Classifying global state preparation via deep reinforcement learning

IOP Publishing Mach. Learn.: Sci. Technol. 2 (2021) 01LT02 https://doi.org/10.1088/2632-2153/abc81f

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
25 June 2020

REVISED
20 October 2020

ACCEPTED FOR PUBLICATION
5 November 2020

PUBLISHED
24 December 2020

LETTER

Classifying global state preparation via deep reinforcement learning

Tobias Haug[1], Wai-Keong Mok[1,2], Jia-Bin You[2], Wenzu Zhang[2], Ching Eng Png[2] and Leong-Chuan Kwek[1,3,4,5]

[1] Centre for Quantum Technologies, National University of Singapore, 3 Science Drive 2, Singapore 117543, Singapore
[2] Department of Electronics and Photonics, Institute of High Performance Computing, 1 Fusionopolis Way, 16-16 Connexis, Singapore 138632, Singapore
[3] MajuLab, CNRS-UNS-NUS-NTU International Joint Research Unit, UMI 3654, Singapore
[4] National Institute of Education and Institute of Advanced Studies, Nanyang Technological University, 1 Nanyang Walk, Singapore 637616, Singapore
[5] School of Electrical and Electronic Engineering Block S2.1, 50 Nanyang Avenue, Singapore 639798, Singapore

E-mail: tobiasxhaug@gmail.com

Keywords: machine learning, quantum control, NV centers, quantum mechanics, deep reinforcement learning

Supplementary material for this article is available online

Abstract

Quantum information processing often requires the preparation of arbitrary quantum states, such as all the states on the Bloch sphere for two-level systems. While numerical optimization can prepare individual target states, they lack the ability to find general control protocols that can generate many different target states. Here, we demonstrate global quantum control by preparing a continuous set of states with deep reinforcement learning. The protocols are represented using neural networks, which automatically groups the protocols into similar types, which could be useful for finding classes of protocols and extracting physical insights. As application, we generate arbitrary superposition states for the electron spin in complex multi-level nitrogen-vacancy centers, revealing classes of protocols characterized by specific preparation timescales. Our method could help improve control of near-term quantum computers, quantum sensing devices and quantum simulations.

1. Introduction

Deep reinforcement learning has revolutionized computer control over complex games [1–3], which are notoriously difficult to optimize with established methods [3]. Recently, reinforcement learning has also been successfully applied to a wide array of physics problems [4–15]. A particularly promising application is optimizing quantum control problems [6, 8–10, 12, 16], which are of utmost importance to enable quantum technologies and quantum devices for information processing and quantum computation. Quantum systems are controlled by unitary operations engineered by a set of physical operations on the quantum system, which we refer to as a protocol. To efficiently manipulate quantum sensing devices and programmable quantum computers, a large set of different unitary operations and protocols has to be mastered. This is in general a non-trivial task. For example, to generate an arbitrary quantum state in a two-level system, a two-dimensional set of protocols has to be learned. Of course, for specific types of driving the parameterization is well-known (e.g. in terms of rotations around the Bloch sphere) and there is a simple understanding of how to control the spin dynamics on the Bloch sphere. However, in higher-dimensional quantum systems or systems where the driving is constrained, no general description is available, which is a limiting factor in the control of quantum devices. Standard numerical tools available can find control protocols for individual target quantum states, however it is difficult to find a class of protocols that can parameterize all the protocols as it can be highly disordered with many near-optimal solutions [5]. Thus, we ask: how can one find classes of solutions to generate arbitrary states in higher-dimensional quantum systems and/or with constrained driving parameters? And can they be used to extract physical insights?

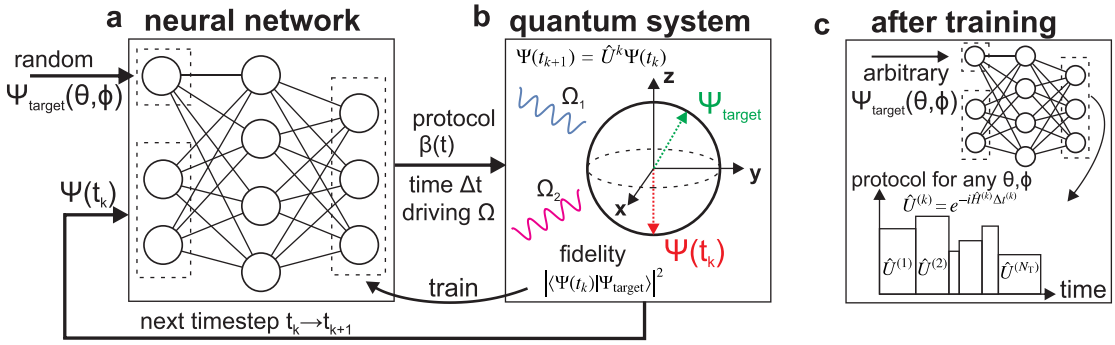

Figure 1. Overview of deep reinforcement algorithm to learn protocols to generate any target states. (a), (b) Learning scheme to generate piece-wise constant control protocol with proximal policy optimization (PPO, details in method section): at the first iteration step, a random target state Ψ_{target} (parameterized by angles θ, ϕ) and the initial state $\Psi(t_0)$ is chosen. At iteration step $n+1$ of the algorithm, the neural network takes in Ψ_{target} and the current wavefunction $\Psi(t_n)$. The output of the neural network generates the driving parameters (timestep Δt, driving strength Ω) to manipulate the quantum system to reach $\Psi(t_{n+1})$, which is fed back to the neural network. These steps are repeated until final step N_{T}. The fidelity of the generated state with the target state is calculated and used to update the weights of the neural network. For training, proximal policy optimization (PPO) uses two separate neural networks, called policy and value network. The policy network decides the next driving parameter, whereas the value network estimates the quality of that choice [17]. This concludes one epoch of the training protocol. Over many training epochs, the neural network learns the control protocols for arbitrary target states parameterized with θ, ϕ. (c) After the training, the neural network returns optimized piece-wise constant control protocols to reach any target state Ψ_{target}.

Here, instead of finding the optimal driving protocol for a single state, we propose a scalable method to learn all the driving protocols for global state preparation (over the continuous two-dimensional subspace represented by the Bloch sphere, embedded in a higher-dimensional Hilbert space) using deep reinforcement learning and parameterizing the protocols with neural networks. We discover that this approach automatically finds clusters of similar protocols. For example, it groups protocols according to how much time is needed to generate specific target states, which could be used to identify patterns and physical constraints in the protocols. For this multi-dimensional problem, conventional optimization often finds uncorrelated protocols in the target space, especially if there are multiple distinct protocols achieving similar fidelity. This makes it very difficult to interpolate between the protocols. Using our approach, the clustering of similar protocols is a consequence of effective interpolation by the neural network of nearby protocols within the same cluster, thereby achieving effective arbitrary state preparation. The neural network is trained with random target states as input. This makes it naturally suited for parallelization and could allow it to scale to higher dimensions. In essence, our proposal shows a path to solve the key limitations of single-state learning or a transfer-learning approach as shown in [12].

As a demonstration of our method, we apply it to control the electron spin in multi-level nitrogen-vacancy (NV) centers [18–21]. The multiple levels give rise to complicated coherent and dissipative dynamics, thus ruling out simple driving protocols such as those in two-level systems (e.g. quantum dot, transmon) and further increasing the difficulty of optimization. The electron spin triplet ground states are characterized by a long lifetime which makes them ideal candidates for solid-state qubits used in quantum information processing [22–24]. However these states cannot be coupled directly using lasers, which would be important for many applications [25, 26]. Optical control can instead be achieved by indirect optical driving via a manifold of excited states, which renders finding fast control protocols a difficult problem. With our algorithm we find protocols to prepare arbitrary superposition states with a preparation speed of about half a nanosecond, which is much faster than protocols based on STIRAP control [19] and reduces the impact of dissipation on the state preparation. Additionally, our protocols require only nine steps, which is considerably less than in comparable approaches [13, 21]. More importantly, our algorithm finds near-optimal protocols that are automatically grouped into distinct classes by the neural network. We find that the protocols within a class have similar driving protocols as well as similar total protocols times T. Each class prepares a subset of all the target states on the Bloch sphere with high fidelity.

2. Learn global control protocols

A quantum system is evolved by a unitary operator $\hat{U}$, which takes a quantum state Ψ to the final state $\Psi' = \hat{U}\Psi$. The unitary is physically realized by evolving with the Hamiltonian $\hat{H}(\Omega)$, parameterized by controllable parameters Ω, over a time t with $\hat{U} = \exp(-i\hat{H}t)$. We can engineer more sophisticated unitaries by piece-wise applying N_{T} different unitaries $\hat{U} = \hat{U}^{(N_{\text{T}})}\hat{U}^{(N_{\text{T}}-1)}\ldots\hat{U}^{(2)}\hat{U}^{(1)}$, each unitary given by $\hat{U}^{(k)} = \exp(-i\hat{H}(\Omega^{(k)})\Delta t^{(k)})$ with driving strength $\Omega^{(k)}$ and timestep $\Delta t^{(k)}$. To apply the unitary on a

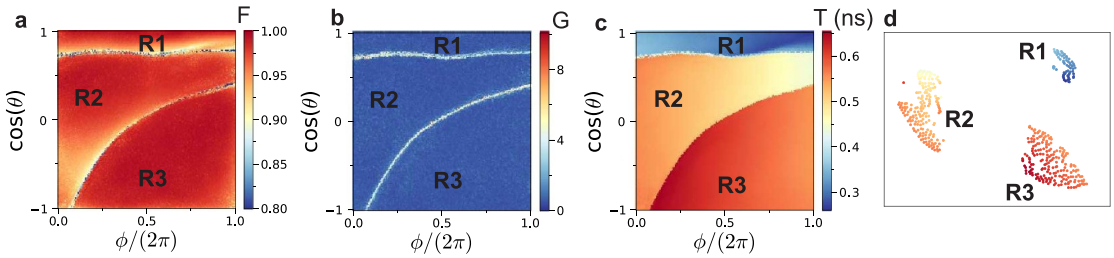

Figure 2. Create arbitrary quantum state $\Psi(\theta,\phi)$ (equation (2); states parameterized by angles θ, ϕ) using deep reinforcement learning. (a) Fidelity F (equation (1)) of preparing state (mean fidelity $\langle F\rangle = 0.972$). (b) Protocol gradient $G(\theta,\phi)$ (equation (3)). Protocols are grouped into three distinct areas of low gradient (areas numbered with R1 to R3), divided by a sharp lines of large gradient. (c) Protocol time T. The three areas are characterized by different protocol times T. (d) Two-dimensional representation of the distance between protocols (determined by equation (4)) using t-SNE algorithm. Color indicates protocol time. Protocols are close to each other if they are similar. Corresponding regions to the gradient plot in (b) are marked with R. Parameters: $N_T = 9$ timesteps, variable time per step with maximal time $0.2\,\text{ns} < T < 0.8\,\text{ns}$, $-20\,\text{GHz} < \Omega_{1,2} < 20$ GHz, detuning $\delta_1 = 50$ GHz, $\delta_2 = 0$ and $B_{\text{ext}} = 0.15\text{T}$ (all variables in units of $\hbar$), 800 000 training epochs and $n = 600$ neurons in two fully-connected layers. Using approximated closed system dynamics (result for full open system in the supplementary materials).

physical system, a total time $T = \sum_k \Delta t^{(k)}$ is needed. The minimal time needed to perform a specific task may vary depending on the target unitary and the driving Hamiltonian [27]. We define a protocol as a vector $\beta = \{\Omega^{(1)}, \Delta t^{(1)}, \ldots, \Omega^{(N_T)}, \Delta t^{(N_T)}\}$ that contains all the control parameters to generate a specific target state Ψ_{target}.

We now outline the algorithm to generate control protocols $\beta_{\theta,\phi}(t)$ to create all possible target states $\Psi_{\text{target}}(\theta,\phi)$ parametrized by angles θ and ϕ (see figure 1). At time t_k the input to the neural network a randomly sampled target state Ψ_{target} and a description of the system state (e.g. the current wavefunction $\Psi(t_k)$). The neural network output determines the parameters (driving strength $\Omega^{(k)}$, timestep $\Delta t^{(k)}$) for the piece-wise constant protocol used to evolve the wavefunction to the new state $\Psi(t_{k+1}) = \hat{U}^{(k)}\Psi(t_k)$ with unitary $\hat{U}^{(k)}$ given by $\hat{U}^{(k)} = \exp(-i\hat{H}(\Omega^{(k)})\Delta t^{(k)})$. This new wavefunction is then input to the neural network to determine the next protocol step. This is repeated until the final timestep N_T, which concludes one training episode. The goal is to drive the system such that wavefunction is as close as possible to a target state Ψ_{target}, measured by maximizing the fidelity:

$$F = \left|\langle \Psi(t_{N_T}) | \Psi_{\text{target}} \rangle\right|^2, \tag{1}$$

as a reward function. With each training episode, the neural network is trained with randomly sampled target states $\Psi_{\text{target}}(\theta,\phi)$ as input, until it converges and learns to represent protocols for all possible target states. The input to the neural network is vector of real numbers. To generate the protocol, the neural network is fed with information about the current state of the system at a specific timestep as well as the target state to be achieved. The neural network is given the angles θ and ϕ, which parametrize the target state. Further, the current wavefunction $\Psi(t_k)$ is fed to the neural network with the real and imaginary parts of the probability amplitudes of each quantum level.

The neural network is trained using PPO [17]. The neural network is composed of two parts: the critic estimates at every step the final quality of the protocol to optimize itself, while the actor at each step returns a part of the protocol. We use the PPO algorithm implemented by the OpenAI Spinning Up library [28]. Similar results can be achieved using the numerically more complex Trust Region Policy Optimization method (see results in supplementary materials (available online at stacks.iop.org/MLST/2/01LT02/mmedia)) [29].

3. Electron spin control in NV center

The goal is to achieve coherent control between the states $|-1\rangle$ and $|+1\rangle$ of the triplet ground state manifold $\{|-1\rangle, |0\rangle, |+1\rangle\}$ via coupling to a manifold of excited states. Starting from the ground state $\Psi_0 = |-1\rangle$, we would like to achieve a general superposition state of the Bloch sphere spanned by the two states

$$\Psi(\theta,\phi) = \cos\left(\frac{\theta}{2}\right)|-1\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|+1\rangle\,, \tag{2}$$

where $\theta \in \{0,\pi\}$ and $\phi \in \{0,2\pi\}$. These two degenerate ground states are not coupled directly. The degeneracy can be lifted with an external magnetic field B_{ext}. To coherently control them, we couple the ground states via two lasers to the excited state manifold $\{A_2, A_1, E_X, E_Y, E_1, E_2\}$, which is separated in energy

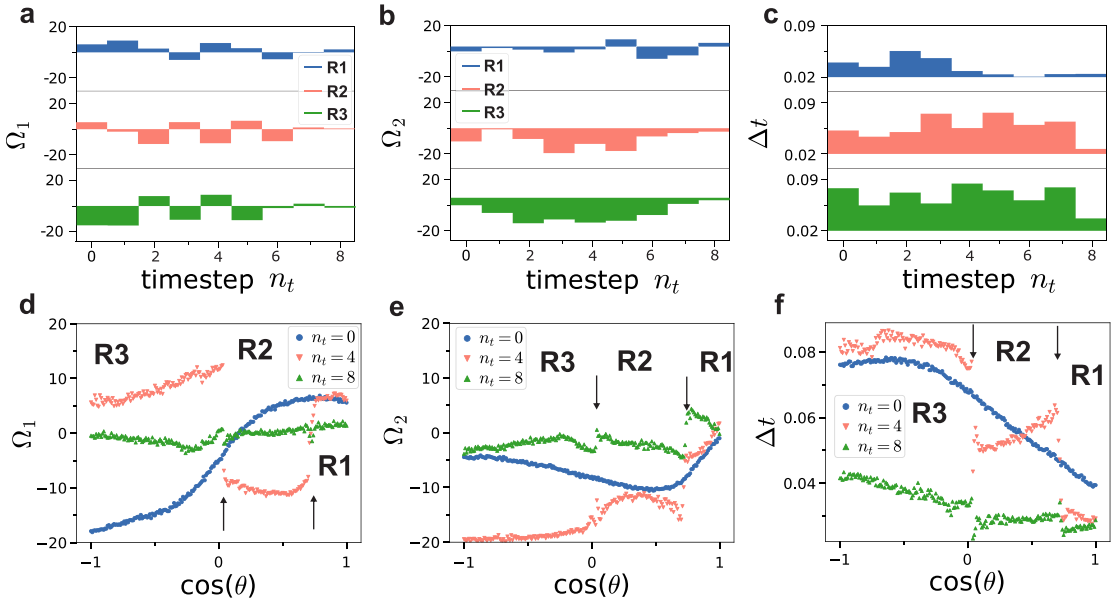

Figure 3. Examples of optimized driving protocols for target states calculated in figure 2. Driving protocols at step k are parameterized by driving strength Ω_1^k, Ω_2^k and timestep length Δt^k. (a)–(c) Example protocols for three representative examples from different protocol 'phases' for driving parameter: (a) Ω_1^k, (b) Ω_2^k, (c) timestep length Δt^k. Examples taken from regions as defined in figure 2, for R1: $\cos(\theta) = 0.9$, $\phi = \pi$, R2: $\cos(\theta) = 0.5$, $\phi = \pi$, R3: $\cos(\theta) = -0.5$, $\phi = \pi$. (d)–(f) Driving protocol for cut along θ with $\phi = \pi$ for different protocol timesteps n_t and driving parameters (d) Ω_1, (e) Ω_2, (f) Δt. Protocol show distinct 'phases' that jump at $\cos(\theta) \approx 0$ and $\cos(\theta) \approx 0.75$. See supplementary materials for further example protocols.

from the ground state triplet within the optical frequency range. Through dissipative couplings (see Methods), a further metastable state can be occupied, thus the NV center is described by an open ten-level system. We investigate the limit where the protocol time is much faster than the dissipation, thus we can approximately treat the system as an effective closed eight-level system (comparison with full dynamics in the supplementary materials). We apply two driving lasers, with time-dependent strengths chosen by the protocol $\Omega_1(t)$, $\Omega_2(t)$. They have a relative detuning δ_1, δ_2 to the energy difference of states $|-1\rangle$ ($|+1\rangle$) and $|A_2\rangle$ (see section 5). This system resembles the well-known Λ system with three levels, however with an additional complicated set of levels that has to be excited and controlled. The excited levels interact non-trivially with one another as well. For these systems, simple or analytic solutions are difficult to find, especially for the non-adiabatic regime considered here.

The goal is to learn protocols to reach arbitrary superposition states of the two levels parameterized by angles θ and ϕ. To evaluate how similar protocols are, we introduce the following measures: Firstly, to evaluate the local change of protocols, we define the norm of the protocol gradient in respect to the target parameters θ and ϕ:

$$G(\theta, \phi) = ||\partial_\theta \beta(\theta, \phi)||_1 + ||\partial_\phi \beta(\theta, \phi)||_1 , \tag{3}$$

where $\beta(\theta, \phi)$ is the protocol for a given θ and ϕ and $||.||_1$ the L_1 norm. This measures how much the protocol is changing across the Bloch sphere. Secondly, to measure how close two different protocols are, we define the protocol distance:

$$C(\theta, \theta', \phi, \phi') = ||\beta(\theta, \phi) - \beta(\theta', \phi')||_1 . \tag{4}$$

If two protocols are identical, the measure is zero. Else, the protocol distance is positive. Our protocols β are a vector of length $3N_T$ consisting of N_T timesteps with the two driving strengths $\Omega_1^{(k)}$, $\Omega_2^{(k)}$ and length of timestep $\Delta t^{(k)}$. To calculate the correlation measure and gradient, we map the parameters between -0.5 and 0.5.

The result after training is shown in figure 2. We see that the fidelity of reaching the target state is high in most areas, with very sharp lines of low fidelity (see figure 2(a) that divides it into three areas; marked as R1 to R3). The sharpness of these lines increases with number of neurons as the neural network can represent more complex features and abrupt changes. (Further demonstration of this feature on a simpler spin rotation problem in the supplementary materials (available online at stacks.iop.org/MLST/2/01LT02/mmedia).) At these lines, the protocol changes drastically, while it changes only slowly in the other areas. We visualize this

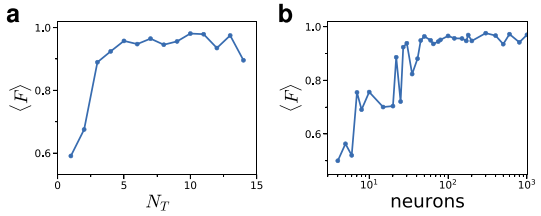

Figure 4. Fidelity by reinforcement learning trained protocols to generate state $\Psi(\theta, \phi)$ averaged over all θ,ϕ. (a) Average fidelity $\langle F \rangle$ against number of timesteps for $n = 150$ neurons. (b) Average fidelity against number of neurons for $N_T = 9$ timesteps. Same parameters as in figure 2.

with the protocol gradient equation (3) in figure 2(b). It measures how much the protocol changes with θ and ϕ. The gradient is low within those three distinct areas, where the protocol is changing only slowly and protocols are very similar. Protocols in these areas are very similar to each other. We observe specific sharp lines where the gradient is very large. These correlate with the lines of low fidelity in figure 2(a). These lines are borders between two different protocol classes. At the lines of low fidelity, the protocol changes drastically, as the neural network is trying to trying to find a protocol between the two classes. As an analogy, we describe this phenomena as a kind of phase diagram: the areas of low gradient represent 'phases' (class of protocols of specific type), that are separated lines of high gradient akin to a 'phase transition'. At these lines, the algorithm struggles to find good protocols. These 'phases' are very well visible in the protocol time T, shown in figure 2(c) with distinct areas of similar T. These directly reveal some physical insight into the state generation. For states with $\cos(\theta) \approx 1$ (close to initial state), fast protocols are sufficient. For increasing θ, protocols require a longer duration. Our algorithm automatically groups protocols into areas of similar protocol time T. Intermediate timed protocols (area R2) is concentrated for $\phi < \pi$, while protocols with long duration (area R3) is mostly located in $\phi > \pi$. This asymmetry comes from the magnetic field B that lifts the degeneracy between the states $|\pm 1\rangle$. We also note that $\phi = 0$ and $\phi = 2\pi$ have different results in terms of fidelity and protocol, although they represent the same quantum state. The reason is that the neural network receives as input only the angles θ and ϕ of the target state and does not know about the symmetry of the problem (e.g. that ϕ is periodic). This symmetry can be enforced by hand, yielding similar fidelities (see supplementary materials). In figure 2(d), we show the distances between different protocols using equation (4). We chose a 21×21 sampling grid on the Bloch sphere. Although the protocols are actually parameterized by a $3N_\mathrm{T}$ dimensional vector, using the t-SNE algorithm we can map the distances between different protocols (given by equation (4)) onto a 2D representation. The color indicates the protocol time T. We see again distinct clusters of similar protocols, corresponding to the areas marked in figure 2(b).

Representative examples of driving protocols from the three different protocol 'phases' are shown in figures 3(a)–(c). The protocols belonging to different 'phases' look distinct, while protocols within the same 'phase' look very similar (see supplementary materials for further example protocols). Figure 3(c) shows the length of each time step, where the total time (given by the integral over the steps) increases from R1 to R3. We observe also distinct shapes in the sequence of driving strengths, which vary for R1 to R3 (see figures 3(a) and (b)). In figures 3(d)–(f) we show the change of the protocol parameters for a cut at $\phi = \pi$ along the θ axis. We again observe the three distinct 'phases' for varying θ in the driving parameters. They change contentiously with θ, however remain similar within one 'phase'. However, sudden jumps where the protocol changes drastically are observed at $\cos(\theta) \approx 0$ and $\cos(\theta) \approx 0.75$ (most visible for timestep $n_t = 4$ and $n_t = 8$). At the border between two protocol 'phases', the protocols change drastically in order to choose between them. This creates the sudden jumps observed. The protocols found are near-optimal solutions within a complicated optimization landscape [5]. As such, there are many possible solutions of nearly the same fidelity. By varying hyperparameters of the neural network (such as the number of neurons) or external parameters (such as the applied magnetic field) one may find different types of the protocols in the two-dimensional subspace. However, in most cases there are either two or three distinct areas (see other example in supplementary materials).

The hyperparameters of the neural network and the reinforcement learning affect the quality of the learning protocols. In figure 4, we show the average fidelity over all target states for varying number of protocol timesteps (figure 4(a)) and number of neural network neurons (figure 4(b)). We observe that beyond a certain number of neurons or timesteps, the average fidelity does not increase anymore.

To compare with a standard method, we optimize the state preparation with Nelder–Mead (using Python *scipy* library implementation). The result is shown in figure 5. The two-dimensional space of target states is subdivided into a discrete grid and optimized independently. The achieved average fidelity over all target

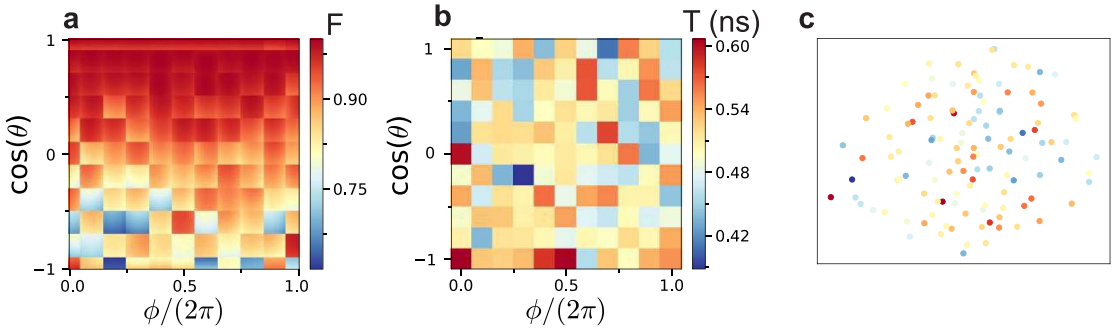

Figure 5. Create arbitrary quantum state in two-level subspace equation (2) by optimizing with a standard optimization tool (Nelder–Mead). (a) Fidelity F for preparing state $\Psi(\theta, \phi)$. Optimizing a 11×11 grid over the parameter space, intermediate points interpolated by using nearest protocol (mean fidelity $\langle F \rangle = 0.90$). (b) Protocol time T. No clustering of protocols is observed. (c) Two-dimensional representation of the distance between protocols (determined by equation (4)) using t-SNE algorithm. Color indicates protocol time of a specific datapoint. Same parameters as in figure 2. Optimization took in total 5.7×10^6 training episodes. Each grid point is optimized with Nelder–Mead for up to 20 000 optimization steps, repeated over up to 5 optimization runs starting from a random initially guessed protocol. Optimization is stopped early when fidelity $F > 0.99$ is reached.

states is lower compared to the reinforcement learning as it tends to get stuck in local optimization minimas. To reduce this problem, each trained target state was optimized up to five times, using random initial guesses. This optimization required also nearly an order of magnitude more training episodes as every grid point is learned independently and sequentially, instead of training all target states at the same time. The resulting optimized protocols across ϕ and θ is quite different compared to the one achieved by reinforcement learning. As an example, we investigate the protocol time T of the optimized protocols in figure 5(b). For reinforcement learning we find a connected landscape with well defined classes. For the grid optimization, however, there is a large spread in the protocol parameter and T varies largely between different target states, even for neighboring θ and ϕ. Thus, a small change in the target state can translate into a very different protocol. If one wants to find a protocol for a target state lying in between two grid points by interpolating between the two grid points, the interpolated protocol would achieve a low fidelity. A two-dimensional representation of the distance between the different protocols reveals no clustering or order (figure 5(c)), in contrast to reinforcement learning (figure 2(c)). From this result, we gather the following explanation: the control problem has many nearly equivalent solutions [5, 30], which yield similar fidelity but can have widely different control schemes. Reinforcement learning tends to find solutions belonging to the same control class, which produces the observed clusters, whereas Nelder–Mead converges to one of the solutions at random, giving uncorrelated protocol schemes. Further, this optimization method takes more training epochs than reinforcement learning since each grid point is optimized individually. Scaling to a higher density of discretization points or more than two target parameters would require even more training time for the grid approach. The reinforcement learning approach in [12] also suffers from similar scalability issues, whereas our deep learning algorithm can overcome such scalability problems as all target states can be trained in parallel.

4. Discussion

We demonstrated how to learn all control protocols for a two-dimensional set of quantum states. The neural network is trained using randomly sampled target states. After training, the neural network knows how to generate all possible target states and classifies the near-optimal protocols automatically in specific groups, e.g. the time needed to generate specific states as well as the driving strengths. The clustering feature could be useful tool to identify physical principles and find generalized driving protocols. For practical usage, the clustering of solutions is also advantageous as the driving protocols within a cluster do not need to change drastically for a small change in target state. The drop in fidelity at the boundary between two clusters is a result of the finite capacity of neural networks to represent sharp changes and could be mitigated by choosing a discontinuous activation function. However, the exact mechanism how the clusters are found by the neural network remains unclear and requires further studies. To speed up training, our algorithm could be easily parallelized by calculating multiple target states at the same time. This is more difficult for other optimization techniques such as transfer learning as it requires input from previously trained neural networks and does not scale well with number of sampling points as well as the dimension of the problem (i.e. number of target parameters) [12].

In this article, we have shown how to optically control the electron spin in a complex multi-level NV center. The electron spin is only indirectly coupled via several other interacting levels, making it difficult to construct good control pulses. Our approach via reinforcement learning gives us access to a tailored protocol for any superposition state. We are able to create arbitrary states within a short time $T \approx 0.5$ ns, avoiding slower dissipative processes and superseding other adiabatic-type protocols to create superposition states [19]. Our protocol requires only nine timesteps to achieve high fidelity for arbitrary states, which is considerably less than comparable approaches [13, 21]. Our results indicate that even less timesteps could be sufficient to reach high fidelity (see figure 4(a)). The control protocols we found could help to efficiently control NV centers for quantum processing [31] and could be applied for optimal control of other physical systems [27].

Our neural network-based algorithm learns the full two-dimensional set of target states. The same concept could help to identify control unitaries for continuous variable quantum computation [32] or serve as an alternative to transfer learning in quantum neural network states [33]. The neural network finds patterns, which may be useful to identify phase transitions in quantum control [5], as well as identify physical concepts in the way the protocols create quantum states [34, 35]. The resulting classification of protocols also opens up the potential of using reinforcement learning in identifying phase transitions in physical systems [36–38]. Furthermore, our approach could help correcting drifts in superconducting qubits [39]. When the system parameters change, the control unitaries have to be re-trained to accommodate the change. If one has multiple control unitaries, this may take a long time since every unitary has to be retrained individually. Our method allows one to retrain all the protocols at the same time, which can significantly shorten the time needed to correct errors and drifts. Finally, in our study we have trained the neural network by randomly sampling from all possible target states. It would be interesting to study training from a reduced subset of target states, and then see how well the neural network is able to generalize to yet unseen target states or whether over-fitting occurs.

5. Methods

5.1. NV center

In this paper we consider the 10-level model of the NV^- center, which comprises three ground states, six excited states and one metastable state. The ground states form a spin-1 triplet with a zero-field splitting of $D_{gs} \approx 2\pi \times 2.88$ GHz between the $m_s = \pm 1$ and $m_s = 0$ sublevels. The energy gap of $E_g = 1.94$ eV between the ground states and excited states due to Coulomb interaction gives rise to the well-known zero-phonon line (ZPL) optical transition. The level structure is shown in figure 6.

The ground state Hamiltonian can be written in the basis $\{|-1\rangle, |0\rangle, |+1\rangle\} \equiv \{|m_s = -1\rangle, |m_s = 0\rangle, |m_s = +1\rangle\}$ as (setting $\hbar = 1$ and the energy of $|0\rangle$ as zero) [21]:

$$H_{gs} = (D_{gs} - g_{gs}\mu_B B_{ext})\,|-1\rangle\langle -1| + (D_{gs} + g_{gs}\mu_B B_{ext})\,|+1\rangle\langle +1| \tag{5}$$

where $g_{gs} = 2.01$ is the Landé g-factor for the ground state, μ_B is the Bohr magneton, and B_{ext} is the external magnetic field applied along the NV quantization axis. The magnetic field splits the degeneracy of the states $|-1\rangle$ and $|+1\rangle$. Considering low temperatures, the phononic effects in the diamond are suppressed (and not considered here), while the splittings in the excited state due to spin–spin and spin–orbit interactions become significant. Taking into account these interactions, the excited-state Hamiltonian can be written in the basis of $\{|A_2\rangle, |A_1\rangle, |E_X\rangle, |E_Y\rangle, |E_1\rangle, |E_2\rangle\}$ as

$$H_{es} = E_g I + \begin{pmatrix} H_1 & 0 \\ 0 & H_2 \end{pmatrix} \tag{6}$$

where I is the 6×6 identity matrix and

$$H_1 = \begin{pmatrix} \Delta + 2l_z & g_{es}\mu_B B_{ext} \\ g_{es}\mu_B B_{ext} & -\Delta + 2l_z \end{pmatrix}, \tag{7}$$

$$H_2 = \begin{pmatrix} -D_{es} + l_z & 0 & 0 & \Delta'' \\ 0 & -D_{es} + l_z & i\Delta'' & 0 \\ 0 & -i\Delta'' & 0 & -g_{es}\mu_B B_{ext} \\ \Delta'' & 0 & -g_{es}\mu_B B_{ext} & 0 \end{pmatrix}, \tag{8}$$

Table 1. Decay channels and rates for the NV center. $|m\rangle$ represents the metastable state which comprises the singlet states $|^1A_1\rangle$ and $|^1E\rangle$.

Transition	Decay rate (ns^{-1})
$\|A_2\rangle, \|A_1\rangle, \|E_1\rangle, \|E_2\rangle \rightarrow \|+1\rangle$	1/24
$\|A_2\rangle, \|A_1\rangle, \|E_1\rangle, \|E_2\rangle \rightarrow \|-1\rangle$	1/31
$\|A_2\rangle, \|A_1\rangle, \|E_1\rangle, \|E_2\rangle \rightarrow \|0\rangle$	1/104
$\|A_2\rangle, \|A_1\rangle, \|E_1\rangle, \|E_2\rangle \rightarrow \|m\rangle$	1/33
$\|E_x\rangle, \|E_y\rangle \rightarrow \|0\rangle$	1/13
$\|E_x\rangle, \|E_y\rangle \rightarrow \|+1\rangle, \|-1\rangle$	1/666
$\|E_x\rangle, \|E_y\rangle \rightarrow \|m\rangle$	0
$\|m\rangle \rightarrow \|0\rangle$	1/303
$\|m\rangle \rightarrow \|+1\rangle, \|-1\rangle$	0

describe the level splittings in the excited-state manifold with $\Delta = 2\pi \times 1.55$ GHz, $D_{es} = 2\pi \times 1.42$ GHz and $\Delta'' \approx 2\pi \times 0.2$ GHz denotes the spin–spin interactions. $l_z = 2\pi \times 5.3$ GHz is the axial spin–orbit splitting, and $g_{es} \approx 2.01$ is the Landé g-factor for the excited state.

Coherent control of the NV^- center is accomplished by applying two laser fields with frequencies ω_1 and ω_2 and Rabi frequencies Ω_1 and Ω_2.

The electric-dipole coupling between the ground and excited states is given by the interaction Hamiltonian:

$$H_{\text{int}} = \begin{pmatrix} 0 & v \\ v^\dagger & 0 \end{pmatrix}, \tag{9}$$

where

$$v = \begin{pmatrix} i\epsilon_x & -i\epsilon_x & 0 & 0 & -i\epsilon_x & -i\epsilon_x \\ 0 & 0 & 0 & 2\epsilon_x & 0 & 0 \\ -i\epsilon_x & -i\epsilon_x & 0 & 0 & i\epsilon_x & -i\epsilon_x \end{pmatrix}, \tag{10}$$

is the 3×6 coupling matrix with the rows forming the ground state basis and the columns forming the excited state basis. $\epsilon_x = 2\Omega_1 \cos(\omega_1 t) + 2\Omega_2 \cos(\omega_2 t)$. The total Hamiltonian including driving is given as $H_{NV} = H_{gs} + H_{es} + H_{int}$. We now move into the rotating frame by transforming the Hamiltonian to the interaction picture $H_I = e^{iH_{Eg}t}\left(H_{tot} - H_{Eg}\right)e^{-iH_{Eg}t}$, with $H_{tot} = H_{gs} + H_{es} + H_{int}$ and $H_{Eg} = E_g \sum_{k=4}^{9} |k\rangle\langle k|$. Neglecting the counter-rotating terms, the ε_x terms in the interaction Hamiltonian are replaced by $\epsilon_x' = \Omega_1 \cos(\delta_1 t) + \Omega_2 \cos(\delta_2 t)$, with the detuning $\delta_i = \omega_i - E_g$, $i = 1, 2$. In addition, there is a metastable state $|m\rangle$ in the Hamiltonian, totaling to a 10 level system. The NV center is subject to dissipation via decay of excited states as shown in table 1. The full system including dissipation is solved using the Lindblad Master equation [40]

$$\frac{\partial \rho}{\partial t} = -\frac{i}{\hbar}[H, \rho] - \frac{1}{2}\sum_m \{\hat{L}_m^\dagger \hat{L}_m, \rho\} + \sum_m \hat{L}_m \rho \hat{L}_m^\dagger, \tag{11}$$

where $\hat{L}_m$ are the Lindblad operators describing the dissipation.

For timescales shorter than the fastest dissipation channel, the effect of dissipation can be neglected. For the NV center, this would be for $T \ll 13$ ns. In this limit, the dynamics is effectively governed by the coherent Hamiltonian only and reduces to a eight level system, as both state $|0\rangle$ and $|m\rangle$ are only accessible via dissipation.

5.2. Deep reinforcement learning

Here, we describe our machine learning algorithm in more detail. We learn the driving protocol via a deep Q-learning network [41], utilizing the actor-critic method with PPO [17]. We use [28] implementation of the algorithm in Tensorflow [42]. A sketch is shown in figure 7. The quantum system is controlled by an agent, that depending on the state s_t of the system acts with an action a_t (e.g. driving parameters for time t) using the probabilistic policy $\pi(a_t|s_t)$. At every timestep a reward (e.g. the fidelity of quantum state) is paid out. The goal is to repeatedly interact with the quantum system and learn the best policy that gives the highest final reward. One normally starts with a random policy, that explores many possible trajectories. Over the course of the training, the policy is refined and converges (hopefully) to the optimal (deterministic) policy. However, optimizing the policy directly can be difficult, as one round of the protocol is played out over N_T timesteps. The question is how to optimize the policy at each step such that one finds the optimal

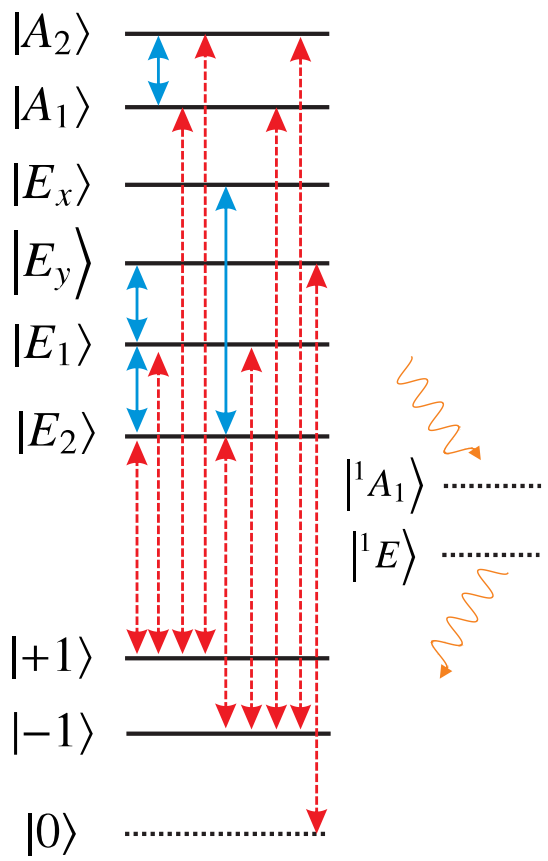

Figure 6. Level structure of the NV center. Consists of two target ground state levels $\{|-1\rangle, |+1\rangle\}$, third ground state level $|0\rangle$, excited states $\{|A_2\rangle, |A_1\rangle, |E_X\rangle, |E_Y\rangle, |E_1\rangle, |E_2\rangle\}$ and metastable states $\{|^1A_1\rangle, |^1E\rangle\}$. Laser coupling of levels is shown as red dashed lines, coupling via internal level structure as blue solid lines. For short driving protocols $\{|^1A_1\rangle, |^1E\rangle\}$ and $|0\rangle$ are disregarded as approximation.

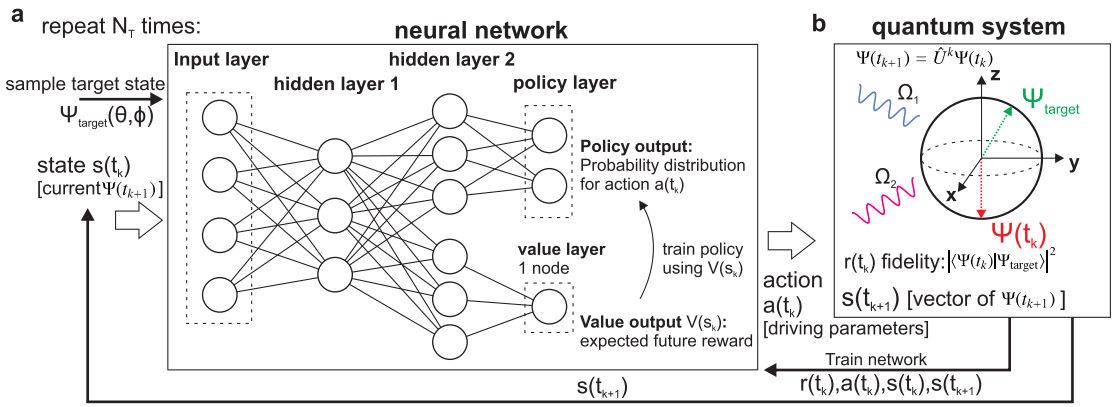

Figure 7. Deep reinforcement learning with proximal policy optimization (PPO) to control arbitrary target quantum states $\Psi_{target}(\theta,\phi)$.

final reward. A common problem in optimization is that one does not find the global optimal solution, but only a local maximum of the optimization landscape.

Here, it has been shown one can overcome the difficulties of policy learning with Q-learning. The idea is to find the Q-function $Q_\pi(s_t, a_t)$ that estimates the future reward (from the point of timestep t) that is paid out at the end the full protocol with this policy. The goal is to learn a policy that prioritizes long-term rewards over smaller short-term gains. The optimal Q-function is determined by the Bellman equation:

$$\begin{aligned} Q(s_t, a_t, \pi) &= \mathbb{E}\left[r_t + \gamma Q(s_{t+1}, a_{t+1}, \pi)\right] \\ &= \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots\right], \end{aligned}$$

where $\mathbb{E}[.]$ indicates sampling over many instances. $\gamma \leq 1$ is a discount factor that weighs future rewards against immediate rewards.

PPO is based on the idea of combining both methods, policy learning and value learning, into the actor-critic method. The idea is to have two neural networks: a policy network and a value network. The policy network (actor) decides on the next action by determining the parameters of the policy. The value-based network (critic) evaluates the taken action on how well it solves the task and estimates the future expected reward. It is used as an input to train the policy network.

Better performance can be achieved if the Q-function is split into two parts [43]: $Q(s_t, a_t) = A(s_t, a_t) + V(s_t)$, where $A(s_t, a_t)$ is the advantage function and $V(s_t)$ the value function. $V(s_t)$ gives the expected future reward averaged over the possible actions according to the policy. This is the output of the critic network. $A(s_t, a_t)$ gives the improvement in reward for action a_t compared to the mean of all choices.

Learning is achieved by optimizing the network parameters with a loss function via gradient descent [44]. The loss function of the value network is the square of the difference of the value function of the network and the predicted reward in the next timestep $L_V(\theta) = \mathbb{E}_t\left[(V_\theta(s_t) - y_t)^2\right]$, where θ are the current network parameters, $y_t = r_t + V_{t+1}$, where V_{t+1} is the output of the value network for the next timestep (it is set to zero if this is the last timestep).

The advantage function $A(s_t, a_t)$ tells us how good a certain action a_t is compared to other possible actions. The advantage function is the input to train the policy network (the actor). Following the idea of PPO [17], the goal is to minimize the loss function of the policy network:

$$L_{\mathrm{p}}(\theta) = -\mathbb{E}_t\left[\frac{\pi_\theta(s_t, a_t)}{\pi_{\theta_{\mathrm{old}}}(s_t, a_t)} A(s_t, a_t)\right], \tag{12}$$

where θ are the network parameters and θ_{old} are the network parameters of a previous instance. Maximizing $L_{\mathrm{p}}(\theta)$ for the network parameters θ over many sampled instances guides the distribution $\pi_\theta(s_t, a_t)$ such that it returns actions a_t with maximal advantage. However, the ratio

$$b_t(\theta) = \frac{\pi_\theta(s_t, a_t)}{\pi_{\theta_{\mathrm{old}}}(s_t, a_t)},$$

can acquire excessively large values, causing too large changes in the policy in every training step and making convergence difficult. For PPO, it was proposed to use a clipped ratio ε [17]:

$$L_{\mathrm{p}}(\theta) = -\mathbb{E}_t\left[\min\left\{b_t(\theta)A(s_t, a_t), \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A(s_t, a_t)\right\}\right],$$

such that the update at each step stays in reasonable bounds.

The input state s_t to the neural network are the wavefunction $\Psi(t_n)$ (or density matrix $\rho(t_n)$ at previous timestep n, as well as a random target state Ψ_{target}. The output of the policy network are the parameters for the policy $\pi(a_t|s_t, \mu, \sigma)$, where the actions (pulse amplitude and time step duration) are sampled from a normal distribution with mean value μ and width σ. μ is chosen by the neural network (given the input state) and σ is a global variable that is initially large (ensuring that driving parameters are initially sampled mostly randomly to explore many possible trajectories). It is optimized via the loss function and decreases over the training, until at convergence it is close to zero and the sampled driving parameters converge to μ. We constrain the possible output values for the driving parameters by punishing values outside the desired range with a negative reward.

We optimize the neural network over many epochs N_{E}. In each epoch, the Schrödinger equation (or master equation) is propagated for a total time T with N_T discrete timesteps of width Δt, with respective times t_n. For one epoch, the system runs the network N_T times. From the policy network the driving parameters for the $n+1$ timestep are sample. The output of the value network is used to train the policy network. Each network is composed of two hidden layers of fully connected neurons of size N_{H} with ReLu activation functions. The neural networks are trained with the loss function after calculating the full time evolution to time T over N_T timesteps. Training data is sampled from a buffer storing earlier encountered trajectories. For the actual implementation, we choose the following parameters: learning rate for both value and policy network $\alpha = 10^{-5}$, training over $N_{\mathrm{E}} = 800\,000$ epochs and clip ratio $\varepsilon = 0.05$. Out of bounds driving parameters are punished by a factor of 0.2.

To improve the mean fidelity over all target states, we choose the target states not completely random, but biased towards areas of lower fidelity. This is achieved by laying a 20×20 grid across the θ and ϕ space, then binning the last 10 000 results and the achieved fidelity. We choose the next target state by sampling the probability distribution:

$$P(\theta, \phi) = \frac{1}{2\mathcal{N}}(\eta + (1-\eta)(1 - \langle F(\theta, \phi)\rangle)),$$

where $\mathcal{N}$ is a normalization factor and η determines how strongly the sampling is biased toward low fidelity. We chose $\eta = 0.5$.

Acknowledgments

The computational work for this article was partially performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg).

Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.

Author contributions

T H and W-K M conceived the project, performed the calculations and wrote the paper. All authors discussed the results and implications at all stages.

Competing interest

The authors declare that there are no competing interests.

Code availability

Computer code is accessible online on Github in [45].

ORCID iDs

Tobias Haug https://orcid.org/0000-0003-2707-9962
Wai-Keong Mok https://orcid.org/0000-0002-1920-5407

References

[1] Schulman J, Moritz P, Levine S, Jordan M and Abbeel P 2015 arXiv:1506.02438
[2] Mnih V, Badia A P, Mirza M, Graves A, Lillicrap T, Harley T, Silver D and Kavukcuoglu K 2016 *Int. Conf. on Machine Learning* pp 1928–37 arXiv:1602.01783
[3] Silver D *et al* 2016 *Nature* **529** 484
[4] Chen C, Dong D, Li H-X, Chu J and Tarn T-J 2013 *IEEE Trans. Neural Netw. Learn. Syst.* **25** 920
[5] Bukov M, Day A G, Sels D, Weinberg P, Polkovnikov A and Mehta P 2018 *Phys. Rev.* X **8** 031086
[6] Zhang X-M, Wei Z, Asad R, Yang X-C and Wang X 2019 *NPJ Quant. Inf.* **5** 1
[7] Bharti K, Haug T, Vedral V and Kwek L-C 2019 arXiv:1912.10783
[8] Haug T, Dumke R, Kwek L-C, Miniatura C and Amico L 2019 arXiv:1911.09578
[9] Dalgaard M, Felix M, Sørensen J J and Jacob S 2020 *NPJ Quant. Inf.* **6**
[10] An Z and Zhou D 2019 *EPL* **126** 60002
[11] Carleo G, Cirac I, Cranmer K, Daudet L, Schuld M, Tishby N, Vogt-Maranto L and Zdeborová L 2019 *Rev. Mod. Phys.* **91** 045002
[12] Niu M Y, Boixo S, Smelyanskiy V N and Neven H 2019 *NPJ Quant. Inf.* **5** 33
[13] Porotti R, Tamascelli D, Restelli M and Prati E 2019 *Commun. Phys.* **2** 1
[14] Xu H, Li J, Liu L, Wang Y, Yuan H and Wang X 2019 *NPJ Quant. Inf.* **5** 1
[15] Bharti K, Haug T, Vedral V and Kwek L-C 2020 arXiv:2003.11224
[16] Arrazola J M, Bromley T R, Izaac J, Myers C R, Brádler K and Killoran N 2019 *Quant. Sci. Technol.* **4** 024004
[17] Schulman J, Wolski F, Dhariwal P, Radford A and Klimov O 2017 arXiv:1707.06347
[18] Yale C G, Buckley B B, Christle D J, Burkard G, Heremans F J, Bassett L C and Awschalom D D 2013 *Proc. Natl. Acad. Sci. USA* **110** 7595
[19] Zhou B B *et al* 2017 *Nat. Phys.* **13** 330
[20] Yale C G, Heremans F J, Zhou B B, Auer A, Burkard G and Awschalom D D 2016 *Nat. Photon.* **10** 184
[21] Tian J, Du T, Liu Y, Liu H, Jin F, Said R S and Cai J 2019 *Phys. Rev.* A **100** 012110
[22] Gruber A, Dräbenstedt A, Tietz C, Fleury L, Wrachtrup J and Von Borczyskowski C 1997 *Science* **276** 2012
[23] Balasubramanian G *et al* 2009 *Nat. Mater.* **8** 383
[24] Maurer P C *et al* 2012 *Science* **336** 1283
[25] Chu Y, Lukin M D 2015 *Quantum Optics and Nanophotonics* eds, C Fabre, V Sandoghdar, N Treps and L F Cugliandolo (Oxford: Oxford University Press) p 229 arXiv:1504.05990
[26] Wang Z-Y, Cai J-M, Retzker A and Plenio M B 2014 *New J. Phys.* **16** 083033
[27] Werschnik J and Gross E 2007 *J. Phys.* B **40** R175
[28] Achiam J Openai spinning up (available at: https://spinningup.openai.com/)
[29] Schulman J, Levine S, Abbeel P, Jordan M and Moritz P 2015 *Int. Conf. on Machine Learning* pp 1889–97 arXiv:1502.05477
[30] Rabitz H A, Hsieh M M and Rosenthal C M 2004 *Science* **303** 1998
[31] Chen Y, Stearn S, Vella S, Horsley A and Doherty M W 2020 arXiv:2002.00545

[32] Hillmann T, Quijandría F, Johansson G, Ferraro A, Gasparinetti S and Ferrini G 2020 arXiv:2002.01402
[33] Zen R, My L, Tan R, Hebert F, Gattobigio M, Miniatura C, Poletti D and Bressan S 2020 (*Phys. Rev. E* **101**) 053301
[34] Nautrup H P, Metger T, Iten R, Jerbi S, Trenkwalder L M, Wilming H, Briegel H J and Renner R 2020 arXiv:2001.00593
[35] Iten R, Metger T, Wilming H, Del Rio L and Renner R 2020 *Phys. Rev. Lett.* **124** 010508
[36] Wang L 2016 *Phys. Rev.* B **94** 195105
[37] Rem B S, Käming N, Tarnowski M, Asteria L, Fläschner N, Becker C, Sengstock K and Weitenberg C 2019 *Nat. Phys.* **15** 917
[38] Ming Y, Lin C-T, Bartlett S D and Zhang W-W 2019 *NPJ Comput. Mater.* **5** 88
[39] Foxen B *et al* 2020 (available at: arXiv:2001.08343)
[40] Breuer H-P and Petruccione F 2002 *The Theory of Open Quantum Systems* (Oxford: Oxford University Press)
[41] Mnih V *et al* 2015 *Nature* **518** 529
[42] Abadi M *et al* 2015 TensorFlow: large-scale machine learning on heterogeneous systems software available from tensorflow.org (available at: http://tensorflow.org/)
[43] Wang Z, Schaul T, Hessel M, Van Hasselt H, Lanctot M and De Freitas N 2015 arXiv:1511.06581
[44] Kingma D P and Ba J 2014 arXiv:1412.6980
[45] Haug T and Mok W-K Deep q-control (available at: https://github.com/txhaug/deepQControl)