IEEE Transactions on Quantum Engineering

Received 19 September 2023; revised 18 June 2024; accepted 20 June 2024; date of publication 27 June 2024; date of current version 11 September 2024.

*Digital Object Identifier 10.1109/TQE.2024.3419773*

# Convolutional Neural Decoder for Surface Codes

**HYUNWOO JUNG, INAYAT ALI, AND JEONGSEOK HA (Senior Member, IEEE)**

School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea

Corresponding author: Jeongseok Ha (e-mail: jsha@kaist.edu).

This work was supported in part by the National Research Foundation of Korea under Grant 2022M1A3C2069728 funded by the Ministry of Science and ICT, in part by the Future Space Education Center, and in part by BK21 FOUR (Connected Artificial Intelligence Education and Research Program for Industry and Society Innovation, School of Electrical Engineering, Korea Advanced Institute of Science and Technology EE) under Grant 4120200113769.

**ABSTRACT** To perform reliable information processing in quantum computers, quantum error correction (QEC) codes are essential for the detection and correction of errors in the qubits. Among QEC codes, topological QEC codes are designed to interact between the neighboring qubits, which is a promising property for easing the implementation requirements. In addition, the locality to the qubits provides unusual tolerance to local errors. Recently, various decoding algorithms based on machine learning have been proposed to improve the decoding performance and latency of QEC codes. In this work, we propose a new decoding algorithm for surface codes, i.e., a type of topological codes, by using convolutional neural networks (CNNs) tailored for the topological lattice structure of the surface codes. In particular, the proposed algorithm takes advantage of the syndrome pattern, which is represented as a part of a rectangular lattice given to the CNN as its input. The remaining part of the rectangular lattice is filled with a carefully selected incoherent value for better logical error rate performance. In addition, we introduce how to optimize the hyperparameters in the CNN, according to the lattice structure of a given surface code. This reduces the overall decoding complexity and makes the CNN-based decoder computationally more suitable for implementation. The numerical results show that the proposed decoding algorithm effectively improves the decoding performance in terms of logical error rate as compared to the existing algorithms on various quantum error models.

**INDEX TERMS** Convolutional neural network (CNN), decoding algorithm, lattice structure, surface codes, topological quantum error correction codes.

## I. INTRODUCTION

Qubit is the basic unit of information storage in quantum computers, and it is vulnerable to noise, even when a quantum computational process is halted or not applied [1]. Due to the unreliable characteristic associated with the qubits, quantum error correction (QEC) codes are essential in quantum processing to protect the information contained in the corresponding qubits. In quantum computers, perfect fidelity in quantum processing is required to achieve scalability. Therefore, designing good QEC codes for protecting quantum information is an important and challenging task for building an efficient and universal quantum computer [1], [2], [3], [4]. Topological QEC codes, such as surface codes and color codes, are designed according to the interaction between the qubits physically close to each other. Since the locality of topological codes is especially useful for easing implementation and improving fault tolerance, these codes have recently attracted a great deal of attention [4], [5], [6], [7], [8], [9], [10], [11].

In the decoding process of QEC codes, errors are detected and corrected based on syndromes, which are calculated and measured from the stabilizer. Since the errors continue to accumulate in quantum processing, it is important to keep the latency of the decoding algorithm low, with good logical error rate performance. In [12] and [13], the minimum weight perfect matching (MWPM) algorithm for decoding the surface codes is presented. A real-space renormalization group (RG) algorithm is proposed in [14] to reduce the complexity of MWPM. The RG algorithm combines the real-space renormalization methods and the belief propagation algorithm (by using mean-field equations). By taking into account the correlations between bit- and phase-flip errors

in the qubits, a Markov chain Monte Carlo algorithm is proposed in [15] to achieve lower logical error rates than those of MWPM.

### A. PREVIOUS MACHINE-LEARNING-BASED DECODERS

Recently, machine-learning-based decoding algorithms using restricted Boltzmann machines (RBMs), feedforward neural networks (FFNNs), and convolutional neural networks (CNNs) are proposed in [16], [17], [18], and [19]. These algorithms take the syndromes as their input and produce the recovery operators as their output. Both the decoding algorithms showed a lower logical error rate than that of the MWPM. However, there are two major disadvantages of these algorithms: 1) the decoding latency is not constant since the decoding process is repeated until the recovery operator corresponding to the syndromes is found and 2) the number of nodes in the output layer is often excessively large, which makes the neural network dense and, thus, highly complex. In [20], [21], and [22], these disadvantages are solved by using the concept of pure error of the surface codes. The decoding problem can be reduced to a classification problem by using a neural network. All the error operators are classified into several categories by adopting pure errors, and the output of the neural network is replaced with the value of the error operator in the corresponding category. In [20], an FFNN is used to train and classify error operators in the decoding algorithm, whereas recurrent neural networks (RNNs) and the CNN are used in [21] and [22], respectively. Nonetheless, all these machine-learning-based decoding algorithms are proposed without considering the topological lattice structure of the surface codes.

### B. MAIN CONTRIBUTIONS

In this work, we propose a new decoding algorithm based on the CNN that is designed to take the topological lattice structure of the surface codes into the decoding. In particular, the existing decoding algorithms ignore the 2-D structure of syndrome pattern and simply transform the syndromes into a 1-D or 2-D bit string for the input to the neural network. Unlike the existing algorithms, we maintain the structure of the syndrome pattern in such a way that the syndrome pattern is first placed into a rectangular lattice without disturbing the structure of pattern. Then, the missing pieces in the lattice are filled with an incoherent symbol.

Moreover, no method is presented in the existing decoding algorithms for determining various hyperparameters used in the neural network. They simply use Bayesian optimization for determining the hyperparameters instead of analyzing possible syndrome patterns. In the proposed decoding algorithm, we determine the hyperparameter values according to the structural features of syndrome patterns. By carefully selecting these hyperparameter values, the overall decoding complexity is minimized without compromising the error rate performance. Through numerical simulations, we show that the logical error rate performance of the proposed decoding algorithm is better than that of the existing decoding algorithms in the depolarizing error model, depolarizing with the measurement error model, and the circuit noise error model.

The rest of this article is organized as follows. In Section II, a brief description of the stabilizer and surface codes is given. A preview of the decoding algorithms of these codes and the introduction of the CNN are also presented in this section. The existing machine-learning-based decoding algorithms of QEC codes are described in Section III. We present the proposed CNN-based decoding algorithm that utilizes the features of syndrome pattern induced by the structure of surface codes in Section IV. We then introduce how to determine the hyperparameters and explain the method of selecting the training samples. In Section V, we present the numerical simulation results and analyze the performance of the proposed and existing decoding algorithms in the depolarizing error model, depolarizing with the measurement error model, and the circuit noise error model. Finally, Section VI concludes this article.

## II. BACKGROUND

### A. STABILIZER CODES

In an $[\![n, k, d]\!]$ QEC code, $k$ logical qubits with quantum information/states are encoded into a QEC codeword of length $n$ to protect them from errors during quantum processing, and $d$ is the code distance, which will be defined shortly. The logical and physical qubits are also called message and data qubits, respectively. The stabilizer codes are a class of QEC codes that has recently attracted the most attention for the reliable processing of quantum information/states [3], [4], [5], [6], [7], [8], [9], [10], [11].

An $[\![n, k, d]\!]$ stabilizer code is defined over a stabilizer group $\mathcal{S}$, which is the Abelian subgroup of Pauli group $\mathcal{P}^{\otimes n}$ and does not include $-I^{\otimes n}$ as a group element (here, $I$ is the $2 \times 2$ identity matrix in $\mathcal{P}$). The code space $\mathcal{C}$ of a stabilizer code is defined as follows:

$$\mathcal{C} = \{|\psi\rangle \mid S|\psi\rangle = |\psi\rangle \ \forall\, S \in \mathcal{S}\}.$$

The stabilizer group $\mathcal{S}$ consists of $n-k$ independent generators $S_i$, $i = 1, \ldots, n-k$. The elements in $\mathcal{S} = \langle S_1, S_2, \ldots, S_{n-k}\rangle$ are called stabilizers and have the same effect on data qubits, even though each of them is a different operator (strictly speaking, the stabilizers do not change the states of the data qubits).

The logical operators are defined as Pauli operators, which commute with all the elements in $\mathcal{S}$. We define the centralizer $C(\mathcal{S})$ of the stabilizer group $\mathcal{S}$ as follows:

$$C(\mathcal{S}) = \{L \mid LS = SL \forall\, S \in \mathcal{S}\}$$

where $L$ is a logical operator, and we can note that any nontrivial logical operator is an element in $C(\mathcal{S})\backslash\mathcal{S}$ and vice versa.[1] Since the stabilizers do not affect the data qubits, any two logical operators $L_a$ and $L_b$ are called equivalent if

[1] In general, stabilizers can be viewed as trivial logical operators.

$L_a L_b \in \mathcal{S}$. We define the logical state group as follows:

$$\mathcal{L} = \langle L_1, L_2, \ldots, L_{2k} \rangle$$

where $L_i$, $i = 1, 2, \ldots, 2k$, are logical operators that are independent and not equivalent to each other. To find the logical state group, we consider the quotient group $\mathcal{C}(\mathcal{S})/\mathcal{S}$ as follows:

$$\mathcal{C}(\mathcal{S})/\mathcal{S} = \langle Q_1, Q_2, \ldots, Q_{2k} \rangle$$

where $Q_i$, $i = 1, 2, \ldots, 2k$, are independent cosets. Since all the elements in each coset are equivalent to each other, $L_i$ can be any element in coset $Q_i$. In general, $L_i$, $i = 1, 2, \ldots, 2k$, are represented by $\bar{X}_1, \bar{X}_2, \ldots, \bar{X}_k, \bar{Z}_1, \bar{Z}_2, \ldots, \bar{Z}_k$, where $\bar{X}_j$ anticommutes with $\bar{Z}_j$. The code distance $d$ is defined as the minimum weight of all the logical operators (here, the weight is defined as the number of qubits on which the operator causes information loss).

Let $T_i$ be the pure error element in $\mathcal{T} = \langle T_1, T_2, \ldots, T_{n-k} \rangle$. $T_i$ anticommutes with the stabilizer element $S_i$, and it is an operator that commutes with other stabilizer elements $S_j$ : $\forall\ j \neq i$. The $n-k$ pure error elements in $\mathcal{T}$ commute with each other and all the logical operators. By using an $[\![n, k, d]\!]$ stabilizer code, we can successfully correct errors on qubits when the number of errors is less than or equal to $\lfloor \frac{d-1}{2} \rfloor$.

### B. SURFACE CODES

The surface codes are a type of topological stabilizer codes, which are designed to measure all the stabilizer generators by using only local interactions between the neighboring qubits. Different types of surface codes can be defined by the topology of the lattice on the surface, such as toric codes [8], planar codes [9], and rotated surface codes [20], [21], [22], [23]. In this work, we focus only on the rotated surface codes, which require the least physical qubits per logical qubit. For brevity, we refer to rotated surface codes as surface codes throughout this article.

The basic structure of the $[\![n = d^2, k = 1, d]\!]$ surface codes is shown in Fig. 1, where the code distance $d$ amounts to $\sqrt{n}$. The data qubits are placed on the vertex $v$ of the lattice structure, and the ancilla qubits are placed on the faces $f$ of the squares/semicircles in the lattice. The stabilizer generators of surface codes are defined as follows:

$$X_{f_r} := \prod_{v \in f_r} X_v \qquad Z_{f_b} := \prod_{v \in f_b} Z_v \tag{1}$$

where $f_r$ and $f_b$ represent the red and blue planes, respectively. Moreover, $X_v$ (respectively, $Z_v$) performs the $X$ (respectively, $Z$) operation on the qubits placed on the vertex $v$. Note that the qubits placed on the vertex $v$ are called the support of stabilizer. There are $n-1$ independent generators in the surface codes. The logical state group of the surface codes is given by $\mathcal{L} = \{\bar{I}, \bar{X}_1, \bar{Y}_1, \bar{Z}_1\}$, where $\bar{Y}_1 = \bar{X}_1 \bar{Z}_1$ and $\bar{I} \in \mathcal{S}$. Each element of $C(\mathcal{S})$ uniquely corresponds to an element in $\mathcal{L}$.

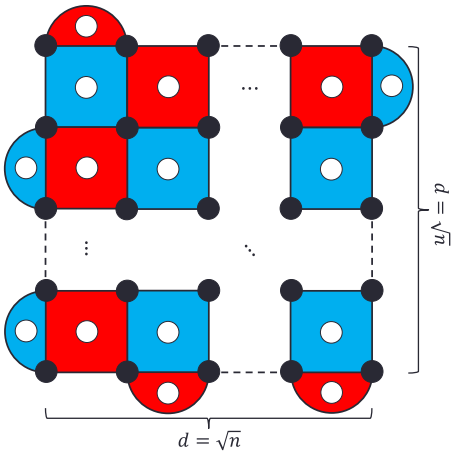



**FIGURE 1. Basic structure of the surface codes. The black circles represent the data qubits, and the white circles represent the ancilla qubits.**

### C. DECODING PROCESS AND ALGORITHMS OF SURFACE CODES

The decoding process of the $[\![n, k, d]\!]$ surface codes involves three steps: 1) detect the errors in qubits; 2) determine the errors; and 3) correct the errors. To detect errors in qubits, we need to measure the quantum states in the qubits. If we apply a direct measurement on a data qubit, the quantum information is lost. Therefore, we need a quantum circuit and apply measurements using an ancilla qubit to detect errors without losing the quantum information in the data qubit. The quantum circuit is also called the syndrome extraction circuit, and the measured values from these circuits are called the syndromes, which are represented in a vector form, $\vec{s} = \{s_1, s_2, \ldots, s_{n-k}\}$. Let $E \notin C(\mathcal{S})$ be an error operator that does not commute for at least one stabilizer element. If $E$ is not a commute for the $i$th stabilizer generator $S_i \in \mathcal{S}$, then the $i$th syndrome $s_i$ is 1, and it is 0 otherwise. The $i$th syndrome is measured by the syndrome extraction circuit corresponding to the $i$th stabilizer generator. The syndrome extraction circuits used by the surface codes are shown in Fig. 2, where there are two types of syndrome extraction circuits. The circuits in Fig. 2(a) and (b) are devised to detect $Z$ and $X$ errors, respectively. Note that in Fig. 2, we measure only the ancilla qubits for extracting the syndrome with a predetermined observable represented by $X$ and $Z$, and thus, the states of data qubits remain intact.[2] Next, the decoding algorithm determines the errors on $\vec{s}$ and computes the recovery operator $R$ to be used for error correction. The maximum likelihood estimation (MLE) algorithm is the optimal decoding algorithm for decoding QEC codes. It calculates the likelihood of a syndrome occurring and finds an operator that maximizes it. The recovery operator determined by MLE

[2]Hermitian operator is used as an observable to specify the measurements. See details in [24, Ch. 4].

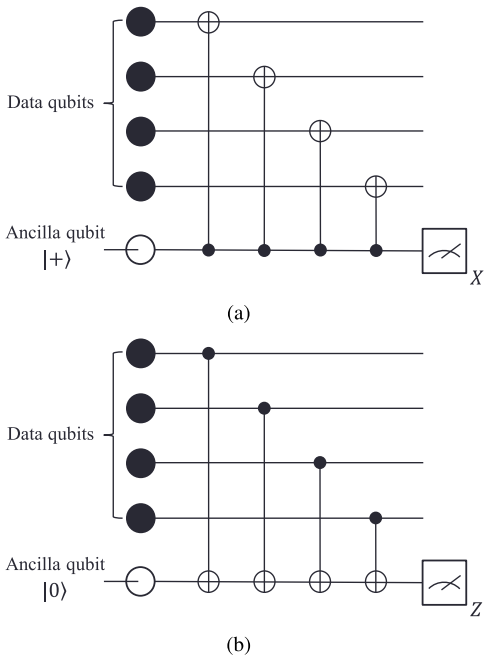



**FIGURE 2.** Syndrome extraction circuits. (a) *Z* error detection circuit. (b) *X* error detection circuit.

is defined as follows:

$$R_{\mathrm{MLE}} = A \cdot \arg\max_{L\in\mathcal{L}} \sum_{S\in\mathcal{S}} \Pr(ALS)$$

where $A$ is an arbitrary operator corresponding to the syndromes, and the sum of probabilities, i.e., $\sum_{S\in\mathcal{S}} \Pr(ALS)$, is equivalent to the likelihood of a syndrome occurring. Although MLE is an optimal decoding algorithm, its computational complexity is often prohibitively expensive for practical consideration. The errors accumulate in data qubits even during the decoding is in progress. Therefore, both the computation and time complexities must be minimized when selecting a decoding algorithm. As a lower complexity solution, one can consider the MWPM whose complexity is known to be $\mathcal{O}(m \cdot d^4 \log d)$ for code distance $d$ and constant $m$ [25]. The MWPM is, however, suboptimal in terms of error rate performance, since it does not take into account all the possible operators. In addition, the MWPM independently deals with $X$ and $Z$ errors and, thus, is susceptible to $Y$ errors.

## III. MACHINE-LEARNING-BASED DECODING ALGORITHMS

The machine-learning-based decoding algorithms are classified into two classes, and the block diagrams of these two classes are shown in Fig. 3(a) and (b), respectively. To test whether an error $E$ occurs in the data qubits, the syndrome vector $\vec{s}$ is extracted utilizing the extraction circuit. Then, when the syndrome vector $\vec{s}$ is nonzero, the decoding process is performed using a neural network that takes $\vec{s}$ as its input. The errors in the data qubits are corrected with the corresponding recovery operator. At this stage, as shown in Fig. 3(a) and (b), the machine-learning-based decoding algorithms are divided into two classes according to the output of the neural network. We refer the ones in Fig. 3(a) and (b) to as *low*- and *high*-level decoding algorithms, respectively, depending on the type of output of the neural network. The low-level decoding algorithm uses the output of the neural network (i.e., an array of bits) as the recovery operator to the data qubits, whereas, in the high-level decoding algorithm, the neural network finds the logical state of the recovery operator to the data qubits. The details of the low- and high-level algorithms will be given in the subsequent sections.

### A. LOW-LEVEL DECODING ALGORITHM

In the low-level decoding algorithm, the output of the neural network is a bit array representing the recovery operator. The bit array of the recovery operator is divided into two strings of bits that represent $X$ and $Z$ errors. In case of an error, the value of the bit is 1 and is 0 otherwise. The $Y$ error occurs when the $X$ and $Z$ errors occur at the same time. Therefore, both the bits are 1 when a $Y$ error occurs. The total number of bits used to express the recovery operator is equal to the number of data qubits multiply by two. We illustrate the recovery operator with the following example:

$$\mathrm{Operator}X_1I_2Y_3Z_4 \;\leftrightarrow\; \mathrm{BitArray}(\underbrace{1,0,1,0}_{\text{X error}},\ \overbrace{0,0,1,1}^{\text{Z error}}).$$

The RBM and FFNN-based low-level decoding algorithms are proposed in [16] and [17], respectively. However, the low-level decoding algorithm has two disadvantages. First, as the code length increases, the number of nodes in the output layer increases, and as a result, the structure of the neural network becomes complex. Moreover, the decoding latency becomes unpredictable. If the syndrome vector $\vec{s_r}$ that corresponds to the recovery operator is not equal to initial syndrome vector $\vec{s}$, we resample the recovery operator. We define $M = \{m|\vec{s_r}(m) \neq s_m\}$ as the set of indices, where $\vec{s_r}$ and $\vec{s}$ do not equal. The operators for the supports of the $m$th stabilizer generator are changed to correspond with $s_m$. Until the recovery operator corresponds to $\vec{s}$, the resampling is recursively repeated.

### B. HIGH-LEVEL DECODING ALGORITHM

The error $E$ that occurs in the surface codes can be decomposed as follows:

$$E = S \cdot T \cdot L \tag{2}$$

where $S \in \mathcal{S}$, $L \in \mathcal{L}$, and $T$ is the product of elements in $\mathcal{T}$. For the syndromes in $\vec{s} = \{s_1, s_2, \ldots, s_{n-k}\}$ corresponding to the error $E$, $T$ is uniquely determined as follows:

$$T = \prod_{i=1}^{n-k} T_i^{s_i}$$

where $T_i^0 = I^{\otimes n}$ and $T_i^1 = T_i$ for $i = 1, 2, \ldots, n-k$. Since there are $n-k$ pure errors in $\mathcal{T}$, we can easily arrange them

IEEE Transactions on Quantum Engineering

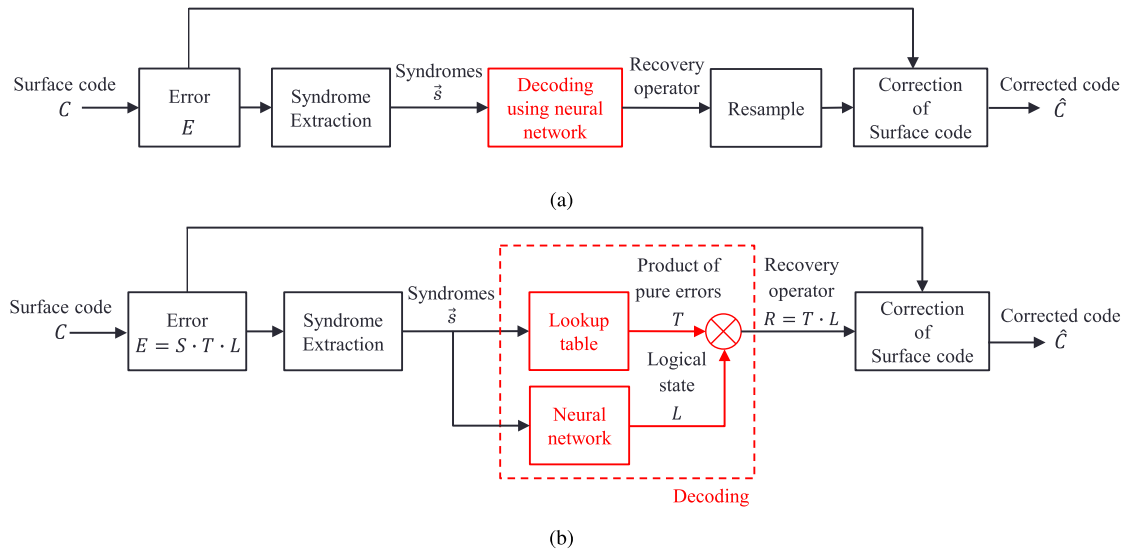



**FIGURE 3.** Block diagram of (a) low-level decoding algorithm and (b) high-level decoding algorithm.

in a lookup table. For a given $T$, the stabilizer $S$ and the logical state $L$ satisfying the equality in (2) can also be uniquely determined. That is, the error $E$ uniquely corresponds to one of the $|\mathcal{L}|$ logical states, which is a classification problem for the errors to the corresponding logical states.

However, in the decoding problem, the error $E$ is not available, and thus, the classification must be carried out with the syndrome vector $\vec{s}$. It has been reported that the classification can be effectively performed with a neural network, and thus, the high-level decoding algorithm employs the neural network for finding the logical state $L$ corresponding to the syndrome vector $\vec{s}$. Note that the mapping from the error $E$ to the syndrome vector $\vec{s}$ is surjective, due to which the classifications with $E$ and $\vec{s}$ may result in different logical states. When errors corresponding to different logical states have the same syndrome vector, the neural network is trained to map the syndrome vector to the logical state for the majority of errors.

Based on given syndromes, the high-level decoding algorithm determines $T$ and $L$ by the lookup table and the neural network, respectively, as shown in Fig. 3(b). We determine the recovery operator as $R = T \cdot L$. Although it is an error different from the actual error (i.e., $E = S \cdot T \cdot L$), the stabilizer $S$ does not affect the states of the data qubits. Note that the high-level decoding algorithm solves the shortcomings of the low-level decoding algorithm. That is, we determine the elements of $\mathcal{L}$ from the output of the neural network, so the number of nodes in the output of the neural network is fixed to $|\mathcal{L}|$ regardless of the code length. Therefore, even if the code length increases, the structure of the neural network does not become dense or complicated. Moreover, since the recovery operator always corresponds to the syndrome vector $\vec{s}$, the decoding latency is constant. FFNN-, RNN-, and CNN-based high-level decoding algorithms are proposed in [20], [21], and [22], respectively.

## IV. PROPOSED CONVOLUTIONAL NEURAL DECODER

Since the surface codes are characterized as topological QEC codes, the syndromes adjacent to each other are highly correlated. The existing decoding algorithms based on machine learning techniques do not take the lattice structure of the surface codes into consideration. Moreover, in the existing CNN-based decoders for QEC codes in [18], [19], and [22], the inputs are modified to separately employ $X$ and $Z$ syndromes. In addition, the input of the CNN-based decoder in [18] is transformed to include data qubits, thereby increasing its complexity. In this section, we propose a high-level CNN-based decoding algorithm, which considers the lattice structure of surface codes for improving the decoding performance and reducing the complexity. We introduce a method for transforming the syndromes into a rectangular structure to be used as an input to the CNN. We also optimize the hyperparameters used in the CNN (i.e., the size and number of filters) considering the lattice structure of the surface codes. Furthermore, a method for the selection of training samples used to train the CNN is introduced.

### A. INPUT TRANSFORMATION OF THE CNN-BASED DECODER

In the syndrome extraction process, the initial syndromes are stacked in a 2-D structure to form a pattern, as shown in Fig. 4(a). The initial syndrome patterns form irregular structures, and we do not use them directly as an input to the CNN. We transform these syndrome patterns into rectangular structures without disturbing the initial placement of the syndromes in the 2-D structure. In Fig. 4(b), an incoherent new value (displayed by white boxes) is assigned to the edge of the syndromes to form a rectangular structure. For the surface codes with code distance $d$, the input size is the same both horizontally and vertically (i.e., the incoherent value introduced in the 2-D structure transforms it into a square). For the

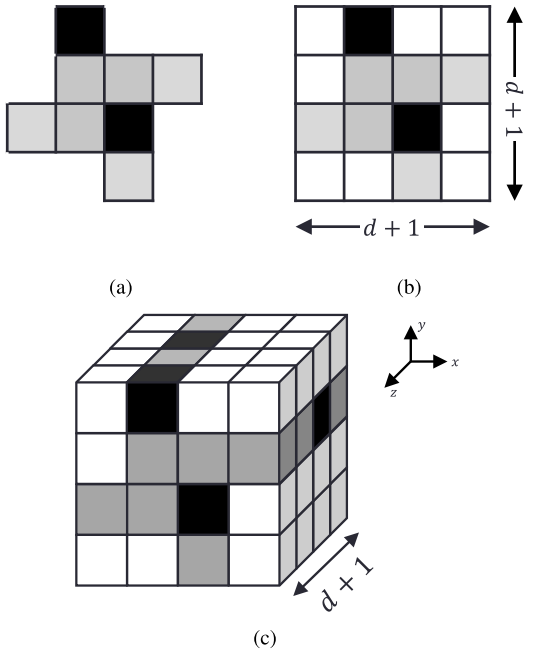



**FIGURE 4.** Input images of the CNN-based decoder. Black and gray boxes represent the syndromes of value 0 and 1, respectively. The white boxes represent an incoherent value.

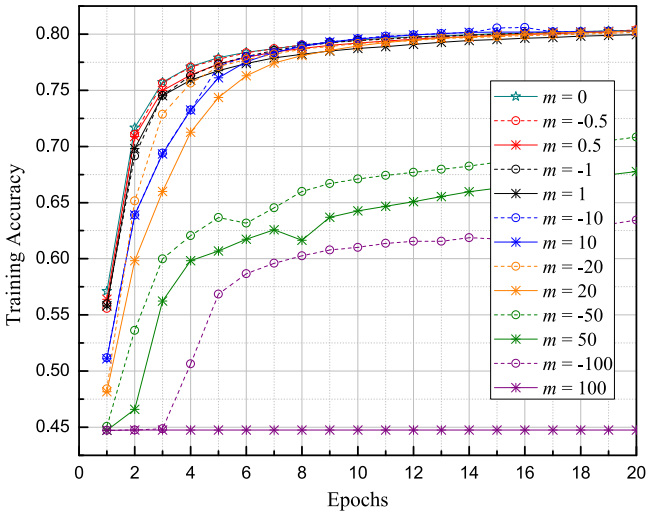



**FIGURE 5.** Comparison of training accuracy for different values of $m$ used in a surface code with code distance $d = 5$.

surface codes, we propose two conditions for the selection of the incoherent value (denoted by $m \in \mathbb{R}$) as follows.

1) $m < 0$: Negative numbers are ignored in the calculation process since the CNN uses the rectified linear unit (ReLu) as the activation function.
2) $|m| < 1$: The input normalization is known in [26] to train the neural network more accurately and faster. Thus, $|m|$ is set to a value between the two syndrome values, i.e., zero and one.

In Fig. 5, we compare the training accuracy of CNNs by using inputs with different values of $m$. We can observe that

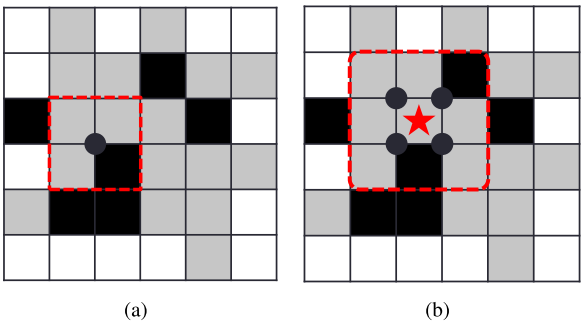



**FIGURE 6.** Size of the filter. The black circle represents the data qubits. (a) (2,2) filter for four correlated syndromes. (b) (3,3) filter for nine correlated syndromes.

the value of $m$ satisfying the proposed conditions exhibits the best accuracy and has fast convergence. Although we can select any arbitrary value of $m$ that satisfies the aforementioned conditions, our selection of $m$ is based on the training and test accuracy of numerical simulations. We adopt $m = -0.5$ as the incoherent value in our proposed decoder, since it exhibits the best training and test accuracy.

For an error model using an ideal syndrome extraction circuit (i.e., with no error occurring during syndrome extraction), one measurement value is used as the input to the decoder, whereas multiple measurement values are used as the input to the decoder when errors occur during the syndrome extraction process, e.g., the depolarization with measurement error and the circuit noise error models in Section V. In Fig. 4(c), several measurements are stacked in the $z$-direction and are used as the input to the decoder with depth $d + 1$. Then, the syndromes in the input not only correlated in the $x$- and $y$-directions but also the $z$-direction.

### B. OPTIMIZATION OF HYPERPARAMETERS

#### 1) SIZE OF THE FILTER $(w_i, h_i)$

The data are processed locally depending on the size of the filters used in a CNN, and therefore, the selection of an appropriate filter size is important to process highly correlated syndromes. We decide the size of the filters by observing the correlation of neighboring syndromes. In other words, we analyze how the effects of errors on the syndromes are localized, which allows us to determine the size of the filters. For example, in Fig. 6(a), it is shown that an error in one qubit affects only four adjacent syndromes due to the topological lattice structure of the surface codes. Meanwhile, in Fig. 6(b), the syndrome labeled with a red star is only affected by the errors in adjacent four data qubits, which affect nearby eight syndromes in the red box. Consequently, local errors only affect adjacent syndromes, and we select filters that extract local features (i.e., syndrome patterns) inside the red boxes, shown in Fig. 6. In this article, we use two filter sizes that fit the scenarios, i.e., (2,2) filter for four correlated syndromes and (3,3) filter for nine correlated syndromes in Fig. 6(a) and (b), respectively.

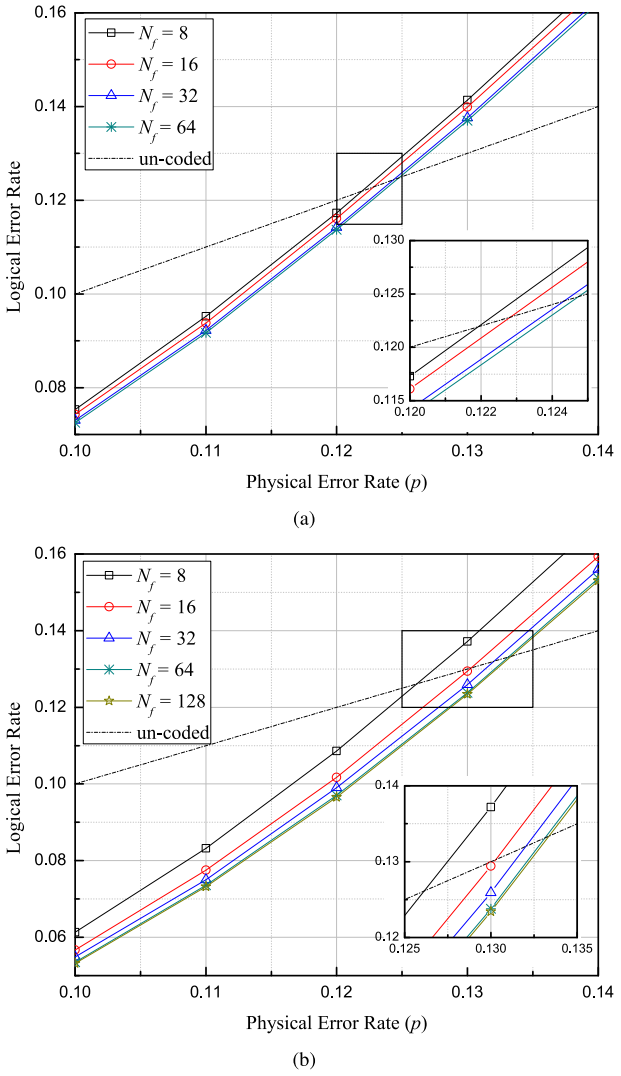



**FIGURE 7.** Comparison of decoding performances for different number of filters $N_f$ and code distance. (a) $d = 5$ and (b) $d = 7$.

### 2) NUMBER OF FILTERS $N_f$

For a low-complexity CNN-based decoder, the selection of the number of filters $N_f$ in the CNN plays a key role in determining the tradeoff between the decoding performance and the computational complexity. Each filter identifies a specific input pattern. It is important to determine the number of filters sufficient to identify all the patterns since using too many filters will complicate the neural network. For surface codes with code distance $d$, the number of all possible patterns of the syndromes is $2^{d^2-1}$. If $d^2 - 1$ independent patterns are identified by a CNN, then all other patterns can be identified by it. Thus, the minimum number of filters is fixed to $d^2 - 1$, and we propose a constraint to optimize the number of filters $N_f$ used in the CNN as follows:

$$N_f = 2^u \text{ subject to } 2^{u-1} < d^2 - 1 \leq 2^u \tag{3}$$

where $u \in \mathbb{N}$ is an arbitrary positive integer. In Fig. 7(a) and 7(b), the error rate performances of surface codes decoded using the proposed CNN-based decoder with different $N_f$ and $d$ are shown. In Fig. 7(a), the error rate is evaluated for $d = 5$, and thus, the constraint defined in (3) is satisfied when $N_f = 32$. It is noticed that the error rate performance gets improved as the number of filters $N_f$ is increased from 8 to 32. Meanwhile, when we further increase the filter size, the performance improvement becomes diminished. Thus, the results in Fig. 7(a) confirm the proposed way to selected the number of filters in (3). In Fig. 7(b), a similar trend in error rate performance is observed.

## C. SELECTION OF TRAINING SAMPLES

After determining the basic structure of the CNN-based decoder and finding the associated values of hyperparameters, we create training samples for training the CNN. We randomly generate error operators and determine syndrome vectors $\vec{s}$ and logical states corresponding to the errors. Then, the training samples consist of pairs of syndrome vectors and corresponding logical states, which are used as inputs and outputs, respectively, during the training of the CNN-based decoder. In the training of the existing neural-network-based decoders, the aforementioned training process should be repeated for a different physical error rate. For the error models changing over time, it is difficult to select a neural network among various trained neural networks.

In other work [22], the neural network is trained by the training samples at a single physical error rate and is tested against a large variety of physical error rates. However, this approach had a higher logical error rate than the former approach, i.e., the one with multiple trained networks. Let $D$ be the set of error operators at a specific physical error rate for a given error model. Then, we define a subset $D_{\vec{s}}$ of $D$ whose elements are the all error operators corresponding to the syndrome vector $\vec{s}$. All the error operators in $D_{\vec{s}}$ are classified into four subclasses, each of which corresponds to one of four logical states. The neural network determines the logical state with the highest set probability for the input $\vec{s}$ as its output, where set probability is the sum of the probabilities of all the elements of a set. The joint probability distribution of the syndrome vector and the corresponding logical state varies at different physical error rates. Thus, the outputs from the neural networks trained at different physical error rates may not be the same, which deteriorates the error rate performance of the decoding scheme in [22], i.e., the one trained at only one physical error rate.

In this work, we train a single neural network by taking and mixing training samples from various physical error rates of interest. Then, the joint probability distribution of the syndrome vector and the logical state for mixed training samples is now turned out to be the average of joint distributions at the physical error rates. By performing the training, the trained network is not biased to the statistical characteristic of the syndrome vector and the logical state pairs at a specific physical error rate. The way to mix the training samples obtained at different physical error rates is also discussed in [27] and [28].

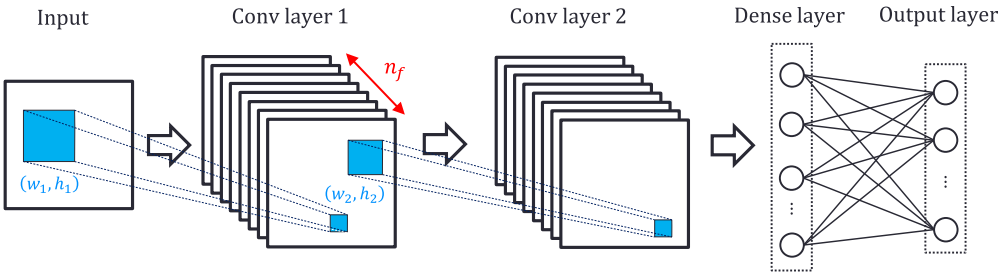



**FIGURE 8.** Structure of the CNN. $(w_1, h_1)$ and $(w_1, h_1)$ are filter size, and $n_f$ is the number of filters.

**TABLE 1.** Numbers of Nodes in the Hidden Layer for the FFNN

| Code distance | Depolarizing | Depolarizing with measurement | Circuit noise |
|---|---|---|---|
| 3 | $128^{\dagger}$ | $768^{\dagger}$ | $704^{\dagger}$ |
| 5 | $660^{\dagger}$ | 200 | 400 |
| 7 | $256^{\dagger}$ | 900 | 600 |

## V. NUMERICAL RESULTS

In this section, we present performance evaluations of the proposed CNN-based decoding algorithm and compare them with those of competing algorithms, i.e., MWPM, FFNN [20], and an existing CNN-based decoding algorithm [22], over three different quantum error models.

1) *Depolarizing error model:* $X$, $Y$, and $Z$ errors occur with equal probability $p/3$ on the data qubits.
2) *Depolarizing with the measurement error model:* The measurement error in syndrome extraction circuits occurs with probability $p$ in addition to the depolarizing errors on data qubits.
3) *Circuit noise error model:* The data qubits along with gates are noisy, whereas preparation and measurement errors also occur. The data qubits are affected by the depolarizing error model. At each single-qubit gate, $X$, $Y$, and $Z$ errors occur with equal probability $p/3$, and at each two-qubit gate, $E \in \mathcal{P}^{\otimes 2} \backslash \mathcal{I}^{\otimes 2}$ errors occur with equal probability $p/15$. Each of the preparation and measurement encounters an error with probability $p$.

The FFNN uses a one hidden layer, and the numbers of nodes in the hidden layer for different combinations of the code distance and the channel model are given in Table 1, where the values designated with the symbol † are taken from [20]. On the other hand, the remaining four values are chosen as the ones with the best error rate performance, i.e., logical error rate, by performing error rate evaluations within the range of 50–1000 by a step of 50. In this work, we assume that all the CNN-based decoding algorithms have the structure in Fig. 8, where there are two convolutional layers with the ReLu activation function. In particular, two types of filters in the proposed CNN-based decoding algorithm are of sizes $(w_1, h_1) = (3, 3)$ and $(w_2, h_2) = (2, 2)$. The numbers of filters of two types are set equal, i.e., $n_f$ in Fig. 8, which depends on the code distance $d$ according to (3). That is, $n_f$ is determined as 8, 32, and 64 for a surface code with $d = 3$, 5, and 7, respectively. The numbers of nodes in the dense and output layers are fixed to 50 and 4, respectively. The number of nodes in the output layer amounts to the number of logical state group $|\mathcal{L}|$. Meanwhile, a good choice of the number of nodes in the dense layer depends on the code distance and/or error model. We find out that the choice of 50 provides reasonable performances for various combinations of the code distance and the error model.

The existing CNN-based decoding algorithm in [22] requires two CNNs in Fig. 8 to decode the $X$ and $Z$ errors in parallel, and the outputs from the two CNNs are later merged. Note that the proposed algorithm needs only one CNN without the merging process. The sizes of two filters are set to $(w_1, h_1) = (3, 3)$ and $(w_2, h_2) = (4, 4)$, which are from [22]. Meanwhile, the numbers of nodes in the dense and output layers are determined as 50 and 2, respectively. Note that the existing CNN-based decoding algorithm separately deals with $X$ and $Z$ errors, and thus, the output layer has two nodes. The numbers of filters are decided in the same way for the proposed algorithm. All the neural networks are trained with more than $N_T = 10^6$ training samples in 0.1 validation split by using the stochastic gradient descent optimizer, cross-entropy loss function, and adaptive learning rate from $10^{-2}$ to $10^{-5}$.

We further evaluate the performance of our proposed CNN-based decoding algorithm in a more realistic setup using the experimental data from the recent experiments conducted by Google Quantum AI [29]. We train our neural networks using the Pauli error model, which is based on the measured error rates of each operation detailed in [29]. The performance evaluations are also compared with various decoding algorithms: MWPM, correlated modification of MWPM [30], belief matching [31], and tensor network [32], which approximates maximum likelihood decoding

In this article, we evaluate and compare the existing and proposed decoding algorithms in terms of logical error rates and pseudothreshold. The pseudothreshold will be shortly defined in Section V-B. Additional comparisons are performed for the proposed and the existing CNN-based decoding algorithms in terms of computational complexity and decoding latency. Complexity and latency are measured by counting the total number of operations and steps required by

the algorithms, respectively, assuming that a parallel decoding architecture is employed in the decoders. In Section V-D, we discuss the scalability of the proposed CNN-based decoding algorithms.

### A. LOGICAL ERROR RATE

Error rate performances of the competing algorithms for the depolarizing, depolarizing with measurement, and circuit noise error models are presented in Figs. 9–11, respectively. For all the error models, the error rate performances are evaluated over a range of physical error rates, which include the pseudothresholds.

In Fig. 9(a), we compare error rate performances of the algorithms with a surface code of $d = 3$. The total number of syndromes is given by $2^{n-k} = 2^{d^2-1} = 2^8$, which is much smaller than the size of training set (i.e., $10^6$). Thus, most of the syndrome patterns are included in the training set, which makes the proposed CNN-based decoding algorithm and FFNN have the same logical error rate performance. However, MWPM has a relatively poorer logical error rate performance since it is more vulnerable to $Y$ errors as compared to the machine-learning-based decoding algorithms. Note that the comparison in Fig. 9(a) does not include the existing CNN-based decoding algorithm since the existing algorithm is not applicable to the surface code of $d = 3$.

In Fig. 9(b), the performance comparison is also conducted with a surface code of $d = 5$, for which the total number of syndromes amounts to $2^{d^2-1} = 2^{24}$ and is more than the number of training samples (i.e., $10^6$). It is observed that the proposed algorithm achieves the best performance. It should be noticed that on the contrary to the comparison in Fig. 9(a), the proposed algorithm has better error rate performance as compared to that of the FFNN when the number of training samples is limited. It is observed that both the existing CNN-based decoding algorithm and MWPM have similar error rate performances, which is due to the fact that both the algorithms separately deal with $X$ and $Z$ and thus are vulnerable to $Y$ errors.

In Fig. 9(c), a surface code of $d = 7$ is considered for comparing performances of the decoding algorithms. The proposed CNN-based decoding algorithm has the best logical error rate, followed by the FFNN-based decoding algorithm. However, it is observed that the logical error rate of the existing CNN-based decoding algorithm becomes poorer than that of the MWPM. The total number of possible syndromes is $2^{d^2-1} = 2^{48}$, which is now much larger than the number of training samples. Due to the limited number of training samples, it is difficult to classify diverse errors.

Figs. 10 and 11 compare the decoding performances of different algorithms for the depolarizing with the measurement error model and the circuit noise error model, respectively. Simulation results in both the error models show similar trends. In particular, for the code distance of $d = 3$ in Figs. 10(a) and 11(a), all the algorithms have almost the same error rate performance. Meanwhile, in the case of $d = 5$

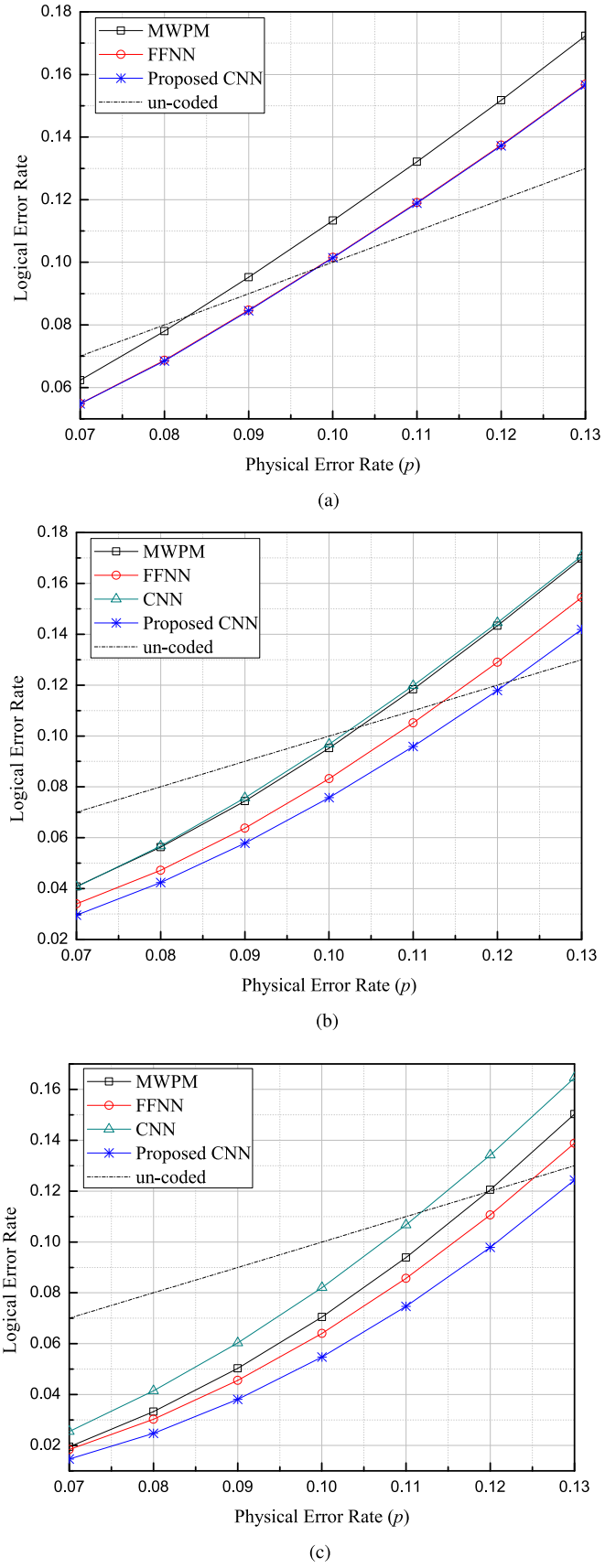
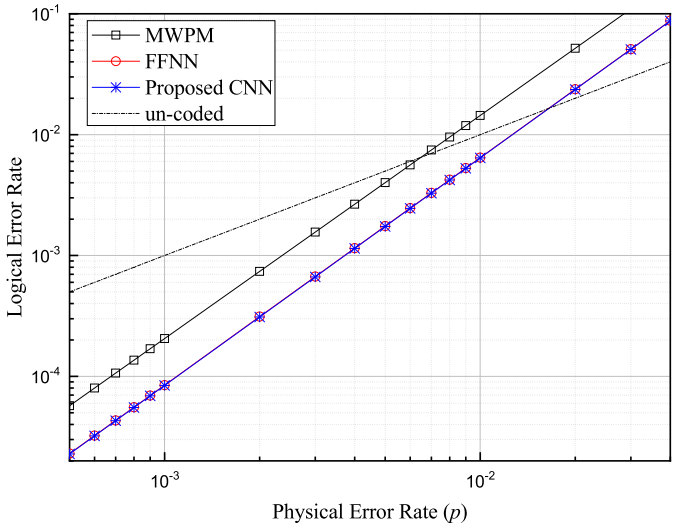
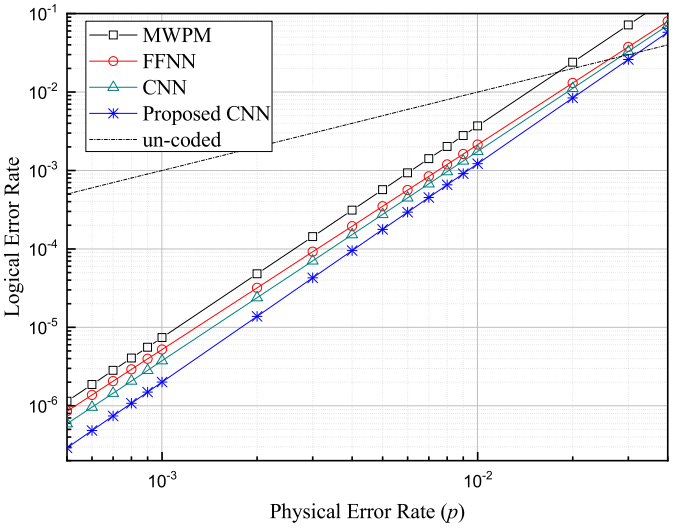



**FIGURE 9. Simulation results of decoding performances over the depolarizing error model. (a) $d = 3$. (b) $d = 5$. (c) $d = 7$.**



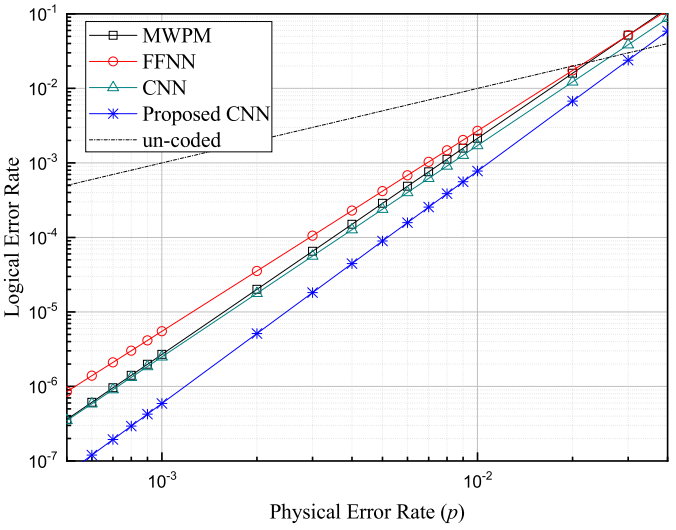
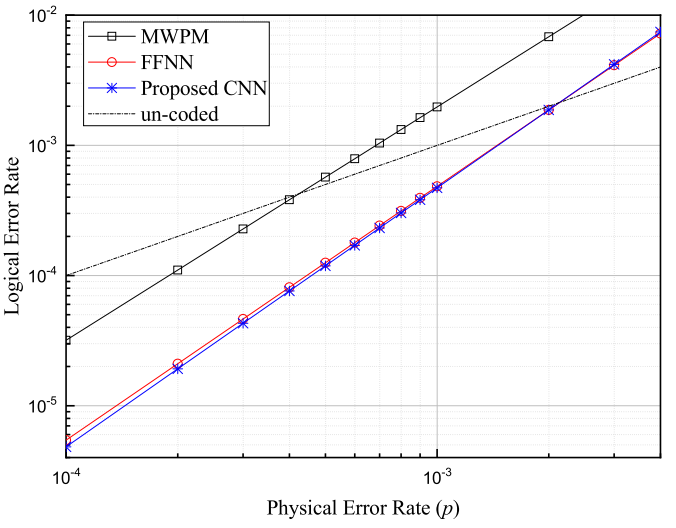
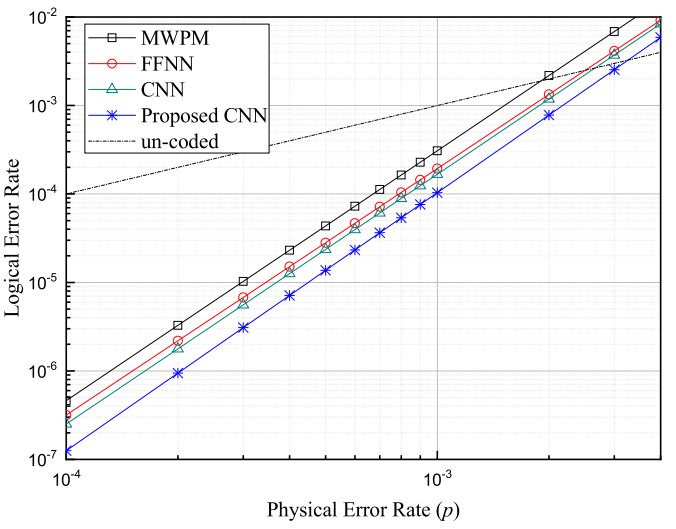




**FIGURE 10.** Simulation results of decoding performances over depolarizing with measurement error model. (a) $d = 3$. (b) $d = 5$. (c) $d = 7$.



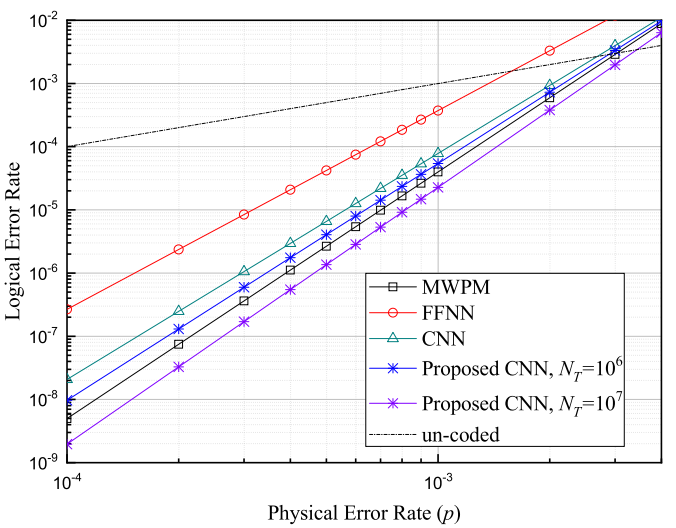




**FIGURE 11.** Simulation results of decoding performances over the circuit noise error model. (a) $d = 3$. (b) $d = 5$. (c) $d = 7$.

IEEE Transactions on Quantum Engineering

**TABLE 2. Comparisons of Pseudothresholds for Different Combinations of Surface Codes and Decoding Algorithms**

| | Depolarizing | | | Depolarizing with measurement | | | Circuit noise | | |
|---|---|---|---|---|---|---|---|---|---|
| Code distance | 3 | 5 | 7 | 3 | 5 | 7 | 3 | 5 | 7 |
| MWPM | 0.0828 | 0.1036 | 0.1194 | 0.0065 | 0.0180 | 0.0225 | 0.0004 | 0.0020 | 0.0031 |
| FFNN | 0.0977 | 0.1135 | 0.1249 | 0.0165 | 0.0261 | 0.0217 | 0.0020 | 0.0025 | 0.0016 |
| CNN | - | 0.1025 | 0.1119 | - | 0.0284 | 0.0262 | - | 0.0026 | 0.0027 |
| Proposed CNN | 0.0980 | 0.1215 | 0.1326 | 0.0165 | 0.0325 | 0.0334 | 0.0020 | 0.0033 | 0.0034 |

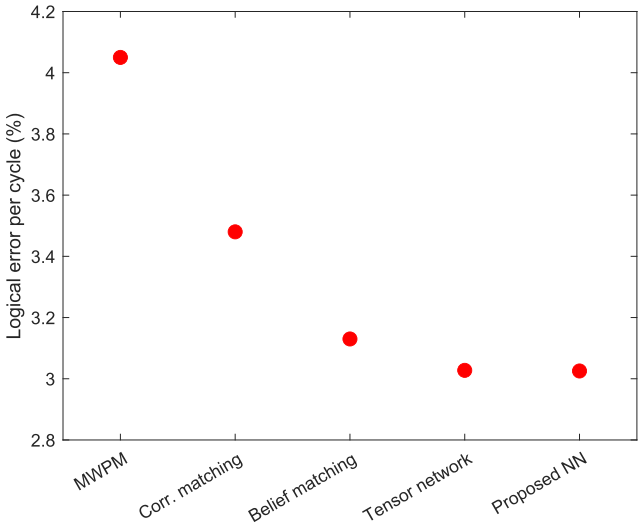



**FIGURE 12. Simulation results of decoding performances based on the experimental data in [29].**

in Fig. 10(b) and 11(b), the proposed CNN-based decoding algorithm achieves the lowest logical error rate, which is also witnessed for the case of $d = 7$ in Figs. 10(c) and 11(c). Note that the proposed CNN-based decoding algorithm addresses errors occurred during the syndrome extraction process by judiciously stacking multiple measurements at different times as its input, as shown in Fig. 4(c). The stacked input has a 3-D correlation, which leads to the improved error rate performance observed in Figs. 10 and 11.

In Fig. 12, we further compare error rate performance of the various algorithms for code distance $d = 3$ based on the experimental data in [29]. The comparisons demonstrate that the proposed algorithm outperforms all the competing decoding algorithms in the realistic evaluation setup, while the tensor network decoding has similar performance.

### B. PSEUDOTHRESHOLDS

The pseudothreshold is defined as the physical error rate at which the logical error rate of a QEC scheme, i.e., a combination of QEC code and decoding algorithm, is the same as the physical error rate. Thus, at a physical error rate higher than the pseudothreshold of a QEC scheme, qubits protected by the QEC scheme have a higher logical error rate than the qubits without the QEC scheme. Table 2 shows the pseudothresholds for various combinations of surface codes of different code distances and decoding algorithms. In the case of $d = 3$, the pseudothresholds of the proposed CNN-based and FFNN decoding algorithms for the depolarizing error model are 0.0980 and 0.0977, respectively which are much larger than that of the MWPM, 0.0828. Meanwhile, for $d = 5$ and $d = 7$, the proposed CNN-based decoding algorithm has the highest pseudothresholds for all the error models.

### C. COMPLEXITY/LATENCY

In this section, we compare the computational complexity and latency of the proposed CNN-based decoding algorithm and the existing decoding algorithms. The complexity is determined by the number of operations, i.e., addition, multiplication, and evaluation functions, whereas the latency is determined by the number of steps, assuming parallel executions of operations. The complexities and latencies of the decoding algorithms are summarized in Table 3, where the latencies are denoted by the number in the parenthesis. The comparison in Table 3 tells that the complexities of the proposed CNN-based decoding algorithm for the depolarizing error model and the circuit noise error model are reduced by five times as compared to those of the existing CNN-based decoding algorithm when the latencies of the decoding algorithms are fixed.

### D. SCALABILITY

In this section, we consider the scalability of our proposed CNN-based decoding algorithm. The complexity of the proposed decoding scheme depends on the size of the lookup table and the four parameters of the CNN network: 1) number of convolutional layers; 2) number of input nodes; 3) number of output nodes; and 4) number of filters. We will discuss that the complexity scales as $d^4$, where $d$ is the code distance as follows.

1) The lookup table contains all pure errors and thus consist of a total count of $d^2 - 1$. That is, the size of the lookup table grows in the order of $d^2$.
2) There has not been an explicit relation between the input size and the number of convolutional layers. However, many successful CNN architectures, such as VGG [33], ResNet [34], and MobileNet [35], have achieved competitive performance for a wide range of input sizes with a fixed number of convolutional layers. For instance, the VGG network in [33], which has 16 convolutional layers, performs reasonably well for a wide range of input sizes up to $512 \times 512$. Since the convolutional layers in the proposed decoding algorithm play a similar role to those in the CNN networks in [33], [34], and [35], the results in the open literature [33], [34], [35] imply that a fixed number of

**TABLE 3. Comparisons of Complexity and Latency for Different Combinations of Surface Codes and Decoding Algorithms**

| | Depolarizing | | | | | |
|---|---|---|---|---|---|---|
| Code distance | 5 | | | 7 | | |
| Calculation | Add. | Mul. | Eva. | Add. | Mul. | Eva. |
| CNN | 1,092,576 (29) | 1,096,520 (4) | 3,944 (4) | 1,178,976 (31) | 1,182,920 (4) | 1,972 (4) |
| Proposed CNN | 213,266 (28) | 215,624 (4) | 2,358 (4) | 263,960 (30) | 267,464 (4) | 2,358 (4) |
| | Circuit noise | | | | | |
| Code distance | 5 | | | 7 | | |
| Calculation | Add. | Mul. | Eva. | Add. | Mul. | Eva. |
| CNN | 6,641,760 (32) | 6,654,152 (4) | 12,392 (4) | 7,028,832 (35) | 7,041,224 (4) | 12,392 (4) |
| Proposed CNN | 1,282,194 (30) | 1,290,440 (4) | 8,246 (4) | 1,540,242 (33) | 1,548,488 (4) | 8,246 (4) |

The values in the parenthesis indicate the latency. Add.: addition; Mul.: multiplication; Eva.: evaluation.

convolutional layers, say $n_c$, effectively address most of the code distances of interest.

3) In this work, we propose a high-level decoding algorithm in which the output of the neural decoder represents the logical states of the errors. Given that there are only four logical states, the output remains fixed at four, regardless of the increase in code distance $d$.
4) The input size of the proposed CNN-based decoding algorithm is given by $d^2$ for the 2-D.
5) As discussed in this article, for surface codes with code distance $d$, the number of all possible patterns of syndromes is $2^{d^2-1}$. If $d^2-1$ independent patterns are identified by a CNN, then all other patterns can be identified by it. Thus, the minimum number of filters is fixed to $d^2-1$ and grows in the order of $d^2$. To further support our proposed method, we include numerical results for code distance $d=9$ in Fig. 13. The numerical experiments have been performed on the depolarizing error model and the circuit noise error model, and the results are presented in Figs. 13(a) and (b), respectively. The performance comparisons in Fig. 13 confirm that the proposed decoding scheme designed based on the theory about scaling of filters still outperforms MWPM in the case of code distance $d=9$.

In conclusion, with the increasing code distance $d$, the computations in the filters dominate the decoding complexity. Since the complexity required by one filter is expressed as $\mathcal{O}(d^2)$, the overall complexity for the proposed CNN-based decoding algorithms is given by $\mathcal{O}(n_c \cdot d^2 \cdot d^2) = \mathcal{O}(n_c \cdot d^4)$, which is lower than that of MWPM, i.e., $\mathcal{O}(m \cdot d^4 \log d)$.

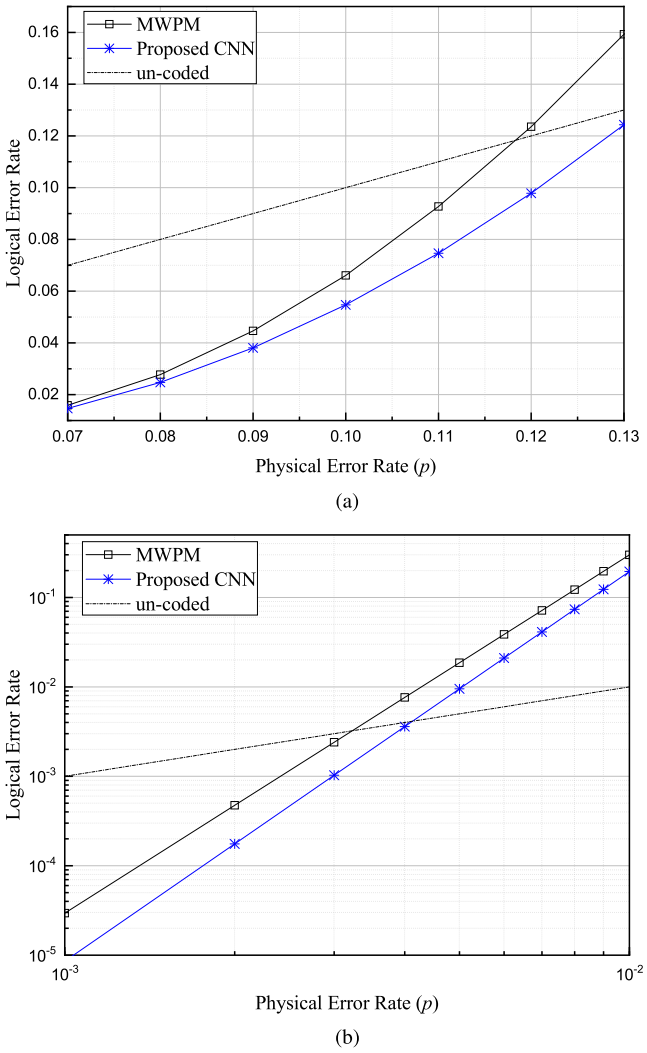



**FIGURE 13. Simulation results of decoding performances for a code distance $d=9$. (a) Depolarizing error model. (b) Circuit noise error model.**

## VI. CONCLUSION

Considering the unreliable characteristic of qubits, QEC codes are an essential part of quantum computing. The topological QEC codes, such as surface codes, are a class of QEC codes that are designed according to the interaction of qubits physically close to each other. In this article, we propose a machine-learning-based decoding algorithm that is designed considering the topological lattice structure of the surface codes. First, we select a CNN that efficiently solves the classification problem of pictures. To retain the topological structure of the syndromes in the surface codes, we use incoherent new values, and these syndromes are then made input to the CNN. In addition, the values of the hyperparameters (i.e., number of filters, size of filters, etc.) used in the CNN are optimized according to the lattice structure of the surface codes. We confirm the efficacy of the proposed

CNN-based decoding algorithm from numerical simulation results and show that it has better performance than the existing decoding algorithms in various quantum error models. Furthermore, if the decoding process is slow (i.e., complexity and latency are high), then errors accumulate in data qubits, while the decoding algorithm is in progress. The proposed CNN-based decoding algorithm has low complexity and latency than the existing CNN-based decoding algorithm.

## REFERENCES

[1] A. M. Steane, "Simple quantum error-correcting codes," *Phys. Rev. A*, vol. 54, no. 6, 1996, Art. no. 4741, doi: 10.1103/PhysRevA.54.4741.

[2] E. Knill and R. Laflamme, "Theory of quantum error-correcting codes," *Phys. Rev. A*, vol. 55, no. 2, 1997, Art. no. 900, doi: 10.1103/PhysRevA.55.900.

[3] D. Gottesman, "Stabilizer codes and quantum error correction," 1997, *arXiv:quant-ph/9705052*, doi: 10.48550/arXiv.quant-ph/9705052.

[4] B. M. Terhal, "Quantum error correction for quantum memories," *Rev. Modern Phys.*, vol. 87, no. 2, 2015, doi: 10.1103/RevModPhys.87.307, Art. no. 307.

[5] H. Bombín, "An introduction to topological quantum codes," 2013, *arXiv:1311.0277*, doi: 10.48550/arXiv.1311.0277.

[6] S. B. Bravyi and A. Y. Kitaev, "Quantum codes on a lattice with boundary," 1998, *arXiv:quant-ph/9811052*, doi: 10.48550/arXiv.quant-ph/9811052.

[7] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, "Topological quantum memory," *J. Math. Phys.*, vol. 43, no. 9, pp. 4452–4505, 2002, doi: 10.1063/1.1499754 .

[8] A. Y. Kitaev, "Fault-tolerant quantum computation by anyons," *Ann. Phys.*, vol. 303, no. 1, pp. 2–30, 2003, doi: 10.1016/S0003-4916(02)00018-0.

[9] M. H. Freedman and D. A. Meyer, "Projective plane and planar quantum codes," *Found. Comput. Math.*, vol. 1, no. 3, pp. 325–332, 2001, doi: 10.1007/s102080010013.

[10] H. Bombin and M. A. Martin-Delgado, "Topological quantum distillation," *Phys. Rev. Lett.*, vol. 97, no. 18, 2006, Art. no. 180501, doi: 10.1103/PhysRevLett.97.180501.

[11] A. J. Landahl, J. T. Anderson, and P. R. Rice, "Fault-tolerant quantum computing with color codes," 2011, *arXiv:1108.5738*, doi: 10.48550/arXiv.1108.5738.

[12] J. Edmonds, "Paths, trees, and flowers," *Can. J. Math.*, vol. 17, pp. 449–467, 1965, doi: 10.4153/CJM-1965-045-4.

[13] V. Kolmogorov, "Blossom V: A new implementation of a minimum cost perfect matching algorithm," *Math. Program. Comput.*, vol. 1, no. 1, pp. 43–67, 2009, doi: 10.1007/s12532-009-0002-8.

[14] G. Duclos-Cianci and D. Poulin, "Fast decoders for topological quantum codes," *Phys. Rev. Lett.*, vol. 104, no. 5, 2010, Art. no. 50504, doi: 10.1103/PhysRevLett.104.050504.

[15] A. Hutter, J. R. Wootton, and D. Loss, "Efficient Markov chain Monte Carlo algorithm for the surface code," *Phys. Rev. A*, vol. 89, no. 2, 2014, Art. no. 022326, doi: 10.1103/PhysRevA.89.022326.

[16] G. Torlai and R. G. Melko, "Neural decoder for topological codes," *Phys. Rev. Lett.*, vol. 119, no. 3, 2017, Art. no. 030501, doi: 10.1103/PhysRevLett.119.030501.

[17] S. Krastanov and L. Jiang, "Deep neural network probabilistic decoder for stabilizer codes," *Sci. Rep.*, vol. 7, no. 1, 2017, Art. no. 11003, doi: 10.1038/s41598-017-11266-1.

[18] Y. Ueno, M. Kondo, M. Tanaka, Y. Suzuki, and Y. Tabuchi, "NEO-QEC: Neural network enhanced online superconducting decoder for surface codes," 2022, *arXiv:2208.05758*, doi: 10.48550/arXiv.2208.05758.

[19] A. Davaasuren, Y. Suzuki, K. Fujii, and M. Koashi, "General framework for constructing fast and near-optimal machine-learning-based decoder of the topological stabilizer codes," *Phys. Rev. Res.*, vol. 2, no. 3, 2020, Art. no. 033399, doi: 10.1103/PhysRevResearch.2.033399.

[20] S. Varsamopoulos, B. Criger, and K. Bertels, "Decoding small surface codes with feedforward neural networks," *Quantum Sci. Technol.*, vol. 3, no. 1, 2017, Art. no. 015004, doi: 10.1088/2058-9565/aa955a.

[21] P. Baireuther, T. E. O'Brien, B. Tarasinski, and C. W. Beenakker, "Machine-learning-assisted correction of correlated qubit errors in a topological code," *Quantum*, vol. 2, 2018, Art. no. 48, doi: 10.22331/q-2018-01-29-48.

[22] C. Chamberland and P. Ronagh, "Deep neural decoders for near term fault-tolerant experiments," *Quantum Sci. Technol.*, vol. 3, no. 4, 2018, Art. no. 044002, doi: 10.1088/2058-9565/aad1f7.

[23] B. Criger and I. Ashraf, "Multi-path summation for decoding 2D topological codes," *Quantum*, vol. 2, 2018, Art. no. 102, doi: 10.22331/q-2018-10-19-102.

[24] E. G. Rieffel and W. H. Polak, *Quantum Computing: A Gentle Introduction*. Cambridge, MA, USA: MIT Press, 2011. [Online]. Available: https://mitpressbookstore.mit.edu/book/9780262526678

[25] O. Higgott, "PyMatching: A Python package for decoding quantum codes with minimum-weight perfect matching," *ACM Trans. Quantum Comput.*, vol. 3, no. 3, pp. 1–16, 2022, doi: 10.1145/3505637.

[26] M. R. Shamsuddin, S. Abdul-Rahman, and A. Mohamed, "Exploratory analysis of MNIST handwritten digit for machine learning modelling," in *Proc. Int. Conf. Soft Comput. Data Sci.*, 2018, pp. 134–145, doi: 10.1007/978-981-13-3441-2_11.

[27] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *Proc. IEEE 54th Annu. Allerton Conf. Commun., Control, Comput.*, 2016, pp. 341–346, doi: 10.1109/ALLERTON.2016.7852251.

[28] D. George and E. Huerta, "Deep neural networks to enable real-time multimessenger astrophysics," *Phys. Rev. D*, vol. 97, no. 4, 2018, Art. no. 044039, doi: 10.1103/PhysRevD.97.044039.

[29] Google Quantum AI, "Suppressing quantum errors by scaling a surface code logical qubit," *Nature*, vol. 614, no. 7949, pp. 676–681, 2023, doi: 10.1038/s41586-022-05434-1.

[30] A. G. Fowler, "Optimal complexity correction of correlated errors in the surface code," 2013, *arXiv:1310.0863*, doi: 10.48550/arXiv.1310.0863.

[31] O. Higgott, T. C. Bohdanowicz, A. Kubica, S. T. Flammia, and E. T. Campbell, "Improved decoding of circuit noise and fragile boundaries of tailored surface codes," *Phys. Rev. X*, vol. 13, no. 3, 2023, Art. no. 031007, doi: 10.1103/PhysRevX.13.031007.

[32] S. Bravyi, M. Suchara, and A. Vargo, "Efficient algorithms for maximum likelihood decoding in the surface code," *Phys. Rev. A*, vol. 90, no. 3, 2014, Art. no. 032326, doi: 10.1103/PhysRevA.90.032326.

[33] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*, doi: 10.48550/arXiv.1409.1556.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[35] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*, doi: 10.48550/arXiv.1704.04861.