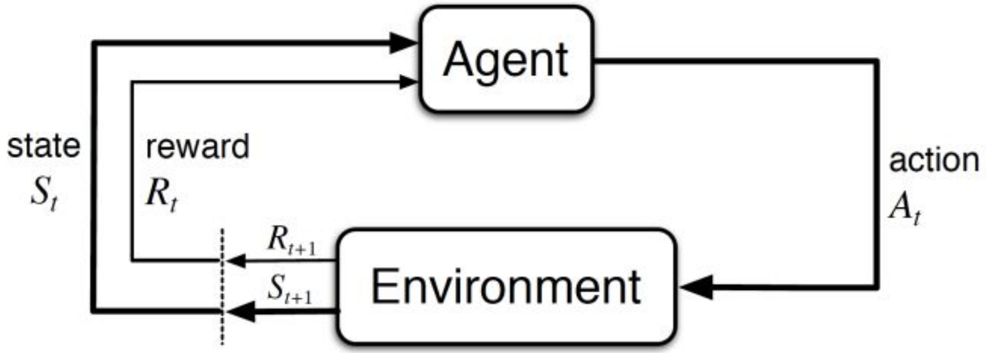
# Quantum Reinforcement Learning

Group 12

吳冠緯、吳育丞、陳宥辰

## Review on RL

- At each time step, the agent will conduct an action to the environment according to the state it receives and the environment will generate the next state and reward for the action.
- Goal: learn a policy for agent to get maximum reward in single episode.



## What we do

- Reproduce the result of the reference work, which replace the DNN model with a quantum circuits model.

- Do extra experiments with this model.

## Q-Learning

- Value-based RL

  At each time step, the agent will calculate the expected rewards for all possible actions. (restriction: action space is better to be discrete.)

$$Q_\pi(s,a) = \mathbb{E}_\pi[G_t | s_t = s, a_t = a]$$

- Greedy policy

  The agent will conduct the action with the highest expected reward.

$$\pi_*(a|s) = \underset{a}{\operatorname{argmax}}\ Q_*(s,a)$$

$\rightarrow$ Model should map the state vector to a Q vector.

## Environment - Cart Pole (1/2)

- State space

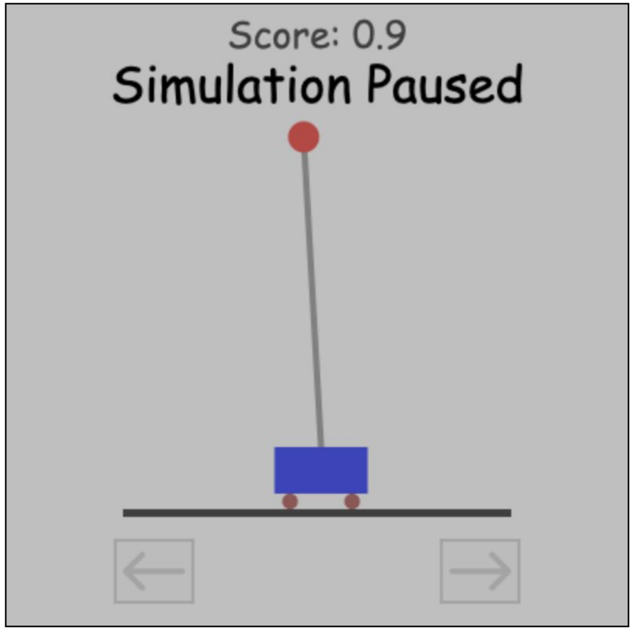
  Continuous and 4-dimensional

  (Cart position, velocity; Pole angle, angle velocity)

- Action Space

  Discrete and 2-dimensional

  (Left, Right)

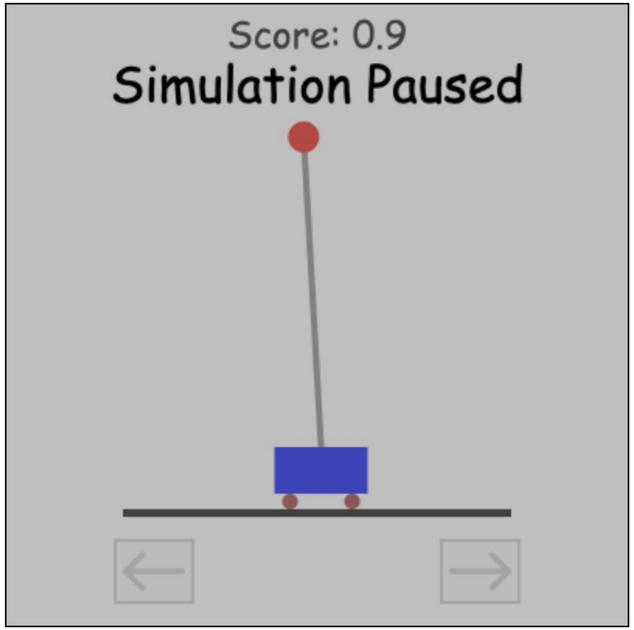


## Environment - Cart Pole (2/2)

- Reward

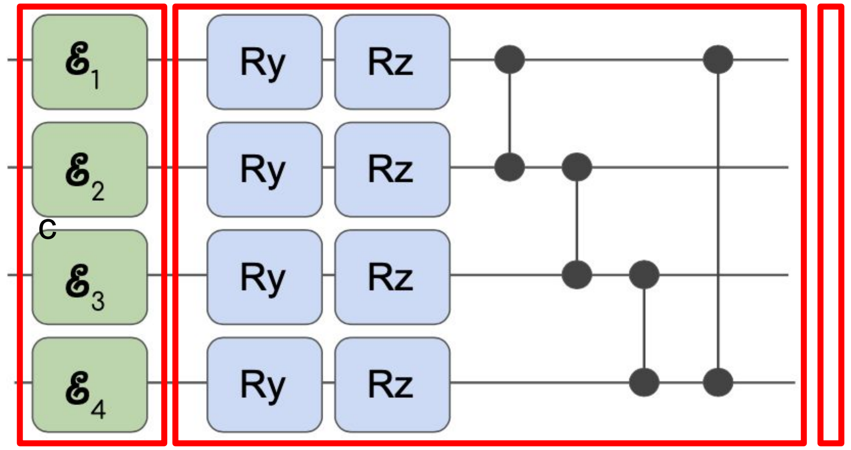
  1 if the game is not terminated at this time step.

- Termination
  - Pole is out of bound.
  - Cart is out of plate.
  - Reward = 200.



# Quantum Model vs DNN Model

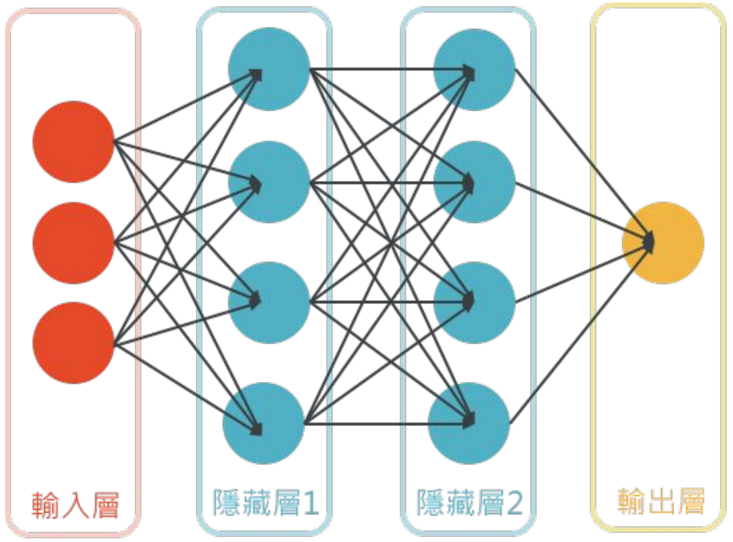
- The number of qubits is the same with the dimension of state space.



Encoder
(Input layer)

One hidden layer

Output Layer
→ Measure



## Encoder

- Variantational encoding
  - Multiply a trainable weight for each entry in state vector.
  - Apply arctan function on each entry to get an angle vector.
  - Apply a RX gate with corresponding angle to each qubit.

- Data re-uploading
  - According to the reference work, adding an encoder in front of every layer will enhance model performance.

## Output Layer

- Apply Pauli-Z gates on every qubit and measure.

  (q1, q2 for left; q3, q4 for right)

  → **Q values are bounded**

- Apply a trainable weight to each output to enhance model capability.

  → (Q+1) / 2 * w

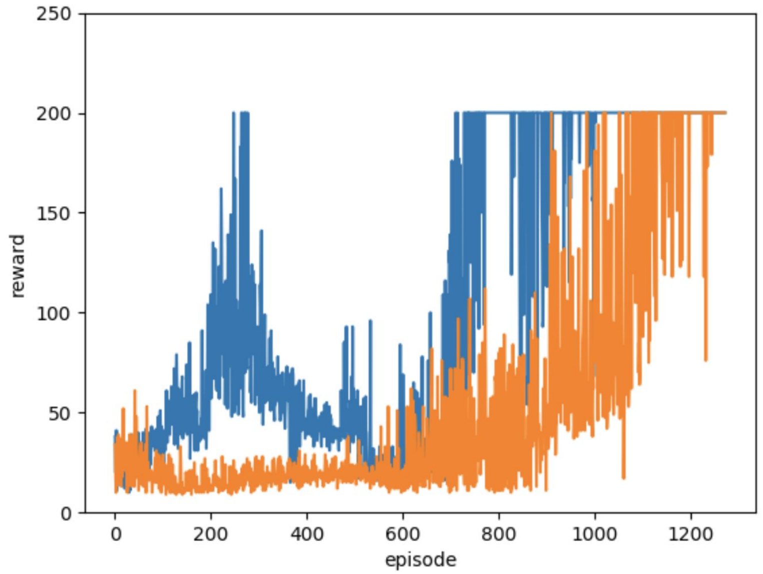
## Results - QRL vs DRL

- Training process will terminate if the model's average reward of the last 100 episodes >= 195.
- Settings of Q-learning are the same.
- Loss function and optimizer are the same.
- Number of layers = 5.

| | Number of parameters | Total episodes |
|---|---|---|
| **DRL** | 1172 | 1666 |
| **QRL** | **62** | **1038** |

# Whether the order of features matters?

- Blue: cart position, **cart velocity, pole angle,** pole angle velocity.
- Orange: cart position, **pole angle, cart velocity**, pole angle velocity.



Number of episodes
**Blue: 1038**
**Orange: 1272**

→ The order may affect the convergence speed.

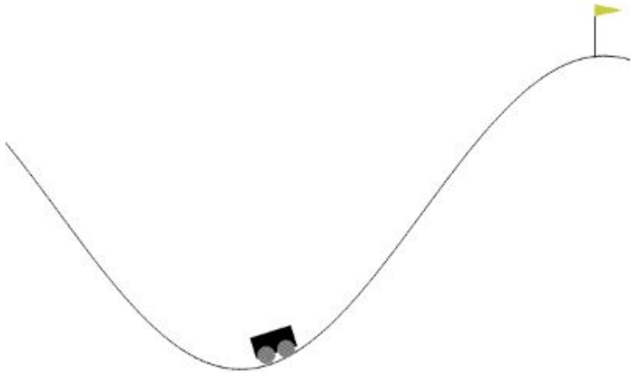
## To Do

Try to train on another task called “Mountain Car”.

(1)State space: 2-dimensional, Action space: 3-dimensional.

(2)Only get reward when the car get to the top of the mountain

→ More difficult to train.



## Reference

[1] Quantum agents in the Gym:

a variational quantum algorithm for deep Q-learning

Andrea Skolik, Sofiene Jerbi, and Vedran Dunjko