LETTER

Deep reinforcement learning for quantum gate control

To cite this article: Zheng An and D. L. Zhou 2019 *EPL* **126** 60002

View the article online for updates and enhancements.

Deep reinforcement learning for quantum gate control

Zheng An[1,2] and D. L. Zhou[1,2,3,4(a)]

[1] *Institute of Physics, Beijing National Laboratory for Condensed Matter Physics, Chinese Academy of Sciences Beijing 100190, China*
[2] *School of Physical Sciences, University of Chinese Academy of Sciences - Beijing 100049, China*
[3] *Collaborative Innovation Center of Quantum Matter - Beijing 100190, China*
[4] *Songshan Lake Materials Laboratory, Dongguan - Guangdong 523808, China*

Abstract – How to implement multi-qubit gates efficiently with high precision is essential for realizing universal fault-tolerant computing. For a physical system with some external controllable parameters, it is a great challenge to control the time dependence of these parameters to achieve a target multi-qubit gate efficiently and precisely. Here we construct a dueling double deep Q-learning neural network (DDDQN) to find out the optimized time dependence of controllable parameters to implement two typical quantum gates: a single-qubit Hadamard gate and a two-qubit CNOT gate. Compared with traditional optimal control methods, this deep reinforcement learning method can realize efficient and precise gate control without requiring any gradient information during the learning process. This work attempts to pave the way to investigate more quantum control problems with deep reinforcement learning techniques.

Introduction. – High-fidelity quantum gate plays an essential role in achieving quantum supremacy [1] and fault-tolerant quantum computing [2]. In the present days, the study of quantum control has developed a series of methods in practice, such as nuclear magnetic resonance experiments [3], trapped ions [4,5], superconducting qubits [6], and nitrogen vacancy centers [7]. Further, based on gradient or evolutionary algorithms, the development of control algorithms provides robust control strategies and have been intensively used. However, it is hard to get such high-quality gates under limited control resources with a precise choice of the control signal, like time discretization of the fields or fixed amplitude. In a previous work [8], under certain limitations, the quantum control landscape was non-convex but will get dumped in the vicinity of the quantum speed limit time. Even though the result of the topology of quantum control landscapes has been intensively tested and studied [9–11], it is hard to minimize errors of some quantum systems. In addition, these problems can be generalized to hard quantum control problems [12]. All these limitations are hard to be solved with common quantum-control techniques but meaningful for being discussed in the physical world.

On the other hand, machine learning (ML), already explored as a tool in many aspects of physics [13,14], provides a complete paradigm to achieve an analysis of various quantum systems [14–17]. With tremendous aspects studied in ML, reinforcement learning (RL) has been a focus on the study of artificial intelligence agents to interact with the real world. Equipped with deep neural network, the deep RL techniques has revolutionized traditional optimal control which provides efficient, precise, and robust performance. Further empowered by advanced optimization techniques, the artificial intelligence agent is able to solve high-dimensional optimization problems such as video games and go [18–20]. Recently, researchers have begun to utilize some RL algorithms in the quantum control studies [21,22]. The novel RL algorithm provides advanced optimization techniques which are able to solve more difficult optimization problems.

In this article, we investigate the traditional quantum gate control problem where an efficient strategy for preparing a high-fidelity quantum gate is proposed by an artificial intelligence agent. With deep RL, we propose a

(a)E-mail: zhoudl72@iphy.ac.cn

framework to connect optimal decision making of the underlying quantum dynamics with state-of-the-art RL techniques. In particular, within the present framework, the agent performs optimal discrete, sequential controls to get two typical quantum gates: a single-qubit Hadamard gate and a two-qubit CNOT gate. The results provide a general way to investigating the quantum control problem with deep RL techniques.

The rest of this paper is structured as follows. First, we briefly overview our quantum gate control model. Next, we present some relative RL algorithms and the DDDQN method for two quantum gate control models. Finally, we show the numerical results and draw our conclusions.

Bang-bang control model to implement quantum gates. – In this section, we give a bang-bang control model to implement quantum gates, which explains the physical problems we solve in this paper.

We consider a quantum system whose Hamiltonian is

$$H(\vec{\epsilon}(t)) = H_d + H_c(\vec{\epsilon}(t)), \qquad (1)$$

where the term H_d, called the drifted Hamiltonian, is the free evolution part of the Hamiltonian $H(\vec{\epsilon}(t))$. Another part of the Hamiltonian, $H_c(\vec{\epsilon}(t))$, called the control Hamiltonian, is under control by some time-dependent external parameter vector $\vec{\epsilon}(t)$.

In our bang-bang control protocol, our total control time T is fixed, which is divided into N short time periods with the same duration $\delta t = T/N$. In the i-th time period with $(i-1)\delta t \leq t \leq i\delta t$ ($1 \leq i \leq N$), the control parameter vector is constant, *i.e.*, $\vec{\epsilon}(t) = \vec{\epsilon}_i$, where the control parameter vector $\vec{\epsilon}_i$ is selected from a set $\mathcal{A}(\vec{\epsilon})$ of d possible choices. The unitary evolution operator in the i-th time period is

$$U(i\delta t, (i-1)\delta t; \vec{\epsilon}_i) = e^{-iH(\vec{\epsilon}_i)\delta t}. \qquad (2)$$

When all the N control parameter vectors $\{\vec{\epsilon}_1, \vec{\epsilon}_2, \ldots, \vec{\epsilon}_N\}$ are selected, the unitary operator at time T is determined by the iterative equations

$$U(i\delta t) = U(i\delta t, (i-1)\delta t; \vec{\epsilon}_i)U((i-1)\delta t), \qquad (3)$$
$$U(0) = I, \qquad (4)$$

where I is the identity operator in the Hilbert space of our system.

Our aim is to select the parameter vectors $\{\vec{\epsilon}_1, \vec{\epsilon}_2, \ldots, \vec{\epsilon}_N\}$ to make the unitary operator $U(T)$ approximate the target unitary gate U_f as well as possible, which is formulated by maximizing the fidelity

$$\mathcal{F}(T) = \max_{\vec{\epsilon}_1, \vec{\epsilon}_2, \ldots, \vec{\epsilon}_N} \mathcal{F}(T; \vec{\epsilon}_1, \vec{\epsilon}_2, \ldots, \vec{\epsilon}_N) \qquad (5)$$

with the fidelity

$$\mathcal{F}(T; \vec{\epsilon}_1, \vec{\epsilon}_2, \ldots, \vec{\epsilon}_N) = \left| \frac{\text{Tr}\{U_f^\dagger U(T)\}}{D} \right|^2, \qquad (6)$$

where D is the dimension of the Hilbert space. We observe that $\mathcal{F}(T; \vec{\epsilon}_1, \vec{\epsilon}_2, \ldots, \vec{\epsilon}_N) \in [0, 1]$, and that $\mathcal{F}(T; \vec{\epsilon}_1, \vec{\epsilon}_2, \ldots, \vec{\epsilon}_N) = 1$ if and only if $U(T)$ is equal to U_f up to a phase factor.

In particular, the size of the set of the parameter vectors is d^N, which implies that it is impossible to exhaustively search for the optimal parameter vector sequence for a large N.

Here we focus on two typical target quantum gates, one is the Hadmard gate, the other is the CNOT gate.

Hadamard gate. When the target quantum gate is the single-qubit Hadmard gate

$$U_f = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \qquad (7)$$

we consider a two-level system whose Hamiltonian is

$$H(\epsilon(t)) = \sigma_z + \epsilon(t)\sigma_x, \qquad (8)$$

where σ_z and σ_x are Pauli matrices, and $\epsilon(t)$ is a real control parameter. This simple model has been widely applied in quantum physics, *e.g.*, it describes the non-adiabatic transition [23], the Landau-Zener-Stuckelberg interferometry [24] and the Kibble-Zurek mechanism [25].

Based on the Pontryagin maximum principle, we take the set of $d = 2$ possible control parameter $\mathcal{A}(\epsilon) \in \{\pm 4\}$ in our bang-bang protocol.

CNOT gate. When the target quantum gate is the CNOT gate

$$U_f = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \qquad (9)$$

we consider the Hamiltonian

$$\begin{aligned} H(\epsilon(t)) = {}& \sigma_z^{(1)} \otimes \sigma_z^{(2)} + \epsilon_1(t)\sigma_x^{(1)} \otimes \mathbb{I}^{(2)} + \epsilon_2(t)\mathbb{I}^{(1)} \otimes \sigma_x^{(2)} \\ & + \epsilon_3(t)\sigma_y^{(1)} \otimes \mathbb{I}^{(2)} + \epsilon_4(t)\mathbb{I}^{(1)} \otimes \sigma_y^{(2)}, \end{aligned} \qquad (10)$$

where $\mathbb{I}$ is the 2×2 identity matrix, and $\vec{\epsilon}(t) = (\epsilon_1(t), \ldots, \epsilon_4(t))$ is a 4-component parameter vector.

Similarly as in the case of the Hadamard gate, we take the set of $d = 16$ possible choices of the parameter vector as

$$\mathcal{A}(\vec{\epsilon}) = \{(\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4) \text{ with } \epsilon_i \in \{\pm 4\}\}. \qquad (11)$$

Deep reinforcement learning methods. – In this section, we show how to apply the deep RL to approximately solve the maximization problem specified by eq. (5) in our bang-bang control quantum gate implementation protocol. To this end, we firstly review the necessary concepts in deep RL methods, especially the framework of the dueling double deep Q-learning neural network, which is adopted in our problem. Then we show how to combine our bang-bang control protocol with the deep RL methods.

Reinforcement learning. RL is a kind of ML method in which an intelligent agent aims to find a series of actions on a given environment to optimize its performance by delayed scalar rewards received [26].

The problem of RL is described as a finite Markov decision process [26]. At time $t = 0$, the state of the environment is S_0, and the agent chooses an action A_0. At time $t = 1$, the state of the environment becomes S_1 after the action A_0, and the environment also gives a scalar reward R_1. Then the agent chooses an action A_1, and repeats the above procedure. In general, this Markov process is described as a state-action-reward sequence

$$S_0, A_0, R_1, S_1, A_1, R_2, \ldots .$$

For a finite Markov decision process, the sets of the states, the actions and the rewards are finite. The total discounted return at time t reads

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{12}$$

where γ is the discount rate and $0 \leq \gamma \leq 1$.

In RL, the agent selects the actions according to a policy π, which is specified by a conditional probability of selecting an action A for each state S, denoted as $\pi(A|S)$. The task of the agent is to learn an optimal policy π_*, which maximizes the expected discounted return

$$V_\pi(s) = E_\pi(G_t|S_t = s), \tag{13}$$

where E_π denotes the average expectation under the policy π.

It has been shown that the optimal policy π^* exists and can be found iteratively as follows. Let us introduce the value of the state-action function, the conditional discount return reads

$$Q_\pi(s, a) = E_\pi(G_t|S_t = s, A_t = a). \tag{14}$$

If we have a policy π, then we calculate the value of the state-action function $Q_\pi(S, A)$. For each state s, we take an action maximizing the value of the state action $Q_\pi(s, A)$, which forms our new policy π'. Then we calculate the value of the state-action function $Q_{\pi'}(S, A)$ by repeating the above procedure until the new policy equals the updated one, which is the optimal policy π^* we are looking for.

Another well-known method to get the optimal policy π^* is the Q-learning [27], an off-policy temporal-difference control algorithm defined as

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \Delta Q \tag{15}$$

with

$$\Delta Q = \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)], \tag{16}$$

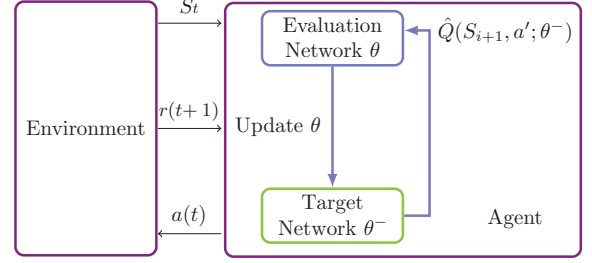where α is the step size parameter.

Fig. 1: An overview of the deep RL: at each time step of training, the evaluation network of the agent proposes a control action of $a(t)$, the environment takes the proposed action and evaluates the gate of eq. (2) for the time duration δt to obtain a new unitary gate U_{t+1} and calculates the reward of eq. (21), both of which are fed into the RL agent. The evaluation network of the agent is updated with the loss function of eq. (17) by back-propagation. With fixed numbers of steps, the agent updates the parameters of target network by transferring the parameters of the evaluation network.

Dueling double deep Q-learning. In this section, we introduce the dueling double deep Q-learning neural network (DDDQN), which will be used in our quantum gate control problem. The advantage of this method has been discussed in previous research [28].

First, we begin by introducing the double Q-learning method [29] in the training of our agent. As shown in fig. 1, the agent consists of the evaluation network and the target network with the same architecture. The evaluation network evaluates the state-action value $Q(S, A; \theta)$, and the target network evaluates the TD target $Q(S, A; \theta^-)$. At each learning step, we fed the agent with a minibatch of experiences $\{S_t, A_t, R_{t+1}, S_{t+1}\}$ with the prioritized experience replay (PER) method [30]. The state S_t is fed into the evaluation network to calculate the state-action value $Q(S_t, A_t; \theta)$. At the same time, the target network is to calculate $\max_{a'} Q(S_{t+1}, a'; \theta^-)$ in eq. (17). At the end of each step of training, the evaluation network is updated through the back-propagation by minimizing the loss. Based on eq. (15), the loss is the mean square error (MSE) of the difference between the evaluation $Q(S_t, A_t; \theta)$ and the target $\max_{a'} Q(S_{t+1}, a'; \theta^-)$,

$$\text{loss} = \text{MSE}((R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a'; \theta^-)) - Q(S_t, A_t; \theta)). \tag{17}$$

During the learning episodes (see fig. 1), the agent updates the parameters of the target network $\theta^- \rightarrow \theta$ to make better decisions.

Further, the detailed architecture of each network in our agent is shown in fig. 2. Each network consists of three parts: an encoder, an advantage network and a value network. The encoder extracts information about the states S_t for the next two neural networks. Based on the Q-learning, the state-action value $Q(S_t, A_t)$ represents the expected return for the agent to select the action A_t on the state S_t of the environment. In the architecture of
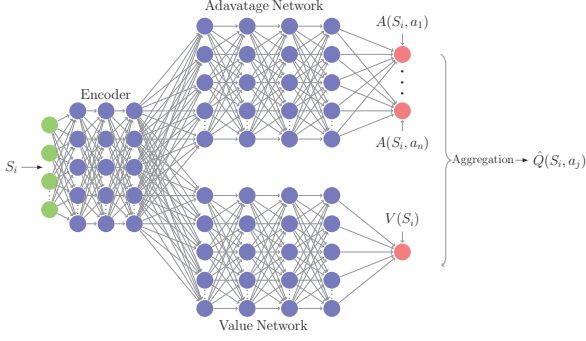
Fig. 2: The deep neural network architecture of our agent: For each state fed into the neural network, the encoder extracts the information of the state for further calculation. The value network and advantage network get the information of the encoder to calculate the value of the state for each action. Based on eq. (18), the neural network aggregates the value of the state and the advantage of the action by the state to get the state-action values $\hat{Q}(S_i, a_j)$.

the dueling network [28] in deep RL, we decompose the state-action value as

$$Q(S_t, A_t) = A(S_t, A_t) + V(S_t), \qquad (18)$$

where $V(S_t)$ is the state value for each state, and $A(S_t, A_t)$ is the advantage for each action. The state value $V(S_t)$ is calculated by the advantage network, and the advantage of action $A(S_t, A_t)$ is calculated by the value network. Then we combine these two values to get an estimate of $Q(S_t, A_t)$ through an aggregation layer.

Quantum gate control with DDDQN. To apply the reinforcement ML to our bang-bang control protocol, we need to build a map between their concepts. The state of the environment at time t is

$$S_t = U(t\delta t) = \{\Re(U_{ij}(t\delta t)), \Im(U_{ij}(t\delta t))\}, \qquad (19)$$

where $U_{ij}(t\delta t)$ is the matrix element of $U(t\delta t)$, and $\Re, \Im$ mean taking the real part and the imaginary part. The action of the agent at time t reads

$$a(\vec{\epsilon}) = U(t\delta t, (t-1)\delta t; \vec{\epsilon}). \qquad (20)$$

Note that the action does not depend on time t. The reward of the agent received in each step is

$$R_t = \begin{cases} 0, & t \in \{0, 1, \ldots, N-1\} \\ -\mathcal{L}(\mathcal{F}(T; \vec{\epsilon}_1, \vec{\epsilon}_1, \ldots, \vec{\epsilon}_N)), & t = N, \end{cases} \qquad (21)$$

where $\mathcal{L}(\mathcal{F})$ is the logarithmic infidelity, $\mathcal{L}(\mathcal{F}) = \log_{10}(1 - \mathcal{F})$. In other words, the agent will not get a reward immediately, but at time N.

Our algorithm for quantum gate control with DDDQN is given in Algorithm 1.

Algorithm 1: Deep RL for quantum gate control

Initialize memory R to empty;
Randomly initialize the evaluation network with random weights θ;
Randomly initialize the target network with random weights θ^-;
for *episode= 0, M* **do**
  Initialize s_0, $s_0 = f(U_0)$;
  for $t = 0, \ldots, t_N$ **do**
    With probability ϵ select a random action a_t, otherwise $a_t = argmax_a Q(s_t, a; \theta)$;
    Execute action a_t and observe the reward r_{t+1}, and the next state s_{t+1};
    Store experience $e_t = (s_t; a_t; r_{t+1}; s_{t+1})$ in R;
    if $t = t_N$ **then**
      Sample minibatch of experiences e_i with PER method;
      Set $y_i =$
      $\begin{cases} r_{i+1}, & \text{if } t_{i+1} = t_N, \\ r_{i+1} + \gamma \arg\max_{a'} Q(s_{t+1}, a'; \theta^-), & \text{otherwise} \end{cases}$
      Update θ by minimizing
      loss $= (y_i - Q(s_t, a_i; \theta))^2$;
    end
    Every C times of learning, set $\theta^- = \theta$;
  end
end

Numerical results. – In this section, we give the numerical results of the logarithmic infidelity $\mathcal{L}$ with target gates being the single-qubit Hadamard gate and the two-qubit CNOT gate from the deep reinforcement learning. To show the effectiveness of our deep RL method, we also calculate the logarithmic infidelity with three different algorithms: gradient ascent pulse engineering (GRAPE), differential evolution (DE), and genetic algorithm (GA). We then present our analysis of the performance of our deep RL algorithm against the other three algorithms.

Figure 3 shows the minimal logarithmic infidelities of preparing a single-qubit Hadmard gate in different evolution time T with different algorithms. For $T < 0.8$, the results on the infidelities from the four algorithms agree well. At $T = 0.9$, the results on the logarithmic from RL and DE agree well, which is better than that from GA, and worse than that from GRAPE. At $T = 1.0$, the infidelities obtained from RL and GRAPE abruptly decrease, which possibly implies that the speed limit time of the problem is in the region $[0.9, 1.0]$. In particular, these two algorithms find protocols to achieve infidelity $\mathcal{L} < -3$ (red line) or fidelity $\mathcal{F} > 99.9\%$ at $T = 1.0$. While GRAPE has the best performance out of the four methods, the algorithm requires the fidelity gradients at all time, and it is not readily accessible through experimental measurements. Further, GRAPE allows for the control field $\epsilon(t)$ to take any value in the interval $[-4, 4]$.
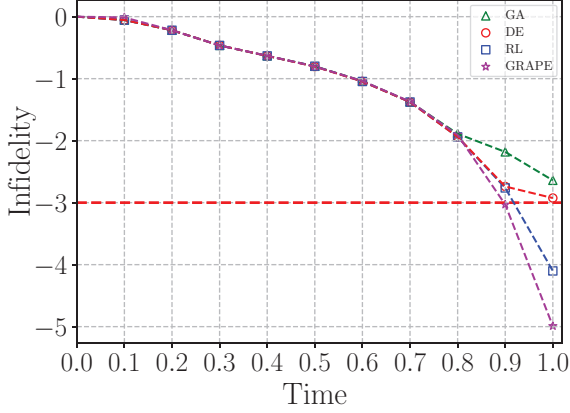
Fig. 3: Best infidelities of preparing a single-qubit Hadamard gate at different evolution time T. The markers correspond to the algorithms RL (blue □), GRAPE (purple ☆), DE (red ○) and GA (green △). The time step $N = 28$ for different T. Here we set 400 iterations for GRAPE DE and GA, 50000 training episodes for RL.
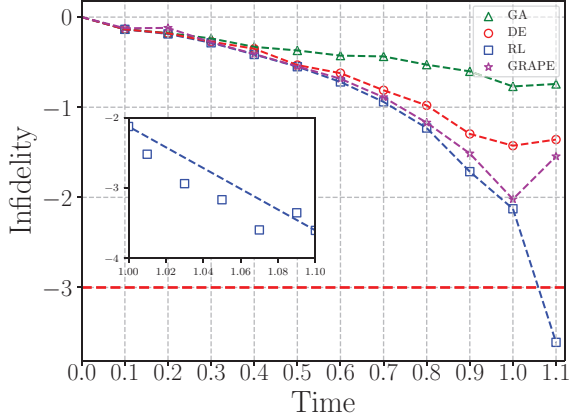

Fig. 4: Best infidelities of preparing a CNOT gate at different evolution time T. The markers correspond to the algorithms RL (blue □), GRAPE (purple ☆), DE (red ○) and GA (green △). The time step $N = 38$ for different T. Here we set 5000 iterations for GRAPE DE and GA, 150000 training episodes for RL.

In fig. 4, we compare the results of the CNOT gate control task from the four algorithms. Similarly as in the previous task, all the algorithms perform well for $T < 0.4$. RL, DE and GRAPE find optimal protocols in the time region $0.4 < T < 0.9$, but the performance of GA is poor for $T > 0.4$. After $T = 0.9$, only RL and GRAPE find optimal protocols, and the results of our RL agent are better than those of the GRAPE. Notice that at $T = 1.1$, the landscape seems to get dumped for the problem and all the algorithms except for RL get trapped. Like the state transfer problem [21,31], we believe this region may have a similar phase transition phenomenon and traditional algorithms are hard to maintain good performance. However, our RL agent ignores the dumped landscape and finds good protocols compared with other algorithms. To

Table 1: Training hyper-parameters.

Hyper-parameter	Values
Neurons in decoder network	$\{600, 600, 600\}$
Neurons in advantage(value) network	$\{600, 600, 600, 600\}$
Minibatch size	[a]
Replay memory size	100000
Learning rate	0.001[b]
Update period	100
Reward decay γ	0.95
Total episode	[c]

[a] 72 for Hadamard gate problem, 128 for CNOT gate problem.
[b] With Adam algorithm.
[c] 50000 for Hadamard gate problem, 150000 for CNOT gate problem.

investigate the performance of RL in this region, we plot detailed results in the inset of fig. 4. The results show that the agent has good and robust performance in the region.

Conclusion. – In this article, we apply the deep RL to explore the fast and high-precision quantum gate control problem. The quantum gate control problem is then mapped into a deep RL algorithm. Further, we build an RL agent to solve the quantum optimal control problem. We compare the numerical results among the four different algorithms on two typical quantum gate control problems. Our results demonstrate that the intelligent agent is able to effectively learn the optimal control schemes in approximating the target quantum gates. The success of our agent lies in its suitability for solving discrete action problems and its state-of-the-art RL technique of balancing exploration and exploitation.

The numerical results show that the performance of deep RL is robust and efficient in implementing arbitrary single and two qubit gates. However, there are still some challenges to extend RL algorithms to multi-qubit control problem. The main challenge needs to solve is that the control space will grow exponentially with the increase of the qubit number. We hope that our approach can inspire more applications of deep RL methods in the quantum control domain.

$* * *$

APPENDIX

Our RL agent makes use of a deep neural network to approximate the Q values for the possible actions of each state. The network (see fig. 2) consists of 4 layers of each sub-network. All layers have ReLU activation functions except for the output layer which has linear activation.
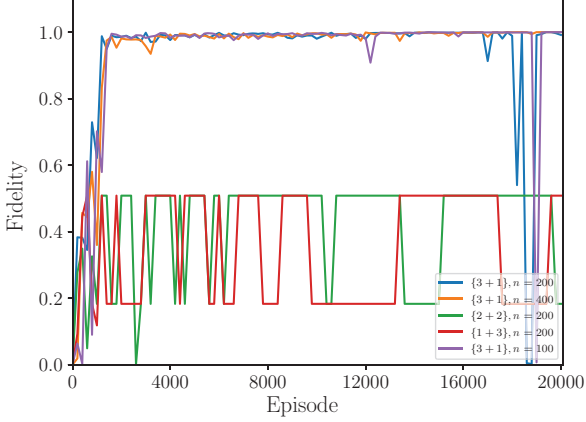
Fig. 5: Learning curves of the RL agent for the Hadamard gate at $T = 1$ with different neural network architectures. With different layer numbers {encoder network + Advantage (Value) network} and neuron numbers n of each architecture.

Table 2: Training time of different algorithms. The computation iterations are the same as in fig. 3 and fig. 4.

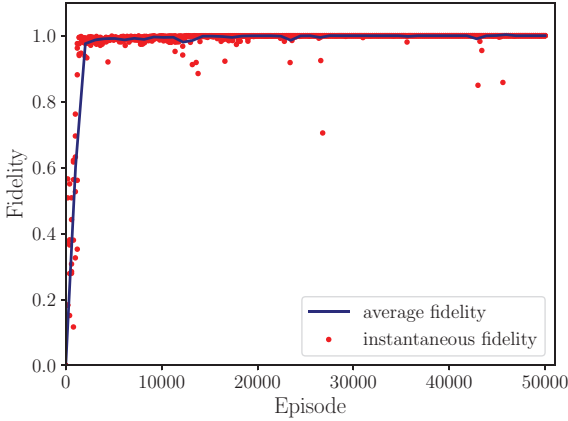| Algorithm	Time	
	Hadamard gate	CNOT gate
GRAPE	< 20 s	about 7 min
GA	about 20 min	about 10 h
DE	about 40 min	about 18 h
RL	about 5 h	about 31 h

Fig. 6: Learning curves of the RL agent for the Hadamard gate at $T = 1$. The red dots show the instantaneous fidelity at every episode with 5 times, while the blue line shows the average fidelity of the 5 agents.

The hyper-parameters of the network are summarized in table 1. As shown in fig. 5, the learning result highly depends on the layer number of the neural network. The computational time is summarized in table 2. Notice that the training time of the two-qubit gate is from 6 to 30 times larger than that of the one-qubit gate. Among all
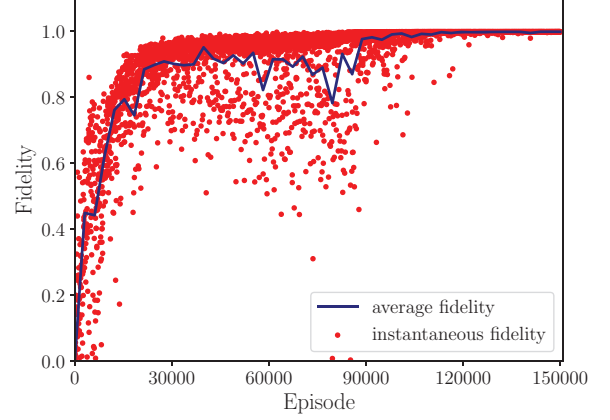

Fig. 7: Learning curves of the RL agent for the CNOT gate at $T = 1.1$. The red dots show the instantaneous fidelity at every episode with 5 times, while the blue line shows the average fidelity of the 5 agents.

algorithms discussed in the paper, the resources needed by our RL agent increase slowest. The learning curves for the two quantum gates are shown in fig. 6 and fig. 7. All algorithms are implemented with Python 3.6, and have been run on two 14-core 2.60 GHz CPU with 188 GB memory and four GPUs.

REFERENCES

[1] HARROW ARAM W. and MONTANARO ASHLEY, *Nature*, **549** (2017) 203.
[2] PRESKILL JOHN, *Fault-Tolerant Quantum Computation*, in *Introduction to Quantum Computation and Information* (World Scientific) 1998, pp. 213–269.
[3] VANDERSYPEN L. M. K. and CHUANG I. L., *Rev. Mod. Phys.*, **76** (2005) 1037.
[4] ISLAM R., EDWARDS E. E. *et al.*, *Nat. Commun.*, **2** (2011) 377.
[5] JURCEVIC P., LANYON B. P. *et al.*, *Nature*, **511** (2014) 202.
[6] BARENDS R., SHABANI A. *et al.*, *Nature*, **534** (2016) 222.
[7] ZHOU BRIAN B., BAKSIC ALEXANDRE *et al.*, *Nat. Phys.*, **13** (2016) 330.
[8] LAROCCA MARTÍN, POGGI PABLO and WISNIACKI DIEGO, *J. Phys. A: Math. Theor.*, **51** (2018) 385305.
[9] NIELSEN MICHAEL A., DOWLING MARK R., GU MILE and DOHERTY ANDREW C., *Phys. Rev. A*, **73** (2006) 062323.
[10] WU RE-BING, LONG RUIXING, DOMINY JASON *et al.*, *Phys. Rev. A*, **86** (2012) 013405.
[11] NANDURI ARUN, DONOVAN ASHLEY, HO TAK-SAN and RABITZ HERSCHEL, *Phys. Rev. A*, **88** (2013) 033425.
[12] ZAHEDINEJAD EHSAN, SCHIRMER SOPHIE and SANDERS BARRY C., *Phys. Rev. A*, **90** (2014) 032310.
[13] HEZAVEH YASHAR D., PERREAULT LEVASSEUR LAURENCE and MARSHALL PHILIP J., *Nature*, **548** (2017) 555.
[14] BIAMONTE JACOB, WITTEK PETER, PANCOTTI NICOLA, REBENTROST PATRICK, WIEBE NATHAN and LLOYD SETH, *Nature*, **549** (2017) 195.

[15] CARLEO GIUSEPPE and TROYER MATTHIAS, *Science*, **355** (2017) 602.

[16] CARRASQUILLA JUAN and MELKO ROGER G., *Nat. Phys.*, **13** (2017) 431.

[17] VAN NIEUWENBURG EVERT P. L., LIU YE-HUA and HUBER SEBASTIAN D., *Nat. Phys.*, **13** (2017) 435.

[18] MNIH VOLODYMYR, KAVUKCUOGLU KORAY, SILVER DAVID *et al.*, *Nature*, **518** (2015) 529.

[19] SILVER DAVID, HUANG AJA, MADDISON CHRIS J. *et al.*, *Nature*, **529** (2016) 484.

[20] SILVER DAVID, SCHRITTWIESER JULIAN, SIMONYAN KAREN *et al.*, *Nature*, **550** (2017) 354.

[21] BUKOV MARIN, DAY ALEXANDRE G. R., SELS DRIES, WEINBERG PHILLIP, POLKOVNIKOV ANATOLI and MEHTA PANKAJ, *Phys. Rev. X*, **8** (2018) 031086.

[22] NIU MURPHY YUEZHEN, BOIXO SERGIO, SMELYANSKIY VADIM and NEVEN HARTMUT, *Universal Quantum Control through Deep Reinforcement Learning*, in *AIAA Scitech 2019 Forum, 7–11 January 2019, San Diego, California* (AIAA) 2019, 2019-0954.

[23] ZENER CLARENCE, *Proc. R. Soc. London A: Math. Phys. Eng. Sci.*, **137** (1932) 696.

[24] SHEVCHENKO S. N., ASHHAB S. and NORI FRANCO, *Phys. Rep.*, **492** (2010) 1.

[25] ZUREK WOJCIECH H., DORNER UWE and ZOLLER PETER, *Phys. Rev. Lett.*, **95** (2005) 105701.

[26] SUTTON R. S. and BARTO A. G., *Reinforcement Learning: An Introduction* (MIT Press) 1998.

[27] WATKINS CHRISTOPHER, *Learning from delayed rewards*. PhD Thesis, Cambridge University Psychology Department (May 1989).

[28] WANG ZIYU, DE FREITAS NANDO and LANCTOT MARC, *Dueling Network Architectures for Deep Reinforcement Learning*, arXiv preprint, arXiv:1511.06581 (2015).

[29] HASSELT HADO V., *Double q-learning*, in *Advances in Neural Information Processing Systems 23*, edited by LAFFERTY J. D., WILLIAMS C. K. I., SHAWE-TAYLOR J., ZEMEL R. S. and CULOTTA A. (Curran Associates, Inc.) 2010, pp. 2613–2621.

[30] SCHAUL TOM, QUAN JOHN, ANTONOGLOU IOANNIS and SILVER DAVID, *Prioritized experience replay*, arXiv:1511.05952 [cs.LG] (2016).

[31] DAY ALEXANDRE G. R., BUKOV MARIN, WEINBERG PHILLIP, MEHTA PANKAJ and SELS DRIES, *Phys. Rev. Lett.*, **122** (2019) 020601.