Decoders for Topological Quantum Error Correction

Arthur Pesah[1]

[1] *Department of Physics and Astronomy, University College London, London WC1E 6BT, UK*

(Dated: November 2020)

Decoders are an essential component of quantum error correction, having a large influence on the performance of a given code. After having introduced the basic of quantum error correction and the surface code, we review the different decoding algorithms proposed in the literature.

I. INTRODUCTION

In the early days of quantum computing, most of the scepticism towards this new technology came from the hardness of experimentally keeping and manipulating a quantum state error-free for a long period of time [1]. The invention of quantum error correction created a new wave of optimism in the field, proving that building a fault-tolerant quantum computer, while challenging, was not completely beyond our reach: as long as we manage to engineer qubits that resist to noise below a certain threshold, it is possible to detect and correct errors at an arbitrary precision.

Since then, many quantum error correcting codes have been proposed. One of the most promising classes of codes is the surface code, belonging to the more general category of topological quantum error correction. The original idea, proposed by Alexei Kitaev in 1997, is to encode each single logical qubit in a 2D lattice of physical qubits with periodic boundary conditions, or in other words, a torus. He showed that by encoding quantum information in a specific way on the torus, it is possible to correct a wide ranges of errors affecting the qubits.

An important subroutine of quantum error correction is the decoding algorithm: given the result of partial measurements on the system, what is the most likely error that occurred and how can we correct it? While an optimal procedure exists to correct the most likely errors, the corresponding algorithm takes an exponential time with respect to the number of qubits and is therefore often considered unpractical [2]. Therefore, the main challenge in designing a decoding scheme lies in the trade-off between performance and running time (or complexity) of the algorithm. A lot of creative solutions have been proposed, based on a variety of techniques from graph algorithms [3, 4] to neural networks [5].

In this paper, we will review the main ideas behind quantum error correction and the surface code, as well as the different types decoding algorithms.

II. QUANTUM ERROR CORRECTION

When running a circuit on a quantum computer, we expect a state $|\psi\rangle$ to be produced at each cycle of the device. However, due to the noise, an alternative state $|\tilde{\psi}\rangle$ might be produced with a certain probability. Any quantum error correction process will consist of the following four steps:

1. **Encode** each logical state $|\psi\rangle$ as a state $|\psi\rangle_E$ defined on more qubits.
2. **Detect** if an error happened by performing non-destructive measurements on the physical state $|\tilde{\psi}\rangle_E$
3. **Correct** those errors and **decode** the qubits to get the corresponding logical state
4. **Compute** logical operations on the state by redefining a universal gate set on the code.

In this review, we will mostly be interested in the first three steps, which can be summarized by the diagram below:

$$|\psi\rangle \xrightarrow{\text{encoding}} |\psi\rangle_E \xrightarrow{\text{noise}} |\tilde{\psi}\rangle_E \xrightarrow{\text{decoding}} |\psi\rangle$$

We review in this section the basic formalism of quantum error correction.

A. Noise models

An apparent problem with the concept of error correction in the quantum domain is that noise is able to deform our state in infinitely many ways. For instance, we could construct a noise model where a rotation gate $\hat{\mathrm{R}}_z(\theta)$ is applied randomly to each qubit at every cycles:

$$|\psi\rangle \xrightarrow{noise} \hat{\mathrm{R}}_z(\theta)\,|\psi\rangle$$

However, this apparent challenge can easily be solved by noticing that all quantum errors can in fact be decomposed into only two types of errors:

$$\text{Bit-flips: } |\psi\rangle \longrightarrow \hat{\mathrm{X}}\,|\psi\rangle$$
$$\text{Phase-flips: } |\psi\rangle \longrightarrow \hat{\mathrm{Z}}\,|\psi\rangle$$

where $\hat{\mathrm{X}}$ and $\hat{\mathrm{Z}}$ are Pauli operators. For instance, our rotation $R_x(\theta)$ can be written:

$$\hat{\mathrm{R}}_x(\theta) = \cos\left(\frac{\theta}{2}\right)\hat{\mathrm{I}} - i\sin\left(\frac{\theta}{2}\right)\hat{\mathrm{X}}$$

and our model would be equivalent to applying a bit-flip with probability $\sin^2\left(\frac{\theta}{2}\right)$ and nothing with probability

$\cos^2\left(\frac{\theta}{2}\right)$. Another example is the $\hat{Y}$ Pauli error, which is equivalent to applying both a phase-flip and a bit-flip, due to the relation:

$$\hat{Y} = i\hat{X}\hat{Z}$$

Therefore, a noise model only needs to specify the probability of having $\hat{I}$ (no error), $\hat{X}$, $\hat{Z}$, or both (i.e. $\hat{Y}$) at each cycle. For example, the depolarizing noise model assigns equal probability $p/3$ to all three Pauli operators:

$$\hat{I} : (1-p)$$
$$\hat{X}, \hat{Y}, \hat{Z} : p/3$$

Another important noise model is the independent (or uncorrelated) noise model, in which $\hat{X}$ and $\hat{Z}$ errors occur independently with equal probability:

$$\hat{I} : (1-p)^2$$
$$\hat{X}, \hat{Z} : p(1-p)$$
$$\hat{Y} : p^2$$

This decomposition of all errors into $\hat{X}$ and $\hat{Z}$ errors will greatly simplify the error correcting process.

B. Encoding of a qubit

A code is a function that takes a logical state $|\psi\rangle$ to another state $|\psi\rangle_E$ defined on more qubits. For instance, the *repetition code* is defined in the following way:

$$|\psi\rangle = a|0\rangle + b|1\rangle \xrightarrow{encoding} |\psi\rangle_E = a|000\rangle + b|111\rangle$$

The set $\{a|000\rangle + b|111\rangle\}$ of all the possible encoding results is called the *codespace*. If an error occurs on one qubit, for instance a bit-flip on the first qubit, the state will move out of the codespace:

$$|\psi\rangle_E = a|000\rangle + b|111\rangle \xrightarrow{X_1} |\tilde{\psi}\rangle_E = a|100\rangle + b|011\rangle$$

and it will be possible—at least in principle—to detect the error and potentially correct it.

An important characteristic of a code is its *distance*: the minimal number of one-qubit errors necessary to pass from one codespace element to another. For the repetition code, considering only $\hat{X}$ errors, the distance is $d = 3$. From the distance we can deduce two important things:

- the minimal number of errors that can be detected: $d-1$
- the minimal number of errors that can be corrected: $\lfloor \frac{d-1}{2} \rfloor$

This finally allows us to define the notion of *logical operator*: operator acting on d qubits of the code and used to change its logical state. For instance, the operator $\bar{X} = \hat{X}_1\hat{X}_2\hat{X}_3$ change the state from $|\bar{0}\rangle = |000\rangle$ to $|\bar{1}\rangle = |111\rangle$ and vice-versa.

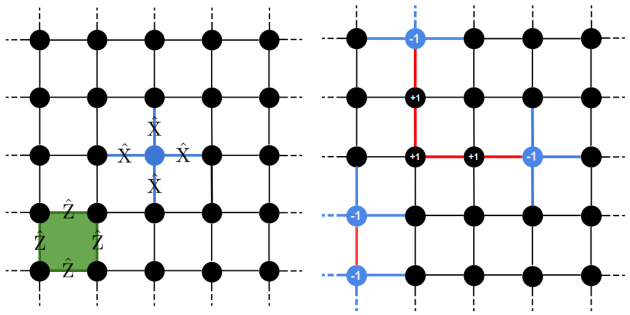

FIG. 1. (Left) Representation of the surface code for $L = 5$, along with vertex operators (blue) and plaquette operators (green). The physical qubits are placed on the edges. (Right) Strings of $\hat{Z}$ errors (red) along with the syndrome that results from measuring vertex operators

C. Stabilizer formalism

To detect errors, that are states outside of the codespace, we can use a set of non-destructive measurements called *stabilizers*. For a given codespace $\mathcal{C}$, we say that a set $\mathcal{S}$ of operators is a stabilizer set of $\mathcal{C}$ if:

- All the operators in $\mathcal{S}$ commute with each others, or in other words, can be measured simultaneously
- Each operator in $\mathcal{S}$ has two possible eigenvalues: $+1$ (no error detected) and -1 (error detected)
- For all $|\psi\rangle \in \mathcal{C}$ and $S \in \mathcal{S}$, $S|\psi\rangle = |\psi\rangle$ (elements of the codespace are considered error-free)

The result of all the stabilizer measurements on a given state is called the *syndrome*, and each -1 of the syndrome is called an *excitation*.

Let's illustrate this definition with the repetition code. For this code, we can define the stabilizer set $\mathcal{S} = \{\hat{Z}_1\hat{Z}_2, \hat{Z}_2\hat{Z}_3, \hat{Z}_3\hat{Z}_1\}$. Indeed, all those operators commute and measuring $\hat{Z}_i\hat{Z}_j$ will give $+1$ if the two qubits i and j are the same, and -1 if they are different. Measuring those three stabilizers will therefore tell us whether our three physical qubits are the same, or to put in another way, if the state is in the codespace. For example, if we apply those three operators to the state $|\psi\rangle = |001\rangle$, the syndrome will be $\{1, -1, -1\}$. This syndrome indicates that only the first two qubits are identical, and that therefore a bit-flip probably occurred on the third qubit.

While the power of the stabilizer formalism might not be obvious from the repetition code example, we will see that it plays a crucial role when defining topological quantum error correcting codes.

III. SURFACE CODE

A. Definition

We will now consider a particular type of code, called the *surface code* [6]. In this code, the physical qubits are placed on the edges of a 2D lattice with periodic boundary conditions, as represented in Figure 1. The code is defined in a slightly different way than before: instead of starting from the codespace and deducing the stabilizers, we will start from the stabilizers. For the surface code, we construct two types of stabilizers: vertex operators—defined on each vertex as the product of $\hat{X}$ operators on the adjacent qubits—, and plaquette operators—defined on each face as the product of $\hat{Z}$ operators on the bordering qubits. We can easily verify that all those operators commute, as either zero or two $\hat{X}$ and $\hat{Z}$ operators intersect between plaquette and vertex stabilizers. Moreover, it is possible to define a dual lattice where all the plaquette operators become vertex operators and vice-versa. Therefore, every results obtained using one type of operators can be readily generalized to the other.

B. Error detection

Let us now see what happens when a $\hat{Z}$ error occurs on our code. One can see from Figure 1 (right) that all $\hat{Z}$ errors can be seen as a set of strings on the lattice. Since any plaquette operator P is made only of $\hat{Z}$ operators, it will always commute with our $\hat{Z}$ error E and returns $+1$ when measured: $PE|\psi\rangle = EP|\psi\rangle = E|\psi\rangle$. Therefore, we need to use vertex operators if we want to be able to detect our error. Measuring those stabilizers, we can see that only the boundary of the string will be detected by vertex operators, since boundary vertices are the only ones that intersect with the errors on an odd number of qubits (exactly one). The picture for $\hat{X}$ errors is very similar due the lattice duality. Continuous strings of edges are replaced by strings of parallel edges, and plaquette operators can be used to detect errors at the string boundaries.

What happens if a string of errors form a loop? If the loop is trivial, meaning that it does not go around the torus, the error will correct itself. Indeed, a trivial loop of $\hat{Z}$ errors is always a product of plaquette operators, and is therefore a stabilizer itself.

From this, one can see that the strings of errors going around the torus will be the only undetectable errors. Since the shortest such string has length L—with L the number of rows or columns of the lattice—we can deduce that the distance of the code is $d = L$. Those strings will also form our logical operators.

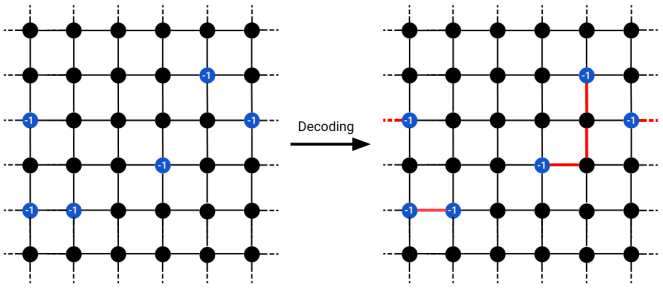

FIG. 2. A decoding algorithm takes as input a syndrome and returns a set of correcting operators (represented here as red edges)

IV. BASIC OF DECODING

A. Idea and challenges

As we saw previously, a string of errors that does not loop gives rise to a syndrome with -1 at the borders of the string. However, a given syndrome can lead to several strings of errors. The goal of decoding is to restore the logical state of the code from a given syndrome, by applying some operators on the lattice, as represented in Figure 2. We can identify two important challenges that a decoding algorithm needs to solve:

- A pair of excitations can be caused by several strings. As illustrated in Figure 3, applying a correction on a different string can be fine, as long as the resulting loop is trivial. Indeed, in that case, the total operation formed by both the errors and the correction will be inside the stabilizer group, and therefore give back the initial state. On the other hand, if the correction and the errors form a loop around the torus, the logical state of the code will be transformed and a logical error will appear.
- Given many pairs of excitation, we need to find the correct matching: which pairs of excitations have been created together with a string?

B. Evaluating a decoding algorithm

How can we evaluate and compare different decoding algorithms? There are two main metrics in the decoding literature:

- **Error threshold**: the minimal error rate such that adding more physical qubits in the code results in less logical errors. The threshold depends on the type of code, the decoding algorithm and the noise model, and can be either calculated theoretically (in some specific cases) or estimated numerically.
- **Running time**: since the decoding algorithm must be run at each clock cycle of the quantum com-

puter, it is especially important to have a low running time. This can be measured by the time complexity of the algorithm (with respect to the number N of physical qubits)

Another metric sometimes considered in the literature is the logical error rate below threshold, which depends on the size of the lattice and can give some valuable information on the practical decoding performance [7].

C. Optimal decoding

A remarkable fact about decoding is that it is possible to design an optimal algorithm—one that maximizes the threshold for all noise models—and compute this threshold theoretically in many cases. This algorithm is the Maximum Likelihood Decoder (MLD) and simply consists in finding the error compatible with the syndrome which had the highest probability of occurring. In quantitative terms, if S denotes the syndrome and $\mathcal{E}(S)$ the set errors compatible with the syndrome, the MLD solves the following problem exactly:

$$\max_{E \in \mathcal{E}(S)} P(E|S) \tag{1}$$

The optimal error threshold has been calculated to be around 11% [8] for the independent noise model and around 18.9% for depolarization [9].

However, solving the optimization problem of Eq. (1) requires to go through the exponential number of possible errors, and it can be proven that this problem is NP-Hard in general [2]. Therefore, approximations are necessary to implement a practical decoder, trading-off the threshold for a better time complexity. Let's start by describing one of the first algorithms proposed in the literature, that formulates decoding as a Minimum-Weight Perfect Matching (MWPM) problem [8, 10].

D. Minimum-Weight Perfect Matching

The MWPM formulation of decoding solves the problem of pairing excitations under two assumptions: 1) the $\hat{\mathrm{X}}$ and $\hat{\mathrm{Z}}$ errors are uncorrelated and can therefore be considered independently, and 2) the most likely pairing is the one that minimizes the total error weight, the weight of an error being defined here as the number of affected qubits. With those assumptions, the decoding problem can be seen as a graph problem: we define a complete graph whose nodes are the excitations. Each edge between two excitations s_i and s_j is given by the minimal distance in the lattice between s_i and s_j. The goal is to match all the nodes such that the total weight of the selected edges is minimal. This problem of minimum-weight perfect matching is well-known in the graph theory literature and can be solved in polynomial time using for instance the Blossom algorithm, invented by Edmonds in 1965 [11]. The threshold of this decoder can be computed theoretically, giving a value of 10.3% for an independent noise model [8], which is very close to the optimal threshold of 11%. However, this algorithm has several drawbacks:

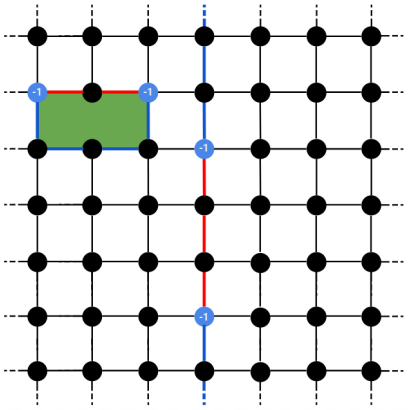

FIG. 3. The two types of loops that can occur when correcting a syndrome. On the left, the correction operator combined to the error forms a trivial loop, which is equivalent to a product of plaquette operators (green). In the middle, the correction creates a loop around the torus, resulting in a logical error

- Blossom's algorithm takes $O(N^3)$ operations in the worst case (with N the number of physical qubits), which can be too slow when the code becomes large.
- Since it assumes that errors are uncorrelated, the threshold under more realistic noise models such as depolarization is lower than many other algorithms: 14.2% compared to 18.9% for the MLD.
- The second assumption is wrong in general, since it does not take into account the degeneracy of errors with equal weight

While the last two issues are challenging to solve within this framework, solutions have been proposed for the first one. An algorithm designed by Fowler et al. approximates the MWPM problem by removing a set of well-chosen edges that are likely not to contribute to the matching [10]. Using this trick, the authors provide both theoretical and numerical evidence that the algorithm runs in average in $O(N)$ instead of $O(N^3)$. Another ingenious method to accelerate the algorithm is to parallelize it: provided a classical 2D square array that can process groups of qubits in parallel, it is possible to run the algorithm with an average constant time [3]. It also requires the error rate to be below a certain threshold, which the authors argue is probably around the MWPM error threshold.

V. MODERN DECODING ALGORITHMS

We review here some of the most promising decoding methods proposed in the recent literature. A summary

of our comparison can be found in Table 1.

A. Approximate MLD methods

While the maximum likelihood problem cannot be solved exactly, it is possible to approximate it using some conventional statistical methods. Monte Carlo Markov Chains (MCMC) is an example of approximation algorithm that can be used for decoding [13]. To find the most likely error, the idea of MCMC is to start with a random error compatible with the syndrome. Then, a random change is applied to the error (for instance a random stabilizer). If this change has a higher likelihood than the previous error, we accept it and repeat the procedure with the new error. It can be proven that this iterative process will converge to the most likely error. Another version has been proposed in [7], where the threshold is slightly lower, but the complexity is provably quadratic, the algorithm easily parallelizable and the logical error rate (below threshold) always lower than the error rate of the MWPM algorithm.

Finally, [12] proposes a different class of approximate MLD algorithms based on tensor networks. They obtain a complexity of $O(N\chi^3)$, where χ is an approximation parameter related to the bond dimension of a Matrix Product State used in the algorithm.

B. Renormalization Group

A completely different technique based on the Renormalization Group (RG) was introduced in [14] and generalized to higher-dimensional codes in [15]. The idea is to divide the lattice into small cells of four qubits. A decoder is then applied to each cell and returns the probability for each qubit to be erroneous. The probabilities in each cell are then averaged to give rise to a unique probability per cell. This process is repeated iteratively and can be used to find a relatively accurate correction in a time scaling as $O(N\log(N)$. The authors obtain a threshold of 16.4% for the depolarizing noise model, which is not far from the optimal threshold of 18.9%.

C. Neural Network-based decoders

Let's now consider a class of decoding algorithms that has gain popularity in the recent years, which uses neural networks as a way to approximate the decoding function [5, 16–21]. The motivation behind this approach is that neural networks can be trained to learn a large variety of noise models and have a very low running time. It has been shown that their pseudo-threshold is usually very high, reaching for instance 16.4% for the depolarizing noise [5]. Neural network-based decoding usually consist in the following steps:

1. Create a dataset of errors and associated syndromes using simulations

2. Using this dataset, train a neural network that takes a syndrome as input and returns the most likely error.

3. Use the trained neural network as a decoder, possibly for different noise model or on a real device

Many different architectures have been proposed for this task, including Boltzmann Machines [16], Recurrent Neural Networks [17], Convolutional Neural Networks [21], Neural Ensembles [19], and policy networks trained using reinforcement learning [22]. An important problem with this approach is the dataset generation: how to generate a dataset for codes that are too large to be simulated? To solve this issue, [18] proposes to combine RG and neural networks: by solving the decoding problem iteratively on small cells, it is possible to use a neural network trained only on a small and easily simulable code.

D. Other methods

Other promising methods have recently been proposed in the literature. An example is the Union-Find decoder [4, 23], which iteratively turns Pauli errors into losses and uses the Union-Find data structure to correct those losses. Cellular automaton decoders form another class of decoders that has recently been shown to be particularly useful for higher dimensional surface codes [24].

VI. DISCUSSION

We have introduced the main ideas behind quantum error correction and seen how the stabilizer formalism can be used to define a powerful type of encoding: the surface code. We have also studied the principles of decoding and compared some recent algorithms proposed for this task.

While a lot of progress has been made towards building more efficient and accurate decoders, many challenges still remain. An important one would be to identify codes and decoders that can be used to approximate quantum error correction on near-term devices, where the number of qubits is low and the time available for decoding is very low. In this direction, the authors of [25] designed an approximate decoding scheme tailored towards the Single-Flux Quantum technology, whose decoding time has been reduced to less than 20 ns.

Another challenge concerns neural network decoders. While they seem to have good performance in practice, it could be interesting to gain more theoretical knowledge on this type of decoders. In particular, can we interpret the function learned by the neural network as in [26]? Can we derive some scaling results showing that they will still be trainable for large codes?

Algorithm	Complexity	Complexity (parallel)	Threshold (independent)	Threshold (depolarization)
MLD	NP-Hard [2] / $O(N\chi^3)$ [12]	?	11% [8]	18.9% [9]
MWPM	$O(N)$ [10]	$O(1)$ [3]	10.3% [8]	14.3%
MCMC	$O(N^2)$ [7]	$O(N)$ [7]	?	18.5% [13] / 15.5% [7]
RG	$O(N\log(N))$ [14]	$O(\log(N))$ [14]	9% [14]	16.4% [14]
Union-Find	$O(N)$ [4]	?	9.9% [4]	?
Neural Networks	?	?	?	16.4% [5]

TABLE I. Comparison of the different decoding algorithms on four criteria: complexity without parallelization, complexity with parallelization, threshold with an independent noise model and threshold with a depolarization model. Question marks mean that we were not able to find a value in the literature

Finally, the field of decoding would benefit from having a large empirical review of all the existing decoders: many papers evaluate their algorithm on only one noise model or surface code architecture, and different numerical procedures are used by different authors. Having a general benchmark of all the existing algorithm on realistic noise models could help to shape the future directions of the field.

[1] JM Raimond and S Haroche, "Quantum computing: dream or nightmare?" DARK MATTER IN COSMOLOGY QUANTUM MEASUREMENTS EXPERIMENTAL GRAVITATION , 341 (1996).

[2] Kao-Yueh Kuo and Chung-Chin Lu, "On the hardnesses of several quantum decoding problems," arXiv preprint arXiv:1306.5173 (2013).

[3] Austin G Fowler, "Minimum weight perfect matching of fault-tolerant topological quantum error correction in average $o(1)$ parallel time," arXiv preprint arXiv:1307.1740 (2013).

[4] Nicolas Delfosse and Naomi H Nickerson, "Almost-linear time decoding algorithm for topological codes," arXiv preprint arXiv:1709.06218 (2017).

[5] Stefan Krastanov and Liang Jiang, "Deep neural network probabilistic decoder for stabilizer codes," Scientific reports **7**, 1–7 (2017).

[6] A Yu Kitaev, "Fault-tolerant quantum computation by anyons," Annals of Physics **303**, 2–30 (1997).

[7] Adrian Hutter, James R Wootton, and Daniel Loss, "Efficient markov chain monte carlo algorithm for the surface code," Physical Review A **89**, 022326 (2014).

[8] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill, "Topological quantum memory," Journal of Mathematical Physics **43**, 4452–4505 (2002).

[9] Héctor Bombin, Ruben S Andrist, Masayuki Ohzeki, Helmut G Katzgraber, and Miguel A Martín-Delgado, "Strong resilience of topological codes to depolarization," Physical Review X **2**, 021004 (2012).

[10] Austin G Fowler, Adam C Whiteside, and Lloyd CL Hollenberg, "Towards practical classical processing for the surface code," Physical review letters **108**, 180501 (2012).

[11] Jack Edmonds, "Paths, trees, and flowers," Canadian Journal of mathematics **17**, 449–467 (1965).

[12] Sergey Bravyi, Martin Suchara, and Alexander Vargo, "Efficient algorithms for maximum likelihood decoding in the surface code," Physical Review A **90**, 032326 (2014).

[13] James R Wootton and Daniel Loss, "High threshold error correction for the surface code," Physical review letters **109**, 160503 (2012).

[14] Guillaume Duclos-Cianci and David Poulin, "Fast decoders for topological quantum codes," Physical review letters **104**, 050504 (2010).

[15] Guillaume Duclos-Cianci and David Poulin, "Fault-tolerant renormalization group decoder for abelian topological codes," arXiv preprint arXiv:1304.6100 (2013).

[16] Giacomo Torlai and Roger G Melko, "Neural decoder for topological codes," Physical review letters **119**, 030501 (2017).

[17] Savvas Varsamopoulos, Koen Bertels, and Carmen Garcia Almudever, "Comparing neural network based decoders for the surface code," IEEE Transactions on Computers **69**, 300–311 (2019).

[18] Savvas Varsamopoulos, Koen Bertels, and Carmen G Almudever, "Decoding surface code with a distributed neural network–based decoder," Quantum Machine Intelligence **2**, 1–12 (2020).

[19] Milap Sheth, Sara Zafar Jafarzadeh, and Vlad Gheorghiu, "Neural ensemble decoding for topological quantum error-correcting codes," Physical Review A **101**, 032338 (2020).

[20] Paul Baireuther, Thomas E O'Brien, Brian Tarasinski, and Carlo WJ Beenakker, "Machine-learning-assisted correction of correlated qubit errors in a topological code," Quantum **2**, 48 (2018).

[21] Nikolas P Breuckmann and Xiaotong Ni, "Scalable neural network decoders for higher dimensional quantum codes," Quantum **2**, 68 (2018).

[22] Philip Andreasson, Joel Johansson, Simon Liljestrand, and Mats Granath, "Quantum error correction for the toric code using deep reinforcement learning," Quantum **3**, 183 (2019).

[23] Shilin Huang, Michael Newman, and Kenneth R Brown, "Fault-tolerant weighted union-find decoding on the toric code," arXiv preprint arXiv:2004.04693 (2020).

[24] Michael Vasmer, Dan E Browne, and Aleksander Kubica, "Cellular automaton decoders for topological quantum codes with noisy measurements and beyond," arXiv preprint arXiv:2004.07247 (2020).

[25] Adam Holmes, Mohammad Reza Jokar, Ghasem Pasandi, Yongshan Ding, Massoud Pedram, and Fred-

eric T Chong, "Nisq+: Boosting quantum computing power by approximating quantum error correction," arXiv preprint arXiv:2004.04794 (2020).
[26] Anna Dawid, Patrick Huembeli, Michał Tomza, Maciej Lewenstein, and Alexandre Dauphin, "Phase detection with neural networks: Interpreting the black box," arXiv preprint arXiv:2004.04711 (2020).