Reinforcement Learning with Neural Networks for Quantum Feedback

Thomas Fösel, Petru Tighineanu, and Talitha Weiss
Max Planck Institute for the Science of Light, Staudtstraße 2, 91058 Erlangen, Germany

Florian Marquardt
Max Planck Institute for the Science of Light, Staudtstraße 2, 91058 Erlangen, Germany and Physics Department, University of Erlangen-Nuremberg, Staudtstraße 5, 91058 Erlangen, Germany

(Received 23 April 2018; revised manuscript received 12 June 2018; published 27 September 2018)

Machine learning with artificial neural networks is revolutionizing science. The most advanced challenges require discovering answers autonomously. In the domain of reinforcement learning, control strategies are improved according to a reward function. The power of neural-network-based reinforcement learning has been highlighted by spectacular recent successes such as playing Go, but its benefits for physics are yet to be demonstrated. Here, we show how a network-based "agent" can discover complete quantum-error-correction strategies, protecting a collection of qubits against noise. These strategies require feedback adapted to measurement outcomes. Finding them from scratch without human guidance and tailored to different hardware resources is a formidable challenge due to the combinatorially large search space. To solve this challenge, we develop two ideas: two-stage learning with teacher and student networks and a reward quantifying the capability to recover the quantum information stored in a multiqubit system. Beyond its immediate impact on quantum computation, our work more generally demonstrates the promise of neural-network-based reinforcement learning in physics.

DOI: 10.1103/PhysRevX.8.031084 Subject Areas: Computational Physics, Interdisciplinary Physics, Quantum Information

I. INTRODUCTION

We are witnessing rapid progress in applications of artificial neural networks (ANNs) for tasks like image classification, speech recognition, natural language processing, and many others [1,2]. Within physics, the examples emerging during the past two years range across areas like statistical physics, quantum many-body systems, and quantum-error correction [3–11]. To date, most applications of neural networks employ supervised learning, where a large collection of samples has to be provided together with the correct labeling.

However, inspired by the long-term vision of artificial scientific discovery [12,13], one is led to search for more powerful techniques that explore solutions to a given task autonomously. Reinforcement learning (RL) is a general approach of this kind [2], where an "agent" interacts with an "environment." The agent's "policy," i.e., the choice of actions in response to the environment's evolution, is updated to increase some reward. The power of this method, when combined with ANNs, was demonstrated convincingly through learning to play games beyond human expertise [14,15]. In physics, RL *without* neural networks has been introduced recently, for example, to study qubit control [16,17] and invent quantum-optics experiments [18]. Moving to neural-network-based RL promises access to the vast variety of techniques currently being developed for ANNs.

In this paper, we introduce network-based RL in physics (Fig. 1) and illustrate its versatility in the domain of quantum feedback. Specifically, we devise a unified, fully autonomous, human-guidance-free approach for discovering quantum-error-correction (QEC) strategies from scratch in few-qubit quantum systems subject to arbitrary noise and hardware constraints. This approach relies on a network agent that learns feedback strategies, adapting its actions to measurement results. As illustrated in Figs. 1(b)–1(d), our method provides a unified approach to protect a quantum memory from noise. It covers a wide range of scenarios where one would otherwise have to select an existing scheme (stabilizer codes, adaptive phase estimation, etc.) and adapt it to the given situation. Our findings are of immediate relevance to the broad field of quantum-error correction (including quantum-error-mitigation techniques) and are best suited to be used in few-qubit quantum modules.

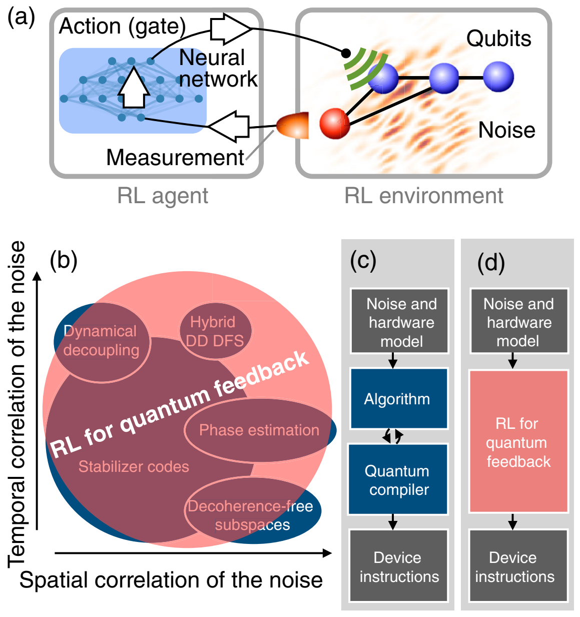

FIG. 1. (a) The general setting of this work: A few-qubit quantum device with a neural-network-based controller whose task is to protect the quantum memory residing in this device against noise. Reinforcement learning lets the controller ("RL agent") discover on its own how to best choose gate sequences, perform measurements, and react to measurement results by interacting with the quantum device ("RL environment"). (b) visualizes the flexibility of our approach (schematic). Depending on the type of noise and hardware setting, different approaches are optimal (DD, dynamical decoupling; DFS, decoherence-free subspace). By contrast, the RL approach is designed to automatically discover the best strategy adapted to the situation. In (c), we show the conventional procedure to select some QEC algorithm and then produce hardware-adapted device instructions (possibly reiterating until an optimal choice is found). We compare this to our approach (d) that takes care of all these steps at once and provides QEC strategies fully adapted to the concrete specifications of the quantum device.

These modules could be used as stand-alone quantum memory or be part of the modular approach to quantum computation, which has been suggested for several leading hardware platforms [19,20].

Given a collection of qubits and a set of available quantum gates, the agent is asked to preserve an arbitrary quantum state $\alpha|0\rangle + \beta|1\rangle$ initially stored in one of the qubits. It finds complex sequences including projective measurements and entangling gates, thereby protecting the quantum information stored in such a few-qubit system against decoherence. This challenge is very complex, where both brute-force searches and even the most straightforward RL approaches fail. The success of our approach is due to a combination of two key ideas: (i) two-stage learning with a RL-trained network receiving maximum input acting as a teacher for a second network and (ii) a measure of the recoverable quantum information hidden inside a collection of qubits being used as a reward.

Recent progress in multiqubit quantum devices [21–30] has highlighted hardware features deviating from often-assumed idealized scenarios. These features include qubit connectivity, correlated noise, restrictions on measurements, or inhomogeneous error rates. Our approach can help in finding "hardware-adapted" solutions, since it builds on the main advantage of RL, namely, its flexibility: It can discover strategies for such a wide range of situations with minimal domain-specific input. We illustrate this flexibility in examples from two different domains: In one set of examples (uncorrelated bit-flip noise), the network is able to go beyond rediscovering the textbook stabilizer repetition code. It finds an adaptive response to unexpected measurement results that allows it to increase the coherence time, performing better than any straightforward nonadaptive implementation. Simultaneously, it automatically discovers suitable gate sequences for various types of hardware settings. In another very different example, the agent learns to counter spatially correlated noise by finding nontrivial adaptive phase-estimation strategies that quickly become intractable by conventional numerical approaches such as brute-force search. Crucially, all these examples can be treated by exactly the same approach with no fine-tuning. The only input consists in the problem specification (hardware and noise model).

In a nutshell, our goal is to have a neural network which can be employed in an experiment, receiving measurement results and selecting suitable subsequent gate operations conditioned on these results. However, in our two-stage learning approach, we do not directly train this neural network from scratch. Rather, we first employ reinforcement learning to train an auxiliary network that has full knowledge of the simulated quantum evolution. Later on, the experimentally applicable network is trained in a supervised way to mimic the behavior of this auxiliary network.

We emphasize that feedback requires reaction towards the observations, going beyond optimal-control type challenges (like pulse-shape optimization or dynamical decoupling), and RL has been designed for exactly this purpose. Specifically, in this work, we consider discrete-time digital feedback of the type that is now starting to be implemented experimentally [31–35], e.g., for error correction in superconducting quantum computers. Other widespread optimization techniques for quantum control, like gradient-ascent pulse engineering (GRAPE), often vary evolution operators with respect to continuous parameters [36,37] but do not easily include feedback and are most suited for optimizing the pulse shapes of individual gates (rather than complex gate sequences acting on many qubits). Another recent approach [38] to quantum-error correction uses the optimization of control parameters in a preconfigured gate sequence. By contrast, RL directly explores the space of discrete gate sequences. Moreover, it is a "model-free" approach [2]; i.e., it does not rely on access to the underlying dynamics. What is optimized is the network agent. Neural-network-based RL promises to complement other successful machine-learning techniques applied to quantum control [39–42].

Conceptually, our approach aims to control a quantum system using a classical neural network. To avoid confusion, we emphasize that our approach is distinct from future “quantum machine learning” devices, where even the network will be quantum [8,43,44].

II. REINFORCEMENT LEARNING

The purpose of RL [Fig. 1(a)] is to find an optimal set of actions (in our case, quantum gates and measurements) that an agent can perform in response to the changing state of an environment (here, the quantum memory). The objective is to maximize the expected return R, i.e., a sum of rewards.

To find optimal gate sequences, we employ a widespread version of reinforcement learning [45,46] where discrete actions are selected at each time step t according to a probabilistic policy π_θ. Here, $\pi_\theta(a_t|s_t)$ is the probability to apply action a_t, given the state s_t of the RL environment. As we use a neural network to compute π_θ, the multidimensional parameter θ stands for all the network's weights and biases. The network is fed s_t as an input vector and outputs the probabilities $\pi_\theta(a_t|s_t)$. The expected return can then be maximized by applying the policy gradient RL update rule [45]

$$\delta\theta_j = \eta \frac{\partial \mathbb{E}[R]}{\partial \theta_j} = \eta \mathbb{E}\left[R \sum_t \frac{\partial}{\partial \theta_j} \ln \pi_\theta(a_t|s_t)\right], \qquad (1)$$

with η the learning rate parameter and $\mathbb{E}$ the expectation value over all gate sequences and measurement outcomes. These ingredients summarize the basic policy gradient approach. In practice, improvements of Eq. (1) are used; for example, we employ a baseline, natural policy gradient, and entropy regularization (see Appendix H). Even so, several further conceptual steps are essential to have any chance of success (see below).

Equation (1) provides the standard recipe for a fully observed environment. This approach can be extended to a partially observed environment, where the policy will then be a function of the observations only instead of the state. The observations contain partial information on the actual state of the environment. In the present manuscript, we encounter both cases.

III. REINFORCEMENT LEARNING APPROACH TO QUANTUM MEMORY

In this work, we seek to train a neural network to develop strategies to protect the quantum information stored in a quantum memory from decoherence. This involves both variants of stabilizer-code-based QEC [47–49] as well as other more specialized (but, in their respective domain, more resource efficient) approaches like decoherence-free subspaces or phase estimation. We remind the reader that for the particular case of stabilizer-code-based QEC, the typical steps are (i) the encoding, in which the logical state initially stored in one qubit is distributed over several physical qubits, (ii) the detection of errors via measurement of suitable multiqubit operators (syndromes), (iii) the subsequent correction, and (iv) the decoding procedure that transfers the encoded state back into one physical qubit. We stress that no such specialized knowledge is provided *a priori* to our network, thus, retaining maximum flexibility in the tasks it might be applied to and in the strategies it can encompass [Fig. 1(b)].

We start by storing an arbitrary quantum state $\alpha|0\rangle + \beta|1\rangle$ inside one physical qubit. The goal is to be able to retrieve this state with optimum fidelity after a given time span. Given hardware constraints such as the connectivity between qubits, the network agent must develop an efficient QEC strategy from scratch solely by interacting with the quantum memory at every time step via a set of unitary gates (such as controlled NOT gates and bit flips) and measurements. They are chosen according to the available hardware and define the action set of the agent. Importantly, the network must react and adapt its strategy to the binary measurement results, providing real-time quantum feedback.

This particular task seems practically unsolvable for the present reinforcement-learning techniques if no extra precautions are taken. The basic challenge is also encountered in other difficult RL applications: The first sequence leading to an increased return is rather long. In our scenarios, the probability to randomly select a good sequence is much less than 10^{-12}. Moreover, any subsequence may be worse than the trivial (idle) strategy: For example, performing an incomplete encoding sequence (ending up in a fragile entangled state) can accelerate decay. Adopting the straightforward return, namely, the overlap of the final and initial states, both the trivial strategy and the error-correction strategy are fixed points. These are separated by a wide barrier—all the intermediate-length sequences with lower return. In our numerical experiments, naive RL was not successful, except for some tasks with very few qubits and gates.

We introduce two key concepts to solve this challenge: a two-stage learning approach with one network acting as teacher of another and a measure of the “recoverable quantum information” retained in any quantum memory.

Before we address these concepts, we mention that from a machine-learning point of view there is another unconventional aspect: Instead of sampling initial states of the RL environment stochastically, we consider the evolution under the influence of the agent's actions for *all* possible states simultaneously. This is required because the quantum memory has to preserve arbitrary input states. Our reward is based on the completely positive map describing the dissipative quantum evolution of arbitrary states. The only

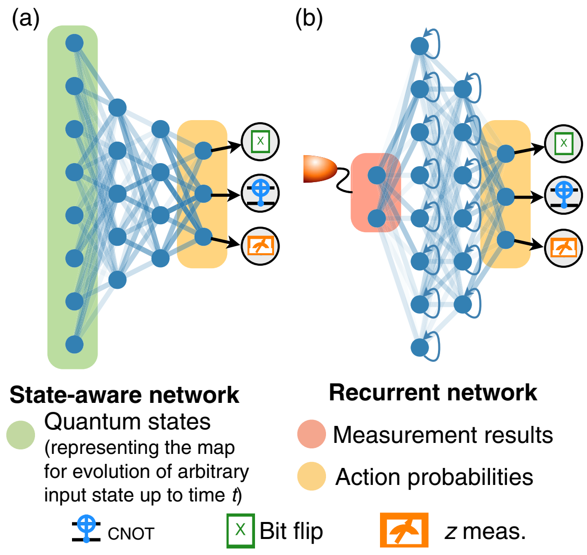

FIG. 2. The neural networks. (a) At each time t, the "state-aware" network receives a representation of the map Φ describing the quantum evolution of arbitrary initial logical qubit states up to that time (represented by four evolved states $\hat{\rho}$; see main text). It outputs the probabilities for the different actions (gates) defining the agent's policy. Like any neural network, it is a nonlinear function that can be decomposed into layerwise linear superposition of neuron values using trainable weights (visualized by connections) and the application of nonlinear activation functions. Examples for actions are shown here (bit flip, CNOT, measurement). Each of those can be applied to various qubits (or qubit pairs), resulting in approximately 10–20 actions. (b) The recurrent network receives the most recent measurement result (if any) and also outputs the action probabilities. Its long short-term memory (LSTM) neurons learn to keep track of information accumulated in previous time steps (schematically indicated by the recurrent connections here).

statistical averaging necessary is over measurement outcomes and the probabilistic action choices. Further below, we comment on how the time evolution is implemented in practice.

As known from other RL applications (like board games [15]), it helps to provide as much information as possible to the network. In our case, this could mean providing the multiqubit quantum state at each time step. However, that information is not available in a real experiment. In order to solve this dilemma, we train two different networks in succession [Figs. 2(a) and 2(b)]: The first network is fully *state aware*. Later on, we use it as a teacher for the second network, which essentially gets only the measurement results as an input (plus the information on which gate or measurement has been applied in the previous step). This approach splits the problem into two subproblems that are easier to solve. In this approach, the main remaining challenge is to train the state-aware network, while the supervised training of the second network is fairly straightforward in our experience. In contrast, directly training the second network via RL would be tremendously harder, if not impossible, because the input would be significantly less comprehensive than the completely positive map.

At this point, we see that evolving all initial states simultaneously is not only more efficient but even required to prevent the state-aware network from "cheating." Otherwise, it might simply memorize the initial state, wait for it to relax, and then reconstruct it, which, of course, is not a valid strategy to preserve a principally *unknown* quantum state. Such a behavior is avoided when the network is asked to preserve all possible logical qubit states with the same gate sequence. It turns out that this can be implemented efficiently by evolving just four initial quantum states $\hat{\rho}$ (for a single logical qubit); tracking their evolution fully characterizes, at any point in time, the completely positive map Φ of the multiqubit system that maps $\hat{\rho}(0)$ to $\hat{\rho}(t)$. Moreover, we find it useful to apply principal component analysis, i.e., to feed only the few largest-weight eigenvectors of the evolved $\hat{\rho}$'s as input to the network (see Appendix F).

We are now ready to define our problem fully from the point of view of reinforcement learning. The state space of the RL environment is the space of completely positive maps. This information is not accessible in a real-world experiment where the measurements provide partial information about the RL environment. This reinforcement-learning problem is therefore classified as a partially observed Markov process. This is what is considered in our second learning stage, and our method to solve it relies on a recurrent network. In the modified input scheme of the first learning stage, the agent observes the full state space, and we therefore deal with a fully observed Markov process. In both cases, the RL environment is stochastic due to the measurements. As we describe above, the action set is defined by the available hardware instructions (unitary gates and measurements).

Two-stage learning with parallel evolution is essential but not yet sufficient for our challenge. We now introduce a suitable reward that indicates the likely final success of an action sequence ahead of time. In our case, we follow the intuitive idea that this reward should quantify whether the original quantum information survives in the complex entangled many-qubit state that results after application of unitary gates and measurements and with the system subject to decoherence.

We note that in the ideal case without decoherence, two initially orthogonal qubit states are always mapped onto orthogonal states. Therefore, they remain 100% distinguishable, and the original state can always be restored. With a suitable encoding, this remains true even after some errors have happened if a suitable error-detection and decoding sequence is applied ("recovery"). By contrast, irreversible loss of quantum information means that perfect recovery becomes impossible. In order to make these notions concrete, we start from the well-known fact that the probability to distinguish two quantum states $\hat{\rho}_1$ and $\hat{\rho}_2$

by optimal measurements is given by the trace distance $\frac{1}{2}\|\hat{\rho}_1 - \hat{\rho}_2\|_1$. Let $\hat{\rho}_{\vec{n}}(t)$ be the quantum state into which the multiqubit system has evolved, given the initial logical qubit state of Bloch vector $\vec{n}$. We now consider the distinguishability of two initially orthogonal states, $\frac{1}{2}\|\hat{\rho}_{\vec{n}}(t) - \hat{\rho}_{-\vec{n}}(t)\|_1$. In general, this quantity may display a nontrivial, nonanalytic dependence on $\vec{n}$. We introduce the "*recoverable quantum information*" as

$$\mathcal{R}_Q(t) = \frac{1}{2}\min_{\vec{n}} \|\hat{\rho}_{\vec{n}}(t) - \hat{\rho}_{-\vec{n}}(t)\|_1. \tag{2}$$

The minimum over the full Bloch sphere is taken because the logical qubit state is unknown to the agent, so the success of an action sequence is determined by the worst-case scenario. In other words, $\mathcal{R}_Q$ specifies a guaranteed value for the remaining distinguishability for all possible logical qubit states. Thus, $\mathcal{R}_Q$ is a property of the completely positive map that characterizes the dissipative evolution.

The recoverable quantum information $\mathcal{R}_Q$ is much more powerful than the overlap of initial and final states, as it can be used to construct an *immediate* reward, evaluating a strategy even at intermediate times. In the idealized case where errors have occurred but they could, *in principle*, be perfectly recovered by a suitable detection and decoding sequence, $\mathcal{R}_Q$ remains 1. As we see below, this behavior steers the network towards suitable strategies. $\mathcal{R}_Q$ can be extended towards multiple logical qubits.

As far as $\mathcal{R}_Q$ is concerned, error-correction steps are required only to prevent the multiqubit system from venturing into regions of the Hilbert space where any further decoherence process will irreversibly destroy the quantum information (and lower $\mathcal{R}_Q$). If one wants the network to actually implement the final decoding sequence to return back an unentangled state, this can be done by adding suitable contributions to the reward (see below).

IV. RESULTS

We now apply the general approach to different settings, illustrating its flexibility. The training of the state-aware network is analyzed in Fig. 3. In the example, the qubits are subject to bit-flip errors uncorrelated in space and time, with a decay term $\dot{\hat{\rho}} = T_{\mathrm{dec}}^{-1}\sum_j(\hat{\sigma}_x^{(j)}\hat{\rho}\hat{\sigma}_x^{(j)} - \hat{\rho})$ in the underlying master equation (see Appendix B). All of the four qubits may be measured, and there is full connectivity. During training [Figs. 3(a) and 3(b)], the network first learns to avoid destructive measurements which reveal the logical qubit state. Afterwards, it discovers a sequence of CNOT gates that creates an entangled state, implementing some version of the three-qubit repetition code [48,49]. The particular CNOT sequence shown in the figure generates one possible encoded state out of several equally good ones. The symmetry between these alternative encodings is broken spontaneously during training. The encoding already increases the reward above the trivial level (obtained

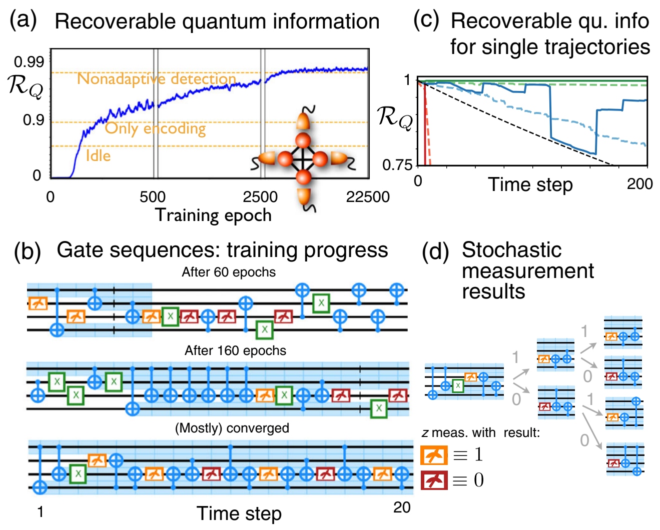

FIG. 3. Reinforcement learning for the state-aware network with a four-qubit setup (all qubits can be measured, as indicated by the detectors and the red color; all qubit pairs can be subject to CNOT gates, as indicated by the links). (a) Training progress in terms of the average recoverable quantum information $\mathcal{R}_Q$ evaluated at the final step of a 200-step gate sequence. One "epoch" involves training on a batch of 64 trajectories (gate sequences). Eventually, the network performs even better than a combination of encoding and periodic parity checks due to the adaptive recovery sequences that are triggered by unexpected measurements (i.e., upon error detection). In this example, that leads to an approximate 15% increase of the decoherence time over a nonadaptive scheme. (b) Typical gate sequences at different training stages, mostly converged strategy at bottom. Qubits participating in encoding (holding information about the logical qubit state) are indicated with light blue background. (c) Time evolution of $\mathcal{R}_Q$ at the same three training stages (red, blue, green) for individual trajectories (dashed: averaged over many trajectories). Jumps are due to measurements. (d) Evolution depending on stochastic measurement results indicated as "0" or "1" and also via the color (red or orange) of the measurement gate symbol. The policy strongly deviates between the different branches, demonstrating that RL finds adaptive quantum-feedback strategies, where the behavior depends on the measurement outcomes. Note that even the mostly converged strategy is still probabilistic to some degree.

for storing the logical qubit in one physical qubit only). Finally, the network starts doing repeated parity measurements of the type CNOT $(B \mapsto A)$, CNOT $(C \mapsto A)$, $M(A)$, flipping the state of ancilla A only if the states of B and C differ (here, M is a measurement). This protocol implements error detection, helping to preserve the quantum information by preventing the leakage into states with two bit flips that cannot be corrected if undetected. Figure 3(b) illustrates the progression from random quantum circuits to a nearly converged strategy. During any single trajectory, the recoverable quantum information can have sudden jumps when measurements are performed [Fig. 3(c)] with collapses and revivals (see also Sec. II of the Supplementary Material [51]).

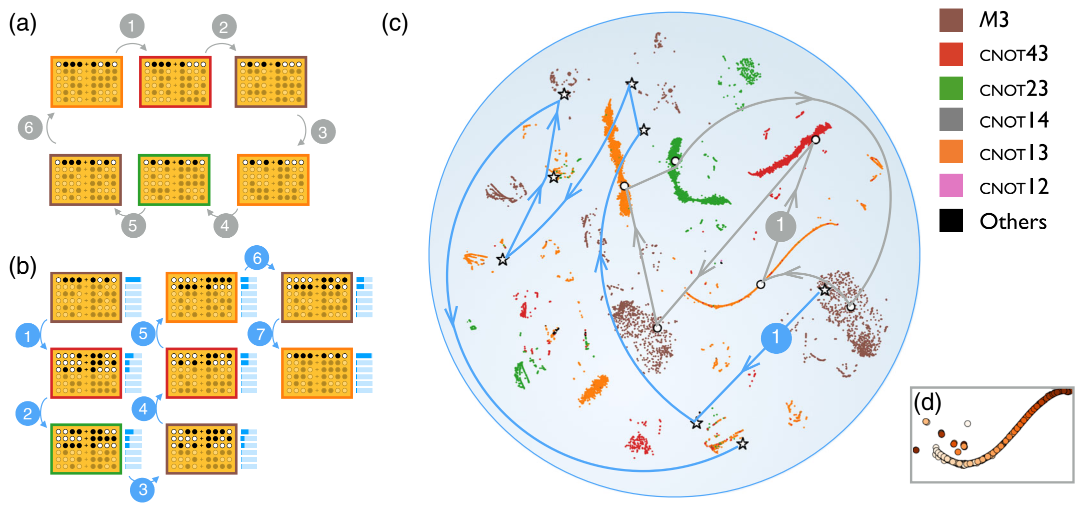

FIG. 4. Visualizing the operation of the state-aware network. (Same scenario as in Fig. 3) (a) Sequence of quantum states visited during a standard repetitive detection cycle (after encoding) displayed for an initial logical qubit state $|{+x}\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ in the absence of unexpected measurement outcomes (no errors). Each state $\hat{\rho}$ is represented by its decomposition into eigenstates, where, e.g., ○●●● + ●○●○ ≡ $|0111\rangle + |1010\rangle$. The eigenstates are sorted according to decreasing eigenvalues (probabilities). Eigenstates of less than 5% weight are displayed semitransparently. In this particular example, each eigenstate is a superposition of two basis states in the z basis. (b) Gate sequence triggered upon encountering an unexpected measurement. Again, we indicate the states $\hat{\rho}$ with bars now showing the probabilities. By further measurements, this sequence tries to disambiguate the error. (c) Visualization of neuron activations (300 neurons in the last hidden layer), in response to the quantum states (more precisely, maps Φ) encountered in many runs. These activations are projected down to 2D using the t-SNE technique [50], which is a nonlinear mapping that tries to preserve neighborhood relations while reducing the dimensionality. Each of the 2×10^5 points corresponds to one activation pattern and is colored according to the action taken. The sequences of (a) and (b) are indicated by arrows, with the sequence (b) clearly proceeding outside the dominant clusters (which belong to the more typically encountered states). Qubits are numbered 1,2,3,4, and gate CNOT13 has control qubit 1 and target 3. (d) The enlargement shows a set of states in a single cluster which is revisited periodically during the standard detection cycle; this means we stroboscopically observe the time evolution. The shading indicates the time progressing during the gate sequence with a slow drift of the state due to decoherence. (See the Supplemental Material [51] for an extended discussion.)

Can we understand better how the network operates? To this end, we visualize the responses of the network responses to the input states [Fig. 4(c)], projecting the high-dimensional neuron activation patterns into the 2D plane using the t-distributed stochastic neighbor embedding (t-SNE) technique [50]. Similar activation patterns are mapped close to each other, forming clearly visible clusters, each of which results in one type of action. During a gate sequence, the network visits states in different clusters. The sequence becomes complex if unexpected measurement results are encountered [Fig. 4(b)]. In the example shown here, the outcome of the first parity measurement is compatible with three possibilities (one of two qubits is flipped, or the ancilla state is erroneous). The network learns to resolve the ambiguity through two further measurements, returning to the usual detection cycle. It is remarkable that RL finds these nontrivial sequences (which would be complicated to construct *ab initio*), picking out reward differences of a few percent.

The flexibility of the approach is demonstrated by training on different setups, where the network discovers from scratch other feedback strategies [Fig. 5(a)] adapted to the available resources. For example, we consider a chain of qubits where CNOT gates are available only between nearest neighbours, and in addition, we fix a single measurement location. Then, the network learns that it may use the available CNOT gates to swap through the chain. However, if every qubit can be measured, the net discovers a better strategy with fewer gates, where the middle two qubits of the chain alternate in playing the role of the ancilla. We also show, specifically, the complex recovery sequences triggered by unexpected measurements. They are *a priori* unknown, and RL permits their discovery from scratch without extra input. Generally, additional resources (such as enhanced connectivity) are exploited to yield better improvement of the decoherence time [Fig. 5(b)]. In another scenario [Fig. 5(c)], we find that the network successfully learns to adapt to unreliable measurements by redundancy.

In a separate class of scenarios, we consider dephasing of a qubit by a fluctuating field (Fig. 6). If the field is spatially homogeneous and also couples to nearby ancilla qubits, then the dephasing is collective: $\hat{H}(t) = B(t)\sum_j \mu_j \hat{\sigma}_z^{(j)}$, where $B(t)$ is white noise and μ_j are the coupling strengths (to qubit and ancillas). Note that in this situation, one can use neither dynamical decoupling (since the noise is uncorrelated in time) nor decoherence-free subspaces (since the μ_j can be arbitrary, in general). However, the

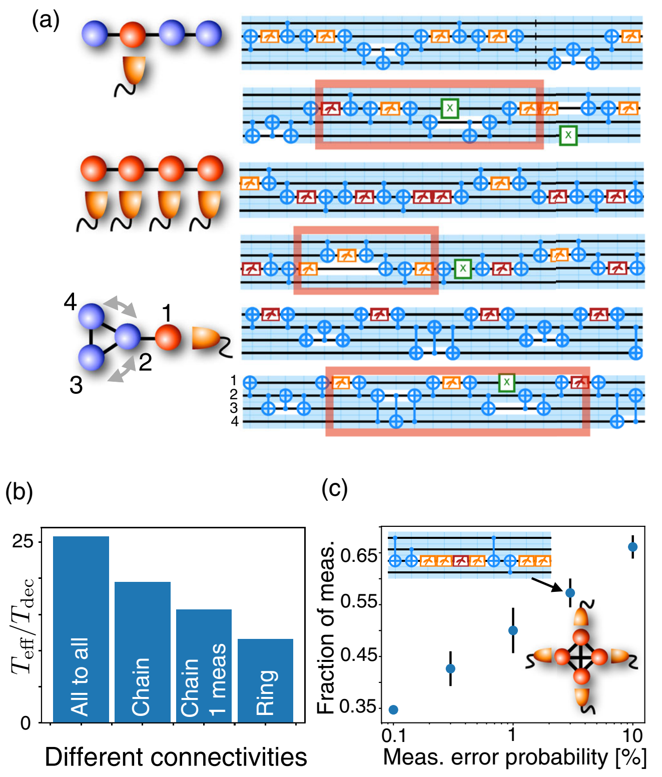

FIG. 5. (a) Scenarios with different qubit connectivity. CNOT gates are allowed only between qubits connected by a line. In each case, we display the "standard" gate sequence discovered by the net, as well as a sequence involving corrective actions (triggered by an unexpected measurement). From top to bottom: chain with fixed measurement location, chain with arbitrary measurements, ring connected to an ancilla. The red rectangle highlights an interval during which the quantum state is not dominated by a single component, indicating that the precise location of the error still must be pinpointed. These nontrivial detection and recovery sequences are considerably more complex than the periodic detection cycle. (b) The effective enhancement of the decoherence time via error correction for the different scenarios. Here, $T_{\text{dec}} = 1200$ is the single-qubit decoherence time (in units of the gate time that defines the time step), and T_{eff} is extracted from the decay of $\mathcal{R}_Q$ after 200 time steps. The differences can be traced back to the lengths of the detection cycles. (c) Behavior as a function of the measurement error. The network discovers that redundancy is needed; i.e., the number of measurements in the gate sequences increases (in this plot, from one per cycle to about six).

same RL program used for the examples above also finds solutions here (Fig. 6) without any input specific to the situation (except the available gates). It discovers that the field fluctuations can be tracked and corrected (to some extent) by observing the evolution of the nearby ancillas, measuring them in suitable time intervals. For more than one ancilla, the network discovers a strategy that is adaptive: The choice of measurement basis depends on the history of previous observations. Brute-force searches in this setting become quickly impossible due to the double-exponential growth of possibilities. The computational effort involved in such a brute-force approach is analyzed in detail in the Supplemental Material [51].

Up to now, the network only encodes and keeps track of errors by suitable collective measurements. By revising the reward structure, we can force it to correct errors and finally decode the quantum information back into a single physical qubit. Our objective is to maximize the overlap between the initial and final states for any logical qubit state (see Appendix E). Moreover, we find that learning the decoding during the final time steps is reinforced by punishing states where the logical qubit information is still distributed over multiple physical qubits. The corresponding rewards are added to the previous reward based on the recoverable quantum information. The network now indeed learns to decode properly [Fig. 7(a)]. In addition, it corrects errors. It does so typically soon after detecting an error, instead of at the end of the gate sequence. We conjecture that this is because it tries to return as soon as possible back to the known, familiar encoded state. For the same reason, error correction sometimes even happens without an explicit reward.

So far, we have trained the state-aware network. However, this cannot yet be applied to an experiment, where the quantum state is inaccessible to us. For this purpose, we need a network whose only input consists of the measurement results (and the selected gates, since the policy is probabilistic), requiring some sort of memory. An elegant solution consists of a *recurrent* neural network. We use the widespread LSTM approach [52].

Once the first, state-aware network is trained successfully, it is used as a teacher in supervised learning to train the second, recurrent network [Fig. 7(b)]. This network could then be applied as a controller to experimental runs, deciding on gate sequences depending on measurements. It might also be refined by RL, e.g., to adapt to changes in the parameters (decoherence rates, etc.). If the state-aware network is taught to properly restore the original state at the end of the time evolution, including corrections of possible errors, the second network needs to acquire a long memory time since it has to consider measurement results distributed in time and deduce the proper corrective actions.

We train the recurrent network based on a fully converged state-aware network. Inspecting the LSTM neuron activations [Fig. 7(c)], we see that different neurons activate for different events and some clearly display prolonged memory (remaining active during certain time intervals relevant for the strategy). For example, one neuron switches on during the recovery sequence after an unexpected measurement, while another seems like an internal counter operating during the periodic detection sequence.

We now come back to the statement in the Introduction that our approach is fully autonomous and can be applied to a broad range of problems with small human effort. In all the preceding examples, and also in general, the only human input to our approach is the problem specification, primarily the noise model (specifying the dissipative time

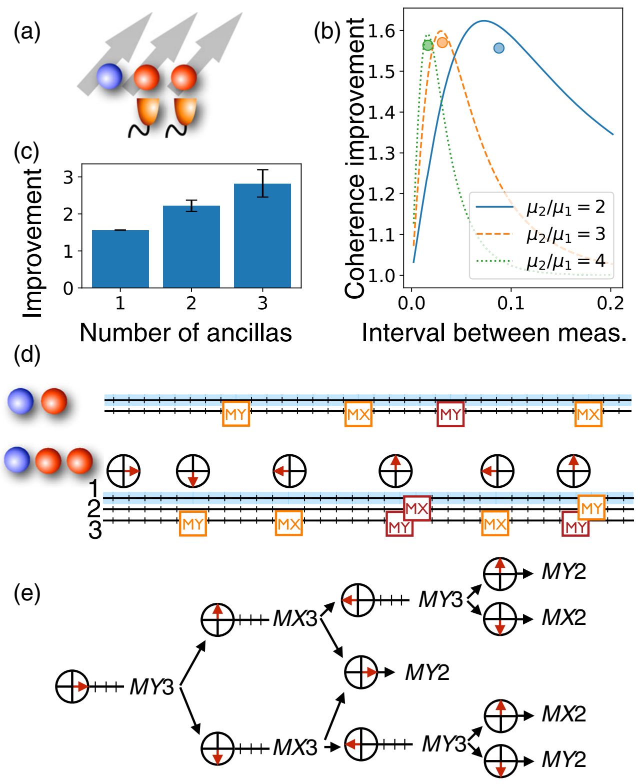

FIG. 6. Countering dephasing by measurements (adaptive phase estimation). (a) The setting: A data qubit whose phase undergoes a random walk due to a fluctuating field. Since the field is spatially correlated, its fluctuations can be detected by measuring the evolution of nearby ancilla qubits, which can then be exploited for correction. In this example, the allowed actions are measurements along x, y (MX, MY), and the idle operation. (b) For one ancilla, the network performs measurements periodically, finding the optimal measurement interval. This can be seen by comparing the coherence time enhancement as a function of the interval (in units of the single-qubit decoherence time T_{single}) for the network (circles) with the analytical predictions (curves; see the Supplemental Material [51]). The remaining differences are due to the discretization of the interval (not considered in the analytics). The coupling between noise and ancilla (μ_2) is different from that between noise and data qubit (μ_1), and the strategy depends on the ratio indicated here. (c) Coherence time enhancement for different numbers of ancillas (here, $\mu_2 = \mu_3 = \mu_4 = 4\mu_1$). (d) Gate sequences. For two ancillas, the network discovers an adaptive strategy, where measurements on qubit 2 are rare, and the measurement basis is decided based on previous measurements of qubit 3. The arrows show the measurement result for qubit 3 in the equator plane of the Bloch sphere. (e) The two-ancilla adaptive strategy (overview; see, also, the Supplemental Material [51]). Here, a brute-force search for strategies is still (barely) possible, becoming infeasible for higher ancilla numbers due to the exponentially large search space of adaptive strategies [$\mu_2 = 3.8\mu_1$, $\mu_3 = 4.1\mu_1$ in (d),(e)].

evolution governing the quantum state) and the particular action set (i.e., the available hardware instructions related to the setup and its connectivity). Importantly, fine-tuning the hyperparameters (like learning rate, network architecture, etc.) is not required; in the Supplemental Material [51], we demonstrate that a common set of hyperparameters can be used for all the scenarios.

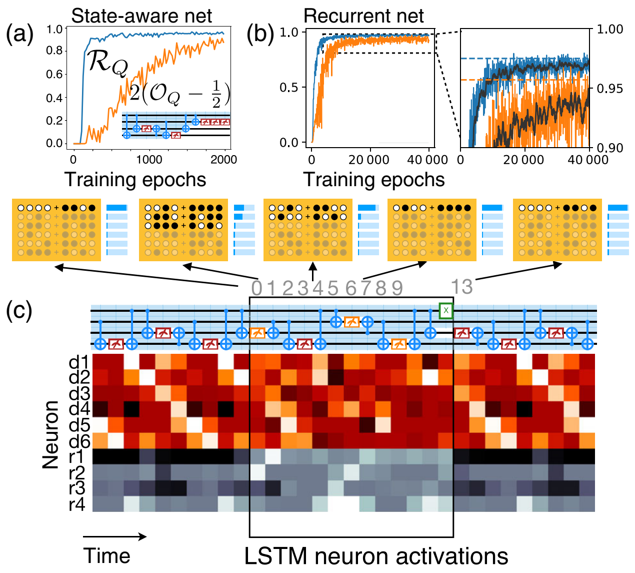

FIG. 7. (a) Training the state-aware network to implement decoding back into the original data qubit near the end of a 200-step gate sequence. Blue: recoverable quantum information $\mathcal{R}_Q$. Orange: (rescaled, shifted) overlap $\mathcal{O}_Q$ between final and initial states of the data qubit. Inset displays the decoding gate sequence. (b) Training the recurrent network in a supervised way to imitate the state-aware network. Again, we show $\mathcal{R}_Q$ and $\mathcal{O}_Q$ evolving during training. Dashed blue or orange lines depict the performance of the state-aware network. (c) To investigate the workings of the recurrent network, we display some of the LSTM neuron activations in the second-to-last layer. Quantum states are illustrated even though the network is unaware of them. The network's input consists of the observed measurement result (if any) and of the actual gate choice (made in the previous time step, here aligned on top of the current neuron activations). The example gate sequence displayed here first shows the repetitive standard error-detection pattern of repeated parity measurements that is interrupted once an unexpected measurement is encountered, indicating an error. During error recovery (boxed region), the network first tries to pinpoint the error and then applies corrections. The neurons whose behavior changes markedly during recovery are depicted in gray. Neuron $1r$ obviously keeps track of whether recovery is ongoing, while $3r$ acts like a counter keeping time during the standard periodic detection pattern. (See the Supplemental Material [51] for more information.)

V. POSSIBLE FUTURE APPLICATIONS

The physical setups considered in today's quantum-computing platforms contain many components and features that go beyond the simplest scenario of short-range coupled qubits. Conceptually, the approach developed in the present work is general enough to find future application in any of the following experimentally relevant domains.

An important example is cavities, which can be used as long-lived quantum memory, especially in the microwave domain. When they are coupled to qubits, nonlinear operations can be performed that may aid in error correction of the cavity state, as the Yale group has demonstrated ("kitten" and "cat" codes [53,54]). Our approach allows us to cover such situations without any changes to the reinforcement-learning method. Only the description of the physical scenario via the set of available actions and, of course, the physics simulation must be updated. Cavities also give access to unconventional controls, e.g., naturally occurring long-distance multiqubit entangling gates provided by the common coupling of the qubits to the cavity. In addition, they permit direct collective readout that is sensitive to the joint state of multiple qubits, which may be used to speed up error-detection operations. Again, RL-based quantum feedback of the type proposed here can naturally make use of these ingredients.

Novel hardware setups, like cross-bar-type geometries [55,56], give rise to the challenge to exploit the unconventional connectivity, for which our approach is well suited. In the future, it may even become possible to cooptimize the hardware layout (taking into account physical constraints) and the strategies adapted to the layout. In the simplest case, this means discovering strategies for automatically generated alternative layouts and comparing their performance.

The actions considered by the agent need not refer to unitary operations. They might also perform other functions, like restructuring the connectivity itself in real time. This is the case for the proposed 2D ion-trap architecture where the ions are shuffled around using electrodes [19]. Similar ideas have been proposed for spins in quantum dots, which can be moved around using electrodes or surface-acoustic waves. Again, no changes to our approach are needed. The modifications are confined to the physics simulation. Depending on the present state of the connectivity, the set of effective qubit gates will change.

Like any numerical approach, our method is invariably limited to modest qubit numbers (of course, these will increase with further optimizations, possibly up to about ten). It is important, therefore, to recall that even an improvement of the decoherence rate in an isolated few-qubit module can have useful applications (as a quantum memory, e.g., in a quantum repeater). More generally, it is clear that classical simulation of a full-scale quantum computer in the domain of quantum supremacy is out of the question, by definition. This is a challenge widely acknowledged by the entire community, affecting not only optimization but also design, testing, and verification of a quantum computer. One promising way to address this challenge at least partially advocated by a growing number of experimental groups is the so-called modular approach to quantum computation and quantum devices. This consists of connecting small few-qubit quantum modules together via quantum network links [19,20]. The main advantage of this approach is the ability to control and debug small quantum modules as opposed to an entire large monolithic quantum computer. Our approach is very well suited to this strategy. In principle, one can even envision a hierarchical application of the quantum-module concept (with error-correction strategies applied to multiple modules coupled together), but for that case, our approach needs to be extended (e.g., by using RL to find one- and two-qubit gates acting on the logical qubits stored inside the modules).

VI. CONCLUSIONS

We see how a network can discover quantum-error-correction techniques from scratch. It finds *a priori* unknown nontrivial detection or recovery sequences for diverse settings without any input beyond the available gate set. The trained neural networks can, in principle, be used to control experimental quantum devices. The present approach is flexible enough to be applied directly to a range of further qualitatively different physical situations, like non-Markovian noise, weak measurements, qubit-cavity systems, and error-corrected transport of quantum information through networks. An obvious challenge for the future is to successfully discover strategies on even more qubits, where eventually full protection against all noise sources and multiple logical qubits can be realized. There is still considerable leeway in improving the speed of the physics simulation and of GPU-based training (for further details on the current computational effort, see Appendix L).

On the machine-learning side, other RL schemes can be substituted for the natural policy gradient adopted here, like Q learning or advantage-actor-critic techniques, or RL with continuous controls. Recurrent networks might be employed to discover useful subsequences. The two-stage learning approach introduced here can also be applied in other RL scenarios, where one first trains based on expanded state information. In general, we show that neural-network-based RL promises to be a flexible and general tool of wide-ranging applicability for exploring feedback-based control of quantum and classical systems in physics.

The data that support the plots within this paper and other findings of this study are available from the authors on request (florian.marquardt@mpl.mpg.de).

ACKNOWLEDGMENTS

We thank Hugo Ribeiro and Vittorio Peano for fruitful comments on the manuscript.

All authors contributed to the ideas, their implementation, and the writing of the manuscript. The numerics was performed by T. F., P. T., and T. W.

Note added.—Recently, a preprint [57] appeared exploring RL with recurrent networks for optimal quantum control (without feedback).

APPENDIX A: PHYSICAL TIME EVOLUTION

To track the time evolution for an *arbitrary* initial logical qubit state (identified by its Bloch vector $\vec{n}$), we start from $\hat{\rho}_{\vec{n}}(0) = \frac{1}{2}(\mathbb{1} + \vec{n}\,\hat{\vec{\sigma}}) \otimes \hat{\rho}_{\text{rest}}$, factorizing out the (fixed) state of all the other qubits. Now, consider the four quantities

$$\hat{\rho}_0(0) = \frac{1}{2}[\hat{\rho}_{\vec{e}_j}(0) + \hat{\rho}_{-\vec{e}_j}(0)], \tag{A1a}$$

$$\delta\hat{\rho}_j(0) = \frac{1}{2}[\hat{\rho}_{\vec{e}_j}(0) - \hat{\rho}_{-\vec{e}_j}(0)], \tag{A1b}$$

where $j \in \{x, y, z\}$ and $\vec{e}_j$ are the basis vectors; note that the right-hand side of Eq. (A1) is independent of j. $\hat{\rho}_0$ and the $\delta\hat{\rho}_j$ are evolved stepwise for each time interval $[t_i, t_f]$ according to the update rule

$$\hat{\rho}_0(t_f) = \frac{\phi[\hat{\rho}_0(t_i)]}{\text{tr}\{\phi[\hat{\rho}_0(t_i)]\}}, \tag{A2a}$$

$$\delta\hat{\rho}_j(t_f) = \frac{\phi[\delta\hat{\rho}_j(t_i)]}{\text{tr}\{\phi[\hat{\rho}_0(t_i)]\}}. \tag{A2b}$$

In the absence of measurements, ϕ is the completely positive map for the given time interval. In the presence of measurements, it is an unnormalized version (see below). We explicitly renormalize such that always $\text{tr}[\hat{\rho}_0(t)] = 1$. $\hat{\rho}_0$ and the $\delta\hat{\rho}_j$ give us access to the density matrix for every logical qubit state, at any time t:

$$\hat{\rho}_{\vec{n}}(t) = \frac{\hat{\rho}_0(t) + \sum_j n_j \delta\hat{\rho}_j(t)}{1 + \text{tr}[\sum_j n_j \delta\hat{\rho}_j(t)]}. \tag{A3}$$

APPENDIX B: PHYSICAL SCENARIOS

We always start from the initial condition that the logical qubit is stored in one physical qubit, and the others are prepared in the down state ($|1\rangle$). If explicit recovery is desired, we use this original qubit also as the target qubit for final decoding. The time evolution is divided into discrete time steps of uniform length Δt (set to 1 in the main text). At the start of each of these time slices, we perform the measurement or gate operation (which is assumed to be quasi-instantaneous) chosen by the agent; afterwards, the system is subject to the dissipative dynamics. Thus, the map ϕ for the time interval $[t, t+\Delta t]$ is of the form $\phi[\hat{\rho}] = e^{\Delta t \mathcal{D}}(\hat{U}\,\hat{\rho}\,\hat{U}^\dagger)$ for unitary operations $\hat{U}$ and $\phi[\hat{\rho}] = e^{\Delta t \mathcal{D}}(\hat{P}_m \hat{\rho} \hat{P}_m^\dagger)$ for projection operators $\hat{P}_m$ (m indicates the measurement results), where $\mathcal{D}$ is the dissipative part of the Liouvillian (we consider only Markovian noise). Note that the measurement results are chosen stochastically according to their respective probability $\text{tr}(\hat{P}_m \hat{\rho}_0)$. In the examples discussed in the figures, we use two different error models, the bit-flip error (BF) and the correlated noise error (CN),

$$\mathcal{D}_{\text{BF}}\hat{\rho} = T_{\text{dec}}^{-1} \sum_q \hat{\sigma}_x^{(q)} \hat{\rho} \hat{\sigma}_x^{(q)} - \hat{\rho}, \tag{B1a}$$

$$\mathcal{D}_{\text{CN}}\hat{\rho} = T_{\text{dec}}^{-1} \left(\hat{L}_{\text{CN}} \hat{\rho} \hat{L}_{\text{CN}}^\dagger - \frac{1}{2}\{\hat{L}_{\text{CN}}^\dagger \hat{L}_{\text{CN}}, \hat{\rho}\} \right), \tag{B1b}$$

where $\hat{\sigma}_{x,y,z}^{(q)}$ applies the corresponding Pauli operator to the qth qubit and $\hat{L}_{\text{CN}} = (1/\sqrt{\sum_q \mu_q^2}) \sum_q \mu_q \hat{\sigma}_z^{(q)}$. Here, μ_q denotes the coupling of qubit q to the noise. Note that in the bit-flip scenario, the single-qubit decay time $T_{\text{single}} = T_{\text{dec}}$, whereas in the presence of correlated noise, it is $T_{\text{single}} = T_{\text{dec}}(\sum_q \mu_q^2)/\mu_1^2$.

APPENDIX C: RECOVERABLE QUANTUM INFORMATION

Based on Eq. (A3), $\mathcal{R}_Q$ as introduced in the main text can be written as

$$\mathcal{R}_Q(t) = \min_{\vec{n}} \left\| \sum_j n_j \delta\hat{\rho}_j(t) \right\|_1 \tag{C1}$$

in the (for us relevant) case that $\text{tr}[\delta\hat{\rho}_j(t)] = 0$ for all j.

The trace distance $\frac{1}{2}\|\sum_j n_j \delta\hat{\rho}_j(t)\|_1$ has often a nontrivial dependence on the logical qubit state $\vec{n}$, and finding its minimum can become nontrivial. However, the location of the minimum can sometimes be "guessed" in advance. For any circuit that can be decomposed into CNOT, Hadamard and $R_Z(\pi/2)$ gates (CHZ) [48], i.e., for all the bit-flip examples considered here, the anticommutator relation $\{\delta\hat{\rho}_j, \delta\hat{\rho}_k\} = 0$ is satisfied for all distinct $j \neq k$; it can be shown that this circumstance restricts the minimum to lie along one of the coordinate axes: $\min_{\vec{n}} \|\sum_j n_j \delta\hat{\rho}_j(t)\|_1 = \min_{j\in\{x,y,z\}} \|\delta\hat{\rho}_j(t)\|_1$. For the correlated noise, the trace distance $\frac{1}{2}\|\sum_j n_j \delta\hat{\rho}_j(t)\|_1$ is symmetric around the z axis and takes its minimal value at the equator.

After a measurement, the updated value of $\mathcal{R}_Q$ may vary between the different measurement results. To obtain a measure that does not depend on this random factor, we introduce $\bar{\mathcal{R}}_Q$ as the average over all possible values of $\mathcal{R}_Q$ (after a single time step) weighted by the probability to end up in the corresponding branch. If the action is not a measurement, there is only one option and, thus, $\bar{\mathcal{R}}_Q = \mathcal{R}_Q$.

APPENDIX D: PROTECTION REWARD

The goal of the "protection reward" is to maximize $\mathcal{R}_Q$ at the end of the simulation; i.e., the ability to, *in principle,* recover the target state. A suitable (immediate) reward is given by

$$r_t = \begin{cases} \underbrace{1+\frac{\bar{\mathcal{R}}_Q(t+\Delta t)-\mathcal{R}_Q(t)}{2\Delta t/T_{\mathrm{single}}}}_{} & \underbrace{+0}_{} \quad \text{if } \bar{\mathcal{R}}_Q(t+\Delta t)>0, \\ 0 & -P \quad \text{if } \mathcal{R}_Q(t)\neq 0 \\ & \quad \wedge\, \mathcal{R}_Q(t+\Delta t)=0, \\ \underbrace{0}_{=:r_t^{(1)}} & \underbrace{+0}_{=:r_t^{(2)}} \quad \text{if } \mathcal{R}_Q(t)=0 \end{cases} \tag{D1}$$

with $\mathcal{R}_Q$ and $\bar{\mathcal{R}}_Q$ as defined above, T_{single} the decay time for encoding in one physical qubit only ("trivial" encoding), Δt the discrete time step of the simulation, and P a punishment for measurements which reveal the logical qubit state. Based on this reward, we choose the return (the function of the reward sequence used to compute the policy gradient) as

$$R_t = (1-\gamma)\left(\sum_{k=0}^{T-t-1}\gamma^k r_{t+k}^{(1)}\right) + r_t^{(2)}, \tag{D2}$$

where γ is the return discount rate; for more information on the (discounted) return, see, e.g., Ref. [58].

APPENDIX E: RECOVERY REWARD

The protection reward does not encourage the network to finally decode the quantum state. If this behavior is desired, we add suitable terms to the reward (only employed for Fig. 7):

$$r_t^{(\mathrm{recov})} = \beta_{\mathrm{dec}}(D_{t+1}-D_t) + \beta_{\mathrm{corr}}\, C_T \delta_{t,T-1}, \tag{E1}$$

where $D_t = \frac{1}{2}(I_t^{(q')} - \sum_{q\neq q'} I_t^{(q)})$, unless $t \leq T_{\mathrm{signal}}$, where we set $D_t = 0$. This means decoding is rewarded only after T_{signal}. We set $I_t^{(q)} = 1$ if $\mathrm{tr}_{\bar{q}}[\delta\hat{\rho}_j(t)] \neq 0$ for any j, and otherwise $I_t^{(q)} = -1$. $\mathrm{tr}_{\bar{q}}$ denotes the partial trace over all qubits except q, and q' labels the target qubit. The condition $I_t^{(q)} = 1$ implies that the logical qubit state is encoded in the specific qubit q (this is not a necessary criterion). C_T is 1 if (at the final time T) the logical qubit state is encoded in the target qubit only, and this qubit has the prescribed polarization (i.e., not flipped), and otherwise 0.

As the return, we use

$$R_t^{(\mathrm{recov})} = (1-\gamma)\sum_{k=0}^{T-t-1}\gamma^k r_{t+k}^{(\mathrm{recov})} \tag{E2}$$

with the same return discount rate γ as for the protection reward.

With this reward, we aim to optimize the minimum overlap $\mathcal{O}_Q = \min_{\vec{n}}\langle\phi_{\vec{n}}|\mathrm{tr}_{\bar{q}'}(\hat{\rho}_{\vec{n}})|\phi_{\vec{n}}\rangle$ between the (pure) target state $|\phi_{\vec{n}}\rangle$ and the actual final state $\hat{\rho}_{\vec{n}}$ reduced to the target qubit given by the partial trace $\mathrm{tr}_{\bar{q}'}(\hat{\rho}_{\vec{n}})$ over all other qubits.

APPENDIX F: INPUT OF THE STATE-AWARE NETWORK

The core of the input to the state-aware network is a representation of the density matrices $\hat{\rho}_0$, $\hat{\rho}_1 \coloneqq \hat{\rho}_0 + \delta\hat{\rho}_x$, $\hat{\rho}_2 \coloneqq \hat{\rho}_0 + \delta\hat{\rho}_y$, and $\hat{\rho}_3 \coloneqq \hat{\rho}_0 + \delta\hat{\rho}_z$. Together, they represent the completely positive map of the evolution (for arbitrary logical qubit states). For reduction of the input size (especially in view of higher qubit numbers), we compress them via principal component analysis (PCA); i.e., we perform an eigendecomposition $\hat{\rho}_j = \sum_k p_k|\phi_k\rangle\langle\phi_k|$ and select the eigenstates $|\phi_k\rangle$ with the largest eigenvalues p_k. To include also the eigenvalue in the input, we feed all components of the scaled states $|\tilde{\phi}_k\rangle \coloneqq \sqrt{p_k}|\phi_k\rangle$ (which yield $\hat{\rho}_j = \sum_k |\tilde{\phi}_k\rangle\langle\tilde{\phi}_k|$) into the network, where the states are in addition sorted by their eigenvalue. For our simulations, we select the six largest components, so we need $768 = 4\times 6\times 16\times 2$ input neurons (four density matrices, 16 is the dimension of the Hilbert space, two for the real and imaginary parts).

In addition, at each time step, we indicate to the network whether a potential measurement would destroy the quantum state by revealing the quantum information. Explicitly, we compute for each measurement whether $\mathrm{tr}(\hat{P}\delta\hat{\rho}_j) = 0$ for all $j \in \{x,y,z\}$ and every possible projector $\hat{P}$ (i.e., every possible measurement result) and feed these Boolean values into the network. Note that this information can be deduced from the density matrix (so, in principle, the network can learn that deduction on its own but giving it directly speeds up training).

Because all relevant information for the decision about the next action is contained in the current density matrix, knowledge about the previous actions is not needed. However, we find that providing this extra information is helpful to accelerate learning. Therefore, we provide also the last action (in a one-hot encoding). We note that it is not necessary to feed the latest measurement result to the network since the updated density matrix is conditional on the measurement outcome and, therefore, contains all relevant information for a future decision.

To train the state-aware network to restore the original state at the end of a trajectory, it becomes necessary to add the time to the input. It is fully sufficient to indicate the last few time steps where $t > T_{\mathrm{signal}}$ (when decoding should be performed) in a one-hot encoding.

APPENDIX G: LAYOUT OF THE STATE-AWARE NETWORK

Our state-aware networks have a feedforward architecture. Between the input layer and the output layer (one neuron per action), there are two or three hidden layers (the specific numbers are summarized in Appendix K). All neighboring layers are densely connected, the activation function is the rectified linear unit. At the output layer, the softmax function $x_j \mapsto e^{x_j}/\sum_k e^{x_k}$ is applied such that the result can be interpreted as a probability distribution.

APPENDIX H: REINFORCEMENT LEARNING OF THE STATE-AWARE NETWORK

Our learning scheme is based on the policy gradient algorithm [45]. The full expression for our learning gradient (indicating the change in θ) reads

$$g = \lambda_{\mathrm{pol}} F^{-1}\mathbb{E}\left[\sum_t (R_t - b_t)\frac{\partial}{\partial\theta}\ln \pi_\theta(a_t|s_t)\right] - \lambda_{\mathrm{entr}}\mathbb{E}\left[\sum_a \frac{\partial}{\partial\theta}(\pi_\theta(a|s)\ln \pi_\theta(a|s))\right], \tag{H1}$$

where R_t is the (discounted) return [compare Eqs. (D2) and (E2)]. This return is corrected by an (explicitly time-dependent) baseline b_t, which we choose as the exponentially decaying average of R_t; i.e. for the training update in epoch N, we use $b_t = (1-\kappa)\sum_{n=0}^{N-1}\kappa^n \bar{R}_t^{(N-1-n)}$, where κ is the baseline discount rate, and $\bar{R}_t^{(n)}$ is the mean return at time step t in epoch n. We compute the natural gradient [59–61] by multiplying F^{-1}, the (Moore-Penrose) inverse of the Fisher information matrix $F = \mathbb{E}\{[(\partial/\partial\theta)\ln\pi_\theta(a|s)][(\partial/\partial\theta)\ln\pi_\theta(a|s)]^{\mathsf{T}}\}$. The second term is entropy regularization [62]; we use it only to train the state-aware network shown in Fig. 7. As update rule, we use adaptive moment estimation (Adam) (see Ref. [63]) without bias correction.

APPENDIX I: LAYOUT OF THE RECURRENT NETWORK

The recurrent network is designed such that it can, in principle, operate in a real-world experiment. This means, in particular, that (in contrast to the state-aware network) its input must not contain directly the quantum state (or the evolution map); instead, measurements are its only way to obtain information about the quantum system. Hence, the input to the recurrent network contains the present measurement result (and, additionally, the previous action). Explicitly, we choose the input as a one-hot encoding for the action in the last time step, and in case of measurements, we additionally distinguish between the different results. In addition, there is an extra input neuron to indicate the beginning of time (where no previous action is performed). Since this input contains only the most recent event, the network requires a memory to perform reasonable strategies; i.e., we need a recurrent network. Therefore, the input and output layers are connected by two successive LSTM layers [52] with tanh interlayer activations. After the output layer, the softmax function is applied (like for the state-aware network).

APPENDIX J: SUPERVISED LEARNING OF THE RECURRENT NETWORK

The training data are generated from inference of a state-aware network which is trained to sufficiently good strategies (via reinforcement learning); for every time step in each trajectory, we save the network input and the policy, i.e., the probabilities for all the actions, and we train on these data. It is possible to generate enough data such that overfitting is not a concern (for the example in Fig. 7, each trajectory is reused only five times during the full training process). For the actual training of the recurrent network, we use supervised learning with categorical cross-entropy as the cost function (q is the actual policy of the recurrent network to train, and p the desired policy from the state-aware network):

$$C(q,p) = -\sum_a p(a)\ln q(a). \tag{J1}$$

Because of the LSTM layers, it is necessary to train on full trajectories (in the true time sequence) instead of individual actions. Dropout [64] is used for regularization. The training update rule is adaptive moment estimation (Adam) (see Ref. [63]).

APPENDIX K: PHYSICAL PARAMETERS AND HYPERPARAMETERS

The physical parameters used throughout the main text are summarized in Table I. Times are always given in units of the time step (gate time).

We use a few separately trained neural networks throughout this work which differ slightly in hyperparameters (e.g., in the number of hidden layers and neurons per layer). This is not due to fine-tuning, and in the Supplemental Material [51], we demonstrate that we can successfully train the neural networks in all scenarios with one common set of hyperparameters. The strategies found by the neural networks are not influenced by using different sets of hyperparameters. Different hyperparameters may influence the training time, etc. In Table II, we summarize the architecture of the networks; i.e., we list the number of neurons in each layer.

Each output neuron represents one action that can be performed by the agent, and, thus, the output layer size is

TABLE I. Physical parameters.

Figures 3–5,7 decoherence time T_{dec}	1200
Figure 6 single-qubit decoherence time $T_{\mathrm{single}} = T_{\mathrm{dec}}/\mu_1^2$	500
Figures 3–5,7 number of time steps T	200
Figure 6 number of time steps T	100

TABLE II. Network architectures.

	Neurons per layer
Figure 3	(793, 300, 300, 21)
Figure 4	(793, 300, 300, 300, 21)
Figures 5(a) and 5(b) all-to-all connected:	(793, 300, 300, 21)
Figures 5(a) and 5(b) chain:	(787, 300, 300, 300, 15)
Figures 5(a) and 5(b) chain with one meas qubit:	(781, 600, 300, 12)
Figures 5(a) and 5(b) circle:	(783, 300, 300, 300, 14)
Figure 5(c) (all identical)	(793, 300, 300, 300, 21)
Figures 6(b)–6(d): one ancilla	(101,100,50,3)
Figures 6(c) and 6(d): two ancillas	(265, 100, 50, 5)
Figures 6(c): three ancillas	(781, 200, 100, 7)
Figure 7(a)	(803, 300, 300, 21)
Figure 7(b) and 7(c) (hidden layers with LSTM units)	(26, 128, 128, 21)

equal to the number of actions. In the bit-flip scenarios (Figs. 3–5,7), the actions are CNOT gates according to connectivity, measurements along z as indicated in the corresponding sketches, deterministic bit flips on each qubit, and idle. When dealing with correlated noise, cf. Fig. 6, the available actions are instead measurements along x and y on all ancilla qubits, and the idle operation. Note that our whole approach is general and able in principle to deal with arbitrary quantum gates.

The hyperparameters used for training the state-aware networks are summarized in Table III.

The hyperparameters used for training the recurrent network [cf. Figs. 7(b) and 7(c)] are summarized in Table IV.

TABLE III. Hyperparameters state-aware network.

Training batch size	64
Adam parameters η, β_1, β_2	0.0003,[a] 0.9, 0.999 (no bias correction)
Return discount rate γ	0.95
Baseline discount rate κ	0.9
Punishment reward coefficient P	0.1
Decoding reward coefficient β_{dec} [used only in Figure 7(a)]	20
Correction reward coefficient β_{corr} [used only in Figure 7(a)]	10
Decoding signal time T_{signal} [used only in Figure 7(a)]	$T_{\mathrm{signal}} = T - 10 = 190$
Reward scale λ_{pol}	4.0
Entropy regularization λ_{entr} [used only in Figure 7(a)]	$\lambda_{\mathrm{entr}} = 5 \times 10^{-3}$ for the first 12 000 training epoches, $\lambda_{\mathrm{entr}} = 0$ afterwards

[a]The exact value for the learning rate used in the simulations is, in fact, $0.0001\sqrt{10}$; the irrational factor of $\sqrt{10}$ is caused by a slight deviation between our implementation and the standard Adam scheme, which, in the end, results only in a redefinition of the learning rate.

TABLE IV. Hyperparameters recurrent network.

Training batch size	16
Adam parameters η, β_1, β_2	0.001, 0.9, 0.999
Dropout level (after each LSTM layer)	0.5

Our RL implementation relies on the Theano framework [65] (and Keras for defining networks).

APPENDIX L: COMPUTATIONAL RESOURCES

The computationally most expensive tasks in this paper are the training runs of the state-aware networks depicted in Figs. 3, 5, and 7(a). Full training for a given scenario can be achieved using 1.6×10^6 training trajectories, which can be run within 6 h on a single CPU + GPU node (CPU, Intel Xeon E5-1630 v4; GPU, Nvidia Quadro P5000). Currently, more than 2/3 of the time is spent on the numerical simulation of the physics time evolution, which is still performed on the CPU. We expect that the total run-time can be improved significantly by a more efficient implementation and more powerful hardware (including also an implementation of the physics simulation on the GPU). The memory consumption is very modest (for these examples, below 200 MB) dominated mostly by the need for storing the input to the network for all trajectories inside a batch (here, 64 trajectories with 200 time steps each) and less by the network weights (here, up to about 600 000). For a more detailed discussion, we refer the reader to the Supplemental Material [51] (Sec. VI).

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

[2] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach* (Pearson, Boston, 2018).

[3] G. Carleo and M. Troyer, *Solving the Quantum Many-Body Problem with Artificial Neural Networks*, Science **355**, 602 (2017).

[4] J. Carrasquilla and R. G. Melko, *Machine Learning Phases of Matter*, Nat. Phys. **13**, 431 (2017).

[5] E. P. L. van Nieuwenburg, Y.-H. Liu, and S. D. Huber, *Learning Phase Transitions by Confusion*, Nat. Phys. **13**, 435 (2017).

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

[7] M. August and X. Ni, *Using Recurrent Neural Networks to Optimize Dynamical Decoupling for Quantum Memory*, Phys. Rev. A **95**, 012335 (2017).

[8] V. Dunjko and H. J. Briegel, *Machine Learning and Artificial Intelligence in the Quantum Domain*, arXiv:1709.02779.

[9] P. Baireuther, T. E. O'Brien, B. Tarasinski, and C. W. J. Beenakker, *Machine-Learning-Assisted Correction of Correlated Qubit Errors in a Topological Code*, Quantum **2**, 48 (2018).

[10] S. Krastanov and L. Jiang, *Deep Neural Network Probabilistic Decoder for Stabilizer Codes*, Sci. Rep. **7**, 11003 (2017).

[11] P. Mehta, M. Bukov, C.-H. Wang, A. G. R. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, *A High-Bias, Low-Variance Introduction to Machine Learning for Physicists*, arXiv:1803.08823.

[12] R. D. King, J. Rowland, S. G. Oliver, M. Young, W. Aubrey, E. Byrne, M. Liakata, M. Markham, P. Pir, L. N. Soldatova, A. Sparkes, K. E. Whelan, and A. Clare, *The Automation of Science*, Science **324**, 85 (2009).

[13] M. Schmidt and H. Lipson, *Distilling Free-Form Natural Laws from Experimental Data*, Science **324**, 81 (2009).

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, *Human-Level Control through Deep Reinforcement Learning*, Nature (London) **518**, 529 (2015).

[15] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, *Mastering the Game of Go without Human Knowledge*, Nature (London) **550**, 354 (2017).

[16] C. Chen, D. Dong, H. X. Li, J. Chu, and T. J. Tarn, *Fidelity-Based Probabilistic Q-Learning for Control of Quantum Systems*, IEEE Trans. Neural Netw. Learn. Syst. **25**, 920 (2014).

[17] M. Bukov, A. G. R. Day, D. Sels, P. Weinberg, A. Polkovnikov, and P. Mehta, *Reinforcement Learning in Different Phases of Quantum Control*, arXiv:1705.00565.

[18] A. A. Melnikov, H. P. Nautrup, M. Krenn, V. Dunjko, M. Tiersch, A. Zeilinger, and H. J. Briegel, *Active Learning Machine Learns to Create New Quantum Experiments*, Proc. Natl. Acad. Sci. U.S.A. **115**, 1221 (2018).

[19] C. Monroe and J. Kim, *Scaling the Ion Trap Quantum Processor*, Science **339**, 1164 (2013).

[20] M. H. Devoret and R. J. Schoelkopf, *Superconducting Circuits for Quantum Information: An Outlook*, Science **339**, 1169 (2013).

[21] 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).

[22] 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).

[23] G. Waldherr, Y. Wang, S. Zaiser, M. Jamali, T. Schulte-Herbrüggen, H. Abe, T. Ohshima, J. Isoya, J. F. Du, P. Neumann, and J. Wrachtrup, *Quantum Error Correction in a Solid-State Hybrid Spin Register*, Nature (London) **506**, 204 (2014).

[24] J. Kelly *et al.*, *State Preservation by Repetitive Error Detection in a Superconducting Quantum Circuit*, Nature (London) **519**, 66 (2015).

[25] M. Veldhorst, C. H. Yang, J. C. C. Hwang, W. Huang, J. P. Dehollain, J. T. Muhonen, S. Simmons, A. Laucht, F. E. Hudson, K. M. Itoh, A. Morello, and A. S. Dzurak, *A Two-Qubit Logic Gate in Silicon*, Nature (London) **526**, 410 (2015).

[26] 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).

[27] A. Potocnik, A. Bargerbos, F. A. Y. N. Schröder, S. A. Khan, M. C. Collodo, S. Gasparinetti, Y. Salathé, C. Creatore, C. Eichler, H. E. Türeci, A. W. Chin, and A. Wallraff, *Studying Light-Harvesting Models with Superconducting Circuits*, Nat. Commun. **9**, 904 (2018).

[28] S. Debnath, N. M. Linke, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe, *Demonstration of a Small Programmable Quantum Computer with Atomic Qubits*, Nature (London) **536**, 63 (2016).

[29] 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).

[30] T. F. Watson, S. G. J. Philips, E. Kawakami, D. R. Ward, P. Scarlino, M. Veldhorst, D. E. Savage, M. G. Lagally, Mark Friesen, S. N. Coppersmith, M. A. Eriksson, and L. M. K. Vandersypen, *A Programmable Two-Qubit Quantum Processor in Silicon*, Nature (London) **555**, 633 (2018).

[31] D. Ristè, C. C. Bultink, K. W. Lehnert, and L. DiCarlo, *Feedback Control of a Solid-State Qubit Using High-Fidelity Projective Measurement*, Phys. Rev. Lett. **109**, 240502 (2012).

[32] P. Campagne-Ibarcq, E. Flurin, N. Roch, D. Darson, P. Morfin, M. Mirrahimi, M. H. Devoret, F. Mallet, and B. Huard, *Persistent Control of a Superconducting Qubit by Stroboscopic Measurement Feedback*, Phys. Rev. X **3**, 021008 (2013).

[33] L. Steffen, Y. Salathe, M. Oppliger, P. Kurpiers, M. Baur, C. Lang, C. Eichler, G. Puebla-Hellmann, A. Fedorov, and A. Wallraff, *Deterministic Quantum Teleportation with Feed-Forward in a Solid State System*, Nature (London) **500**, 319 (2013).

[34] W. Pfaff, B. J. Hensen, H. Bernien, S. B. van Dam, M. S. Blok, T. H. Taminiau, M. J. Tiggelman, R. N. Schouten, M. Markham, D. J. Twitchen, and R. Hanson, *Unconditional Quantum Teleportation between Distant Solid-State Quantum Bits*, Science **345**, 532 (2014).

[35] D. Ristè and L. DiCarlo, in *Superconducting Devices in Quantum Optics*, edited by R. H. Hadfield and G. Johansson (Springer, New York, 2016).

[36] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbrüggen, and S. J. Glaser, *Optimal Control of Coupled Spin Dynamics: Design of NMR Pulse Sequences by Gradient Ascent Algorithms*, J. Magn. Reson. **172**, 296 (2005).

[37] S. Machnes, U. Sander, S. J. Glaser, P. de Fouquières, A. Gruslys, S. Schirmer, and T. Schulte-Herbrüggen, *Comparing, Optimizing, and Benchmarking Quantum-Control Algorithms in a Unifying Programming Framework*, Phys. Rev. A **84**, 022305 (2011).

[38] P. D. Johnson, J. Romero, J. Olson, Y. Cao, and A. Aspuru-Guzik, *QVECTOR: An Algorithm for Device-Tailored Quantum Error Correction*, arXiv:1711.02249.

[39] A. Hentschel and B. C. Sanders, *Machine Learning for Precise Quantum Measurement*, Phys. Rev. Lett. **104**, 063603 (2010).

[40] A. Hentschel and B. C. Sanders, *Efficient Algorithm for Optimizing Adaptive Quantum Metrology Processes*, Phys. Rev. Lett. **107**, 233601 (2011).
[41] P. Palittapongarnpim, P. Wittek, E. Zahedinejad, S. Vedaie, and B. C. Sanders, *Learning in Quantum Control: High-Dimensional Global Optimization for Noisy Quantum Dynamics*, Neurocomputing **268**, 116 (2017).
[42] M. Tiersch, E. J. Ganahl, and H. J. Briegel, *Adaptive Quantum Computation in Changing Environments Using Projective Simulation*, Sci. Rep. **5**, 12874 (2015).
[43] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, *Quantum Machine Learning*, Nature (London) **549**, 195 (2017).
[44] J. Romero, J. P. Olson, and A. Aspuru-Guzik, *Quantum Autoencoders for Efficient Compression of Quantum Data*, Quantum Sci. Technol. **2**, 045001 (2017).
[45] R. J. Williams, *Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning*, Mach. Learn. **8**, 229 (1992).
[46] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, MA, 2016).
[47] P. W. Shor, *Scheme for Reducing Decoherence in Quantum Computer Memory*, Phys. Rev. A **52**, R2493 (1995).
[48] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* 10th anniv. ed. (Cambridge University Press, Cambridge, England, 2011).
[49] B. M. Terhal, *Quantum Error Correction for Quantum Memories*, Rev. Mod. Phys. **87**, 307 (2015).
[50] L. van der Maaten and G. Hinton, *Visualizing Data Using t-SNE*, J. Mach. Learn. Res. **9**, 2579 (2008).
[51] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevX.8.031084 for further information on the physical time evolution, the recoverable quantum information, conception and training of the state-aware and the recurrent network, the t-SNE analysis, and the numerical implementation, plus background information on the figures.
[52] S. Hochreiter and J. Schmidhuber, *Long Short-Term Memory*, Neural Comput. **9**, 1735 (1997).
[53] M. Mirrahimi, Z. Leghtas, V. V. Albert, S. Touzard, R. J. Schoelkopf, L. Jiang, and M. H. Devoret, *Dynamically Protected Cat-Qubits: A New Paradigm for Universal Quantum Computation*, New J. Phys. **16**, 045014 (2014).
[54] R. W. Heeres, P. Reinhold, N. Ofek, L. Frunzio, L. Jiang, M. H. Devoret, and R. J. Schoelkopf, *Implementing a Universal Gate Set on a Logical Qubit Encoded in an Oscillator*, Nat. Commun. **8**, 94 (2017).
[55] F. Helmer, M. Mariantoni, A. G. Fowler, J. von Delft, E. Solano, and F. Marquardt, *Cavity Grid for Scalable Quantum Computation with Superconducting Circuits*, Europhys. Lett. **85**, 50007 (2009).
[56] R. Li, L. Petit, D. P. Franke, J. P. Dehollain, J. Helsen, M. Steudtner, N. K. Thomas, Z. R. Yoscovits, K. J. Singh, S. Wehner, L. M. K. Vandersypen, J. S. Clarke, and M. Veldhorst, *A Crossbar Network for Silicon Quantum Dot Qubits*, arXiv:1711.03807.
[57] M. August and J. M. Hernández-Lobato, *Taking Gradients through Experiments: LSTMs and Memory Proximal Policy Optimization for Black-Box Quantum Control*, arXiv:1802.04063.
[58] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, MA, 1998), Vol. 1.
[59] S.-i. Amari, *Natural Gradient Works Efficiently in Learning*, Neural Comput. **10**, 251 (1998).
[60] S. M. Kakade, *A Natural Policy Gradient*, in *Advances in Neural Information Processing Systems* (2001), pp. 1531–1538, https://papers.nips.cc/paper/2073-a-natural-policy-gradient.
[61] J. Peters and S. Schaal, *Natural Actor-Critic*, Neurocomputing **71**, 1180 (2008).
[62] R. J. Williams and J. Peng, *Function Optimization Using Connectionist Reinforcement Learning Algorithms*, Connection Science **3**, 241 (1991).
[63] D. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in *Proceedings of the International Conference for Learning Representations*, 2015, Vol. 6 [arXiv:1412.6980].
[64] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, *Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors*, arXiv:1207.0580.
[65] R. Al-Rfou *et al.*, *THEANO: A PYTHON Framework for Fast Computation of Mathematical Expressions*, arXiv:1605.02688.