Discovering Quantum Error Correcting Codes with Deep Reinforcement Learning

Jan Olle
Max Planck Institute for the Science of Light
Staudtstraße 2, 91058 Erlangen, Germany
jan.olle@mpl.mpg.de

Remmy Zen
Max Planck Institute for the Science of Light
Staudtstraße 2, 91058 Erlangen, Germany
remmy.zen@mpl.mpg.de

Matteo Puviani
Max Planck Institute for the Science of Light
Staudtstraße 2, 91058 Erlangen, Germany
matteo.puviani@mpl.mpg.de

Florian Marquardt
Max Planck Institute for the Science of Light
Staudtstraße 2, 91058 Erlangen, Germany
florian.marquardt@mpl.mpg.de

Abstract

Reinforcement Learning (RL) stands out in finding optimal strategies for complex tasks without prior knowledge of the dynamics of the system. In this work, we use RL for automatically discovering Quantum Error Correction (QEC) codes and their encoding circuits for a given gate set and error model. Our approach is very general and our implementation is extremely efficient. In particular, we find all possible stabilizer codes and their encoding circuits that correct single errors for up to 15 qubits, for an illustrative gate set. All in all, our framework is a versatile tool for QEC across diverse quantum hardware platforms of interest.

1 Introduction

Fault-tolerant quantum computers promise to solve computational problems that are inaccessible to the best known classical algorithms. However, quantum information is fragile and protecting it from the effects of noise remains an outstanding problem. The leading approach to preserve quantum information is *quantum error correction* (QEC) [1]. The idea is to redundantly embed the quantum information within a subspace (called *code* space) of a *larger* Hilbert space in such a way that different errors map the code to mutually orthogonal subspaces. If successful, the action of each of these errors can be reverted and the computation can continue error-free.

In this work, we propose a new Machine Learning (ML) tool to discover arbitrary stabilizer QEC codes from scratch using Reinforcement Learning (RL) [2]. Our approach allows us not only to discover the code itself, but also an optimal sequence of gates that prepare such code, for an error model and gate set of choice.

Related work: Previous computational methods for QEC code construction were either restricted to finding a subclass of codes (but not the encoding circuit) associated with graphs [3], or were based on expensive numerical greedy search for finding stabilizer codes [4]. ML-based methods have also been developed, e.g. [5–8]. While [7] also set themselves the task of finding both codes and their encoding circuits, this was done using variational quantum circuits involving continuously parametrized gates, which leads to much more costly numerical simulations and eventually only an approximate QEC scheme. By contrast, our RL-based approach does not rely on any human-provided circuit ansatz, can use directly any given discrete gate set and is able to exploit highly efficient Clifford simulations. In particular, we are able to discover codes and encoding circuits for both larger number of qubits (14 vs 15) and larger code distances (4 vs 5).

Machine Learning and the Physical Sciences Workshop, NeurIPS 2023.

2 Stabilizer codes

Some of the most promising QEC codes are based on the stabilizer formalism of quantum mechanics [9], which leverages the properties of the Pauli group G_n on n qubits. The basic idea of the stabilizer formalism is that many quantum states can be more compactly described by listing the set of n operators that *stabilize*[1] them instead of listing their 2^n coefficients. The Pauli group of a single qubit G_1 is defined as the set of Pauli matrices X, Y, Z with the overall phases ± 1, $\pm i$, which form a group under matrix multiplication. The generalization to n qubits consists of all n-fold tensor products of Pauli matrices (called *Pauli strings* and represented as binary arrays of size $2n$), and where global phases can be ignored for the purposes of QEC.

A code that encodes k logical qubits into n physical qubits is a 2^k-dimensional subspace (the *code space* $\mathcal{C}$) of the full 2^n-dimensional Hilbert space. It is completely specified by the set of Pauli strings $S_\mathcal{C}$ that *stabilize* it, i.e. $S_\mathcal{C} = \{s_i \in G_n \mid s_i|\psi\rangle = |\psi\rangle,\ \forall |\psi\rangle \in \mathcal{C}\}$. $S_\mathcal{C}$ is called the stabilizer group of $\mathcal{C}$ and is usually written in terms of its group generators g_i as $S_\mathcal{C} = \langle g_1, g_2, \ldots, g_{n-k}\rangle$.

Noise affecting quantum processes can be represented using the so-called *operator-sum* representation [10], where a quantum noise channel $\mathcal{E}$ induces dynamics on the state ρ according to

$$\mathcal{E}(\rho) = \sum_\alpha E_\alpha \rho E_\alpha^\dagger \ , \tag{1}$$

where E_α are *Kraus operators*, satisfying $\sum_\alpha E_\alpha^\dagger E_\alpha = I$. The most elementary example is the so-called *depolarizing* noise channel,

$$\mathcal{E}_{\text{DP}}(\rho) = p_I \rho + p_X X\rho X + p_Y Y\rho Y + p_Z Z\rho Z \ , \tag{2}$$

where $p_I + p_X + p_Y + p_Z = 1$ and the set of Kraus operators are $E_\alpha = \{\sqrt{p_I}I, \sqrt{p_X}X, \sqrt{p_Y}Y, \sqrt{p_Z}Z\}$. Arbitrary errors on n qubits can be described using local depolarizing channels acting on each qubit j independently. A commonly used simplification is the following. Assume that all error probabilities are identical, i.e. $p_X = p_Y = p_Z \equiv p$ (and $p_I = 1 - 3p$). Then, the probability that a given error occurs decreases with the number of qubits it affects. This leads to the concept of the *weight* of a Pauli string as the number of qubits on which it differs from the identity and to a hierarchical approach to building QEC codes. In particular, stabilizer codes are described by specifying what is the minimal weight in the Pauli group that they cannot detect.

The fundamental theorem in QEC is a set of necessary and sufficient conditions discovered by Knill and Laflamme (KL conditions) in [11] that state that a code $\mathcal{C}$ with associated stabilizer group $S_\mathcal{C}$ can *detect* a set of errors $\{E_\mu\} \subseteq G_n$ if and only if

$$\{E_\mu, g_i\} = 0 \ , \tag{3}$$

for at least one g_i, or the error itself is harmless, i.e.

$$E_\mu \in S_\mathcal{C} \ . \tag{4}$$

The smallest weight in G_n for which any of the above two conditions do not hold is called the *distance* of the code. A quantum code that can correct up to weight-t errors must have a distance of at least $d = 2t + 1$. If all the errors that are detected with weight smaller than d are detected through (3), the code is *non-degenerate*. On the other hand, if some of the errors satisfy (4), the code is called *degenerate*. We follow standard notation and write quantum codes of distance d that encode k logical qubits into n physical qubits as $[\![n, k, d]\!]$. Finally, it is worth noting that codes with the same code parameters $[\![n, k, d]\!]$ may have different error-correcting capabilities. Therefore, we will refer to *families* of codes with the same code parameters using as classification criteria their quantum weight enumerators [12]. These are two polynomials whose coefficients count the number of error operators of weight j in $S_\mathcal{C}$ and the number of error operators of weight j that commute with all elements of $S_\mathcal{C}$, respectively.

3 Reinforcement learning approach to QEC code discovery

The main objective of this work is to automatize the process of QEC code discovery using RL. The goal in any RL task is encoded by choosing a suitable *reward* r, a quantity that measures how well

[1] An operator O stabilizes a state $|\psi\rangle$ if $O|\psi\rangle = |\psi\rangle$.

the task has been solved, and consists of an *agent* (the entity making the decisions) interacting with an *environment* (the physical system). In each time step t, the environment's state s_t is observed. Based on this observation, the agent takes an action a_t which then affects the current state of the environment. A *trajectory* is a sequence of state and action pairs that the agent takes. An *episode* is a trajectory from an initial state to a terminal state. For each action, the agent receives a reward r_t, and the goal of RL algorithms is to maximize the expected cumulative reward (return), $\mathbb{E}\left[\sum_t r_t\right]$. The agent's behavior is defined by the *policy* $\pi_\theta(a_t|s_t)$, which denotes the probability of choosing action a_t given observation s_t, and we parametrized it by a neural network with parameters θ.

Within RL, policy gradient methods [13] optimize the policy by maximizing the expected return with respect to the parameters θ with gradient ascent. One of the most successful algorithms within policy gradient methods is the actor-critic algorithm [14]. The idea is to have two neural networks: an actor network that acts as the agent and that defines the policy, and a critic network, which measures how good was the action taken by the agent. In this paper, we use a state-of-the-art policy-gradient actor-critic method called Proximal Policy Optimization (PPO) [15], which improves the efficiency and stability of policy gradient methods.

In order to encode the state of k logical qubits on n physical qubits one must find a sequence of quantum gates that will entangle the quantum information in such a way that QEC is possible with respect to a target noise channel. Initially, we imagine the first k qubits as the original containers of our (yet unencoded) quantum information, which can be in any state $|\psi\rangle \in (\mathbb{C}_2)^{\otimes k}$. The remaining $n-k$ qubits are chosen to each be initialized in the state $|0\rangle$. These will be turned into the corresponding logical state $|\psi\rangle_L \in (\mathbb{C}_2)^{\otimes n}$ via the application of a sequence of discrete Clifford[2] gates (actions) on any of the n qubits, updating its code generators (observation) until they satisfy the KL conditions (3), (4) for the chosen error channel. In Fig.1a we schematize our approach, while Fig.1c shows how the code generators change when gates are applied.

The most delicate matter in RL problems is building a suitable reward for the task at hand. An option is to use a scheme where the cumulative reward (which RL optimizes) simply is maximized whenever all the KL conditions are fulfilled. One implementation of this idea uses the weighted KL sum as an instantaneous reward:

$$r_t = -\sum_\mu \lambda_\mu K_\mu \,, \tag{5}$$

where $K_\mu = 0$ if either (3) or (4) are satisfied, and $K_\mu = 1$ otherwise. The index μ runs from 1 to $|\{E_\mu\}|$, see (6). If all errors in $\{E_\mu\}$ can be detected, the reward is zero, and is negative otherwise. Here λ_μ are real numbers that weigh how important each error is. These can in principle be taken to be $\lambda_\mu = p_\mu$, where p_μ is the probability that E_μ occurs, but we will view them as hyperparameters. A convenient feature of our reward is that one can favor certain types of codes. For instance, we can target non-degenerate codes only by ignoring (4), or we can favor degenerate codes by choosing larger λ_μ values of those errors that satisfy (4). Moreover, making the reward non-positive favors short gate sequences, which is desirable when wanting to prepare these codes in actual quantum devices.

Regarding the target error channel, here there are in principle several choices that can be made. The most straightforward one is choosing a symmetric depolarizing channel of distance d. This means that the set $\{E_\mu\}$ are all Pauli strings of weight up to $d-1$. There are

$$|\{E_\mu\}|_{w\leq d-1} = \sum_{w=0}^{d-1} 3^w \binom{n}{w} \,, \tag{6}$$

which grows exponentially with d. As we will see, this exponential growth of the number of error operators that have to be tracked will impose the most severe limitation in our approach.

4 Results

RL algorithms exploit fast trial-and-error loops until a signal of a good strategy is picked up and convergence is reached. It is thus of paramount importance that simulations of our RL environment

[2]Clifford gates are defined to be those that map Pauli strings to Pauli strings and are generated by the Hadamard H, the Phase S and the CNOT gates.

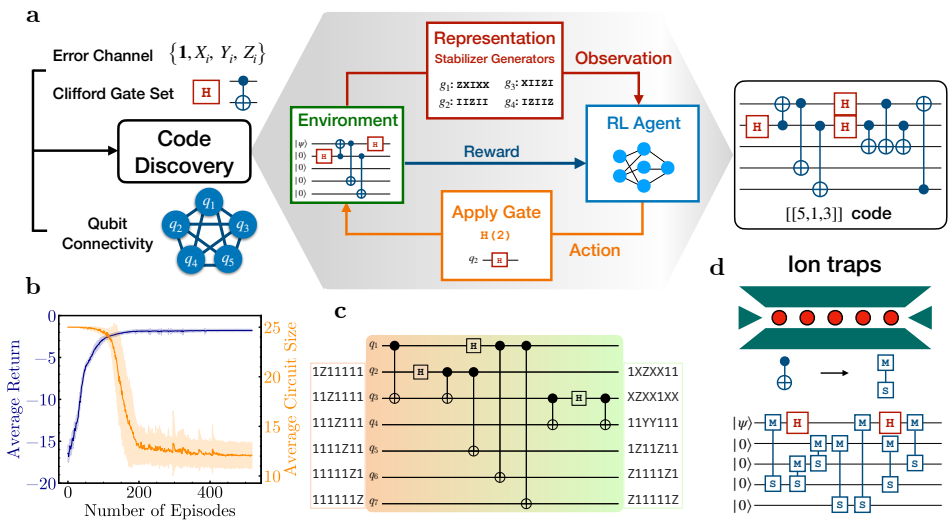

Figure 1: QEC code discovery using RL. **a.** Scheme of our approach. An error channel, a gate set and qubit connectivity is chosen. The agent then builds a circuit using the available gate set and connectivity that detects the most likely errors from the target error model by using a reward based on the Knill-Laflamme QEC conditions according to Eq. (5). After training, a single RL agent is able to find a suitable encoding for *any* state $|\psi\rangle$ of choice. **b.** Example of a training trajectory for $[\![7,1,3]\!]$ code discovery. Here, 16 parallel agents each interact with batches of 32 circuits processed in parallel. Each agent finds a different encoding circuit, and the training finishes in 16 sec on a single GPU. **c.** Example of how the generators change upon applying gates. Here the $[\![7,1,3]\!]$ bare code [4] is rediscovered. **d.** Example of code discovery in an ion-trap-like platform, where the native 2-qubit gate is the Mølmer–Sørensen gate.

are extremely fast. Thanks to the Gottesman-Knill theorem, the Clifford circuits needed here can be simulated efficiently classically [16, 17]. Optimized numerical implementations exist, e.g. STIM [18]. However, in an RL application we want to be able to run multiple agents in parallel in an efficient, vectorized way that is compatible with modern machine learning frameworks. For that reason, we have implemented our own special-purpose vectorized Clifford simulator. Briefly, we use the symplectic binary formalism of the Pauli group [17] to represent the stabilizer generators. $S_{\mathcal{C}}$ is then represented by a *check matrix* H [17], which is a $(n-k)\times 2n$ binary matrix where each row i represents the Pauli string g_i from $S_{\mathcal{C}}$. Clifford gates are also implemented using binary matrices. Second, we achieve a massive speedup of the training process by running simulations of quantum circuits in parallel, meaning that the agent interacts with a batch of RL environments at every timestep. Finally, we also train multiple RL agents in parallel on a single GPU. This is achieved by interfacing with PUREJAXRL [19], a library that offers a high-performance end-to-end JAX [20] RL implementation. Our code is written in PYTHON and is ran on a single NVIDIA Quadro RTX 6000 GPU.

In the following we fix the error model to be a symmetric depolarizing channel of weight $d-1$ and we vary d from 3 to 5. The gateset is $\{H_i, \text{CNOT}(i<j)\}$, but others can be considered, see Fig.1(d). To show the efficiency of our implementation, we first display an example of a training trajectory in Fig.1b. There, 16 agents are tasked to find $[\![7,1,3]\!]$ codes, which each of them completes successfully running in parallel in 16 sec on a single GPU. The average circuit size starts being 25 by design, i.e. if no code has been found after 25 gates, the circuit gets reinitialized. This number starts decreasing when codes start being found and it saturates to a final value, which is in general different for each agent. The noticeable variance in the circuit sizes that are found in Fig.1b is a testament that different codes (in the sense of possibly belonging to different code families) with the same code parameters $[\![7,1,3]\!]$ may have been found, yet making such a distinction requires further postprocessing.

The main results are summarized in Fig.2, where we show a classification of stabilizer codes with many different parameters $[\![n,k,d]\!]$. The circuit size shown for each $[\![n,k,d]\!]$ in Fig. 2 is the minimal one found across all discovered families. In general, different families have different circuit sizes,

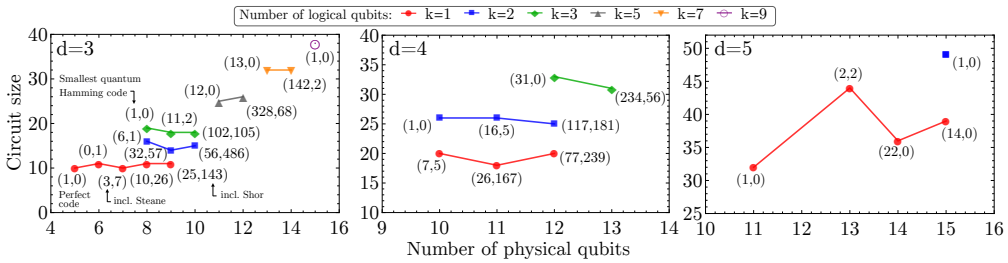

Figure 2: Discovering codes and encoding circuits for various numbers of physical qubits, logical qubits, and distances. Selection of families of stabilizer codes tailored to symmetric depolarizing noise channels, found with our RL framework. The labels (x, y) indicate the number of non-degenerate (x) and degenerate (y) code families. The circuit size shown is the absolute minimum throughout all families, and different families in general have different minimal circuit sizes. Since further $d = 3, 4$ training runs do not increase family populations, it is likely that there are no more stabilizer codes for the shown $[\![n, k, d = 3, 4]\!]$. Codes $[\![n, k = 4, 6, 8, d = 3]\!]$ are also found but not shown to keep the figure uncluttered. For the computationally harder case of $d = 5$ we display the results found in a finite allotted time.

and even within the same family we find variations in circuit sizes. This approach discovers suitable encoding circuits, given the assumed gate set, for a large set of codes. Among them are the following known codes for $d = 3$: The first one is the five-qubit perfect code [21], which consists of a *single* non-degenerate $[\![5, 1, 3]\!]$ code family and is the smallest stabilizer code that corrects an arbitrary single-qubit error. Next are the 10 families [22] of $[\![7, 1, 3]\!]$ codes (see Fig. 1c for an example), one of which corresponds to Steane's code [23]. The smallest single-error-correcting surface code, Shor's code [24], is rediscovered as one of the 143 degenerate code families with parameters $[\![9, 1, 3]\!]$. The smallest quantum Hamming code [25] $[\![8, 3, 3]\!]$ is obtained as well. Our approach is efficient enough to reach up to 15 physical qubits, discovering codes in less than 20 minutes. Extending to more physical qubits is certainly doable.

Distance-5 codes are more challenging to find due to the many more error operators (6) to be kept track of. Nevertheless, we find codes with the smallest known parameters $[\![11, 1, 5]\!]$, and as large as $[\![15, 2, 5]\!]$. For instance, each of these training runs needs 1-4 hours, depending on the code parameters and whether degenerate codes are also targeted. Nevertheless, future performance improvements are likely possible.

5 Outlook

We have presented an RL framework that is able to discover QEC codes and their encoding circuits from scratch, for a given gate set and error model. Our approach is very general, and we are able to scale up to $d = 5$ codes without much effort. Larger code distances are challenging due to the exponential number of errors to be kept track of, (6), which imposes a memory bottleneck. However, $d \leq 7$ fit in present-day GPUs. Moreover, agent reuse in a transfer-learning setup could alleviate this bottleneck.

As a comment, it is worth noting that our encoding circuits are not fault tolerant. We have assumed that gates are perfect, but gates are in general faulty and errors propagate through the circuit, often becoming incorrigible. However, flag fault tolerance [26] on top of our encoding circuits could turn these encodings fault tolerant, which is a task that is also suitable for an RL implementation. In any case, our methods could find direct application in quantum communication setups, where errors mostly occur while sending data, and not in the encoding part.

Ongoing extensions of this work include designing a more general noise-aware RL agent that is capable of finding codes in many different noise channels at once. This would be valuable already in near-term quantum devices, where phase flips are dominant with respect to bit flips [27]. Here, a *single* agent would be able to switch its encoding strategy for a range of noise models, thus leveraging transfer of insights between different situations.

Acknowledgments and Disclosure of Funding

Fruitful discussions with Sangkha Borah, Jonas Landgraf and Maximilian Naegele are thankfully acknowledged. This research is part of the Munich Quantum Valley, which is supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus. The authors declare no competing interests.

References

[1] Steven M. Girvin. Introduction to quantum error correction and fault tolerance. *SciPost Physics Lecture Notes*, 2023.

[2] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. MIT press, 2018.

[3] Isaac Chuang, Andrew Cross, Graeme Smith, John Smolin, and Bei Zeng. Codeword stabilized quantum codes: Algorithm and structure. *Journal of Mathematical Physics*, 2009.

[4] Muyuan Li, Mauricio Gutiérrez, Stanley E. David, Alonzo Hernandez, and Kenneth R. Brown. Fault tolerance with bare ancillary qubits for a [[7,1,3]] code. *Phys. Rev. A*, 2017.

[5] Thomas Fösel, Petru Tighineanu, Talitha Weiss, and Florian Marquardt. Reinforcement learning with neural networks for quantum feedback. *Phys. Rev. X*, 2018.

[6] Hendrik Poulsen Nautrup, Nicolas Delfosse, Vedran Dunjko, Hans J. Briegel, and Nicolai Friis. Optimizing quantum error correction codes with reinforcement learning. *Quantum*, 2019.

[7] Chenfeng Cao, Chao Zhang, Zipeng Wu, Markus Grassl, and Bei Zeng. Quantum variational learning for quantum error-correcting codes. *Quantum*, 2022.

[8] Vincent Paul Su, ChunJun Cao, Hong-Ye Hu, Yariv Yanay, Charles Tahan, and Brian Swingle. Discovery of optimal quantum error correcting codes via reinforcement learning, 2023.

[9] Daniel Gottesman. Stabilizer codes and quantum error correction, 1997.

[10] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010.

[11] Emanuel Knill and Raymond Laflamme. Theory of quantum error-correcting codes. *Phys. Rev. A*, 1997.

[12] Peter Shor and Raymond Laflamme. Quantum analog of the macwilliams identities for classical coding theory, Feb 1997.

[13] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[14] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[15] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[16] Daniel Gottesman. The heisenberg representation of quantum computers, 1998.

[17] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Phys. Rev. A*, 2004.

[18] Craig Gidney. Stim: a fast stabilizer circuit simulator. *Quantum*, 5:497, July 2021.

[19] Chris Lu, Jakub Kuba, Alistair Letcher, Luke Metz, Christian Schroeder de Witt, and Jakob Foerster. Discovered policy optimisation. *Advances in Neural Information Processing Systems*, pages 16455–16468, 2022.

[20] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, Adam Paszke George Necula, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[21] Raymond Laflamme, Cesar Miquel, Juan Pablo Paz, and Wojciech Hubert Zurek. Perfect quantum error correcting code. *Phys. Rev. Lett.*, 77:198–201, Jul 1996.

[22] Sixia Yu, Qing Chen, and C. H. Oh. Graphical quantum error-correcting codes, 2007.

[23] A. M. Steane. Simple quantum error-correcting codes. *Phys. Rev. A*, 54:4741–4751, Dec 1996.

[24] A. R. Calderbank and Peter W. Shor. Good quantum error-correcting codes exist. *Phys. Rev. A*, 54:1098–1105, Aug 1996.

[25] Daniel Gottesman. Class of quantum error-correcting codes saturating the quantum hamming bound. *Phys. Rev. A*, 54:1862–1868, Sep 1996.

[26] Rui Chao and Ben W. Reichardt. Quantum error correction with only two extra qubits. *Phys. Rev. Lett.*, 2018.

[27] Lev Ioffe and Marc Mézard. Asymmetric quantum error-correcting codes. *Phys. Rev. A*, 2007.