Reinforcement Learning with Neural Networks for Quantum Feedback

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

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

Artificial neural networks are revolutionizing science. While the most prevalent technique involves supervised training on queries with a known correct answer, more advanced challenges often require discovering answers autonomously. In reinforcement learning, control strategies are improved according to a reward function. The power of this approach has been highlighted by spectactular recent successes, such as playing Go. So far, it has remained an open question whether neural-network-based reinforcement learning can be successfully applied in physics. Here, we show how to use this method for finding quantum feedback schemes, where a network-based "agent" interacts with and occasionally decides to measure a quantum system. We illustrate the utility by finding gate sequences that preserve the quantum information stored in a small collection of qubits against noise. This specific application will help to find hardware-adapted feedback schemes for small quantum modules while demonstrating more generally the promise of neural-network based reinforcement learning in physics.

We are witnessing rapid progress in applications of artificial neural networks (ANN) 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 [3–10]. 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 [11, 12], 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 was demonstrated convincingly through learning to play games beyond human expertise [13, 14].

In physics, RL *without* neural networks has been introduced recently, for example to study qubit control [15] and invent quantum optics experiments [16]. Moving to neural-network-based RL promises access to the vast variety of techniques currently being developed for ANNs. In this work, we introduce network-based RL in physics. We illustrate its versatility in the domain of quantum feedback. Specifically, we consider a network agent that tries to preserve the quantum information stored in a few-qubit system against decoherence. It discovers a discrete sequence of gates and measurements ("quantum circuit"), adapting to the measurement outcomes. First applications of neural networks to quantum memories and error correction already exist [6, 7, 9, 10], but here we employ RL as a general strategy that naturally incorporates feedback.

Our goal is to present RL as a *flexible* approach to quantum feedback. Even if in any specific case one could select other, more specialized techniques, the advantage of RL is that it can discover strategies from scratch with minimal input. 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. Regarding the particular example of quantum error correction in few-qubit setups, finding a suitable encoding is only a minor part of the challenge (and is actually known in the examples we consider). However, the best sequences for error detection and correction depend crucially on the available gates, are not easy to discover, and are often unknown a-priori, even for the 4-qubit cases we discuss. This is where the RL approach excels.

Indeed, the progress in multi-qubit quantum devices [17–26] has highlighted hardware features deviating from often-assumed idealized scenarios. These include qubit connectivity, correlated noise, restrictions on measurements, or uneven error rates. Our approach can help finding "hardware-adapted" solutions.

Other wide-spread optimization techniques for quantum control, like GRAPE, often vary evolution operators with respect to continuous parameters [27, 28], but do not easily include feedback. A recent approach [29] to quantum error correction uses optimization of control parameters in a pre-configured gate sequence. By contrast, policy gradient 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 [30, 31].

Conceptually, we are dealing with a "hybrid" RL approach, where the RL-environment is a quantum system. To avoid confusion, we emphasize our approach is distinct

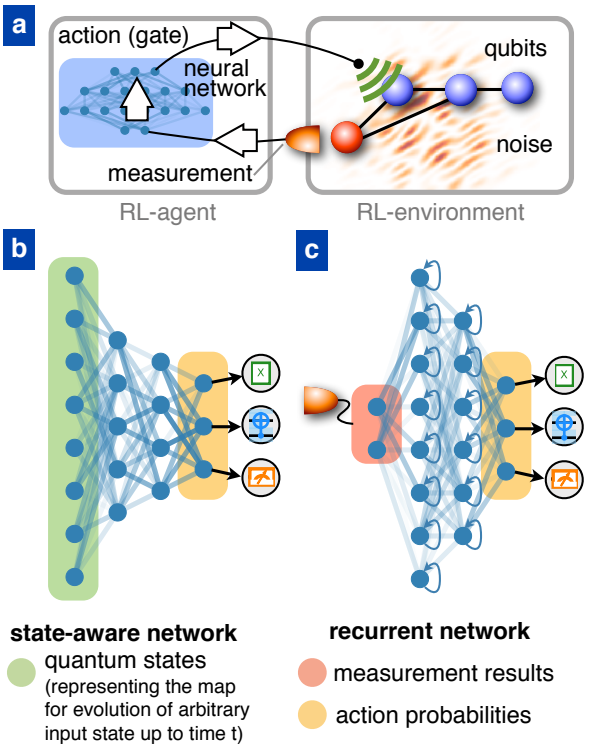

Figure 1. (a) The general setting of this work: A few-qubit quantum device with a neural-network-based controller whose task is to protect the quantum memory residing in this device against noise. Reinforcement learning (RL) lets the controller ("RL-agent") discover on its own how to best choose gate sequences, perform measurements, and react to measurement results, by interacting with the quantum device ("RL-environment"). (b) The state-aware network receives a representation of the map Φ describing the evolution of arbitrary initial logical qubit states (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 layer-wise 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 around 10-20 actions. (c) The recurrent network receives the most recent measurement result (if any) and also outputs 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).

from future "quantum machine learning" devices, where even the network will be quantum [8, 32].

REINFORCEMENT LEARNING

The purpose of RL (Fig. 1a) is to find an optimal set of actions (in our case, quantum gates and measurements) that an "agent" (controller) 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. In our application, we initialize a single qubit in an arbitrary superposition $|\Psi_i\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$. A naive version of the return might be the overlap between the final quantum state of that physical qubit and $|\Psi_i\rangle$, though we will see that a more sophisticated approach is required.

It is clear that a quantum system playing the role of an RL-environment is significantly different from its classical counterpart. A quantum measurement fundamentally alters the state, instead of merely revealing information about a pre-existing state.

To find optimal gate sequences, we employ a widespread version of reinforcement learning [33, 34] 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 current state s_t of the RL-environment. As we will use a neural network to compute π_θ, the multi-dimensional parameter θ stands for all the network's weights and biases. The network is fed s_t (or a pre-processed representation) 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 [33]:

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

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 Methods). Even so, several further conceptual steps are needed to have any chance of success.

REINFORCEMENT LEARNING APPROACH TO QUANTUM MEMORY

The particular task we are addressing here seems practically unsolvable for the present reinforcement learning techniques if no extra precautions are taken. The basic challenge has also been encountered in other difficult RL applications: the first sequence leading to an increased return is rather long, in our case including all the encoding, detection and correction steps. In our scenarios, the probability to randomly select the correct sequence is much less than 10^{-12}. Moreover, any subsequence may be worse than the trivial (idle) operation: for example,

encoding the quantum state only partially can accelerate decay. Adopting the straightforward return, namely the overlap of the final and initial states, both the trivial solution (1-qubit storage) and the error-corrected memory are fixed points. These are separated by a wide barrier – all the intermediate-length sequences with lower return. In our numerical experiments, direct RL was not successful, except for some tasks with very few qubits and gates.

We found that we need to 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, 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 will be based on the completely positive map describing the dissipative quantum evolution of arbitrary states. The only statistical averaging necessary is over measurement outcomes and the probabilistic action choices. Further below, we comment on how this is implemented in practice.

As known from other RL applications (like board games [14]), it helps to provide as much information as possible to the network. In our case, this could mean providing the multi-qubit quantum state at each time step. However, that information is not available in a real experiment. In order to solve this dilemma, we have found that the best strategy involves training two different networks in succession (Fig. 1b,c): The first network is fully *state-aware*. Later on, we will use it as a teacher for the second network which essentially only gets the measurement results as an input.

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 multi-qubit system that maps $\hat{\rho}(0)$ to $\hat{\rho}(t)$. Moreover, we have found it useful to apply principal component analysis, i. e. to feed only the few largest-weight eigenvectors of $\hat{\rho}$ as input to the network (see Methods).

Two-stage learning with parallel evolution is essential, but not yet sufficient for our challenge. We now introduce a suitable immediate reward that indicates the likely final success of an action sequence ahead of time. Constructing such a reward requires domain-specific knowledge. 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 multi-qubit 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 non-trivial, non-analytic 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 $\mathcal{R}_Q$ yields a very powerful immediate reward scheme. In the idealized case where errors have occured, but they could *in principle* be perfectly recovered by a suitable detection/decoding sequence, it would remain 1, independently of whether that sequence has already been found by the network. As we will 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 only required to prevent the multi-qubit system from venturing into regions of the Hilbert space where any further decoherence process would irreversibly destroy the quantum information (and lower $\mathcal{R}_Q$). If one wants the network to implement also the decoding sequence, to return back an unentangled state, this can be done by adding suitable contributions to the reward (see below).

RESULTS

We now apply the general approach to different settings, illustrating its flexibility. The training of the state-

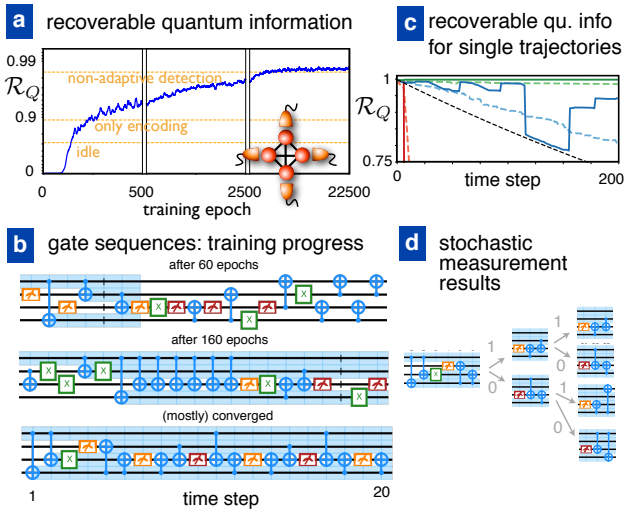

Figure 2. Reinforcement learning for the state-aware network, with a 4-qubit setup (all qubits can be measured, as indicated by the detectors and the red color; all qubit pairs can be subject to CNOT, as indicated by the links). (a) Training progress in terms of the average recoverable quantum information $\mathcal{R}_Q$, evaluated at the final step of a 200-step gate sequence. One training "epoch" involves training on a batch of 64 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). (b) Typical gate sequences at different training stages, mostly converged strategy at bottom. Qubits participating in encoding (holding information about the logical qubit amplitudes) are indicated with light blue background. (c) Time-evolution of $\mathcal{R}_Q$, at the same three different training stages, for individual trajectories (dashed: averaged over many trajectories). Jumps are due to measurements. (d) Evolution depending on stochastic measurement results, indicated as "0"/"1" and also via the color (bright/dark) of the measurement gate symbol. Note that even the mostly converged strategy is still probabilistic to some degree.

aware network is analyzed in Fig. 2. In this example, the qubits are subject to bit-flip errors, with a decay term $\dot{\hat{\rho}} = T_{\rm dec}^{-1} \sum_j (\hat{\sigma}_x^{(j)} \hat{\rho} \hat{\sigma}_x^{(j)} - \hat{\rho})$ in the underlying master equation (see Methods). All of the four qubits may be measured, and there is full connectivity. During training (Fig. 2a,b), the network first learns to avoid destructive measurements which reveal the logical qubit state. Afterwards, it discovers a gate sequence of CNOTs that creates an entangled state, implementing some version of the 3-qubit repetition code [35, 36]. The symmetry between alternative encodings is broken spontaneously during training. The encoding already increases the reward above the trivial level (storage in one qubit). Finally, the network starts doing repeated parity measurements, of the type CNOT($B \mapsto A$), CNOT($C \mapsto A$), M(A), where the state of ancilla A is flipped only if the states of B and C differ (here M is a measurement). These help to preserve the quantum information, preventing the leakage into states with two bit flips that cannot be corrected if undetected. Fig. 2b illustrates the progression from random quantum circuits to a near-ideal approach.

During any single trajectory, the recoverable quantum information can have sudden jumps when measurements are performed (Fig. 2c), with collapses and revivals.

How does the network operate? Does it group together states that merit the same action? To understand better the internal operation, we visualize the network's response to the input state $\hat{\rho}$ (Fig. 3c), obtaining a nonlinear projection from the high-dimensional space of neuron activations into the 2D plane. This indicates clusters of quantum states that yield similar activations. During a gate sequence, the network visits states in different clusters. The sequence becomes complex if unexpected measurement results are encountered (Fig. 3b). In the example shown here, the outcome of the first parity measurement is compatible with three possibilities (one of two qubits has been flipped, or the ancilla state is erroneous). The network has learned that the ambiguity can be resolved through two further measurements, returning to the usual detection cycle. It is remarkable that reinforcement learning can find these nontrivial sequences (which would be hard to construct using any other method), picking out small 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. 4a) adapted to the available resources. For example, in a chain, with a single measurement location, the network learns that it may use nearest-neighbor CNOTs to swap through the chain. However, if any 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 ancilla. We also show, specifically, the complex recovery sequences triggered by unexpected measurements. Generally, additional resources are exploited to yield better performance (Fig. 4b). The network successfully adapts its strategy to changes in parameters (Fig. 4c).

In a different class of scenarios, we consider dephasing of a qubit by a fluctuating field (Fig. 5). If the field is spatially homogeneous, it can be tracked and corrected for by observing the evolution of nearby ancillas that also couple to the field. The situation involves collective dephasing, with $\hat{H}(t) = B(t) \sum_j \mu_j \hat{\sigma}_z^{(j)}$, where $B(t)$ is white noise and μ_j are the coupling strengths (to qubit and ancillas). The same RL program used for the examples above also finds solutions here (Fig. 5), without any input specific to the situation (except the available gates). 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 exponential growth of possibilities.

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

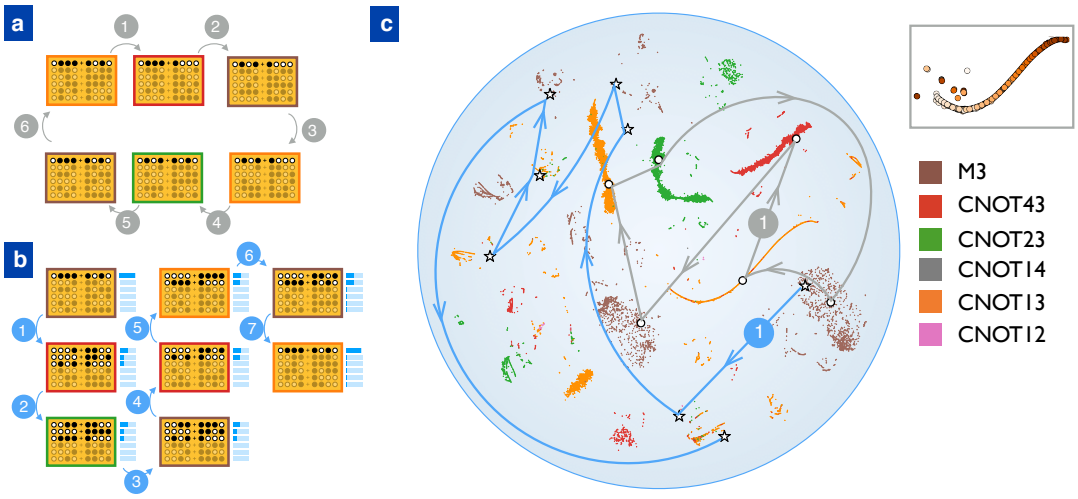

Figure 3. Visualizing the operation of the state-aware network. (Same scenario as in Fig. 2) (a) Sequence of quantum states visited during a standard repetitive detection cycle, displayed for an initial logical qubit state $(|0\rangle + |1\rangle)/\sqrt{2}$, in the absence of unexpected measurement outcomes (no errors). Each state $\hat{\rho}$ is represented by its decomposition into eigenstates, sorted according to decreasing eigenvalues (probabilities). The eigenstates of less than 5% weight are displayed semi-transparently. Here, each eigenstate is a superposition of two basis states in the z-basis, but the RL approach is independent of this fact. (b) For a particular gate sequence that is triggered upon encountering an unexpected measurement, we also indicate the states $\hat{\rho}$ (with bars now showing the probabilities). This sequence tries to disambiguate, by further measurements, 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 [37], which is a nonlinear mapping that tries to preserve neighborhood relations while reducing the dimensionality. Each of the $2 \cdot 10^5$ points corresponds to one activation pattern and is colored according to the most probable action the network suggests for the corresponding input map Φ. 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 CNOT13 has control-qubit 1 and target 3. The zoom-in shows a set of states in a single cluster; the shading indicates the time progressing during the gate sequence, with a slow drift of the state due to decoherence.

physical qubit. Our objective is to maximize the overlap between the initial and final states, for any logical qubit state (see Methods). Moreover, we found 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. 6a). 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 this is because it tries to return as soon as possible back to the known encoded state, for which the proper gate sequence and interpretation of measurement results are already familiar. 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. This requires a network whose only input consists in the measurement results (and the selected gates, since the policy is probabilistic), requiring some sort of memory. An elegant solution consists in a *recurrent* neural network. We use the widespread long short-term memory (LSTM) approach [38].

Once the first, state-aware network has been trained successfully, it is used as a teacher in supervised learning to train the second, recurrent network (Fig. 6b). This 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.). Learning to correct (see above) is essential for successful training of the recurrent network, since the latter must learn to consider measurement results distributed in time and deduce the proper corrective actions.

We have trained the recurrent network based on a fully converged state-aware network. Inspecting the LSTM neuron activations (Fig. 6c), we see that different neurons activate for different events and some clearly display memory behavior (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.

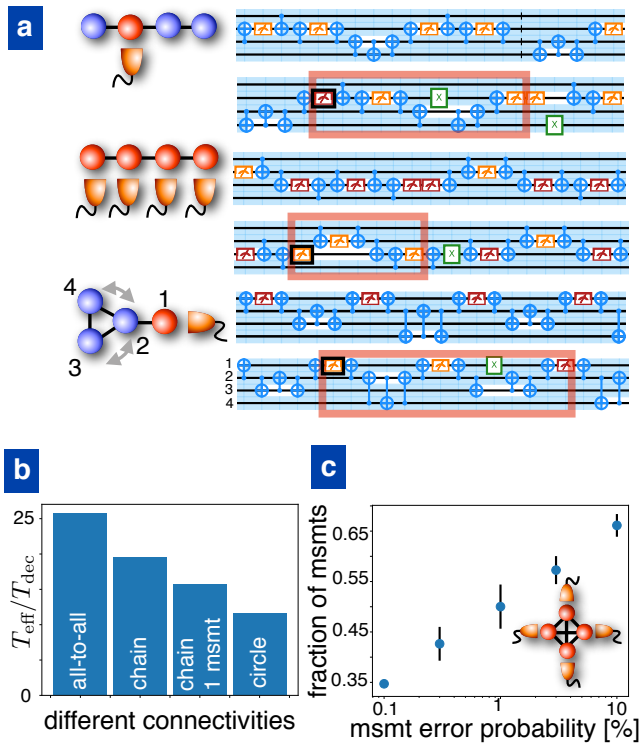

Figure 4. (a) Scenarios with different qubit connectivity. CNOTs are allowed only between qubits connected by a line. In each case, we display the "standard" gate sequence discovered by the net, as well as a sequence involving corrective actions (triggered by an unexpected measurement). From top to bottom: 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 has to be pinpointed (see main text). These nontrivial detection/recovery sequences are considerably more complex than the periodic detection cycle. They are a-priori unkown, and RL permits to discover them from scratch without extra input. (b) The effective enhancement of the decoherence time via error correction, for the different scenarios. Here, $T_{\rm dec} = 1200$ is the single-qubit decoherence time (in units of the gate time that defines the time step), and $T_{\rm eff}$ was 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 measurement error. The network discovers that redundancy is needed, i. e. the number of measurements in the gate sequences increases (in this plot, from 1 per cycle to about 6).

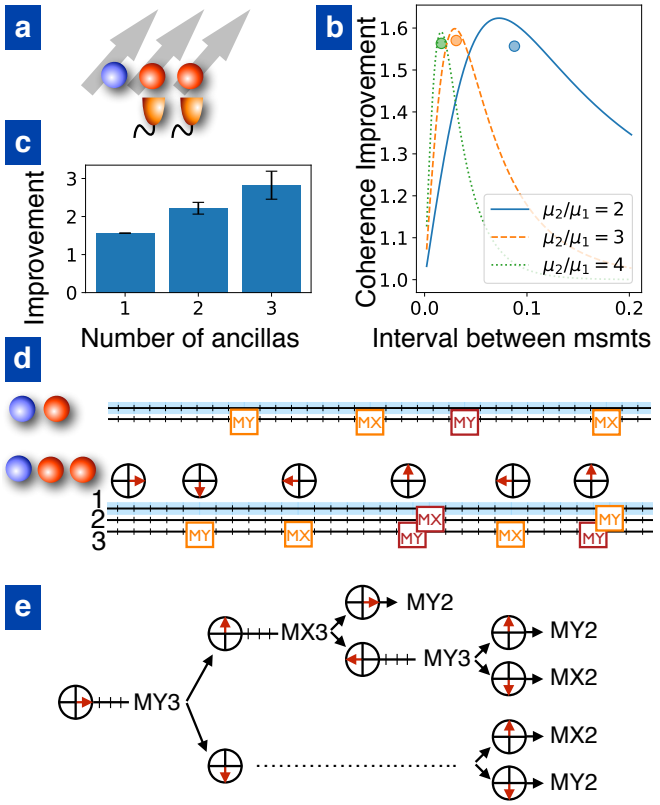

Figure 5. Countering dephasing by ancilla measurements. (a) The setting: a data qubit, whose phase undergoes a random walk due to a fluctuating field. Since the field is spatially correlated, its fluctuations can be detected by nearby ancilla qubits, and the measurement results can then be used to feedback on the data qubit and correct (to some extent) the accumulated random phase. This works even in situations where both dynamical decoupling and decoherence-free subspaces fail (because the noise is uncorrelated in time and the ancillas couple to the noise with arbitrary strengths). (b) For one ancilla qubit, the network performs periodically repeated ancilla measurements. It finds the optimal measurement interval. This can be seen by comparing the network (circles) with the analytical predictions (curves) of the coherence time enhancement as a function of the interval (in units of the single-qubit decoherence time $T_{\rm single}$). 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 coupling ratio (indicated here). (c) Different scenarios, classified according to the number of ancilla qubits ($\mu_2 = \mu_3 = \mu_4 = 4$). (d) Gate sequences indicate that for two ancilla qubits the network starts to implement an adaptive strategy, where measurements (MX, MY: along x,y) on the second ancilla are less frequent and the measurement basis is decided upon the parity of previous measurements of the first ancilla. (e) The 2-ancilla adaptive measurement sequence. For 2 ancillas, a brute-force search for strategies is still (barely) possible, but for higher ancilla numbers it becomes infeasible due to the exponentially large search space of adaptive strategies (also time-intervals between measurements can vary). The arrows always show the measurement result for qubit #3. [$\mu_2 = 3.8$, $\mu_3 = 4.1$ in (d),(e)]

CONCLUSIONS

We have seen how a network can discover quantum error correction techniques from scratch. It finds a-priori unknown nontrivial detection/recovery sequences for diverse settings without any more input than 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 6 qubits or more, where full protection against all noise sources and eventually multiple logical qubits could be realized. There is still considerable leeway in improving the speed of the physics simulation and of GPU-based training, which should enable this.

On the machine learning side, other RL schemes can be substituted for policy gradient, 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 could also be applied in other RL scenarios, where one would first train based on expanded state information. In general, we have shown 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.

Note added: Shortly before submission of the present manuscript, a preprint [39] appeared exploring reinforcement learning with recurrent networks for optimal quantum control (without feedback) .

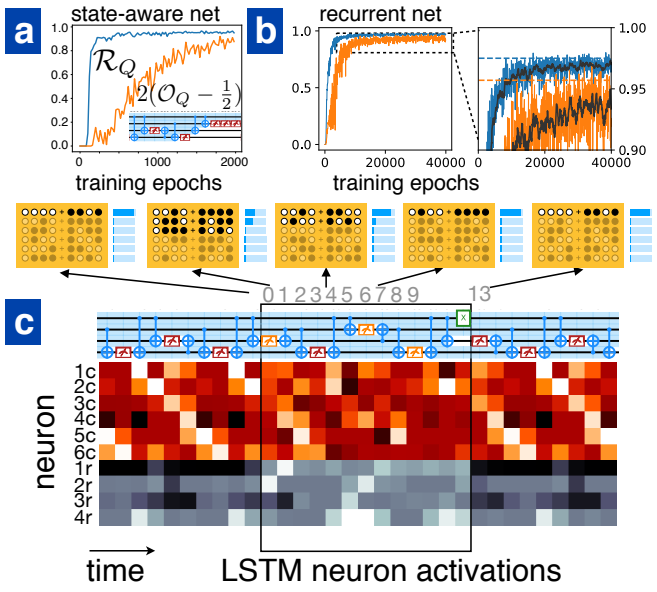

Figure 6. (a) Training the state-aware network to implement decoding back into the original data qubit, at the end of a 200-step gate sequence. Blue: recoverable quantum information $\mathcal{R}_Q$. Orange: (rescaled, shifted) overlap $\mathcal{O}_Q$ between final and initial state of the data qubit. Inset displays the decoding gate sequence. (b) Training the recurrent network in a supervised way by imitating the state-aware network. Again, we show $\mathcal{R}_Q$ and $\mathcal{O}_Q$ evolving during training. (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 activation). 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. Neuron 1r obviously keeps track of whether recovery is ongoing, while 3r acts like a counter keeping time during the standard periodic detection pattern. The neurons whose behavior changes markedly during recovery are depicted in gray.

METHODS

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 $\hat{\rho}_{\vec{n}}(0) = \frac{1}{2}(1 + \vec{n}\hat{\vec{\sigma}}) \otimes \hat{\rho}_{\text{Rest}}$, factorizing out the (fixed) state of all the other qubits. Now consider the four quantities

$$\hat{\rho}_0(0) = \frac{1}{2}\big(\hat{\rho}_{\vec{e}_j}(0) + \hat{\rho}_{-\vec{e}_j}(0)\big) \tag{M1a}$$

$$\delta\hat{\rho}_j(0) = \frac{1}{2}\big(\hat{\rho}_{\vec{e}_j}(0) - \hat{\rho}_{-\vec{e}_j}(0)\big) \tag{M1b}$$

where $j \in \{\mathrm{x}, \mathrm{y}, \mathrm{z}\}$ and $\vec{e}_j$ are the basis vectors; note that the right-hand side of Eq. (M1a) is independent of j. $\hat{\rho}_0$ and the $\delta\hat{\rho}_j$ are evolved stepwise, for each time-interval $[t_\mathrm{i}, t_\mathrm{f}]$ according to the update rule

$$\hat{\rho}_0(t_\mathrm{f}) = \frac{\phi[\hat{\rho}_0(t_\mathrm{i})]}{\mathrm{tr}(\phi[\hat{\rho}_0(t_\mathrm{i})])} \tag{M2a}$$

$$\delta\hat{\rho}_j(t_\mathrm{f}) = \frac{\phi[\delta\hat{\rho}_j(t_\mathrm{i})]}{\mathrm{tr}(\phi[\hat{\rho}_0(t_\mathrm{i})])} \ . \tag{M2b}$$

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 $\mathrm{tr}(\hat{\rho}_0(t)) = 1$. $\hat{\rho}_0$ and the $\delta\hat{\rho}_j$ give us access to the density matrix for every logical qubit state, at any time t:

$$\hat{\rho}_{\vec{n}}(t) = \frac{\hat{\rho}_0(t) + \sum_j n_j\, \delta\hat{\rho}_j(t)}{1 + \mathrm{tr}(\sum_j n_j\, \delta\hat{\rho}_j(t))} \tag{M3}$$

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 $\phi[\hat{\rho}] = \mathrm{e}^{\Delta t \mathcal{D}}(\hat{U}\hat{\rho}\hat{U}^\dagger)$ for

unitary operations $\hat{U}$ and $\phi[\hat{\rho}] = \mathrm{e}^{\Delta t \mathcal{D}}(\hat{P}\hat{\rho}\hat{P})$ for projection operators $\hat{P}$ where $\mathcal{D}$ is the dissipative part of the Liouvillian (we only consider Markovian noise). Specifically, we use two different error models, the bit-flip error (bf) and the correlated noise error (cn):

$$\mathcal{D}_{\mathrm{bf}}\hat{\rho} = T_{\mathrm{dec}}^{-1} \sum_n \hat{\sigma}_x^{(n)} \hat{\rho} \hat{\sigma}_x^{(n)} - \hat{\rho} \tag{M4a}$$

$$\mathcal{D}_{\mathrm{cn}}\hat{\rho} = T_{\mathrm{dec}}^{-1} \left(\hat{L}_{\mathrm{cn}} \hat{\rho} \hat{L}_{\mathrm{cn}}^\dagger - \frac{1}{2} \left\{ \hat{L}_{\mathrm{cn}}^\dagger \hat{L}_{\mathrm{cn}}, \hat{\rho} \right\} \right) \tag{M4b}$$

where $\hat{\sigma}_x^{(n)}$ applies the corresponding Pauli operator to the nth qubit and $\hat{L}_{\mathrm{cn}} = \frac{1}{\sqrt{\sum_n \mu_n^2}} \sum_n \mu_n \hat{\sigma}_z^{(n)}$. Here, μ_n denotes the coupling of qubit n to the noise. Note that in the bit-flip scenario the single qubit decay time $T_{\mathrm{single}} = T_{\mathrm{dec}}$, whereas in the presence of correlated noise it is $T_{\mathrm{single}} = T_{\mathrm{dec}}/\mu_1^2$.

Recoverable quantum information Based on Eq. (M3), $\mathcal{R}_{\mathrm{Q}}$ as introduced in the main text can be written as

$$\mathcal{R}_{\mathrm{Q}}(t) = \min_{\vec{n}} \left\| \sum_j n_j \, \delta\hat{\rho}_j(t) \right\|_1 \tag{M5}$$

in the (for us relevant) case that $\mathrm{tr}(\delta\hat{\rho}_j(t)) = 0$ 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. In the bit-flip scenarios, the anti-commutator relation $\{\delta\hat{\rho}_j, \delta\hat{\rho}_k\} = 0$ is satisfied for all distinct $j, k \in \{\mathrm{x}, \mathrm{y}, \mathrm{z}\}$; it can be shown that this restricts the minimum to lie along one of the coordinate axes: $\min_{\vec{n}} \| \sum_j n_j \, \delta\hat{\rho}_j(t)\|_1 = \min_{j\in\{\mathrm{x},\mathrm{y},\mathrm{z}\}} \|\delta\hat{\rho}_j(t)\|_1$. 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 $\mathcal{R}_{\mathrm{Q}}$ may vary between the different measurement results. To obtain a measure that does not depend on this, we introduce $\bar{\mathcal{R}}_{\mathrm{Q}}$ as the average over all possible values of $\mathcal{R}_{\mathrm{Q}}$ (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 $\bar{\mathcal{R}}_{\mathrm{Q}} = \mathcal{R}_{\mathrm{Q}}$.

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

$$r_t = \begin{cases} 1 + \frac{\bar{\mathcal{R}}_{\mathrm{Q}}(t+1) - \mathcal{R}_{\mathrm{Q}}(t)}{2\Gamma_{\mathrm{triv}}\Delta t} & +0 \quad \text{if } \bar{\mathcal{R}}_{\mathrm{Q}}(t+1) > 0 \\ 0 & -P \quad \text{if } \bar{\mathcal{R}}_{\mathrm{Q}}(t) \neq 0 \\ & \quad \wedge \bar{\mathcal{R}}_{\mathrm{Q}}(t+1) = 0 \\ 0 & +0 \quad \text{if } \bar{\mathcal{R}}_{\mathrm{Q}}(t) = 0 \end{cases} \tag{M6}$$

$$\underbrace{\qquad\qquad}_{=:r_t^{(1)}} \quad \underbrace{\qquad}_{=:r_t^{(2)}}$$

with $\mathcal{R}_{\mathrm{Q}}$ and $\bar{\mathcal{R}}_{\mathrm{Q}}$ as defined above, T_{single} the decay time for encoding in one physical qubit only ("trivial" encoding), Δt the time step, and P an extra 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

$$R_t = (1-\gamma) \Big(\sum_{k=0}^{T-t-1} \gamma^k r_{t+k}^{(1)} \Big) + r_t^{(2)} \tag{M7}$$

where γ is the return discount rate; for more information on the (discounted) return, see e. g. [40].

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:

$$r_t^{(\mathrm{recov})} = \beta_{\mathrm{dec}} \left(D_{t+1} - D_t \right) + \beta_{\mathrm{corr}} \, C_T \, \delta_{t,T-1} \tag{M8}$$

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 only rewarded after T_{signal}. We set $I_t^{(q)} = 1$ if $(\mathrm{tr}_{\bar{q}}(\delta\hat{\rho}_{\mathrm{x}}(t)) \neq 0) \vee (\mathrm{tr}_{\bar{q}}(\delta\hat{\rho}_y(t)) \neq 0) \vee (\mathrm{tr}_{\bar{q}}(\delta\hat{\rho}_z(t)) \neq 0)$, and otherwise $I_t^{(q)} = 0$. $\mathrm{tr}_{\bar{q}}$ denotes the partial trace over all qubits but 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 return, we use

$$R_t^{(\mathrm{recov})} = (1-\gamma) \sum_{k=0}^{T-t-1} \gamma^k r_{t+k}^{(\mathrm{recov})} \tag{M9}$$

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

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

Input of the state-aware network The core of the input to the state-aware network is a representation of the density matrices $\hat{\rho}_0$, $\hat{\rho}_1 := \hat{\rho}_0 + \delta\hat{\rho}_{\mathrm{x}}$, $\hat{\rho}_2 := \hat{\rho}_0 + \delta\hat{\rho}_{\mathrm{y}}$, and $\hat{\rho}_3 := \hat{\rho}_0 + \delta\hat{\rho}_{\mathrm{z}}$. 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 $\hat{\rho}_j = \sum_n p_n \, |\phi_n\rangle \langle\phi_n|$ and select the eigenstates $|\phi_n\rangle$ with the largest eigenvalues p_n. To include also the eigenvalue in the input, we feed all components of the scaled states $|\tilde{\phi}_n\rangle := \sqrt{p_n} \, |\phi_n\rangle$ (which yield $\hat{\rho}_j = \sum_n |\tilde{\phi}_n\rangle \langle\tilde{\phi}_n|$) into the network, where the states are in addition sorted by their eigenvalue. For our simulations, we select the 6 largest components, so we need $768 = 4 \cdot 6 \cdot 16 \cdot 2$ input neurons (4 density matrices, 16 is the dimension of the Hilbert space, 2 for real and imaginary part).

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 $j \in \{\mathrm{x}, \mathrm{y}, \mathrm{z}\}$ 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 could 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 have found 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 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.

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 the last section of the Methods). All neighboring layers are densely connected, the activation function is the rectified linear unit (ReLU). At the output layer, the softmax function $x_j \mapsto \mathrm{e}^{x_j} / \sum_k \mathrm{e}^{x_k}$ is applied such that the result can be interpreted as a probability distribution.

Reinforcement learning of the state-aware network Our learning scheme is based on the policy gradient algorithm [33]. The full expression for our learning gradient (indicating the change in θ) reads

$$\begin{aligned} g =& F^{-1} \cdot \mathbb{E}\Big[\sum_t (R_t - b_t) \frac{\partial}{\partial\theta} \ln \pi_\theta(a_t|s_t) \Big] \\ & - \lambda\, \mathbb{E}\left[\sum_a \frac{\partial}{\partial\theta} \big[\pi_\theta(a|s) \ln \pi_\theta(a|s)\big] \right] \end{aligned} \tag{M10}$$

where R_t is the (discounted) return (cmp. Eqs. (M7) and (M9)). This return is corrected by an (explicitly time-dependent) baseline b_t which we choose as 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 [41–43] by multiplying F^{-1}, the (Moore–Penrose) inverse of the Fisher information matrix $F = \mathbb{E}[(\frac{\partial}{\partial\theta} \ln \pi_\theta(a|s))(\frac{\partial}{\partial\theta} \ln \pi_\theta(a|s))^\mathsf{T}]$. The second term is entropy regularization [44]; we use it only to train the state-aware network shown in Fig. 6. As update rule, we use adaptive moment estimation (Adam [45]).

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 was 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 layer are connected by two successive long short-term memory (LSTM) layers [38] with tanh inter-layer activations. After the output layer, the softmax function is applied (like for the state-aware network).

Supervised learning of the recurrent network The training data is generated from inference of a state-aware network which has been 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 this data. It is possible to generate enough data such that overfitting is not a concern (for the example in Fig. 6, each trajectory is reused only 5 times during the full training process). For the actual training of the recurrent network, we use supervised learning with categorical cross-entropy as cost function (q is the actual policy of the recurrent network to train, and p the desired policy from the state-aware network):

$$C(q,p) = -\sum_a p(a) \ln q(a) \tag{M11}$$

Due to the LSTM layers, it is necessary to train on full trajectories (in the true time sequence) instead of individual actions. Dropout [46] is used for regularization. The training update rule is adaptive moment estimation (Adam [45]).

Physical parameters and hyperparameters The physical parameters used throughout the main text are summarized in the following table. Times are always given in units of the time step (gate time).

Physical parameters	
Figs. 2, 3, 4, 6 decoherence time T_{dec}	1200
Fig. 5 single qubit decoherence time $T_{\mathrm{single}} = T_{\mathrm{dec}}/\mu_1^2$	500
Figs. 2, 3, 4, 6 number of time steps T	200
Fig. 5 number of time steps T	100

We have used a few separately trained neural network

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, but instead shows that our results are robust. Such hyperparameters choices can have slight impact on the training time etc. but the strategies found by the neural networks were equivalent. In the following table, we summarize the architecture of the networks, i. e., we list the number of neurons in each layer.

Network architectures	**Neurons per layer**
Fig. 2	(793, 300, 300, 21)
Fig. 3	(793, 300, 300, 300, 21)
Fig. 4a-Fig. 4b all-to-all connected:	(793, 300, 300, 21)
Fig. 4a-Fig. 4b chain:	(787, 300, 300, 300, 15)
Fig. 4a-Fig. 4b chain with one msmt qubit:	(781, 600, 300, 12)
Fig. 4a-Fig. 4b circle:	(783, 300, 300, 300, 14)
Fig. 4c (all identical)	(793, 300, 300, 300, 21)
Fig. 5b-Fig. 5d: one ancilla	(101, 100, 50, 3)
Fig. 5c, Fig. 5d: two ancillas	(265, 100, 50, 5)
Fig. 5c: three ancillas	(781, 200, 100, 7)
Fig. 6a	(803, 300, 300, 21)
Fig. 6b-Fig. 6c (hidden layers with LSTM units)	(26, 128, 128, 21)

Each output neuron represents one action that can be performed by the agent and thus, the output layer size is equal to the number of actions. In the bit-flip scenarios (Figs. 2, 3, 4, 6), the actions are CNOTs according to connectivity, measurements along z as indicated in the corresponding sketches, bit flips on each qubit, and idle. When dealing with correlated noise, cf. Fig. 5, the available actions are instead measurements along x and y on all ancilla qubits, and the idle operation.

The hyperparameters used for training the state-aware networks are summarized in the following table:

Hyperparameters state-aware network	
training batch size	64
Adam parameters η, β_1, β_2	0.001, 0.9, 0.999
return discount rate γ	0.95
baseline discount rate κ	0.9
punishment reward coefficient P	40
decoding reward coefficient β_{dec}(only used in Fig. 6a)	20
correction reward coefficient β_{corr}(only used in Fig. 6a)	10
decoding signal time T_{signal} (only used in Fig. 6a)	$T_{\text{signal}} = T - 10 = 190$
entropy regularization λ(only used in Fig. 6a)	$\lambda = 5 \cdot 10^{-3}$ for the first 12 000 training epoches, $\lambda = 0$ afterwards

The hyperparameters used for training the recurrent network (cf. Fig. 6b and Fig. 6c) are summarized in the following table:

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 [47] (and Keras for defining networks).

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

[2] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach* (CreateSpace Independent Publishing Platform, 2016) google-Books-ID: PQI7vgAACAAJ.

[3] G. Carleo and M. Troyer, Science **355**, 602 (2017).

[4] J. Carrasquilla and R. G. Melko, Nat Phys **13**, 431 (2017).

[5] E. P. L. van Nieuwenburg, Y.-H. Liu, and S. D. Huber, Nature Physics **13**, 435 (2017).

[6] G. Torlai and R. G. Melko, Physical Review Letters **119**, 030501 (2017).

[7] M. August and X. Ni, Physical Review A **95** (2017), 10.1103/PhysRevA.95.012335, arXiv: 1604.00279.

[8] V. Dunjko and H. J. Briegel, arXiv:1709.02779 [quant-ph] (2017), arXiv: 1709.02779.

[9] P. Baireuther, T. E. O'Brien, B. Tarasinski, and C. W. J. Beenakker, arXiv:1705.07855 [cond-mat, physics:quant-ph] (2017), arXiv: 1705.07855.

[10] S. Krastanov and L. Jiang, Scientific Reports **7**, 11003 (2017).

[11] 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, Science **324**, 85 (2009).

[12] M. Schmidt and H. Lipson, Science **324**, 81 (2009).

[13] 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, Nature **518**, 529 (2015).

[14] 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. v. d. Driessche, T. Graepel, and D. Hassabis, Nature **550**, 354 (2017).

[15] M. Bukov, A. G. R. Day, D. Sels, P. Weinberg, A. Polkovnikov, and P. Mehta, arXiv:1705.00565 [cond-mat, physics:quant-ph] (2017), arXiv: 1705.00565.

[16] A. A. Melnikov, H. P. Nautrup, M. Krenn, V. Dunjko, M. Tiersch, A. Zeilinger, and H. J. Briegel, Proceedings of the National Academy of Sciences , 201714936 (2018).

[17] 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,

Nature **432**, 602 (2004).
[18] P. Schindler, J. T. Barreiro, T. Monz, V. Nebendahl, D. Nigg, M. Chwalla, M. Hennrich, and R. Blatt, Science **332**, 1059 (2011).
[19] 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, Nature **506**, 204 (2014).
[20] 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, A. N. Cleland, and J. M. Martinis, Nature **519**, 66 (2015).
[21] 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, Nature **526**, 410 (2015).
[22] 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, Nature **536**, 441 (2016).
[23] 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, arXiv:1710.07466 [physics, physics:quant-ph] (2017), arXiv: 1710.07466.
[24] S. Debnath, N. M. Linke, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe, Nature **536**, 63 (2016).
[25] M. Takita, A. W. Cross, A. D. Córcoles, J. M. Chow, and J. M. Gambetta, Physical Review Letters **119**, 180501 (2017).
[26] T. F. Watson, S. G. J. Philips, E. Kawakami, D. R. Ward, P. Scarlino, M. Veldhorst, D. E. Savage, M. G. Lagally, M. Friesen, S. N. Coppersmith, M. A. Eriksson, and L. M. K. Vandersypen, arXiv:1708.04214 [cond-mat, physics:quant-ph] (2017), arXiv: 1708.04214.
[27] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbrüggen, and S. J. Glaser, Journal of Magnetic Resonance **172**, 296 (2005).
[28] S. Machnes, U. Sander, S. J. Glaser, P. de Fouquières, A. Gruslys, S. Schirmer, and T. Schulte-Herbrüggen, Physical Review A **84**, 022305 (2011).
[29] P. D. Johnson, J. Romero, J. Olson, Y. Cao, and A. Aspuru-Guzik, arXiv:1711.02249 [quant-ph] (2017), arXiv: 1711.02249.
[30] A. Hentschel and B. C. Sanders, Physical Review Letters **104**, 063603 (2010).
[31] P. Palittapongarnpim, P. Wittek, E. Zahedinejad, S. Vedaie, and B. C. Sanders, arXiv:1607.03428 [quant-ph, stat] (2016), arXiv: 1607.03428.
[32] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Nature **549**, 195 (2017).
[33] R. J. Williams, Machine Learning **8**, 229 (1992).
[34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (The MIT Press, Cambridge, Massachusetts, 2016).
[35] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, anniversary edition ed. (Cambridge University Press, Cambridge ; New York, 2011).
[36] B. M. Terhal, Reviews of Modern Physics **87**, 307 (2015).
[37] L. v. d. Maaten and G. Hinton, Journal of Machine Learning Research **9**, 2579 (2008).
[38] S. Hochreiter and J. Schmidhuber, Neural Computation **9**, 1735 (1997).
[39] M. August and J. M. Hernández-Lobato, arXiv:1802.04063 [quant-ph] (2018), arXiv: 1802.04063.
[40] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, Vol. 1 (MIT press Cambridge, 1998).
[41] S.-i. Amari, Neural Computation **10**, 251 (1998).
[42] S. M. Kakade, in *Advances in neural information processing systems* (2002) pp. 1531–1538.
[43] J. Peters and S. Schaal, Neurocomputing **71**, 1180 (2008).
[44] R. J. Williams and J. Peng, Connection Science **3**, 241 (1991).
[45] D. Kingma and J. B. Adam, in *International Conference for Learning Representations*, Vol. 6 (2015).
[46] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, arXiv preprint arXiv:1207.0580 (2012).
[47] R. Al-Rfou et al., arXiv e-prints **abs/1605.02688** (2016).