Optimizing quantum error correction codes with classical machine learning

Sean Ovens, Rahul Shekhawat, and Isaac Waller

December 2019

Abstract

A major obstacle in the practical realization of quantum computing is the noise introduced by a real-world environment. The no-cloning theorem prevents the use of classical error correction codes; therefore, the development of quantum error correction codes is a requirement for the implementation of real-world quantum systems. Such codes spread the information in one 'logical' qubit among many physical qubits in such a way that noise can be measured and the opposite transformation can be applied before measurement, restoring the original value.

In this paper, we explore cutting edge applications of stabilizer codes for quantum error correction. We will begin with an review of error models and quantum error correcting codes, along with a brief overview of the stabilizer formalism. Following this, we will introduce surface codes, which are a subclass of stabilizer codes that embed physical qubits onto a surface and encode logical states in global topological properties. Finally, we discuss some recent applications of machine learning for optimizing surface codes.

Contents

1 Introduction

We cannot clone, perforce; instead, we split
Coherence to protect it from that wrong
That would destroy our valued quantum bit
And make our computation take too long.

Correct a flip and phase - that will suffice.
If in our code another error's bred,
We simply measure it, then God plays dice,
Collapsing it to X or Y or Zed.

We start with noisy seven, nine, or five
And end with perfect one. To better spot
Those flaws we must avoid, we first must strive
To find which ones commute and which do not.

With group and eigenstate, we've learned to fix
Your quantum errors with our quantum tricks.

Daniel Gottesman [9]

Noise and interference are inescapable when operating in the physical world. While this fact may be familiar to physicists, most computer scientists choose to ignore noise in their theoretical models of computing, instead opting for more elegant noise-free models. This poses issues for the translation of these abstract models into real computers. The construction of useful physical computers, both classical and quantum, is not possible without the development of safeguards that prevent noise from destroying the result of a computation. Therefore, the development of error correction mechanisms that make computation possible in the presence of noise with minimal overhead is crucial.

A number of physical systems are being explored for the implementation of quantum computing, including ions, spins in semiconductors, and superconducting circuits. However, none of these systems allow us to implement a proper computational qubit owing to their poor efficiency and susceptibility to noise. To overcome the efficiency issues with noisy qubits, we use a collection of physical qubits to build a logical qubit, which can be much more resilient than individual qubits. Quantum error correction schemes aim to encode logical information in such a way that small errors do not change the stored information.

In this paper, we survey the frontier of quantum error correction. We begin with a brief introduction to basic quantum error correction techniques along with the stabilizer formalism in Section 2. In Section 3 we motivate the use of topological codes for quantum error correction before introducing the class of surface codes. Finally, in Section 4 we describe several state-of-the-art applications of machine learning for optimizing surface code implementations.

2 Quantum error correction

Quantum error correction is necessarily different from classical error correction due to the unique characteristics of qubits. Due to the no-cloning theorem, it is impossible to copy a qubit. Due to the nature of quantum measurement, it is impossible to observe a qubit's value without possibly destroying some of its encoded quantum information, making it impossible to simply measure all qubits and take the majority result. Finally, due to the 2^n-dimensional nature of n-dimensional quantum systems, errors can take the form of not only a 'bit flip' as in classical bitstrings, but any rotation in a 2^n-dimensional complex Hilbert space [18], meaning there is an infinite number of possible errors. Therefore, it is impossible to simply apply typical classical error correction techniques to quantum states.

However, all hope is not lost. Shor [14] made a major breakthrough in quantum error correction when they described how any arbitrary quantum error could be measured without destroying the original quantum state. Extra qubits, called syndrome qubits, can be added to the system, and then an operation can be applied such that these extra qubits represent the nature of the error that perturbed the original quantum state. When the syndrome qubit is measured, the continuous quantum error is projected onto a discrete basis. The syndrome measurement can then be used to apply a unitary operation to the original qubit that reverses the error, restoring the system to its error-free state.

Indeed, it can be shown that the ability to correct some set of quantum errors implies the ability to correct any linear combination of these errors [8]. Hence, it suffices to correct some basis of the error space. Typically, we consider the Pauli basis $\{I, X, Y, Z\}$. Since $Y = iXZ$ is simply a combination of an X error and a Z error, we do not specifically address the Y error when designing a quantum error correcting scheme. Instead, we aim to correct just the Pauli X (bit flip) and Pauli Z (phase flip) errors:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad\qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Shor's 9-qubit code protects against both of errors on a single qubit, and therefore any type of single qubit error. While Shor's 9-qubit code is an important quantum error correction code, it has weaknesses which make it unsuitable for real world general purpose quantum computation. Its *rate*—ratio of logical to physical qubits—is fairly low, meaning that an enormous amount of physical qubits are required for even a modest number of logical qubits. Since current quantum computers have relatively few qubits, this is a major problem.

Effective codes generally have codewords that are "as different as possible". This idea is formalized with the *distance* of the code, which is the minimum *weight* of any operator that can transform some codeword into a different codeword. The *weight* of an operator is the number of qubits on which it acts nontrivially. For instance, the operator $Z \otimes I^{\otimes 7} \otimes Z$ has weight 2, since it applies the nontrivial Z operator at two locations. The 9-qubit code has distance 3 since, for example, the operator $Z_1 Z_4 Z_7$ (that is, the Z operator applied to the first, fourth, and seventh qubits) can transform the logical $|\tilde{0}\rangle$ state into the logical $|\tilde{1}\rangle$ state. Shor's 9-qubit code is just one example of a larger class of quantum error correction codes called *stabilizer codes*.

2.1 The stabilizer formalism

A *stabilizer code* is a type of quantum error correction code where encoding and decoding circuits can be implemented entirely using the Pauli gates I, X, Y, and Z. Most notable quantum error correction codes are stabilizer codes. These codes have the useful property that every code state is 'stabilized' by the code's 'stabilizer group'. An operator U is said to *stabilize* a quantum state $|\psi\rangle$ if $U|\psi\rangle = |\psi\rangle$. For example, I stabilizes all quantum states, while X stabilizes $|+\rangle$:

$$X|+\rangle = \frac{X|0\rangle + X|1\rangle}{\sqrt{2}} = \frac{|1\rangle + |0\rangle}{\sqrt{2}} = |+\rangle$$

A stabilizer code on n physical qubits is specified by its *stabilizer group*, which is a subgroup $\mathcal{S}$ of the 'Pauli group' $\mathcal{P}_n = \{I, X, Y, Z\}^{\otimes n}$, i.e. the group of all tensor products of Pauli matrices [18]. Here we use a shorthand $ABC...$ for the tensor product $A \otimes B \otimes C \otimes ...$ of Pauli matrices. Such

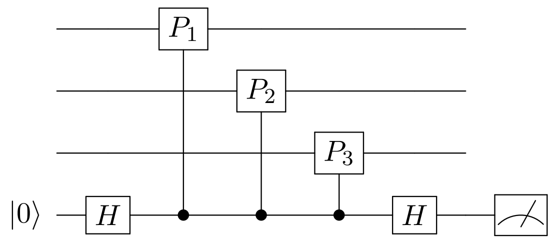

Figure 1: An example circuit to measure three stabilizers P_1, P_2, P_3 without destroying the encoded quantum state. The last qubit is the syndrome qubit. [4]

a code can be compactly described by the 'generators' of $\mathcal{S}$, i.e. the set of linearly independent members of $\mathcal{P}_n$ where each element of $\mathcal{S}$ can be expressed as a product of these elements.

For example, consider the simple three-qubit repetition code, where a qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ is encoded as $\alpha|000\rangle + \beta|111\rangle$. The stabilizer group of this code is $\mathcal{S} = \{ZZI, ZIZ, IZZ, III\}$ [5]. It can easily be verified that a generator set of this group is $\{ZZI, ZIZ\}$ because the other two Pauli matrices can be obtained as a product of these. It can also easily be verified that any member of the stabilizer group stabilizes any codeword; for example:

$$(ZZI)(\alpha|000\rangle + \beta|111\rangle) = \alpha|000\rangle + \beta(-|1\rangle \otimes -|1\rangle \otimes |1\rangle = \alpha|000\rangle + \beta|111\rangle$$

The size of the stabilizer group is 2^n, where n is the number of physical qubits. If the number of logical qubits represented by the code is k and the number of stabilizer generators is m, then $m = n - k$ [4].

The principle behind stabilizer codes is that 'correct' codewords will be stabilized by the generators while non-codewords (produced by an error) will not be. Of course, an error which maps from one codeword to another cannot be detected [18], but there exist many stabilizer codes which can always correct the commonly used bit flip and phase shift errors. For example, Shor's code can be represented as a stabilizer code (Figure 2). It has been proven that the smallest stabilizer code that can correct any single error is a five qubit code [10].

Error detection on a stabilizer code can be done by measuring the stabilizer generators using a circuit like the one in Figure 1. The syndrome measurements can be used to apply error correction operators to correct the qubit. Notice that there is not a one-to-one relationship between potential errors and stabilizer measurements. Given any error E and a stabilizer S, the error SE will be indistinguishable from E under the stabilizer code [4]. Thus, an algorithm is necessary to decide which error correction operator to apply for any given stabilizer measurement. In fact, the general case of decoding a stabilizer code is an NP-hard problem [12]. These algorithms are called 'decoders' and we will examine them more closely in later sections.

3 Topological codes

Stabilizer codes have been extended into more complex quantum error correction codes, like topological codes, which encode qubits in topology. Recall that the repetition code can be used to protect a qubit from bit-flip errors. We have already seen that the repetition code is unsuitable to correct general single-qubit errors, and in particular it is incapable of handling phase errors. For the sake of demonstration, we present one additional scenario in which the repetition code fails: consider the state $|\psi\rangle = \frac{1}{\sqrt{2}}(|\tilde{0}\rangle + |\tilde{1}\rangle) = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$ protected by the three-qubit repetition code.

$$\begin{Bmatrix} Z & Z & I & I & I & I & I & I & I \\ Z & I & Z & I & I & I & I & I & I \\ I & I & I & Z & Z & I & I & I & I \\ I & I & I & Z & I & Z & I & I & I \\ I & I & I & I & I & I & Z & Z & I \\ I & I & I & I & I & I & Z & I & Z \\ X & X & X & X & X & X & I & I & I \\ X & X & X & I & I & I & X & X & X \end{Bmatrix}$$

Figure 2: The generator set for Shor's nine-qubit code in the stabilizer formalism [10]

Clearly, a projective measurement in the computational basis of any single physical qubit causes $|\psi\rangle$ to collapse into either $|\tilde{0}\rangle = |000\rangle$ or $|\tilde{1}\rangle = |111\rangle$. That is, the superposition can be destroyed by a single errant measurement (where a "measurement" could consist of a simple interaction with the environment).

In Section 2.1 we observed that Shor's nine-qubit code can be used to detect and correct an arbitrary single-qubit error. We also discovered that the distance of Shor's code is 3, meaning there exists an error operator of weight 3 that transforms a code word into a different code word. We can extend the idea of Shor's code to obtain a family of codes:

$$|\tilde{0}\rangle = \frac{1}{\sqrt{2^k}}\left(|0\rangle^{\otimes k} + |1\rangle^{\otimes k}\right)^{\otimes k}$$
$$|\tilde{1}\rangle = \frac{1}{\sqrt{2^k}}\left(|0\rangle^{\otimes k} - |1\rangle^{\otimes k}\right)^{\otimes k}$$

Naturally, the distance of such a code is k. The code uses k^2 physical qubits to represent a single logical qubit, and hence its stabilizer has $k^2 - 1$ generators. Notice that as k increases, the weight of the stabilizer generators associated with phase parity checking also increases. For example, the operator that determines the phase parity of the first two "bundles" of k qubits is the following:

$$M = X_1 \otimes \ldots \otimes X_{2k} \otimes I^{\otimes k^2 - 2k}$$

The weight of the operator M is $2k$. Therefore, the number of qubits involved in a stabilizer measurement increases as we increase the number of physical qubits in the code. This becomes an issue if our error model allows individual qubit measurements to return incorrect values with small probability; in this scenario, the probability that at least one measurement in a stabilizer generator returns an incorrect value increases with the weight of the stabilizer. Owing to the large number of qubits covered by the operator M (along with the other stabilizer generators that detect phase errors), we say that the stabilizer measurements of this family of codes are *non-local*. However, since we aim to address errors that occur locally (i.e. independently and on individual physical qubits), we would prefer to develop a local code in which the weight of every stabilizer is bounded above by a constant (relative to the code distance).

Our goal is to investigate a quantum error correcting code with the following constraints (as suggested by Wootton [17] and Bombin [2]):

(1) The logical qubit depends on *global* properties of the code. That is, the logical qubit can not be determined by the measurement of a single physical qubit,

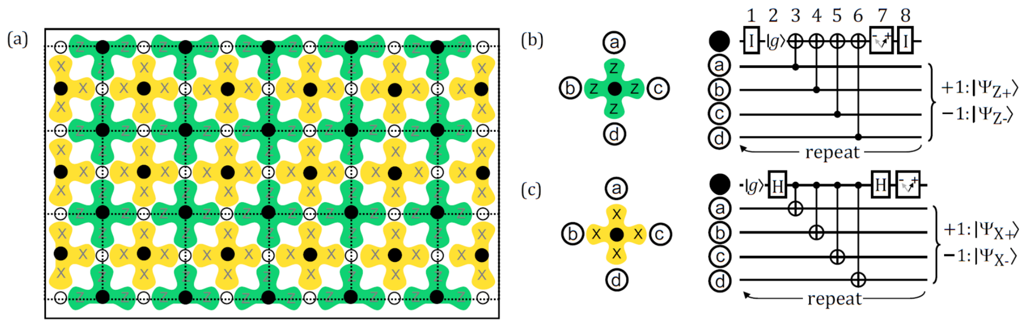

Figure 3: Implementation of a surface code [7].

(2) the distance of the code increases with the number of physical qubits, and

(3) the weight of a stabilizer operator is bounded above by a constant (as the number of physical qubits grows).

The use of topological codes is immediately suggested by (1). Since topological properties are inherently global, we can encode logical states in these properties. By embedding the system of qubits onto a surface, we will show how (2) and (3) can be satisfied as well. In the following section we introduce the surface code, an instructive example of a topological quantum error correcting code.

3.1 Surface codes

Originally defined by Kitaev [11], the toric code is a two-dimensional lattice of physical qubits which represents a pair of logical qubits. On an $L \times L$ square lattice, there are $2L^2$ edges and $2L^2$ physical qubits, with one physical qubit on each edge. Since the links of the square lattice are drawn on a torus (equivalently, on a square with opposite edges identified), the codes are referred to as toric codes. It was later discovered that the toroidal geometry was unnecessary and planar versions (surface codes) were developed by Bravyi and Kitaev [3]. In this paper, we limit our discussion to toroidal geometry. However, the ideas discussed canbe extended to any class of surface codes. Surface codes are a subclass of stabilizer codes where information is encoded in homological degrees of freedom [2]. For surface codes, nontrivial logical operators are tensor products of Pauli operators along non-contractible paths of the torus (i.e. topologically nontrivial cycles).

Figure 3 (a) depicts an implementation of a surface code. The data qubits are white circles ($\circ$) and black circles ($\bullet$) represent the syndrome qubits (also referred to as measure qubits). The storing and manipulating of quantum information is done by data qubits and the syndrome qubits measure the stabilizer operators for the system. Figure 3 (b) and (c) represent the sequence of operations (geometric and quantum circuit) for measure-Z qubits and measure-X qubits, respectively. The stabilizer generators of the surface code are the set of Z operators that surround each *plaquette* (or cell) of the lattice, along with the set of X operators that surround each *vertex* of the lattice. These are represented by the green and yellow crosses in Figure 3, respectively. The circuits in Figure 3 (b) and (c) are the equivalent circuits for the surface code illustrated in Figure 3 (a). Running the circuit is equivalent to measuring the syndromes corresponding to that plaquette or vertex.
The codespace $\mathcal{C}$ is defined by the $+1$ eigenspace of the stabilizer generators. To determine if the system is in the codespace, the stabilizers are measured repeatedly, forcing the physical qubits into some eigenstate of the stabilizer. Thus, any vector $|\psi\rangle \in \mathcal{C}$ if and only if $S|\psi\rangle = |\psi\rangle$ for $\forall S \in \mathcal{S}$,

where $\mathcal{S}$ is the stabilizer. If any of the measurements results in the outcome associated with the -1 eigenvalue of the stabilizer, then an error has occurred. Measuring each stabilizer generator allows us to isolate any affected data qubits, since an X error on a single data qubit will result in -1 outcomes on the pair of adjacent Z (i.e. plaquette) stabilizer generators. Similarly, a Z error on a single data qubit will result in -1 outcomes on the pair of adjacent X (i.e. vertex) stabilizer generators.

Beyond its ease-of-analysis, the surface code offers several practical advantages. First, it is highly degenerate (that is, many code words represent the same logical qubit), and correction only needs to be performed up to homology [2]. That is, when presented with a particular syndrome, a decoder does not need to determine the exact error that produced it. Instead, a decoder only needs to decide on the most-likely homology class to which the error string belongs (though this is also a difficult problem, as we will see in Section 3.2). Second, the stabilizer generators of the surface code are all local. That is, each plaquette/vertex stabilizer acts on a constant number of spatially close qubits, regardless of the number of physical qubits in the system. However, there is a tradeoff for these advantages: a logical qubit with a reasonably high error threshold (about 1%) needs on the order of 10^3 to 10^4 physical qubits to represent it [7].

3.2 Decoding topological codes

The decoding problem can be roughly stated as follows: given a syndrome of a particular state $E|\psi\rangle$ indicating that some error E has occurred, decide on an operator E' such that $E'E|\psi\rangle = |\psi\rangle$ (i.e. the operator $E'E$ is a member of the stabilizer). As described earlier, the syndrome of the surface code consists of the set of stabilizer measurements that return the outcome associated with their -1 eigenspace. We will henceforth refer to such measurements as *incorrect* stabilizer measurements. Any single physical qubit error in a surface code results in a pair of incorrect stabilizer measurements. More generally, any *string* of physical qubit errors in the underlying lattice results in a pair of incorrect stabilizer measurements at the endpoints of the string. We also note that the product of any set of stabilizer generators results in a boundary of Pauli operators. That is, any trivial cycle of Pauli operators on the surface code is a member of the stabilizer. Therefore, given the endpoints of some error string E, the decoder aims to find an error string E' that connects the endpoints of E, resulting in a cycle. Notice that there are many possible choices for E'. Some of these choices will result in a nontrivial cycle, the application of which amounts to a logical X or Z operation. In this circumstance, the decoder fails to correct the error E. Therefore, in order to avoid accidentally introducing a logical X or Z operation, the decoder must decide which error string E is most likely given the endpoints of E. The differences in error strings are only significant up to homology; see Figure 4 for further explanation.

Given the endpoints of some error string, the *optimal decoder* determines the probability of every equivalence class of the first homology group, and selects a correction string from the most likely class. Precisely calculating the probability of each equivalence class is extremely inefficient [4], so we cannot hope to use the optimal decoder in practice. Instead, the optimal decoder acts as a theoretical upper bound on the performance of other decoding solutions. The probability of each equivalence class depends on the chosen noise model. In the *independent noise model*, in which each physical qubit suffers an X error with probability p and a Z error (independently) with probability p. This model is popular in practice [4], since it allows one to perform analysis on X and Z errors separately. The probability of a particular X (or Z) error string of weight m in the independent noise model is $p^m(1-p)^{n-m}$, implying that short error strings are generally more likely. Using this idea, the decoding problem admits an efficient and deterministic approximation using the *minimum weight perfect matching* (MWPM) algorithm [6]. The *depolarizing noise model*

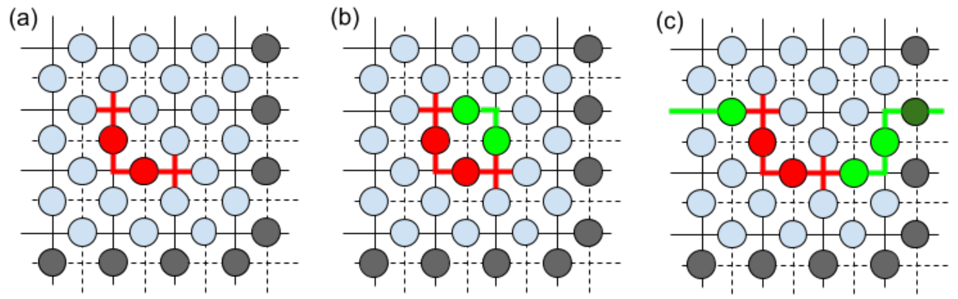

Figure 4: Suppose that the error string displayed in (a) occurs. That is, a Z error is applied to the two qubits highlighted in red. An incorrect stabilizer measurement is observed at the endpoints of this string, represented by the two red crosses in the image. If the decoder chooses to apply the error correction operator in (b) (i.e. the decoder applies Z operators to each of the qubits highlighted in green), then the resulting product of Z operators is a member of the stabilizer. Indeed, if the decoder chooses *any* string of correction operators that is homologically equivalent to the string depicted in (b), then the error is corrected. However, if the string of corrections in (c) is applied, then the resulting string of error operators comprises a nontrivial cycle on the surface. That is, the resulting product of Z operators is not a member of the stabilizer, and the stored logical state is corrupted.

introduces correlations between X and Z errors, and tends to be more difficult to analyze.

4 Machine learning for quantum error correction

While we have described the fundamentals of surface codes, we have left many important questions unanswered that would prevent us from developing an efficient and effective surface code implementation. For instance, how many physical qubits should be used in practice? What can we gain by using different lattice cellulations? How can we use objects with more homological degrees of freedom to our advantage? None of these questions has a simple answer, which is a consequence of the following unfortunate (but unsurprising) fact: the optimal set of parameters for the surface code heavily depends on the error rate and model. Each of the above questions can be difficult to answer analytically, so it seems natural to leverage machine learning to optimize surface codes for various environments. We will focus on two promising applications of machine learning for the surface code. First, we will examine decoding strategies that are facilitated by machine learning. Second, we will examine how reinforcement learning techniques can be used to optimize the structure of a surface code.

4.1 Learning decoding agents

As discussed in Section 3.2, decoding is not trivial. While MWPM provides a baseline, it is not an effective decoding strategy for correlated noise models. Finding efficient decoding algorithms for more complex error models is an active area of study. Machine learning can discover structure without explicit instruction, and has already produced some promising results in this field.

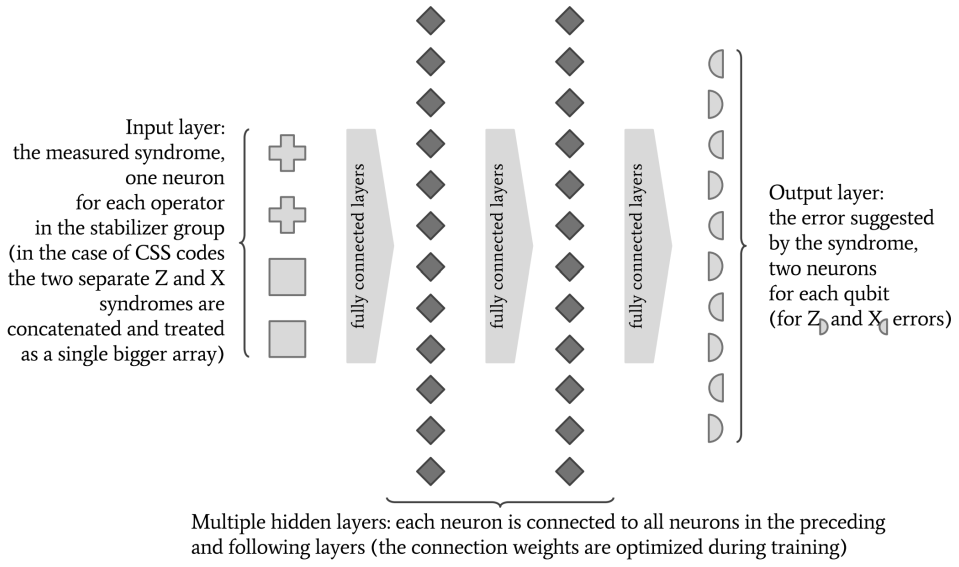

Figure 5: Architecture of the neural network used for syndrome classification (reproduced from Krastanov et al. [12])

4.1.1 Neural networks

Decoding a stabilizer code can be trivially reformulated as a classification task. Syndrome measurements are the input, which must be classified as errors. An ideal decoder will classify every syndrome measurement as the physical error that produced it. Krastanov et al. [12] use a fully-connected neural network to solve this classification task. Their network has a simple structure, with the input layer being the measured syndrome for each stabilizer operator, a small number of hidden layers, and an output layer representing the likelihoods of each type of error (see Figure 5).

The network is trained on pairs of errors and syndromes obtained by randomly generating errors and obtaining the corresponding syndrome using a quantum depolarization model, with the size and number of hidden layers determined by hyperparameter search. The trained neural network is used to decode by evaluating it forwards given a syndrome measurement as an input. The result is a vector of real numbers, which can be viewed as a probability distribution over possible errors.

Krastanov et al. test their approach on the toric code, because it has structure the neural network can take advantage of. They find that this algorithm significantly outperforms MWPM, suggesting that there exists hidden structure in the probability distribution over errors that a neural network is able to take advantage of. While Krastanov et al.'s approach is simple, it requires few explicit assumptions about the quantum computer it operates on, as the machine learning algorithm infers any structure during training—making it applicable to any stabilizer code.

4.1.2 Reinforcement learning

The decoding problem also admits a simple reformulation as a reinforcement learning problem [1, 13, 16]. We can imagine an agent whose goal is to preserve some logical quantum state. The agent can achieve this by applying correction operators to the physical qubits. Naturally, the agent succeeds as long as the initial logical quantum state is preserved by the physical corrections. The agent fails when it mistakenly applies a set of corrections that result in some erroneous logical operation. This simple framework offers significant flexibility (i.e. reward schemes, exploration, and actions that can be taken by the agent are all independent of the framework), and in this section we will provide a brief overview of two recent projects in this domain.

Sweke et al. [16] describe a reinforcement learning scheme that is agnostic to the stabilizer code used, but they use a surface code as a concrete example. Formally, the authors translate the problem of achieving fault-tolerant quantum computation using a stabilizer code into a *finite Markov decision process* [15], which defines a sequence of interactions between an *agent* and its *environment.* At each discrete time step in such a process, the agent is presented with a *state* (from some finite set of states) along with a finite set of *actions* that may be performed in that state. The agent chooses one of these actions, which results in a *reward.* Afterwards, the environment enters a new state and the agent is presented with its next available set of actions, marking the beginning of a new time step. Given an agent that follows some policy π, the *action-value function* (or q-function) for π maps state/action pairs (s, a) to the expected value generated by policy π given that action a is selected in state s. That is, if $q_\pi(s, a)$ is the action-value function for some policy π, then

$$q_\pi(s, a) = \mathbb{E}_\pi\big(G_t \;:\; S_t = s, A_t = a\big),$$

where S_t represents the state of the environment at time t, A_t represents the action chosen by the agent at time t, and G_t is some *discounted cumulative reward function.* That is, G_t represents the cumulative reward obtained by policy π from time t, where the reward obtained at step $i > t$ is scaled by $\gamma^{(i-t)}$, for some $\gamma \in [0, 1]$. Effectively, the parameter γ encourages the agent to prioritize immediate returns when it is close to 0, and allows the agent to prioritize future rewards when it is close to 1. While the definition of the action-value function is quite natural, finding the optimal action-value function can be challenging in practice, especially if the state and actions spaces are large. To overcome this barrier, Sweke et al. [16] use a class of neural networks to estimate the optimal action-value function.

Using this basic framework for reinforcement learning, Sweke et al. [16] define a class of *decoding agents*, which can learn effective decoding schemes for the surface code. The action space for the decoding agents consists of the set of all Pauli X and Z operators on each physical qubit in the code, while the state space consists of every possible set of stabilizer measurements (i.e. every possible syndrome). The environment consists of an initial logical state, a *hidden state* that contains a list of errors that have occurred, and a *referee decoder.* Training of the decoding agent progresses in a sequence of *episodes.* An episode begins by resetting the environment (i.e. a new initial state logical state and hidden state is generated). The agent is then provided with the syndrome of the initial hidden state. Note that each stabilizer measurement also has a small possibility of returning the incorrect value, so this may not be the true syndrome of the current hidden state. Given this information, the agent decides to either

(a) apply a Pauli correction operator to a single physical qubit, or

(b) request a new syndrome measurement from the environment.

In case (a) the chosen Pauli operator is applied to the hidden state. If the resulting state is equal to the initial logical state, then the agent is given a reward of 1 scaled by the discount factor. Otherwise, the agent is given a reward of 0. Afterwards, the referee decoder is given the true syndrome of the hidden state; if the referee is incapable of retrieving the initial logical state from the hidden state, then the logical state has been lost and the episode ends. In case (b) a new set of errors is applied to the hidden state according to some error model, and then the agent is provided with a (possibly faulty) syndrome of the new hidden state. Since an episode ends only when the stored logical state is lost, an effective decoding agent will induce relatively long episodes. Sweke

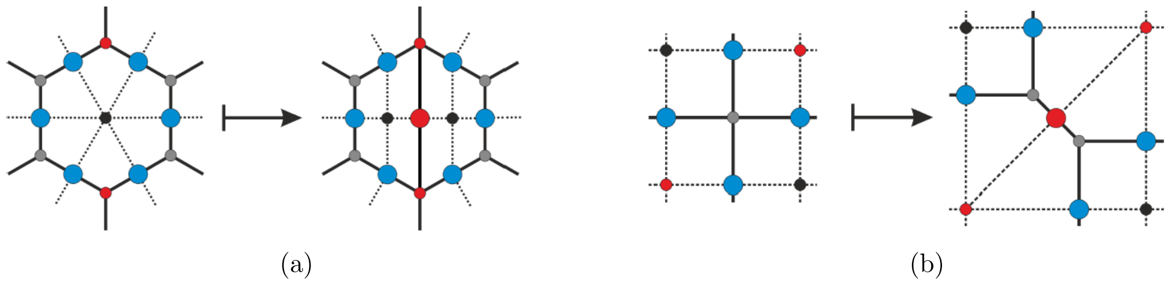

Figure 6: Illustrations of lattice adaptations. Blue dots represent data qubits, black and grey dots represent syndrome qubits. X and Z-type stabilizers are represented by lines connecting blue dots with plaquettes or vertices respectively. (a) The chosen vertices (red dots) are connected across a plaquette, which is split into two. (b) Red dots correspond to two plaquettes which are connected across a vertex. The vertex is split into two by a new edge . In both cases the number of data qubits (blue dots) is increased by one [13].

et al. [16] used their training framework to generate decoding agents that significantly extend the lifetime of logical 1-qubit states (relative to an unencoded single qubit) using the surface code. Their experiments considered not only the independent noise model, but also the depolarizing noise model. In the independent noise model (resp. depolarizing noise model), the generated decoding agents were capable of extending the lifetime of a single qubit as long as the error rate did not exceed approximately 0.13% (resp. 0.11%). Interestingly, these promising results were obtained using a surface code with a modest distance of $d = 5$.

4.2 Adaptable surface codes

Efficiently correcting arbitrary noise is a complex optimization problem. The difficulty of this task can be attributed by the diversity of environmental noise: the noise may not be independent and identically distributed, may be highly correlated, or may be completely unknown in realistic settings. As we saw in Section 4.1, efficient decoding is central to any fault-tolerant [16] quantum computing system. However, since decoders are dependent on the underlying code structure, only a limited improvement is possible by optimizing the decoding procedure. In this section, we explore the benefits of changing the structure of the quantum memory, both before and during computation. This is achieved through reinforcement learning and adaptive surface codes.

In Section 3.1, we discussed surface codes defined on a square lattice. However, this is not a requirement; surface codes can be defined on arbitrary lattices, which can change the performance of the code. This flexibility allows us to fine-tune our surface codes for biased noise models. We refer to these codes as *adaptable* surface codes [13].

The basic moves depicted in Figure 6 can be implemented fault-tolerantly and map a surface code to another while changing the underlying stabilizer group [13]. These adaptations or code deformations let us edit the error correction mechanism on the fly and thus enable us to more effectively combat arbitrary noise. If the optimization procedure is performed on a real quantum computer, then we assume that the device has the capacity to perform the basic code deformations depicted in Figure 6 efficiently. Nautrup et al. claim that these deformations are designed with the constraints of a physical device in mind [13].

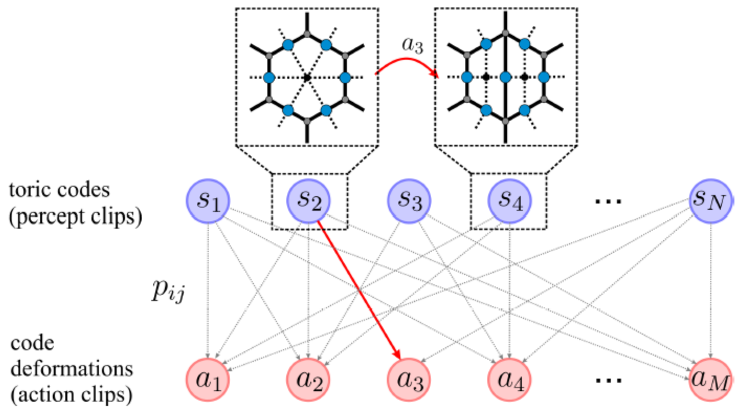

Figure 7: Illustration of a two-layered clip network of a PS agent. Surface codes are represented as percept clips in the upper layer. Code deformations are represented as actions in lower layer. Each edge has an associated weight according to its transition probability. In the illustration, given a percept s_2 the agent decides to take action a_3. This results in a new surface code s_4 which is added to the set of percept clips [13].

4.2.1 Optimizing adaptable surface codes

In the scheme proposed by Nautrup et al.[13], a Projective Simulation (PS) reinforcement learning agent interacts with the environment and modifies the given quantum error correcting code based on feedback. First, the agent is trained in a *simulated* environment under various scenarios: i.i.d. error channels, correlated error channels, and changing error threshold requirements. Interestingly, it has been shown that the resulting agents have the ability to transfer their learning experience to different scenarios (*transfer learning*). Thus, the agent does not need to be trained from scratch when its environment or threshold requirement changes. Each trial is initialized with a distance 3 surface code Λ consisting of 18 qubits embedded on a square lattice. The code is then subjected to an unknown Pauli noise channel which may change over time and can differ across qubits.

The core component of a PS agent is its clip network which comprises of episodic memory called *clips*. The agent receives information about the environment through activation of *percept* clip, $s_i \in P$, which is information about the current state of the code. The agent can perform certain actions as action clips, $a_i \in A$ and add up to 50 additional qubits to reduce the logical error rate below a threshold.

Figure 7 presents an illustrative two-layer clip network. In this network each percept, $s_i \in P, i \in N^{(t)}$ (where $N^{(t)}$ is the number of percepts available at time t) is connected to a action clip, $s_1 \in A, j \in [1, M_i]$ (M_i is the set of actions available for a s_i) via an edge which represents the transition probability of taking an action a_i based on percept s_i, $p_{ij} := p(a_i|s_i)$. As the the agent learns, the clip network is adapted by creating new clips and updating transition probabilities. When a new percept is encountered by performing a particular action, it is added to P. The agent's goal is to use the basic code deformations depicted in Figure 6 to achieve a given target logical error threshold. If the agent is capable of reaching this threshold with no more than 50 additional qubits, it is given a reward of 1. Otherwise, the agent is given a reward of 0. After the agent is rewarded, the trial ends and the surface code is reset to Λ. As before, the agent's objective is to maximize its expected reward.

The fact that agents can transfer their learning experiences between environments allows us to bootstrap agents on (relatively) cheap classical simulation hardware, before using these trained agents on real quantum computers. In a new environment, by employing exploration and exploitation the agent can figure out a strategy to adapt a sub-optimal code to achieve the desired error rate. Additionally, the method makes no assumptions about the environment and the agents receive no information about the noise model. Offline classical simulations of surface code optimization are much faster and cheaper than running such an optimization procedure on a noisy small-scale quantum computer. Hence, transfer learning offers a significant performance advantage for near-term quantum computing devices.

5 Conclusion

Quantum error correcting codes are a fundamental step towards the realization of fault-tolerant quantum computing. It is convenient to express quantum error correcting codes in the stabilizer formalism, as stabilizer codes are relatively simple to analyze. A particularly promising subclass of stabilizer codes are the so-called surface codes, which are constructed by embedding physical qubits onto the edges of a lattice that is itself embedded onto some surface. Logical states are encoded in global topological properties of the surface. Surface codes are desirable because their stabilizers are local. That is, every stabilizer generator of a surface code acts on only a constant number of qubits, and each physical qubit is acted upon by only a constant number of generators. Decoding a surface code is nontrivial, since a syndrome only reveals the endpoints of error strings on the surface. Therefore, a decoder must decide which homology class the error belongs to, before applying a correction operator from this same class.

We investigated several applications of machine learning regarding the surface code. The decoding problem can be translated into a classification task and fed through a neural network. Krastanov et al. [12] demonstrated that neural networks are capable of effectively choosing error correction operators on the surface code, outperforming deterministic approaches in the depolarizing noise model. This approach is generalized by Sweke et al. [16], who describe a framework for generating decoding agents that extend the lifetime of logical qubit states. Finally, Nautrup et al. demonstrate a technique for optimizing the structure of the surface code itself, which could be used to improve the performance of the surface code in different error models. With the large number of parameters required to describe a particular surface code, the variety of machine learning techniques being applied to this class of codes is unsurprising. We observed how machine learning can be used to develop decoding schemes, and how it can be used to find optimal surface code structures; a natural next step would be to investigate how these approaches work in tandem.

References

[1] Philip Andreasson et al. "Quantum error correction for the toric code using deep reinforcement learning". In: *Quantum* 3 (Sept. 2019), p. 183. ISSN: 2521-327X. DOI: `10.22331/q-2019-09-02-183`. URL: `http://dx.doi.org/10.22331/q-2019-09-02-183`.

[2] H. Bombin. *An Introduction to Topological Quantum Codes.* 2013. arXiv: `1311.0277 [quant-ph]`.

[3] S. B. Bravyi and Alexei Y. Kitaev. "Quantum codes on a lattice with boundary". In: 1998.

[4] Dan Browne. *Lectures on Topological Codes and Quantum Computation.* June 2014. URL: `https://sites.google.com/site/danbrowneucl/teaching/lectures-on-topological-codes-and-quantum-computation`.

[5] Shubham Chandak, Jay Mardia, and Meltem Tolunay. *Implementation and analysis of stabilizer codes in pyQuil*. 2019. URL: `https://cs269q.stanford.edu/projects2019/stabilizer_code_report_Y.pdf`.

[6] Jack Edmonds. "Paths, Trees, and Flowers". In: *Canadian Journal of Mathematics* 17 (1965), pp. 449–467. DOI: `10.4153/CJM-1965-045-4`.

[7] Austin G. Fowler et al. "Surface codes: Towards practical large-scale quantum computation". In: *Physical Review A* 86.3 (Sept. 2012). ISSN: 1094-1622. DOI: `10.1103/physreva.86.032324`. URL: `http://dx.doi.org/10.1103/PhysRevA.86.032324`.

[8] Daniel Gottesman. "An introduction to quantum error correction". In: *Proceedings of Symposia in Applied Mathematics*. 2002.

[9] Daniel Gottesman. *Quantum Error Correction Sonnet*. 1999. URL: `https://www.perimeterinstitute.ca/personal/dgottesman/sonnet.html` (visited on 12/03/2019).

[10] Daniel Gottesman. *Stabilizer Codes and Quantum Error Correction*. 1997. arXiv: `quant-ph/9705052 [quant-ph]`.

[11] A Yu Kitaev. "Fault-tolerant quantum computation by anyons". In: *Annals of Physics* 303.1 (2003), pp. 2–30.

[12] Stefan Krastanov and Liang Jiang. "Deep Neural Network Probabilistic Decoder for Stabilizer Codes". In: *Scientific Reports* 7.1 (2017), p. 11003. ISSN: 2045-2322. DOI: `10.1038/s41598-017-11266-1`. URL: `https://doi.org/10.1038/s41598-017-11266-1`.

[13] Hendrik Poulsen Nautrup et al. *Optimizing Quantum Error Correction Codes with Reinforcement Learning*. 2018. arXiv: `1812.08451 [quant-ph]`.

[14] Peter W. Shor. "Scheme for reducing decoherence in quantum computer memory". In: *Phys. Rev. A* 52 (4 Oct. 1995), R2493–R2496. DOI: `10.1103/PhysRevA.52.R2493`. URL: `https://link.aps.org/doi/10.1103/PhysRevA.52.R2493`.

[15] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2018.

[16] Ryan Sweke et al. *Reinforcement Learning Decoders for Fault-Tolerant Quantum Computation*. 2018. arXiv: `1810.07207 [quant-ph]`.

[17] J. Wootton. *Decodoku: Gaming for science!* 2016. URL: `http://decodoku.blogspot.com/2016/03/6-toric-code.html`.

[18] Yixuan Xie. *Quantum Error Correction and Stabilizer Codes*. 2016. URL: `http://unsworks.unsw.edu.au/fapi/datastream/unsworks:39470/SOURCE02`.