Advantages of versatile neural-network decoding for topological codes

Nishad Maskara, Aleksander Kubica, and Tomas Jochym-O'Connor

Phys. Rev. A **99**, 052351 — Published 30 May 2019

DOI: 10.1103/PhysRevA.99.052351

Advantages of versatile neural-network decoding for topological codes

Nishad Maskara,[1] Aleksander Kubica,[2,3] and Tomas Jochym-O'Connor[4]

[1] *California Institute of Technology, Pasadena, CA 91125, USA*
[2] *Perimeter Institute for Theoretical Physics, Waterloo, ON N2L 2Y5, Canada*
[3] *Institute for Quantum Computing, University of Waterloo, Waterloo, ON N2L 3G1, Canada*
[4] *Walter Burke Institute for Theoretical Physics and Institute for Quantum Information & Matter, California Institute of Technology, Pasadena, CA 91125, USA*

(Dated: April 26, 2019)

Finding optimal correction of errors in generic stabilizer codes is a computationally hard problem, even for simple noise models. While this task can be simplified for codes with some structure, such as topological stabilizer codes, developing good and efficient decoders still remains a challenge. In our work, we systematically study a very versatile class of decoders based on feedforward neural networks. To demonstrate adaptability, we apply neural decoders to the triangular color and toric codes under various noise models with realistic features, such as spatially-correlated errors. We report that neural decoders provide a significant improvement over leading efficient decoders in terms of the error-correction threshold. In particular, the neural decoder threshold for the two-dimensional color code is very close to the toric code threshold. Using neural networks simplifies the design of decoders and does not require prior knowledge of the underlying noise.

I. INTRODUCTION

Recent small-scale experiments [1–4] have shown an increasing level of control over quantum systems, constituting an important step towards the demonstration of quantum error correction and fault-tolerance [5, 6]. In order to scale up quantum devices and maintain their computational power, one needs to protect logical information from unavoidable errors by encoding it into quantum error-correcting codes [7]. One of the most successful class of quantum codes, stabilizer codes [8], allows one to detect errors by measuring stabilizer operators without altering the encoded information. Subsequently, errors can be corrected by implementing a recovery operation. A classical algorithm, which allows one to find an appropriate correction from the available classical data, i.e., the ± 1 measurement outcomes of stabilizers for the given code, is called a decoder.

Optimal decoding of generic stabilizer codes is a computationally hard problem, even for simple noise models [9, 10]. If codes have some structure, then the task of decoding becomes more tractable and efficient decoders with good performance may be available. For example, in the case of topological stabilizer codes [11–15], whose stabilizer generators are geometrically local, any unsatisfied stabilizer returning -1 measurement outcome indicates the presence of errors on some qubits in its neighborhood. By exploiting this pattern, many decoding schemes have been developed, some of which are based on cellular automata [16–23], the Minimum-Weight Perfect Matching algorithm [24–26], tensor networks [27, 28], renormalization group [29–33] or other approaches [34–38].

Efficient decoders with good performance are often tailor-made for specific codes and are not easily adaptable to other settings. For instance, despite a local unitary equivalence of two families of topological codes [39], the color and toric codes, one cannot straightforwardly use toric code decoders in the color code setting; rather, some careful modifications are needed [22, 25]. Moreover, decoding strategies are typically designed and analyzed for simplistic noise models, which may not truly reflect errors present in the experimental setup. Importantly, the best approach to scalable quantum devices is still under debate and dominant sources of noise are yet to be thoroughly explored. Thus, it would be very desirable to develop decoding methods without full characterization of quantum hardware, which are adaptable to various quantum codes and realistic noise models.

threshold of the triangular color code			
noise \ decoder	neural	projection	optimal
bit-/phase-flip	$\sim 19.0\%$	$\sim 16.2\%$	20.6(4)% [40]
depolarizing	$\sim 17.4\%$	$\sim 12.6\%$	18.9(3)% [41]
NN-depolarizing	$\sim 15.0\%$	$\sim 13.5\%$	?

threshold of the triangular toric code with a twist			
noise \ decoder	neural	MWPM	optimal
bit-/phase-flip	$\sim 19.6\%$	$\sim 19.2\%$	20.68(4)% [24]
depolarizing	$\sim 18.0\%$	$\sim 15.3\%$	18.9(3)% [41]
NN-depolarizing	$\sim 16.7\%$	$\sim 14.2\%$	?

TABLE I. The error-correction threshold for neural decoders compared with standard decoding methods based on the Minimum-Weight Perfect Matching algorithm and the projection decoder. Neural decoders were applied to the 2D toric and color codes of distance up to $d = 11$. Numerical simulations were performed for various noise models, including the nearest-neighbor spatially-correlated depolarizing noise, assuming perfect syndrome measurements. To fairly compare different noise models, threshold error rates are expressed in terms of the effective error rate p_{eff}, see Section II D. The quoted optimal threshold values were found for models without boundaries.

The main goal of our work is to systematically explore recently proposed decoding strategies based on artificial neural networks [42–46]. We consider a two-step decoding strategy. In step 1, for any given configuration of unsatisfied stabilizers we deterministically find a Pauli operator, which returns corrupted encoded information into the code space. After this step, all stabilizers are satisfied but a non-trivial logical operator may have been implemented by the attempted Pauli correction combined with the initial error. In step 2, we use a feedforward neural network to determine what (if any) non-trivial logical operator is likely to be introduced in step 1, so that we can account for it in the recovery. We emphasize that step 2 is a classification problem, particularly well-suited for machine learning.

In our work, we convincingly demonstrate the versatility of neural decoders by applying them to two families of codes, the two-dimensional (2D) triangular color and toric codes, under different noise models with realistic features, such as spatially-correlated errors. We observe that, irrespective of the noise model, neural-network decoding outperforms standard strategies, including the Minimum-Weight Perfect Matching algorithm [24] and the projection decoder [25]; see Table I. In particular, the neural decoder threshold for the 2D color code is higher than the threshold of any other efficient decoder known up to date which, in turn, implies that the performance of the color code can be close to the toric code. It is worth emphasizing that only the training datasets, but not the explicit knowledge of the noise or the geometric structure of the codes, were needed to train neural decoders. We also analyze how computational costs of training and neural network parameters scale with the growing code distance. Our work indicates that, due to its adaptability, neural-network decoding is a promising error-correction method which can be used in a wide range of future small-scale quantum devices, especially if the dominant sources of errors are not well characterized.

Our work differs from previous machine learning based decoders [42–46] as it emphasizes first and foremost adaptability. The distinguishing aspects of our work are as follows: (i) We specifically pick two code families with the same 2D qubit layout that differ in their choice of local stabilizer generators; (ii) We systematically study different noise models, including spatially correlated noise; (iii) We consider code distances larger than previously studied, which allows us to reliably determine the performance and error correction thresholds of neural decoders. It should be noted that we optimize the size of the networks for decoder performance rather than computational efficiency. However, we believe that by using more sophisticated neural network models, such as convolutional neural networks, efficient neural decoders could be achieved for larger codes [47].

The organization of the article is as follows. We start by discussing quantum error correction from the perspective of topological codes, the triangular color code and the toric code with a twist. In particular, in Section II C we explain how to construct the excitation graph, which leads to a simple algorithm for step 1 of the neural decoder. In Section II D we introduce a new notion of the effective error rate, which allows us to easily compare threshold error rates for different noise models. Then, we describe neural decoding and its performance under different noise models, including the spatially-correlated depolarizing noise. In Section III B we explain how the training of deep neural networks is accomplished by successively increasing the error rate used to generate the training dataset. This training method likely has significant impact, since it may lead to faster convergence and better final performance of neural networks for quantum error-correction applications. We conclude the article with the discussion of our results and their implications for future neural decoders used in practice.

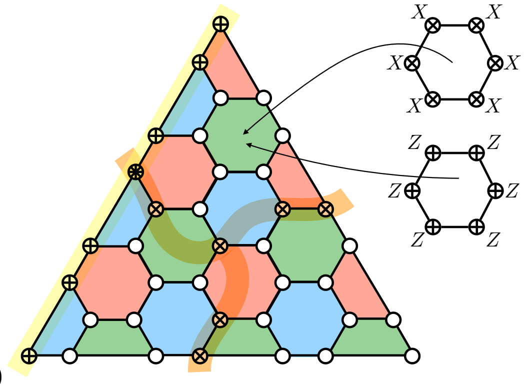
II. ERROR CORRECTION WITH TOPOLOGICAL CODES

A. Topological stabilizer codes

Stabilizer codes [8] are an important class of quantum error-correcting codes [7] specified by a stabilizer group $\mathcal{S}$. The stabilizer group $\mathcal{S}$ is an Abelian subgroup of the Pauli group generated by n-qubit Pauli operators $P_1 \otimes \ldots \otimes P_n$, where $P_i \in \{I, X, Y, Z\}$ and $-I \notin \mathcal{S}$. The logical information is encoded into the codespace, which is the $(+1)$-eigenspace of all the elements of $\mathcal{S}$. Logical Pauli operators $\overline{L} \in \mathcal{L}$ are identified with elements of the normalizer $\mathcal{S}$ of the stabilizer group $\mathcal{S}$ in the Pauli group. An operator L which implements a non-trivial logical Pauli operator $\overline{L} \neq \overline{I}$ can be chosen to be a product of Pauli operators, which commute with all the elements in the stabilizer group but do not belong to $\mathcal{S}$. The weight of the minimal-support non-trivial logical Pauli operator is the distance of the code.

Physical qubits of the stabilizer code can be affected by noise, which can take encoded logical information outside of the codespace. No information about the original encoded state is revealed by measuring stabilizer generators. Rather, one effectively projects errors present in the system onto some Pauli operators and subsequently gains some knowledge about them. The set of unsatisfied stabilizers returning a -1 measurement outcome is called a syndrome. The syndrome serves as a classical input to a decoding algorithm, which allows one to find a recovery Pauli operator bringing the corrupted encoded state back to the codespace. For a special class of stabilizer codes, the CSS codes [48], whose stabilizer generators are products of either X- or Z-type Pauli operators, one can independently correct Z- and X-type errors using the appropriate X- and Z-type syndrome.

Topological stabilizer codes [11–15] are a family of stabilizer codes exhibiting particularly good resilience to noise. The distinctive feature of topological stabilizer codes is the geometric locality of their generators. Namely, physical qubits can be arranged to form a lattice



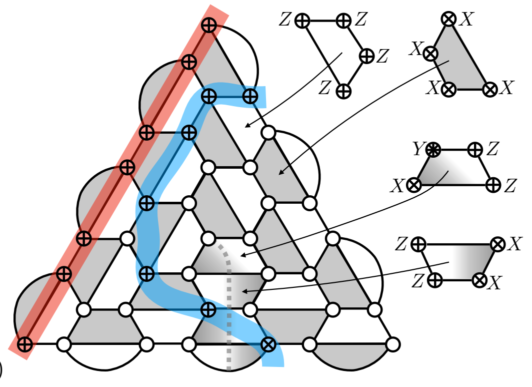


FIG. 1. (a) 2D triangular color code on a patch of the hexagonal lattice with 3-valent vertices and 3-colorable faces. Every face supports both X- and Z-stabilizers. The string of Pauli Z operators (yellow $\oplus$) implements a logical $\overline{Z}$ operator, while the string of Pauli X operators (orange $\otimes$) implements a logical $\overline{X}$. Both operators connect all three boundaries. (b) 2D triangular toric code with a twist. Dark and white faces support X- and Z-stabilizers, respectively. Depending on the coloring of mixed dark/white faces along a 1D defect line (dashed line), stabilizers are mixed products of Pauli X and Z. Red and blue strings depict two equivalent representatives of a logical $\overline{Z}$ operator. Upon crossing the defect line, the string changes from X-type (blue $\otimes$) to Z-type (blue $\oplus$).

in such a way that every stabilizer generator is supported on a constant number of qubits within some geometrically local region[1]. At the same time, no logical Pauli operator can be implemented via a unitary acting on physical qubits in any local region. By enlarging the system size, one increases the distance and error-correction capabilities of the topological code without changing the required complexity of local stabilizer measurements. This is in stark contrast with other quantum codes, such as concatenated codes [49], whose stabilizer weight necessarily increases with the distance and thus makes those constructions experimentally more challenging.

Two well-known examples of topological stabilizer codes are the toric and color codes. The triangular color code is defined on a two-dimensional lattice with a boundary, whose vertices are 3-valent [2] and faces $f \in F$ are 3-colorable; see Fig. 1(a). Qubits are identified with vertices. The color code is a CSS code and its stabilizer group is defined as follows

$$\mathcal{S}_{CC} = \langle X_f, Z_f | f \in F \rangle, \tag{1}$$

where X_f and Z_f are Pauli X and Z operators supported on all qubits belonging to a face $f \in F$. Accordingly, X- and Z-type errors can be independently corrected using the Z- and X-type syndrome.

The triangular toric code with a twist [50] can be defined for the same arrangement of physical qubits as the triangular color code. Its lattice can be obtained from the color code lattice by keeping all the vertices, adding extra edges and modifying some faces; see Fig. 1(b). The resulting lattice is 4-valent [3] and the faces are 2-colorable, except for the "mixed" faces along a 1D defect line. The color of the face indicates the type of the stabilizer generator identified with that face. Namely, dark $f \in F_D$ and white $g \in F_W$ faces support X-type X_f and Z-type Z_g stabilizers. Depending on the coloring of mixed faces $h \in F_M$, stabilizers S_h are defined to be mixed products of both Pauli X and Z operators. We emphasize that the choice of mixed stabilizer generators along the defect line is needed for the stabilizers S_h to commute with X_f and Z_g for all $f \in F_D, g \in F_W, h \in F_M$. The full stabilizer group is thus given by

$$\mathcal{S}_{TC} = \langle X_f, Z_g, S_h | f \in F_D, g \in F_W, h \in F_M \rangle. \tag{2}$$

We remark that due to mixed stabilizer generators, the toric code with a twist is not a CSS code and thus one should not decode X and Z errors independently. As we will explain in Sec. II C, the excitation graph of the toric code with a twist does not split into two disconnected components for X and Z errors; see Fig. 3.

Logical Pauli operators of the 2D topological stabilizer codes can be thought of as deformable non-contractible 1D string-like operators. In the case of the triangular color and toric codes, logical operators connect certain boundaries as depicted in Fig. 1.

B. Quasiparticle excitations

It is illustrative to establish a connection between quantum error-correcting codes and quantum many-body

[1] The topological codes studied here are also within a more general class of stabilizer codes, low-density parity check (LDPC) codes. These codes have stabilizers whose support is bounded on a constant number of qubits, and each qubit is only in the support of few stabilizers. However, for general LDPC codes, there are no further constraints on the support of the stabilizers, such as geometric locality, rendering their decoding more difficult.

[2] All the vertices are 3-valent except for three corner vertices on the boundary.

[3] All the vertices are 4-valent except for three corner vertices on the boundary and one vertex in the bulk, which corresponds to a twist, i.e., the end of the defect line.

systems described by commuting Hamiltonians. For a topological stabilizer code with the stabilizer group $\mathcal{S}$ we can define a commuting *stabilizer Hamiltonian* $H(\mathcal{S})$ to be a sum of stabilizer generators of $\mathcal{S}$ with a negative sign. In particular, for the color code and the toric code with a twist we choose their stabilizer Hamiltonians to be

$$H_{CC} = -\sum_{f\in F} X_f - \sum_{f\in F} Z_f, \tag{3}$$

$$H_{TC} = -\sum_{f\in F_D} X_f - \sum_{g\in F_W} Z_g - \sum_{h\in F_M} S_h. \tag{4}$$

Note that all the terms in the stabilizer Hamiltonian $H(\mathcal{S})$ are mutually commuting, thus any eigenstate of $H(\mathcal{S})$ has to be an eigenstate of every single term. Since eigenstates of stabilizer generators can only have ± 1 eigenvalues, we conclude that the code space defined as the $(+1)$-eigenspace of all the elements of $\mathcal{S}$ coincides with the ground space of $H(\mathcal{S})$. We will highlight some of the important properties of the Hamiltonian formulation of stabilizer codes in this subsection, see Ref. [51] for an extended description.

We can think of errors affecting information encoded in the topological stabilizer code as operators creating localized quasiparticle excitations in the related quantum many-body system. Namely, consider any Pauli error which anticommutes with some stabilizer generators. The error moves the encoded logical state outside the code space or, equivalently, the ground state outside the ground space. The resulting state is excited in the sense that its energy is larger than the ground space energy by the amount proportional to the number of violated stabilizer Hamiltonian terms. The unsatisfied stabilizer terms can be identified with quasiparticle excitations [11, 52–54]. Depending on whether the unsatisfied stabilizer is of X- or Z-type, we will call the excitation electric e_K or magnetic m_K. [4] The subscript K indicates the color of the face supporting the excitation. In particular, for the toric code we can only have e_D and m_W, whereas the color code excitations can be supported on faces of any color, i.e., e_K and m_K for any $K \in \{R, G, B\}$.

In order to understand excitation configurations arising from any Pauli errors, it suffices to know what excitations geometrically local Pauli operators can create and how to combine them. We now discuss these constraints, also known as *fusion rules* for topological stabilizer codes. In case of the toric code, a single-qubit Pauli X or Z error on the qubit in the bulk of the system violates two Z- or X-type stabilizers on neighboring faces and thus necessarily creates two excitations of the same type, either magnetic or electric; see Fig. 2(b). If two errors with non-overlapping support independently create the same excitation on a face $f \in F$, then the product of both errors will not create any excitation at that location. For an illustration, let us consider two single-qubit errors X_i and X_j on qubits i and j belonging to the edge $\{i, j\}$. Each error independently creates a magnetic excitation on the face f containing the edge $\{i, j\}$; however, the combined error $X_i X_j$ results in no excitation on f. The above discussion can be summarized by the toric code fusion rules

$$e_D \times e_D = m_W \times m_W = 1, \tag{5}$$

which express the fact that in the bulk excitations of the same type can only be created (by geometrically local operators) or annihilated in pairs. Note that 1 denotes no excitation.

The fusion rules for the color code are slightly more complicated than for the toric code. Namely, we have

$$e_K \times e_K = m_K \times m_K = 1, \tag{6}$$

$$e_R \times e_G \times e_B = m_R \times m_G \times m_B = 1, \tag{7}$$

where $K \in \{R, G, B\}$. Similarly as for the toric code, combining two excitations of the same type and color results in no excitation. However, in the bulk of the color code it is also possible to create (by a local operator) or annihilate a *triple* of excitations. We can see that by considering a single-qubit Pauli X or Z error. It violates three Z- or X-type stabilizers on neighboring red, green and blue faces and thus creates a triple of magnetic or electric excitations; see Fig. 2(a).

The topological stabilizer codes we consider are defined on lattices with boundaries. By acting with a local Pauli operator on the qubits near the boundary of the system it is possible to create or annihilate a *single* magnetic or electric excitation. We emphasize that the type of the boundary determines the type of the allowed excitation [55]. For the triangular toric code, there are two types of boundaries, rough or smooth [12], and a single electric (respectively magnetic) excitation can only be created on the rough (smooth) boundary; see Fig. 2(b). In case of the triangular color code, there are three types of boundaries, red, green or blue [13], and single electric and magnetic excitations of given color can be created on the boundary of the matching color; see Fig. 2(a).

Once a quasiparticle excitation is created, it can always be moved in the bulk of the 2D topological stabilizer code by applying an appropriate 1D string-like Pauli operator [56, 57]. Given fusion rules, the excitation movement can be understood as a process of creating pairs of excitations along some path and fusing them together with the initial one, which results in the excitation changing its position. When the quasiparticle excitation moves its type does not change, unless it passes through a *defect line*. A defect line, also known as a transparent domain wall[5] [58–60], is a 1D object, along which the stabilizer

[4] For the mixed stabilizers along the defect line, there is ambiguity in associating the type of the excitation since the electric and magnetic excitations are exchanged upon crossing the defect line. Thus, we would refer to those excitations without specifying their type.

[5] A transparent domain wall can be thought of as an automor-

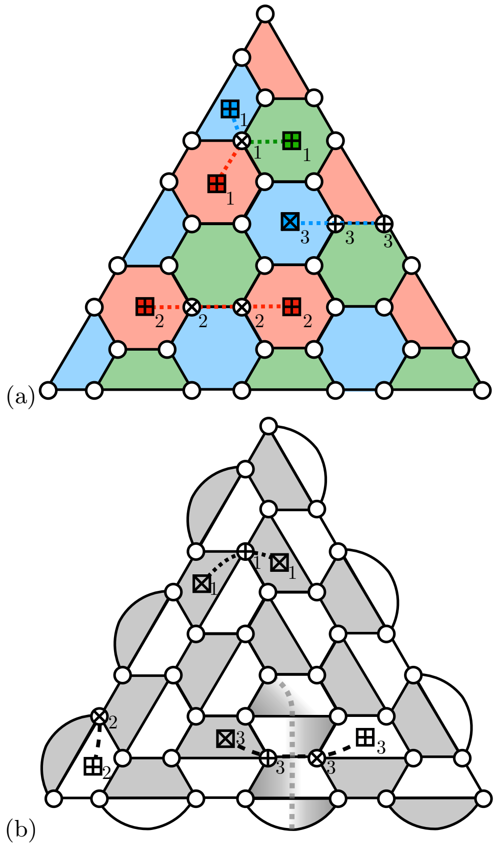


FIG. 2. Quasiparticle excitations in the 2D triangular color and toric codes. (a) A single X-error (white $\otimes_1$) in the bulk of the color code leads to three unsatisfied Z-stabilizers on neighboring faces, thus creates a triple of magnetic excitations (red, green and blue $\boxplus_1$). A string of X-errors (white $\otimes_2$) creates a pair of magnetic excitations (red $\boxplus_2$). A string of Z-errors (white $\oplus_3$) terminating at the blue boundary creates a single electric excitation (blue $\boxtimes_3$). (b) A single Z-error (white $\oplus_1$) in the bulk of the toric code with a twist leads to two unsatisfied X-stabilizers on neighboring dark faces, thus creates a pair of electric excitations (gray $\boxtimes_1$). A single X-error (white $\otimes_2$) on the rough boundary creates a single magnetic excitation (white $\boxplus_2$). A pair of electric (gray $\boxtimes_3$) and magnetic (white $\boxplus_3$) excitations can be created by a string of errors (white $\otimes_3$ and $\otimes_3$) across the defect line (dashed line).

generators are appropriately modified. In case of the triangular toric code with a twist, one chooses stabilizers on faces intersected by the defect line to be mixed products of Pauli X and Z operators; see Fig. 1(b). When an electric excitation e_D crosses the defect line, it becomes a magnetic excitation m_W, and vice versa, $e_D \leftrightarrow m_W$. We emphasize that logical Pauli operators for the triangular color and toric codes can be implemented by creating a single excitation on one of the boundaries and transporting it to the other boundary, where it can annihilate; see Fig. 1 for examples of logical operators.

phism of the excitation labels which preserves the braiding and fusion rules of the quasiparticle excitations.

We remark that there are only two possible types of defect lines in the toric code, one of which is trivial. However, in case of the color code, there are 72 different defect lines [61]. We encourage readers to explore [62] for an illuminating discussion of all the possible boundaries and defect lines in the 2D color code.

C. Decoding of topological codes as a classification problem

As we already discussed, generic errors affect the encoded information by moving it outside the code space, which results in some stabilizers being unsatisfied. A classical algorithm which takes the syndrome as an input and finds an appropriate recovery restoring all stabilizers to yield $+1$ measurement outcome is called a *decoder*. For stabilizer codes the recovery operator is a Pauli operator. We say that decoding is successful if no non-trivial logical operator has been implemented by the recovery combined with the error.

We can view decoding as a process of removing quasiparticle excitations from the system and returning the state to the ground space of the stabilizer Hamiltonian. To facilitate the discussion, we introduce an *excitation graph* $G = (V, E)$, which captures how the excitations can be moved (and eventually removed) within the lattice of the topological stabilizer code, see for example Refs. [34, 63]. The vertices V of the excitation graph G correspond to the (possible locations of) quasiparticle excitations. Note that there is one vertex for every single electric, as well as for magnetic excitation. We also include in V one special vertex w, called the boundary vertex. Two different vertices $v_1, v_2 \in V \setminus \{w\}$ are connected by an edge $\{v_1, v_2\} \in E$ if there is a Pauli operator P_{v_1,v_2} with geometrically local support which can move an excitation from v_1 to v_2 without creating any other excitations. We say that $v \in V \setminus \{w\}$ and the boundary vertex w are connected by an edge $\{v, w\}$ if one can locally create a single excitation at v. In case of the toric and color codes, we restrict our attention to local operators, which are supported on respectively one or at most two neighboring qubits. We identify the edges $\{v_1, v_2\}$ in E with the local operators P_{v_1,v_2}. We illustrate how to construct the excitation graph in Fig. 3.

We consider a very simple deterministic procedure, the excitation removal algorithm, which efficiently eliminates quasiparticle excitations from the toric and color codes. The excitation removal algorithm builds on the idea of error decomposition [30, 64] and resembles the simple decoder in Ref. [45]. Let Q be some Pauli error operator, which results in the excitation configuration $U \subset V \setminus \{w\}$ in the system. The input of the algorithm is U, but not Q. For every excitation $u \in U$ we find the shortest path $(v_1, v_2, \dots, v_n)$ in the excitation graph G between $u = v_1$ and the boundary vertex $w = v_n$, where $v_i \in V$ and $\{v_i, v_{i+1}\} \in E$. We define an operator P_u to be a prod-

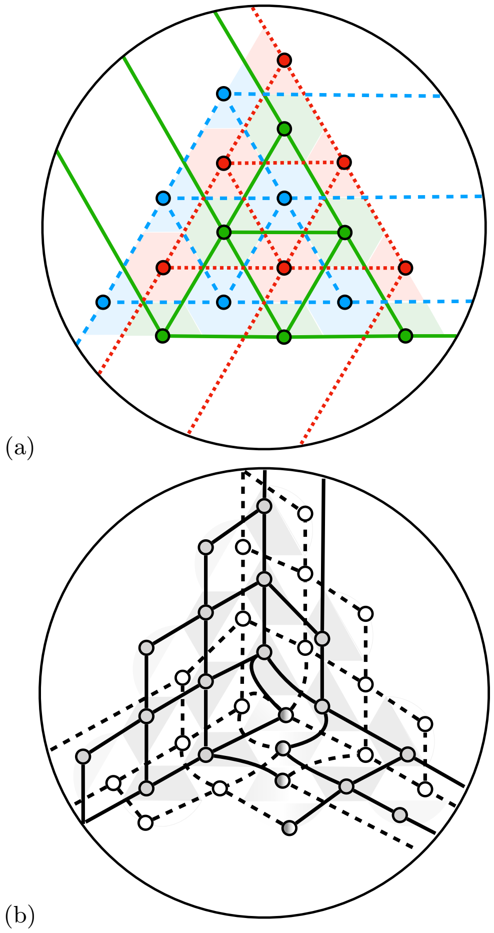


FIG. 3. Construction of the excitation graph $G = (V, E)$ for (a) the color code and (b) the toric code with a twist. For every face f of the topological code lattice we add a vertex v_f to the set of vertices V of G. We also include the boundary vertex w (enclosing circle) in V. (a) It is not possible to move a single excitation in the bulk (without creating more excitations) by applying a single-qubit operator. However, since a two-qubit operator XX or ZZ can move an excitation between two nearby faces f and g of the same color, we add an edge $\{v_f, v_g\}$ to E. (b) A single-qubit Pauli X or Z error can move an excitation between two neighboring faces f and g of the same color, thus we add an edge $\{v_f, v_g\}$ between v_f and v_g to the set of edges E of G. We connect a vertex v_f with the boundary vertex w if one can create a single excitation on f by (a) a single- or two-qubit operators and (b) a single-qubit operator. Note that in (a) we depict only a part of the excitation graph corresponding to electric excitations and Z-type errors, since the part for magnetic excitations and X-type errors is identical.

uct of local Pauli operators $P_{v_i,v_{i+1}}$ identified with the edges $\{v_i, v_{i+1}\}$ along the path $(v_1, v_2, \ldots, v_n)$, namely $P_u = \prod_{i=1}^{n-1} P_{v_i,v_{i+1}}$. The operator P_u moves an excitation from u to the boundary where it is annihilated. As the output of the algorithm we choose an operator $R_U = \prod_{u\in U} P_u$. We remark that the operator R_U returns the state to the ground space since it removes all the excitations, and thus $R_U Q \in \mathcal{L}$. At the same time, the output R_U combined with the initial error Q likely implements some non-trivial logical operator. Thus, the excitation removal algorithm viewed as a decoder would perform rather poorly.

Algorithm 1: excitation removal

```
Require: the excitation graph $G = (V, E)$
Input: positions $U \subset V \setminus \{w\}$ of excitations
Output: Pauli operator $R_U$ removing all excitations
initialize $R_U \leftarrow I$
for every $u \in U$:
    1. find the shortest path $(v_1, \ldots, v_n)$ in $G$ between
       $u = v_1$ and the boundary vertex $w = v_n$
    2. find an operator $P_u = P_{v_1,v_2} \cdot \ldots \cdot P_{v_{n-1},v_n}$
       corresponding to the path $(v_1, \ldots, v_n)$
    3. $R_U \leftarrow R_U P_u$
return $R_U$
```

Now we explain how to reduce the decoding problem to a classification problem by using the excitation removal algorithm. The task of classification is to assign labels, typically from some small set, to the elements of some high-dimensional dataset. In the decoding problem, we know positions $U \subset V \setminus \{w\}$ of the excitations and want to find a recovery operator removing all the excitations and implementing the trivial logical operator. We do not know, however, the Pauli operator Q resulting in the excitation configuration U. Using the excitation removal algorithm we easily find the operator R_U. Clearly, we would be able to successfully decode if we chose $R_U L$ as a recovery operator, where L is any operator implementing the same logical operator $\overline{L} \in \mathcal{L}$ as $R_U Q$. Unfortunately, there are many different error operators creating the same configuration of excitations U. We can split all those error operators Q into equivalence classes identified with different logical operators $\overline{L}$ implemented by $R_U Q$. Then, for any given excitation configuration U we can find the most probable equivalence class of errors creating U. What we would like to achieve is to label U by a logical operator $\overline{L}$, which is implemented by the output R_U of the excitation removal algorithm and any operator Q from the most probable class of errors. Such a problem is well-suited for machine learning techniques, in particular for artificial neural networks. We defer further discussion of the classification problem to Section III A, where we explain it in the context of neural-network decoding.

D. Noise models and thresholds

In order to test versatility of neural decoders, we numerically simulate their performance for various noise models. In particular, we consider the following three Pauli error models specified by just one parameter, the error rate p.

- *Bit-/phase-flip noise*: every qubit is independently affected by an X error with probability p, and by a Z error with the same probability p.
- *Depolarizing noise*: every qubit is independently affected with probability p by an error, which is uniformly chosen from three errors X, Y and Z.
- *NN-depolarizing noise*: the spatially-correlated depolarizing noise on nearest-neighbor qubits, i.e., every pair of qubits i and j sharing an edge in the lattice is independently affected with probability p by a non-trivial error, which is uniformly chosen from 15 errors of the form P_iP_j, where $P_i, P_j \in \{I, X, Y, Z\}$ and $P_iP_j \neq II$.

We emphasize that one should not necessarily think of the aforementioned noise models as accurately describing errors in the experimental setup. Rather, we choose those models since they are easy to specify and simulate but, at the same time, they also capture realistic noise features, such as spatial correlations of errors, which any good decoder should be able to handle [26]. In addition, in the current proposed circuit-based models for syndrome measurement [65] correlated errors across neighboring qubits would naturally arise.

We would like to easily compare the bit-/phase-flip, depolarizing and NN-depolarizing noise models. However, the error rate p has a different meaning depending on the considered model. This motivates us to introduce a new figure of merit for Pauli error models, the *effective error rate* p_{eff}. For any physical qubit we define the effective error rate p_{eff} to be the probability of any non-trivial error affecting that qubit. Note that in the scenarios we consider the effective error rate is the same for all the qubits (except for the ones identified with the corner vertices and the twist for the NN-depolarizing noise). Thus, we can unambiguously talk about the effective error rate without specifying which qubit we are referring to. For the depolarizing noise we simply have $p_{\text{eff}} = p$, whereas for the the bit-/phase-flip noise we find $p_{\text{eff}} = 1 - (1-p)(1-p) = 2p - p^2$. In case of the NN-depolarizing noise, the effective error rate depends on the local structure of the lattice. Namely, if n denotes the number of nearest neighbors for some qubit, then the effective error rate $p_{\text{eff}}^{(n)}$ for that qubit can be recursively calculated as

$$p_{\text{eff}}^{(n)} = p_{\text{eff}}^{(n-1)}\left(1 - \frac{4}{15}p\right) + \left(1 - p_{\text{eff}}^{(n-1)}\right)\frac{12}{15}p \quad (8)$$

$$= \frac{4}{5}np + O(p^2), \quad (9)$$

where we use $p_{\text{eff}}^{(0)} = 0$ and denote by $o(p^2)$ the second-order corrections in p. In particular, for the analyzed color and toric code lattices we respectively have $p_{\text{eff}}^{(3)}$ and $p_{\text{eff}}^{(4)}$.

In order to assess the performance of a decoder for the given family of codes with growing code distance d and specified noise model, we use the quantity called the *error-correction threshold*. The error-correction threshold is defined as the largest p_{th}, such that for all effective error rates $p_{\text{eff}} < p_{\text{th}}$ the probability of unsuccessful decoding $p_{\text{fail}}(p_{\text{eff}}, d)$ for the code with distance d goes to zero in the limit of infinite code distance, $\lim_{d\to\infty} p_{\text{fail}}(p_{\text{eff}}, d) = 0$. Note that in the definition of the threshold we assume perfect stabilizer measurements. We remark that one typically estimates the threshold p_{th} by plotting the decoder failure probability $p_{\text{fail}}(p_{\text{eff}}, d)$ as a function of the effective error rate p_{eff} for different code distances d and identifying their crossing point; see Figs. 6 and 7.

III. PERFORMANCE OF NEURAL-NETWORK DECODING

A. Neural decoders

We have already seen in Section II C that the task of successful decoding can be deterministically reduced to the following problem: for any configuration of excitations $U \subset V \setminus \{w\}$ created by some unknown Pauli operator Q assign a label $\overline{L}$ from the set of logical operators $\mathcal{L}$, such that $\overline{L}$ is the logical operator implemented by R_UQ, where R_U is the output of the excitation removal algorithm with U as the input. We approach this classification problem by using one of the leading machine learning techniques, feedforward neural networks [66, 67]. The input layer encodes the configuration of excitations U. Then, there are H_d hidden layers, each containing N_d nodes. Nodes from layer $l+1$ are fully connected with nodes from the preceding layer l. For each code of distance d, we therefore train a neural network consisting of $H_d + 2$ layers; see Fig. 4. Every node ν in layer $l+1$ evaluates an activation function $\sigma(w_\nu \cdot o_l + b_\nu)$ on the output o_l of nodes from layer l, where w_ν and b_ν are the weights and biases associated with the node ν. We choose the rectified linear unit activation function $\sigma(x) = \max(0, x)$. The output layer uses the softmax classifier, which converts an output vector to a discrete probability distribution describing the likelihood of different logical operators $\overline{L} \in \mathcal{L}$ being implemented by R_UQ.

We are now ready to describe neural-network decoding for topological stabilizer codes. The neural decoder is an algorithm which returns a recovery operator R for any configuration of excitations $U \subset V \setminus \{w\}$ created by some unknown operator Q. We emphasize that error operators Q are chosen according to some a priori unknown noise model. The neural decoders we consider consist of the following two steps. In step 1, we use a simple deterministic procedure, the excitation removal algorithm, to find a Pauli operator R_U, which removes quasiparticle excitations by moving them to the boundaries of the system, where they disappear. In step 2, we use a neural

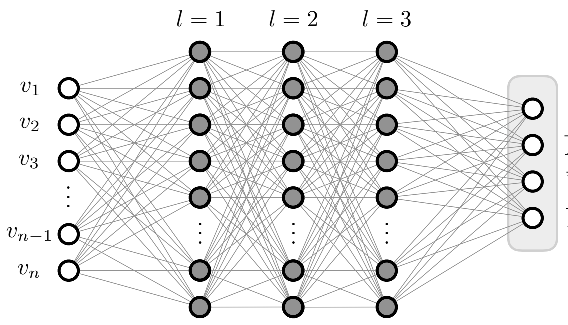


FIG. 4. A feedforward neural network with $H_d = 3$ hidden layers can be viewed as an acyclic directed graph. Each hidden layer has the same number of nodes N_d. Nodes from layer $l+1$ are fully connected with nodes from the preceding layer l. The input layer encodes all the initial excitation configuration $U \subset V \setminus \{w\}$. The output layer encodes the likelihood of each logical operator $\overline{L} \in \{\overline{I}, \overline{X}, \overline{Y}, \overline{Z}\}$ assigned to the input configuration U.

network to guess what are the most likely errors Q resulting in U and which logical operator $\overline{L}$ is subsequently implemented by $R_U Q$, as was done in Ref. [45]. As the output, the operator $R_U L$ is returned, where L is any operator implementing the logical operator $\overline{L}$. We emphasize that the neural decoder always returns a valid recovery operator but decoding succeeds if and only if the neural network correctly identifies the logical operator $\overline{L}$ implemented by $R_U Q$. The output of the trained neural network is found by multiplying matrices of size determined by the number of nodes in each layer, and thus the decoding complexity will scale polynomially [6] with the number of nodes. We see that in step 1 we implicitly make use of the excitation graph, which contains information about the topological code lattice and the fusion rules. However, no information about the topological code is required to train the neural network, which is used in step 2.

[6] One may increase the computation speed by sparsifying the matrices of the final trained network. Another possibility of reducing the complexity of decoding is to consider convolutional neural networks, which may have much fewer nodes and links than the networks we consider.

Algorithm 2: neural decoder

```
Require: excitation removal algorithm, trained neural network
Input: locations of excitations $U \subset V \setminus \{w\}$ created by some unknown operator $Q$
Output: recovery operator $R$
using the excitation removal algorithm with $U$ as the input, find an operator $R_U$
using the neural network with $U$ as the input, find the logical operator $\overline{L}$ implemented by $R_U Q$
$R \leftarrow R_U L$, where $L$ is any operator implementing $\overline{L}$
return $R$
```

We emphasize that the details of step 1 in the neural decoder do not matter as long as the returned operator R_U is found in an efficient deterministic way. We choose the excitation removal algorithm because it is simple and has an intuitive explanation — it removes all the excitations by moving them to the boundaries of the system. We point out that we could use a similar version of the neural decoder for other topological codes (or even codes without geometric structure), as long as we knew how to efficiently find the operator R_U. For instance, if we considered the toric or color codes on a torus, with or without boundaries, then we could always find a simple removal procedure which deterministically moves all excitations of the same color to the same location in the bulk or on the boundary, where they are guaranteed to disappear. Such a procedure can then be used to create the training dataset for the neural network. We remark that step 1 becomes more challenging for codes without string-like operators, such as the cubic code [15].

B. Training deep neural networks

Before a neural network can be used for decoding, it needs to be trained. We do this via supervised learning, where the network is trained on a dataset of preclassified samples. Sample Pauli errors are generated using Monte Carlo sampling according to the appropriate probability distribution determined by the noise model. For each generated error configuration Q, we determine the corresponding syndrome, i.e., the excitation configuration $U \subset V \setminus \{w\}$, which is the input to the neural network. Then, using the excitation removal algorithm, we find the Pauli operator R_U, and check what logical operator $\overline{L}$ is implemented by $R_U Q$. This allows us to label each input excitation configuration U with the corresponding classification label $\overline{L}$ we want the neural network to output. We remark that the testing samples used to numerically estimate thresholds are created in the same way as the training samples.

Training the neural network can now be framed as a minimization problem. The network parameters, i.e., the weights and biases, are optimized to minimize classification error on the training dataset. We use the categorical cross entropy cost function C to quantify the error, namely

$$C = \sum_i \vec{y_i} \cdot \log\left(\vec{f}(\vec{x_i})\right) + (\vec{1} - \vec{y_i}) \cdot \log\left(\vec{1} - \vec{f}(\vec{x_i})\right), \quad (10)$$

where $\vec{y_i}$ is the classification bit-string for the input $\vec{x_i}$, $\vec{f}(\vec{x_i})$ is the likelihood vector returned by the neural network, and $\vec{1} = (1, \dots, 1)$. Importantly, this cost function is differentiable, which allows us to use backpropagation to efficiently compute the gradient of the cost function with respect to network parameters in a single backwards pass of the network. The minimization is performed using Adam optimization [68], a highly effective variant of gradient descent, whose learning parameters do not need to be fine-tuned for good performance. In practice, we find that Adam optimization converges significantly faster than standard gradient descent, with the effects becoming more pronounced for larger networks.

Instead of computing the cost function on the entire training set, which becomes computationally expensive for very large datasets, we use mini-batch optimization. This is a standard technique, which estimates the cost function on individual batches, i.e., small subsets of the training datasets; see e.g. [69]. We define a training step as one round of backpropagation and a subsequent network parameter update, using the cost function C in Eq. (10) estimated on a single batch. The batch size controls the accuracy of this estimate and needs to be manually adjusted.

Until recently, training deep neural networks had been next to impossible. However, innovations by the machine learning community have made it easy to train extremely deep networks. We too were unable to successfully train networks with more than three hidden layers, until we implemented two of these improvements: He initialization and batch normalization. He initialization [70] ensures that learning is efficient for the rectified linear unit activation function, whereas batch normalization [71] stabilizes the input distribution for each layer. Batch normalization makes it possible to train deeper networks and improves performance on shallower three-layer networks.

One disadvantage of larger networks is the potential for overfitting. This occurs when free parameters of a model learn features in the training dataset which do not generalize to other samples from the target distribution, causing performance to suffer as shown in Fig 5(a). Overfitting can be countered by reducing the size of the network or by increasing the size of the training dataset. In our application, training data can be generated with little computational overhead. As such, we avoid the problem of overfitting by generating training samples in parallel with training. Each training step uses a new batch of data, ensuring the model generalizes perfectly to other samples from the same distribution as evidence from Fig. 5(b).

The training set is generated according to the noise model and some chosen error rates. Once the neural network is trained, it should be able to successfully label

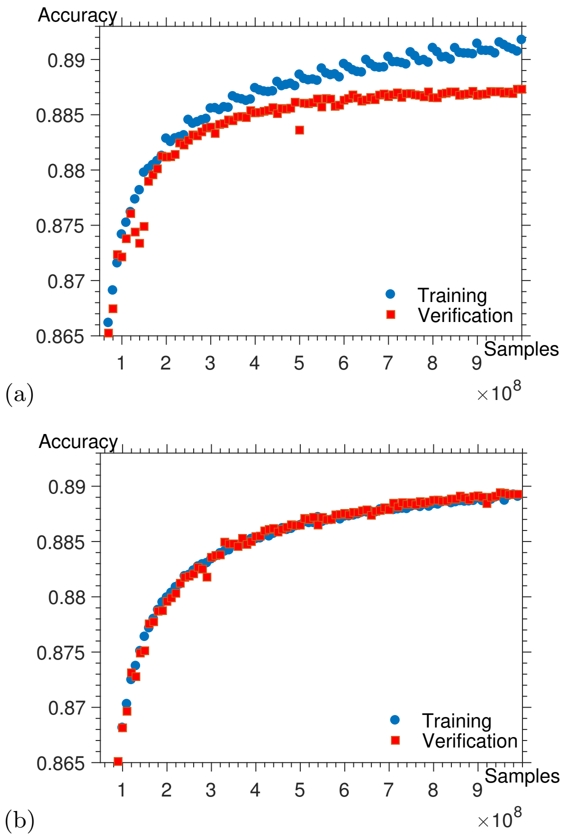


FIG. 5. Performance of the network as a function of training steps. (a) Finite training set of 5×10^7 samples is repeatedly iterated over. The network begins overfitting as performance on a validation dataset begins decreasing. (b) Training samples are generated in parallel and no overfitting occurs even after many training steps.

syndromes for error configurations generated at various error rates below the threshold. In particular, any fine-tuning of the network for specific error rates is not desired. Since the error syndromes for higher error rates are in general more challenging to classify, it would be desirable to train the neural network mainly on configurations corresponding to error rates close to the threshold. However, during training of the networks for higher-distance codes and correlated noise models the optimization algorithm is very likely to get stuck in local minima if we start training on the high error-rate dataset directly. This problem is manifested in the network not effectively learning the noise features and the resulting performance showing only small improvements over random guessing. A solution we propose is to first pre-train the network on a lower error-rate dataset, and only then use the training data corresponding to the near-threshold error rate. We find that as the code size increases, then one benefits from increasing the size of the dataset from the low error-rate regime (as well as the number of considered different low error rates) used in the pre-training stage; further study of the optimal pre-training would be valuable. We believe that this is an important observation for any future implementations of neural networks for decoding quantum error-correcting codes. We also speculate that a similar strategy might help to speed up training of neural networks for experimental systems. Namely, we imagine pretraining the neural network for some simple theoretical error models at low error rates, and then using the experimental data for further training.

training cost for the triangular color code					
noise \ parameters	d	H_d	N_d	B_d	T_d
bit-/phase-flip	5	3	100	10^3	3×10^4
	7	5	200	5×10^3	5×10^4
	9	7	400	10^4	1.1×10^5
	11	9	800	10^4	2.1×10^5
depolarizing	5	3	200	10^4	1.1×10^5
	7	5	600	10^4	3×10^5
	9	7	1400	10^4	4.1×10^5
NN-depolarizing	5	3	200	5×10^3	6×10^4
	7	5	400	10^4	1.1×10^5
	9	7	800	10^4	2.1×10^5
	11	9	1600	10^4	4.1×10^5

training cost for the triangular toric code with a twist					
noise \ parameters	d	H_d	N_d	B_d	T_d
bit-/phase-flip	5	3	100	10^3	3×10^4
	7	5	200	10^4	6×10^4
	9	7	400	10^4	1.6×10^5
	11	9	800	10^4	2.6×10^5
depolarizing	5	3	200	5×10^3	3×10^4
	7	5	600	10^4	1.1×10^5
	9	7	1200	10^4	2.1×10^5
NN-depolarizing	5	3	200	5×10^3	6×10^4
	7	5	400	10^4	1.1×10^5
	9	7	800	10^4	2.1×10^5
	11	9	1600	10^4	4.1×10^5

TABLE II. Optimal neural-network hyperparameters of the neural decoder for the triangular color code (top) and the triangular toric code with a twist (bottom) with distance d under different noise models. Hyperparameters varied are: the number of hidden layers H_d, the number of nodes in the hidden layer N_d, the batch size B_d and the number of training steps T_d. The total number of training samples seen during training is $B_d T_d$.

C. Selecting neural-network hyperparameters

In addition to network parameters, there are also hyperparameters which cannot be trained via backpropagation. These include the number of hidden layers H_d, the number of nodes per hidden layer N_d, the size of each batch B_d, and the total number of training steps T_d. We optimize these parameters using a grid search based approach; see Table II for the optimal values we find. A heuristic rule for determining the size of a well-performing neural network for the code with distance d is to use $H_d = d - 2$ hidden layers and $N_d \propto 2^{d/2}$ nodes per layer. Whether or not this exponential trend continues for larger code sizes is an open question. Note that the runtime of the decoder, once the trained network is found, scales polynomially with the number of nodes and layers, and as such exponential scaling would be limiting. However, we considered fully-connected neural networks here when more sparsely connected graphs along the lines of the RG decoding, such as convolutional neural networks [72, 73], may allow to push to higher distances more easily.

We notice that very large training sets are needed for optimal performance. In order to save on computational memory, we choose to generate training samples in parallel to training, since it can be done efficiently. Note that with this strategy the number of different samples seen during training is $B_d T_d$. We observe that the training time appears to scale exponentially with code distance, approximately doubling as the distance increases by two.

We find evidence that there is some minimal batch size below which the gradient estimates are too noisy for the network to converge to a solution that outperforms random guessing. However, increasing the batch size beyond that minimal value does not improve the final network performance. Rather, it reduces the number of training steps needed for convergence, but with diminishing returns. The batch size we choose is primarily optimized to minimize the training time.

D. Thresholds of neural decoders

In order to assess the versatility of neural-network decoding, we qualitatively study its performance for the toric and color codes under three different noise models: bit-/phase-flip, depolarizing and NN-depolarizing. First, we train a neural network for every code with the code distance up to $d = 11$. The optimized hyperparameters of considered neural networks are presented in Table II. Then, we numerically find the decoder failure probability $p_{\text{fail}}(p_{\text{eff}}, d)$ of the neural decoder as a function of the effective error rate p_{eff}. By plotting the decoder failure probability $p_{\text{fail}}(p_{\text{eff}}, d)$ for different code distances d and finding their intersection we numerically establish the existence of non-zero threshold for the neural decoder and estimate its value; see Figs. 6 and 7.

We benchmark the performance of the neural decoder against the leading efficient decoders of the toric and color code. In particular, we analyze the standard decoders based on the Minimum-Weight Perfect Matching algorithm and the projection decoder. In our implementation, we use the Blossom V algorithm provided by Kolmogorov [74].

We report that the neural decoder for the color code significantly outperforms the projection decoder for all considered noise models, even for the simplest bit-/phase-flip noise model. The neural decoder threshold values we

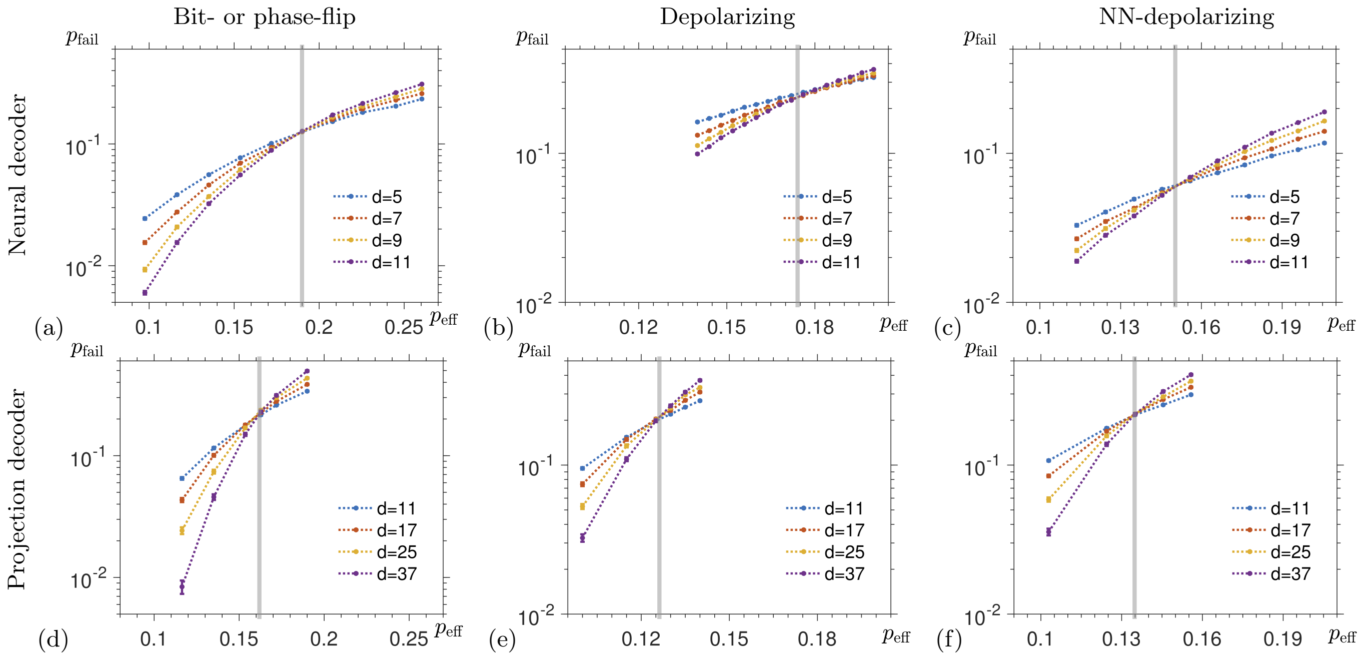


FIG. 6. The failure probability $p_{\mathrm{fail}}(p_{\mathrm{eff}}, d)$ of (a)-(c) the neural decoder and (d)-(f) the projection decoder for the 2D triangular color code of distance d as a function of the effective error rate p_{eff}. We consider three noise models: (a),(d) bit-/phase-flip, (b),(e) depolarizing and (c),(f) NN-depolarizing. We report that the neural decoder outperforms the projection decoder for all types of noise, exhibiting threshold near the optimal one. Threshold values are given by light grey lines.

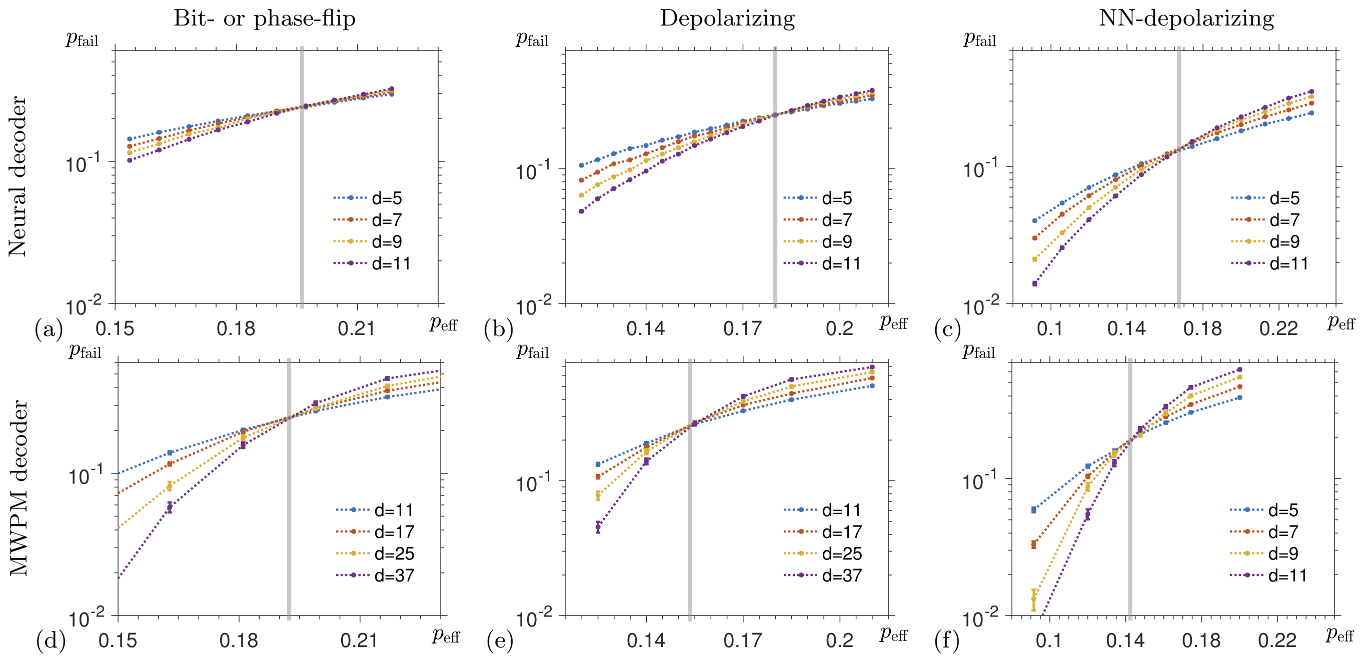


FIG. 7. The failure probability $p_{\mathrm{fail}}(p_{\mathrm{eff}}, d)$ of the (a)-(c) the neural decoder and (d)-(f) the Minimum-Weight Perfect Matching (MWPM) decoder for the 2D triangular toric code with a twist of distance d as a function of the effective error rate p_{eff}. We consider three noise models: (a),(d) bit-/phase-flip, (b),(e) depolarizing and (c),(f) NN-depolarizing. We report that the neural decoder significantly outperforms the Minimum-Weight Perfect Matching decoder for noise models with correlated errors and exhibits threshold near the optimal one. Threshold values are given by light grey lines.

find approach the upper bounds from the maximum likelihood decoder. The neural decoder for the toric code shows comparable performance as the Minimum-Weight Perfect Matching decoder for the bit-/phase-flip noise, however offers noticeable improvements for correlated noise models. We remark that optimal decoding thresholds for topological codes can be found via statistical-mechanical mapping; see [24, 40, 41, 75]. The threshold values we find are expressed in terms of the effective error rate p_{eff} and are listed in Table I. It should be noted that for the correlated noise model we considered, the optimal threshold has yet to be established. However a recent extension of the ideas of statistical-mechanical mappings could be used to study the nearest-neighbor noise model [76].

As with all learning models, it is important to address the possibility of overfitting. We know that the test samples are different (with high probability) from the training samples, since they are randomly chosen from a set that scales exponentially with the number of physical qubits. We remark that the required training set seems to scale exponentially with the code distance, however it constitutes a vanishing fraction of all possible syndrome configurations. Moreover, the classification accuracy on the test samples is the same as the final training accuracy. Thus, we can conclude that the neural network learns to correctly label syndromes typical for the studied noise models, resulting in well-performing neural decoders.

IV. DISCUSSIONS

We have conclusively demonstrated that neural-network decoding for topological stabilizer codes is very versatile and clearly outperforms leading efficient decoders. We focused on the triangular color code and the toric code a twist, whose physical qubits are arranged in the same way but their stabilizer groups are different. We studied the performance of neural-network decoding for different noise models, including the spatially-correlated depolarizing noise. In particular, we numerically established the existence of non-zero threshold and found significant improvements of the color code threshold over the previously reported values; see Table I and Figs. 6 and 7. This result indicates that the relatively low threshold of the color code, which was considered to be one of its main drawbacks, can be easily increased, making quantum computation with the color code more appealing than initially perceived [77–79].

We emphasize that the neural network does not explicitly use any information about the topological code or the noise model. The neural network is trained on very simple data usually available from the experiment, which includes the information about the measured syndrome and whether the simple deterministic decoding, i.e., the excitation removal algorithm, succeeds. Importantly, this raw data can not only be used to train the neural network, but also to characterize the quantum device [80]. Without assuming any simplistic noise models the neural network efficiently detects the actual error patterns in the system and subsequently "learns" about the correlations between observed errors. This provides a heuristic explanation why neural decoding is currently the best strategy to decode the color code, since the correlations between errors in the color code are difficult to account for in standard approaches [81]. Using neural networks simplifies and speeds up the process of designing good decoders, which is rather challenging due to its heavy dependency on the choice of the quantum error-correcting code as well as the noise model.

Our results show that neural-network decoding can be successfully used for quantum error-correction protocols, especially in the systems affected by a priori unknown noise with correlated errors. Moreover, the neural decoder yields a higher threshold for decoding the triangular color code than any previously known method. In particular, it shows similar threshold behavior for the 2D color and toric codes, suggesting that the color code may be more competitive than previously thought with the surface-code architecture. However, in order to solidify such a claim, neural decoders addressing circuit-level noise should be considered. We note that the networks studied in this work were not optimized for efficiency, but rather served as a proof of principle that decoding with such networks is possible and yield good performance regardless of the noise type. Moreover, we stress that neural-network decoding already provides an enormous data-compression advantage over methods based on (partial) look-up tables, even for small-distance quantum codes. However, an important question of scalability has to be addressed if neural decoders are ever going to be used for practical purposes on future fault-tolerant universal quantum devices. One possible approach to scalable neural networks is to reduce the connectivity between the layers by exploiting the information about the topological code lattice and geometric locality of stabilizer generators. We imagine incorporating convolutional neural networks as well as some renormalization ideas in the future scalable neural decoders. Such an idea was recently studied in a follow-up work in the context of the toric code [47]. Also, a fully-fledged neural decoder should account for the possibility of faulty stabilizer measurements [82–84]. We do not perceive any fundamental reasons why neural-network decoding, possibly based on recurrent neural networks, would not work for the circuit level noise model. However, in that setting the training dataset as well as the size of the required neural network grow substantially, making the training process computationally very challenging.

ACKNOWLEDGMENTS

We would like to thank Ben Brown, Jenia Mozgunov and John Preskill for valuable discussions, as well as Evert van Nieuwenburg for his feedback on this manuscript. During the preparation of the manuscript two related preprints were made available [85, 86], however their scope and emphasis are different from our work. NM acknowledges funding provided by the Caltech SURF program. AK acknowledges funding provided by the Simons Foundation through the "It from Qubit" Collaboration.

Research at Perimeter Institute is supported by the Government of Canada through Industry Canada and by the Province of Ontario through the Ministry of Research and Innovation. TJ acknowledges the support from the Walter Burke Institute for Theoretical Physics in the form of the Sherman Fairchild Fellowship. The authors acknowledge the support from the Institute for Quantum Information and Matter (IQIM).

[1] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. C. White, J. Mutus, A. G. Fowler, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, C. Neill, P. O'Malley, P. Roushan, A. Vainsencher, J. Wenner, A. N. Korotkov, A. N. Cleland, and J. M. Martinis, Nature **508**, 500 (2014).

[2] A. D. Córcoles, E. Magesan, S. J. Srinivasan, A. W. Cross, M. Steffen, J. M. Gambetta, and J. M. Chow, Nature communications **6**, 6979 (2015).

[3] J. Kelly, R. Barends, A. G. Fowler, A. Megrant, E. Jeffrey, T. C. White, D. Sank, J. Y. Mutus, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, I. C. Hoi, C. Neill, P. J. J. O'Malley, C. Quintana, P. Roushan, A. Vainsencher, J. Wenner, A. N. Cleland, and J. M. Martinis, Nature **519**, 66 (2015).

[4] D. Nigg, M. Mueller, E. A. Martinez, P. Schindler, M. Hennrich, T. Monz, M. A. Martin-Delgado, and R. Blatt, Science **345**, 302 (2014).

[5] A. Kitaev, A. Shen, and M. Vyalyi, *Classical and Quantum Computation* (American Mathematical Society, 2002) p. 257.

[6] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*, 10th ed. (Cambridge University Press, 2010) p. 702.

[7] P. W. Shor, Physical Review A **52**, R2493 (1995), arXiv:0506097 [arXiv:quant-ph].

[8] D. Gottesman, Physical Review A **54**, 1862 (1996), arXiv:9604038 [quant-ph].

[9] M.-H. Hsieh and F. Le Gall, Physical Review A **83**, 052331 (2011).

[10] P. Iyer and D. Poulin, IEEE Transactions on Information Theory **61**, 5209 (2015), arXiv:1310.3235.

[11] A. Y. Kitaev, Annals Phys. **303**, 2 (2003), arXiv:9707021 [quant-ph].

[12] S. B. Bravyi and A. Y. Kitaev, , 6 (1998), arXiv:9811052 [quant-ph].

[13] H. Bombin and M. A. Martin-Delgado, Physical Review Letters **97**, 180501 (2006), arXiv:0605138 [quant-ph].

[14] H. Bombin, in *Topological Codes*, edited by D. A. Lidar and T. A. Brun (Cambridge University Press, 2013) arXiv:1311.0277.

[15] J. Haah, Physical Review A **83**, 042330 (2011).

[16] J. Harrington, *Analysis of quantum error-correcting codes: symplectic lattice codes and toric codes*, Ph.D. thesis, California Institute of Technology (2004).

[17] M. B. Hastings, , 1 (2013), arXiv:1312.2546.

[18] M. Herold, E. T. Campbell, J. Eisert, and M. J. Kastoryano, npj Quantum Information **1** (2015), 10.1038/npjqi.2015.10, arXiv:1406.2338.

[19] M. Herold, M. J. Kastoryano, E. T. Campbell, and J. Eisert, New Journal of Physics **19** (2017), 10.1088/1367-2630/aa7099, arXiv:1511.05579.

[20] K. Duivenvoorden, N. P. Breuckmann, and B. M. Terhal, (2017), arXiv:1708.09286.

[21] G. Dauphinais and D. Poulin, Commun. Math. Phys. **355**, 519 (2017), arXiv:1607.02159.

[22] A. Kubica, *The ABCs of the color code: A study of topological quantum codes as toy models for fault-tolerant quantum computation and quantum phases of matter*, Ph.D. thesis, California Institute of Technology (2018).

[23] A. Kubica and J. Preskill, arXiv:1809.10145 (2018).

[24] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Journal of Mathematical Physics **43**, 4452 (2002), arXiv:0110143 [quant-ph].

[25] N. Delfosse, Physical Review A **89**, 012317 (2014), arXiv:1308.6207.

[26] N. H. Nickerson and B. J. Brown, (2017), arXiv:1712.00502.

[27] S. Bravyi, M. Suchara, and A. Vargo, Physical Review A - Atomic, Molecular, and Optical Physics **90** (2014), 10.1103/PhysRevA.90.032326, arXiv:1405.4883.

[28] A. S. Darmawan and D. Poulin, (2018), arXiv:1801.01879.

[29] G. Duclos-Cianci and D. Poulin, Physical Review A **87**, 062338 (2013), arXiv:1302.3638.

[30] G. Duclos-Cianci and D. Poulin, Quantum Information & Computation **14**, 721 (2014).

[31] N. P. Breuckmann, K. Duivenvoorden, D. Michels, and B. M. Terhal, Quantum Information and Computation **17**, 0181 (2017), arXiv:1609.00510.

[32] S. Bravyi and J. Haah, Physical Review Letters **111**, 200501 (2013).

[33] B. J. Brown, N. H. Nickerson, and D. E. Browne, Nature Communications **7**, 4 (2015), arXiv:1503.08217.

[34] T. M. Stace and S. D. Barrett, Physical Review A **81**, 022317 (2010).

[35] S. Bravyi, M. Suchara, and A. Vargo, Physical Review A **90**, 032326 (2014).

[36] N. Delfosse and G. Zémor, (2017), arXiv:1703.01517.

[37] N. Delfosse and N. H. Nickerson, (2017), arXiv:1709.06218.

[38] B. Criger and I. Ashraf, arXiv:1709.02154 (2017).

[39] A. Kubica, B. Yoshida, and F. Pastawski, New Journal of Physics **17**, 083026 (2015).

[40] H. G. Katzgraber, H. Bombin, and M. A. Martin-Delgado, Physical Review Letters **103**, 090501 (2009).

[41] H. Bombin, R. S. Andrist, M. Ohzeki, H. G. Katzgraber, and M. A. Martin-Delgado, Physical Review X **2**, 021004 (2012).

[42] G. Torlai and R. G. Melko, Physical Review Letters **119** (2017), 10.1103/PhysRevLett.119.030501, arXiv:1610.04238.

[43] P. Baireuther, T. E. O'Brien, B. Tarasinski, and C. W. J. Beenakker, (2017), 10.22331/q-2018-01-29-48, arXiv:1705.07855.

[44] S. Krastanov and L. Jiang, Scientific Reports **7** (2017), 10.1038/s41598-017-11266-1, arXiv:1705.09334.

[45] S. Varsamopoulos, B. Criger, and K. Bertels, (2017), arXiv:1705.00857.

[46] N. P. Breuckmann and X. Ni, (2017), arXiv:1710.09489.

[47] X. Ni, arXiv:1809.06640 (2018).

[48] A. Calderbank, E. Rains, P. Shor, and N. Sloane, Physical Review Letters **78**, 405 (1997), arXiv:9605005 [quant-ph].

[49] E. Knill and R. Laflamme, arXiv: quant-ph/9608012 (1996).

[50] T. J. Yoder and I. H. Kim, Quantum **1**, 2 (2017), arXiv:1612.04795.

[51] B. J. Brown, D. Loss, J. K. Pachos, C. N. Self, and J. R. Wootton, Rev. Mod. Phys. **88**, 045005 (2016).

[52] F. Wilczek, Physical Review Letters **49**, 957 (1982).

[53] J. Preskill, *Lecture notes for Physics 219: Quantum computation* (1999).

[54] H. Bombin and M. Martin-Delgado, Physical Review B **75**, 075103 (2007), arXiv:0607736 [cond-mat].

[55] M. Levin, Physical Review X **3** (2013), 10.1103/PhysRevX.3.021009, arXiv:1301.7355.

[56] H. Bombín, Communications in Mathematical Physics **327**, 387 (2014), arXiv:1107.2707.

[57] J. Haah, Communications in Mathematical Physics **324**, 351 (2013).

[58] H. Bombin, Physical Review Letters **105** (2010), 10.1103/PhysRevLett.105.030403, arXiv:1004.1838.

[59] H. Bombin, New Journal of Physics **13** (2011), 10.1088/1367-2630/13/4/043005, arXiv:1006.5260.

[60] A. Kitaev and L. Kong, Communications in Mathematical Physics **313**, 351 (2012), arXiv:1104.5047.

[61] B. Yoshida, Physical Review B - Condensed Matter and Materials Physics **91** (2015), 10.1103/PhysRevB.91.245131, arXiv:1503.07208.

[62] M. Kesselring, B. Brown, , F. Pastawski, and E. J., in preparation (2018).

[63] A. G. Fowler, Quantum Information and Computation **15**, 0145 (2015).

[64] D. Poulin, Phys. Rev. A **74**, 052333 (2006).

[65] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Physical Review A **86**, 032324 (2012).

[66] K. Hornik, M. Stinchcombe, and H. White, Neural networks **2**, 359 (1989).

[67] A. Géron, *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems* (" O'Reilly Media, Inc.", 2017).

[68] D. P. Kingma and J. Ba, arXiv preprint arXiv:1412.6980 (2014).

[69] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning-lecture 6a-overview of mini-batch gradient descent," (2012).

[70] K. He, X. Zhang, S. Ren, and J. Sun, in *Proceedings of the IEEE international conference on computer vision* (2015) pp. 1026–1034.

[71] S. Ioffe and C. Szegedy, in *International conference on machine learning* (2015) pp. 448–456.

[72] Y. LeCun *et al.*, Connectionism in perspective , 143 (1989).

[73] A. Krizhevsky, I. Sutskever, and G. E. Hinton, in *Advances in neural information processing systems* (2012) pp. 1097–1105.

[74] V. Kolmogorov, Mathematical Programming Computation **1**, 43 (2009).

[75] A. Kubica, M. E. Beverland, F. Brandão, J. Preskill, and K. M. Svore, Phys. Rev. Lett. **120**, 180501 (2018).

[76] C. T. Chubb and S. T. Flammia, arXiv preprint arXiv:1809.10704 (2018).

[77] D. S. Wang, A. G. Fowler, C. D. Hill, and L. C. L. Hollenberg, Quantum Information and Computation **10**, 780 (2010), arXiv:0907.1708.

[78] A. G. Fowler, Physical Review A - Atomic, Molecular, and Optical Physics **83** (2011), 10.1103/PhysRevA.83.042310, arXiv:0806.4827.

[79] A. J. Landahl and C. Ryan-Anderson, , 13 (2014), arXiv:1407.5103.

[80] J. Combes, C. Ferrie, C. Cesare, M. Tiersch, G. J. Milburn, H. J. Briegel, and C. M. Caves, , 16 (2014), arXiv:1405.5656.

[81] N. Delfosse and J. P. Tillich, in *IEEE International Symposium on Information Theory - Proceedings* (2014) pp. 1071–1075, arXiv:1401.6975.

[82] P. Baireuther, T. E. O'Brien, B. Tarasinski, and C. W. Beenakker, Quantum **2**, 48 (2018).

[83] C. Chamberland and P. Ronagh, arXiv preprint arXiv:1802.06441 (2018).

[84] P. Baireuther, M. Caio, B. Criger, C. Beenakker, and T. O'Brien, arXiv preprint arXiv:1804.02926 (2018).

[85] A. Davaasuren, Y. Suzuki, K. Fujii, and M. Koashi, (2018), arXiv:1801.04377.

[86] Z.-A. Jia, Y.-H. Zhang, Y.-C. Wu, L. Kong, G.-C. Guo, and G.-P. Guo, (2018), arXiv:1802.03738.