New Journal of Physics
The open access journal at the forefront of physics

Deutsche Physikalische Gesellschaft DPG
IOP Institute of Physics

Published in partnership with: Deutsche Physikalische Gesellschaft and the Institute of Physics

OPEN ACCESS

RECEIVED
12 December 2022

REVISED
20 February 2023

ACCEPTED FOR PUBLICATION
21 March 2023

PUBLISHED
31 March 2023

PAPER

Quantum generative adversarial imitation learning

Tailong Xiao[1], Jingzheng Huang[1], Hongjing Li[1], Jianping Fan[2] and Guihua Zeng[1,*]

[1] State Key Laboratory of Advanced Optical Communication Systems and Networks, and Center for Quantum Sensing and Information Processing, Shanghai Jiao Tong University, Shanghai 200240, People's Republic of China
[2] Department of Computer Science, University of North Carolina-Charlotte, Charlotte, NC 28223, United States of America
[*] Author to whom any correspondence should be addressed.

E-mail: ghzeng@sjtu.edu.cn

Keywords: quantum machine learning, quantum sensing, quantum control

Abstract

Investigating quantum advantage in the NISQ era is a challenging problem whereas quantum machine learning becomes the most promising application that can be resorted to. However, no proposal has been investigated for arguably challenging inverse reinforcement learning to demonstrate the potential advantage. In this work, we propose a hybrid quantum–classical inverse reinforcement learning algorithm based on the variational quantum circuit with the generative adversarial framework. We find an important connection between the quantum gradient anomaly and the performance degradation, which suggest a gradient clipping strategy to stabilize the training process. In light of the algorithm, we study three classic control problems and the Hamiltonian parameter estimation in quantum sensing with shallow quantum circuits. The numerical results showcase that the control-enhanced quantum sensor can saturate quantum Cramér-Rao bound only with a single variational layer, empirically demonstrating a parameter complexity advantage over the classical learning control. The proposed generative adversarial reinforcement learning algorithm achieves state-of-the-art performance in classical and quantum sensor control in terms of required number of parameters.

1. Introduction

Quantum computation attracts intensive attention from the academy and industry for its unique characteristics such as quantum superposition and entanglement which may provide substantial speedup for classical computation [1, 2]. Demonstrating quantum computational advantage is always a great challenge over the past decade. Numerous quantum algorithms are proposed to demonstrate the quantum advantage based on the different theoretical models. The most encouraging progresses are the experimental study of random circuit sampling [3] and Boson sampling [4] where the quantum advantages are firstly verified in practical superconducting and photonics circuits.

The strong computation capability of fault-tolerant quantum computer stimulates the research interests of using quantum computer to speedup machine learning algorithms [5–8]. Most quantum machine learning (QML) algorithms are the quantum version of classical statistical learning models exploited based on quantum linear algebra. These QML algorithms are assumed to process logical qubit using logical quantum gates based on the quantum oracle model [9]. These QML algorithms are hard to be realized in noisy intermediate-scale quantum (NISQ) devices to show quantum advantage [10, 11]. NISQ machine learning focuses on using variational quantum circuit as a core algorithmic component to demonstrate the potential advantage of quantum computation. The promising candidate capable of quantum advantage is variational quantum circuit (VQC) model [12, 13]. Previous seminal VQC-based QML algorithms concentrate on classification [14, 15] and generative modeling [16] to demonstrate its advantage in handling artificial data [17, 18].

A few studies concerning quantum reinforcement learning (QRL) to show the learning capability to benchmark the results over classical models [19–21]. While RL requires a reward function to be defined for an agent to learn from, inverse RL (IRL) allows us to infer a reward function from expert demonstrations, which can be more difficult to define manually [22]. IRL imitates human behavior which is particularly important for applications where the agent interacts with humans, such as in healthcare or customer service [23]. Besides, IRL can also improve the robustness of agents to changes in the environment, as the inferred reward function is often more generalizable than a manually specified reward function. IRL can be used to learn from large-scale datasets of expert demonstrations, which can be more efficient than collecting rewards manually. Generative adversarial imitation learning (GAIL) is a representative IRL algorithm that uses a generative adversarial network (GAN) to learn a reward function from expert demonstrations. GAIL has been applied to a variety of domains, including robotics [24], game playing [25], and autonomous driving [26], and has shown promising results in learning human-like behaviors. As our knowledge, there is no study on inverse QRL (IQRL) algorithm to exploits the quantum advantage. Consequently, IQRL requires to be investigated to showcase its learning capability and potential advantage in classical controls and more crucial quantum control problems.

In this work, we propose a model-free IQRL algorithm called quantum generative adversarial imitationg learning (QGAIL). QGAIL inherits the architecture of GAIL where the reward function is not required to be designed compared to conventional RL methods. The quantum agent in QGAIL is trained based on a discriminator network in which the expert trajectories are input as the supervised reward signal. The quantum agent imitates the behavior of the expert trajectories aiming to render the discriminator network cannot distinguish the two strategies between the agent and expert. Our QGAIL algorithm adopts an actor-critic architecture where the quantum policy network is trained based on the proximal policy optimization (PPO) method. Furthermore, QGAIL is naturally suited for learning discrete distributions by sampling from the quantum circuits, which may be useful in some complex discrete control problems. Based on QGAIL, we provide plenty of training demonstrations to show the feasibility and parameter complexity advantage in the openAIopenAI gym environment such as the required number of parameters is polynomial fewer than classical RL methods. More significantly, we apply QGAIL to quantum sensing to estimate the parameters of the quantum Hamiltonian. The precision of the estimated parameter can saturate quantum Cramér-Rao bound (QCRB) through quantum controls provided by QGAIL with single variational layer. It is the first study, to our knowledge, of inverse QRL in quantum parameter estimation for quantum sensing. The learning capability and parameter complexity advantage of inverse QRL are highlighted for quantum controls.

The work is organized as follows. In section 2, the related works are discussed. In section 3, the physical model and the hybrid quantum–classical (QC) algorithm are analyzed. In section 4, we introduce two typical applications of QGAIL including classical and quantum sensor controls. In section 5, we present the simulation results based on QGAIL for classical and quantum controls, respectively. Section 6 summarizes the work.

2. Related work

While there has been significant research on VQC-based QML, the investigation of VQC-based RL has been limited. However, there have been several recent developments in this area. For instance, Chen *et al* [27] proposed a QRL algorithm that employs VQC to estimate the value function for discrete state spaces. Lockwood and Si [28] extended this VQC-based QRL to continuous state spaces, and in [29], the authors demonstrated that simple VQC-based Q-networks are insufficient for solving Atari games like Pong and Breakout. Additionally, Jerbi *et al* [30] investigated a hybrid QC algorithm for value-based RL, utilizing an energy-based neural network such as a quantum Boltzmann machine. However, these studies were restricted to value-based QRL methods and evaluated only on classic problems. Jerbi *et al* [20] proposed a hybrid QC policy-based QRL for classic problems and revealed that QRL, as opposed to RL, can solve supervised learning problems based on discrete logarithmic hardness. Furthermore, in [31], the authors presented a hybrid QC policy-based QRL approach to address real-world problems such as vehicle routing. Sequeira *et al* [32] explored a hardware-efficient VQC-based QRL approach for both classical and quantum control problems and demonstrated that VQC requires a smaller number of parameters to solve quantum control problems. Moving on to the full quantum setting, Wu *et al* [33] studied a deterministic policy-based RL method in which both the environment and agent are quantum. They suggested that VQC-based RL can solve quantum control problems with fewer optimizations. Meanwhile, Jerbi *et al* [34] studied the quantum policy gradient algorithm to demonstrate the quantum advantage of the full quantum setting, with VQC potentially providing quadratic speed-ups in sample complexity. Finally, Yun *et al* [35] proposed a quantum multi-agent RL approach based on VQC and demonstrated that it can improve the total reward in a single-hop environment where edge agents offload packets to clouds. In our work, we investigate QGAIL for classical and quantum control problems in the IRL setting. We employ a hardware-efficient shallow VQC to approximate a policy and examine the learning capability of QGAIL and its advantage in parameter complexity. Additionally, we examine the relationship between gradient anomalies and performance.

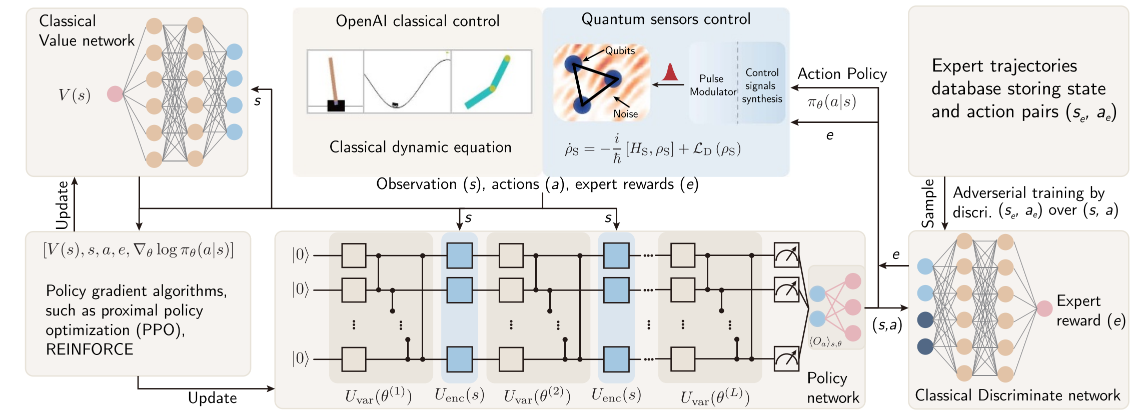
Classical
Value network
$V(s)$
OpenAI classical control
Classical dynamic equation
Quantum sensors control
Qubits
Noise
Pulse Modulator
Control signals synthesis
$\dot{\rho}_S = -\frac{i}{\hbar}[H_S, \rho_S] + \mathcal{L}_D(\rho_S)$
Action Policy
$\pi_\theta(a|s)$
Expert trajectories database storing state and action pairs (s_e, a_e)
Observation (s), actions (a), expert rewards (e)
Update
Sample
Adverserial training by discri. (s_e, a_e) over (s, a)
$[V(s), s, a, e, \nabla_\theta \log \pi_\theta(a|s)]$
Policy gradient algorithms, such as proximal policy optimization (PPO), REINFORCE
$|0\rangle$
$U_{\rm var}(\theta^{(1)})$ $U_{\rm enc}(s)$ $U_{\rm var}(\theta^{(2)})$ $U_{\rm enc}(s)$ $U_{\rm var}(\theta^{(L)})$
$\langle O_a \rangle_{s,\theta}$
Policy network
(s,a)
Expert reward (e)
Classical Discriminate network

Figure 1. The schematic of QGAIL. The algorithm adopts the expert trajectories as the supervised reward to train the policy and value networks. The environment can be classic controls and quantum sensor control in which the emulator can be characterized by classical or quantum dynamics. The value and discriminate networks are classical to evaluate the policy and discriminate the policy from expert trajectories, respectively. The policy network is constructed based on the quantum variational circuit with data re-uploading technique in which the observations are encoded into each variational layer shown in the blue squares in the variational quantum circuit. The parameters of value and policy networks can be optimized via the proximal policy optimization method.

3. Physical model and algorithm

The basic structure of RL consists of two core parts: the agent and the environment. The agent and the environment can be classical or quantum. When we adopt a quantum agent to interact with the classical environment, it is generally referred to as QC in RL. On the contrary, when the environment is quantum, it is referred to as QQ in RL. The CQ and CC can also be defined accordingly. There are many works in CC and CQ such as Alpha Go in playing the game of Go with a human player [36] and classical deep RL method in handling quantum tasks [37–39]. Here we handle QC and QQ tasks based on QGAIL.

A general RL algorithm considers the observations, actions, and rewards. States $\mathcal{S}$ are referred to as the position set of the agents at a specific time-step in the (C/Q) environment. Rewards $\mathcal{R}$ are the numerical values that the agents perform an action $\mathcal{A}$ given an observation from the states. A new state will be internally updated when the environment receives the action. The probability that the agent moves from one state to its successor state is called state transition probability obeying the distribution $p(\cdot)$ with which the environment updates the states. $p(\cdot)$ is updated according to the action the environment received. Notably, a Markov decision process (MDP) is exactly defined that one state moving to another state with $p(\cdot)$ when given an action [40]. At the same time, a reward value is also provided by the environment. An MDP can also be described by a tuple $(\mathcal{S}, \mathcal{A}, p(\cdot), \gamma)$, where $\gamma \in (0,1)$ denotes the discount rate that balances the importance of current reward and future reward.

3.1. Hybrid QC actor-critic network

QGAIL is a quantum version of classical GAIL in which we leverage a quantum policy neural network to replace the classical policy neural network. The quantum policy neural network is chosen as the parameterized quantum circuit (PQC) composed of interleaved rotation and entangling layers as can be seen in figure 1. PQC is proved to be universal in approximating arbitrary functions [41]. However, the structure of PQC has an impact on the final performance in practical situations. In this work, QGAIL consists of one classical value network, one quantum policy network, and one discriminate network. Three networks cooperate to constitute the hybrid QC neural network.

We adopt the data re-uploading technique in PQC to enhance the capability of the model [42] where the classical input $s \in \mathcal{S}$ is encoded by local unitary rotations R_x, R_y, R_z. The final quantum state after the operation of quantum policy network can be given by

$$|\varphi(s, \boldsymbol{\theta})\rangle = \mathcal{U}(s, \boldsymbol{\theta})|0\rangle^{\otimes n}, \tag{1}$$

with

$$\mathcal{U}(s, \boldsymbol{\theta}) = U_{\rm var}\left(\nu^{(L)}\right) \prod_{l=1}^{L-1} U_{\rm enc}(s, \lambda_l) U_{\rm var}\left(\nu^{(l)}\right), \tag{2}$$

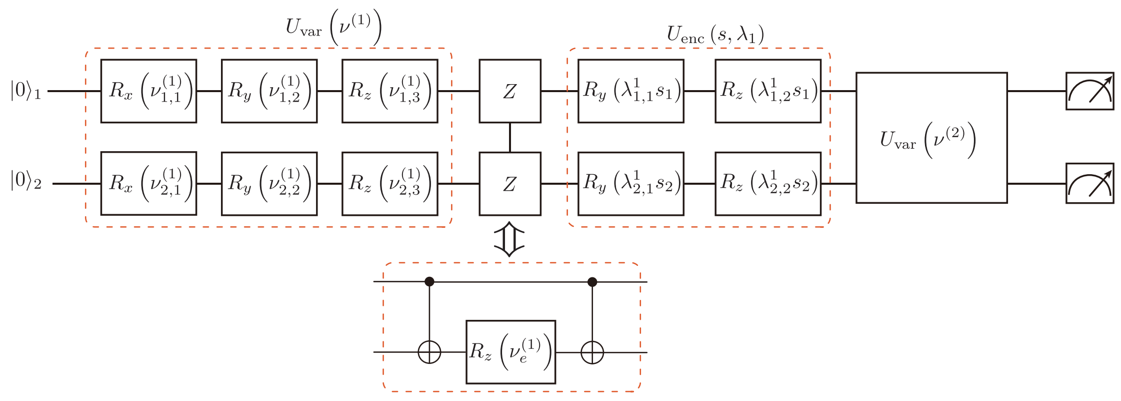
$U_{\text{var}}\left(\nu^{(1)}\right)$
$U_{\text{enc}}\left(s,\lambda_1\right)$
$|0\rangle_1$
$|0\rangle_2$
$R_x\left(\nu_{1,1}^{(1)}\right)$
$R_y\left(\nu_{1,2}^{(1)}\right)$
$R_z\left(\nu_{1,3}^{(1)}\right)$
$R_x\left(\nu_{2,1}^{(1)}\right)$
$R_y\left(\nu_{2,2}^{(1)}\right)$
$R_z\left(\nu_{2,3}^{(1)}\right)$
Z
Z
$R_y\left(\lambda_{1,1}^{1}s_1\right)$
$R_z\left(\lambda_{1,2}^{1}s_1\right)$
$R_y\left(\lambda_{2,1}^{1}s_2\right)$
$R_z\left(\lambda_{2,2}^{1}s_2\right)$
$U_{\text{var}}\left(\nu^{(2)}\right)$
$R_z\left(\nu_e^{(1)}\right)$

Figure 2. A typical PQC with two qubits and two layers. The entangling layer of ZZ gate is trainable with parameter ν_e. The architecture is composed of alternating layers of encoding unitaries $U_{\text{enc}}(s,\lambda)$ and variational unitaries $U_{\text{var}}(\nu)$. The entangling layer can also be CZ gates without training parameters. The number of training parameters for L layers (ZZ circular structure) and $n \geqslant 3$ qubits can be calculated by $(6L-3)n$. In case for CZ entangling gates, the number of parameters scales for $(5L-2)n$.

where $U_{\text{var}}\left(\nu^{(l)}\right)$ denotes the lth variational quantum layer, $U_{\text{enc}}\left(s,\lambda_l\right)$ represents the lth data encoding layer with scaling parameters λ, the trainable parameters $\boldsymbol{\nu}=\{\nu^{(1)},\cdots,\nu^{(L)}\}$, $\boldsymbol{\nu}\in[0,2\pi]^{|\boldsymbol{\nu}|}$ with L denoting the number of layers, and the trainable scaling parameters $\boldsymbol{\lambda}=\{\lambda_1,\cdots,\lambda_{L-1}\}$, $\boldsymbol{\lambda}\in\mathbb{R}^{|\boldsymbol{\lambda}|}$. Encoding and variational parameters are denoted as $\boldsymbol{\theta}=(\boldsymbol{\nu},\boldsymbol{\lambda})$. Variational quantum layer is composed of local qubit rotations and two-local entangling operations given by

$$U_{\text{var}}\left(\nu^{(l)}\right)=\bigotimes_{i=1}^{n-1}\mathrm{CZ}_{i,i+1}\bigotimes_{i=1}^{n}\left[R_x(\nu_{i,1}^{(l)})R_y(\nu_{i,2}^{(l)})R_z(\nu_{i,3}^{(l)})\right], \tag{3}$$

where n is the number of or the size of the state space of RL environment. In reality, we can also choose a CNOT gate or rotatable ZZ gate as the entangling operator. The two-local structure can be linear, circular, and full style. In figure 1, we show a circular entanglement structure. Data encoding layer with scaling parameters can be given by

$$U_{\text{enc}}(s,\lambda_l)=\bigotimes_{i=1}^{n}\left[R_y(\lambda_{i,1}^{l}s_i)R_z(\lambda_{i,2}^{l}s_i)\right], \tag{4}$$

where $\lambda_{i,1/2}^{l}$ denotes trainable scaling parameter of lth encoding layer for qubit i. The state or observation from C/Q RL environment $s=(s_1,s_2,\cdots,s_n)$ with $s_i\in\mathbb{R}$. Finally, the observable O_a is chosen to measure the quantum state. Arbitrary Hermitian operator O_a can be decomposed into $O_a=\sum_i w_{a,i}h_{a,i}$ where $h_{a,i}$ denotes the sub-Hamiltonian in the n-qubit Pauli group $h_{a,i}\in\mathcal{P}_n$. Then the expectation of the action observable via quantum expectation estimation is given by

$$\langle O_a\rangle_{s,\boldsymbol{\theta}}=\sum_i w_{a,i}\langle\varphi(s,\boldsymbol{\theta})|h_{a,i}|\varphi(s,\boldsymbol{\theta})\rangle=\boldsymbol{w}\cdot\boldsymbol{h}_{s,\boldsymbol{\theta}}^{a}, \tag{5}$$

where we have defined $\boldsymbol{h}_{s,\boldsymbol{\theta}}^{a}=\langle\boldsymbol{h}_a\rangle_{s,\boldsymbol{\theta}}$ as the expectation vector by observing each local sub-Hamiltonian. Therefore, the action observation can be obtained by post-processing the local observation with a classical trainable neural layer $\boldsymbol{w}$. For clarity, we let $\boldsymbol{\theta}=(\boldsymbol{\nu},\boldsymbol{\lambda},\boldsymbol{w})$ to denote all the trainable parameters in hybrid QC policy network. The PQC with trainable parameters is displayed in figure 2. The number of training parameters is polynomially reduced compared to classical neural networks.

For discrete action space, we can adopt the softmax operation to obtain the final quantum policy with a tunable temperature β given by

$$\pi_{\boldsymbol{\theta}}^{d}(a|s)=\frac{e^{\beta}\langle O_a\rangle_{s,\boldsymbol{\theta}}}{\sum_{a'}e^{\beta\langle O_{a'}\rangle_{s,\boldsymbol{\theta}}}}. \tag{6}$$

The softmax operation to obtain the quantum policy is necessary for discrete actions. For continuous action space, the softmax operator is not applicable to obtain the quantum policy for it maps the action observation into the value range of $[0,1]$. Since the local observable is in Pauli group, we have $||h_{s,\theta}||\leq 1$ i.e. the expectation value of each local observable is limited in range $[-1,1]$. Then we use a trainable weight to map the observation value to an arbitrary range viewed as the mean value of the quantum policy. Then we also randomly initialize a trainable parameter σ_a^2 as the variance of the quantum policy. Consequently, the quantum policy for continuous control can be given by

$$\pi_{\boldsymbol{\theta}}^c(a|s) = \mathcal{N}\left(\langle O_a\rangle_{s,\boldsymbol{\theta}}, \sigma_a^2\right). \tag{7}$$

Note that we choose Gaussian distribution as the policy distribution according to the common practice.

We adopt the multi-layer perceptron (MLP) to serve as the classical value network used to evaluate the observation from the environment given by

$$V_{W_v}(s) = W_v^{(L_v)}(\cdots(\sigma(W_v^{(1)}s + b_1))\cdots) + b_{L_v}, \tag{8}$$

where σ is the activation function, L_v is the number of layers, $W_v^{(l)} \in \mathbb{R}^{H\times n}, b_{l_v} \in \mathbb{R}^H$ denote the trainable weights and biases and H is the number of hidden neurons. Here we train a separate value network rather than a branch from quantum policy network.

3.2. QGAIL

In previous part, we have presented the hybrid QC actor-critic architecture used to characterize the quantum policy. In IRL setting, it is assumed that the agent has no access to the environment reward. The classical GAIL adopts the generative adversarial learning as the framework to directly train the classical policy network through obtaining the expert reward by feeding the expert trajectories into a discriminator. More details can be found in appendix A.3. QGAIL shares the same algorithmic framework with GAIL but the policy is quantum. QGAIL consists of two training phases: training discriminator and training actor-critic. Training discriminator is accomplished by optimizing a min-max game given by

$$\min_{G_\pi}\max_{D}\mathbb{E}_\pi[\log(D(s,a))] + \mathbb{E}_{\pi_E}[\log(1-D(s,a))], \tag{9}$$

where G_π denotes the quantum policy generator, π_E is the expert policy, D denotes the discriminator used to distinguish the trajectories between the expert and the quantum policy generated. D can be functional approximated by an MLP which is given by

$$D_{W_d}(s,a) = W_d^{(L_d)}(\cdots(\sigma(W_d^{(1)}\mathrm{Cat}(s,a) + b_1))\cdots) + b_{L_d}, \tag{10}$$

where $W_d^{(l_d)}, b_{l_d}$ represent the trainable weights and biases, L_d denotes the number of hidden layers, $\mathrm{Cat}(\cdot)$ operation denotes concatenating the observation and action into a vector.

Maximizing the inner phase can make use of binary cross-entropy cost function to calculate the loss and then the gradients to update the parameters of D. Minimizing the outer phase however can adopt PPO algorithm to update the quantum actor-critic network. The PPO algorithm is a simplified version of TRPO algorithm and achieves the state-of-art performance in numerous RL games [43]. Firstly, PPO calculates the probability ratio between old and new policies at time-step t given by

$$r_t(\boldsymbol{\theta}) = \frac{\pi_{\boldsymbol{\theta}}(a|s)}{\pi_{\boldsymbol{\theta}_{\mathrm{old}}}(a|s)}. \tag{11}$$

Then, PPO imposes the constraint by forcing $r_t(\boldsymbol{\theta})$ to stay within a small interval around 1, precisely $[1-\epsilon, 1+\epsilon]$, where ϵ is a hyperparameter. The clipped objective is given by

$$\mathcal{J}_Q^{\mathrm{CLIP}}(\boldsymbol{\theta}) = \mathbb{E}_t\left[\min\left(r_t(\boldsymbol{\theta})\hat{A}_{\pi_{\boldsymbol{\theta}_{\mathrm{old}}}}(s_t,a_t), \mathrm{clip}(r_t(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon)\hat{A}_{\pi_{\boldsymbol{\theta}_{\mathrm{old}}}}(s_t,a_t)\right)\right], \tag{12}$$

where the function $\mathrm{clip}(r_t(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon)$ clips the ratio to be no more than $1+\epsilon$ and no less than $1-\epsilon$, the advantage function with old policy can be calculated by

$$A_{\pi_{\boldsymbol{\theta}_{\mathrm{old}}}}(s_t,a_t) = Q_{\pi_{\boldsymbol{\theta}_{\mathrm{old}}}}(s_t,a_t) - V_{\pi_{\boldsymbol{\theta}_{\mathrm{old}}}}(s_t), \tag{13}$$

with state-action value function $Q_{\pi_{\boldsymbol{\theta}_{\text{old}}}}(s,a)$ given by

$$Q_{\pi_{\boldsymbol{\theta}_{\text{old}}}}(s_t,a_t) = \mathbb{E}_{(s_{t+1:\infty},a_{t+1:\infty})\sim\pi_{\boldsymbol{\theta}_{\text{old}}}}\left[\sum_{l=0}^{\infty}\gamma^l e_{t+l}\right], \tag{14}$$

and the state value function given by

$$V_{\pi_{\boldsymbol{\theta}_{\text{old}}}}(s_t) = \mathbb{E}_{(s_{t+1:\infty},a_{t:\infty})\sim\pi_{\boldsymbol{\theta}_{\text{old}}}}\left[\sum_{l=0}^{\infty}\gamma^l e_{t+l}\right]. \tag{15}$$

In reality, we can use a truncated version of generalized advantage estimation given by

$$\hat{A}_t = \delta_t + (\xi\gamma)\delta_{t+1} + \cdots + (\xi\gamma)^{T-t+1}\delta_{T-1}, \tag{16}$$

where $\delta_t = e_t + \gamma V(s_{t+1}) - V(s_t)$. Note that ξ is hyperparameter similar to γ, T is the maximum time step and the expert reward is estimated by $e_t = -\log D_{W_d}(s_t,a_t)$. Then, we maximum $\mathcal{J}_Q^{\text{CLIP}}(\boldsymbol{\theta})$ via stochastic gradient ascent method which requires the policy gradient of PQC. The value network is trained through minimizing the mean square error between the accumulated expert reward and state value given by

$$\mathcal{J}^V = \frac{1}{|\mathcal{D}|T}\sum_{\tau\in\mathcal{D}}\sum_{t=0}^{T}\left(V_{W_v}(s_t) - \sum_{k=0}^{T}\gamma^k e_{t+k}\right)^2, \tag{17}$$

where $\mathcal{D} = \{\tau_i\}$ denotes a set of trajectories. The procedure of QGAIL algorithm is presented in Algorithm 1.

The quantum policy step in equation (23) involves calculating the derivative of the CLIP objective function which can be calculated by the derivate of the probability ratio

$$\nabla_{\boldsymbol{\theta}} r_t(\boldsymbol{\theta}) = r_t(\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a|s). \tag{18}$$

The gradient of the log-policy for discrete action space is given by

$$\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}^d(a|s) = \beta\left(\nabla_{\boldsymbol{\theta}}\langle O_a\rangle_{s,\boldsymbol{\theta}} - \sum_{a'}\pi_{\boldsymbol{\theta}}^d(a'|s)\nabla_{\boldsymbol{\theta}}\langle O_{a'}\rangle_{s,\boldsymbol{\theta}}\right). \tag{19}$$

As for continuous action space, the derivative of the log-policy is given by

$$\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}^c(a|s) = \frac{(a-\langle O_a\rangle_{s,\boldsymbol{\theta}})\nabla_{\boldsymbol{\theta}}\langle O_a\rangle_{s,\boldsymbol{\theta}}}{\sigma_a^2}. \tag{20}$$

The partial derivative of equation (5) over observable weights trivially gives rise to $\boldsymbol{h}_{s,\boldsymbol{\theta}}^a$. However, the derivatives with respect to variational and scaling parameters $\boldsymbol{\nu},\boldsymbol{\lambda}$ can be estimated by the parameter-shift rule [44]:

$$\partial_i\langle O_a\rangle_{s,\boldsymbol{\theta}} = \frac{1}{2}\left(\langle O_a\rangle_{s,\boldsymbol{\theta}+\frac{\pi}{2}\boldsymbol{e_i}} - \langle O_a\rangle_{s,\boldsymbol{\theta}-\frac{\pi}{2}\boldsymbol{e_i}}\right), \tag{21}$$

where $\partial_i\langle O_a\rangle_{s,\boldsymbol{\theta}}$ is also called quantum gradient which is the partial derivative of measurement observables over variational parameters. We remark that parameter-shift rule is a standard method to estimate the gradient of PQC over trainable parameters in real quantum device. However, when simulating the quantum circuit in classical computer, back-propagation and adjoint method are faster to be executed compared to the parameter-shift rule. The updating of parameters in value and discriminator networks can be referred to the classical training techniques.

Algorithm 1. Quantum GAIL.

```
Input: Expert trajectories: $\tau_E \sim \pi_E$, initial quantum policy network parameters $\theta_0$, value network parameters ${w^v}_0$ and
    discriminator network parameters ${w^d}_0$, learning rate $\alpha$
1: for $i = 0, 1, 2 \cdots$ do
2:     Sample a batch of trajectories $\mathcal{D} = \{\tau_i\}, \tau_i \sim \pi_{\theta_i}$
3:     Update the discriminator parameters from $w_i^d$ to $w_{i+1}^d$ with the gradient
           $$\hat{\mathbb{E}}_{\tau_i}[\nabla_{w^d} \log(D_{w^d}(s,a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_{w^d} \log(1 - D_{w^d}(s,a))] \quad (22)$$
4:     Take a quantum policy step from $\theta_i$ to $\theta_{i+1}$, using the PPO algorithm with expert reward $\{e_t\}$. Specifically,
    update the parameters by gradient ascent
           $$\theta_{i+1} \leftarrow \theta_i + \alpha \nabla_{\theta_i} \mathcal{J}_Q^{\mathrm{CLIP}} \quad (23)$$
5:     Take a value step to minimize the error function $\mathcal{J}^V$ by using gradient descent algorithm. Specifically,
           $$w_{i+1}^v \leftarrow w_i^v - \alpha \nabla_{w_i^v} \mathcal{J}^V \quad (24)$$
6: end for
```

4. Classic and quantum controls

Classic control is mainly considered to find the control signals to complete the games in environment. We adopt three tasks as the classical environment. They are CarPole-v1, MountainCar-v0 and Acrobot-v1, respectively. These environments require discrete policy to maximize the accumulated reward. We make use of the proposed QGAIL method to handle these classic games by finding out optimal control signals. The detailed specifications including the observation, action space of the environment can be found in appendix B. In general, classic environments can be described by classical dynamics which can be well-characterized with an MDP. The quantum agent learns a good policy, producing a control sequence to render the classical environment complete the task.

On the other hand, we choose a quantum sensor as the representative quantum environment to study the performance of QGAIL in producing optimal quantum control signals. Quantum sensor consists of two critical parts: quantum evolution to sense the unknown parameters and quantum or classical processing unit (quantum processing unit (QPU) or CPU) to generate optimal control signals. The quantum evolution can be characterized by quantum Lindblad equation given by

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} = -\frac{i}{\hbar}[H(\omega), \rho] + \mathcal{L}(\rho), \tag{25}$$

where ρ denotes the density matrix of the quantum sensor, $\mathcal{L}$ is the Lindblad operator to characterize the Markov noise process, $H(\omega)$ is the Hamiltonian of the quantum sensor used to sense the unknown parameters ω. In general, quantum sensor Hamiltonian $H(\omega)$ can be given by

$$H(\omega) = H_0(\omega) + \sum_{j=1}^{p} \mu_j(t) H_j, \tag{26}$$

where $H_0(\omega)$ is the time-independent Hamiltonian of the sensor but encoding the unknown parameters, H_j denotes the jth control Hamiltonian and μ_j is its control field, p is the number of control fields. Note that $H_0(\omega)$ can also be time-dependent evolution. In quantum sensor, one of important goals is to achieve the most accuracy parameter estimation to reach the Heisenberg limit (HL). A key quantity relating HL to parameter estimation is quantum Fisher information (QFI). QFI characterizes the maximum information that can be observed from the quantum state in quantum sensor. QFI can be calculated by

$$\mathcal{F}(t) = \mathrm{Tr}[\rho(t) L_s^2(t)], \tag{27}$$

where $L_s(t)$ is the symmetric logarithm derivative operator that can be obtain by solving the equation

$$\partial_\omega \rho(t) = \frac{1}{2}[\rho(t) L_s(t) + L_s(t) \rho(t)]. \tag{28}$$

In reality, we can estimate $\partial_\omega \rho(t)$ through numerical first order difference over ω. According to the Cramér-Rao bound [45], the QFI provides a saturable lower bound on the estimation can be achieved given by

$$\delta\hat{\omega} \geqslant \frac{1}{\sqrt{M\mathcal{F}(t)}}, \tag{29}$$

where $\delta\hat{\omega} = \sqrt{\mathbb{E}[(\hat{\omega}-\omega)^2]}$ denotes the standard variance of an unbiased estimator $\hat{\omega}$, and M denotes the repeated measurements. The goal of quantum sensor is to find an optimal control sequences to maximize the QFI, thus minimizing the standard deviation.

The quantum sensor evolution is continuous which is not suitable for RL environment. We require discretizing the continuous time evolution into discrete time Δt. To simplify the notation, let $\mathcal{L}_t$ denotes the Lindblad superoperator at time t written as

$$\mathcal{L}_t[\circ] = -\frac{i}{\hbar}[H(t),\circ] + \sum_i \eta_i \left(A_i \circ A_i^\dagger - \frac{1}{2}\left\{ A_i^\dagger A_i, \circ \right\} \right), \tag{30}$$

where $A_i(t)$ denotes the quantum noise operator such as the decoherence and phase damping, $\eta_i \geqslant 0$ means the noise is Markovian, otherwise non-Markovian which beyonds the scope of our work. Then equation (25) can be rewritten as

$$\rho_\omega(T) = \exp\left\{ \int_0^T \mathcal{L}_t \mathrm{d}t \right\} \rho(0) = \lim_{\Delta t \to 0} \prod_k \exp\{\mathcal{L}_{k\Delta t}\} \rho(0), \tag{31}$$

where T is the total evolution time. Equation (31) can be used to simulate the quantum evolution in RL style. The observation for kth interaction with the quantum sensor can be given by

$$s_k = [\Re\{\rho_\omega^{mn}(k\Delta t)\}, \Im\{\rho_\omega^{mn}(k\Delta t)\}, m,n \in \{i,j\}], \tag{32}$$

where $i,j \in [1,2^n]$, n is the number of of the quantum sensor. The actions of the quantum agent are $a = \{\mu_j\}$. Since we consider imitation learning, the reward from the quantum sensor is not required. However, in producing the expert trajectories based on classical RL methods, we should design a reward function to train the agent. The reward function is given by

$$r_k = \frac{\mathcal{F}(k+1) - \xi \mathcal{F}_{nc}(k+1)}{\mathcal{F}_{nc}(k+1)} \times 10, k+1 < T/\Delta t, \tag{33}$$

and for the final time step $T/\Delta t$, the reward signal is amplified with C, i.e. $r \leftarrow C \times r$. $\mathcal{F}_{nc}$ denotes the QFI without control signals. This reward function is different from the classical environments in which the reward is sparse. The reward function of equation (33) uses the QFI as the feedback to evaluate the quality of the quantum controls. Intuitively, maximizing the cumulated reward value can lead to a larger QFI which meets our goal of parameter estimation in quantum sensor. Moreover, we amplify the reward with 10 times so that the gradient norm of the neural network is also amplified when updating the parameters, which is beneficial for the training process.

We remark that the quantum sensor is regarded as the quantum environment to be interacted with the quantum agent in our work. There are some other typical quantum environments such as the quantum gate control aiming to maximize the gate fidelity, quantum circuit optimization to reduce the quantum resources overhead, etc. These quantum environments are highly important and our proposed algorithm can also be leveraged into these tasks to demonstrate the feasibility and potential advantage of QRL.

5. Simulation results

5.1. QGAIL for classic control

To begin with, we conduct the numerical simulation for analyzing the performance of QGAIL in classic discrete environments. To train the quantum agent with QGAIL, we first collect the expert trajectories by using the PPO optimized classical policy to interact with the environments to collect the state and action pairs for Acrobot-v1 and CartPole-v1. PPO is not suitable for training the MountainCar-v0 for its extreme reward sparsity. Therefore, we use Deep-Q network (DQN) to train the agent to collect the expert trajectories as DQN can save the success trajectories into the replay buffer such that the agent can learn from the buffer with multi-epochs.

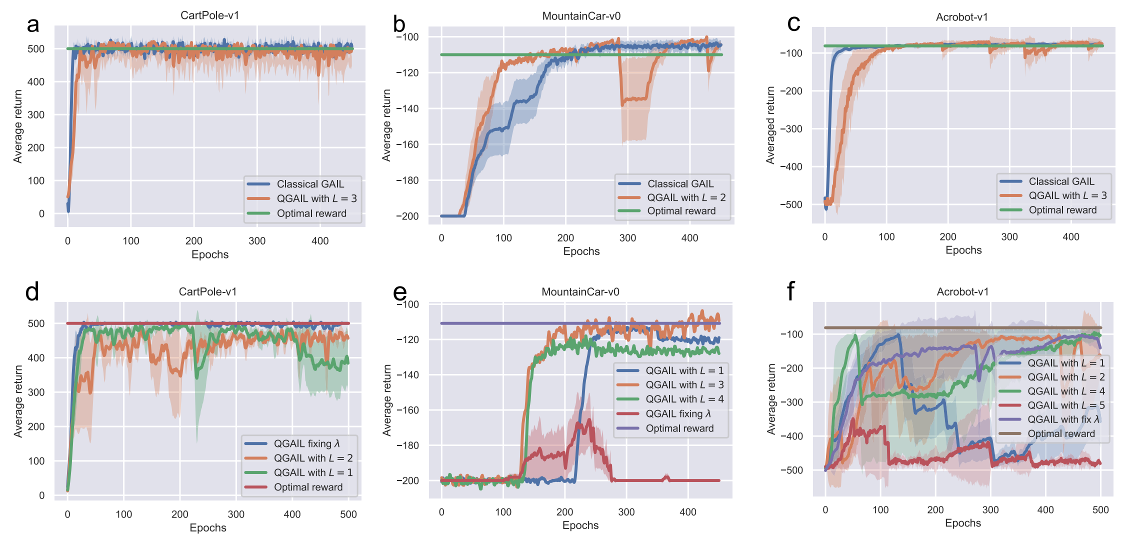

Figure 3. The numerical results of QGAIL for classic control. (a)–(c) are the average returns of QGAIL for CartPole-v1, MountainCar-v0, and Acrobot-v1, respectively, where the layers of the quantum policy networks are chosen to behave best. (d)–(f) are the hyper-parameters tuning details. Each environment is simulated 5 times independently and the shadow area denotes the standard variance.

For each environment, total 100 expert trajectories are collected and each trajectory consists of state-action pairs with the largest horizon of the environment. We make use of three Adam optimizers to update the parameters $\boldsymbol{\nu}, \boldsymbol{w}, \boldsymbol{\lambda}$, respectively. We use another two Adam optimizers to update the parameters of value and discriminate networks. It is important to remark that encoding and weight parameters should have different learning rates for variational parameters. In appendix D, we show that in case the parameters in PQC share the same large learning rate, the model will break and not converge. These results may relate to the small norm of the quantum gradients. In figure 3, we present the numerical simulation results of QGAIL for three representative classic control tasks. In figures 3(a)–(c), we present the best performance of QGAIL where the layers are adjusted to behave well. The convergence speed of our proposed QGAIL is smaller than other QRL methods [19, 20]. This is likely caused by the intrinsic advantage of the IRL algorithm. We remark that the classical GAIL algorithm cannot or hardly work well in the MountainCar-v0 environment since the highly spare reward during interaction with the environment. The inefficiency is amplified in a generative adversarial regime. The online policy gradient methods such as PPO and REINFORCE behave poorly in this scenario. However, QGAIL performs well in these environments which empirically demonstrates its feasibility and efficiency.

We further present the results of different hyper-parameters settings of QGAIL as figures 3(d)–(f) shows. In figure 3(d), we find that different layers of quantum neural network (QNN) have slightly different performances in the CartPole-v1 environment. Especially, fixing or training hyper-parameters λ has no distinct performance difference. Besides, even a single layer of QNN i.e. $L = 1$ can fast saturate the maximum rewards. Compared to the classical neural network, QNN exhibits superior performance in terms of the number of trainable parameters. In figure 3(e), the overall performance of QGAIL is more unstable compared to CartPole-v1. Larger layers have better performance indicating that more variational parameters have a more powerful capability. Fixing tuning parameters behaves poorly. In figure 3(f), we find that the number of layers $L = 5$ of QNN is not promised to show the best performance. $L = 2, 3$ in general demonstrate a more stable and better performance over another number of layers. Fixing parameters is beneficial for the behavior learning process.

In general, more layers i.e. more variational parameters in QGAIL are not necessary to have better performance in classic discrete control environments. We deliberate that when QNN is ‘fat’ (the number of is large), the number of layers should be chosen within a moderate range such as $L = 2, 3$, i.e. shallow quantum circuit. In contrast, when QNN is ‘thin’ (the number of qubit is small), the number of layers can be chosen up to 5 or 6 layers. This empirical observation also implies that the number of trainable parameters should be designed up to a moderate number. Too many or too few variational parameters are likely to lead to poor performance in classic RL environments. This feature is distinct from the classical neural network and may be caused by the Barren Plateau (BP) in optimizing random Haar unitaries. The optimization of the variational parameters of the hardware-efficient QNN remains a challenging problem and there are many efforts aimed to tackle it. We also note that fixing tuning parameters has no significant improvement over classic environments. For different RL environments, we require conducting many simulations to choose the optimal hyperparameters.

During the numerical simulation, we find the training process is not stable as the classical NN. We speculate that this unstable training process is not a principle problem. The simulation software when calculating the gradient of variational parameters (especially the entangling parameters) limits the maximum float precision. When the gradient explodes during training, the performance drops greatly as figure 3(f) shows. In appendix D, we illustrate the gradient norm of different parameter groups during the training process. We find the gradient anomaly is related to the averaged return drop. Therefore, we propose the gradient clipping strategy to mitigate the gradient anomaly to avoid the return drop. We deliberate that the gradient clipping strategy can increase the stability of the learning process. Besides, our QNN does not suffer from BP problems since we design the number of variational parameters into a moderate range. The relation between the gradient anomaly and performance drop can be eliminated by using the gradient clipping strategy.

The classical GAIL also can obtain the optimal rewards as figures 3(a)–(c) shows. Since the classical GAIL is a mature algorithm and investigated extensively, we do expect the our QGAIL can surpass the classical GAIL in terms of the final rewards. However, in figure 3(b), it turns out that QGAIL has a faster convergence speed in MountainCar-v0 environment, an environment with highly sparse rewards. In addition, the number of parameters used to train the QNN is notably less than the classical neural network.

5.2. QGAIL for quantum control

In the quantum environment, we analyze the performance of QGAIL for parameter estimation in quantum sensors. We use the PPO and A3C algorithms to train the quantum agent. The expert trajectories are collected by training the classical PPO and A3C algorithm in a quantum sensor environment. The parameters of the quantum sensor environment determine the quantum evolution. The dephasing and spontaneous emission of the qubit is regarded as the quantum noise effect leading to the purity of the density matrix smaller than 1. We note that the quantum noise also leads to a linear QFI shrink. Quantum control in this work is continuous and we assume the actions are Gaussian distributed.

When we consider the qubit dephasing noise, the evolution can be described by master equation

$$\partial_t\rho_t = -i\,[H(t),\rho_t] + \frac{\eta}{2}[\sigma_{\boldsymbol{n}}\rho_t\sigma_{\boldsymbol{n}} - \rho_t], \tag{34}$$

where

$$H(t) = \frac{1}{2}\omega_0\sigma_3 + \sum_i \mu_i(t)\sigma_i. \tag{35}$$

The control operator $\sigma_i, i\in\{1,2,3\}$ denote the Pauli-X,Y,Z operator. The external control field combining these operators is sufficient to obtain arbitrary single qubit gate. The dephasing direction is given by $\boldsymbol{n} = (\sin\vartheta\cos\phi, \sin\vartheta\sin\phi, \cos\vartheta)$. The parameter to be estimated is ω_0 and we take $\omega_0^{-1} = 1$ as our time unit. The optimal probe state calculated by the standard metrology theory is the superposition state $(|0\rangle + |1\rangle)/\sqrt{2}$. The optimal measurement that extracts the largest QFI is chosen as the projective measurement on the Pauli-X basis and the measurement operator is $\Pi = |+\rangle\langle+|$. This optimal measurement can obtain the largest QFI given the optimal quantum control signals. The dephasing direction is $\vartheta = \pi/4, \phi = 0$ to simulate the quantum noise in the evolution. In this case, we consider the total evolution time $T = 5$ and $\Delta T = 0.1$ thus giving rise to 50 time steps in one episode. The ideal QFI during the quantum sensing process that can be obtained is given by

$$\mathcal{F}(T) = \left(\int_0^T \lambda_{\max}(\partial_\omega H(t)) - \lambda_{\min}(\partial_\omega H(t))\mathrm{d}t\right)^2, \tag{36}$$

where $\lambda_{\max}(\cdot)$ ($\lambda_{\min}(\cdot)$) refers to the largest (smallest) eigenvalue of the operator. The largest QFI in our case thus is T^2, which is related to the quantum speed limit. However, as for the impact of the quantum dephasing noise, the largest QFI cannot be obtained even the optimal control is provided.

As we consider the spontaneous emission noise, we can make use of the following master equation,

$$\partial_t\rho_t = -i[H(t),\rho_t] + \gamma_+\left[\sigma_+\rho_t\sigma_- - \frac{1}{2}\{\sigma_-\sigma_+,\rho_t\}\right] + \gamma_-\left[\sigma_-\rho_t\sigma_+ - \frac{1}{2}\{\sigma_+\sigma_-,\rho_t\}\right], \tag{37}$$

where $\sigma_\pm = (\sigma_1 \pm i\sigma_2)/2$ and the relaxation rates are taken as $\gamma_+ = 0.1, \gamma_- = 0$ throughout our discussion. We note that since our free Hamiltonian only has the Pauli-Z terms and the spontaneous emission noise only

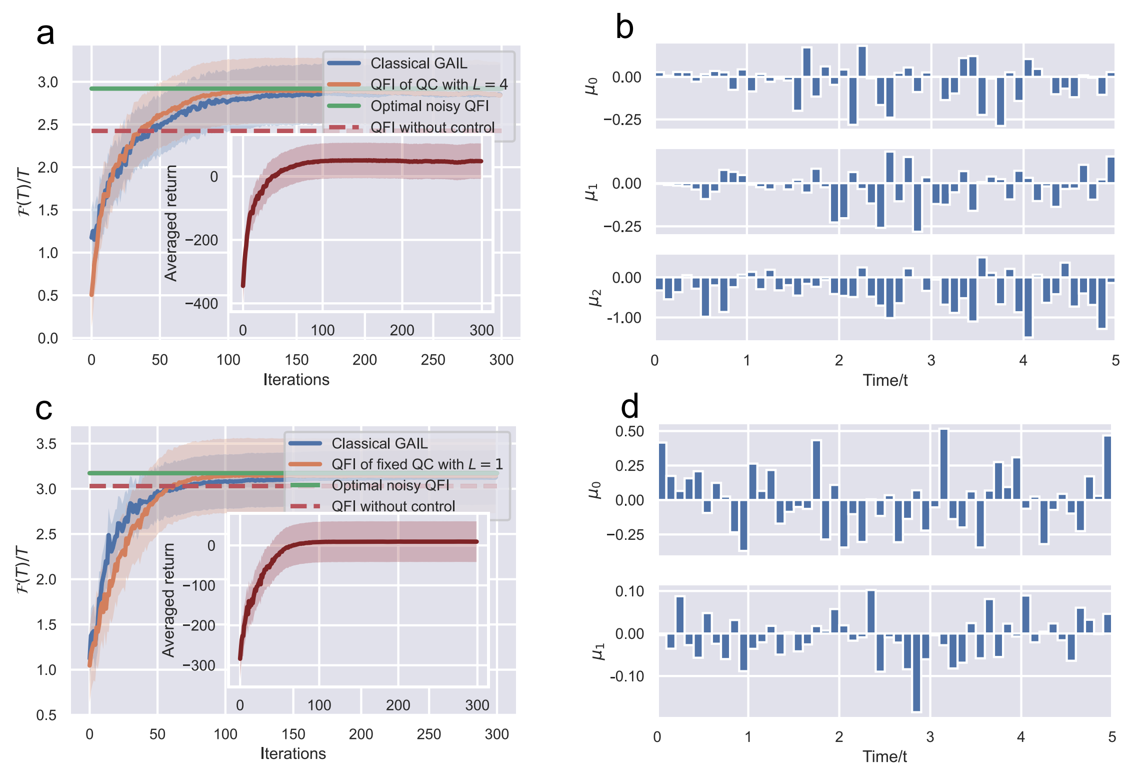

Figure 4. The numerical performance of QC architecture with QGAIL for quantum parameter estimation in quantum sensor environment. (a) and (c) are the QFI curve during the training process in which the total evolution time is $T = 5, \Delta t = 0.1$ and the quantum noise are dephasing and spontaneous emission, respectively. The insert plots in (a)–(c) denote the averaged return. Each simulation is conducted for five independent runs and the shadow area denotes the standard variance. (b)–(d) are the box-shaped quantum control signals produced from the best-trained quantum agent.

affect the Pauli-X and Y terms, we only consider Pauli-X and Y controls. The number of control terms only is 2 compared to the dephasing noise with 3 control terms. Similarly, the ideal QFI also cannot be obtained in spontaneous emission case even given the optimal control. In both cases, although quantum optimal control cannot recover the largest QFI, it shows an QFI enhancement compared to the case without control. The optimal noisy control is the benchmark generated by GRAPE algorithm for fair comparison.

In figure 4, it turns out that QGAIL can produce the optimal control policy to maximize the ultimate QFI of the quantum parameter estimation. The convergence speed is notably fast and the algorithm consumes nearly 100 iterations. The red line in figures 4(a)–(c) denotes the benchmark result of the GRAPE algorithm and the dotted green lines represent the baseline that no control signals are fed into the quantum sensor. Under spontaneous emission noise, it is not necessary to apply the Pauli-Z control so that we only show two control amplitudes as figure 4(d) shows. Overall, quantum control signals can enhance the precision of quantum parameter estimation compared to no control case. QGAIL also shows a competitive performance over classical GAIL in terms of the convergence speed (slightly faster as figures 4(a)–(c) shows). The QGAIL has single variational layer and the number of parameter is significantly less than the classical algorithm. We also fix the tuning parameters to reduce the trainable parameters. The fixed QC architecture does not affect the performance of the QGAIL algorithm. Besides, the learning process is stable and has no return drop since we apply the gradient clipping during training to avoid the gradient anomaly phenomenon. In appendix D, we present more additional results about different L, architectures and longer evolution times. On the other hand, expert trajectories can enhance the learning progress of the quantum policy network through adversarial training. The simulation results imply that hardware-efficient QNN is highly powerful enough and can show practical capability in quantum controls. Compared to QGAIL in classic controls, the performance is more surprising since only a single quantum layer can achieve optimality. By integrating generative adversarial training in RL, we can train the quantum agent without the reward signals from interacting with the quantum sensor environments. The reward design of the environment is a complicated task and hard to engineer in practice. However, the expert trajectories are sometimes easy to be obtained such as in Robotics [46]. The QGAIL algorithm can relax the dependence on the reward design. Moreover, it is related to the quantum generative adversarial neural network, which is widely studied to showcase the potential quantum advantage of quantum variational circuits [47–49].

Table 1. The scaling of training parameters in QGAIL and GAIL for classic and quantum controls.

Algorithm	Policy	Environment	Operator	Param. Comp.
GAIL	On-policy	Classic	State-Value	$\mathcal{O}(n^2)$
GAIL	On-policy	Quantum	State-Value	$\mathcal{O}(n^2)$
QGAIL	On-policy	Classic	State-Value	$\mathcal{O}(n)$
QGAIL	On-policy	Quantum	State-Value	$\mathcal{O}(n)$
QGAIL[a]	On-policy	Quan./Clas.	State-Value	$\mathcal{O}(\log(n))$

[a] QGAIL with amplitude encoding can obtain full logarithmic less parameters compared to classical models.

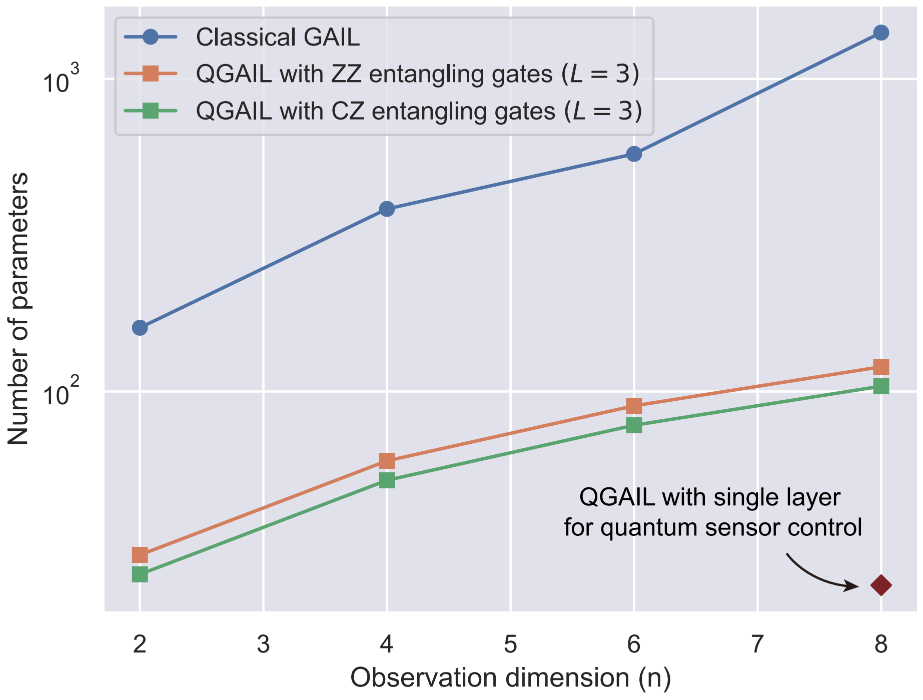

Figure 5. The number of parameters used in GAIL and QGAIL for four different tasks with different observation dimensions. Classical GAIL uses one hidden layer with 32, 64, 64, 128 neurons, respectively. QGAIL uses CZ entangling gates and ZZ gates and 3 variational layers for all observation dimension.

The number of parameters cannot be large such that the QNN can be executed on current devices. Our proposed QGAIL algorithm does not require deep quantum layers but only a single layer is sufficient to saturate the optimal noisy QFI. We summarize the parameter complexity of the QNN for classical and quantum environments as table 1 shows.

The number of training parameters mainly involves memory consumption as figure 5 shows. In GAIL, the parameter complexity in a neural network is commonly viewed as the $\mathcal{O}(n^2)$ scaling [27] for classic controls. Since we can treat the quantum states as classical information, the complexity is still $\mathcal{O}(n^2)$. It is not known that quantum samples with classical processing can show a provable quantum advantage. In figure 5, we simulate GAIL with one hidden layer and $32, 64, 64, 128$ neurons for four observation dimensions. Current deep RL methods generally will choose more hidden layers but we find for our problems, single hidden layer is adequate. In our simulation, the number of parameters surpasses the $\mathcal{O}(n^2)$ since the latter shows the parameter scaling when n is large. For QGAIL where the data is classical, the complexity is scaled as $\mathcal{O}(n)$ for L is generally viewed as a constant scaling [16] as figure 5 shows. The number of parameters shows a linear increment as the observation dimension linearly increases. As for the quantum environment, the complexity is scaled as $\mathcal{O}(n)$ as our simulation results demonstrate or $\mathcal{O}(\log n)$ [50]. Moreover, the scaling may be lower than the case of QGAIL for classic controls. The QGAIL with amplitude encoding has a logarithmic scaling but there is no known efficient algorithm to encode the arbitrary classical data into the quantum memory in a superposition way [51]. The parameter complexity advantage in VQC-based supervised learning is also empirically observed in [52–55]. These quantum algorithms demonstrates the parameter complexity advantage of VQC in QML for classical tasks. For quantum sensor control, based on the quantum input, QGAIL shows a lower parameter requirement compared to classical tasks.

6. Conclusions

In summary, we propose a generative adversarial quantum imitation learning algorithm based on shallow VQCs. We find a critical association between the gradient anomaly and the performance drop, which may widely exist in variational QML models. We deliberate that the gradient clipping strategy is an efficient method to eliminate the gradient anomaly to stabilize the learning process in QML. We argue that our algorithm architecture is flexible and can produce many different IRL algorithms by choosing the classical or quantum version of value, discriminate and policy networks. Based on QGAIL, we testify three representative classic controls in classical environments. It turns out that the total training parameters of the QNN should be restricted to a moderate range to obtain better performance. Besides, we also apply QGAIL to quantum sensing where the learning model is used to generate the optimal controls to steer the state evolution so that the final quantum state can be measured to obtain the maximum QFI. QGAIL is robust against quantum noise such as dephasing and spontaneous emission noise. More surprisingly, QGAIL only requires a single variational layer to produce the optimal quantum control signals so that the parameter estimation can saturate QCRB. The parameter complexity of QGAIL compared with its classical RL models has a polynomial reduction. We further reason that QML may be better suited for quantum problems. Quantum sensing is highly likely to be the most promising application of quantum RL algorithms since the number of required is not large. Therefore, our proposed algorithm is feasible for current NISQ devices. We highlight the importance of generative QML models, which are widely investigated to exploit the advantage in learning discrete distributions. IQRL may be an interesting area that can exploits the expert information to train the agent and it can be used to enhance the sample efficiency of RL and more safe-demanding situations such as automatic driving. In future work, we will study QGAIL in multi-parameter quantum sensing and more complex automatic control tasks.

Data availability statement

All data that support the findings of this study are included within the article (and any supplementary files).

Acknowledgments

The authors thank the fruitful discussion with Binke Xia. This work is supported by National Natural Science Foundation of China (Grant Nos. 61701302 and 61631014).

Appendix A. Classical RL basics

A.1. Classical DQN

Here, we briefly present the classical algorithms of DQN. DQN is an offline RL algorithm which possesses an experience replay memory to store the historical trajectories. The goal of RL is to maximize the cumulated reward (also referred to as return) at time step t written as $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$. The optimal action-value function is defined by maximizing expected return of equation (14) by find an optimal policy $\pi^\star$. The optimal Q function obeys Bellman equation, which is based on such an intuition: if the optimal value $Q^\star(s',a')$ of the sequence $s^{'}$ at the next time-step was known for all possible actions $a^{'}$, then the optimal strategy is to choose the action $a^{'}$ maximizing the expected value of $r+\gamma Q^\star(s',a')$ with

$$Q^\star(s,a) = \mathbb{E}_{s'\sim\mathcal{S}}\left[r+\gamma \max_{a'} Q^\star(s',a') \mid s,a\right]. \tag{A.1}$$

RL wants to estimate the Q function by using the Bellman function as an iterative update rule given by

$$Q_{i+1}(s,a) = \mathbb{E}\left[r+\gamma \max_{a'} Q_i(s',a')|s,a\right]. \tag{A.2}$$

Crucially, the value iteration algorithms converge to $Q^\star$ theoretically [56] when $i\to\infty$. However, in reality we use a function approximator such as a neural network to estimate Q function, i.e. $Q(s,a';\theta)\approx Q^\star(s,a)$. The neural network is referred to as Q network. A Q network can be trained by minimizing a sequence of loss functions $\mathcal{L}_i^Q(\theta_i)$ that changes at each iteration i,

$$\mathcal{L}_i^Q(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot)}\left[(y_i - Q(s,a;\theta_i))^2\right], \tag{A.3}$$

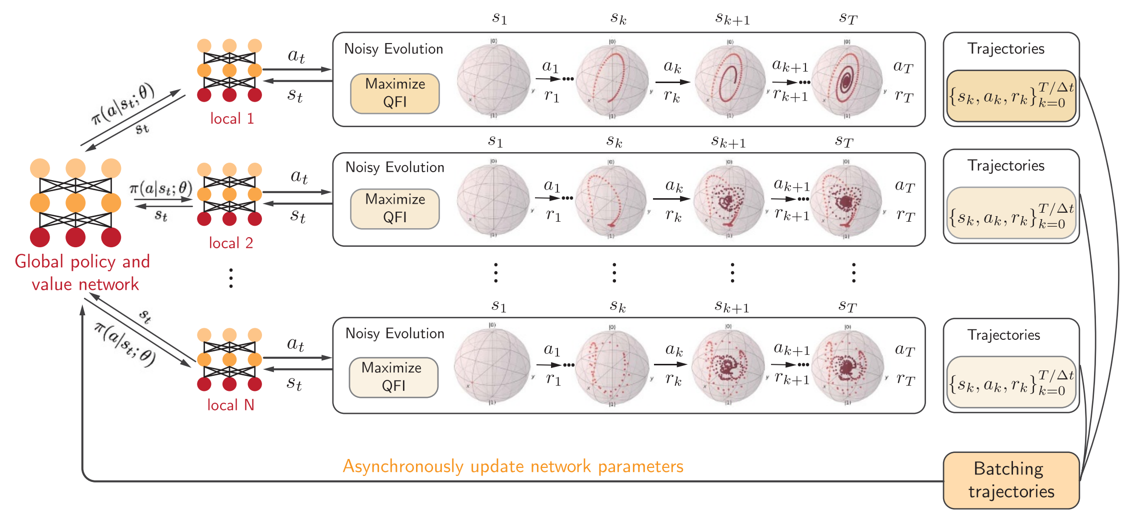

Figure A1. The schematic of A3C for quantum sensor control. There are N local quantum sensor evolutions that are simulated in parallel. Each local agent interacts with its local environment to obtain independent trajectories. These trajectories are batched to train the global actor-critic network. Once the global network is trained and its parameters are synced to each local network.

where $y_i = \mathbb{E}_{s'\sim\mathcal{S}}[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1})|s, a]$ is the target calculated by the trajectories at iteration i and $\rho(s, a)$ denote the behavior distribution. We note that the θ_{i-1} is fixed when optimizing the loss function at iteration $i - 1$. We can use stochastic gradient descent to optimize the Q network by calculating the gradient

$$\nabla_{\theta_i}\mathcal{L}_i^Q(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot);s'\sim\mathcal{S}}\left[\left(r + \gamma \max_{a'} Q(s, a; \theta_{i-1}) - Q(s, a; \theta_i)\right)\nabla_{\theta_i} Q(s, a; \theta_i)\right]. \tag{A.4}$$

The DQN algorithm is model-free and solves RL tasks directly using samples interacting with environments. It is also off-policy which learns about the greedy strategy by maximizing the Q function. The DQN algorithm is highly suited for sparse reward environments such as MountainCar. We remark that the quantum DQN algorithm is also proposed by realizing the Q network with a quantum neural network i.e. PQC. Our work can also be used to exploit the quantum DQN algorithm.

A.2. Classical A3C

Asynchronous advantage actor-critic (A3C) is a representative policy gradient method with a special focus on parallel training. In the task of quantum sensor control, A3C is leveraged to generate optimal control sequences to showcase RL's advantage compared to conventional GRAPE algorithm [57]. In A3C, the critics learn the value function while multiple actors are trained in parallel to keep synced with global parameters. Hence, A3C works well for parallel training since multiple actors can search for more possible actions. In quantum optimal control, as the reward function is QFI-related which is not a direct measure as rewards in classic environments, A3C can increase more explorations of the agent to search for optimal control signals. In addition, A3C has the same theoretical algorithmic framework as actor-critic in GAIL. Here we present a schematic of A3C in quantum sensor environments in figure A1.

A.3. Classical GAIL

Classical GAIL is a representative algorithm in imitation learning where the agent can learn a policy without interaction with the expert reward or access to reinforcement signal. There are two main approaches suitable for imitation learning: behavioral cloning (BC) [58], which learns a policy as a supervised learning problem over state-action pairs from expert trajectories; and IRL [59], which learns a cost function under which the expert is uniquely optimal. BC suffers from the compounding error caused by covariate shift and requires large amounts of data to perform well. On the other hand, IRL learns a cost function that prioritizes entire trajectories over others such that compounding error is not an issue. However, many IRL algorithms are extremely expensive to run, requiring RL in an inner loop [22]. GAIL integrates generative adversarial learning with IRL to overcome the extensive overhead of IRL through directly learning the policy without learning a cost function.

IRL primitive procedure aims to find a cost function such that the expert behaves better than all other policies and is defined with the cost regularized ψ given by

$$\mathrm{IRL}_{\psi}(\pi_E) = \underset{c\in\mathbb{R}^{\mathcal{S}\times\mathcal{A}}}{\arg\max}\left(\min_{\pi\in\Pi} -H(\pi) + \mathbb{E}_{\pi}\left[c(s,a)\right]\right) - \mathbb{E}_{\pi_E}[c(s,a)] - \psi(c), \tag{A.5}$$

where $\mathbb{E}_{\pi}[c(s,a)] = \mathbb{E}[\sum_{t=0}^{\infty}\gamma^t c(s_t,a_t)]$ denotes the expectation with respect to the trajectory it generates where c denotes the cost function, $s_0 \sim p_0, a_t \sim \pi(\cdot|s_t)$, and $s_{t+1} \sim p(\cdot|s_t,a_t)$ for $t > 0$, the expert policy is denoted as π_E, $H(\pi)$ is the γ-discounted causal entropy of the policy π given by

$$H(\pi) = \mathbb{E}_{\pi}\left[-\log\pi(a|s)\right] = -\mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t\log\pi(a|s)\right]. \tag{A.6}$$

Maximum casual entropy IRL seeks a cost function $c \in \mathcal{C}$ that assigns low cost to the expert policy and high cost to other policies. Therefore, the expert policy can be found via one RL procedure given by

$$\mathrm{RL}(c) = \underset{\pi\in\Pi}{\arg\min} -H(\pi) + \mathbb{E}_{\pi}[c(s,a)], \tag{A.7}$$

which maps a cost function to high causal entropy policies that minimize the expected cumulative cost. Designing appropriate ψ leads to different learning regimes. For example, in case ψ is a constant function, equation (A.5) becomes the conventional IRL that can find the optimal cost function but has large overhead. However, in case $\psi$is chosen to be $\delta_{\mathcal{C}}$ where $\delta_{\mathcal{C}}(c) = 0$ if $c \in \mathcal{C}$ and $+\infty$ otherwise, equation (A.5) becomes apprenticeship learning which suffers from incapability of finding a cost function to recover the expert behavior but is more efficient compared with classic IRL.

GAIL design a new cost regularizer to achieve a tradeoff, given by

$$\psi_{\mathrm{GA}}(c) = \begin{cases} \mathbb{E}_{\pi_E}[g(c(s,a))] & \text{if } c < 0, \\ +\infty & \text{otherwise.} \end{cases} \tag{A.8}$$

where

$$g(x) = \begin{cases} -x - \log(1-e^x) & \text{if } x < 0 \\ +\infty & \text{otherwise.} \end{cases} \tag{A.9}$$

The regularizer assigns a small penalty on cost functions c which places an amount of negative cost on expert state-action pairs. However, in case c approaches zero and assigns large costs to the expert, then ψ_{GA} will heavily penalize c. Note that ψ_{GA} builds the connection between generative adversarial learning and imitation learning. To solve equation (A.5), one can instead optimize the min-max game given by equation (9). Discriminator D aims to distinguish between the distribution of state-action pairs generated by policy generator (G) and the expert state-action pairs. When D fails to distinguish state-action pairs generated by G from the expert state-action pairs, then G has successfully learned the distribution of the expert state-action pairs. Therefore, the generator imitates the behavior successfully from the expert trajectories. In a practical setting, we generally use the neural network as the function approximator to represent G and D. Minimizing the phase over the policy generator can make use of PPO or TRPO algorithms which can prevent the policy from changing too much due to noise in the policy gradient. The detailed training process can be referred to as the training trick of the GAN. We present the pseudo-code of GAIL in Algorithm 2.

Algorithm 2. Classical GAIL.

```
Input: Expert trajectories: $\tau_E \sim \pi_E$, initial policy network and discriminator network parameters $\theta_0, w_0$
1: for $i = 0, 1, 2\cdots$ do
2:     Sample a batch of trajectories $\tau_i \sim \pi_{\theta_i}$
3:     Update the discriminator parameters from $w_i$ to $w_{i+1}$ with the gradient
           $$\hat{\mathbb{E}}_{\tau_i}[\nabla_w\log(D_w(s,a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w\log(1-D_w(s,a))] \tag{A.10}$$
4:     Take a policy step from $\theta_i$ to $\theta_{i+1}$, using the TRPO or PPO algorithm with cost function $\log(D_{w_{i+1}}(s,a))$.
   Specifically, take a natural gradient step with
           $$\hat{\mathbb{E}}_{\tau_i}[\nabla_\theta\log\pi_\theta(a|s)Q(s,a)],\ Q(\bar{s},\bar{a}) = \hat{\mathbb{E}}_{\tau_i}[\log(D_{w_{i+1}}(s,a))|s_0=\bar{s},a_0=\bar{a}] \tag{A.11}$$
5: end for
```

Table B1. The classic environments description with observation, action space and reward function. The horizon of each environment limits the interaction time steps during training.

Env.	Obs.	Ctr.	Hor.	Reward	Opt. re.
CartPole-v1	4	Discrete 2	500	$+1$ until termination	500.0
MountainCar-v0	2	Discrete 3	200	$-1+$ height until termination	-110.0
Acrobot-v1	6	Discrete 3	500	-1 until termination	-81.12
Quantum sensor	8	Continuous 3	$T/\Delta t$	r_k	$\mathcal{F}(T)$

Appendix B. Classical environments description

Classic control in this work chooses three typical games and they are CartPole-v1, Acrobot-v1, and MountainCar-v0. To begin with, CartPole-v1 aims to find a good discrete policy to produce controls (moving the cart) that can render the pole in the cart keep in a stable status. The control or action is moving left or moving right. The observation is the position of the cart on the track, the angle of the pole with the vertical, the cart velocity, and the rate of change of the angle. When the agent moves right or left and the pole is still stable (angle is smaller than the threshold), a $(+1)$ reward will be given. The agent should interact with the environment and present a policy to maximize the reward.

Acrobot-v1 aims to swing up a two-link robot. The system consists of two links connected linearly to form a chain, with one end of the chain fixed. The joint between the two links is actuated. The goal is to apply torques on the actuated joint to swing the free end of the linear chain above a given height while starting from the initial state of hanging downwards. All steps that do not reach the goal incur a reward of -1. Achieving the target height results in termination with a reward of 0.

MountainCar-v0 is a deterministic MDP that consists of a car placed stochastically at the bottom of a sinusoidal valley. The possible actions are the accelerations that can be applied to the car in either direction. The goal of the MDP is to strategically accelerate the car to reach the goal state on top of the right hill. Here we summarize the observation space, action space, horizon of the environments, and the optimal reward as seen in table B1. We note that these classic environments can be well described by MDP, thus RL can solve them and show a good performance. In addition, these classic environments obey classical dynamics and the controls are classical.

The environment of quantum sensor control is also summarized in table B1. We only consider one qubit evolution where the number of density matrix elements is 8 and the number of Pauli control terms is $p = 3$. The reward function can be calculated by equation (33). The maximum horizon is determined by the total evolution time T and the time slot Δt. The goal is to achieve the maximum QFI so that obtaining the most accurate parameter estimation.

Appendix C. Expert trajectory generation of the toy games and quantum parameter estimation

Expert trajectories are viewed as the reinforcement signals to train the QC critic-actor networks. In principle, the number of trajectories can affect the ultimate performance of the QGAIL. Intuitively, more expert trajectories will have better average returns with the overhead of large training episodes. In our work, we generate 100 trajectories for three classic environments. Each trajectory consists of state-action pairs with a maximum horizon. Therefore, there are 50 000 state-action pairs for CartPole-v1, 20 000 state-action pairs for MountainCar-v0, 50, 000 state-action pairs for Acrobot-v1. The generation of expert trajectories is based on different Deep RL methods. Specifically, the expert trajectories of MountainCar-v0 are collected via the training DQN algorithm. The expert trajectories of CartPole-v1 and Acrobot-v1 are collected via the training PPO algorithm. The training parameters such as the learning rate and the number of neurons and the layers can be found in appendix E.

For quantum environments i.e. quantum sensor control, we adopt the A3C+PPO algorithm to produce the optimal quantum control sequences. The maximum horizon is $T/\Delta t$. Thus, the number of expert trajectories is $100T/\Delta t$. In reality, we choose two typical evolution times $T = 5, 10, \Delta t = 0.1$ to simulate the quantum sensor evolution. We also consider two typical quantum noise processes such as qubit dephasing and spontaneous emission to study the noisy evolution. The hyperparameters of the A3C+PPO algorithm can also be found in appendix E.

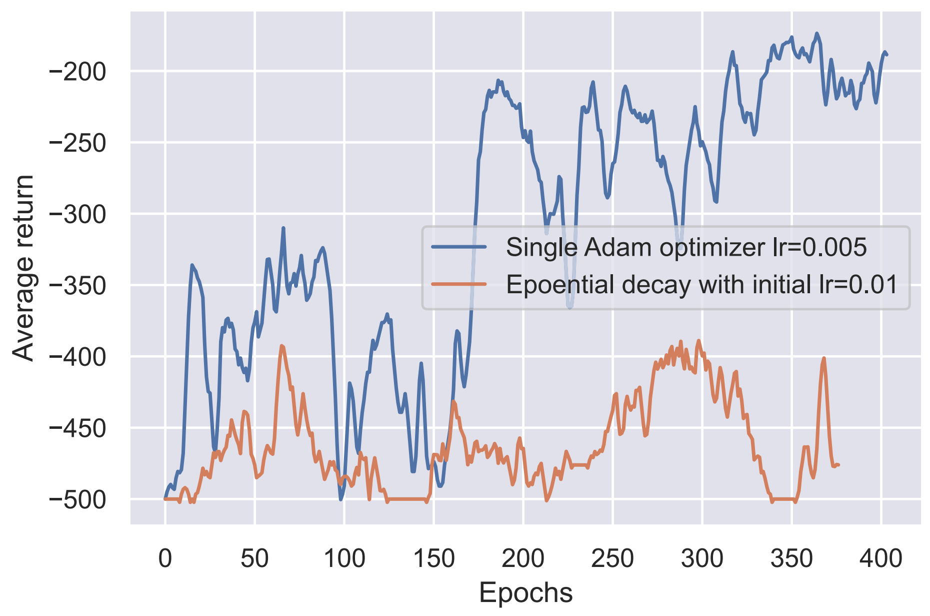

Figure D1. The performance of single Adam optimizer in Acrobot-v1 environment. The blue curve denotes the average return of fixed learning rate. The yellow curve denotes the large initial learning rate with an exponential learning rate decay.

Appendix D. Additional results

We present the additional simulation results of QGAIL in controlling classic and quantum environments. In the main text, we have briefly discussed that a single optimizer for different parameter groups performs worse. The average return is displayed in figure D1. In both two situations, only one optimizer for a different parameter group cannot perform well even if there is an exponential learning rate decay.

Consequently, we can guess that entanglement and variational parameters $\boldsymbol{\theta}_{\text{var}}, \boldsymbol{\theta}_{\text{ent}}$ have distinct updating rule compared to input and output parameters. Quantum gradient over 'quantum' parameters generally has a smaller gradient norm, As a rule of thumb, the learning rate of 'quantum' parameters (showing quantum feature) may be given a smaller learning rate such as $0.01, 0.001$. The 'classical' parameters such as input and output parameters can have a larger learning rate. Recently, there is a study showing that QNN has a better performance with a small learning rate [60].

Here, we present the gradient norm of QNN during training regarding a different group of variational parameters. The gradient norm of the parameter set $\boldsymbol{\theta} = (\boldsymbol{\theta}_{\text{var}}, \boldsymbol{\theta}_{\text{ent}}, \boldsymbol{\lambda}, \boldsymbol{w})$ are recorded separately. Each parameter group has a distinct optimizer and initial learning rate. We do not conduct learning rate decay simulations but do not exclude the possibility of good performance. In the general variational quantum circuit, deep randomly initialized QNN suffers from BP problem, i.e. the gradient vanishes exponentially as the training epoch goes on. In such situation, the QNN cannot be trained well enough. In figure D2, the gradient norm $||\boldsymbol{\theta}||_2$ is recorded with respect to four separate groups. In figures D2(a) and (b), the Acrobot-v1 environment is simulated and the averaged return of QGAIL shows a drop as the blue boxed framed. Interestingly, the gradient norm of entanglement and tuning parameters also have a very large value at the same training iteration. We call the suddenly large gradient norm over entanglement parameters as the gradient anomaly. When the gradient anomaly is observed, QNN has a large parameter updating which leads to model collapse. Note that the gradient anomaly does not always occur for the randomness of the simulation environments. To avoid the model collapse, we can use the gradient clipping technique to manually clip the gradient norm into a restricted range for example setting the maximum gradient norm of specific variational parameter $||\boldsymbol{\theta}||_2 = 3$. We also observe that the gradient norm of output parameters $\boldsymbol{w}$ has a very small value both in figures D2(a) and (c) during the whole training process. Output parameters thus have no large updating gradient. In figures D2(c) and (d), the gradient norm does not have a gradient anomaly, thus the averaged return has no drop. In figure D2, the gradient norm of each parameter group is well distributed in a bounded range. The gradient is not exponentially vanished since we design the number of parameters in a moderate range. Therefore, QNN can be trained well to generate classic control signals with a small number of episodes. Note that each time the averaged return increases to a larger value, the

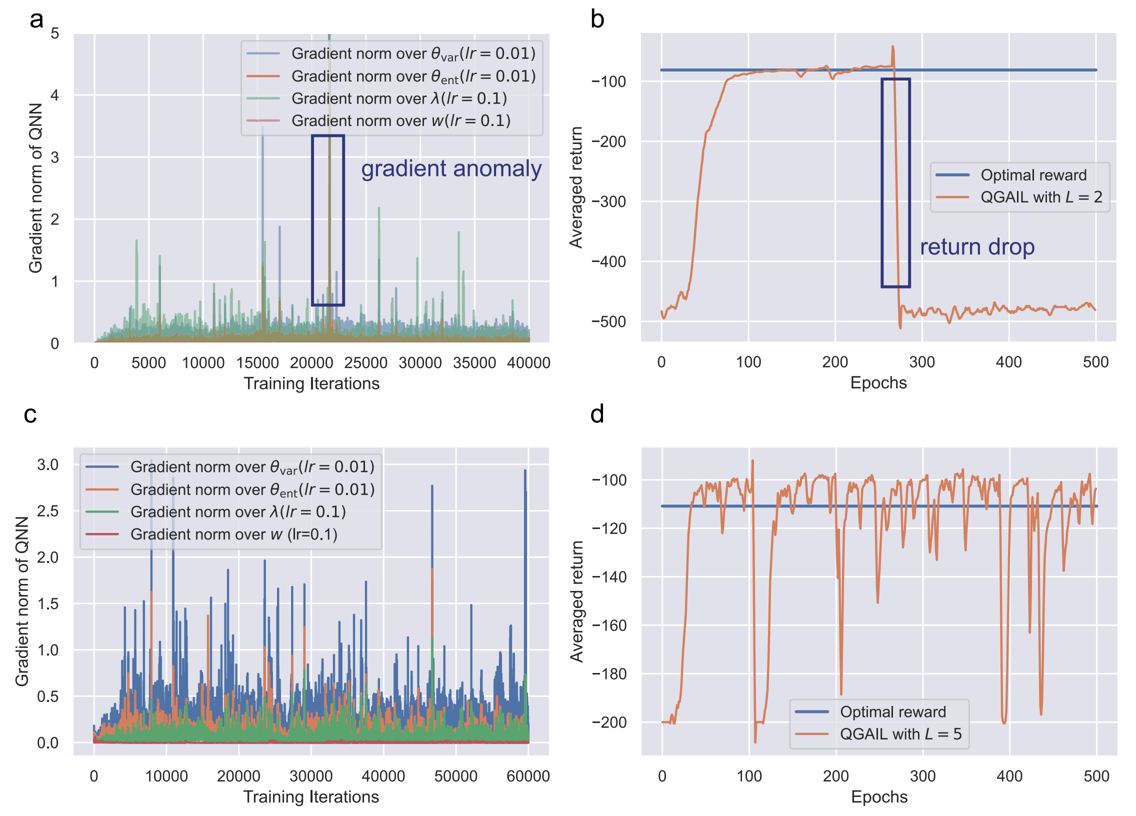

Figure D2. Plotting of the gradient norm over different group of variational parameters during training process. (a) The gradient norm of QGAIL in Acrobot-v1 environment. (b) The averaged return of QGAIL with 2 layers. (c) The gradient norm with respect to four parameter groups in MountainCar-v0. (d) The averaged return of QGAIL with 5 layers.

gradient norm over $\boldsymbol{\theta}_{\text{var}}$ will also increase leading to a subsequent return drop. In case we apply the gradient clipping technique to restrict the gradient value into a specified range, the gradient anomaly does not occur and the averaged return is stable during the whole training process as figure D3 shows.

In the quantum sensor environment, we consider two architectures named classical-quantum–classical (CQC) and QC. The CQC architecture can be applied to the situation where the observation space is large, in which the quantum policy network cannot be classically and efficiently simulated. The first classical network can be viewed as a function of feature extraction or dimension reduction. The extracted feature is subsequently processed by a quantum network. The other architecture is QC where the observations are directly embedded into the multi-layer quantum network. The last classical network in both architectures is aimed to produce continuous quantum control signals. This hybrid QC architecture is flexible and quantum hardware-efficient, which may be commonly used in heterogeneous computing of CPU and QPU.

As figure D4 shows, the numerical performance of the CQC archieture with different variational layers L is illustrated. We can find that both two networks can obtain the optimal QFI but more layers (or parameters) cannot show an advantage in terms of convergence or optimality. Conversely, the performance of $L = 10$ layers is worse than the case of $L = 4$ layers demonstrating that 4 layers are powerful enough to generate optimal quantum controls. Fixing the tuning parameters will reduce the total number of training parameters but do not affect the convergence and optimality as figures D4(a) and (b) shows.

Besides, we simulate the QGAIL for quantum parameter estimation with QC architecture under different L to fully demonstrate the power of the quantum networks as can be seen in figure D5 shows. Figure D5 consists of 12 subplots that show the performance of QGAIL under different settings. Surprisingly, all quantum agents with quantum policy networks can approach or surpass the optimal noisy QFI generated by GRAPE. In figure D5(a), only a single layer of the QNN can showcase the optimality in producing the quantum control signals. Training the tuning parameters has no explicitly advantage compare to fixing λ as figure 4(a). Further, we simulate larger layers such as $L = 4, 6, 10$ to study the performance. We can find that increasing the layers cannot accelerate the training process or obtain a larger QFI, which demonstrates that a single variational layer is powerful enough to generate the optimal control signals. We also find that when

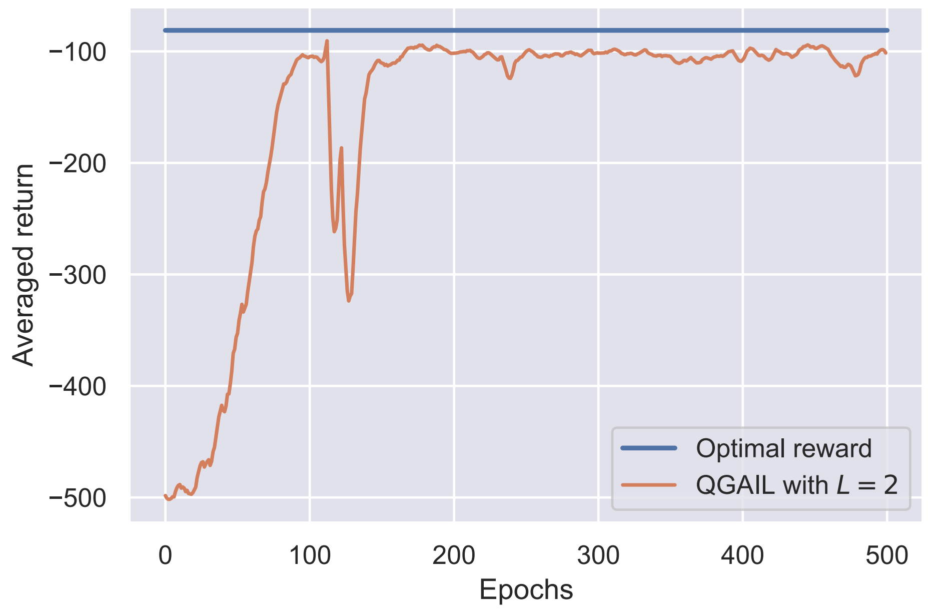

Figure D3. Gradient clipping technique is used to train the QNN and the range of the gradient value is tailored within $[-2, 2]$.

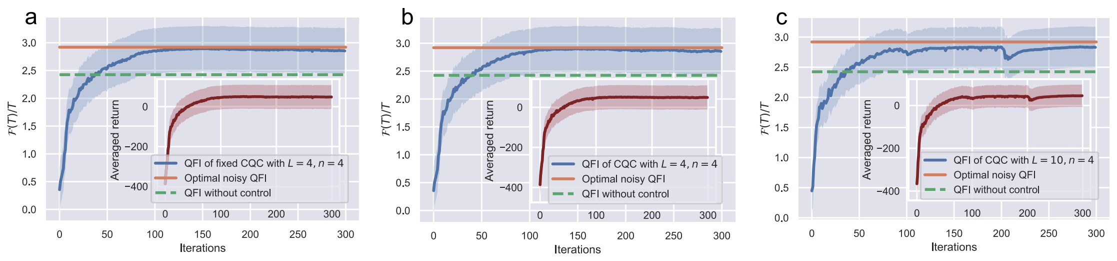

Figure D4. The numerical performance of QGAIL with classical-quantum–classical (CQC) policy for parameter estimation in quantum sensing, where the total evolution time $T = 5, \Delta T = 0.1$ and the dephasing noise rate $\gamma = 0.1$. The number of qubits is $n = 4$ and the first classical neural network is shaped 32×32 with only one hidden layer and the last output classical linear weight is shaped with 4×3 for producing continuous quantum controls. (a) Fixing the tuning parameters λ and $L = 4$. (b) Without fixing the tuning parameters λ and $L = 4$. (c) Without fixing the tuning parameter and $L = 10$.

$L = 6, 10$ as figures D5(d)–(g), the convergence speed is even worse than the case of smaller layers implying that more variational parameters need more episodes (data samples) to train. When considering the case of $T = 10$, the quantum policy generated QFI shows a surpass over optimal noisy QFI even with a single variational layer. Besides, larger layers do not show the problem of BP, which demonstrates that generative adversarial training in RL is likely to avoid the BP problem. This may relate to the cost function where in supervised learning, the global cost function does readily lead to the BP problem when the layers become larger. However, in the IRL scenario, the supervised label is replaced with the reward signal, a weaker label compared to the supervised label. Combining the PPO algorithm which restricts the policy improvement into a specified region, the gradient is reduced stably and will not easily trap into the local minima. When we simulate the quantum agent in controlling quantum sensors with spontaneous emission evolution, the ultimate QFI also approaches the benchmark under $T = 5$. The control enhanced QFI is slightly larger than the case of no control. However, when we simulate longer evolution times, the QFI gap between the control enhanced QFI and QFI without control is amplified. In this case, the quantum agent can produce the optimal control signals to render the ultimate QFI saturate the limit. Through plenty of simulations, we find that QNN may be better suited for controls in a quantum environment compared to the classic controls in the main text. Especially in a quantum sensing application, the information about the density matrix is fed into the QNN to learn a policy that maximizes the QFI. We guess that since our data samples are quantum states which may be better suited for QNN to extract feature information. We remark that although the exact

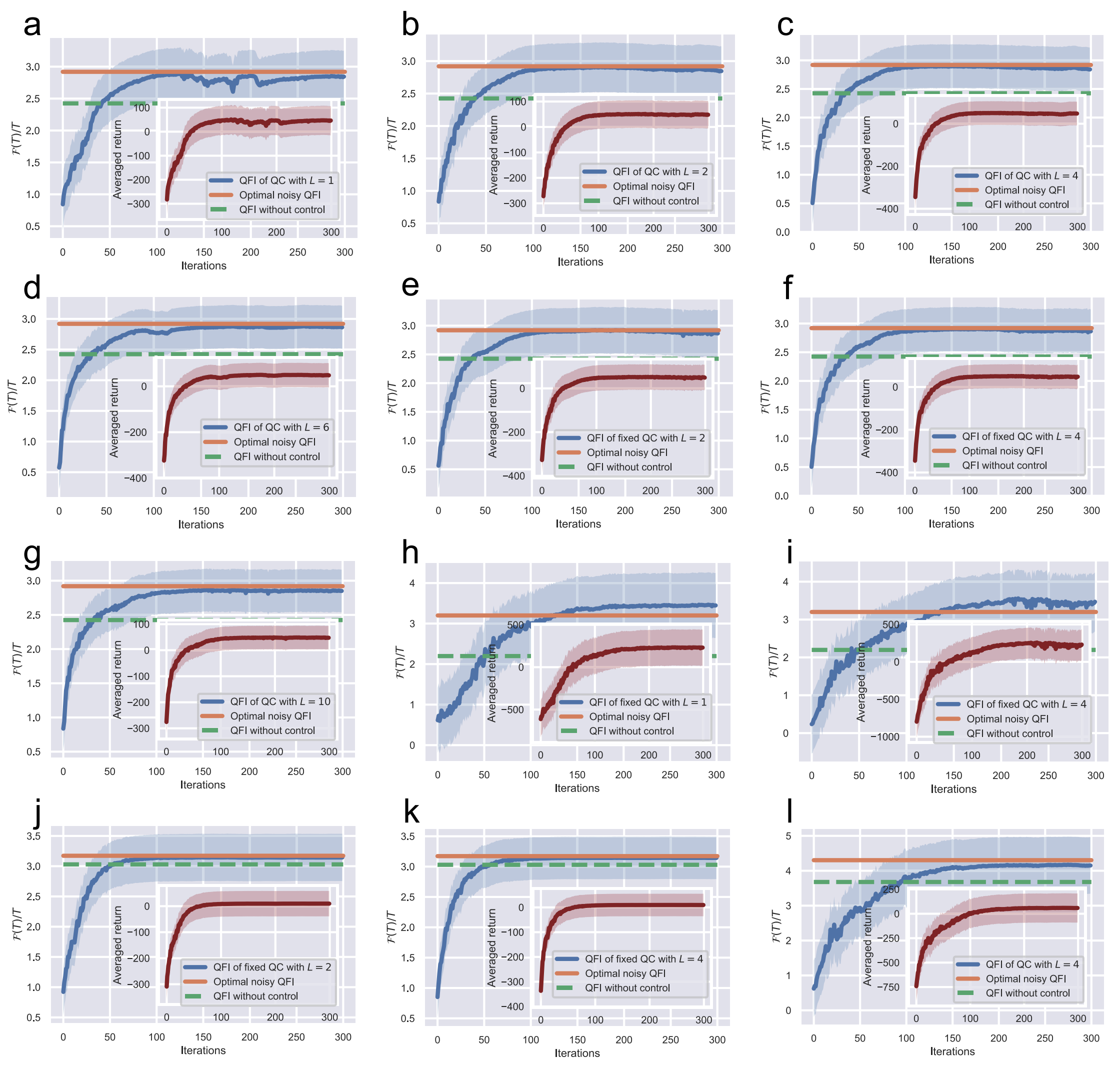

Figure D5. The numerical performance of the proposed QGAIL with QC architecture under different variational layers L, the evolution times T and quantum noise. (a)–(i) are the simulation results under dephasing noise. (j)–(l) are the results under spontaneous emission noise. (a) The total evolution time $T = 5, \Delta t = 0.1, L = 1$ and not fixing the tuning parameters. (b) $T = 5, \Delta t = 0.1, L = 2$ and not fixing the tuning parameters. (c) $T = 5, \Delta t = 0.1, L = 4$ and not fixing the parameters. (d) $T = 5, \Delta t = 0.1, L = 6$ and not fixing the parameters. (e) $T = 5, \Delta t = 0.1, L = 2$ and fixing the parameters. (f) $T = 5, \Delta t = 0.1, L = 4$ and fixing the parameters. (g) $T = 5, \Delta t = 0.1, L = 10$ and not fixing the parameters. (h) $T = 10, \Delta t = 0.1, L = 1$ and fixing the parameters. (i) $T = 10, \Delta t = 0.1, L = 4$ and fixing the parameters. (j) $T = 5, \Delta t = 0.1, L = 2$ and fixing the parameters. (k) $T = 5, \Delta t = 0.1, L = 4$ and fixing the parameters. (l) $T = 10, \Delta t = 0.1, L = 4$ and not fixing the parameters.

form of the density matrix is not regarded as the input of the QNN (amplitude encoded as the initial state), the amplitude and phase information are still obtained and encoded into the QNN. As in [50, 61] demonstrated, the quantum samples are not necessary to be transformed into quantum states but only with the full information of the quantum state are still feasible and beneficial.

Table D1. The hyperparameters summary for all RL methods in this study including PPO, DQN, A3C and QGAIL. In the table, GAE refers to generalized advantage estimation. DQN is used to generate expert trajectories for MountainCar. PPO is used in Acrobot-v1 and CartPole-v1. A3C+PPO is used in quantum sensor environment.

Hyperparameters	Value
Greedy exploration (DQN)	$\varepsilon_0 = 1, \eta_d = 0.9, \varepsilon_e = 0.001$
Batch size	64
Memory size	10^5
Discount rate	$\gamma = 0.99$
Layers and hidden neurons	3, 256
Activation	ReLu
Adam learning rate	5×10^{-4}
GAE (PPO)	$\tau = 0.95$
PPO clipping	$\epsilon = 0.2$
Discount rate	$\gamma = 0.99$
Epochs in one iteration	60
Mini-batch size	512
Batch size for one iteration	2048
Iterations	500
Value network layers and neurons	3, 128
Policy network layers and neurons	3, (256, 128)
Adam leaning rate	3×10^{-4}
Num. local workers (A3C+PPO)	20
Discount rate	$\gamma = 0.9$
Entropy weight	$\eta = 10^{-3}$
Batch size	$T/\Delta T$
Reward fator	$\eta = 1.001$
PPO clipping	$\epsilon = 0.12$
Num. epochs in one iteration	10
Policy layers and neurons	3, 200
Value layers and neurons	3, 128
Activation	Relu
GAE (QGAIL)	$\tau = 0.95$
Discount rate	$\gamma = 0.99$
Batch size for one iteration	horizon $\times$ 30
Mini-batch size	horizon $\times$ 10
Reward fator	$\eta = 1.001$
PPO clipping	$\epsilon = 0.2$
Num. epochs in one iteration	60
QNN layers and qubits	$2 \sim 4$, #states
Value layers and neurons	3, 128
Discri. layers and neurons	3, (256, 128)
Activation	Tanh/Relu
AdamW lr (classical)	3×10^{-4}
AdamW lr (quantum env.)	10^{-3}
Adam lr (classic env.)	$0.1(in), 0.1(out), 0.01(var)$
Gradient clipping range	$[-2, 2]$

Appendix E. Hyperparameters specifications

The hyperparameters designed in this work including the quantum and classical neural networks are presented in table D1. Since GAIL requires the expert trajectories produced from classical RL methods as we have discussed in the previous section, we also present the hyperparameters of classical RL methods. For classical environments, we conduct the numerical simulation of DQN and PPO methods for generating expert trajectories. For quantum environments, we conduct A3C+PPO for quantum sensor control. During the training process, we find that the chosen hyperparameters are not unique and they are crucial for the final performance of machine learning. For example, the mini-batch size and epochs during one iteration should be tuned jointly to control the convergence speed and final performance. In general, for discrete control, we choose the batch size larger than the case in continuous control. The mini-batch size in QNN is better not to be too large since current quantum machine learning simulation packages are not so efficient to support large batch computation. In practical NISQ devices, the large batch computation has two ways: (1) run the same quantum circuit simultaneously on many NISQ devices, and (2) sequentially run the quantum circuit once for a single batch. Both two ways are resource-consuming. We propose training the QGAIL in classical computers and then transferring the parameters into the quantum device with a fine-tuning to adapt to the practical quantum imperfections. This transfer learning proposal can reduce the quantum resource consumption and highlight the advantage of fast inference in NISQ devices.

In classic and quantum environments, the observables used to estimate the quantum policy are different. Through numerical simulation, it turns out that the observables has no vital impact if there are post-processing classical layers. In Acrobot-v1 environment, the observable is $O_a = [w_i\,(Z_1 \cdots Z_6), \cdots]_{1 \leq i \leq 3}$, where w_i is the training parameters. In MountainCar-v0 environment, the observable is $O_a = [w_i\,(Z_1 Z_2), \cdots]_{1 \leq i \leq 3}$. In CartPole-v1 environment, the observable is $O_a = [w_i (Z_1 Z_2 Z_3 Z_4), \cdots]_{1 \leq i \leq 2}$. In continuous quantum control, the observable is $O_a = [\boldsymbol{w}_i \cdot (Z_1, \cdots, Z_8)^T]_{1 \leq i \leq 3}$, where final trainable linear layer is $W \in \mathbb{R}^{8\times 3}$. The final classical layer has smaller size compared to their pure classical models. As in continuous control situation, we also need the variance of the policy. Therefore, we also design the same number of training parameters

Appendix F. Software specifications

Quantum simulation in classical computers is not efficient but in some RL applications, the observation space is not so large so that the qubit encoding technique is still feasible in classical computers. In this study, we make use of Tensorflow quantum [62] to simulate the QNN in discrete classic controls. The classical neural network is based on the Keras implementation. The classical and quantum network constructs a hybrid QC quantum machine learning model and in this model, the data pipeline will automatically calculate the classical and quantum gradients with backpropagation. In continuous quantum control, we make use of Tencent tensorcircuit [63] to build the hybrid model where the backend of the quantum circuit is not based on a state-vector or density matrix but a tensor network. Therefore, tensorcircuit is more efficient than Tensorflow quantum in terms of simulation complexity. Moreover, the tensorcircuit supports GPU, jit, and vmap speedup, so that the training speed of the QNN is greatly enhanced compared to Tensorflow quantum. The classical network is based on the Pytorch module [64]. In the hybrid data pipeline, the classical and quantum gradients are calculated based on backpropagation. The quantum evolution is simulated based on the Qutip package [65].

We implement the hybrid quantum machine learning model in the QSIP artificial intelligence server which consists of two computation nodes and each node has 96 CPU cores and 24 GB Nvidia RTX 3090 GPU. The model of the CPU is Intel(R) Xeon(R) Silver 4210 R CPU @ 2.40 GHz. We also implement quantum machine learning in the SJTU 'Siyuan No.1' HPC cluster which consists of 60 032 CPU cores. The HPC platform supports a large-scale quantum machine learning model.

ORCID iDs

Tailong Xiao https://orcid.org/0000-0002-8741-6900
Jingzheng Huang https://orcid.org/0000-0002-0910-9710
Hongjing Li https://orcid.org/0000-0002-3481-7399

References

[1] Ciliberto C, Herbster M, Ialongo A D, Pontil M, Rocchetto A, Severini S and Wossnig L 2018 *Proc. R. Soc.* A **474** 20170551
[2] Nielsen M A and Chuang I 2002 *Quantum Computation and Quantum Information: 10th edn* (Cambridge: Cambridge University Press) (https://doi.org/10.1017/CBO9780511976667)
[3] Arute F *et al* 2019 *Nature* **574** 505–10
[4] Tillmann M, Dakić B, Heilmann R, Nolte S, Szameit A and Walther P 2013 *Nat. Photon.* **7** 540–4
[5] Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N and Lloyd S 2017 *Nature* **549** 195–202
[6] Rebentrost P, Mohseni M and Lloyd S 2014 *Phys. Rev. Lett.* **113** 130503
[7] Lloyd S, Mohseni M and Rebentrost P 2014 *Nat. Phys.* **10** 631–3
[8] Harrow A W, Hassidim A and Lloyd S 2009 *Phys. Rev. Lett.* **103** 150502
[9] Aaronson S 2015 *Nat. Phys.* **11** 291–3
[10] Preskill J 2018 *Quantum* **2** 79
[11] Mitarai K, Negoro M, Kitagawa M and Fujii K 2018 *Phys. Rev.* A **98** 032309
[12] Farhi E, Goldstone J and Gutmann S 2014 arXiv:1411.4028
[13] Kandala A, Mezzacapo A, Temme K, Takita M, Brink M, Chow J M and Gambetta J M 2017 *Nature* **549** 242–6
[14] Schuld M and Killoran N 2019 *Phys. Rev. Lett.* **122** 040504
[15] Havlíček V, Córcoles A D, Temme K, Harrow A W, Kandala A, Chow J M and Gambetta J M 2019 *Nature* **567** 209–12
[16] Zhu D *et al* 2019 *Sci. Adv.* **5** eaaw9918
[17] Sarma S D, Deng D-L and Duan L-M 2019 *Phys. Today* **72** 48–56
[18] Benedetti M, Lloyd E, Sack S and Fiorentini M 2019 *Quantum Sci. Technol.* **4** 043001
[19] Dunjko V, Taylor J M and Briegel H J 2016 *Phys. Rev. Lett.* **117** 130501
[20] Jerbi S, Gyurik C, Marshall S, Briegel H and Dunjko V 2021 Parametrized quantum policies for reinforcement learning *Advances Neural Information Processing Systtems* **vol 34** pp 28362–75

[21] Skolik A, Jerbi S and Dunjko V 2022 *Quantum* **6** 720
[22] Ho J and Ermon S 2016 *Advances in Neural Information Processing Systems* vol 29
[23] Yu C, Liu J, Nemati S and Yin G 2021 *ACM Comput. Surv.* **55** 1–36
[24] Ravichandar H, Polydoros A S, Chernova S and Billard A 2020 *Annu. Rev. Control Robot. Auton. Syst.* **3** 297–330
[25] Aytar Y, Pfaff T, Budden D, Paine T, Wang Z and De Freitas N 2018 *Advances in Neural Information Processing Systems* vol 31
[26] Kiran B R, Sobh I, Talpaert V, Mannion P, Al Sallab A A, Yogamani S and Pérez P 2021 *IEEE Trans. Intell. Transp. Syst.* **23** 4909–26
[27] Chen S Y C, Yang C H H, Qi J, Chen P Y, Ma X and Goan H S 2020 *IEEE Access* **8** 141007–24
[28] Lockwood O and Si M 2020 Reinforcement learning with quantum variational circuit *Proc. AAAI Conf. on Artificial Intelligence and Interactive Digital Entertainment* vol 16 pp 245–51
[29] Lockwood O and Si M 2021 Playing atari with hybrid quantum-classical reinforcement learning *NeurIPS 2020 Workshop on Pre-Registration in Machine Learning* (PMLR) pp 285–301
[30] Jerbi S, Trenkwalder L M, Nautrup H P, Briegel H J and Dunjko V 2021 *PRX Quantum* **2** 010328
[31] Sanches F, Weinberg S, Ide T and Kamiya K 2022 *Phys. Rev.* A **105** 062403
[32] Sequeira A, Santos L P and Barbosa L S 2022 arXiv:2203.10591
[33] Wu S, Jin S, Wen D and Wang X 2020 arXiv:2012.10711
[34] Jerbi S, Cornelissen A, Ozols M and Dunjko V 2022 arXiv:2212.09328
[35] Yun W J, Kwak Y, Kim J P, Cho H, Jung S, Park J and Kim J 2022 Quantum multi-agent reinforcement learning via variational quantum circuit design *2022 IEEE 42nd Int. Conf. on Distributed Computing Systems (ICDCS)* (IEEE) pp 1332–5
[36] Silver D *et al* 2016 *Nature* **529** 484–9
[37] Xiao T, Fan J and Zeng G 2022 *npj Quantum Inf.* **8** 1–12
[38] Xu H, Li J, Liu L, Wang Y, Yuan H and Wang X 2019 *npj Quantum Inf.* **5** 1–8
[39] Hentschel A and Sanders B C 2010 *Phys. Rev. Lett.* **104** 063603
[40] Sutton R S and Barto A G 2018 *Reinforcement Learning: An Introduction* (Cambridge, MA: MIT press)
[41] Goto T, Tran Q H and Nakajima K 2021 *Phys. Rev. Lett.* **127** 090506
[42] Pérez-Salinas A, Cervera-Lierta A, Gil-Fuster E and Latorre J I 2020 *Quantum* **4** 226
[43] Schulman J, Wolski F, Dhariwal P, Radford A and Klimov O 2017 arXiv:1707.06347
[44] Wierichs D, Izaac J, Wang C and Lin C Y Y 2022 *Quantum* **6** 677
[45] Liu J, Yuan H, Lu X-M and Wang X 2019 *J. Phys. A: Math. Theor.* **53** 023001
[46] Heimann D, Hohenfeld H, Wiebe F and Kirchner F 2022 arXiv:2202.12180
[47] Niu M Y, Zlokapa A, Broughton M, Boixo S, Mohseni M, Smelyanskyi V and Neven H 2022 *Phys. Rev. Lett.* **128** 220505
[48] Zoufal C, Lucchi A and Woerner S 2019 *npj Quantum Inf.* **5** 1–9
[49] Lloyd S and Weedbrook C 2018 *Phys. Rev. Lett.* **121** 040502
[50] Huang H Y *et al* 2022 *Science* **376** 1182–6
[51] Mottonen M, Vartiainen J J, Bergholm V and Salomaa M M 2004 arXiv:quant-ph/0407010
[52] Park G, Huh J and Park D K 2022 *Mach. Learn.: Sci. Technol.* **4** 015006
[53] Li G, Song Z and Wang X 2021 Vsql: Variational shadow quantum learning for classification *Proc. AAAI Conf. on Artificial Intelligence* vol 35 pp 8357–65
[54] Blance A and Spannowsky M 2021 *J. High Energy Phys.* **2021** 1–20
[55] Jerbi S, Fiderer L J, Poulsen Nautrup H, Kübler J M, Briegel H J and Dunjko V 2023 *Nat. Commun.* **14** 517
[56] Fan J, Wang Z, Xie Y and Yang Z 2020 A theoretical analysis of deep q-learning *Learning for Dynamics and Control* (PMLR) pp 486–9
[57] Mnih V, Badia A P, Mirza M, Graves A, Lillicrap T, Harley T, Silver D and Kavukcuoglu K 2016 Asynchronous methods for deep reinforcement learning *Int. Conf. on Machine Learning* (PMLR) pp 1928–37
[58] Codevilla F, Santana E, López A M and Gaidon A 2019 Exploring the limitations of behavior cloning for autonomous driving *Proc. IEEE/CVF Int. Conf. on Computer Vision* pp 9329–38
[59] Arora S and Doshi P 2021 *Artif. Intell.* **297** 103500
[60] Huembeli P and Dauphin A 2021 *Quantum Sci. Technol.* **6** 025011
[61] Huang H Y, Broughton M, Mohseni M, Babbush R, Boixo S, Neven H and McClean J R 2021 *Nat. Commun.* **12** 1–9
[62] Broughton M *et al* 2020 arXiv:2003.02989
[63] Zhang S X *et al* 2022 arXiv:2205.10091
[64] Paszke A *et al* 2019 *Advances in Neural Information Processing Systems* vol 32
[65] Johansson J R, Nation P D and Nori F 2012 *Comput. Phys. Commun.* **183** 1760–72