Proc. of the 9th Intl. Conf. of Asian Society for Precision Engg. and Nanotechnology (ASPEN 2022)
15–18 November 2022, Singapore. Edited by Nai Mui Ling Sharon and A. Senthil Kumar
doi://10.3850/978-981-18-6021-8_OR-02-0142

Investigating Quantum Reinforcement Learning structure to the CartPole control task

Nguyen Truong Thu Ngo[1], Tien-Fu Lu[1], James Quach[2] and Peter Bruza[3]

[1]School of Mechanical Engineering, Faculty of Sciences, Engineering and Technology, The University of Adelaide, SA 5000, Australia
[2]School of Physical Science, Faculty of Sciences, Engineering and Technology, The University of Adelaide, SA 5000, Australia
[3]School of Information Systems, Faculty of Science, Queensland University of Technology, QLD 4059, Australia
Corresponding Author / Email: nguyentruongthu.ngo@adelaide.edu.au, TEL: +61-411-205-833

KEYWORDS: Reinforcement learning, Quantum, Quantum Variational Circuit, CartPole

In recent years, reinforcement learning (RL) has been proven to effectively provide solution to complex problems in various engineering problems such as self-driving cars, industry automation in production lines. These applications also extend to other fields including providing a platform for financial trading and healthcare. Reinforcement learning comes with the trade-offs between exploitation and exploration regardless of applied techniques and algorithms. In more complex engineering system, which often consists of large action and state-space. This space can increase exponentially, and most RL techniques fail to efficiently compute optimal policies to these problems. The inefficiency in RL models leads to the extensive requirement of computational process, which does not often come cheap and affordable. To tackle this challenge, quantum computational models were studied and achieved different levels of success. Since its first theory in the 1940s, quantum algorithm has been improved and advanced to provide exponential speed up compared to classical solutions. Quantum computation offers great potential improvements to traditional RL models due to its ability to create superpositions and entanglement. In past research, quantum variational circuits (QVC) were created as alternatives to neural networks commonly used in RL and experimented on several RL benchmarks in OpenAI Gym environments: CartPole, Acrobat and Lunar Landing. One of such QVCs were designed and tested to achieve greater efficiency in learning speed while offer reduced number of trainable parameter than classical RL. Our research aims to investigate the QVC to balance the CartPole problem running on both local PC simulator and quantum computer. The research signifies the first time a quantum RL could learn to obtain the optimal policy to acquire the maximum expected reward to a control problem and effectively apply the trainable parameter to balance the pole. The system is first tailored into a RL environment with state space and action space. For each time step, the input data from the environment (state space) is encoded into quantum states. Through the QVC, the algorithm learns to optimize the RL policy to calculate the probability of future action to obtain the optimal reward for each state-action pair. The trainable parameters from the QVC are again optimized using gradient descent. The QVC design is vital to the success of the RL model; therefore, we vary the QVC gates and investigate the model performance. It is expected that the Quantum RL model would outperform the classical RL in learning the optimal policy in term of speed and computational resources. The successful control of the pole on a quantum simulation would be proof that quantum models could offer real solution to future computation-intensive problems where classical solutions are unaffordable.

NOMENCLATURE

H – Hadamard gate
Rx – x-axis rotation
Ry – y-axis rotation
Rz – z-axis rotation
θ – trainable parameter

1. Introduction

Model-free techniques like RL has been widely employed as a machine learning model that has the potential to efficiently solve control and decision-making problems. RL emerged recently as the world shown interest in machine learning and has achieved notable results in several applications. These include self-playing chess and shogi (Silver et al., 2017), playing StarCraft 2 game (Vinyals et al., 2019) and learning to configure a nuclear fusion problem (Degrave et al., 2022). RL was also used to produce controllers for multilegged robot that can perform complicated task (Peng et al., 2020). The general application for RL can be narrowed down to decision making problem where a learning agent interact with the environment and choose an action that contributes to completing the task. The challenge is to define and optimise the policy and reward function so

that the agent can effectively learn to support the wearer. The actions and reward can be viewed as trials and errors as the agent learns to choose the best action to get the highest expected reward (Hsiao et al., 2022). Even though RL has been applied and achieved significant results, the difficulty when designing a RL model is that the state-action space of the tasks could be exponentially large (Du et al., 2020). It is challenging for RL techniques to balance between exploitation and exploration the larger the space. Exploration is the act where the RL agent searches and understand the environment through the state-action space, whereas exploitation is where it saves the information. The balance between the exploration and exploitation optimises the RL model policy leading to best cumulated rewards. Nevertheless, when the state-action space is too large and requires immense and expensive computational power, the optimal policy cannot be explored efficiently (Hsiao et al., 2022). This inefficiency affects the ability of RL models in solving larger-scale problems which present in many real-world tasks. Choosing a RL model that has improved efficiency (speed of convergence) is the key to many applications.

This study selects quantum computing as a solution to tackle the RL inefficiency. A potential candidate for our design is a quantum RL agent, which is benchmarked on an OpenAI Gym task – CartPole (Brockman, 2016). OpenAI Gym contains widely used RL benchmarking environments to compare RL models performance. The question here remained how a quantum RL algorithm should be designed and chosen to gain a performance advantage over classical RL techniques; this is the gap in the field of quantum reinforcement learning that we try to address.

2. Model Design

2.1 Benchmarking task

In this study, CartPole is considered as a benchmarking task for testing. In the CartPole task, the agent aims to balance a pole placed on a cart by moving the cart left and right as shown in figure 1. The agent is rewarded +1 for each time step the pole stay upright. The goal is to balance the pole until the reward reaches 500 (balance for 500 steps). The episode fails when the pole falls above 12 degrees from the initial vertical point, or the cart move more than 2.4 units from the initial position. The benchmarking task allows us to test our RL model without the need to build another environment. The task is simple but provide a reliable way to compare each RL models and algorithms. The criterion for comparison is the number of episodes it takes to achieve the reward of 500.

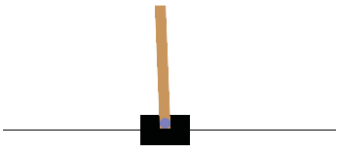

Figure 1: CartPole Task

2.2 Reinforcement Learning

A RL environment is designed with a set of states' S and a set of actions A which the agent can perform in order to change the environment states. By performing an action and changing the environment states, the agent obtains a reward which evaluates the quality of that action based on the learning objective. The simple goal is to maximise the total reward over the learning episode and the strategy to achieve this goal is called policy.

Mathematically, a classical RL can be described by Markov decision process: $M = (S, A, P, R, \gamma)$ Where S is the state space, A is the action space, $P(s'|s,a)$ is the probability of transitioning from state s to state s' after performing action a, R is the reward space and γ is the discount factor . The learning process starts at an initial state s0 and implements the policy $\pi(a_i|s_i)$ to pick action a_i at time step i from state s_i to move to state s_{i+1}. Each action rewards the agent with r_i. The goal is to maximise the discount expected reward according to the equation:

$$R_i(\tau) = \hat{E}_\pi\left[\sum_{i=0}^{T} \gamma^i r_i(s_i, a_i)\right] \quad (1)$$

Where $\tau = (s_0, a_0, s_1, a_1, \ldots, s_i, a_i)$ are the state-action pairs in an episode. $\hat{E}_\pi$ is the expectation value under all policies. We implement the proximal policy optimization (PPO) algorithms as our training model (Schulman et al., 2017). The proximal policy optimization agent is a policy gradient algorithm containing an actor and a critic. The actor is responsible for the policy $\pi_{\theta i}(a_i|s_i)$, where θ_i is the trainable parameter and chooses action a_i during state s_i at time step i follows the policy for each episode of training. The actor receives the state features as inputs through the QVC and computes a list of probabilities for each available action of the task. The probabilities form a distribution, and an action is chosen based on this. The critic represents the state value function $V(s)$ and computes the discount reward. The agent objective is to learn and improve the policy by updating the trainable parameter via gradient ascent:

$$\theta_i \leftarrow \theta_i + \eta \nabla L(s_i, \theta_i) \quad (2)$$

Where L is the loss function and η is the learning rate. The loss function is defined by the objective function PPO-clip proposed by (Schulman et al., 2017) in equation 3. Experiments have proved the robustness and efficiency of PPO-clip with good performance.

$$L(s_1, \theta_i) = \hat{E}\_\pi\ \left[\min(r_i(\theta_i)\,\hat{A}_i, \text{clip}\,(r_i(\theta_i), 1-\varepsilon, 1+\varepsilon)\,\hat{A}_i)\right] \quad (3)$$

Where $r_i(\theta_i) = \frac{\pi_{\theta i}(a_i|s_i)}{\pi_{\theta' i}(a_i|s_i)}$ denotes the ratio between the new and old policy, θ'_i is the trainable parameter before being updated and ε is a hypermeter in the form of a small number (Schulman et al., 2017).

2.3 Quantum Variational Circuit structure

In quantum computation, states are represented in qubits. Two of the computational basic states of qubits are $|0\rangle$ and $|1\rangle$, which corresponds to the state 0 and 1 in classical bits (Nielsen and Chuang, 2002). Qubits are denoted in Dirac notation and contain a characteristic that make it distinctive in computational performance

called superposition. Unlike classical bits, which can only be flipped between 0 and 1 states, quantum qubits can form linear combinations of the basic states, e.g., ψ= α |0⟩+β |1⟩. α and β are complex number which means a qubit state is a 2-D complex vector space. Due to superpositions, information stored in N qubits scales by O(2^N) instead of O(N), which is an advantage. Superposition has a probabilistic nature, the probabilities of a qubit collapsing to its states must add up to 1, e.g., |α|^2+ |β|^2=1. When measurement is performed on a qubit, the superposition only collapses into one state, either 0 or 1 based on its probability.

QVC provides computational advantage by using parameterized circuit running in quantum environment and output the parameter to classical optimizer (Cerezo et al., 2021). QVC also uses the vast Hilbert space to encode the RL environment's states. The CartPole task contains four states features making up its entire state space: cart location, cart velocity, pole angle and pole angle velocity, which are denoted as s^0, s^1, s^2, s^3 respectively. The environment sends its state features s_i^q at time step i to be encoded into quantum state as $|\Psi_{in}(s_i)\rangle = U(s_i)|0\rangle$, q is the state feature number, $|0\rangle \in \mathbb{C}^{2^N}$ is the initial quantum basic states, the number of qubits N depends on the application. In this task, we only use 1 qubit. Operator $U(s_i)$ is a unitary containing the set of quantum gates dependent on s_i. The set contains a Hadamard gate to create superposition and three single rotation gates to encode states s_i^1, s_i^2, s_i^3 as angles of the rotation. For this task we only use the cart velocity, pole angle and pole angle velocity as our input state features and Rx, Ry, Rz as the three gates to encode our features

The encoded state $|\Psi_{in}(s_i)\rangle$ then undergoes the unitary operator $U(\theta)$ which contains the trainable parameter θ in the QVC. $U(\theta)$ contain the Rx gate to encode the trainable parameter into the circuit. The last operator of the QVC measures the quantum state in the form of circuit output in equation (4).

$$f(s_i, \theta) = \langle 0|U^{\dagger}(s_i, \theta) M U(s_i, \theta)|0\rangle \quad (4)$$

Where $U(s_i, \theta_i)$ is the combination of the first two unitary operator dependent on the state input and trainable parameter θ_i. M is the measurement operator.

The output is then scaled by the following linear equation.

$$y'_{ik} = a_{ik} \times f(s_i, \theta) + b_{ik} \quad (5)$$

Where a_{ik}, b_{ik} are the trainable parameter at time step i while k is the action index available in the environment. The scaled output decides whether the policy chooses action a based on the probability:

$$P_{\theta i}(a_i|s_i) = \pi_{\theta i}(a_i|s_i) = Softmax(y'_{ik}) = \frac{e^{y'_{ik}}}{\sum_{k=1}^{p} e^{y'_{ik}}} \quad (6)$$

Where a_i is the corresponding action of state s_i and p dictates the number of available actions in the RL task. $Softmax$ indicates a mathematical function that converts the scaled output into a vector of probabilities. The agent receives a reward after the above steps and the next state feature s_{i+1}. The loss function is updated using the scaled output and cumulative reward. The entire model is summarized in figure 2.

3. Preliminary Results

In our preliminary investigation, we focus on the effect of QVC design on the efficiency of the training model. The criterion is the number of episodes the model takes to complete the CartPole task, i.e., achieving an average reward of 500 under 300 episodes. The average reward is calculated using the result in the previous 20 episodes of the run. Classical algorithms have shown to achieve an average reward of 500 under 500 episodes (Hsiao et al., 2022). In order to investigate the computational advantage of using QVC over classical RL, we implement a threshold of 300 episode for the model to achieve 500 average rewards in order to complete the task. In the results below, we vary the rotational gates used to encode the input state features into the QVC and obtain different levels of training performance.

In the first instance, we use Rx, Ry and Rz to encode the three state features: cart velocity, pole angle and pole respectively. The model could learn to achieve a 500 reward in less than 50 episodes. However, the learning algorithm struggles to maintain the performance and unable to reach the average reward of 500 within 300 episodes. This QVC fails to complete the CartPole task. Figure 3 shows the performance of the QVC using Rx, Ry and Rz. Our second set of gates contains Ry, Rz and Rz encoding the state features in the same order. Figure 4 shows the performance of this set of gates. We now see the model has completed the task under 200 episodes, which reflects the better performance over classical models. This time the model takes over 100 episodes to reach the 500 rewards for the first time. This QVC design is more reliable than the previous set and

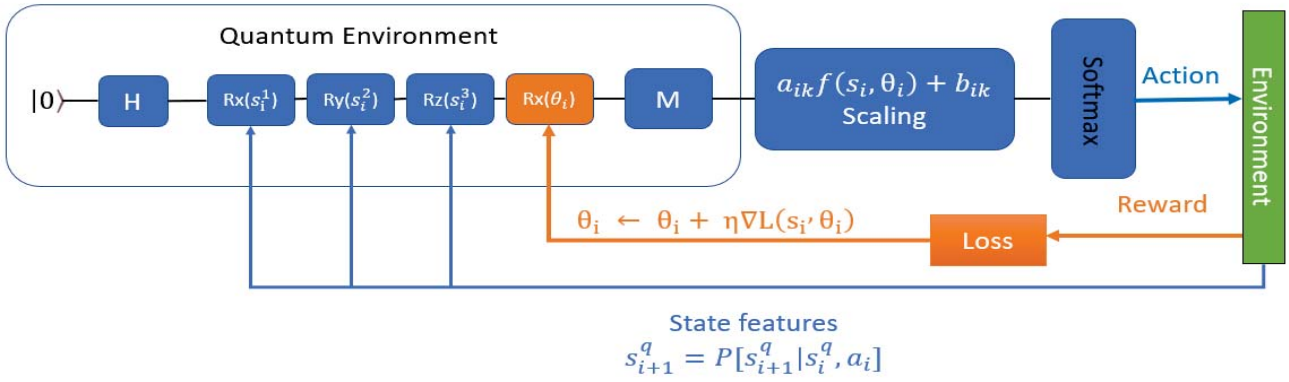

Figure 2: One QVC structure example in PPO RL algorithm (Hsiao et al., 2022)

manage to satisfy our criterion. Our third set results in a major increase in performance when it completes the task under 120 episodes, which is illustrated in figure 5. The consistency of this QVC is worth mentioning since it maintains the 500 rewards for the rest of the run. This is our best result yet on the CartPole task.

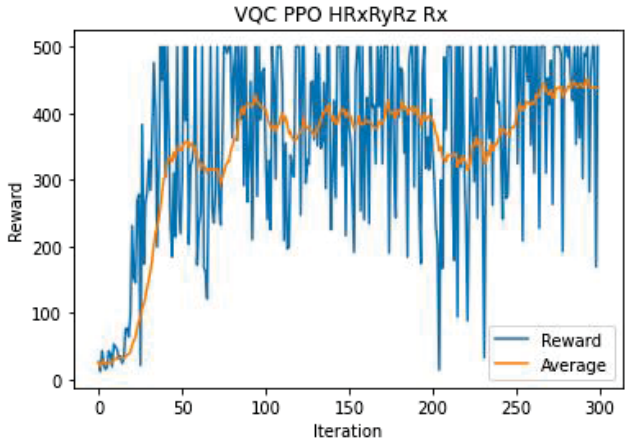

Figure 3: QVC with Rx, Ry, Rz gates

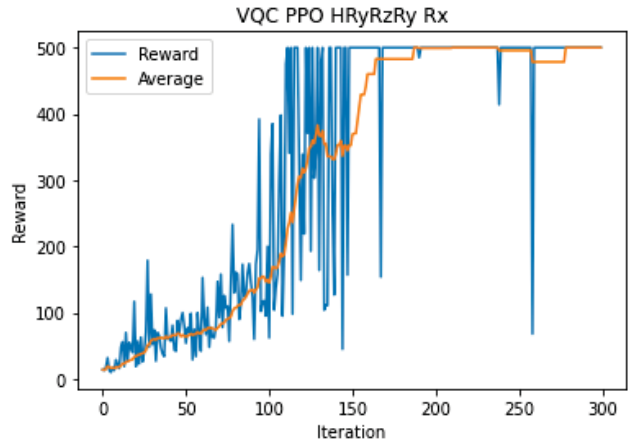

Figure 4: QVC with Ry, Rz and Ry gates

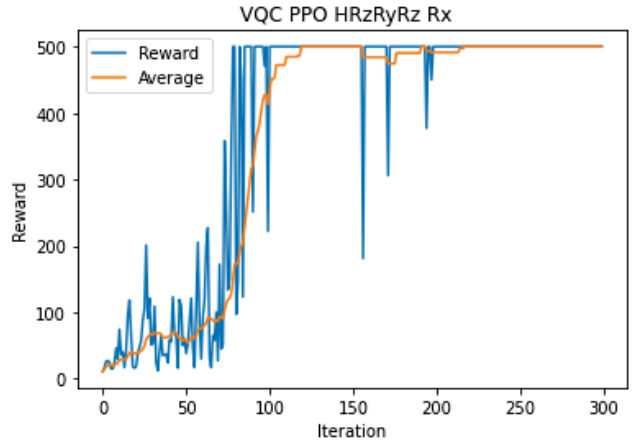

Figure 5: QVC with Rz, Ry and Rz gates

Our early results have shown that the QVC design plays an important role in achieving satisfactory performance. The quantum enhanced RL model has better performance than classical counterparts and provides major speed up in the training process. QVC proposes a platform with potential to overcome the inefficiency of classical RL in solving control tasks. Further investigation is undergoing, and a full result and analysis will be presented on the day of the conference.

ACKNOWLEDGEMENT

We would like to thank the School of Mechanical Engineering, University of Adelaide for providing the resources for this research.

REFERENCES

1. Brockman, G. et al., 2016. Openai gym. arXiv preprint arXiv:1606.01540.
2. Cerezo, M., Arrasmith, A., Babbush, R., Benjamin, S.C., Endo, S., Fujii, K., McClean, J.R., Mitarai, K., Yuan, X., Cincio, L. and Coles, P.J., 2021. Variational quantum algorithms. Nature Reviews Physics, 3(9), pp.625-644.
3. Du, S.S., Lee, J.D., Mahajan, G. and Wang, R., 2020. Agnostic Q-learning with function approximation in deterministic systems: Tight bounds on approximation error and sample complexity. arXiv preprint arXiv:2002.07125
4. Hsiao, J.Y., Du, Y., Chiang, W.Y., Hsieh, M.H. and Goan, H.S., 2022. Unentangled quantum reinforcement learning agents in the OpenAI Gym. arXiv preprint arXiv:2203.14348.
5. Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O., 2017. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.
6. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T. and Lillicrap, T., 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv preprint arXiv:1712.01815.
7. Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P. and Oh, J., 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature, 575(7782), pp.350-354.