Deep Q-learning decoder for depolarizing noise on the toric code

Deep Q-learning decoder for depolarizing noise on the toric code

David Fitzek,[1,2,*] Mattias Eliasson,[3] Anton Frisk Kockum,[1] and Mats Granath[3,†]

[1]*Wallenberg Centre for Quantum Technology, Department of Microtechnology and Nanoscience, Chalmers University of Technology, SE-41296 Gothenburg, Sweden*

[2]*Volvo Group Trucks Technology, 405 08 Gothenburg, Sweden*

[3]*Department of Physics, University of Gothenburg, SE-41296 Gothenburg, Sweden*

(Received 30 December 2019; accepted 5 May 2020; published 26 May 2020)

We present an AI-based decoding agent for quantum error correction of depolarizing noise on the toric code. The agent is trained using deep reinforcement learning (DRL), where an artificial neural network encodes the state-action Q values of error-correcting X, Y, and Z Pauli operations, occurring with probabilities p_x, p_y, and p_z, respectively. By learning to take advantage of the correlations between bit-flip and phase-flip errors, the decoder outperforms the minimum-weight-perfect-matching algorithm, achieving higher success rate and higher error threshold for depolarizing noise ($p_z = p_x = p_y$), for code distances $d \leqslant 9$. The decoder trained on depolarizing noise also has close to optimal performance for uncorrelated noise and provides functional but suboptimal decoding for biased noise ($p_z \neq p_x = p_y$). We argue that the DRL-type decoder provides a promising framework for future practical error correction of topological codes, striking a balance between on-the-fly calculations, in the form of forward evaluation of a deep Q network, and pretraining and information storage. The complete code, as well as ready-to-use decoders (pretrained networks), can be found in the repository github.com/mats-granath/toric-RL-decoder.

DOI: 10.1103/PhysRevResearch.2.023230

I. INTRODUCTION

The basic building block of a quantum computer is the quantum bit (qubit), the quantum entity that corresponds to the bit in a classical computer, but which can store a superposition of 0 and 1 [1]. The main challenge in building a quantum computer is that the qubit states are very fragile and susceptible to noise. Surface codes [2–5] are two-dimensional structures of qubits located on a regular grid which provide fault tolerance by entangling the qubits. In the surface code, logical qubits are topologically protected, which means that only strings of bit flips that stretch from one side to the other of the code cause logical bit flips, whereas topologically trivial loops (contractible to a point) do not. In recent years, experiments have taken first steps in quantum error correction in several promising quantum-computing architectures, e.g., superconducting circuits [6–15], trapped ions [16–20], and photonics [21,22], and work continues toward large-scale implementation of surface codes.

Even though the surface-code architecture provides extra protection to logical qubits, the physical qubits are still susceptible to noise causing bit-flip or phase-flip errors. Such errors need to be monitored and corrected before they proliferate and create nontrivial strings that cause logical failure. The challenge with correcting quantum-mechanical errors is that the errors themselves cannot be detected (because such measurements would destroy the quantum superposition of states), but only the syndrome, corresponding in the surface codes to local 4-qubit parity measurements, can. An algorithm that provides a set of recovery operations for correction of the error given a syndrome is called a *decoder*. As the syndrome does not uniquely determine the errors, the decoder needs to incorporate the statistics of errors corresponding to any given syndrome. Optimal decoders, which give the highest theoretically possible error-correction success rate, are generally hard to find, except for the simplest hypothetical types of noise.

Many types of decoder algorithms exist that deal in different ways with the lack of uniqueness in the mapping from syndrome to error configuration. Methods range from Markov chain Monte Carlo based decoders [23,24], cellular automata [25,26], renormalization group [27], as well as various types of neural-network-based decoders [28–40], which is also the tool used in this paper. The benchmark algorithm for the decoding problem is minimum-weight-perfect-matching (MWPM) [41–44], which is a graph algorithm for shortest pairwise matching of syndrome defects. In the standard formulation, MWPM is set up as two separate graph problems for the two types of syndrome defects, ignoring possible correlations between these or that error channels may have different probabilities.

For a decoder to be used for actual operation in a quantum computer, not only correction success rate, but also speed, is a crucial factor. A long delay for calculating error-correcting

*davidfi@chalmers.se
†mats.granath@physics.gu.se

operations will not only slow down the calculations, but also make the code susceptible to additional errors. For this reason, decoders based on algorithms that do extensive sampling of the configuration space on the fly, such as Monte Carlo based decoders [23,24], may not be viable as practical decoders. Instead, using some level of pretraining to generate and store information for fast retrieval will likely be necessary. Tabulating the information of syndrome versus most likely logical error is expected to be prohibitively expensive in terms of both storage and training, and slow to access, for anything but very small codes. Given these constraints, the need for pretraining, the massive state space and corresponding amount of data, it is natural to consider machine-learning (ML) solutions, especially given the recent deep-learning revolution [45,46] and its applications within quantum physics [47–50]. In particular, reinforcement learning and (DRL) [51,52] has recently emerged as a promising tool for various quantum control tasks [53,54].

In this paper, we use DRL, expanding on the framework for error correction for the toric code (i.e., surface code with periodic boundary conditions) introduced by Andreasson *et al.* [36]. In Ref. [36], only uncorrelated noise (with independent bit- and phase-flip errors) was considered. It was found that the DRL decoder could achieve success rates of error correction on par with MWPM. In this work, we consider depolarizing noise ($p_x = p_y = p_z$) and find that a similar decoder can outperform MWPM for moderate code size $d \leqslant 9$. The performance is instead similar to augmented versions of MWPM, optimal in the limit $p \to 0$, where correlations between phase- and bit-flip errors are taken into account [24,42]. The decoder trained on depolarizing noise is also found to be quite versatile, having MWPM success rates on uncorrelated noise, as well as giving intermediate performance on biased noise. Similarly to the previous work we do not consider syndrome measurement errors, but focus on mastering the more elementary but nevertheless challenging task of efficiently decoding a perfect syndrome with depolarizing noise.

A decoder based on DRL has the potential to offer an ideal balance between calculations on the fly and pretraining. The information about the proper error correction string for a given syndrome is stored in a very efficient way, using two principles:

(1) The step-by-step decoding using the pretrained neural network generates an effective tree structure where many different syndromes will reduce to the same syndrome after one operation, such that subsequent correction steps will use the same information, iteratively reducing the complexity.

(2) The deep neural network is a "generalizer" which can spot and draw conclusions from common features of different syndromes, including syndromes that have not been seen during training.

The paper is organized as follows. In Sec. II, we give a brief introduction to quantum error correction for the toric code. In Sec. III, we introduce deep reinforcement learning and Q learning, and discuss how these are implemented in training and utilizing the decoder. In Sec. IV, the performance of the DRL decoder is presented and benchmarked against both MWPM and analytic expression valid for low error rates. We summarize the main results and give an outlook to further developments in Sec. V.

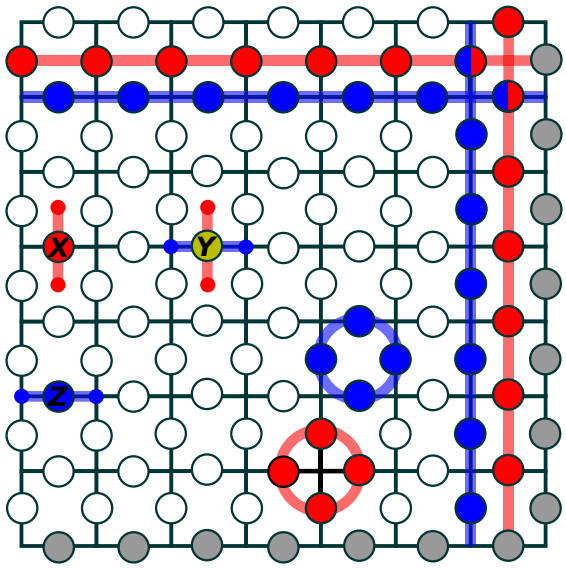

FIG. 1. A $d = 9$ toric code showing the basic operations. Circles represent physical qubits, with shading showing periodic boundaries. Bit-flip X (red), phase-flip Z (blue), and $Y \sim XZ$ (yellow) errors with corresponding plaquette and vertex "defects" as end points of error chains. The defects are measured by the plaquette ($\otimes Z$) and vertex ($\otimes X$) parity-check operators, respectively. Also shown are logical bit- and phase-flip operators corresponding to closed loops spanning the torus.

II. TORIC CODE

The toric code in the form considered here consists of a two-dimensional quadratic grid of physical qubits with periodic boundary conditions. In this section, we provide a high-level summary of the main concepts relevant for our study and refer the reader to the literature for more details [2–5]. A $d \times d$ grid contains $2d^2$ qubits corresponding to a Hilbert space of 2^{2d^2} states, out of which four will form the logical code space. That is, it encodes a fourfold qudit corresponding to two qubits, which we will nevertheless refer to as the logical qubit. It is a stabilizer code where a large set of commuting local parity-check operators (the stabilizers) split the state space into distinct sectors.

The stabilizers for the toric code are divided into two types, here represented as plaquette and vertex operators, consisting of products of Pauli Z or X operators on the four qubits on a plaquette or vertex (see Fig. 1), respectively. Eigenstates of the full set of stabilizers, with eigenvalue ± 1 on each plaquette and vertex of the lattice, are globally entangled, which provides the basic robustness to errors. The logical qubit corresponds to the sector with eigenvalue $+1$ on all stabilizers. We will refer to a stabilizer with eigenvalue -1 as a plaquette or vertex defect. A single bit flip X or phase flip Z on a state in the qubit sector will produce a pair of defects on neighboring plaquettes or vertices, with Pauli $Y \sim XZ$ giving both pairs of defects, as shown in Fig. 1.

The set of stabilizer defects corresponding to any given configuration of X, Y, or Z operations on a state in the logical sector is called the syndrome. Logical operations, which map between the different states in the logical sector,

are given by strings of X or Z operators that encircle the torus, corresponding to logical bit-flip and phase-flip operations, respectively (see Fig. 1). The shortest loop that can encircle the torus has length d; correspondingly, the *code distance* is d. For simplicity, we consider only odd d, as there is an odd-even effect in some quantitative aspects of the problem. The toric code is an example of a topological code, as the logical operations correspond to "noncontractible" loops on the torus, whereas products of stabilizers can only generate "contractible" loops.

Figure 2(a) shows an example of an error configuration (also referred to as an error chain) on a $d = 9$ toric code together with the corresponding syndrome, generated randomly at an error rate $p = 0.22$. Visible for the decoder is only the syndrome [Fig. 2(b)] based upon which the decoder should suggest a sequence of operations (a correction chain) that eliminates the syndrome in such a way that it is least likely to cause a logical bit- and/or phase-flip operation. To evaluate the success rate of a correction chain for a given syndrome, it should be complemented by the full distribution of error chains corresponding to that syndrome, to calculate which fraction of error + correction chains contain an odd number of logical operations of any type.

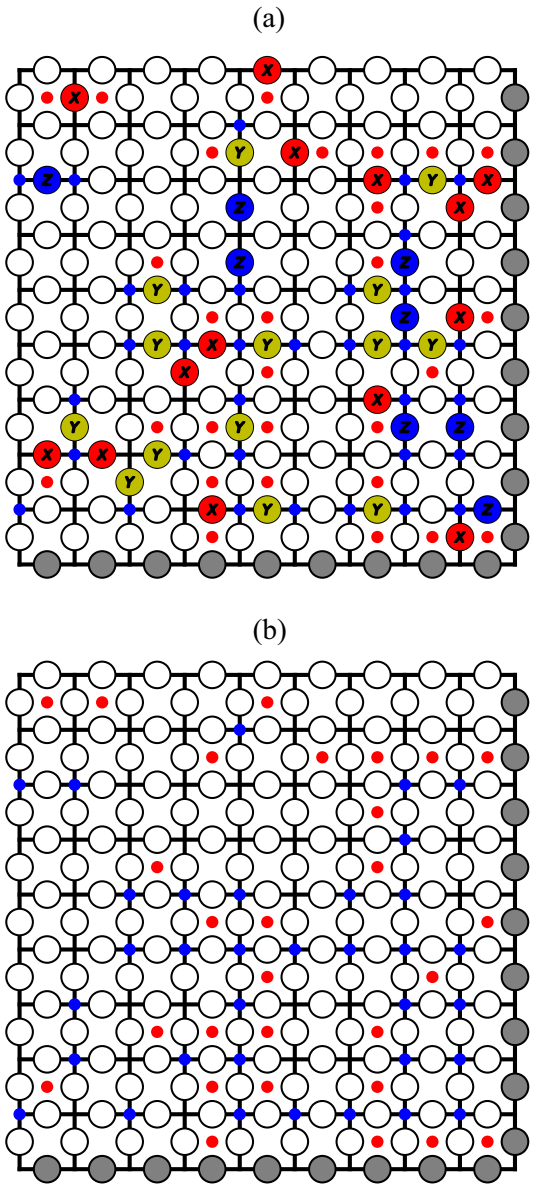

FIG. 2. Example of a random configuration of qubit errors on a $d = 9$ toric code. (a) The qubit state and the corresponding syndrome forming an error chain. (b) Syndrome given by plaquette and vertex defects. The objective of the DRL decoder is to find a correction string which is consistent with the syndrome and which takes the minimal number of qubit operations [55]. The benchmark MWPM decoder instead treats the plaquette and vertex configurations as separate graph problems, suggesting the shortest independent correction chains of X and Z.

III. DEEP REINFORCEMENT LEARNING ALGORITHM

The DRL-based decoder presented in this paper is an agent utilizing reinforcement learning together with a deep convolutional neural network, called the Q network, for approximation of Q values. The agent suggests, step by step, a sequence of corrections that eliminates all defects in the system as illustrated in Fig. 3 (see also Figs. 17 and 18 in Appendix C).

A. Q learning

The purpose of Q learning [56] is for an agent to learn a policy, $\pi(s, a)$, that prescribes what action a to take in state s. An optimal policy maximizes the future cumulative reward of actions within a Markov decision process with the rewards provided by the environment, depending on the initial and final states and the action $r_a(s, s')$. In this paper, we use a deterministic reward scheme, as discussed below. To measure the future cumulative reward, the action value function, or Q function, is given by

$$Q_\pi(s_t, a_t) = \mathbf{E}_\pi[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots], \tag{1}$$

where action a_t is taken at time t, and subsequently following the policy π, with $\gamma \leqslant 1$ a discounting factor. The Q function corresponding to the optimal policy satisfies the Bellman equation

$$Q(s_t, a_t) = r + \gamma \max_{a'} Q(s_{t+1}, a'), \tag{2}$$

such that the optimal policy will self-consistently correspond to the action maximizing Q. As discussed in more detail in Sec. III B, we use one-step Q learning, in which the current measure of $Q(s, a)$ is updated by explicit use of the Bellman equation with some learning rate α, using ϵ-greedy exploration.

The reward scheme that we use is given by

$$r_t = \begin{cases} 100 & \text{if episode terminates at step } t+1, \\ E_t - E_{t+1} & \text{otherwise,} \end{cases} \tag{3}$$

where E_t represents the number of defects in the syndrome at step t, such that X and Z operators can give reward -2, 0, or 2, whereas Y operators can give reward -4, -2, 0, 2, or 4. The terminal reward, given a discounting factor $\gamma < 1$, incites the agent to correct the full syndrome in the minimal

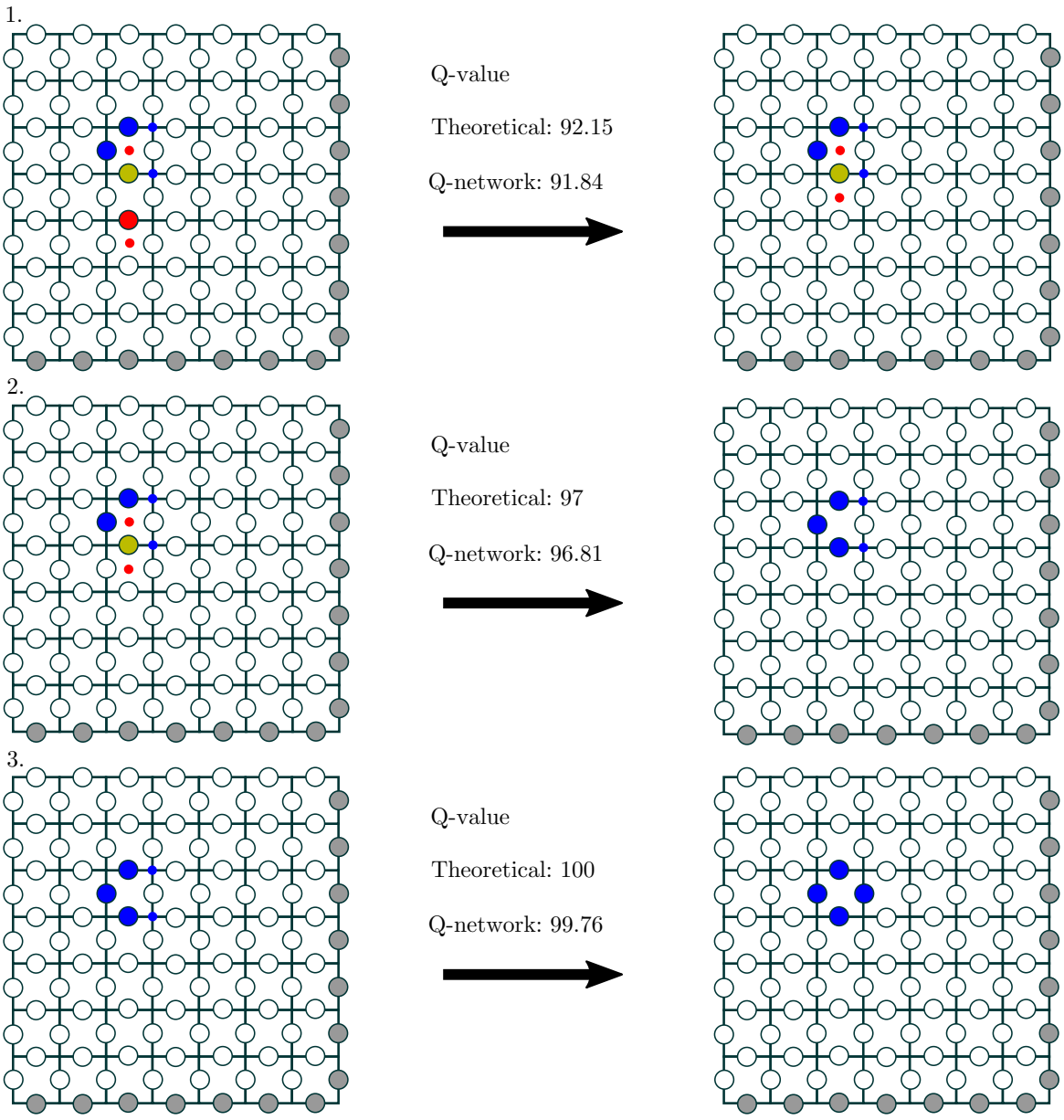

FIG. 3. Value functions $V(s) = \max_a Q(s, a)$ for a sequence of syndromes corresponding to a particular error chain, using the reward scheme in Eq. (3) with $\gamma = 0.95$. For this simple syndrome, the optimal sequence is three steps long and the theoretical state values are compared to those output by the Q network. The error chain itself is irrelevant to the correction sequence; only the syndrome is important.

number of steps. The explicit reward for eliminating defects is implemented to speed up convergence, without which the agent would have to find terminal states by completely random exploration. The reward scheme is not expected to give an optimally performing decoder [36,40]; rather than using the statistics of error chains in an unbiased fashion, it makes the assumption that the most likely error chain is the shortest. As expected (see Sec. IV), for biased noise this gives suboptimal performance.

Figure 3 shows an example of Q-network estimated and exact state values $V(s) = \max_a Q(s, a)$ for an example syndrome, showing that the Q network gives a quantitatively accurate representation of Q values. The numerical accuracy in general deteriorates the larger the syndrome is, i.e., the further it is removed from the terminal state.

Efficient Q-network representation

To improve the representational capacity of the Q network, we use an efficient state-action space representation, which was suggested in Ref. [36] for bit-flip operations and which we now extend to general X, Y, and Z operations. It is built on three basic concepts:

(i) By having the Q network only output action values for one particular qubit, the representational complexity can be reduced significantly.

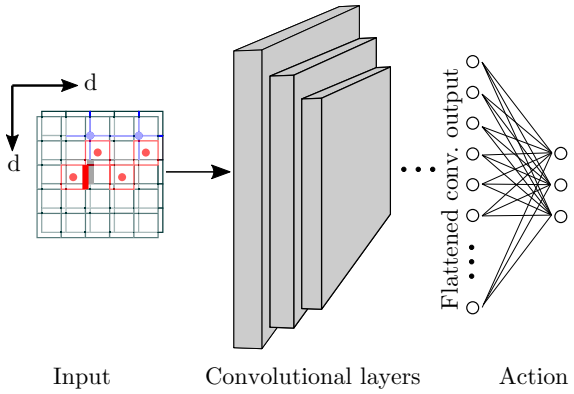

FIG. 4. Input-output structure of the deep Q network. The input is a perspective P, constructed from the syndrome s, as shown in Fig. 5. The hidden layers consist primarily of convolutional layers (see Appendix B for details). The output is the three action Q values, $Q(P, a, \theta)$, for $a \in \{X, Y, Z\}$ operators on the marked (bold) qubit, with θ representing the current state of the network.

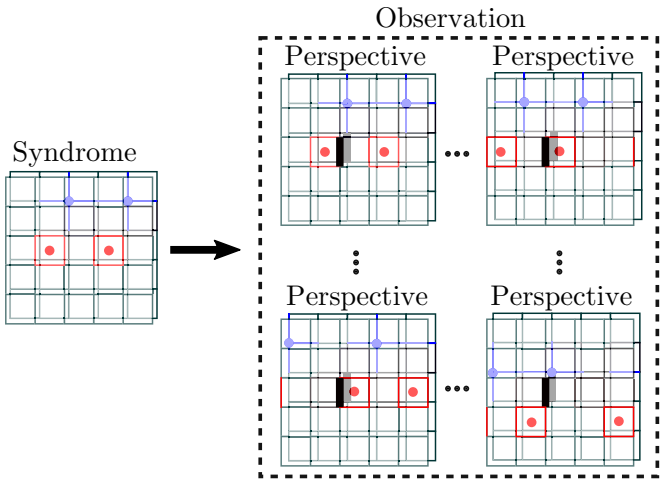

FIG. 5. Expanded representation of a syndrome into different perspectives, based on rotations and translations, used for compact processing in the Q network (Fig. 4). Only the syndrome, visible to the network, is shown, not the physical qubits. The two-layer structure corresponds to separate channels of input for vertex and plaquette defects. The set of all perspectives form an observation $O = \{P_1, P_2, \ldots, P_{N_{\text{per}}}\}$.

(ii) Due to the periodic boundary conditions of the toric code, only the relative positions of syndrome defects are important, i.e., arbitrary translations and fourfold rotations are allowed.

(iii) The converged decoder will never operate on a qubit which is not adjacent to any syndrome defect. Consequently, we have no need to calculate Q values for such actions.

The Q network takes input in the form of two channels of $d \times d$ matrices, corresponding to the location of vertex and plaquette defects, respectively. The output is the three Q values for X, Y, and Z operations on one particular qubit, in a fixed location $\vec{r}_0$ with respect to an external reference frame, as indicated in Fig. 4. To obtain the full set of action values for a syndrome, we thus successively translate and rotate the syndrome to locate each qubit at location $\vec{r}_0$. Each such matrix representation of the syndrome, with a particular qubit at $\vec{r}_0$, is called a "perspective," and the whole set of perspectives makes up an "observation," as exemplified in Fig. 5. In the observation, we only include perspectives for qubits that are adjacent to a syndrome defect.

To obtain the full relevant Q function of a syndrome, the Q function of each individual perspective of an observation is calculated. In decoding mode, the agent chooses greedily the action with the highest Q value. After the chosen action has been performed, a new syndrome is produced and the process repeats until no defects remain. As discussed in the Introduction, and exemplified in Fig. 6, the DRL decoding framework gives a compact structure for information storage and utilization: using a neural network to generalize information between syndromes and using step-by-step decoding to successively reduce syndromes to a smaller subset.

B. Training the Q network

The neural network is trained using the deep Q-learning algorithm utilizing prioritized experience replay [52,57]. To increase stability, two architecturally equivalent neural networks are used, the regular Q network, with parameters θ, and the target Q network, with parameters θ_T. The target network is synchronized with the Q network on a set interval.

Experience replay saves every transition in a memory buffer, from which the agent randomly samples a minibatch of transitions used to update the Q network. Instead of sampling the minibatch uniformly, as is done with regular experience replay, prioritized experience replay prioritizes importance when sampling. This importance is measured with the absolute value of the temporal difference (TD) error

$$\delta_j = r_j + \gamma \max_a [Q(s'_j, a; \theta_T)] - Q(s_j, a_j; \theta), \qquad (4)$$

where the state (syndrome) s'_j follows from action a_j on state (syndrome) s_j, and where the expression $Q(s, a; \theta)$ implies choosing the appropriate perspective for the Q network that corresponds to action a in syndrome s.

Following Ref. [57], the probability of sampling a transition j from the memory buffer is given by $P_j = |\delta_j|^\alpha / \sum_k |\delta_k|^\alpha$ such that values with higher TD error are more likely to be sampled. Here, α controls the amount of prioritization used ($\alpha = 0$ corresponding to uniform sampling) and $k = 1, \ldots, M$, with M the size of the memory buffer. Using nonuniform sampling in this way, however, skews the learning away from the probability distribution used to generate experiences. To partially compensate for this, importance-sampling weights are introduced according to $w_j = (MP_j)^{-\beta}$, with the product of the weights and TD error, $w_j \delta_j$, used as the loss during stochastic gradient descent training of the network. Here, β controls the extent of compensation of the prioritized sampling, with $\beta = 1$ corresponding to full compensation.

The training can be divided into two stages: the action stage and the learning stage. Pseudocode of the algorithm used for training is shown in Algorithm I. The training starts with the action stage. Given a syndrome s_t, the agent suggests an action a_t following an ϵ-greedy policy, such that with probability $(1 - \epsilon)$ the agent takes the action with the highest

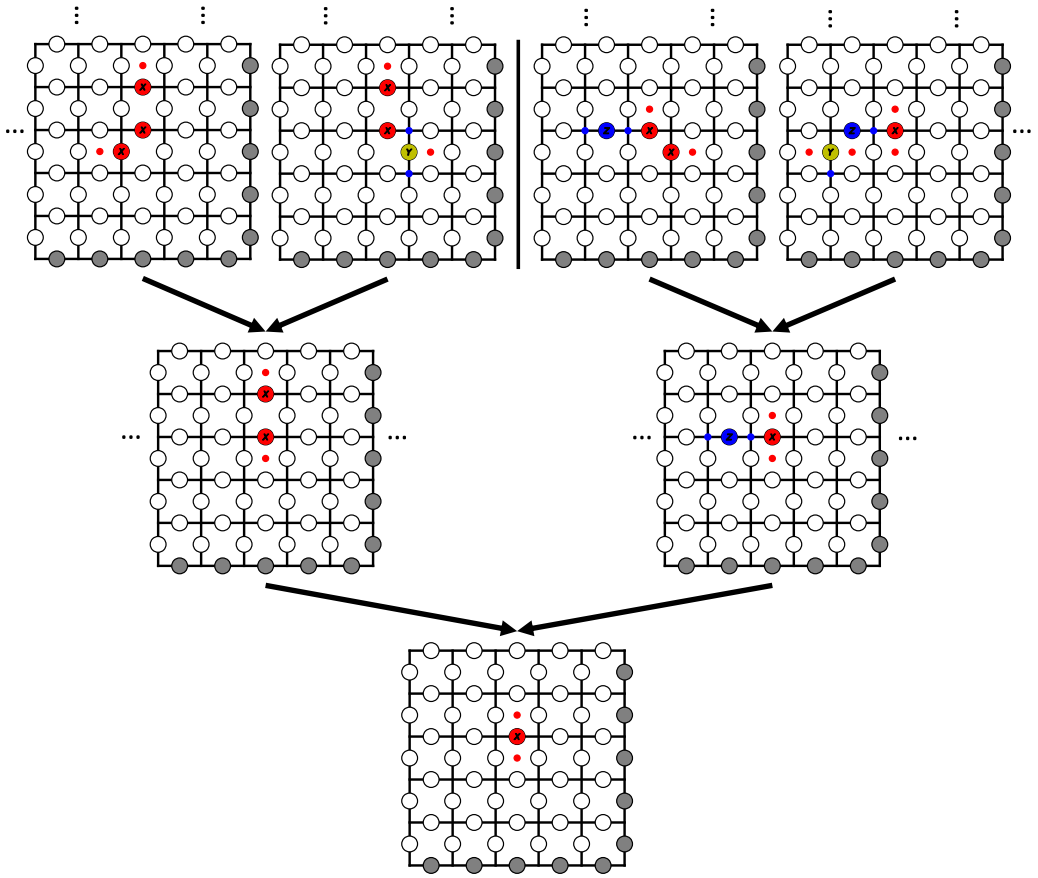

FIG. 6. Schematic of the operation of the deep reinforcement learning (DRL) decoder for several syndromes that successively reduce to a smaller subset of syndromes through step-by-step decoding. Top left are two syndromes that after one step of decoding reduces to the same syndrome, and similarly to the right. Both these branches in turn reduce to the same syndrome after the next decoding step. In this way, the complexity of the decoding problem is reduced, compared to decoding each high-level syndrome independently.

Q value; otherwise, a random action is followed. The agent receives a reward r_t and the syndrome $s'_t = s_{t+1}$, that follows from the action a_t. The transition is stored as a tuple $T = (P_t, a_t, r_t, s_{t+1}, \Theta_{t+1})$, where Θ_{t+1} is a Boolean containing the information whether s_{t+1} is a terminal state (there are no defects left) or not.

Algorithm 1. Training the DRL agent decoder.

while defects remain **do**
Get observation O_t corresponding to syndrome s_t;
With probability ϵ select random action a_t and corresponding perspective P_t;
Otherwise select:$\{P_t, a_t\} = \text{argmax}_{P,a}(Q(P, a; \theta)_{P \in O_t}$;
Execute action a_t and observe reward r_t and syndrome s_{t+1};
Store transition $(P_t, a_t, r_t, s_{t+1}, \Theta_{t+1})$ in replay memory;
Sample random minibatch of transitions $\{T_j\}_{j=1}^N$ from replay memory using prioritized sampling;
Calculate weights used for weighted importance sampling w_j;
If terminal state reached, set $y_j = r_j$; otherwise, set $y_j = r_j + \gamma \max_a Q(s'_j, a; \theta_T)$;
Perform gradient descent step on $w_j |y_j - Q(P_j, a_j; \theta)|$ with respect to the network parameter θ;
Every C steps synchronize the target network with the policy network, $\theta_T = \theta$.
end

After the action stage, the agent continues with the learning stage. For that we use stochastic gradient descent (SGD) and the tuples stored in the replay memory. A minibatch of N transitions, $\{T_j = (P_j, a_j, r_j, s'_j, \Theta_j)\}_{j=1}^N$, is sampled from the replay memory with replacement. The training target value for the policy Q network is given by $y_j = r_j$ if $\Theta_j = 1$, and $y_j = r_j + \gamma \max_a Q(s'_j, a; \theta_t)$ otherwise.

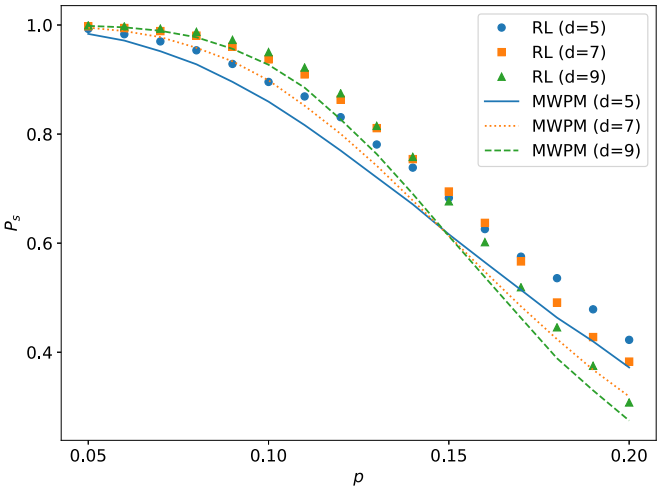

FIG. 7. Error-correction success rate P_s for the DRL decoder on depolarizing noise, as a function of total error probability p, for system sizes $d = 5, 7, 9$ (blue circles, orange squares, and green triangles, respectively), and compared to the corresponding results using the MWPM algorithm (blue solid curve, orange dotted curve, and dashed green curve, respectively). The DRL-based algorithm outperforms the MWPM-based algorithm for all these system sizes and error rates.

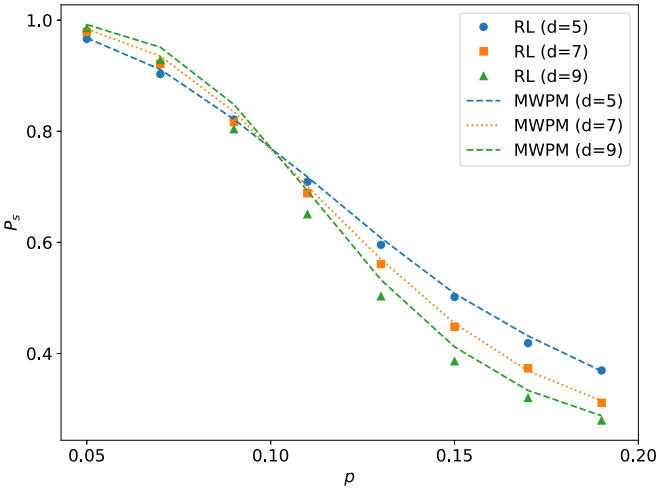

FIG. 8. Error-correction success rate P_s for the DRL decoder trained on depolarizing noise, when applied to pure bit-flip noise, as a function of error probability p. Dashed curves show the corresponding results using the MWPM algorithm.

The agents are initially trained with an error rate of 10% and further during the training with syndromes up to 30% error rate. Details of network architectures and hyperparameters are found in Appendix B.

IV. RESULTS

A. Depolarizing noise

The main result of the paper is displayed in Fig. 7, where the error-correction success rate for depolarizing noise, $p_x = p_y = p_z = p/3$, is shown for decoders trained at three different code dimensions. This is compared to MWPM, which treats the plaquette and vertex defects as separate graph problems. See comment[1] for a discussion about the MWPM decoder for depolarizing noise. We thus find that the DRL decoder has a significantly higher error-correction success rate, which is achievable by learning to account for the correlations between plaquette and vertex defects.

From the crossing of the $d = 5$ and 7 error-correction success rates, we can identify a threshold of around 16.5% (for MWPM, the crossing is close to 15%), below which error correction can be guaranteed, were we able to increase d arbitrarily. The deduced threshold is significantly below the theoretical limit of 18.9% [23,58], but similar to that found for the Markov-chain Monte Carlo decoder based on shortest average correction chain formulated in Ref. [24]. As discussed in the Introduction, for a practical decoder the threshold may not be the most important measure. Nevertheless, we anticipate that the success rate and threshold can be enhanced by further developing the reward scheme to be based on success rate rather than minimum number of operations. (Work along these lines was recently presented by Colomer *et al.* [40].)

We also note that even though the $d = 9$ DRL decoder gives a significant improvement over MWPM, it has not fully converged to the optimal performance within the limitations of the algorithm, as indicated by the earlier crossing with $d = 5$ and 7. We do not anticipate that this is a fundamental limitation of the DRL-type decoder, but could be improved by a more efficient training scheme.

In Fig. 8, we have employed the same DRL decoders, pretrained on depolarizing noise, to decode pure bit-flip noise. Here, we find a performance for $d = 5$ and 7 which is very close to MWPM, thus reproducing the results of our first-generation DRL decoder from Ref. [36]. For $d = 9$, the decoder has slightly worse performance, confirming that this decoder has not yet converged to optimal algorithmic performance.

B. Asymptotic fail rates

In addition to the MWPM benchmark, we also benchmark the DRL decoders for small error rates $p \to 0$, by deriving analytical expressions (see Appendix A) for the fail rate for depolarizing noise to lowest nonvanishing order in p. We can derive such fail rates for both the MWPM algorithm and the algorithm based on finding the shortest correction strings. The latter is similar to, but not exactly equivalent with, what we expect for the DRL decoder based on our reward scheme. These algorithms both have a fail rate that scales as $P_L \sim p^{\lceil \frac{d}{2} \rceil}$, but with different prefactors.

In Fig. 9, we confirm that the DRL decoder indeed performs ideally for $d = 5$ and 7 for short error chains, following very closely the algorithm based on minimal X, Y, Z chains.

[1]The MWPM decoder assumes that X and Z errors are uncorrelated, with independent error rates $p_x = p_z = 2p/3$ and, correspondingly, $p_y = (2p/3)^2$. The MWPM success rate for that problem would be $P_S(p) = (P_{S,X}(2p/3))^2$, with $P_{S,X}(p)$ corresponding to pure bit-flip noise (Fig. 8). This expression is a good approximation to the MWPM success rate for depolarizing noise which is exact in the low-p limit (see Appendix A).

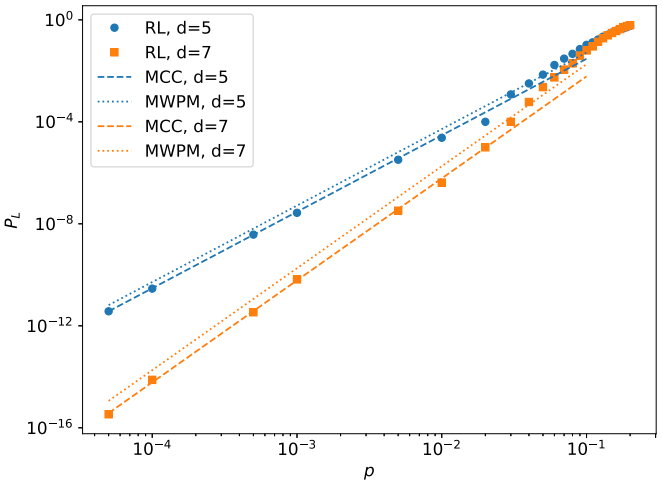

FIG. 9. Error-correction fail rate P_L of the DRL decoder for depolarizing noise ranging from small to large error rates. (The $p \geqslant 0.05$ data are the same as in Fig. 7.) The dashed and dotted lines correspond to analytic expressions [Eqs. (A4) and (A8) in Appendix A], valid to lowest order in p, for a decoder that operates based on the minimal correction chain (MCC) or the MWPM algorithm. The MCC decoder is optimal for $p \to 0$.

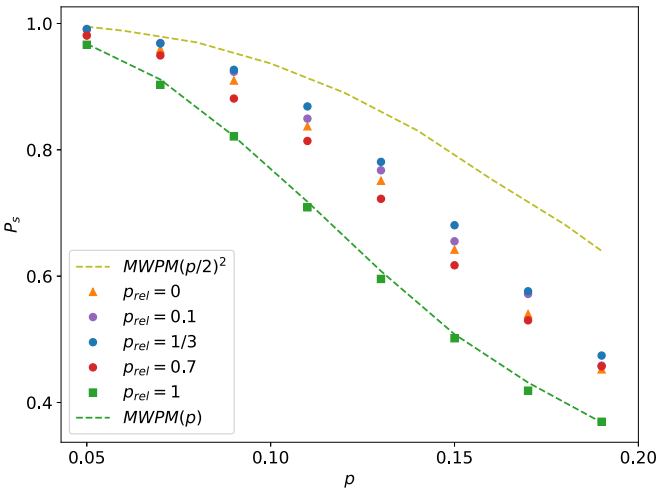

FIG. 10. Error-correction success rate for biased noise ($d = 5$)$p_z = p_{\rm rel}p$, $p_x = p_y = (1 - p_{\rm rel})p/2$, using a decoder trained on depolarizing noise ($p_{\rm rel} = \frac{1}{3}$). For pure phase-flip noise ($p_{\rm rel} = 1$), the decoder is compared to MWPM. The line MWPM $(p/2)^2$ indicates expected performance for an MWPM decoder designed explicitly for $p_z = 0$ noise.

Because of the excessive time consumption to generate good statistics for $d = 9$, we have only compared the performance in the true asymptotic limit, i.e., the rate for only the shortest fallible error chains, as shown in Table I, again confirming the suboptimal performance for $d = 9$. In this limit, data are generated by only considering the subgroup of error chains that are in a single row or column, in contrast to generating completely random error chains that will very rarely fail.

C. Biased noise

For the prospect of an operational decoder on a physical quantum computer, the noise is expected to be biased, such that phase-flip errors are relatively less or more likely [59–64]. To identify the exact error distribution is a challenging problem in itself (see, e.g., Ref. [65]), and the degree of bias can fluctuate in time [62–64], so a decoder that can adequately decode biased noise without retraining might be an alternative. To quantify the performance of the DRL decoder for biased noise, we consider the probability of an error of any type p, probability of phase-flip error $p_z = p_{\rm rel}p$, and consequently $p_x = p_y = (1 - p_{\rm rel})p/2$. Thus, for $p_{\rm rel} = 1$ the syndromes contain only Z errors, which corresponds to uncorrelated noise, whereas $p_{\rm rel} = \frac{1}{3}$ corresponds to depolarizing noise.

In Fig. 10, we show the success rate for the decoder on biased noise. We find that the highest success rate is attained for depolarizing noise, which also is what the decoder is trained for. We can understand this as a consequence of the superlinear decline (for low p) in success rate with the number of defects, such that the majority species dominates the outcome. At $p_{\rm rel} = \frac{1}{3}$ there is an equal mean number of vertex and plaquette defects, while away from this limit, the number of either one or the other grows. That the operation of the trained DRL decoder is suboptimal is clear from the limit $p_{\rm rel} = 0$, corresponding to only X and Y errors, which should, in principle, be a simpler decoding problem, similar to uncorrelated noise with independent error rates $p/2$.[2] Nevertheless, the decoder gives fair performance for the full range of biased noise, which may be an advantage over having a decoder which is specialized to a particular, potentially unknown, bias.

TABLE I. Comparison of asymptotic logical fail rates P_L.

	Analytic	DRL decoder
$d = 5$	1.51×10^{-3}	1.45×10^{-3}
$d = 7$	2.12×10^{-5}	2.07×10^{-5}
$d = 9$	2.50×10^{-7}	4.30×10^{-7}

V. CONCLUSION AND OUTLOOK

We have shown how deep reinforcement learning can be used for quantum error correction of depolarizing noise ($p_x = p_y = p_z$) in the toric code, with significantly improved performance compared to the standard MWPM algorithm. The advantage is gained by learning to account for the correlations between the vertex and plaquette defects. The super-MWPM performance for depolarizing noise was achieved for system sizes up to $d = 9$, corresponding to 162 qubits. However, by

[2]Even though the limit $p_z = 0$ corresponds to a surplus of plaquette defects versus vertex defects, the decoding problem is, in principle, equivalent to the problem of *noncoinciding* X and Z errors with error rates $p_x = p_z = p/2$: the decoder could first decode the vertex defects using Y operators, and subsequently decode the remaining plaquette defects using X. The corresponding uncorrelated problem [with nonzero coincidence probability $(p/2)^2$] would have MWPM success rate $P_S = (P_{S,X}(p/2))^2$, which we expect is still a good approximation (for small p) and also close to optimal for this weakly correlated noise.

applying the trained decoder to decode pure bit-flip noise, it was found that ideal performance was only achieved for $d < 9$. For biased noise ($p_z \neq p_x = p_y$), the decoder gives fair, but suboptimal, success rates.

A crucial question that needs to be explored in subsequent work is how to scale up the DRL decoder to larger codes, and how this will effect the decoder speed. One limitation that we encounter is an increasingly slow training convergence with the increasing network size used for larger d. In contrast to supervised learning using preannotated data, allowing for very high throughput training of the deep neural network, a challenge with DRL is that the training data are generated using the network itself which limits the pace of data generation. To improve this, we are currently implementing distributed reinforcement learning [66], where a large set of agents independently explore the environment to fill a common memory buffer, allowing for better hardware utilization and decreased training times.

The type and depth of network best suited for the task also needs to be explored in a systematic way. For $d = 9$, we are currently using a deep residual neural network, for which skip connections are known to improve convergence [67], and which is the workhorse for DRL [68]. Nevertheless, going to significantly larger networks also increases the hardware requirements, and even if it is possible to train a very large network, the time required for forward propagation through the network will limit the decoding speed. As a primer for a more systematic study of the DRL decoder execution time, we show in Table V in Appendix B the time per step of error correction. As expected, this time grows with code distance, reflecting the time consumption for the policy generation using an increasingly deep neural network.

A promising path to improving the performance of the decoder is to go beyond the conceptually simple but inefficient Q learning. The action-value function contains more information than is actually needed for the decoding task; instead, a policy (best action) for each syndrome is sufficient. (Although, the advantage of a Q network for our implementation is that the Q values allow for independent evaluation for each qubit action.) We are currently working on implementing the ALPHAZERO algorithm that combines a trained policy (and value) network with an on-the-fly Monte Carlo tree search [68,69], and which has recently been applied to quantum control problems [70]. A drawback of this approach is the additional computational demand during operation of the decoder. A simpler approach would be to use a policy-based algorithm, such as the REINFORCE algorithm [71]. This algorithm directly optimizes a policy without calculating action values or performing any kind of tree search. A natural extension is the class of actor-critic methods [72]. These combine concepts from value- and policy-based methods and are more robust and stable during the training. Moreover, it could be worth investigating the possibility of transferring the domain-specific knowledge (transfer learning) obtained from small grid instances to comparably larger grid sizes [73].

Another important limiting component to the DRL decoder performance is the reward scheme. In this work, we use the heuristic to minimize the length of correction chains, which is only optimal for $p \to 0$ [24,42]. To improve performance for larger error rates and for biased noise, with greater or smaller probability of phase-flip errors, we are currently exploring a reward scheme based on a Monte Carlo generated distribution of error chains for each syndrome [23,24].

In addition to improving the prowess of the DRL decoder for the problem discussed in this paper, further developments should include addressing syndrome measurement errors and nontoric topological codes [35]. Even though the DRL-type decoder presented in this paper and in Refs. [36,40] is still limited in scope, we have shown that it can flexibly address various types of noise, and in some regimes give super-MWPM performance. In addition, the information gathered from exploration is stored and used in an efficient and generalizable way using a deep neural network and step-by-step error correction, limiting both the complexity of concurrent calculations and the need for massive information storage, which may be instrumental for future operational decoders.

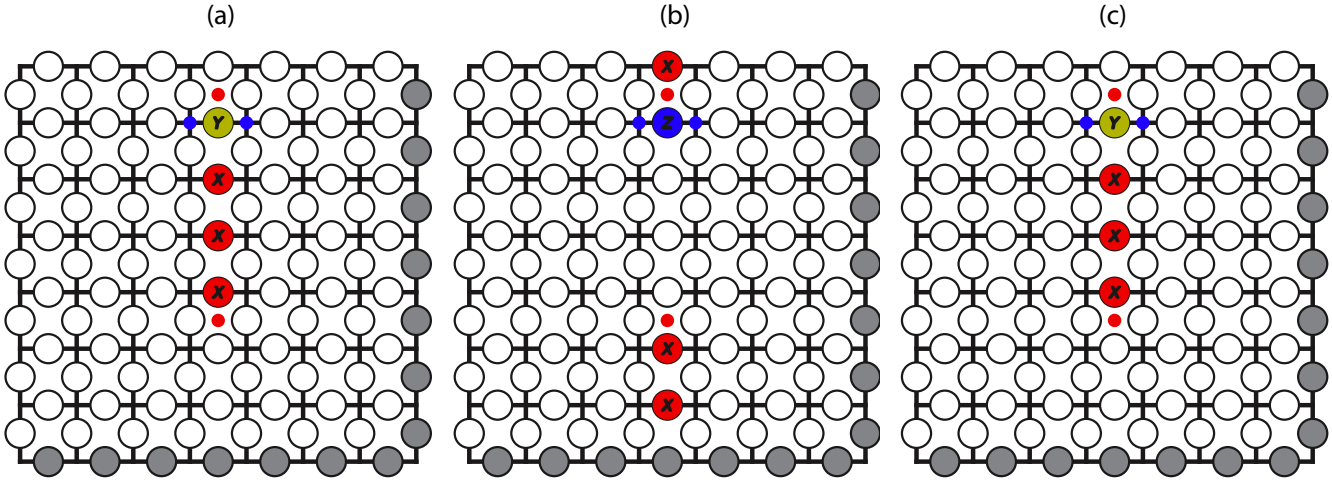

FIG. 11. (a) The initial syndrome corresponding to one Y error and three X errors. (b) MWPM will always introduce a nontrivial loop and therefore fail. The "minimum correction chain" decoder has a 50% probability each for failure and success [correction chains (b) or (c), respectively].

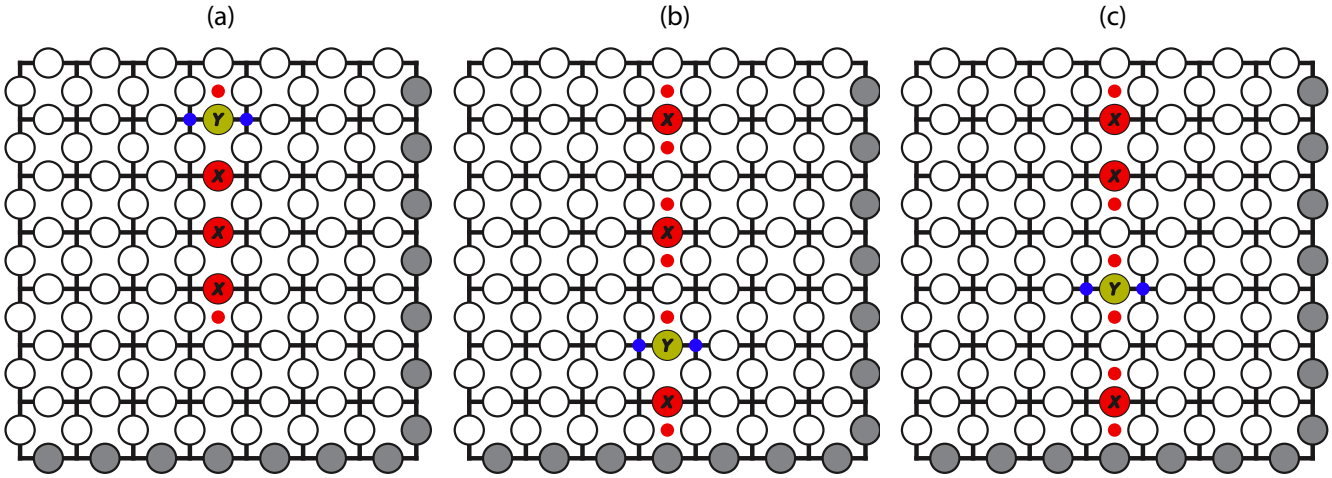

FIG. 12. (a)–(c) For each of these syndromes, the shortest correction chains are of the same length (four steps in all cases). This is also true for other constellations of errors. The length of the error correction chain does not depend on the relative position of the syndrome defects in a row or a column.

ACKNOWLEDGMENTS

Computations were performed on the Vera cluster at Chalmers Centre for Computational Science and Engineering (C3SE). We acknowledge the financial support from the Knut and Alice Wallenberg Foundation through the Wallenberg Center for Quantum Technology (WACQT).

APPENDIX A: SMALL ERROR RATE

It is possible to derive a theoretical expression for the logical fail rate, that becomes exact in the limit of low error probabilities, by considering only the shortest possible error strings that may lead to an error given the decoding scheme. Here, we derive such expressions for depolarizing noise $p_x = p_y = p_z = \frac{p}{3}$ for an algorithm which is based on correction using the minimum number of correction steps, and for an algorithm which is based on using MWPM separately on the graphs given by plaquette and vertex errors. The former algorithm, which we refer to as "minimal correction chain" (MCC), is similar to, but not exactly equivalent to, our trained decoder since our reward scheme, in addition to penalizing steps, also gives reward for annihilating syndrome defects. The latter will give a slight priority to using Y operators (which can annihilate two pairs of defects) at an early stage of the decoding sequence. Nevertheless, we expect that this algorithm serves as a good benchmark for how well our DRL implementation of the algorithm works. In particular, we would like to see that our decoder outperforms the MWPM decoder also for low error rates.

The shortest error strings that can give an error with either of the algorithms are $\lceil \frac{d}{2} \rceil$ long, aligned along one row or column [24,36,42]. This means that the fail rate for both types of decoders will scale as $P_L \sim (p/3)^{\lceil \frac{d}{2} \rceil}$ for small p, but with different prefactors. We will only consider odd d; the scaling is true for even d, but prefactors are different. Figure 11 gives a demonstrative example of an error string, for $d = 7$, where the outcome differs between the two algorithms. Here, MWPM will fail, solving the vertex defects with one Z and the plaquette defects with two X to generate a logical bit flip consisting of a vertical X loop. In contrast, the MCC algorithm will only fail 50% of the time (we assume draws are settled by a coin flip), either using the MWPM-prescribed sequence or using the actual error string ($YXXX$) as the correction string. Interestingly, our specific decoder implementation should succeed 100% of the time for this particular error string since it will prefer to use the Y, but it is not clear that this advantage is general.

To derive the general expressions for the asymptotic fail rates, we go through several examples of error chains. First, one has to keep in mind that we are interested in the minimum amount of steps to annihilate all excitations. The order in which the errors are placed in the chain does not matter (see Fig. 12). Also, the errors do not have to be connected; it is a sufficient criterion that they all are in one column or row.

Now, we can investigate the different combinations that can make the decoder fail. Length $\lceil \frac{d}{2} \rceil$ error chains containing either only X or Z errors will always generate a nontrivial loop (see Fig. 13). Moreover, combinations of X and Y errors can lead to a failure. Figures 11 and 14 show that we have to consider syndromes with exactly one Y error and the rest uniformly X or Z errors. For two or more Y errors, the decoder will always succeed with the error correction. Finally, we have to find out how X and Z errors in combination behave. Figures 15 and 16 show that for exactly one Z error and the rest being X errors, the decoder succeeds with a 50% chance. Here again, the reward scheme of the actual DRL decoder would disfavor using a Y if the Z is isolated, giving a slight discrepancy between this and the MCC algorithm.

We can convince ourselves that the cases presented here generalize to larger odd d, allowing for the derivation of an analytic expression for the logical fail rate. For the MCC algorithm, which we identify as close to the performance of our DRL decoder, the fail rate is given by

$$\begin{aligned} P_{L_{\mathrm{MCC}}} = {} & P(\{XX\ldots X\}) + P(\{ZZ\ldots Z\}) \\ & + P(\{YX\ldots X\}) + P(\{YZ\ldots Z\}) \\ & + P(\{ZX\ldots X\}) + P(\{XZ\ldots Z\}), \end{aligned} \tag{A1}$$

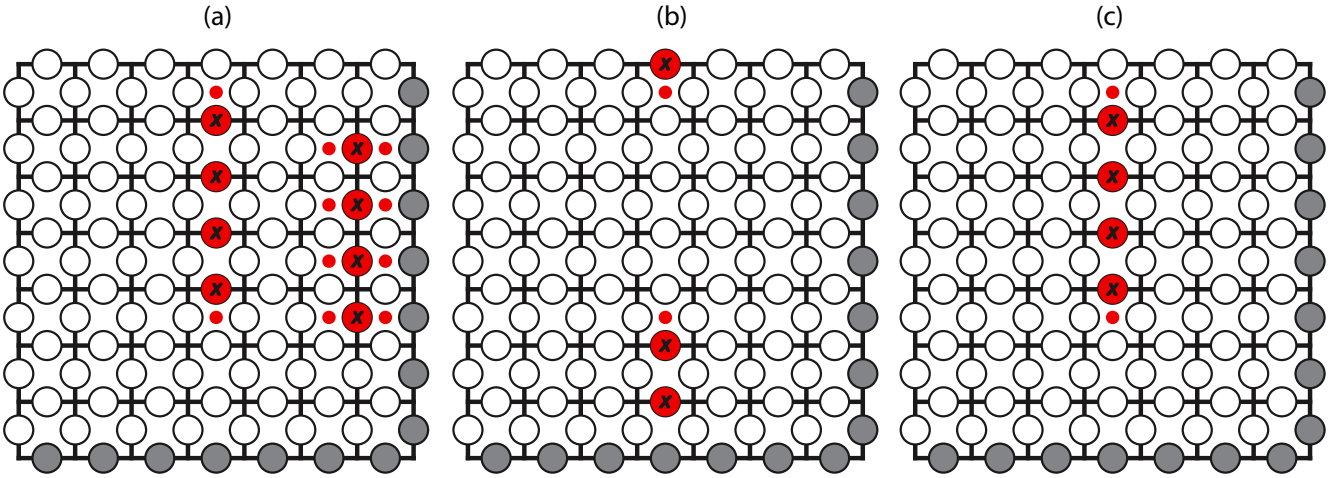

FIG. 13. (a) The initial syndrome with four X errors. (b) The minimum amount of steps, three, to merge the excitations is by introducing a nontrivial loop around the torus. (c) Revoking the errors introduced would take four steps. Any decoder will fail on such error chains with 100% certainty. Note that X chains of errors on the columns with vertical bonds, or rows with horizontal bonds, will not give quantum error correction failure (a).

where $\{\ldots\}$ indicates any configuration of errors in one row or column.

To lowest order in p [i.e., ignoring factors that are powers of $(1-p)$], the probability of $\lceil \frac{d}{2} \rceil$ errors of the same type is given by

$$P(\{XX\ldots X\}) = P(\{ZZ\ldots Z\}) = 2d\binom{d}{\lceil \frac{d}{2} \rceil}\left(\frac{p}{3}\right)^{\lceil \frac{d}{2} \rceil}, \quad \text{(A2)}$$

where the $2d$ corresponds to the number of rows and columns (with the appropriate orientation of bonds; see Fig. 13). The probability of failure from the mixed-type chains is given by

$$\begin{aligned} P(\{YX\ldots X\}) &= P(\{YZ\ldots Z\}) \\ &= P(\{ZX\ldots X\}) = P(\{XZ\ldots Z\}) \\ &= \frac{1}{2}2d\binom{d}{1}\left(\frac{p}{3}\right)\binom{d-1}{\lceil \frac{d}{2} \rceil - 1}\left(\frac{p}{3}\right)^{\lceil \frac{d}{2} \rceil - 1} \\ &= d\left\lceil \tfrac{d}{2} \right\rceil \binom{d}{\lceil \frac{d}{2} \rceil}\left(\frac{p}{3}\right)^{\lceil \frac{d}{2} \rceil}, \end{aligned} \quad \text{(A3)}$$

where the $\frac{1}{2}$ comes from 50% failure for this type of configuration. Inserting Eqs. (A2) and (A3) in Eq. (A1) and simplifying, we obtain the following probability of failure in the case of very low p:

$$P_{L_{\mathrm{MCC}}} = 4d\left(1 + \left\lceil \tfrac{d}{2} \right\rceil\right)\binom{d}{\lceil \frac{d}{2} \rceil}\left(\frac{p}{3}\right)^{\lceil \frac{d}{2} \rceil}. \quad \text{(A4)}$$

For reference, we mention the corresponding expression for d even. Here, pure chains of all X or all Z of length $d/2$ will fail with 50% chance, whereas for all mixed chains error

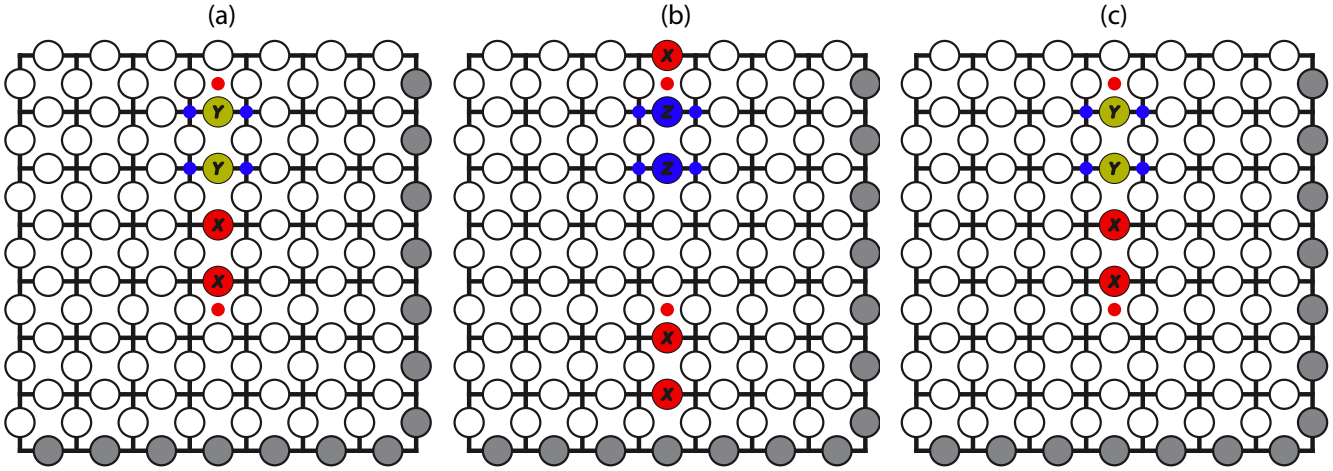

FIG. 14. (a) The initial syndrome with two Y operators in the error chain. (b) Five steps are needed if one uses Z operators. (c) There is only one shortest correction chain with four steps. We can also conclude that with at least two or more Y errors in the chain, the MCC algorithm (and DRL decoder) always succeeds with the error correction. In contrast, MWPM will fail, using the middle chain (b).

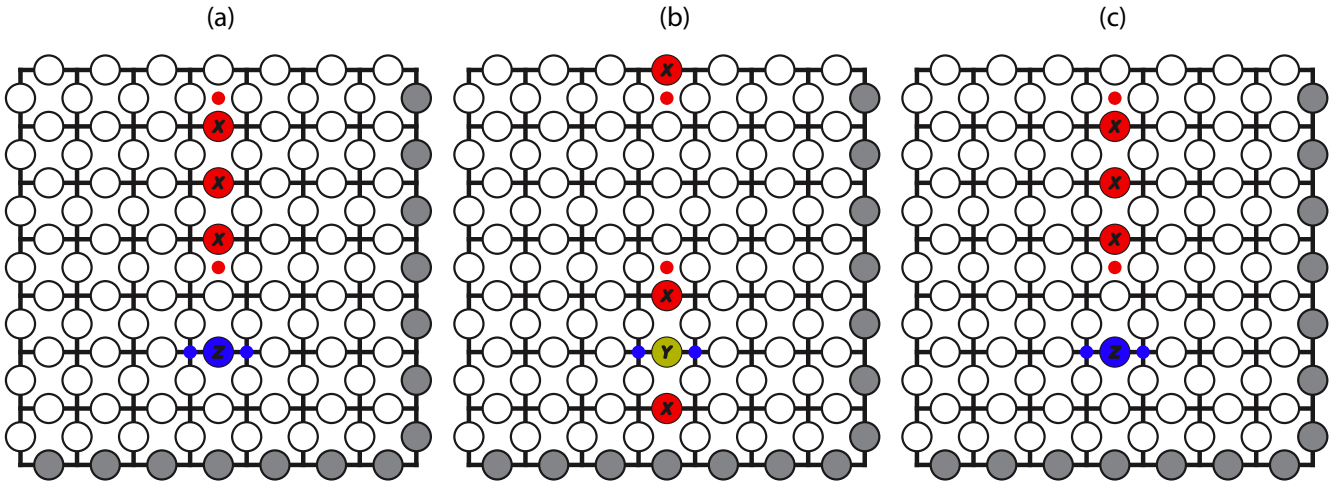

FIG. 15. (a) The initial syndrome with one Z and three X errors. There are two possible minimal error-correction chains, one leading to (b) a failed and one leading to (c) a successful error correction. We assign 50% chance to each outcome. Interestingly, the MWPM algorithm will always succeed on these kinds of syndromes as Y would count as two operators.

correction will succeed. This gives a fail rate

$$P_{L_{\mathrm{MCC,even}}} = 2d\binom{d}{d/2}\left(\frac{p}{3}\right)^{d/2}. \tag{A5}$$

This expression can be compared to Eq. (3) of Fowler [42] for the surface code, where the factor-of-4 difference comes from us counting both X and Z logical failure and from the fact that for the toric code these can be both "horizontal" and "vertical."

To derive the corresponding asymptotic fail rate for the MWPM algorithm, we use the fact that it only uses X and Z for correction. This decoder (similarly to any reasonable decoder) will always fail for chains of length $\lceil \frac{d}{2} \rceil$ in a row or column containing all X or all Z. It will also fail if one or more of the X or Z in such a chain are replaced by Y. This is clear from, e.g., correcting a Y with a Z in a chain $\{YXX\ldots\}$, which will reduce the chain to a pure $\{XXX\ldots\}$ of the type that always fails:

$$\begin{aligned} P_{L_{\mathrm{MWPM}}} &= P(\{XX\ldots X\}) + P(\{ZZ\ldots Z\}) \\ &\quad + P(\{YX\ldots X\}) + P(\{YZ\ldots Z\}) \\ &\quad + \cdots + P(\{YY\ldots Y\}), \end{aligned} \tag{A6}$$

where the ellipsis indicates chains with increasing numbers of Y. The general expression for $N_y \in \{0, 1, \ldots, \lceil \frac{d}{2} \rceil\}$ Y errors in a chain with $\lceil \frac{d}{2} \rceil - N_y$ X (Z) errors reads as

$$\begin{aligned} &P(\{YY\ldots XX\}) \\ &\quad = P(\{YY\ldots ZZ\}) \\ &\quad = 2(1+\delta_{N_y,\lceil \frac{d}{2} \rceil})d\binom{d}{N_y}\binom{d-N_y}{\lceil \frac{d}{2} \rceil - N_y}\left(\frac{p}{3}\right)^{\lceil \frac{d}{2} \rceil}, \end{aligned} \tag{A7}$$

where, compared to Eq. (A3), there is no $\frac{1}{2}$, as these chains always fail using MWPM, and where the chain consisting purely of Y is multiplied by a factor of 2 because it will fail on

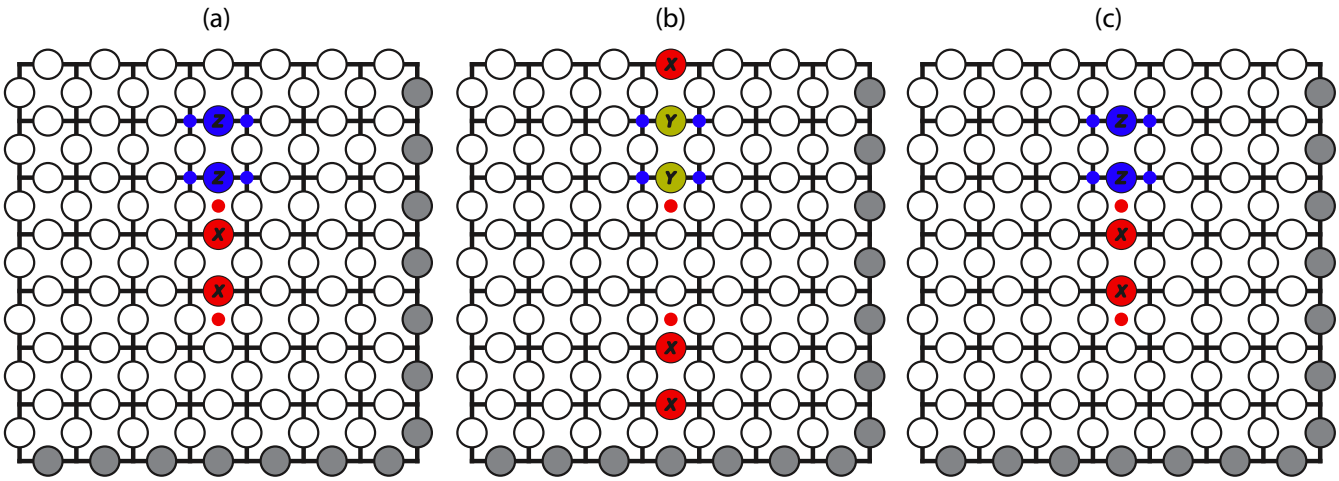

FIG. 16. The shortest error-correction chain for the initial syndrome is (a) four steps by simply (c) reversing the changes. (b) Using Y operators would take five steps and will therefore not be chosen by the decoder. The agent always succeeds on these syndromes.

TABLE II. List of hyperparameters and their values.

Hyperparameter	Value	Description
Minibatch size	32	Number of training samples used for stochastic gradient descent update
Training steps	10 000	Total amount of training steps per epoch
Replay memory size, N	10 000	Total amount of stored memory samples
Priority exponent, α	0.6	Prioritized experience replay parameter
Importance weight, β	0.4	Prioritized experience replay parameter
Target network update frequency, C	1000	The frequency with which the target network is updated with the policy network
Discount factor, γ	0.95	Discount factor γ used in the Q-learning update
Learning rate	0.00025	The learning rate used by Adam
Initial exploration	1	Initial value of ϵ in ϵ-greedy exploration
Final exploration	0.1	Final value of ϵ in ϵ-greedy exploration
		A random policy generates training samples to populate the replay memory before the learning starts
Optimizer	Adam	Adam is an optimization algorithm used to update network weights
Max steps per episode	75	Number of steps before every episode is terminated

both types (X or Z) of rows and columns. Thus, the complete expression for the MWPM asymptotic fail rate reads as (after summation over N_y)

$$P_{L_{\mathrm{MWPM}}} = 4d\ 2^{\lceil \frac{d}{2} \rceil} \binom{d}{\lceil \frac{d}{2} \rceil} \left(\frac{p}{3}\right)^{\lceil \frac{d}{2} \rceil}. \tag{A8}$$

As expected, we find a higher fail rate for the decoder that uses MWPM compared to the decoder using the minimum number of correction steps, with $P_L/P_{L_{\mathrm{MWPM}}} = (1 + \lceil \frac{d}{2} \rceil)/2^{\lceil \frac{d}{2} \rceil} < 1$ for $d \geqslant 3$.

We also note that the asymptotic fail rate for pure bit-flip (or phase-flip) noise with error rate p is given by Eq. (A2) with $p/3 \to p$, $P_{L,X}(p) = 2d\binom{d}{\lceil \frac{d}{2} \rceil} p^{\lceil \frac{d}{2} \rceil}$. Thus, under the assumption of uncorrelated X and Z errors with probability $2p/3$ (corresponding to the rates for depolarizing noise) we find exactly that the total fail rate in Eq. (A8) is given by adding up two independent error channels: $P_{L_{\mathrm{MWPM}}} = 2P_{L,X}(2p/3)$.

Another useful representation is to calculate the ratio of error chains with $\lceil \frac{d}{2} \rceil$ errors that lead to a failure compared to the total number of chains with $\lceil \frac{d}{2} \rceil$ errors:

$$f_{RL} = \frac{4d\left(1 + \lceil \frac{d}{2} \rceil\right)\binom{d}{\lceil \frac{d}{2} \rceil}}{\binom{2d^2}{\lceil \frac{d}{2} \rceil}\lceil \frac{d}{2} \rceil^3}. \tag{A9}$$

Accordingly, for the MWPM,

$$f_{\mathrm{MWPM}} = \frac{4d\ 2^{\lceil \frac{d}{2} \rceil}\binom{d}{\lceil \frac{d}{2} \rceil}}{\binom{2d^2}{\lceil \frac{d}{2} \rceil}\lceil \frac{d}{2} \rceil^3}. \tag{A10}$$

TABLE III. Network architecture for $d = 5$. Every convolutional layer has a kernel size of 3 and stride 1. Periodic padding is applied to the first convolutional layer. The other convolutional layers work with zero padding.

No.	Type	Size	No. parameters
1	Conv2d	128	2432
2	Conv2d	128	147 584
3	Conv2d	120	138 360
4	Conv2d	111	119 991
5	Conv2d	104	104 000
6	Conv2d	103	96 511
7	Conv2d	90	83 520
8	Conv2d	80	64 880
9	Conv2d	73	52 633
10	Conv2d	71	46 718
11	Conv2d	64	40 960
12	Linear	3	1731
			899 320

TABLE IV. Network architecture for $d = 7$. Every convolutional layer has a kernel size of 3 and stride 1. Periodic padding is applied to the first convolutional layer. The other convolutional layers work with zero padding.

No.	Type	Size	No. parameters
1	Conv2d	256	4864
2	Conv2d	256	590 080
3	Conv2d	251	578 555
4	Conv2d	250	565 000
5	Conv2d	240	540 240
6	Conv2d	240	518 640
7	Conv2d	235	507 835
8	Conv2d	233	493 028
9	Conv2d	233	488 834
10	Conv2d	229	480 442
11	Conv2d	225	463 950
12	Conv2d	223	451 798
13	Conv2d	220	441 760
14	Conv2d	220	435 820
15	Conv2d	220	435 820
16	Conv2d	215	425 915
17	Conv2d	214	414 304
18	Conv2d	205	395 035
19	Conv2d	204	376 584
20	Conv2d	200	367 400
21	Linear	3	15 003
			8 990 907

TABLE V. Operational decoding time per correction step t_{step}, i.e., time per Pauli operator of the correction chain, for two different error rates and the three code distances. The bulk of the time consumption is in the forward evaluation of the network.

p	0.05	0.1
$d = 5$	0.0111 s	0.0131 s
$d = 7$	0.0268 s	0.0303 s
$d = 9$	0.0818 s	0.1148 s

APPENDIX B: MODEL DEFINITION, HYPERPARAMETERS, AND RUNNING TIME

In this Appendix, we list relevant parameters for our neural networks. Table II shows the different hyperparameters used in training along with short descriptions of each. The structure of the deep neural network used for most of the training can be seen in Tables III and IV. The network consists of mostly convolutional two-dimensional layers of decreasing size. All layers except the first used zero padding. The first layer used padding with periodic boundary conditions. For grid size $d = 9$, we used the built-in ResNet34 definition provided in the PYTORCHframework. It has 21 277 955 tunable parameters.

The hardware used for the training was one GPU unit (NVIDIA Tesla V100 SMX2 GPU). The training time depends on the grid size. The bigger the grid, the more training is necessary. With the implementation found on github, $d = 5$ converged after 5 h of training. The network for $d = 7$ needs approximately 4 days (96 h) for convergence.

Table V shows execution time t_{step} per correction step for two different error rates of depolarizing noise. This is

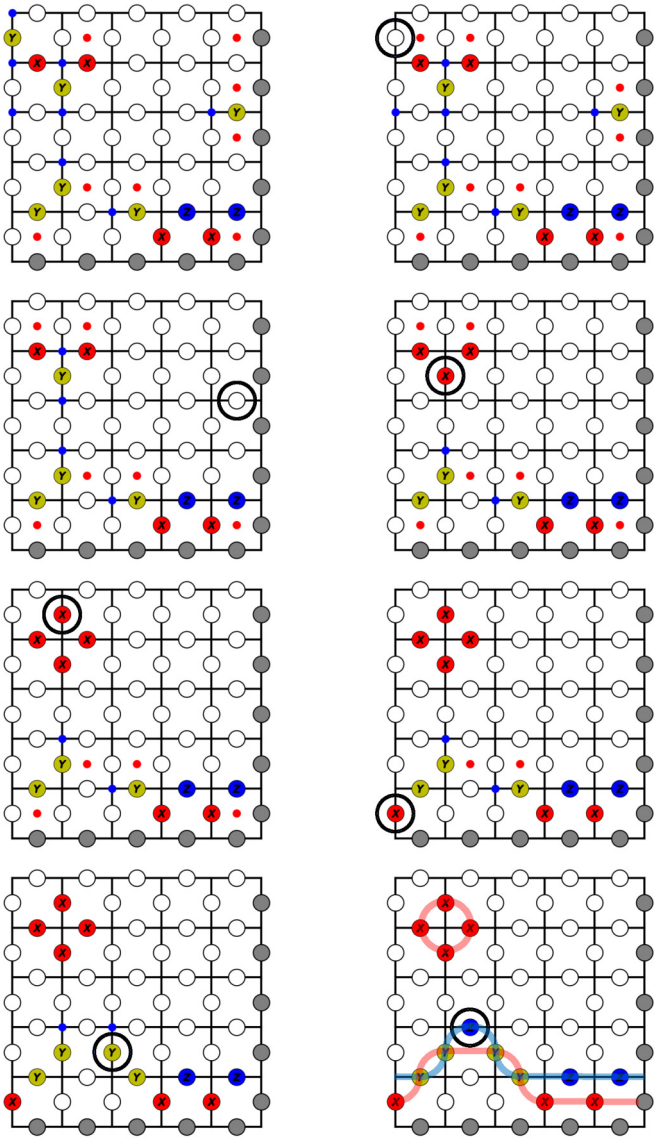

FIG. 17. A selected correction sequence from the fully trained decoder. The sequence goes from left to right and top to bottom. The circles indicate on which qubit an action was performed. In this case, the error correction fails, with the last state corresponding to a logical Y operator, i.e., both bit and and phase flip.

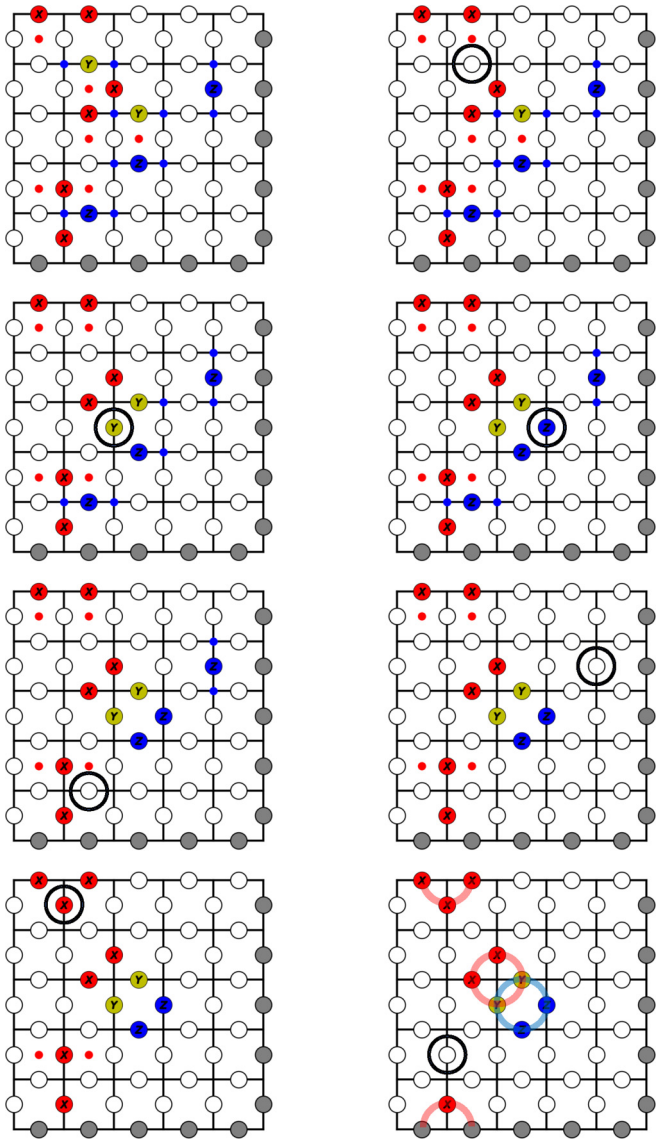

FIG. 18. A selected correction chain from the fully trained decoder. The sequence goes from left to right and top to bottom. The circles indicate on which qubit an action was performed. Here, the error correction is successful, with only trivial loops remaining.

calculated by taking the average time to correct 10 000 randomly generated syndromes divided by the average number of errors $2pd^2$. As expected, time per step depends only weakly on p, but much more strongly on d. The increase with code length is mainly due to the corresponding growing complexity of the networks, which increases the computational time required for the policy generating forward propagation through the network. To estimate how t_{step} scales with d is left for future work as it would require a careful study of the minimal network size and structure, more (even integer) and larger d, as well as optimizing the full computational structure.

APPENDIX C: SELECTED EPISODES

In this Appendix, we present two selected episodes of error correction using the fully trained decoder for $d = 5$. Figure 17 shows an example where the error correction fails and Fig. 18 shows an example of successful error correction.

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2000).

[2] A. Y. Kitaev, Fault-tolerant quantum computation by anyons, Ann. Phys. (NY) **303**, 2 (2003).

[3] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Topological quantum memory, J. Math. Phys. **43**, 4452 (2002).

[4] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Surface codes: Towards practical large-scale quantum computation, Phys. Rev. A **86**, 032324 (2012).

[5] B. M. Terhal, Quantum error correction for quantum memories, Rev. Mod. Phys. **87**, 307 (2015).

[6] M. D. Reed, L. DiCarlo, S. E. Nigg, L. Sun, L. Frunzio, S. M. Girvin, and R. J. Schoelkopf, Realization of three-qubit quantum error correction with superconducting circuits, Nature (London) **482**, 382 (2012).

[7] S. Shankar, M. Hatridge, Z. Leghtas, K. M. Sliwa, A. Narla, U. Vool, S. M. Girvin, L. Frunzio, M. Mirrahimi, and M. H. Devoret, Autonomously stabilized entanglement between two superconducting quantum bits, Nature (London) **504**, 419 (2013).

[8] D. Ristè, S. Poletto, M.-Z. Huang, A. Bruno, V. Vesterinen, O.-P. Saira, and L. DiCarlo, Detecting bit-flip errors in a logical qubit using stabilizer measurements, Nat. Commun. **6**, 6983 (2015).

[9] J. Kelly, R. Barends, A. G. Fowler, A. Megrant, E. Jeffrey, T. C. White, D. Sank, J. Y. Mutus, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, I.-C. Hoi, C. Neill, P. J. J. O'Malley, C. Quintana, P. Roushan, A. Vainsencher, J. Wenner *et al.*, State preservation by repetitive error detection in a superconducting quantum circuit, Nature (London) **519**, 66 (2015).

[10] A. D. Córcoles, E. Magesan, S. J. Srinivasan, A. W. Cross, M. Steffen, J. M. Gambetta, and J. M. Chow, Demonstration of a quantum error detection code using a square lattice of four superconducting qubits, Nat. Commun. **6**, 6979 (2015).

[11] N. Ofek, A. Petrenko, R. Heeres, P. Reinhold, Z. Leghtas, B. Vlastakis, Y. Liu, L. Frunzio, S. M. Girvin, L. Jiang, M. Mirrahimi, M. H. Devoret, and R. J. Schoelkopf, Extending the lifetime of a quantum bit with error correction in superconducting circuits, Nature (London) **536**, 441 (2016).

[12] M. Takita, A. W. Cross, A. D. Córcoles, J. M. Chow, and J. M. Gambetta, Experimental Demonstration of Fault-Tolerant State Preparation with Superconducting Qubits, Phys. Rev. Lett. **119**, 180501 (2017).

[13] A. F. Kockum and F. Nori, Quantum Bits with Josephson Junctions, in *Fundamentals and Frontiers of the Josephson Effect*, edited by F. Tafuri (Springer, Berlin, 2019), pp. 703–741.

[14] M. Gong, X. Yuan, S. Wang, Y. Wu, Y. Zhao, C. Zha, S. Li, Z. Zhang, Q. Zhao, Y. Liu, F. Liang, J. Lin, Y. Xu, H. Deng, H. Rong, H. Lu, S. C. Benjamin, C.-Z. Peng, X. Ma, Y.-A. Chen, X. Zhu, and J.-W. Pan, Experimental verification of five-qubit quantum error correction with superconducting qubits, arXiv:1907.04507.

[15] C. Kraglund Andersen, A. Remm, S. Lazar, S. Krinner, N. Lacroix, G. J. Norris, M. Gabureac, C. Eichler, and A. Wallraff, Repeated quantum error detection in a surface code, arXiv:1912.09410.

[16] J. Chiaverini, D. Leibfried, T. Schaetz, M. D. Barrett, R. B. Blakestad, J. Britton, W. M. Itano, J. D. Jost, E. Knill, C. Langer, R. Ozeri, and D. J. Wineland, Realization of quantum error correction, Nature (London) **432**, 602 (2004).

[17] P. Schindler, J. T. Barreiro, T. Monz, V. Nebendahl, D. Nigg, M. Chwalla, M. Hennrich, and R. Blatt, Experimental repetitive quantum error correction, Science **332**, 1059 (2011).

[18] B. P. Lanyon, P. Jurcevic, M. Zwerger, C. Hempel, E. A. Martinez, W. Dür, H. J. Briegel, R. Blatt, and C. F. Roos, Measurement-Based Quantum Computation with Trapped Ions, Phys. Rev. Lett. **111**, 210501 (2013).

[19] D. Nigg, M. Muller, E. A. Martinez, P. Schindler, M. Hennrich, T. Monz, M. A. Martin-Delgado, and R. Blatt, Quantum computations on a topologically encoded qubit, Science **345**, 302 (2014).

[20] N. M. Linke, M. Gutierrez, K. A. Landsman, C. Figgatt, S. Debnath, K. R. Brown, and C. Monroe, Fault-tolerant quantum error detection, Sci. Adv. **3**, e1701074 (2017).

[21] X.-C. Yao, T.-X. Wang, H.-Z. Chen, W.-B. Gao, A. G. Fowler, R. Raussendorf, Z.-B. Chen, N.-L. Liu, C.-Y. Lu, Y.-J. Deng, Y.-A. Chen, and J.-W. Pan, Experimental demonstration of topological error correction, Nature (London) **482**, 489 (2012).

[22] B. A. Bell, D. A. Herrera-Martí, M. S. Tame, D. Markham, W. J. Wadsworth, and J. G. Rarity, Experimental demonstration of a graph state quantum error-correction code, Nat. Commun. **5**, 3658 (2014).

[23] J. R. Wootton and D. Loss, High Threshold error Correction for the Surface Code, Phys. Rev. Lett. **109**, 160503 (2012).

[24] A. Hutter, J. R. Wootton, and D. Loss, Efficient markov chain monte carlo algorithm for the surface code, Phys. Rev. A **89**, 022326 (2014).

[25] M. Herold, E. T. Campbell, J. Eisert, and M. J. Kastoryano, Cellular-automaton decoders for topological quantum memories, npj Quantum Inf. **1**, 15010 (2015).

[26] A. Kubica and J. Preskill, Cellular-Automaton Decoders with Provable Thresholds for Topological Codes, Phys. Rev. Lett. **123**, 020501 (2019).

[27] G. Duclos-Cianci and D. Poulin, Fast Decoders for Topological Quantum Codes, Phys. Rev. Lett. **104**, 050504 (2010).

[28] G. Torlai and R. G. Melko, Neural Decoder for Topological Codes, Phys. Rev. Lett. **119**, 030501 (2017).

[29] S. Krastanov and L. Jiang, Deep neural network probabilistic decoder for stabilizer codes, Sci. Rep. **7**, 11003 (2017).

[30] S. Varsamopoulos, B. Criger, and K. Bertels, Decoding small surface codes with feedforward neural networks, Quantum Sci. Technol. **3**, 015004 (2017).

[31] P. Baireuther, T. E. O'Brien, B. Tarasinski, and C. W. Beenakker, Machine-learning-assisted correction of correlated qubit errors in a topological code, Quantum **2**, 48 (2018).

[32] N. P. Breuckmann and X. Ni, Scalable neural network decoders for higher dimensional quantum codes, Quantum **2**, 68 (2018).

[33] C. Chamberland and P. Ronagh, Deep neural decoders for near term fault-tolerant experiments, Quantum Sci. Technol. **3**, 044002 (2018).

[34] X. Ni, Neural network decoders for large-distance 2d toric codes, arXiv:1809.06640.

[35] R. Sweke, M. S. Kesselring, E. P. van Nieuwenburg, and J. Eisert, Reinforcement learning decoders for fault-tolerant quantum computation, arXiv:1810.07207.

[36] P. Andreasson, J. Johansson, S. Liljestrand, and M. Granath, Quantum error correction for the toric code using deep reinforcement learning, Quantum **3**, 183 (2019).

[37] H. P. Nautrup, N. Delfosse, V. Dunjko, H. J. Briegel, and N. Friis, Optimizing quantum error correction codes with reinforcement learning, Quantum **3**, 215 (2019).

[38] N. Maskara, A. Kubica, and T. Jochym-O'Connor, Advantages of versatile neural-network decoding for topological codes, Phys. Rev. A **99**, 052351 (2019).

[39] C. Chinni, A. Kulkarni, D. M. Pai, K. Mitra, and P. K. Sarvepalli, Neural decoder for topological codes using pseudo-inverse of parity check matrix, arXiv:1901.07535.

[40] L. D. Colomer, M. Skotiniotis, and R. Muñoz-Tapia, Reinforcement learning for optimal error correction of toric codes, Phys. Lett. A **384**, 126353 (2020).

[41] J. Edmonds, Paths, trees, and flowers, Can. J. Math. **17**, 449 (1965).

[42] A. G. Fowler, Optimal complexity correction of correlated errors in the surface code, arXiv:1310.0863.

[43] A. G. Fowler, Minimum weight perfect matching of fault-tolerant topological quantum error correction in average O(1) parallel time, Quantum Inf. Comput. **15**, 145 (2015), arXiv:1307.1740.

[44] S. Bravyi, M. Suchara, and A. Vargo, Efficient algorithms for maximum likelihood decoding in the surface code, Phys. Rev. A **90**, 032326 (2014).

[45] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, Nature (London) **521**, 436 (2015).

[46] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning* (MIT Press, Cambridge, MA, 2016), Vol. 1.

[47] G. Carleo and M. Troyer, Solving the quantum many-body problem with artificial neural networks, Science **355**, 602 (2017).

[48] J. Carrasquilla and R. G. Melko, Machine learning phases of matter, Nat. Phys. **13**, 431 (2017).

[49] E. P. Van Nieuwenburg, Y.-H. Liu, and S. D. Huber, Learning phase transitions by confusion, Nat. Phys. **13**, 435 (2017).

[50] J. Carrasquilla, Machine learning for quantum matter, arXiv:2003.11040.

[51] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, Playing atari with deep reinforcement learning, arXiv:1312.5602.

[52] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, Human-level control through deep reinforcement learning, Nature (London) **518**, 529 (2015).

[53] M. Bukov, A. G. R. Day, D. Sels, P. Weinberg, A. Polkovnikov, and P. Mehta, Reinforcement Learning in Different Phases of Quantum Control, Phys. Rev. X **8**, 031086 (2018).

[54] T. Fösel, P. Tighineanu, T. Weiss, and F. Marquardt, Reinforcement Learning with Neural Networks for Quantum Feedback, Phys. Rev. X **8**, 031084 (2018).

[55] The full decoding sequence for this syndrome using the DRL decoder is shown at https://github.com/mats-granath/toric-RL-decoder.

[56] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, MA, 2018).

[57] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, Prioritized experience replay, arXiv:1511.05952.

[58] H. Bombin, R. S. Andrist, M. Ohzeki, H. G. Katzgraber, and M. A. Martin-Delgado, Strong Resilience of Topological Codes to Depolarization, Phys. Rev. X **2**, 021004 (2012).

[59] J. Ghosh, A. G. Fowler, and M. R. Geller, Surface code with decoherence: An analysis of three superconducting architectures, Phys. Rev. A **86**, 062318 (2012).

[60] F. Yan, S. Gustavsson, A. Kamal, J. Birenbaum, A. P. Sears, D. Hover, T. J. Gudmundsen, D. Rosenberg, G. Samach, S. Weber, J. L. Yoder, T. P. Orlando, J. Clarke, A. J. Kerman, and W. D. Oliver, The flux qubit revisited to enhance coherence and reproducibility, Nat. Commun. **7**, 12964 (2016).

[61] X. Gu, A. F. Kockum, A. Miranowicz, Y.-x. Liu, and F. Nori, Microwave photonics with superconducting quantum circuits, Phys. Rep. **718–719**, 1 (2017).

[62] P. V. Klimov, J. Kelly, Z. Chen, M. Neeley, A. Megrant, B. Burkett, R. Barends, K. Arya, B. Chiaro, Y. Chen, A. Dunsworth, A. Fowler, B. Foxen, C. Gidney, M. Giustina, R. Graff, T. Huang, E. Jeffrey, E. Lucero, J. Y. Mutus *et al.*, Fluctuations of Energy-Relaxation Times in Superconducting Qubits, Phys. Rev. Lett. **121**, 090502 (2018).

[63] J. J. Burnett, A. Bengtsson, M. Scigliuzzo, D. Niepce, M. Kudra, P. Delsing, and J. Bylander, Decoherence benchmarking of superconducting qubits, npj Quantum Inf. **5**, 54 (2019).

[64] Y. Lu, A. Bengtsson, J. J. Burnett, E. Wiegand, B. Suri, P. Krantz, A. F. Roudsari, A. F. Kockum, S. Gasparinetti, G. Johansson, and P. Delsing, Characterizing decoherence rates of a superconducting qubit by direct microwave scattering, arXiv:1912.02124.

[65] A. Valenti, E. van Nieuwenburg, S. Huber, and E. Greplova, Hamiltonian learning for quantum error correction, Phys. Rev. Res. **1**, 033092 (2019).

[66] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. Van Hasselt, and D. Silver, Distributed prioritized experience replay, in 6th International Conference on Learning Representations, ICLR 2018 Conference Track Proceedings, arXiv:1803.00933.

[67] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, Piscataway, NJ, 2016), pp. 770–778.

[68] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, Mastering the game of go without human knowledge, Nature (London) **550**, 354 (2017).

[69] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play, Science **362**, 1140 (2018).

[70] M. Dalgaard, F. Motzoi, J. J. Sorensen, and J. Sherson, Global optimization of quantum dynamics with AlphaZero deep exploration, NPJ Quantum Info. **6**, 6 (2020).

[71] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, in *Advances in Neural Information Processing Systems* (MIT Press, Cambridge, MA, 2000), pp. 1057–1063.

[72] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee, Natural actor-critic algorithms, Automatica **45**, 2471 (2009).

[73] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, A comprehensive survey on transfer learning, arXiv:1911.02685.