enough to formulate all the constructions described in the existing methods as special cases [24–28] with small technical exceptions. We will explain how this framework is related to the existing methods in Sec. III C.

Since the actual performance of the neural decoder depends on many factors such as configurations of the training process, the size of the training data set, and details of the network construction, we start with considering the problem under an ideal limit. We first consider the problem under the simple 0-1 loss function with an unlimited size of the training data set. Then we relax these impractical assumptions to practical ones. Though we numerically investigate the case of a single logical qubit ($k = 1$) later, we present the formalism for a general value of k.

1. The neural decoder with the 0-1 loss function and an unlimited training data set

We first consider a hypothetical decoder that can minimize any loss function with an unlimited number of the training data set. Though such an assumption is not practical, it is convenient to reveal the conditions for performing optimal decoding with machine learning in the ideal limit. We choose the 0-1 δ function $\delta(\boldsymbol{g}, \boldsymbol{g}')$ as the loss function, which is zero if the predicted and the correct diagnosis are the same and unity otherwise. Let us consider the portion of a training data set with a specific value of $\boldsymbol{s}$ with $\Pr_{\boldsymbol{e}\sim\{p_{\boldsymbol{e}}\}}[\boldsymbol{s}(\boldsymbol{e}) = \boldsymbol{s}] > 0$. If the neural decoder returns diagnosis $\boldsymbol{g}$ for the input $\boldsymbol{s}$, the total loss for this portion is proportional to the following value:

$$\begin{aligned} L_{\boldsymbol{s}}^{(\delta)}(\boldsymbol{g}) &:= \mathbb{E}_{\boldsymbol{e}\sim\{p_{\boldsymbol{e}}\}}[\delta(\boldsymbol{g}, H_g\Lambda\boldsymbol{e}^{\mathrm{T}})|\boldsymbol{s}(\boldsymbol{e}) = \boldsymbol{s}] \\ &= 1 - \Pr_{\boldsymbol{e}\sim\{p_{\boldsymbol{e}}\}}[H_g\Lambda\boldsymbol{e}^{\mathrm{T}} = \boldsymbol{g}|\boldsymbol{s}(\boldsymbol{e}) = \boldsymbol{s}]. \end{aligned} \tag{18}$$

Let $\boldsymbol{g}^{(\delta)}(\boldsymbol{s})$ be the output of the ideally trained neural decoder. Since it should minimize the total loss for every $\boldsymbol{s}$, it satisfies

$$L_{\boldsymbol{s}}^{(\delta)}(\boldsymbol{g}^{(\delta)}(\boldsymbol{s})) = \min_{\boldsymbol{g}} L_{\boldsymbol{s}}^{(\delta)}(\boldsymbol{g}). \tag{19}$$

We call this ideal decoder a *delta diagnosis decoder* and $\boldsymbol{g}^{(\delta)}(\boldsymbol{s})$ a *delta diagnosis vector*.

We show the condition for a diagnosis matrix H_g to guarantee that we can perform the optimal decoding with the delta diagnosis decoder. To this end, we define a property of the diagnosis matrix and introduce a set of diagnosis vectors as follows.

Definition III.1. Faithful diagnosis matrix—Given a check matrix H_c, we say the diagnosis matrix H_g is *faithful* if

$$\mathrm{span}(\{(H_{cg})_i\}) = b(\mathcal{L}), \tag{20}$$

or equivalently,

$$H_{cg}\Lambda\boldsymbol{e}^{\mathrm{T}} = 0 \leftrightarrow \boldsymbol{e} \in \mathcal{L}_0, \tag{21}$$

where

$$H_{cg} := \begin{pmatrix} H_c \\ H_g \end{pmatrix}. \tag{22}$$

Definition III.2. Faithful diagnosis vectors—Given a check matrix H_c, a pure error $\boldsymbol{t}(\boldsymbol{s})$, and a faithful diagnosis matrix H_g, we define 2^{2k} faithful diagnosis vectors $\{\boldsymbol{g}_{\boldsymbol{s}}(\boldsymbol{w})\}$ ($\boldsymbol{w} \in \{0, 1\}^{2k}$) associated with a syndrome vector $\boldsymbol{s}$ by

$$\boldsymbol{g}_{\boldsymbol{s}}(\boldsymbol{w}) := H_g\Lambda(\boldsymbol{w}G \oplus \boldsymbol{t}(\boldsymbol{s}))^{\mathrm{T}}. \tag{23}$$

Note that the faithful condition of H_g implies that

$$\boldsymbol{w} \mapsto \boldsymbol{g}_{\boldsymbol{s}}(\boldsymbol{w}) \tag{24}$$

is injective and

$$H_g\Lambda\boldsymbol{e}^{\mathrm{T}} = \boldsymbol{g}_{\boldsymbol{s}}(\boldsymbol{w}(\boldsymbol{e})), \tag{25}$$

with $\boldsymbol{s} = H_c\Lambda\boldsymbol{e}^{\mathrm{T}}$. As a result, when H_g is faithful, we have

$$1 - L_{\boldsymbol{s}}^{(\delta)}(\boldsymbol{g}) = \Pr_{\boldsymbol{e}\sim\{p_{\boldsymbol{e}}\}}[\boldsymbol{g}_{\boldsymbol{s}}(\boldsymbol{w}(\boldsymbol{e})) = \boldsymbol{g}|\boldsymbol{s}(\boldsymbol{e}) = \boldsymbol{s}] \tag{26}$$

from Eqs. (18) and (25). Then the injective property of $\boldsymbol{g}_{\boldsymbol{s}}(\boldsymbol{w})$ leads to

$$1 - L_{\boldsymbol{s}}^{(\delta)}(\boldsymbol{g}_{\boldsymbol{s}}(\boldsymbol{w})) = q_{\boldsymbol{s}}(\boldsymbol{w}), \tag{27}$$

where $q_{\boldsymbol{s}}(\boldsymbol{w})$ is defined in Eq. (12).

When the diagnosis matrix is faithful, we can construct an optimal decoder as follows. From Eqs. (19) and (27), we see that the delta diagnosis vector $\boldsymbol{g}^{(\delta)}(\boldsymbol{s})$ is one of the faithful diagnosis vectors. We can thus write it in the form

$$\boldsymbol{g}^{(\delta)}(\boldsymbol{s}) = \boldsymbol{g}_{\boldsymbol{s}}(\boldsymbol{w}^*(\boldsymbol{s})). \tag{28}$$

Equations (19), (27), and (28) imply that

$$\begin{aligned}1 - q_s(\boldsymbol{w}^*(\boldsymbol{s})) &= L_s^{(\delta)}(\boldsymbol{g}^{(\delta)}(\boldsymbol{s}))\\ &= \min_{\boldsymbol{w}\in\{0,1\}^{2k}} (1 - q_s(\boldsymbol{w}))\\ &= 1 - \max_{\boldsymbol{w}\in\{0,1\}^{2k}} q_s(\boldsymbol{w}).\end{aligned} \tag{29}$$

Since $\boldsymbol{g}_s(\boldsymbol{w})$ is injective, one can calculate $\boldsymbol{w}^*(\boldsymbol{s})$ from the diagnosis $\boldsymbol{g}^{(\delta)}(\boldsymbol{s})$ and syndrome $\boldsymbol{s}$. The recovery operator is then chosen as

$$\boldsymbol{r}(\boldsymbol{s}) = \boldsymbol{w}^*(\boldsymbol{s})G \oplus \boldsymbol{t}(\boldsymbol{s}). \tag{30}$$

For the optimality, we have

$$\begin{aligned}\Pr_{\boldsymbol{e}\sim\{p_{\boldsymbol{e}}\}}[\boldsymbol{e}\oplus\boldsymbol{r}(\boldsymbol{s}) \in \mathcal{L}_0 | \boldsymbol{s}(\boldsymbol{e}) = \boldsymbol{s}] &= q_s(\boldsymbol{w}^*(\boldsymbol{s}))\\ &= \max q_s(\boldsymbol{w})\end{aligned} \tag{31}$$

for any $\boldsymbol{s}$ with $\Pr_{\boldsymbol{e}\sim\{p_{\boldsymbol{e}}\}}[\boldsymbol{s}(\boldsymbol{e}) = \boldsymbol{s}] > 0$, which satisfies Eq. (13).

We can also prove a converse statement for the cases where H_g is not faithful (see Appendix A), arriving at the following lemma.

Lemma III.1. If the diagnosis matrix H_g is faithful, there exists a map $\boldsymbol{r}^*(\boldsymbol{g}, \boldsymbol{s})$ such that the decoder with $\boldsymbol{r}(\boldsymbol{s}) = \boldsymbol{r}^*(\boldsymbol{g}^{(\delta)}(\boldsymbol{s}), \boldsymbol{s})$ is optimal for arbitrary distribution $\{p_{\boldsymbol{e}}\}$. If the diagnosis matrix H_g is not faithful, no such map exists.

This lemma implies that we can perform optimal decoding with the delta diagnosis decoder only when the diagnosis matrix H_g is faithful. Note that the set of the faithful vectors $\{\boldsymbol{g}_s(\boldsymbol{w}) | \boldsymbol{w} \in \{0, 1\}^{2k}\}$ is independent of the choice of the generator G and the pure error $\boldsymbol{t}(\boldsymbol{s})$. Whether we can perform the optimal decoding or not is dependent only on the construction of H_g.

2. The neural decoder with the L2 loss function and an unlimited training data set

As discussed in Sec. II E, we cannot use the 0-1 loss function in practice since it is not differentiable. In this subsection, we replace the 0-1 loss function with a more practical one, which is the squared L2 distance. The loss of the model is chosen to be $|\boldsymbol{g} - H_g\Lambda\boldsymbol{e}^{\mathrm{T}}|^2$ instead of $\delta(\boldsymbol{g}, H_g\Lambda\boldsymbol{e}^{\mathrm{T}})$. We still consider the limit of an infinite size of the training data set and the perfect loss minimization. In this case, the total loss for a fixed $\boldsymbol{s}$ under an unlimited training data set is proportional to the following value:

$$L_s^{(\mathrm{L2})}(\boldsymbol{g}) = \mathbb{E}_{\boldsymbol{e}\sim\{p_{\boldsymbol{e}}\}}\big[||\boldsymbol{g} - H_g\Lambda\boldsymbol{e}^{\mathrm{T}}||_2^2 \big| \boldsymbol{s}(\boldsymbol{e}) = \boldsymbol{s}\big]. \tag{32}$$

We define a decoder which is ideally trained with the L2 loss function as an *L2 diagnosis decoder*. We also call the output of the L2 diagnosis decoder an *L2 diagnosis vector* $\boldsymbol{g}^{(\mathrm{L2})}(\boldsymbol{s})$. The L2 diagnosis vector satisfies the following equation:

$$L_s^{(L2)}(\boldsymbol{g}^{(\mathrm{L2})}(\boldsymbol{s})) = \min_{\boldsymbol{g}\in\{0,1\}^{L_g}} L_s^{(\mathrm{L2})}(\boldsymbol{g}). \tag{33}$$

When the chosen diagnosis matrix is faithful, we can analytically solve $\boldsymbol{g}^{(\mathrm{L2})}(\boldsymbol{s})$ by differentiating Eq. (32), and the L2 diagnosis vector can be written as follows:

$$\boldsymbol{g}^{(\mathrm{L2})}(\boldsymbol{s}) := \sum_{\boldsymbol{w}\in\{0,1\}^{2k}} q_s(\boldsymbol{w})\boldsymbol{g}_s(\boldsymbol{w}). \tag{34}$$

Let us define a column vector of order 2^{2k} as

$$\boldsymbol{q}_s := (q_s(0^{2k}), \ldots q_s(1^{2k}))^{\mathrm{T}}. \tag{35}$$

It satisfies the following matrix equation:

$$\begin{pmatrix}\hat{\boldsymbol{g}}^{(\mathrm{L2})}(\boldsymbol{s})\\ 1\end{pmatrix} = D_s\boldsymbol{q}_s, \tag{36}$$

where

$$D_s = \begin{pmatrix}\boldsymbol{g}_s(0^{2k}) & \cdots & \boldsymbol{g}_s(1^{2k})\\ 1 & \cdots & 1\end{pmatrix}. \tag{37}$$

We can solve it for $\boldsymbol{q}_s$ if D_s has a left inverse D_s^{-1} such that $D_s^{-1}D_s = I$ in the real-valued calculation, namely, if the rank of D_s as a real-valued matrix is 2^{2k}. If the rank is smaller, solution $\boldsymbol{q}_s$ is not unique, and hence it is not always possible to determine $\boldsymbol{w}$ that maximizes $q_s(\boldsymbol{w})$, which implies we cannot perform the optimal decoding.

Though the rank condition depends apparently on the syndrome $\boldsymbol{s}$, we can formulate it as a condition which is independent of $\boldsymbol{s}$. Any faithful diagnosis $\boldsymbol{g}_s(\boldsymbol{w})$ can be written as

$$\boldsymbol{g}_s(\boldsymbol{w}) = H_g\Lambda(\boldsymbol{w}G)^{\mathrm{T}} \oplus \boldsymbol{\delta}(\boldsymbol{s}) \tag{38}$$

with

$$\boldsymbol{\delta}(\boldsymbol{s}) := H_g\Lambda\boldsymbol{t}(\boldsymbol{s})^{\mathrm{T}} \in (\{0,1\}^{L_g})^{\mathrm{T}}. \tag{39}$$

We define a transformation $\sigma_{\boldsymbol{\delta}}$ by

$$(\sigma_{\boldsymbol{\delta}}(\boldsymbol{v}))_i := \delta_i + (-1)^{\delta_i}v_i \tag{40}$$

for $\boldsymbol{\delta} \in \{0,1\}^{2k}$ and $\boldsymbol{v} \in \mathbb{R}^{2k}$. It is affine, isometric, and involutory. Since $\boldsymbol{g}_s(\boldsymbol{w}) = \sigma_{\boldsymbol{\delta}(\boldsymbol{s})}(H_g\Lambda(\boldsymbol{w}G)^{\mathrm{T}})$, we have

$$D_s = \begin{pmatrix}\sigma_{\boldsymbol{\delta}(\boldsymbol{s})}(H_g\Lambda((0^{2k})G)^{\mathrm{T}}) & \cdots & \sigma_{\boldsymbol{\delta}(\boldsymbol{s})}(H_g\Lambda((1^{2k})G)^{\mathrm{T}})\\ 1 & \cdots & 1\end{pmatrix}. \tag{41}$$

We see that a transformation $\sigma_{\boldsymbol{\delta}}$ is an affine transformation, and this transformation satisfies

$$\sigma_{\boldsymbol{\delta}}(\sigma_{\boldsymbol{\delta}}(\boldsymbol{v})) = \boldsymbol{v}, \tag{42}$$

$$\sigma_0(\boldsymbol{v}) = \boldsymbol{v}. \tag{43}$$

Thus, when we apply the transformation $\sigma_{\boldsymbol{\delta}(\boldsymbol{s})}$ to Eq. (36), we obtain

$$\begin{pmatrix}\sigma_{\boldsymbol{\delta}}(\boldsymbol{g}^{(\mathrm{L2})}(\boldsymbol{s}))\\ 1\end{pmatrix} = D\boldsymbol{q}_s, \tag{44}$$

where

$$D := \begin{pmatrix}H_g\Lambda((0,\ldots,0)G)^{\mathrm{T}} & \cdots & H_g\Lambda((1,\ldots,1)G)^{\mathrm{T}}\\ 1 & \cdots & 1\end{pmatrix}. \tag{45}$$

Thus, we can uniquely calculate $\boldsymbol{q}_s$ for an arbitrary $\boldsymbol{s}$ if a matrix D has a left inverse, which is equivalent to the condition that $\{H_g\Lambda(\boldsymbol{w}G)^{\mathrm{T}} | \boldsymbol{w} \in \{0,1\}^{2k}\}$ is affinely independent. We will call a diagnosis matrix satisfying this condition decomposable:

Definition III.3. Decomposable diagnosis matrix—Given a generator matrix G, we say a diagnosis matrix H_g is decomposable if a set of real vectors $\{H_g\Lambda(\boldsymbol{w}G)^{\mathrm{T}} | \boldsymbol{w} \in \{0,1\}^{2k}\}$ is affinely independent, namely, the rank of a matrix D defined in Eq. (45) is 2^{2k} when we consider D a real-valued matrix.

When H_g is faithful, the above definition is independent of G, because the set $\{H_g\Lambda(\boldsymbol{w}G)^{\mathrm{T}}|\boldsymbol{w}\in\{0,1\}^{2k}\}$ is independent of G then.

We show a scheme to perform the optimal decoding using L2 diagnosis decoder when a diagnosis matrix is faithful and decomposable. When H_g is decomposable, there exists a left inverse D^{-1} such that $D^{-1}D=I$ in real vector space. When we observe a syndrome vector $\boldsymbol{s}$, we obtain the L2 diagnosis $\boldsymbol{g}^{(\mathrm{L2})}(\boldsymbol{s})$ using the trained L2 diagnosis decoder and calculate $\boldsymbol{\delta}(\boldsymbol{s})=H_g\Lambda\boldsymbol{t}(\boldsymbol{s})$. Since the diagnosis matrix is faithful, the probabilities of the faithful diagnosis vectors are given by

$$\boldsymbol{q}_s = D^{-1}\begin{pmatrix}\sigma_{\boldsymbol{\delta}(\boldsymbol{s})}(\boldsymbol{g}^{(\mathrm{L2})}(\boldsymbol{s}))\\ 1\end{pmatrix}. \tag{46}$$

Then we construct a recovery operator as

$$\boldsymbol{r}(\boldsymbol{s})=\boldsymbol{w}^*(\boldsymbol{s})G\oplus\boldsymbol{t}(\boldsymbol{s}), \tag{47}$$

where $\boldsymbol{w}^*(\boldsymbol{s})$ satisfies

$$q_s(\boldsymbol{w}^*(\boldsymbol{s}))=\max_{\boldsymbol{w}} q_s(\boldsymbol{w}). \tag{48}$$

With this recovery operator, we obtain

$$\mathrm{Pr}_{\boldsymbol{e}\sim\{p_{\boldsymbol{e}}\}}[\boldsymbol{e}\oplus\boldsymbol{r}(\boldsymbol{s})\in\mathcal{L}_0|\boldsymbol{s}(\boldsymbol{e})=\boldsymbol{s}]=q_s(\boldsymbol{w}^*(\boldsymbol{s})), \tag{49}$$

and thus this decoder satisfies Eq. (13).

When the diagnosis matrix H_g is faithful, we can also prove a converse statement for the cases where a faithful diagnosis matrix H_g is not decomposable (see Appendix A), arriving at the following lemma.

Lemma III.2. If the diagnosis matrix H_g is faithful and decomposable, there exists a map $\boldsymbol{r}^*(\boldsymbol{g},\boldsymbol{s})$ such that the decoder with $\boldsymbol{r}(\boldsymbol{s})=\boldsymbol{r}^*(\boldsymbol{g}^{(\mathrm{L2})}(\boldsymbol{s}),\boldsymbol{s})$ is optimal for arbitrary distribution $\{p_{\boldsymbol{e}}\}$. If the diagnosis matrix H_g is faithful but not decomposable, no such map exists.

We show a simple example of a faithful and decomposable matrix H_g in the case of $k=1$. We define two binary vectors $\boldsymbol{l}_{01}$ and $\boldsymbol{l}_{10}$ as

$$G=\begin{pmatrix}\boldsymbol{l}_{01}\\ \boldsymbol{l}_{10}\end{pmatrix}, \tag{50}$$

and define $\boldsymbol{l}_{11}=\boldsymbol{l}_{01}\oplus\boldsymbol{l}_{10}$. Then we construct H_g as

$$H_g=\begin{pmatrix}\boldsymbol{l}_{01}\\ \boldsymbol{l}_{10}\\ \boldsymbol{l}_{11}\end{pmatrix}. \tag{51}$$

We see that $\mathrm{span}(\{(H_g)_i\})=b(\mathcal{L})$, and thus H_g is faithful. A set $\{H_g\Lambda(\boldsymbol{w}G)^{\mathrm{T}}|\boldsymbol{w}\in\{00,01,10,11\}\}$ is

$$\{(0,0,0)^{\mathrm{T}},(0,1,1)^{\mathrm{T}},(1,0,1)^{\mathrm{T}},(1,1,0)^{\mathrm{T}}\}, \tag{52}$$

which is affinely independent, and thus H_g is decomposable. We can verify the same by checking the rank of

$$D=\begin{pmatrix}\boldsymbol{g}(00) & \boldsymbol{g}(01) & \boldsymbol{g}(10) & \boldsymbol{g}(11)\\ 1 & 1 & 1 & 1\end{pmatrix} = \begin{pmatrix}0&0&1&1\\0&1&0&1\\0&1&1&0\\1&1&1&1\end{pmatrix} \tag{53}$$

to be 4 in real vector space. We can show that there always exists such a faithful and decomposable diagnosis matrix for all k and H_c. See Appendix A for the proof.

3. The neural decoder with the L2 loss function under a finite training data size

In practical cases, the size of the training data set is limited, and hence the loss is not perfectly minimized. This implies that the output diagnosis from the model deviates from the L2 diagnosis vector. In such a case, it is desirable to construct a decoder such that its prediction is as robust against the deviations as possible. We introduce a slight modification to the optimal decoding scheme in the last subsection, so that it should applicable to an output diagnosis deviated from the L2 diagnosis vector.

We denote the predicted diagnosis as $\boldsymbol{g}^{\mathrm{P}}(\boldsymbol{s})\in\mathbb{R}^{L_g}$, which deviates from the L2 diagnosis vector. Note that $\boldsymbol{g}^{\mathrm{P}}(\boldsymbol{s})$ cannot be represented as a linear combination of the faithful diagnosis vectors in general. In order to construct a decoding scheme which is robust to a small deviation, it is natural to extend the scheme employed in Sec. III A 2 such that we project $\boldsymbol{g}^{\mathrm{P}}(\boldsymbol{s})$ to the hyperplane formed by affine combinations of the faithful diagnosis vectors, and then extract the coefficients $\boldsymbol{q}_s^{\mathrm{P}}$ from the projected point. This projection and extraction is achieved as follows. We perform QR decomposition [34] for D and obtain $D=QR$, where Q is an orthogonal matrix, and R is an upper-triangular matrix. We construct $D^{-1}=R^{-1}Q^{\mathrm{T}}$, which satisfies $D^{-1}D=I$. Then we obtain a predicted vector $\boldsymbol{q}_s^{\mathrm{P}}$ as

$$\boldsymbol{q}_s^{\mathrm{P}}=D^{-1}\begin{pmatrix}\sigma_{\boldsymbol{\delta}(\boldsymbol{s})}(\boldsymbol{g}^{\mathrm{P}}(\boldsymbol{s}))\\ 1\end{pmatrix}, \tag{54}$$

where $\boldsymbol{\delta}(\boldsymbol{s})=H_g\Lambda\boldsymbol{t}(\boldsymbol{s})$. We construct a recovery operator as

$$\boldsymbol{r}(\boldsymbol{s})=\boldsymbol{w}^*(\boldsymbol{s})G\oplus\boldsymbol{t}(\boldsymbol{s}), \tag{55}$$

where $\boldsymbol{w}^*(\boldsymbol{s})$ satisfies

$$q_s^{\mathrm{P}}(\boldsymbol{w}^*(\boldsymbol{s}))=\max_{\boldsymbol{w}} q_s^{\mathrm{P}}(\boldsymbol{w}). \tag{56}$$

Note that though elements of $\boldsymbol{q}_s^{\mathrm{P}}$ may be out of [0,1], the above procedure is still well defined.

4. Criterion for diagnosis matrix

In practice, the number of the training data set is far smaller than the total variation of syndrome vectors $\boldsymbol{s}$ when distance d is larger than about 7. For example, according to the existing methods [24–27], the size of the training data set is at most 10^9. On the other hand, the number of variations in the syndrome, 2^{n-k}, becomes larger than 10^9 at the distance $d=7$ for the $[[d^2,1,d]]$ surface code. This implies that almost all the patterns of the syndrome vector $\boldsymbol{s}$ given in experiments are not found in the training data set. The model should infer the L2 diagnosis vector $\boldsymbol{g}^{(\mathrm{L2})}(\boldsymbol{s})$ of $\boldsymbol{s}$ where $\boldsymbol{s}$ is not included in the training data set. The aim of this subsection is to propose a criterion for H_g which we believe to reflect the robustness of the prediction when we use such a sparsely sampled training data set.

Since the problem is to estimate the vector-valued function $\boldsymbol{g}^{(\mathrm{L2})}(\boldsymbol{s})$ from a sparsely sampled set of values, its difficulty should depend on how rapidly the function changes its output value as the input value $\boldsymbol{s}$ varies. From Eqs. (25) and (34), we see that the function is written as

$$\boldsymbol{g}^{(\mathrm{L2})}(\boldsymbol{s}) = \mathbb{E}_{\boldsymbol{e}\sim\{p_e\}}[H_g\Lambda\boldsymbol{e}^{\mathrm{T}}|\boldsymbol{s}(\boldsymbol{e})=\boldsymbol{s}], \tag{57}$$

which shows that $\boldsymbol{g}^{(\mathrm{L2})}(\boldsymbol{s})$ is implicitly determined from the two functions of errors, $\boldsymbol{g}(\boldsymbol{e}) = H_g\Lambda\boldsymbol{e}^{\mathrm{T}}$ and $\boldsymbol{s}(\boldsymbol{e}) = H_c\Lambda\boldsymbol{e}^{\mathrm{T}}$. In order to quantify how rapidly these functions change, let us introduce a sensitivity $m(H)$ of a binary matrix H as

$$\begin{aligned} m(H) &:= \max_{\substack{\boldsymbol{e},\boldsymbol{e}'\in\{0,1\}^{2n}\\ h(\boldsymbol{e}\oplus\boldsymbol{e}')=1}} ||H\Lambda\boldsymbol{e}^{\mathrm{T}} - H\Lambda\boldsymbol{e}'^{\mathrm{T}}||_2^2 \\ &= \max_{\substack{\boldsymbol{e}\in\{0,1\}^{2n}\\ h(\boldsymbol{e})=1}} h(H\Lambda\boldsymbol{e}^{\mathrm{T}}). \end{aligned} \tag{58}$$

Using the sensitivity, the variation of $\boldsymbol{s}(\boldsymbol{e})$ is bounded as

$$||\boldsymbol{s}(\boldsymbol{e}) - \boldsymbol{s}(\boldsymbol{e}')||_2^2 \leqslant m(H_c)h(\boldsymbol{e}\oplus\boldsymbol{e}'). \tag{59}$$

In the case of topological codes, $m(H_c)$ is a small constant, because each physical qubit is monitored by at most a constant number of the stabilizer operators.

Suppose that $\boldsymbol{g}^{(\mathrm{L2})}(\boldsymbol{s})$ is close to one of the faithful diagnosis $\boldsymbol{g}_s(\boldsymbol{w}^*)$, and let $S(\boldsymbol{s},\boldsymbol{w}^*;0)$ be the set of errors $\boldsymbol{e}$ satisfying $\boldsymbol{w}(\boldsymbol{e}) = \boldsymbol{w}^*$ and $\boldsymbol{s}(\boldsymbol{e}) = \boldsymbol{s}$. We further define a set

$$S(\boldsymbol{s},\boldsymbol{w}^*;h) := \{\boldsymbol{e}|\exists\boldsymbol{e}'\ \text{s.t.}\ \boldsymbol{e}'\in S(\boldsymbol{s},\boldsymbol{w}^*;0),\ h(\boldsymbol{e}\oplus\boldsymbol{e}')\leqslant h\}. \tag{60}$$

We see that any $\boldsymbol{e}\in S(\boldsymbol{s},\boldsymbol{w}^*;h)$ produces a training data $(\boldsymbol{s}',\boldsymbol{g}')$ such that

$$||\boldsymbol{s}'-\boldsymbol{s}||_2^2 \leqslant m(H_c)h, \tag{61}$$

$$||\boldsymbol{g}'-\boldsymbol{g}_s(\boldsymbol{w}^*)||_2^2 \leqslant m(H_g)h. \tag{62}$$

The choice of H_g also affects how precisely $\boldsymbol{g}^{(\mathrm{L2})}(\boldsymbol{s})$ should be estimated in order to determine $\boldsymbol{w}^*$ correctly. To quantify this, we consider how far $\boldsymbol{g}^{\mathrm{P}}(\boldsymbol{s})$ can deviate from a faithful diagnosis $\boldsymbol{g}_s(\boldsymbol{w})$ without affecting the decoding method of Eqs. (54) and (55). When the decoding result changes from $\boldsymbol{w}^*=\boldsymbol{w}$ to $\boldsymbol{w}^*=\boldsymbol{w}'$, the solution of Eq. (54) should satisfy $q_s^{\mathrm{P}}(\boldsymbol{w}) = q_s^{\mathrm{P}}(\boldsymbol{w}')$, namely, $\boldsymbol{g}^{\mathrm{P}}(\boldsymbol{s})$ should be written in the form

$$\boldsymbol{g}^{\mathrm{P}}(\boldsymbol{s}) = \alpha(\boldsymbol{g}_s(\boldsymbol{w}) + \boldsymbol{g}_s(\boldsymbol{w}')) + \sum_{\boldsymbol{w}''\neq\boldsymbol{w},\boldsymbol{w}'} \beta_{\boldsymbol{w}''}\boldsymbol{g}_s(\boldsymbol{w}''). \tag{63}$$

We define the minimum boundary distance $M(H_g)$ so as to ensure that $\boldsymbol{w}^*=\boldsymbol{w}$ as long as $||\boldsymbol{g}^{\mathrm{P}}(\boldsymbol{s}) - \boldsymbol{g}_s(\boldsymbol{w})||_2^2 \leqslant M(H_g)$. Hence $M(H_g)$ can be explicitly defined as

$$M(H_g) := \min_{\boldsymbol{w},\boldsymbol{w}',\alpha,\{\beta_{\boldsymbol{w}''}\}} \left|\left|(1-\alpha)\boldsymbol{g}(\boldsymbol{w}) - \alpha\boldsymbol{g}(\boldsymbol{w}') - \sum_{\boldsymbol{w}''\neq\boldsymbol{w},\boldsymbol{w}'} \beta_{\boldsymbol{w}''}\boldsymbol{g}(\boldsymbol{w}'')\right|\right|_2^2. \tag{64}$$

Note that the above definition is independent of $\boldsymbol{s}$, since the affine transformation $\sigma_{\delta(s)}$ is isometric. $M(H_g)$ is nonzero if and only if H_g is decomposable.

Regarding $M(H_g)$ as the relevant length scale, we define the following quantity to be used as a criterion for a better construction of H_g.

Definition III.4. Normalized sensitivity—We define normalized sensitivity $N(H_g)$ of a faithful and decomposable matrix H_g as

$$N(H_g) := \frac{m(H_g)}{M(H_g)}, \tag{65}$$

where $m(H_g)$ is a sensitivity of H_g defined in Eq. (58), and $M(H_g)$ is a minimum boundary distance of H_g defined in Eq. (64).

Equations (62) and (64) imply that an error belonging to $S(\boldsymbol{s},\boldsymbol{w}^*;h)$ with $h\sim(m(H_g)/M(H_g))^{-1}$ leads to training data useful for estimation of $\boldsymbol{g}^{(\mathrm{L2})}(\boldsymbol{s})$. We thus expect that the use of a diagnosis matrix H_g with a small normalized sensitivity $N(H_g)$ enables high-performance prediction with a small training data set.

5. Uniform data construction

We propose specific constructions which minimize the normalized sensitivity up to the order of d in the case of $k=1$. We first consider a lower bound of the normalized sensitivity. When a diagnosis matrix H_g is faithful, each row vector of H_g corresponds to a logical operator or a stabilizer operator. We denote the number of the logical operators in the rows of H_g as n_L. The minimum boundary distance $M(H_g)$ is upper bounded by

$$M(H_g) \leqslant \frac{n_L}{4}. \tag{66}$$

Since any logical operator has at least d of one-elements in its binary representation, there are at least dn_L of one-elements in the diagnosis matrix. By denoting the number of the one-elements in the diagnosis matrix H_g as $\chi(H_g)$, we have

$$dn_L \leqslant \chi(H_g). \tag{67}$$

Since there are $2n$ columns in H_g, we also have

$$\chi(H_g) \leqslant 2n\max_i h\big((H_g^{\mathrm{T}})_i\big). \tag{68}$$

The sensitivity $m(H_g)$ is equal to the maximum Hamming weight of the column vectors of the diagnosis matrix, namely,

$$\max_i h\big((H_g)_i^{\mathrm{T}}\big) = m(H_g). \tag{69}$$

From Eqs. (66)–(69) we obtain

$$N(H_g) \geqslant \frac{2d}{n}. \tag{70}$$

In particular, when we focus on the two-dimensional topological codes such that $n=\Theta(d^2)$, the order of the normalized sensitivity is lower bounded as

$$N(H_g) = \Omega(d^{-1}), \tag{71}$$

where Θ and Ω are Landau symbols indicating exact order and lower bound, respectively. For surface codes and color codes with the single logical qubit, we found specific constructions of H_g such that $N(H_g)$ scales as $\Theta(d^{-1})$. See Appendix C for the specific constructions. We named these constructions the *uniform data construction* of the data set, since logical operators corresponding to the rows of H_g are chosen uniformly to cover all the physical qubits.

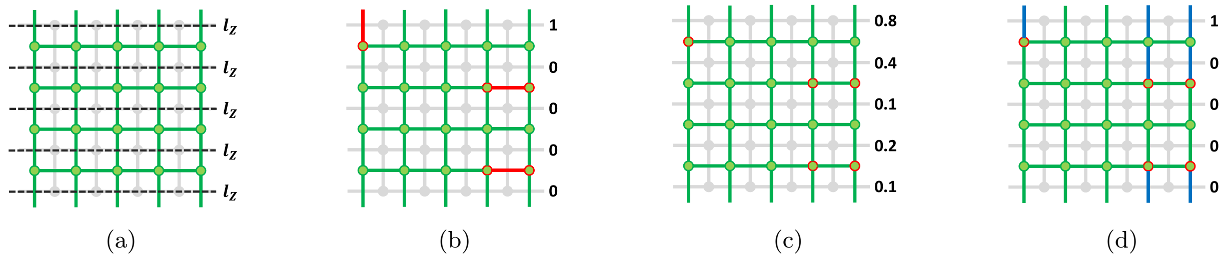


FIG. 4. The decoding process based on the proposed scheme. Each picture shows only the Z lattice, of which the edge corresponds to whether there is a bit-flip error on the physical qubit or not, and the circle shows whether an error is detected through the syndrome measurement. (a) Five logical Z operators which minimize the normalized sensitivity $\frac{m(H_g)}{M(H_g)}$. (b) The actual physical error is drawn as red edges, and the detected syndromes as red circles. The binary numbers shown to the right are the diagnosis vector of the physical errors. The neural network learns the relation between the location of the detected syndromes and the diagnosis vector. (c) The real-valued diagnosis vector is predicted by the neural decoder. (d) With the syndrome pattern, the faithful diagnosis vector is either 10000 or 01111. The chosen faithful diagnosis vector is 10000. Accordingly, we choose the recovery operator shown. In this case, the decoding succeeds.

B. Construction of data set and example

Let us summarize the discussion in Sec. III A. Given the check matrix H_c of the code and the error model $\{p_e\}$, the whole protocol can be described as follows:

Preparation: We construct a faithful and decomposable diagnosis matrix H_g with a small normalized sensitivity, possibly $N(H_g) = \Theta(d/n)$. We choose a pure error $\boldsymbol{t}(\boldsymbol{s})$ and a generator matrix G. We perform QR decomposition to a matrix

$$D := \begin{pmatrix} \boldsymbol{g}(0^{2k}) & \cdots & \boldsymbol{g}(1^{2k}) \\ 1 & \cdots & 1 \end{pmatrix}, \tag{72}$$

where $\boldsymbol{g}(\boldsymbol{w}) = H_g\Lambda(\boldsymbol{w}G)^{\mathrm{T}}$, and obtain Q and R. We calculate the left inverse matrix D^{-1} as

$$D^{-1} := R^{-1}Q^{\mathrm{T}}. \tag{73}$$

Data generation: We generate a set of physical errors $\{\boldsymbol{e}_0, \boldsymbol{e}_1, \ldots\}$ with the probability distribution $\{p_e\}$, and generate data set $\{(\boldsymbol{s}_0, \boldsymbol{g}_0), (\boldsymbol{s}_1, \boldsymbol{g}_1), \ldots\}$ from it, where $\boldsymbol{s}_i := H_c\Lambda\boldsymbol{e}_i^{\mathrm{T}}$ and $\boldsymbol{g}_i := H_g\Lambda\boldsymbol{e}_i^{\mathrm{T}}$.

Training: The model is trained so that it can predict $\boldsymbol{g}$ from $\boldsymbol{s}$. The loss of the prediction is defined as the L2 distance between $\boldsymbol{g}$ and $\boldsymbol{g}^{\mathrm{P}}$, where $\boldsymbol{g}^{\mathrm{P}}$ is a real-valued output vector of the model.

Prediction: When an observed syndrome $\boldsymbol{s}$ is given to the trained model, it predicts $\boldsymbol{g}^{\mathrm{P}}(\boldsymbol{s})$. In parallel, we calculate $\boldsymbol{\delta}(\boldsymbol{s})$ given by

$$\boldsymbol{\delta}(\boldsymbol{s}) = H_g\Lambda\boldsymbol{t}(\boldsymbol{s})^{\mathrm{T}}. \tag{74}$$

We calculate vector $\boldsymbol{q}_s$ defined in Eq. (35) as

$$\boldsymbol{q}_s^{\mathrm{P}} = D^{-1}\begin{pmatrix} \sigma_{\delta(s)}(\boldsymbol{g}^{\mathrm{P}}(\boldsymbol{s})) \\ 1 \end{pmatrix}, \tag{75}$$

where $\sigma_{\delta(s)}$ is an affine transformation such that

$$(\sigma_{\delta(s)}(\boldsymbol{v}))_i = \delta_i + (-1)^{\delta_i}v_i. \tag{76}$$

We choose $\boldsymbol{w}^{\mathrm{P}}$ that satisfies

$$q_s^{\mathrm{P}}(\boldsymbol{w}^{\mathrm{P}}) = \max_{\boldsymbol{w}\in\{0,1\}^{2k}} q_s^{\mathrm{P}}(\boldsymbol{w}), \tag{77}$$

where $\{q_s^{\mathrm{P}}(\boldsymbol{w})\}$ is the elements of $\boldsymbol{q}_s^{\mathrm{P}}$. Then we obtain an estimated recovery operator

$$\boldsymbol{r}(\boldsymbol{s}) = \boldsymbol{w}^{\mathrm{P}}G \oplus \boldsymbol{t}(\boldsymbol{s}). \tag{78}$$

We emphasize that the choice of $\boldsymbol{t}(\boldsymbol{s})$ and G does not affect the performance of the decoder, since the success of the estimation is independent of them. Only the construction of H_g affects the performance of the decoder.

We show a specific example of the decoding scheme. For simplicity, we consider the case where there are only bit-flip errors in the $[[2d^2 - 2d + 1, 1, d]]$ surface code. In this case, it is enough for QEC to consider the stabilizer operator with Pauli Z operators. The simplified picture of the code is shown in Fig. 4.

In this picture, a bit-flip error on a physical qubit is represented by the color of the corresponding edge (green: no error, red: error), and the syndrome value is represented by the color of the circle (green: undetected, red: detected). As shown in Fig. 4(a), the matrix H_g is constructed with logical operators, each of which is the product of the Pauli Z operators on the edges crossing the dotted line. In this case, we see $M(H_g) = O(d)$, $m(H_g) = O(1)$, and $\frac{m(H_g)}{M(H_g)} = O(d^{-1})$.

Suppose that bit-flip errors occur on a set of the physical qubits as shown in Fig. 4(b). The physical error is detected with the syndrome values as shown in the figure. The diagnosis vector is calculated as the commutation relation of the chosen logical operators and the physical error. We show the calculated diagnosis on the right side of the lattice. In the training phase, the model learns the relation between the positions of the red circles and the values of the diagnosis vector. In the prediction phase, only the positions of the red circles are given. The trained neural network outputs a real-valued prediction of the diagnosis vector as shown in Fig. 4(c), for example. From this information, we extract the probabilities of the faithful diagnosis, and we choose the faithful diagnosis which is expected to be the most probable, as shown in Fig. 4(d). Since the chosen diagnosis vector is equivalent to the diagnosis vector generated by the actual physical error, this decoding trial is a success.

C. Relation to existing methods

In this subsection, we explain how existing methods [24–27] can be treated in the linear prediction framework. The method proposed by Varsamopoulos *et al.* [25] used an approach similar to the example shown in Sec. III A 2 in the case of $k = 1$. In this method, a linear map is used for the pure error, which is called a simple decoder. The pure error is then written in the form $\boldsymbol{t}(\boldsymbol{s})^{\mathrm{T}} = T\boldsymbol{s}$, where T is a $2n \times (n-k)$ matrix satisfying $H_c\Lambda T = I$. The label vector used in this method can essentially be regarded as being generated by a diagnosis matrix defined by

$$H_g = \begin{pmatrix} \boldsymbol{l}_{01} \\ \boldsymbol{l}_{10} \\ \boldsymbol{l}_{11} \end{pmatrix} (I \oplus \Lambda T H_c). \tag{79}$$

We see this is faithful and decomposable construction. Let a generator matrix G be

$$G = \begin{pmatrix} \boldsymbol{l}_{01} \\ \boldsymbol{l}_{10} \end{pmatrix}. \tag{80}$$

Then a diagnosis generated from the diagnosis matrix is

$$\boldsymbol{g} = H_g\Lambda\boldsymbol{e} = \begin{pmatrix} w(\boldsymbol{e})_1 \\ w(\boldsymbol{e})_0 \\ w(\boldsymbol{e})_0 \oplus w(\boldsymbol{e})_1 \end{pmatrix}, \tag{81}$$

where $\boldsymbol{w}(\boldsymbol{e}) = (w(\boldsymbol{e})_0, w(\boldsymbol{e})_1)$. The method in Ref. [25] uses a different set of label vectors $\boldsymbol{g}'$ called one-hot representation, which has a one-to-one correspondence with $\boldsymbol{g}$ as

$$\boldsymbol{g} = (0, 0, 0)^{\mathrm{T}} \mapsto \boldsymbol{g}' = (1, 0, 0, 0)^{\mathrm{T}}, \tag{82}$$

$$\boldsymbol{g} = (0, 1, 1)^{\mathrm{T}} \mapsto \boldsymbol{g}' = (0, 1, 0, 0)^{\mathrm{T}}, \tag{83}$$

$$\boldsymbol{g} = (1, 0, 1)^{\mathrm{T}} \mapsto \boldsymbol{g}' = (0, 0, 1, 0)^{\mathrm{T}}, \tag{84}$$

$$\boldsymbol{g} = (1, 1, 0)^{\mathrm{T}} \mapsto \boldsymbol{g}' = (0, 0, 0, 1)^{\mathrm{T}}. \tag{85}$$

The above relation as real vectors can be written as

$$\boldsymbol{g}' = \frac{1}{2}\begin{pmatrix} -1 & -1 & -1 & 1 \\ -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \end{pmatrix}\begin{pmatrix} \boldsymbol{g} \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{86}$$

Since it is an isometric affine transformation, we expect that this transformation has little effect on the performance of the supervised machine learning. The matrix H_g is faithful and decomposable, but its normalized sensitivity is $O(1)$. We thus expect that this decoder becomes near-optimal when the training is ideally performed, but the prediction is not robust when the size of the training data set is small.

The method proposed by Baireuther *et al.* [26] mainly focuses on a model applicable to quantum error correction when we perform various counts of repetitive stabilizer measurements by utilizing recurrent neural network. They use the commutation relation between the physical error and a logical Z operator as the label, since they are concerned only about the logical bit-flip probability with the fixed initial state in the logical space. We can thus consider this method as a case of the linear prediction framework.

Torlai *et al.* [24], Krastanov *et al.* [27], and Breuckmann *et al.* [28] took a different approach from the above two [25,26]. They used the binary representation of the physical error as the label vector. In the linear prediction framework, it corresponds to a choice of $H_g = \Lambda$ leading to

$$\boldsymbol{g} = H_g\Lambda\boldsymbol{e}^{\mathrm{T}} = \boldsymbol{e}^{\mathrm{T}}. \tag{87}$$

Since H_g is not faithful, it cannot constitute an optimal decoder even with the delta diagnosis decoder. Interestingly, the delta diagnosis decoder with this choice of H_g works as an MD decoder, which can be shown by the following lemma.

Lemma III.3. If the matrix H_{cg} has rank $2n$ in GF(2), there exists a map $\boldsymbol{r}^*(\boldsymbol{g}, \boldsymbol{s})$ such that the decoder with $\boldsymbol{r}(\boldsymbol{s}) = \boldsymbol{r}^*(\boldsymbol{g}^{(\delta)}(\boldsymbol{s}), \boldsymbol{s})$ works as an MD decoder for arbitrary distribution $\{p_{\boldsymbol{e}}\}$. If H_{cg} does not have rank $2n$, no such map exists.

Proof. If H_{cg} has rank $2n$, there exists a left inverse binary matrix H_{cg}^{-1} such that $H_{cg}^{-1}H_{cg} = I$. Then we can obtain the physical error $\boldsymbol{e}$ as

$$\Lambda H_{cg}^{-1}\begin{pmatrix} \boldsymbol{s} \\ \boldsymbol{g} \end{pmatrix} = \boldsymbol{e}^T. \tag{88}$$

Thus, we can obtain the most probable physical error $\boldsymbol{e}^*(\boldsymbol{s})$ from the most probable diagnosis.

If H_{cg} does not have rank $2n$ in GF(2), there exist two physical errors which generate the same pair of syndrome and diagnosis. We cannot determine which is more probable. Thus, we cannot perform MD decoding when H_{cg} does not have rank $2n$.

A drawback in this approach is the difficulty arising when we replace a loss function with a practical one such as the L2 distance. In order to satisfy a decomposable property in MD decoding, the length of the diagnosis must be no shorter than 2^{n+k} since there are 2^{n+k} possible candidates of the most probable physical error. This is not practical when the distance is large, and thus it requires heuristics such as repetitive sampling.

D. Numerical result

We numerically show that the uniform data construction improves the performance of the neural decoder in the case of $k = 1$. We trained an MLP model with the uniform data construction and compare it with other data constructions of the neural decoders. We also make a comparison with known decoders such as the MD decoder and the MWPM decoder. We choose the $[[d^2, 1, d]]$ surface code for the comparison, since most of the existing methods were benchmarked with this code. We calculated the performance for two types of error models, the bit-flip noise and depolarizing noise. The probability distribution of the bit-flip noise is described as follows:

$$p_{\boldsymbol{e}} = \begin{cases} p^{w(\boldsymbol{e})}(1-p)^{n-w(\boldsymbol{e})} & \forall i > n, e_i = 0 \\ 0 & \text{otherwise} \end{cases}, \tag{89}$$

where p is an error probability per physical qubit, and $w(\boldsymbol{e})$ is a weight of physical error $\boldsymbol{e}$ defined in Eq. (5). The probability distribution of the depolarizing noise is described as follows:

$$p_{\boldsymbol{e}} = (p/3)^{w(\boldsymbol{e})}(1-p)^{n-w(\boldsymbol{e})}. \tag{90}$$

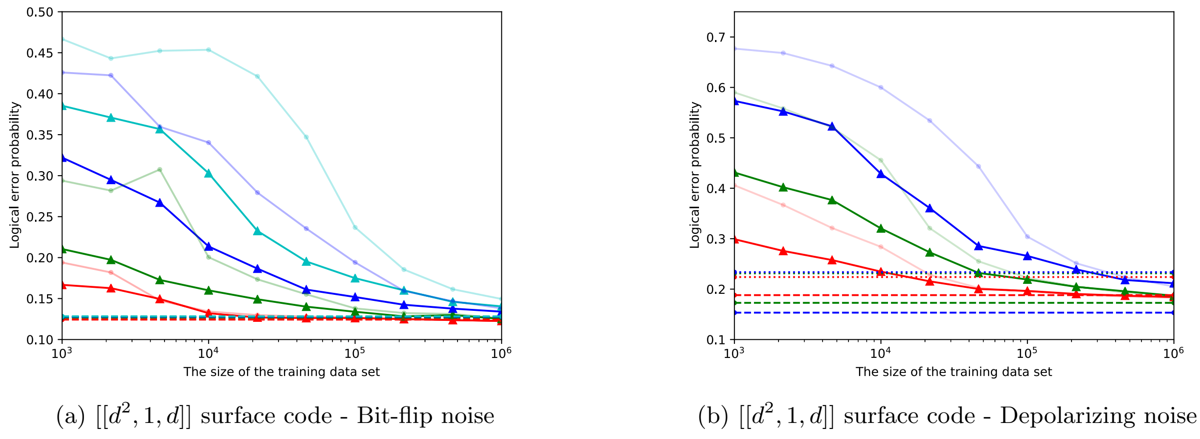


FIG. 5. The performance comparison between the neural decoder with the uniform construction (solid lines) and that with short diagnosis construction (pale lines), the MD decoder (dashed lines), and the MWPM decoder (dotted lines) in the case of the $[[d^2, 1, d]]$ surface code. The logical error probabilities are plotted against of the sizes of the training data set with the fixed physical error probability p. We calculated the performance for distances $d = 5$ (red), 7 (green), 9 (blue), and 11 (cyan). (a) The case for the bit-flip noise with $p = 0.1$. Note that there are no lines of MWPM decoder since the MWPM decoder is equivalent to the MD decoder in this setting. (b) The case for the depolarizing noise with $p = 0.15$.

Note that the occurrences of the bit-flip and phase-flip errors are correlated in the depolarizing noise. We first calculated the performance when the physical error probability is around the error threshold, namely, $p = 0.1$ for the bit-flip noise and $p = 0.15$ for the depolarizing noise. The tunable hyperparameters of the neural network, such as number of layers in network, number of neurons in each layer, and learning rate, are optimized with a grid search for each noise model and for each size of the training data set. See Appendix B for the details of the parameter optimization and implementation.

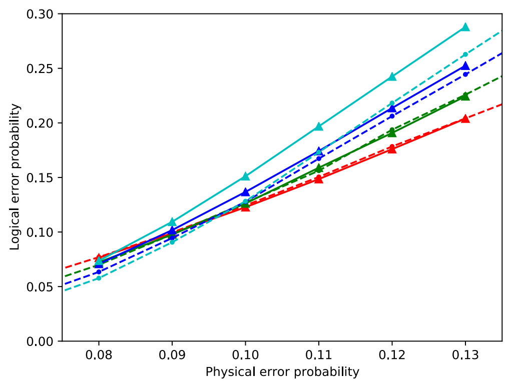
The performance of the neural decoder under the bit-flip noise is shown in Fig. 5(a). The solid lines are the performance of the neural decoder with the uniform data construction. The bottom dashed lines represent the logical error probability achievable with the MD decoder. The colors red, green, blue, and cyan correspond to distances 5, 7, 9, and 11, respectively.

Comparing these two types of decoders, we see that the logical error probability of the neural decoder is near-optimal with a 10^6 data set at distance 11. On the other hand, there are gaps between the converged logical error probabilities of the neural decoder and that of the MD decoder when the distance is large. We speculate that these gaps are caused by imperfect learning of the spatial information of the topological codes, since it is partially improved with the network construction discussed in the next section.

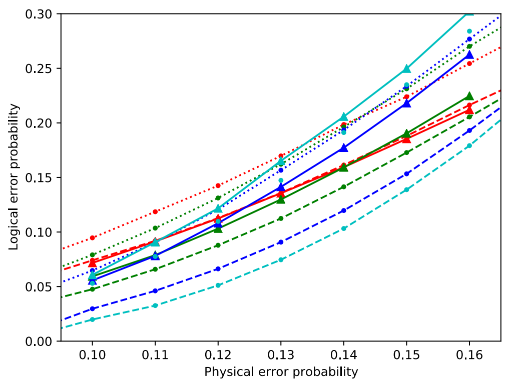
We also implemented the neural decoder with short diagnosis, i.e., the construction with $N_{01} = N_{10} = N_{11} = 1$, where $N_{\boldsymbol{w}}$ is a number of logical operators in the rows of H_g corresponding to the class $\boldsymbol{w}$. This is equivalent to the construction which we showed as an example in Sec. III A 2. We call this construction, with the normalized sensitivity of $O(1)$, short diagnosis construction, which is shown as the pale plots in Fig. 5(a). Note that the performance of this decoder depends on the choice of the logical operators. We have tried this construction with various choices of the logical operators. The plotted data are the best values among the choices of the logical operators which we have tried. Although both constructions become near-optimal in the limit of large training data size, we see that the performance with the uniform data construction achieves smaller logical error probability than that with the short diagnosis construction for any size of the training data set. We have also confirmed that the performance of the neural decoder degrades when the row vectors of H_g consist of the same $O(d)$ logical operators of X, Y, and Z. In this case, while the number of the rows in H_g is the same as that of the uniform data construction, the sensitivity $m(H_g)$ becomes $O(d)$, which makes the normalized sensitivity $\frac{m(H_g)}{M(H_g)}$ to be $O(1)$. Though these results are not plotted, the performance of this construction is almost the same as the short diagnosis construction. These results support our argument that it is essential for the performance of the neural decoder to minimize the normalized sensitivity.

The results with the depolarizing noise are shown in Fig. 5(b). Note that for the surface code under correlated noise such as the depolarizing noise, it is not known how an efficient MD decoder can be constructed. We see that the performance of the neural decoder becomes near-optimal and is superior to that of the MWPM decoder with 10^6 training samples at $d = 5, 7, 9$.

We also calculated the logical error probability in terms of the physical error probability. The plots in the vicinity of the threshold value are shown in Fig. 6. We chose the size of the training data set as 10^6 and calculated the performance for the distance $d = 5, 7, 9, 11$ and for the bit-flip and depolarizing noises. We used hyperparameters which were the best in the calculation of Fig. 5 when the size of the training data set is 10^6. For both of the noise models, the performance is near-optimal when the distance is small. On the other hand, when the distance becomes large, the logical error probability becomes larger than that of the MWPM decoder. The error threshold is usually estimated with the cross point of the performance in terms of the distance. We see that the error threshold based on the distance is worse than that of the MWPM decoder, though the logical error probability is smaller than that of the MWPM decoder. Note that we will



(a) $[[d^2, 1, d]]$ surface code - Bit-flip noise



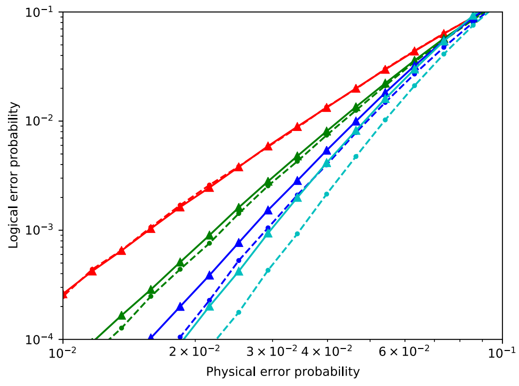
(b) $[[d^2, 1, d]]$ surface code - Depolarizing noise

FIG. 6. The performance comparison between the neural decoder with the uniform construction (solid lines), the MD decoder (dashed lines), and the MWPM decoder (dotted lines) in the case of the $[[d^2, 1, d]]$ surface code. We calculated the performance for distances $d = 5$ (red), 7 (green), 9 (blue), and 11 (cyan) with the same 10^6 training data set. (a) The case of the bit-flip noise. (b) The case of the depolarizing noise.

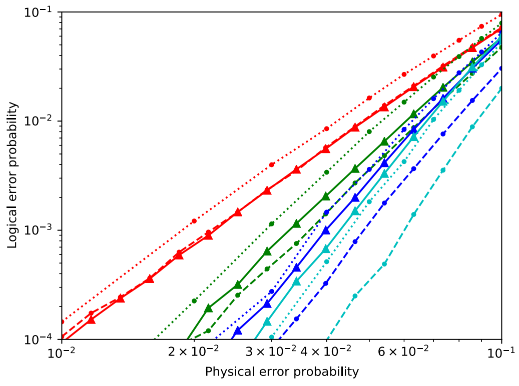
show that the performance can be improved by considering spatial information in Sec. IV where the decoding time is discussed in detail.

The actual experiment is expected to be performed with a physical error probability sufficiently smaller than the threshold value. Therefore, we calculated the performance of the decoder with a small physical error probability. The numerical results are shown in Fig. 7. Since the training data set generated with a small value of p is highly imbalanced, we trained the model with $p = 0.08$ for the bit-flip noise model, and with $p = 0.11$ for the depolarizing noise model. Then we tested the trained model with the data set generated with $p \leqslant 0.1$. We see that the logical error probability is smaller than the MWPM decoder in this region, for all the distances except $d = 11$.

We also calculated the performance of the neural decoder for two types of color codes. We chose the size of training data set as 10^6 and calculated the logical error probability for the distance $d = 3, 5, 7, 9$. Note that we cannot construct an efficient MD decoder in the color code even under independent bit-flip and phase-flip noise. The plots of the logical error probability to the physical error probability p are shown in Fig. 8. The configurations of the plots and lines are the same as those for the surface code. In the case of the bit-flip noise, the near-optimal performance is achieved. The performance is also near-optimal in the case of the depolarizing noise at distances except $d = 9$. We also see that the performance of the [4,8,8]-color code is better than that of the [6,6,6]-color code. We speculate that this is because the number of the physical qubits required in the [4,8,8]-color codes is smaller than that of the [6,6,6]-color code at the same distance. These results suggest that the neural decoder with the uniform data construction is effective also for the color codes.

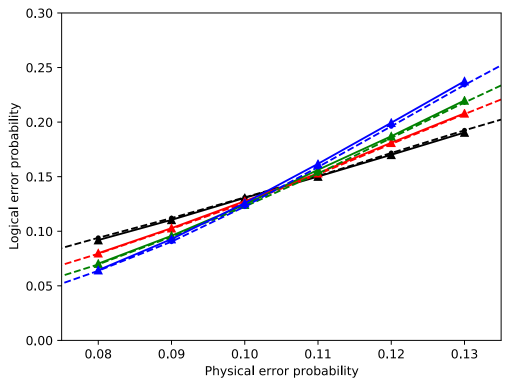


(a) $[[d^2, 1, d]]$ surface code - Bit-flip noise

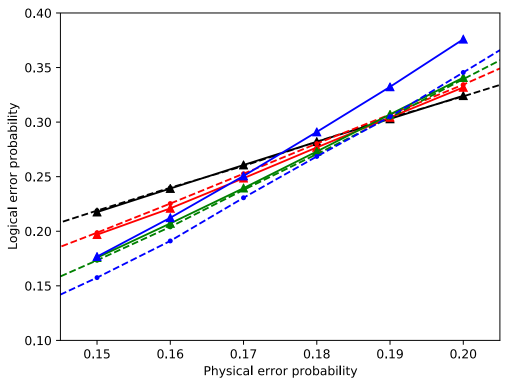


(b) $[[d^2, 1, d]]$ surface code - Depolarizing noise

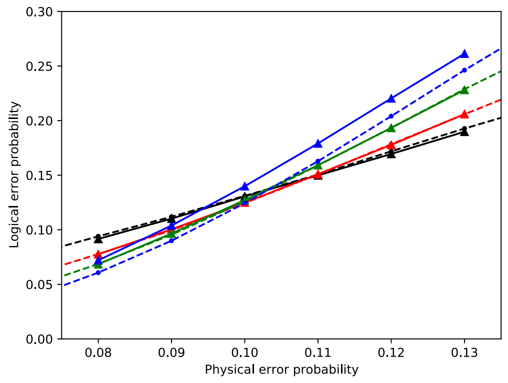
FIG. 7. The performance comparison between the neural decoder with the uniform construction (solid lines), the MD decoder (dashed lines), and the MWPM decoder (dotted lines) in the case of the $[[d^2, 1, d]]$ surface code. The neural decoder is trained with the 10^6 training data set. We calculated the performance for distances $d = 5$ (red), 7 (green), 9 (blue), and 11 (cyan). (a) The case of the bit-flip noise. The training data set is generated at the physical error probability $p = 0.08$. (b) The case of the depolarizing noise. The training data set is generated at the physical error probability $p = 0.11$.



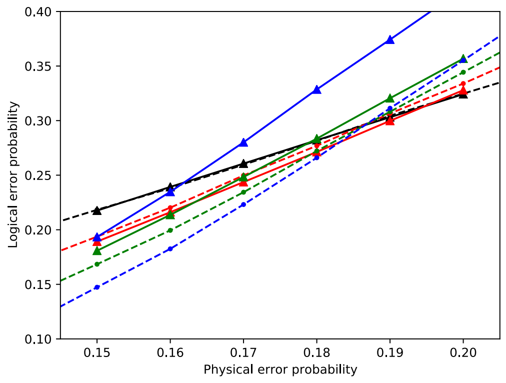
(a) [4,8,8]-color code - Bit-flip noise



(b) [4,8,8]-color code - Depolarizing noise



(c) [6,6,6]-color code - Bit-flip noise



(d) [6,6,6]-color code - Depolarizing noise

FIG. 8. The performance comparison between the neural decoder with the uniform construction (solid lines) and the MD decoder (dashed lines) in the color codes. We calculated the performance for distances $d = 3$ (black), 5 (red), 7 (green), and 9 (blue) with the 10^6 training data set. (a) The case of the bit-flip noise in the [4,8,8]-color code. (b) The case of the depolarizing noise in the [4,8,8]-color code. (c) The case of the bit-flip noise in the [6,6,6]-color code. (d) The case of the depolarizing noise in the [6,6,6]-color code.

IV. UTILIZING SPATIAL INFORMATION

In this section we describe the construction of the neural network with convolutional layers. We first discuss how the required size of the data set is expected to be suppressed if the model can utilize the spatial information of the two-dimensional quantum codes. Then we introduce a construction of the neural network with convolutional layers to utilize the spatial information of the topological codes. We finally show the numerical results and show that the performance of the neural decoder is improved.

A. Importance of the spatial information

In this section, we utilize spatial information of the syndrome by using the Convolutional Neural Network (CNN) [35] as a prediction model. When we use the MLP model, each layer is represented as a vector of neurons, and the neurons are densely connected from layer to layer. On the other hand, each layer of CNN is matrix-shaped, and each element of the next layer is calculated only from the local region of the previous layer using a map called a filter. The local filtering with the same filters can be considered as a convolution. For the mathematical formulation, see the next subsection.

Though the CNN model is frequently used for the image recognition, this can be used for the recognition of a feature where spatial information of the feature is essential for the task. For example, the CNN model is expected to be effective for the classification of the spin-glass phase [22]. In such a task, the local correlations of the spins are essential for prediction. Furthermore, since the patterns of local spins are translational invariant, filters which extract the feature in a local region are expected to be reused in the other regions. These properties match the premise of CNN, and we can expect that the performance of the CNN model is improved compared with other models such as MLP model for a fixed number of the training data set.

In this subsection, we explain why CNN is also expected to be effective for decoding in the topological codes. In the case of the two-dimensional topological codes, the syndrome values have a natural two-dimensional arrangement. By carefully reshaping syndrome values as a matrix-shaped arrangement of the feature vector elements, we can explicitly let the model use local correlations of the observed syndromes using the CNN model. In the topological codes, a flip of a single physical qubit invokes at most a constant number (2 in the surface code, 3 in the color code) of local bit flips in the syndrome value. This implies that whether two (or three) flipped syndrome bits

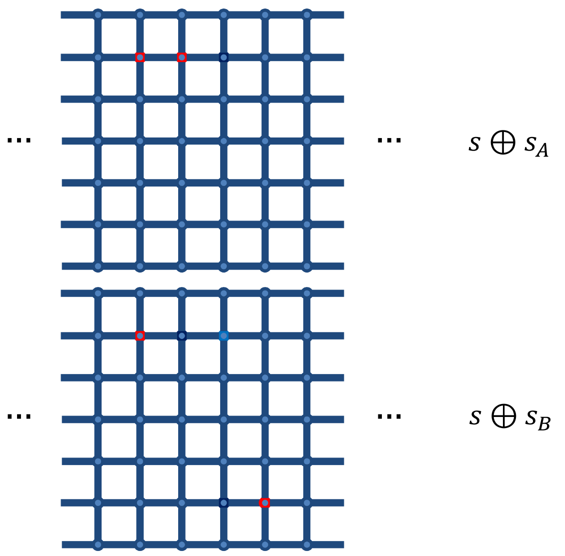


FIG. 9. Example of the difference of the syndrome values. Each node in the figure corresponds to each syndrome value and an edge corresponding to the error status of each physical qubit. The color of the circle corresponds to whether the syndrome measurement detects an error.

are found in a local region or not is useful for predicting the property of the physical errors. For an intuitive understanding, we elaborate the reason through examples. We consider the surface code under bit-flip errors. Suppose that a syndrome vector $\boldsymbol{s}$ is given in the prediction phase, and the model has encountered slightly different syndrome vectors $\boldsymbol{s}_A$ and $\boldsymbol{s}_B$, where the difference from $\boldsymbol{s}$ is shown in Fig. 9, in the training phase. The representation is the same as that of Fig. 4. We ignore the boundary effects of the topological codes for simplicity. In both cases, the syndrome is two Hamming distances away from the original syndrome vector $\boldsymbol{s}$, namely, $h(\boldsymbol{s} \oplus \boldsymbol{s}_A) = h(\boldsymbol{s} \oplus \boldsymbol{s}_B) = 2$. On the other hand, there is a difference between the two syndromes in light of whether it helps the prediction of the diagnosis for the observed syndrome $\boldsymbol{s}$. In Eq. (60) we introduced a set of physical errors $S(\boldsymbol{s}, \boldsymbol{w}^*; h)$ such that any $\boldsymbol{e} \in S(\boldsymbol{s}, \boldsymbol{w}^*; h)$ with $h \sim N(H_g)^{-1}$ produces training data useful for estimating the L2 diagnosis vector of $\boldsymbol{s}$. For a given $\boldsymbol{s}$ and $\boldsymbol{s}'$, if there is no vector $\boldsymbol{e}_\delta$ such that $H_c \Lambda \boldsymbol{e}_\delta^{\mathrm{T}} = \boldsymbol{s} \oplus \boldsymbol{s}'$ and $h(\boldsymbol{e}_\delta) \lesssim N(H_g)^{-1}$, we see no errors $\boldsymbol{e}$ with $\boldsymbol{s}(\boldsymbol{e}) = \boldsymbol{s}'$ are contained in $\cup_{\boldsymbol{w} \in \{0,1\}^{2k}} S(s, \boldsymbol{w}; N(H_g)^{-1})$. In the case of $\boldsymbol{s}_A$ and $\boldsymbol{s}_B$, there is such a physical error $\boldsymbol{e}_\delta$ with a small Hamming weight for $\boldsymbol{s}_A$, but not for $\boldsymbol{s}_B$. Thus, if the prediction model can distinguish the samples with $\boldsymbol{s}_A$ from those with $\boldsymbol{s}_B$, it can recognize that the samples with $\boldsymbol{s}_B$ in the training data set are not relevant to the prediction for $\boldsymbol{s}$. The CNN model can distinguish it since it naturally utilizes the spatial information of the syndrome values. On the other hand, the MLP model cannot easily distinguish them since the model is not provided with the relevant spatial structure before training. This discussion implies the logical error probability under the fixed number of the training data set is expected to be improved with the use of a CNN model.

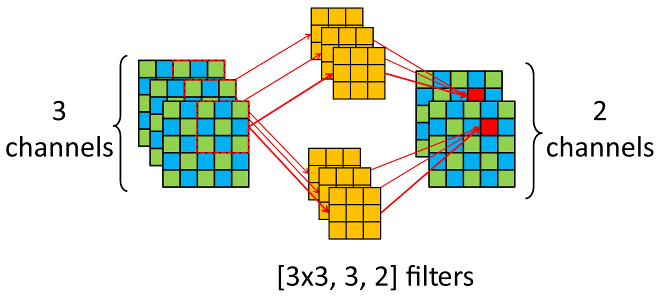


FIG. 10. A simple case of convolutional layer where the input channel is three and the output channel is two.

B. Construction of the network

A convolutional neural network extracts patterns from image data through trainable filters that activate (produce a high value) when there are specific local patterns in the input data. The network usually consists of multiple convolutional layers $C^{(n)}$ each of which consists of different filtered versions of the image data $C_p^{(n)}$, indexed by a channel number p. The $(n-1)$th layer with Q channels are filtered to the nth layer with P channels with $Q \times P$ filters which we can be represented by a matrix $f^{(n-1,n)}$. We can describe this relation as

$$C_{i,j,p}^{(n)} = A\left(\sum_{d_x}\sum_{d_y}\sum_{q} f_{d_x,d_y,q,p}^{(n-1,n)} C_{i+d_x,j+d_y,q}^{(n-1)} + b_p^{(n)}\right), \quad (91)$$

where $C_{i,j,p}^{(n)}$ is the (i, j) element of the pth channel in the nth convolutional layer, and $f_{d_x,d_y,q,p}^{(n-1,n)}$ is the (d_x, d_y) element in the (q, p)th filter from the $(n-1)$th layer to the nth layer. Parameter $b_p^{(n)}$ is the bias added to the pth channel of the nth layer. A simple example is shown in Fig. 10 where one layer has three channels and the next layer has two channels.

To use a CNN in our decoding task, we have to express the syndrome vector $\boldsymbol{s}$ with an appropriate matrix representation. We reallocate the syndrome vector for the $[[2d^2 - 2d + 1, 1, d]]$ and $[[d^2, 1, d]]$ codes as shown in Fig. 11. For the $[[2d^2 - 2d + 1, 1, d]]$ surface code, $\boldsymbol{s}$ is converted into two $d \times (d-1)$ matrices for the X syndrome and the Z syndrome. Similarly, for the $[[d^2, 1, d]]$ surface code, $\boldsymbol{s}$ is converted into two $(d-1) \times (d+1)/2$ matrices. We have not tried it on color codes, since it is hard to interpret the allocation of the syndromes in color codes with rectangular shapes.

Our CNN decoder consists of three convolutional layers followed by a single fully connected hidden layer as shown in Fig. 12. At the last convolutional layer, the output channel is flattened to a single one-dimensional vector. Then it is used as an input for the subsequent fully connected hidden layer. For each convolutional layer, the channel number is chosen to be $10d$ for the first two layers and $5d$ for the last layer. Details about the model architecture are described in Appendix B. It is worth noting that we used the same filters for decoding ofboth X and Z flip errors, and max-pooling is not used as it is observed to reduce the performance of the decoder.

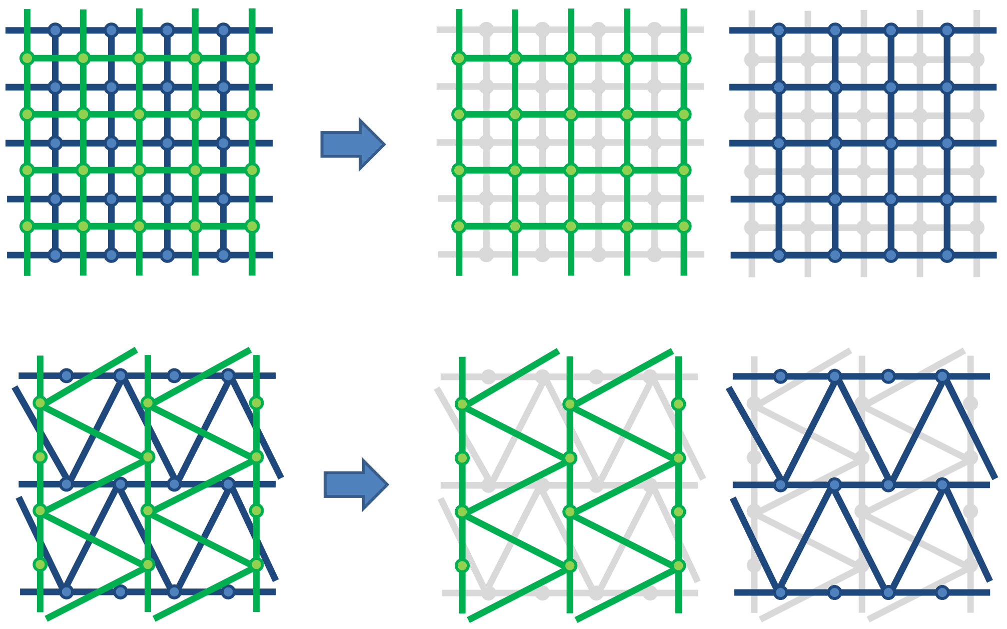


FIG. 11. Visualization of how to split and reallocate the syndrome vectors to the two input layers of the neural network. In the case of the $[[2d^2 - 2d + 1, 1, d]]$ code, the lattice is split into a $(d - 1) \times d$ array of syndrome and a $\pi/4$ rotated one. We input two $d \times (d - 1)$ matrixes as the first layer of the neural network. In the case of the $[[d^2, 1, d]]$ code, we split the syndromes into two $(d - 1) \times \frac{(d+1)}{2}$ arrays.

C. Numerical result

We call a neural decoder with the MLP model as a MLP decoder, and one with the CNN model as a CNN decoder. We compare the performance of the CNN decoder with those of the MLP decoder, MD decoder, and MWPM decoder. Note that the training data set is generated with the uniform data construction.

First, we compare the performance of the CNN decoder and that of the MLP decoder in the case of the surface codes. The numerical results are shown in Fig. 13, where the solid lines and dashed lines are the logical error probability for the CNN decoder and the MLP decoder, respectively. The colors red, green, blue, and cyan correspond to distances $d = 5$, 7, 9, and 11, respectively. For both types of surface codes, the CNN decoder shows superior performance to that of the MLP decoder at large distances. In particular, in the case of the $[[2d^2 - 2d + 1, 1, d]]$ surface code, the CNN decoder shows significant improvement of a logical error probability. We see that the CNN model is effective for improving the performance of the neural decoder at large distances. On the other hand, we see that the CNN decoder shows inferior performance to the MLP decoder at a small distance. We speculate the reason for this as follows. The CNN model assumes that the local features can be extracted using the same filter everywhere. Such an assumption is not necessarily true when the distance is small, since almost all the filtered local regions, of size 3×3, for example, are on or near the boundary of the two-dimensional codes. Note that we tried to avoid this problem by padding the boundaries with various values, such as 0.5 or -1, but the performance in the small distance did not improve.

Next, we compared the performance of the CNN decoder with those of the MD decoder and the MWPM decoder. The results are shown in Fig. 14. The solid lines, the dashed lines, and the dotted lines are the logical error probability for the CNN decoder, the MD decoder, and the MWPM decoder,

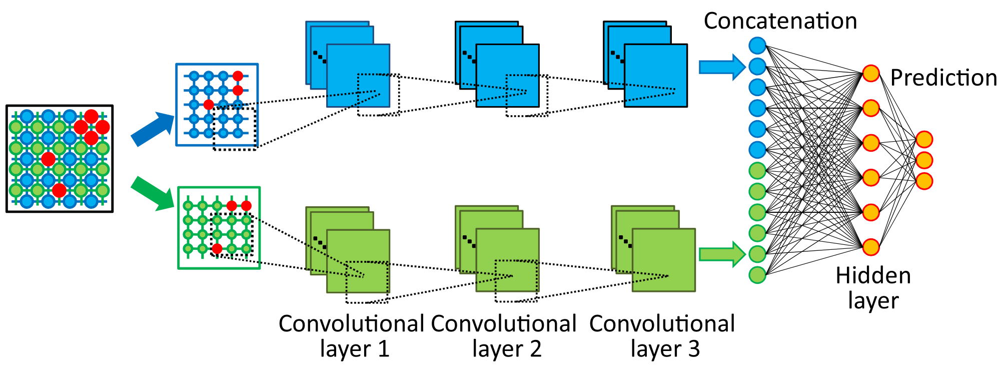
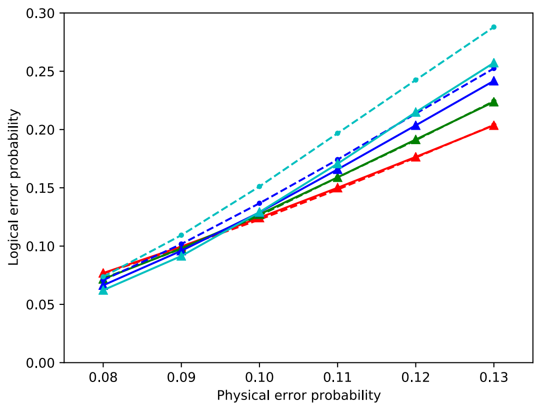
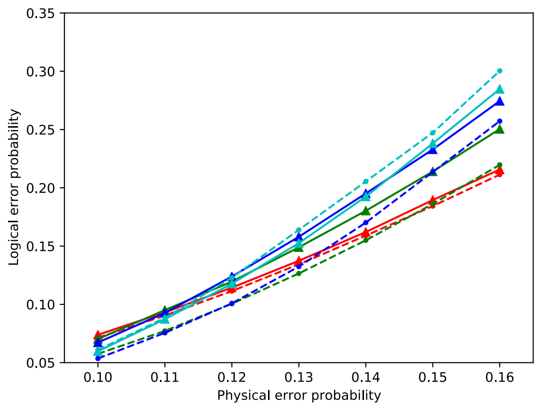


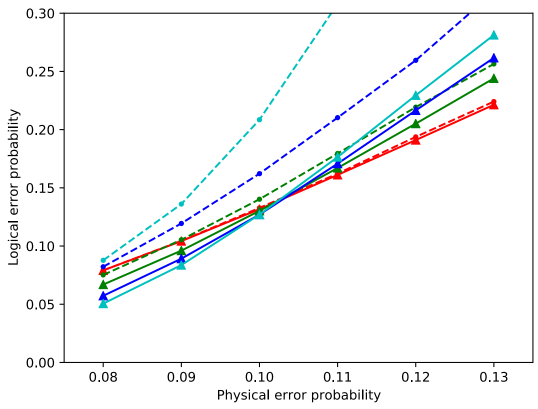
FIG. 12. CNN decoder architecture used in our work. We separately pass X and Z syndrome values through the same convolutional layers, and concatenate them before feeding to the following fully connected hidden layer.



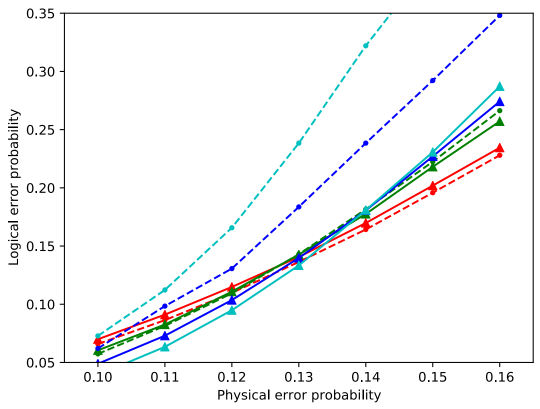
(a) $[[d^2, 1, d]]$ surface code - Bit-flip noise



(b) $[[d^2, 1, d]]$ surface code - Depolarizing noise



(c) $[[2d^2 - 2d + 1, 1, d]]$ surface code - Bit-flip noise



(d) $[[2d^2 - 2d + 1, 1, d]]$ surface code - Depolarizing noise

FIG. 13. The performance comparison between the CNN decoder (solid lines) and the MLP decoder (dashed lines) in the case of the surface code. We calculated the performance for distance $d = 5$ (red), 7 (green), 9 (blue), and 11 (cyan). (a) The bit-flip noise in the $[[d^2, 1, d]]$ code. (b) The depolarizing noise in the $[[d^2, 1, d]]$ code. (c) The bit-flip noise in the $[[2d^2 - 2d + 1, 1, d]$ code. (d) The depolarizing noise in the $[[2d^2 - 2d + 1, 1, d]]$ code.

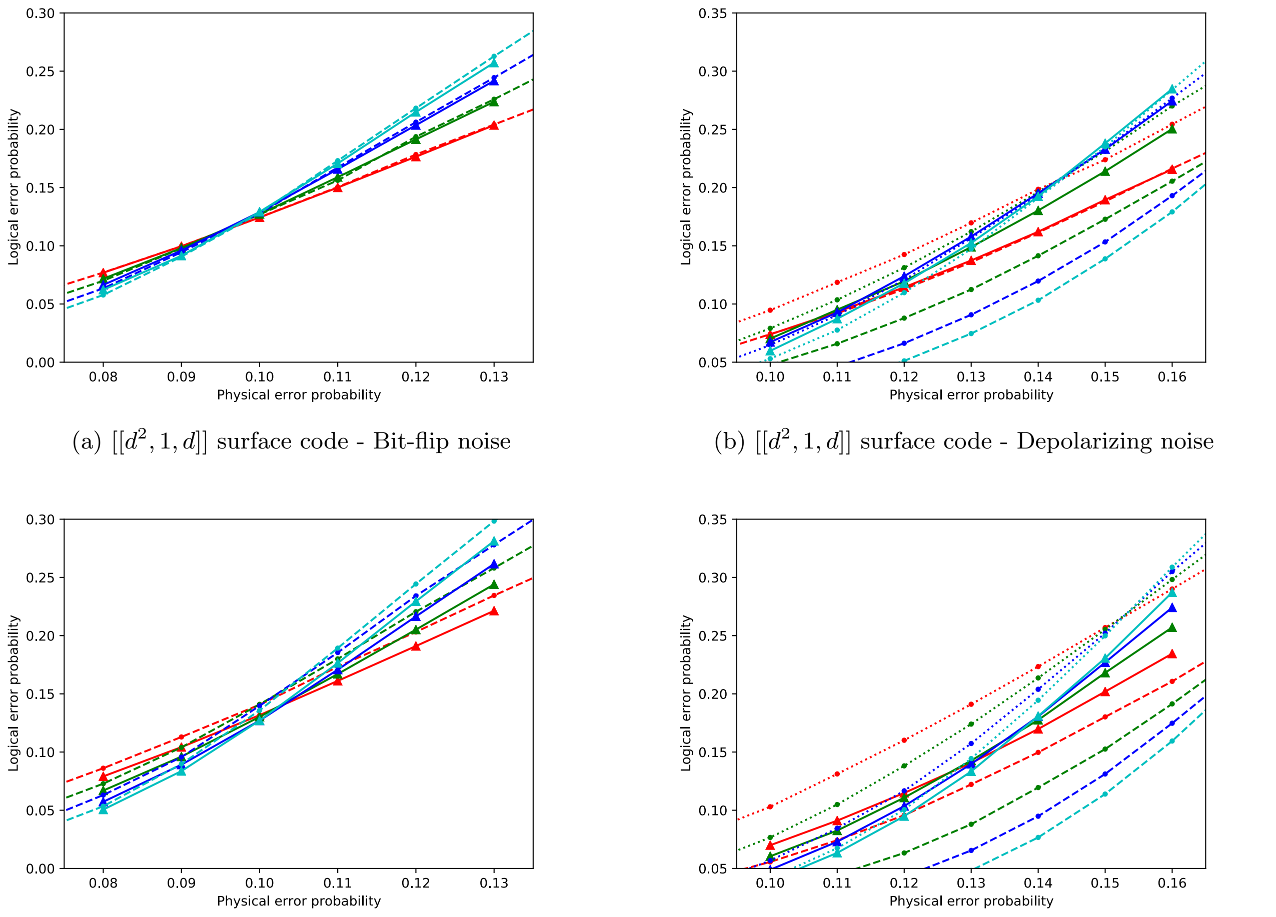
respectively. The colors red, green, blue, and cyan correspond to distances $d = 5$, 7, 9, and 11, respectively. In the case of the bit-flip noise, we see that the logical error probabilities of the CNN decoder iare equal to or slightly better than those of the MD decoder. In the case of the depolarizing noise, though there are gaps between the performances of the CNN decoder and the MD decoder, the performance of the CNN decoder is superior or comparable to that of the MWPM decoder even at the distance $d = 11$. Note that as shown in Ref. [25], the performance of a partial look-up table decoder, which uses the training data set as a simple look-up table and outputs the random guess if the input is not found in the training data set, is much worse than MWPM decoder at $d = 5$. Thus, we can say that neural decoders can learn and guess unseen syndrome patterns from the data set since a high performance of a neural decoder cannot be obtained by using a training data set as a lookup table.

For the decoding time, the MLP decoder and CNN decoder took 2.2 ms and 7 ms, respectively, at $d = 11$ and $p = 0.15$, while the MWPM decoder and MD decoder took 56 μs and 330 ms. Thus, not only does our proposal show better performance than or comparable to that of the MWPM decoder, but also its decoding time is two orders of magnitude faster than the MD decoder. See Appendix B for the benchmarking environment of decoding times.

We also calculated the logical error probability of the CNN decoder at a small physical error probability p in the case of the $[[2d^2 - 2d + 1, 1, d]]$ surface code. We trained the CNN decoder at $p = 0.08$ for the bit-flip noise model and at $p = 0.11$ for the depolarizing noise model. Then the decoder is tested with the data set generated with small physical error probabilities. The plots are shown in Fig. 15. In the case of the bit-flip noise, the CNN decoder achieves performance close to the MD decoder also at small physical error probabilities. In the case of the depolarizing noise, the performance of the neural decoder with the CNN decoder is superior to that of the MWPM decoder at $d = 9$ and comparable at $d = 11$. We can say that the CNN model is effective also for a use of neural decoders at small physical error probabilities.

V. CONCLUSION

In this paper, we theoretically analyzed mechanism of machine-learning-based decoders for QEC, proposed a



(a) $[[d^2, 1, d]]$ surface code - Bit-flip noise

(b) $[[d^2, 1, d]]$ surface code - Depolarizing noise

(c) $[[2d^2 - 2d + 1, 1, d]]$ surface code - Bit-flip noise

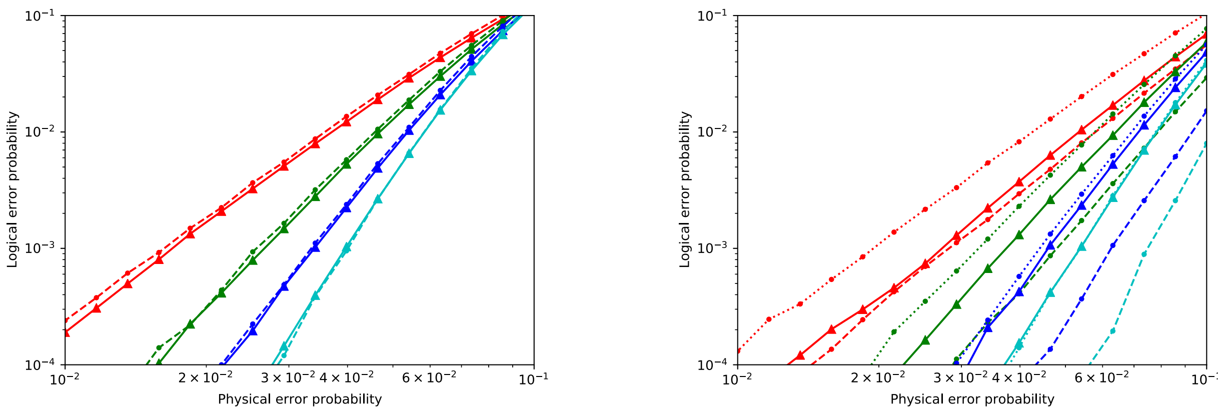
(d) $[[2d^2 - 2d + 1, 1, d]]$ surface code - Depolarizing noise

FIG. 14. The performance comparison between the CNN decoder (solid lines), the MD decoder (dashed lines), and the MWPM decoder (dotted lines) in the surface codes. We calculated the performance for distance $d = 5$ (red), 7 (green), 9 (blue), and 11 (cyan). (a) The bit-flip noise in the $[[d^2, 1, d]]$ code. (b) The depolarizing noise in the $[[d^2, 1, d]]$ code. (c) The bit-flip noise in the $[[2d^2 - 2d + 1, 1, d]]$ code. (d) The depolarizing noise in the $[[2d^2 - 2d + 1, 1, d]]$ code.

general direction to construct the data set and the neural network, and have numerically shown in Secs. III and IV that our direction is effective compared with the existing approaches.

Since the formalism of the machine learning is flexible, there are many possible ways to reduce the decoding problem in QEC to the task of machine learning. In order to clarify what is the best way of reduction, we introduced the linear prediction framework in Sec. III A. We showed this framework essentially includes the existing methods as specific cases and showed that the framework enables us to discuss conditions for satisfying natural requirements for a good decoder for QEC. In particular, we have derived the condition to perform the optimal decoding in the limit of a large training data size as Lemma III.1 and Lemma III.2. We also introduced a measure, normalized sensitivity, defined in Definition III.4, which represents a properly scaled bound on the deviation in the prediction target resulting from a small change in the physical error pattern. We proposed to use this measure as a criterion for constructing a better decoder. We then proposed a general direction for constructing the data set, uniform data construction, which can be applicable to general topological codes. In Sec. III D we numerically confirmed that the performance of the neural decoder is improved with the uniform data construction. Our decoder was found to be superior to known efficient decoders, such as neural decoders proposed in the existing methods and the decoder based on the reduction to minimum-weight perfect matching. We also confirmed in this subsection that the performance of our neural decoder is near-optimal in various situations by comparing it with the minimum-distance decoder, which is known to be near-optimal but not efficient in general. We also confirmed that the neural decoder can achieve near-optimal performance not only for surface codes but also for color codes.

Another important factor of the neural decoder is construction of the neural network. In Sec. IV we discussed the importance of the spatial information of the syndrome measurement in order to let the prediction model recognize useful samples from a given training data set. To utilize the spatial information, we proposed a neural decoder with the convolutional neural network. In Sec. IV C we numerically observed that the performance of the neural decoder is further improved with this network construction in the surface code. In particular, we showed in Figs. 14 and 15 that the proposed



(a) $[[2d^2-2d+1,1,d]]$ surface code - Bit-flip noise (b) $[[2d^2-2d+1,1,d]]$ surface code - Depolarizing noise

FIG. 15. The performance comparison between the CNN decoder (solid lines), the MD decoder (dashed lines), and the MWPM decoder (dotted lines) in the case of the $[[2d^2-2d+1,1,d]]$ surface code, where the decoders are trained with the training data set generated at the fixed error rate. We calculated the performance for distance $d=5$ (red), 7 (green), 9 (blue), and 11 (cyan). (a) The case of the bit-flip noise. The training data set is generated at the physical error probability $p=0.08$. (b) The case of the depolarizing noise. The training data set is generated at the physical error probability $p=0.11$.

neural decoder achieves a smaller logical error probability than that of the decoder based on minimum-weight perfect matching even at distance $d=11$ with a training data set size of 10^6.

Since using machine learning for QEC is an emergent field, there are still many possible extensions and directions of the neural decoders. As we detailed in Appendix B, the prediction time of the neural decoders is smaller than that of the MD decoder, but larger than that of the MWPM decoder in our desktop PC. Since the prediction of the neural decoders can be done with simple matrix multiplications, the time for prediction can be further made short by using an optimized hardware such as field-programmable gate array (FPGA), which is popular in experiments with superconducting qubits [4–6], for example. While we have discussed only a label linearly generated in GF(2), the performance may be more improved by allowing labels nonlinearly generated from the physical error. For example, the relation between the syndrome values and the weight of the physical error, which cannot be generated linearly in GF(2), can be trained and predicted independently with a neural network. Then the recovery map can be predicted with the syndrome values and the predicted weight with another neural network. The linear prediction framework also limits the sample in the training data set to that is sampled from the assumed physical error distribution. However, the distribution which is the best for the training is not necessarily the same as the actual distribution. For example, we saw that the prediction model trained at the physical error probability around the threshold value shows high performance also at low physical error probabilities. There can be a more artificial way to construct the training data set to achieve the performance with a smaller size of the training data set. In the numerical investigation, we observed that the required amount of the data set becomes exponentially large in terms of the distance. This may be suppressed by renormalizing the matrix representation of the syndrome with trained filters as done in the renormalization group decoder [19]. We expect that CNN is also applicable to the color codes by using nonrectangle filters. When the stabilizer measurements themselves suffer from noise, stabilizer measurements are often repetitively performed during QEC. In such a case, the length of the syndrome data is not fixed. In our construction, we need to train the neural network again whenever the length of the syndrome data changes. References [26,28] focused on removing this drawback by utilizing a recurrent neural network and convolutional neural network. Using the technique proposed in Refs. [26,28], our neural decoder also may be applicable to the cases when we perform repetitive stabilizer measurements. In practice, the probability distribution of Pauli noise is not uniform and correlated. If they are biased and correlated, it is expected that the performance of quantum error correction codes is degraded [36]. On the other hand, by properly utilizing information about the noise distribution, we can construct a tailored scheme for such noise models [37,38]. An advantage of neural decoder is that it can learn general bias or correlation from a sampled training data set. Thus, we can expect that a neural decoder may tailored to such a practical case by feeding a sufficiently large number of training data sets.

ACKNOWLEDGMENTS

This work is supported by KAKENHI Grant No. 16H02211; PRESTO, JST, Grants No. JPMJPR1668 and No. JPMJPR1916; CREST, JST, Grants No. JPMJCR1671 and No. JPMJCR1673; ERATO, JST, Grant No. JPMJER1601; MEXT Q-LEAP Grants No. JPMXS0120319794, No. JPMXS0118067394, and No. JPMXS0118068682; and the Photon Frontier Network Program, MEXT. Y.S. is supported by the Advanced Leading Graduate Course for Photon Science. D.A. and Y.S. contributed equally to this work. Y.S. contributed to the construction of the data set in Sec. III, and A.D.

to the construction of the network in Sec. IV. K.F. and M.K. motivated and supervised the idea and discussion of this paper.

APPENDIX A: PROOF OF THE LEMMAS

1. Proof of the converse part in Lemma III.1

Here we prove the last statement of Lemma III.1. When Eq. (20) does not hold, either (1) there exists $\boldsymbol{e}_1$ such that

$$\boldsymbol{e}_1 \notin \mathcal{L}_0, \tag{A1}$$

$$H_{cg}\Lambda\boldsymbol{e}_1^{\mathrm{T}} = 0, \tag{A2}$$

or (2) there exists $\boldsymbol{e}_1$ such that

$$\boldsymbol{e}_1 \in \mathcal{L}_0, \tag{A3}$$

$$H_{cg}\Lambda\boldsymbol{e}_1^{\mathrm{T}} \neq 0. \tag{A4}$$

For (1), consider two probability distributions $\{p_{\boldsymbol{e}}\}$ and $\{p'_{\boldsymbol{e}}\}$ such that

$$\mathrm{Pr}_{\boldsymbol{e}\sim\{p_{\boldsymbol{e}}\}}[\boldsymbol{e}=0|\boldsymbol{s}(\boldsymbol{e})=0] = 0.75, \tag{A5}$$

$$\mathrm{Pr}_{\boldsymbol{e}\sim\{p_{\boldsymbol{e}}\}}[\boldsymbol{e}=\boldsymbol{e}_1|\boldsymbol{s}(\boldsymbol{e})=0] = 0.25, \tag{A6}$$

and

$$\mathrm{Pr}_{\boldsymbol{e}\sim\{p'_{\boldsymbol{e}}\}}[\boldsymbol{e}=0|\boldsymbol{s}(\boldsymbol{e})=0] = 0.25, \tag{A7}$$

$$\mathrm{Pr}_{\boldsymbol{e}\sim\{p'_{\boldsymbol{e}}\}}[\boldsymbol{e}=\boldsymbol{e}_1|\boldsymbol{s}(\boldsymbol{e})=0] = 0.75. \tag{A8}$$

An optimal decoder for each case succeeds with probability 0.75 given $\boldsymbol{s}=0$. On the other hand, since $\boldsymbol{g}^{(\delta)}(0)=0$ in both cases, only the value of $\boldsymbol{r}^*(0,0)$ is relevant. Since $\boldsymbol{w}(0)\neq\boldsymbol{w}(\boldsymbol{e}_1)$, any choice of $\boldsymbol{r}^*(0,0)$ leads to a success probability no greater than 0.25 for at least one of the cases.

For (2), choose $\boldsymbol{w}\neq 0$, and if $H_g\Lambda(\boldsymbol{w}G)^{\mathrm{T}}\neq 0$, define

$$\boldsymbol{e}_2 := \boldsymbol{w}G. \tag{A9}$$

Otherwise, define

$$\boldsymbol{e}_2 := \boldsymbol{e}_1 \oplus \boldsymbol{w}G. \tag{A10}$$

It ensures that $\boldsymbol{s}(\boldsymbol{e}_2)=0$ and $\boldsymbol{g}_2 := H_g\Lambda\boldsymbol{e}_2 \neq 0$. Consider two probability distributions $\{p_{\boldsymbol{e}}\}$ and $\{p'_{\boldsymbol{e}}\}$ such that

$$\mathrm{Pr}_{\boldsymbol{e}\sim\{p_{\boldsymbol{e}}\}}[\boldsymbol{e}=0|\boldsymbol{s}(\boldsymbol{e})=0] = 0.4, \tag{A11}$$

$$\mathrm{Pr}_{\boldsymbol{e}\sim\{p_{\boldsymbol{e}}\}}[\boldsymbol{e}=\boldsymbol{e}_1|\boldsymbol{s}(\boldsymbol{e})=0] = 0.0, \tag{A12}$$

$$\mathrm{Pr}_{\boldsymbol{e}\sim\{p_{\boldsymbol{e}}\}}[\boldsymbol{e}=\boldsymbol{e}_2|\boldsymbol{s}(\boldsymbol{e})=0] = 0.6, \tag{A13}$$

and

$$\mathrm{Pr}_{\boldsymbol{e}\sim\{p'_{\boldsymbol{e}}\}}[\boldsymbol{e}=0|\boldsymbol{s}(\boldsymbol{e})=0] = 0.3, \tag{A14}$$

$$\mathrm{Pr}_{\boldsymbol{e}\sim\{p'_{\boldsymbol{e}}\}}[\boldsymbol{e}=\boldsymbol{e}_1|\boldsymbol{s}(\boldsymbol{e})=0] = 0.3, \tag{A15}$$

$$\mathrm{Pr}_{\boldsymbol{e}\sim\{p'_{\boldsymbol{e}}\}}[\boldsymbol{e}=\boldsymbol{e}_2|\boldsymbol{s}(\boldsymbol{e})=0] = 0.4. \tag{A16}$$

An optimal decoder for each case succeeds with probability 0.6 given $\boldsymbol{s}=0$. On the other hand, since $\boldsymbol{g}^{(\delta)}(0)=\boldsymbol{g}_2$ in both cases, only the value of $\boldsymbol{r}^*(\boldsymbol{g}_2,0)$ is relevant. Since $\boldsymbol{w}(0)\neq\boldsymbol{w}(\boldsymbol{e}_2)$, any choice of $\boldsymbol{r}^*(\boldsymbol{g}_2,0)$ leads to a success probability no greater than 0.4 for at least one of the cases.

TABLE I. Network architecture of the $[[2d^2-2d+1,1,d]]$ surface code.

Distance	Filter size	Channel number	Neuron number
5	[2 × 2], [3 × 3], [3 × 3]	50, 50, 25	1000
7	[2 × 2], [3 × 3], [4 × 4]	70, 70, 35	3000
9	[3 × 3], [4 × 4], [5 × 5]	90, 90, 45	5000
11	[4 × 4], [5 × 5], [6 × 6]	110, 110, 55	7000

2. Proof of the converse part in Lemma III.2

When the diagnosis matrix is not decomposable, there exists a nonempty subset $\mathcal{W}\subset\{0,1\}^{2k}$ such that

$$\sum_{\boldsymbol{w}\in\mathcal{W}}\alpha_{\boldsymbol{w}}\boldsymbol{g}_0(\boldsymbol{w}) = \sum_{\boldsymbol{w}\in\{0,1\}^{2k}\setminus\mathcal{W}}\beta_{\boldsymbol{w}}\boldsymbol{g}_0(\boldsymbol{w}), \tag{A17}$$

where $\alpha_{\boldsymbol{w}}, \beta_{\boldsymbol{w}} \geqslant 0$ and

$$\Gamma := \sum_{\boldsymbol{w}\in\mathcal{W}}\alpha_{\boldsymbol{w}} = \sum_{\boldsymbol{w}\in\{0,1\}^{2k}\setminus\mathcal{W}}\beta_{\boldsymbol{w}} > 0. \tag{A18}$$

Consider two probability distributions $\{p_{\boldsymbol{e}}\}$ and $\{p'_{\boldsymbol{e}}\}$ such that

$$\mathrm{Pr}_{\boldsymbol{e}\sim\{p_{\boldsymbol{e}}\}_A}[\boldsymbol{w}(\boldsymbol{e})=\boldsymbol{w},\boldsymbol{l}(\boldsymbol{e}) = \boldsymbol{l}|\boldsymbol{s}(\boldsymbol{e})=0] = \begin{cases}\alpha_{\boldsymbol{w}}/\Gamma & \boldsymbol{w}\in\mathcal{W},\boldsymbol{l}=0\\ 0 & \text{otherwise}\end{cases}, \tag{A19}$$

$$\mathrm{Pr}_{\boldsymbol{e}\sim\{p'_{\boldsymbol{e}}\}_B}[\boldsymbol{w}(\boldsymbol{e})=\boldsymbol{w},\boldsymbol{l}(\boldsymbol{e}) = \boldsymbol{l}|\boldsymbol{s}(\boldsymbol{e})=0] = \begin{cases}\beta_{\boldsymbol{w}}/\Gamma & \boldsymbol{w}\notin\mathcal{W},\boldsymbol{l}=0\\ 0 & \text{otherwise}\end{cases}. \tag{A20}$$

From Eq. (A17), the L2 diagnosis vector $\boldsymbol{g}^{(\mathrm{L2})}(0)$ is identical for the two distributions. On the other hand, the most probable class $\boldsymbol{w}$ is different for the two probability distributions. This means that a single decoder cannot perform the optimal decoding for both distributions.

3. Proof of the existence of faithful and decomposable diagnosis matrices

In the main text, we showed that a diagnosis matrix should be faithful and decomposable for performing optimal decoding in the ideal limit of the training process, and we showed an example for the case $k=1$. On the other hand, it is not trivial that there exists a faithful and decomposable construction of a diagnosis matrix for an arbitrary stabilizer code. We show that diagnosis matrix $H_g = WG$, where W is a $2^{2k}\times 2k$ binary

TABLE II. Network architecture of the $[[d^2,1,d]]$ surface code.

Distance	Filter size	Channel number	Neuron number
5	[2 × 2], [3 × 3], [3 × 3]	50, 50, 25	1000
7	[2 × 2], [3 × 3], [3 × 4]	70, 70, 35	3000
9	[2 × 3], [3 × 4], [4 × 5]	90, 90, 45	5000
11	[2 × 4], [3 × 5], [4 × 6]	110, 110, 55	7000

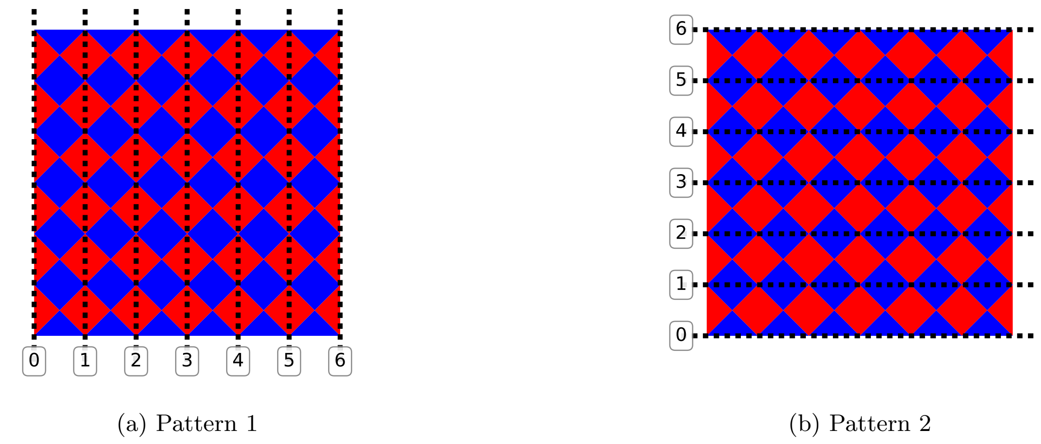


FIG. 16. Logical operators used for the construction of a diagnosis matrix for the surface codes $[[2d^2 - 2d + 1, 1, d]]$. Each dotted black line corresponds to a chosen logical operator.

matrix of which the ith row is a $2k$-bit binary representation of an integer i, is always faithful and decomposable for an arbitrary stabilizer code and for an arbitrary number of logical qubits k. Since row vectors of H_g contain all the logical operators, it is trivial that $\mathrm{span}(\{(H_{cg})_i\})$ is equivalent to the logical space $\mathcal{L}$, and H_g is faithful. The condition for decomposability is equivalent to the condition that $\{\boldsymbol{g}(\boldsymbol{w})|\boldsymbol{w} \in \{0, 1\}^{2k}\}$, where $\boldsymbol{g}(\boldsymbol{w}) := H_g\Lambda(\boldsymbol{w}G)^{\mathrm{T}}$, is affinely independent in real vector space. To show the latter, we first prove that for any pair of binary vectors $\boldsymbol{w}, \boldsymbol{w}' \in \{0, 1\}^{2k}$ such that $\boldsymbol{w} \neq \boldsymbol{w}'$, the weight of $\boldsymbol{g}(\boldsymbol{w}) \oplus \boldsymbol{g}(\boldsymbol{w}')$ is 2^{2k-1}. A 2^k-bit sequence $\boldsymbol{g}(\boldsymbol{w}) \oplus \boldsymbol{g}(\boldsymbol{w}')$ is given by

$$\boldsymbol{g}(\boldsymbol{w}) \oplus \boldsymbol{g}(\boldsymbol{w}') = WG\Lambda G^{\mathrm{T}}(\boldsymbol{w} \oplus \boldsymbol{w}')^{\mathrm{T}}. \tag{A21}$$

Since matrix $G\Lambda G^{\mathrm{T}}$ is invertible and since $\boldsymbol{w} \oplus \boldsymbol{w}' \neq 0$, we have $G\Lambda G^{\mathrm{T}}(\boldsymbol{w} \oplus \boldsymbol{w}')^{\mathrm{T}} \neq 0$. Since matrix W contains all possible $2k$-bit sequences, the half elements in the sequence $\boldsymbol{g}(\boldsymbol{w}) \oplus \boldsymbol{g}(\boldsymbol{w}')$ are 1, and the others are 0. Thus, the weight of $\boldsymbol{g}(\boldsymbol{w}) \oplus \boldsymbol{g}(\boldsymbol{w}')$ is 2^{2k-1}.

Let $\boldsymbol{v} := (1, \dots, 1)^{\mathrm{T}}$ be a real vector of order 2^{2k}. We define a set of vectors $\boldsymbol{h}(\boldsymbol{w}) := 2\boldsymbol{g}(\boldsymbol{w}) - \boldsymbol{v}$ for $\boldsymbol{w} \in \{0, 1\}^{2k}$, where this calculation is done in real vector space. Note that this map is equivalent to replace 0 and 1 to 1 and -1, respectively. Since this map from $\boldsymbol{g}(\boldsymbol{w})$ to $\boldsymbol{h}(\boldsymbol{w})$ is affine, $\{\boldsymbol{g}(\boldsymbol{w})\}$ is affinely independent if $\{\boldsymbol{h}(\boldsymbol{w})\}$ is linearly independent. The inner product $\boldsymbol{h}(\boldsymbol{w})\boldsymbol{h}(\boldsymbol{w}')^{\mathrm{T}}$ for $\boldsymbol{w} \neq \boldsymbol{w}'$ can be calculated as

$$\begin{aligned} \boldsymbol{h}(\boldsymbol{w})\boldsymbol{h}(\boldsymbol{w}')^{\mathrm{T}} &= \sum_i h(\boldsymbol{w})_i h(\boldsymbol{w}')_i \\ &= 2^{2k} - 2w(\boldsymbol{g}(\boldsymbol{w}) \oplus \boldsymbol{g}(\boldsymbol{w}')) \\ &= 0. \end{aligned} \tag{A22}$$

We used the fact that $h(\boldsymbol{w})_i h(\boldsymbol{w}')_i$ is 1 if $g(\boldsymbol{w})_i = g(\boldsymbol{w}')_i$, and -1 otherwise. Thus, the set of vectors $\{\boldsymbol{h}(\boldsymbol{w})\}$ is linearly independent in the real vector space, and the set of vectors $\{\boldsymbol{g}(\boldsymbol{w})\}$ is affinely independent. This means that $H_g = WG$ is faithful and decomposable for an arbitrary stabilizer code.

APPENDIX B: ADDITIONAL INFORMATION FOR THE IMPLEMENTATION OF THE DECODERS

We describe the detail of the implementation of our model, training process, and decoders for the reference. We chose rectified linear units $[\mathrm{ReLU}(x) = \max(0,x)]$ and a sigmoid function $[S(x) = 1/(1 + e^{-x})]$ as the activation function

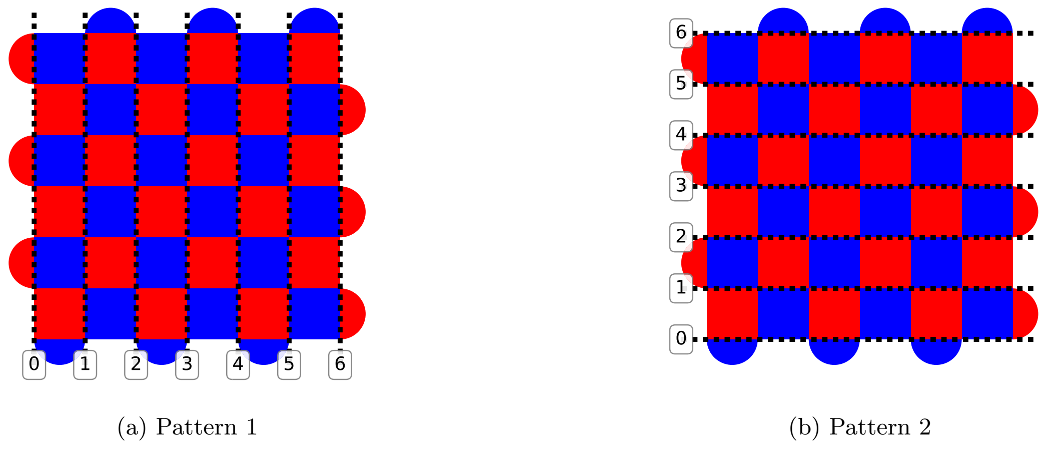


FIG. 17. Logical operators used for the construction of a diagnosis matrix for the surface codes $[[d^2, 1, d]]$. Each dotted black line corresponds to a chosen logical operator.

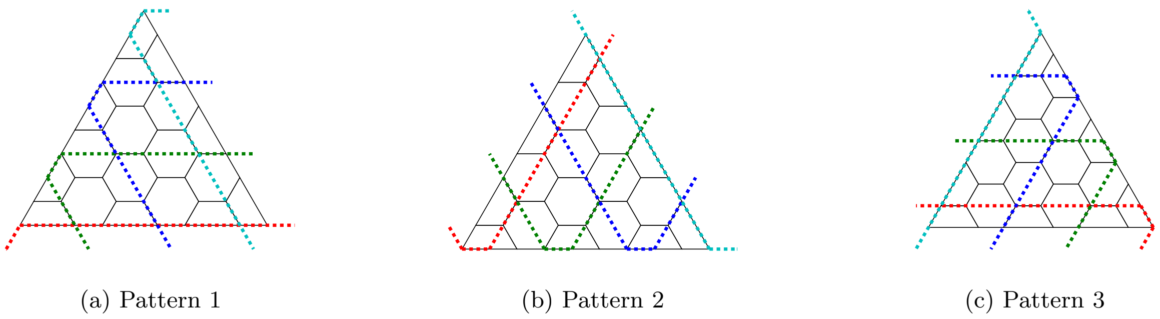


FIG. 18. Logical operators used for the construction of a diagnosis matrix for the [6,6,6]-color codes. Each colored line corresponds to chosen logical operators. The lines are colored only for visibility and are not related to the colors of color codes.

for the hidden layer and that for the final output layer, respectively. Batch normalization was deployed in all of our models and was found to be effective. We also used L2 regularization to avoid overfitting of the model. In the training phase, the Adam optimization method [39] was used. The learning rate was exponentially decreased, and its schedule was optimized by hand. The network was built with the tensorflow v1.2 platform.

1. Details about the multilayer perceptron

We optimized the following parameters of multilayer perceptron using grid search: number of neurons per layer (#unit), number of hidden layers (#layer), size of the batch (#batch), and coefficients of the L2 regularization (β). The parameters were searched in the range #unit $\in \{d^2, d^3, d^4\}$, $\beta \in \{0, 0.01, 0.1\}$, $\#batch \in \{100, 500\}$, and #layer $\in \{2, 3, 4\}$. Note that in the case of $d = 11$, we tuned #unit by hand since we cannot choose #unit $= d^4$ due to the memory limit of the GPU. We started the training with learning rate 10^{-3}, and it was decreased to 10^{-5} according to a schedule which was optimized by hand. We optimized these parameters for each construction of the diagnosis matrix, distance, physical error probability, error model, and size of the training data set. We chose the configuration which achieves the smallest logical error probability for an independently generated validation data set of size 10^5. Then the logical error probability is calculated using another 10^6 test data set.

2. Details about the Convolutional Neural Network

Our CNN model consists of three convolutional layers on top of a single fully connected hidden layer. For each convolutional layer, the channel number was chosen to be $10d$ for the first two layers and $5d$ for the last layer. We chose batch size as 100 in the training of the CNN model.

The network architecture was the same for both bit-flip and depolarizing noise models in the $[[2d^2 - 2d + 1, 1, d]]$ surface code and is described in Table I. As for the $[[d^2, 1, d]]$ code with the bit-flip and depolarizing noise models, we used the network architecture described in Table II. The filter stride was set to 1 in all directions.

3. Implementation of the minimum-distance decoder

The minimum-distance decoder of the surface code under the bit-flip noise can be implemented by reducing the problem into the minimum-weight perfect matching. The minimum-weight perfect matching can be efficiently solved with the Blossom algorithm [31]. We used Kolmogorov's implementation of the Blossom algorithm [40]. In the other cases, we reduced the problem to the following instance of integer programming:

$$\text{Minimize } w(\boldsymbol{e}) \text{ s.t. } H_c \Lambda \boldsymbol{e}^{\mathrm{T}} = s. \tag{B1}$$

This problem was solved with IBM ILOG CPLEX. We obtained at least 10^5 samples for each plot. In all the cases, the solver reached the optimal solution.

4. Time for single prediction, implementation, and environment

We measured the time for single decoding on the $[[2d^2 - 2d + 1, 1, d]]$ surface code with $d = 11$ and $p = 0.15$ under the depolarizing noise for the MD decoder, MWPM decoder, and the proposed neural decoders with the MLP and CNN models. Note that the times of the MD decoder and the MWPM decoder depend on the physical error probability.

We used IBM ILOG CPLEX via python wrapper for constructing the MD decoder. The program was executed on an Intel Xeon E5-2687W v4 with default settings. The MD decoder takes about 330 ms per decoding. Note that the time may be improved by optimizing the settings of CPLEX.

The Kolmogorov's implementation of the Blossom algorithm [31,40] was used for the MWPM decoder. We compiled the codes with Microsoft Visual C++ 2015 and with the O2 option. The program was executed on an Intel Core i7-6700 without parallelization. The MWPM decoder took about 56 μs per decoding.

The proposed neural decoders were implemented with python and tensorflow. We measured the time for single prediction when we set batch size as 1, the number of layer as 2, the number of units per layer as 7000 for the MLP model. The configuration of the CNN model is shown in Table I. The computation was performed using an Intel Core i7-6700 and GeForce GTX 1060 6 GB. The proposed neural decoders with the MLP and CNN models took 2.2 ms and 7 ms, respectively, for feed forwarding the input data and finding the most probable class $\boldsymbol{w}$. Since the prediction of the neural decoders can be done with simple matrix multiplications, we expect that the time for single prediction of the neural decoder can be made shortened by using an optimized hardware such as FPGA.

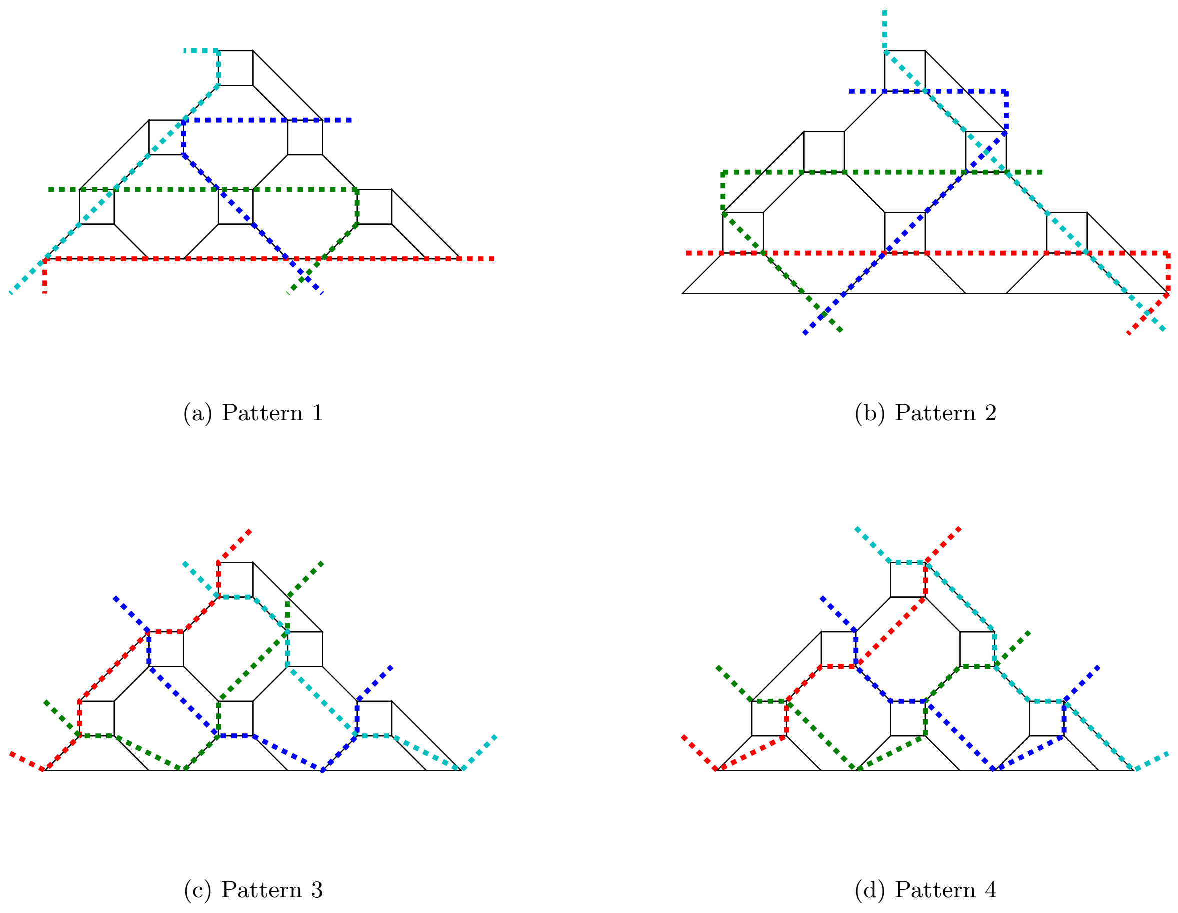


FIG. 19. Logical operators used for the construction of a diagnosis matrix for the [4,8,8]-color codes. Each colored line corresponds to chosen logical operators. The lines are colored only for visibility and are not related to the colors of color codes.

APPENDIX C: THE SPECIFIC CHOICES OF THE UNIFORM DATA CONSTRUCTION

We have introduced the uniform data construction in Sec. III. In this Appendix we show specific uniform data construction for the surface and color codes.

We choose $3d$ logical operators for the $[[2d^2 - 2d + 1, 1, d]]$ surface code using two patterns as shown in Fig. 16. For pattern 1, each dotted line corresponds to a logical X operator, which is the product of the Pauli Z operators on the vertices on the line. For pattern 2, each dotted line corresponds to a logical Z operator, which is the product of the Pauli X operators on the vertices on the line. We choose d logical Y operators written as the product of the ith logical X operator and the ith logical Z operator for $i = 0, \dots, d-1$. We choose $3d$ logical operators for the $[[d^2, 1, d]]$ surface code with two patterns as shown in Fig. 17. The rule of choice is the same as that of the $[[d^2, 1, d]]$ surface code.

We choose $\frac{9}{2}(d+1)$ logical operators for the [6,6,6]-color code as shown in Fig. 18. There are $\frac{1}{2}(d+1)$ lines for each pattern. In all three patterns, each line corresponds to the logical X, Z, and Y operators on the physical qubits on the line. We choose $6(d+1)$ logical operators for the [4,8,8]-color code as shown in Fig. 19. There are $\frac{1}{2}(d+1)$ lines for each pattern. The choice of the logical operators is the same as that of the [6,6,6]-color codes.

In all the patterned choice of the logical operators, we can verify that the sensitivity is constant, since every physical qubit is measured by at most constant number of logical operators. On the other hand, the minimum boundary distance is scaled as $O(d)$, since the same number $O(d)$ of logical X, Y, and Z operators are used. Thus, the normalized sensitivity is scaled as $O(d^{-1})$ with these choices.

APPENDIX D: SIMPLE EXAMPLE FOR OUR NOTATIONS

For the readability of notations introduced in Sec. II, we show a simple example with $d = 3$ surface codes shown in Fig. 20. To identify each physical qubit and stabilizer operator, numbers are assigned for each qubit and stabilizer operators uniquely. In this code, a generator set of stabilizer operators $\{S_i\}$ is as follows:

$$S_0 = X_0X_3, \tag{D1}$$

$$S_1 = X_1X_2X_4X_5, \tag{D2}$$

$$S_2 = X_3X_4X_6X_7, \tag{D3}$$

$$S_3 = X_5X_8, \tag{D4}$$

$$S_4 = Z_1Z_2, \tag{D5}$$

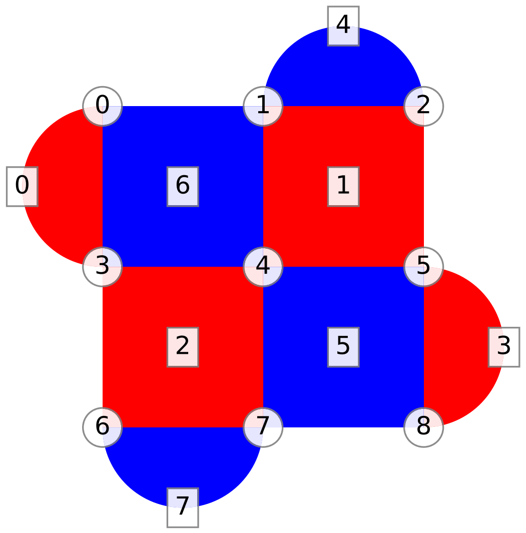


FIG. 20. A picture of surface code $[[d^2, 1, d]]$ in the case of $d = 3$. Integers inside circled white boxes mean identification numbers for data qubits, and those inside squared boxes for stabilizer operators.

$$S_5 = Z_4Z_5Z_7Z_8, \tag{D6}$$

$$S_6 = Z_0Z_1Z_3Z_4, \tag{D7}$$

$$S_7 = Z_6Z_7. \tag{D8}$$

Using the binary representation, the ith row vector of a check matrix H_c is $b(S_i)$. For example, $b(S_0)$ is an 18-bit binary vector, where the zeroth and third bits are unity and the others are zero. The normalizer of stabilizer operators $\mathcal{L}$ is a set of Pauli operators which commute with all the stabilizer operators. For example, $X_L = X_0X_1X_2$ and $Z_L = Z_0Z_3Z_6$ are commute with all the stabilizer operators. Furthermore, since X_L and Z_L are not in the stabilizer group, these are logical operators. We can construct a generator matrix G as a 2×18 matrix where the first row vector is $b(Z_L)$ and the second row is $b(X_L)$.

Suppose we got Pauli operator $E = X_1$ as an error. Its binary representation $\boldsymbol{e} = b(E)$ is given by an 18-bit binary vector where only the first bit is unity. In this case, the fourth and sixth values of syndrome vector $\boldsymbol{s}(\boldsymbol{e})$ are unity and the others are zero, since S_4 and S_6 anticommute with E. The most simple definition of the pure error function $\boldsymbol{t}(\boldsymbol{s}(\boldsymbol{e}))$ is relating the ith syndrome bit to a physical error which anticommutes with S_i and commutes with the other elements in $\{S_i\}$. We can easily find a physical error T_i which anticommutes only with S_i. Here we choose $T_4 = X_2$ and $T_6 = X_0$, for example. Then $\boldsymbol{t}(\boldsymbol{s})$ can be constructed as a binary representation $E' = \prod_i T_i^{s_i} = X_0X_2$. With the fixed pure error function and generator matrix, there is a unique decomposition of error $\boldsymbol{e}$ to binary addition of three vectors $\boldsymbol{l}(\boldsymbol{e}) = \boldsymbol{0}$, $\boldsymbol{w}(\boldsymbol{e}) = (0, 1)^{\mathrm{T}}$, and $\boldsymbol{t}(\boldsymbol{s}(\boldsymbol{e}))$ as written in Eq. (10). Since the operator E' will flip all the flipped syndrome values again, the syndrome values for the Pauli operator $E'E = X_0X_1X_2$ is in the normalizer of the stabilizer group $\mathcal{L}$, and thus $\boldsymbol{t}(\boldsymbol{s}(\boldsymbol{e})) \oplus \boldsymbol{e} \in b(\mathcal{L})$. This means it is trivial task to find a Pauli operator which moves an error operator back to any element in $\mathcal{L}$. On the other hand, $\boldsymbol{t}(\boldsymbol{s}(\boldsymbol{e})) \oplus \boldsymbol{e}$ is not necessarily in the stabilizer group $b(\mathcal{L}_I)$. In this example, $E'E = X_0X_1X_2 = X_L$ is a case where the result is not a stabilizer operator. Since decoding will success when $\boldsymbol{t}(\boldsymbol{s}(\boldsymbol{e})) \oplus \boldsymbol{e} \in b(\mathcal{L}_I)$, our task is to find a binary representation of recovery operator $\boldsymbol{r}(\boldsymbol{s})$ as a function of observable syndrome values $\boldsymbol{s}$ which takes a resultant operator to any element in $b(\mathcal{L}_I)$. The failure probability of this recovery operation is denoted as logical error probability p_{L}.

An optimal decoder can be explained as follows. There are several physical errors which result in the same $\boldsymbol{s}$. For example, physical errors X_0X_2 and X_1 will generate the same syndrome values. As introduced in Eq. (10), any physical error $\boldsymbol{e}$ can be obtained by the decomposition $\boldsymbol{l}(\boldsymbol{e})$, $\boldsymbol{w}(\boldsymbol{e})$, and $\boldsymbol{t}(\boldsymbol{s}(\boldsymbol{e}))$. Thus, possible physical error under an observation of syndrome values $\boldsymbol{s}$ can be classified according to $\boldsymbol{w}(\boldsymbol{e})$ obtained as a result of decomposition. In this case, there are four classes since $\boldsymbol{w}(\boldsymbol{e})$ is a two-bit binary vector. If two physical errors are classified to the same class, whether decoding is success or not is always the same. Thus, the decoding problem is equivalent to choosing a class of which the sum of probabilities of physical errors is maximum. A value $\boldsymbol{q_s}(\boldsymbol{w})$ is the sum of weight for a class $\boldsymbol{w}$. In the above example, physical errors like X_1 and $X_2X_4X_5$ are in the same class and have the maximum value. Thus, any of them can properly correct the occurred error. On the other hand, a minimum distance decoder tries to find a physical error of a maximum probability among possible errors under the observation of syndrome values $\boldsymbol{s}$. In the above example, physical error X_1 is the unique physical error that generates $\boldsymbol{s}$ and has the maximum error probability.

[1] A. Yu Kitaev, Quantum computations: Algorithms and error correction, Russ. Math. Surv. **52**, 1191 (1997).

[2] D. Aharonov and M. Ben-Or, Fault-tolerant quantum computation with constant error, in *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing* (ACM, New York, 1997), pp. 176–188.

[3] E. Knill, R. Laflamme, and W. H. Zurek, Resilient quantum computation: Error models and thresholds, Proc. R. Soc. London A **454**, 365 (1998).

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

[5] A. D Córcoles, E. Magesan, S. J. Srinivasan, A. W. Cross, M. Steffen, J. M. Gambetta, and J. M. Chow, Demonstration of a quantum error detection code using a square lattice of four superconducting qubits, Nat. Commun. **6**, 6979 (2015).

[6] D. Ristè, S. Poletto, M.-Z. Huang, A. Bruno, V. Vesterinen, O.-P. Saira, and L. DiCarlo, Detecting bit-flip errors in a logical qubit using stabilizer measurements, Nat. Commun. **6**, 6983 (2015).

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

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

[9] D. A. Lidar and T. A. Brun, *Quantum Error Correction* (Cambridge University Press, Cambridge, 2013).

[10] S. B. Bravyi and A. Y. Kitaev, Quantum codes on a lattice with boundary, arXiv:quant-ph/9811052 (1998).

[11] 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).

[12] C. Wang, J. Harrington, and J. Preskill, Confinement-Higgs transition in a disordered gauge theory and the accuracy threshold for quantum memory, Ann. Phys. **303**, 31 (2003).

[13] D. S Wang, A. G Fowler, and L. C. L. Hollenberg, Surface code quantum computing with error rates over 1%, Phys. Rev. A **83**, 020302(R) (2011).

[14] 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).

[15] A. M. Stephens, Fault-tolerant thresholds for quantum error correction with the surface code, Phys. Rev. A **89**, 022321 (2014).

[16] M.-H. Hsieh and F. Le Gall, Np-hardness of decoding quantum error-correction codes, Phys. Rev. A **83**, 052331 (2011).

[17] H. Bombin and M. A. Martin-Delgado, Homological error correction: Classical and quantum codes, J. Math. Phys. **48**, 052105 (2007).

[18] N. Delfosse, Decoding color codes by projection onto surface codes, Phys. Rev. A **89**, 012317 (2014).

[19] G. Duclos-Cianci and D. Poulin, Fast Decoders for Topological Quantum Codes, Phys. Rev. Lett. **104**, 050504 (2010).

[20] E. Magesan, J. M. Gambetta, A. D. Córcoles, and J. M. Chow, Machine Learning for Discriminating Quantum Measurement Trajectories and Improving Readout, Phys. Rev. Lett. **114**, 200501 (2015).

[21] G. Carleo and M. Troyer, Solving the quantum many-body problem with artificial neural networks, Science **355**, 602 (2017).

[22] J. Carrasquilla and R. G. Melko, Machine learning phases of matter, Nat. Phys. **13**, 431 (2017).

[23] J. Romero, J. P. Olson, and A. Aspuru-Guzik, Quantum autoencoders for efficient compression of quantum data, Quantum Sci. Tech. **2**, 045001 (2017).

[24] G. Torlai and R. G. Melko, Neural Decoder for Topological Codes, Phys. Rev. Lett. **119**, 030501 (2017).

[25] S. Varsamopoulos, B. Criger, and K. Bertels, Decoding small surface codes with feedforward neural networks, Quantum Sci. Tech. **3**, 015004 (2017).

[26] 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).

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

[28] N. P. Breuckmann and X. Ni, Scalable neural network decoders for higher dimensional quantum codes, Quantum **2**, 68 (2018).

[29] D. Gottesman, Stabilizer codes and quantum error correction, arXiv:quant-ph/9705052 (1997).

[30] G. Duclos-Cianci and D. Poulin, A renormalization group decoding algorithm for topological quantum codes, in *Information Theory Workshop (ITW), 2010 IEEE* (IEEE, Dublin, 2010), pp. 1–5.

[31] J. Edmonds, Paths, trees, and flowers, Can. J. Math. **17**, 449 (1965).

[32] C. M. Bishop, *Pattern Recognition and Machine Learning* (Springer, Berlin, 2006).

[33] K. Hornik, M. Stinchcombe, and H. White, Multilayer feedforward networks are universal approximators, Neural Netw. **2**, 359 (1989).

[34] S. Roman, S. Axler, and F. W. Gehring, *Advanced Linear Algebra*, Vol. 3 (Springer, New York, 2005).

[35] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, Object recognition with gradient-based learning, in *Shape, Contour and Grouping in Computer Vision* (Springer, Berlin, 1999), pp. 319–345.

[36] C. Cafaro and S. Mancini, Quantum stabilizer codes for correlated and asymmetric depolarizing errors, Phys. Rev. A **82**, 012306 (2010).

[37] A. M. Stephens, W. J. Munro, and K. Nemoto, High-threshold topological quantum error correction against biased noise, Phys. Rev. A **88**, 060301(R) (2013).

[38] D. K. Tuckett, A. S. Darmawan, C. T. Chubb, S. Bravyi, S. D. Bartlett, and S. T. Flammia, Tailoring Surface Codes for Highly Biased Noise, Phys. Rev. X **9**, 041031 (2019).

[39] D. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980 (2014).

[40] V. Kolmogorov, Blossom V: A new implementation of a minimum cost perfect matching algorithm, Math. Prog. Comput. **1**, 43 (2009).