Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy”

-Richard Feynman

Quantum Reinforcement Learning (QRL)

Anil Katwal

PHD candidates: Computational Science(Physics)

Research interest : AI/Quantum AI

Computer science and Engineering Department

USM

Agenda

- Introduction
- Reinforcement Learning (RL)
- Quantum Reinforcement Learning
- TensorFlow Quantum platform (Cirq)
- Conclusion and outlook

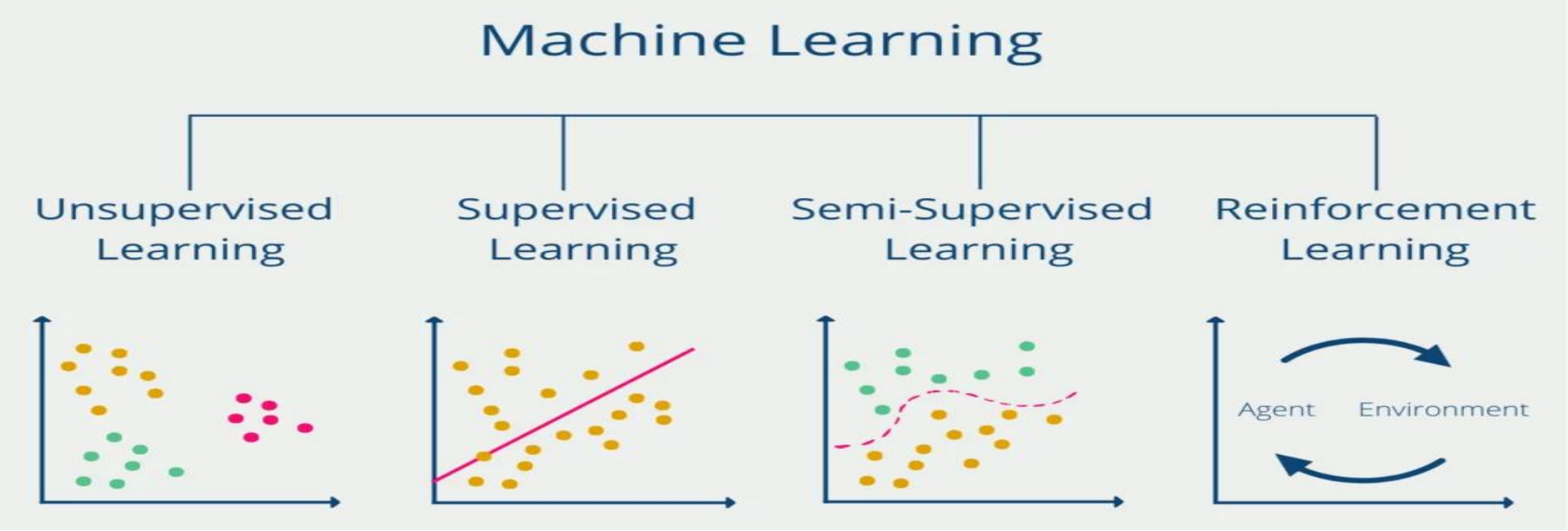
Introduction:

- Reinforcement Learning(RL): Alpha Go success since 2016; RL can solve complex Sequential decision-making problems.
- Quantum Computing (QC) : Hardware advancement since 2017
- Notable QC companies such as IBM, Google, D-wave ….
- QC can solve certain computational problem with performance superior to classical computers

Reinforcement Learning

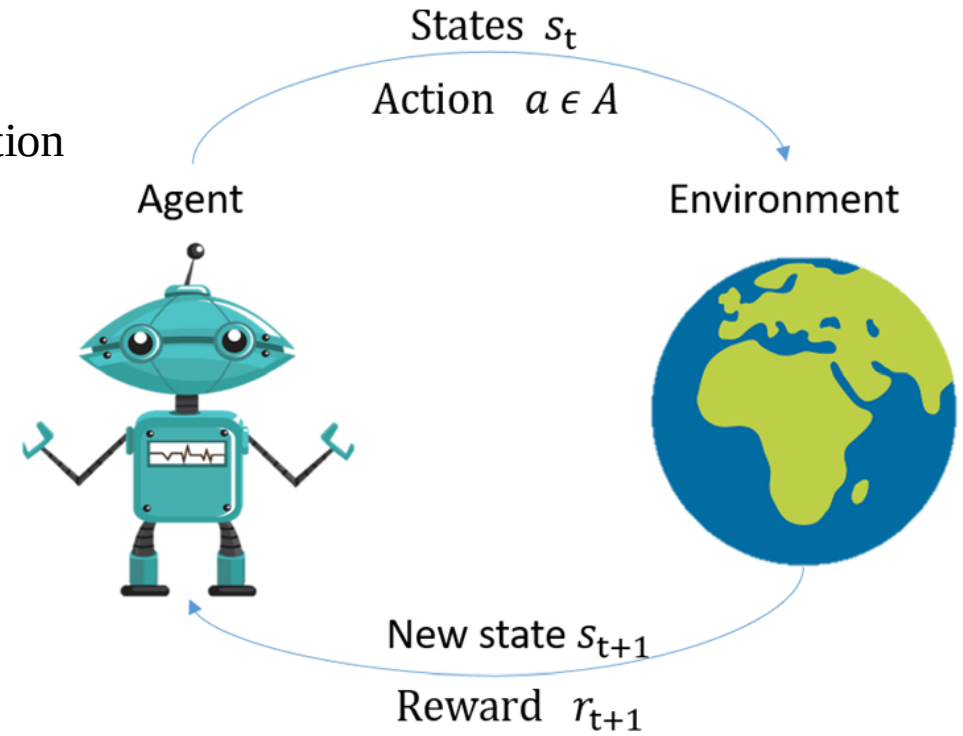
Reinforcement Learning(RL):

Reinforcement Learning (RL) is a type of machine learning technique that enables an agent to learn in an interactive environment by trial and error using feedback from its own actions and experiences.



Reinforcement Learning (RL):

- RL: An agent interact with an environment (E) over several discrete and continuous time steps.
- The agent receives state or Observation S_t and then chooses an action a_t from a set of action A according to policy



Q-value is updated:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

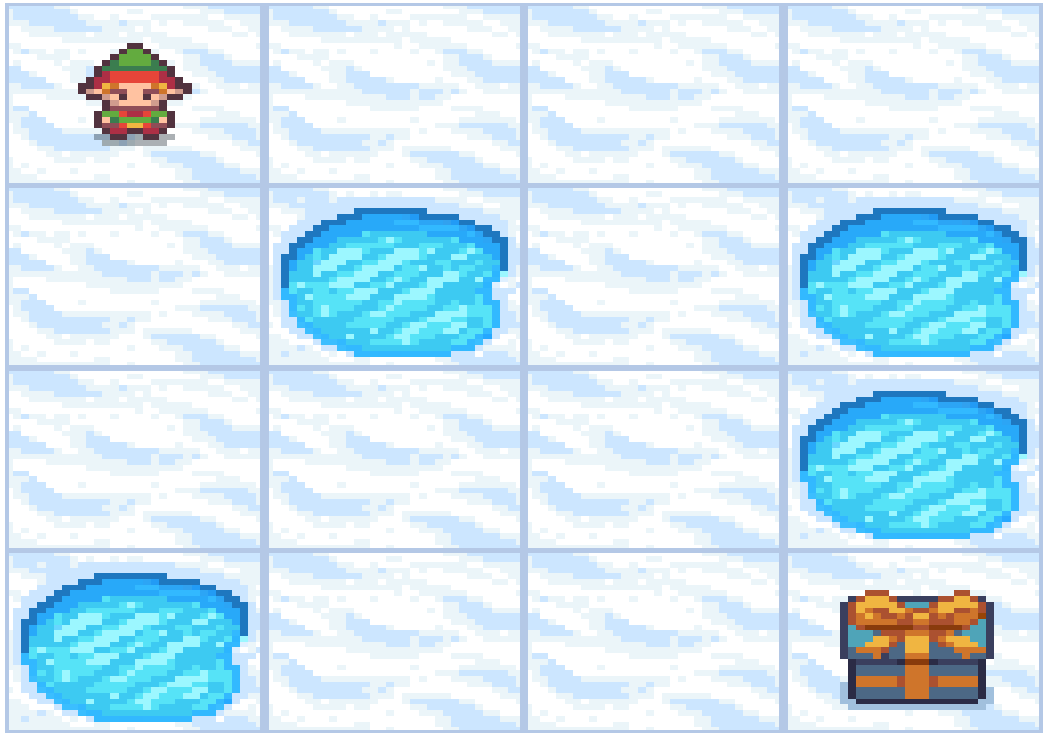
Old Q Value · Reward · Discount Rate $(0 \sim 1)$

New Q Value · Learning Rate $(0 \sim 1)$ · Maximum Q value of transition destination state

TD error

Bellman equation

Q-table for Frozen Lake :-



State Space

Numbers assigned to each state

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Actions

States

	A_1	A_2	...	A_M
S_1	$Q(S_1, A_1)$	$Q(S_1, A_2)$		$Q(S_1, A_M)$
S_2	$Q(S_2, A_1)$	$Q(S_2, A_2)$		$Q(S_2, A_M)$
⋮			⋱	⋮
S_N	$Q(S_N, A_1)$	$Q(S_N, A_2)$	...	$Q(S_N, A_M)$

Action Space

State Space

Q-table	Left (0)	Down (1)	Right (2)	Up (3)
0	0.73509189	0.77378094	0.77378094	0.73509189
1	0.73509189	0	0.81450625	0.77378094
2	0.77378094	0.857375	0.77378094	0.81450625
3	0.81450625	0	0.77378094	0.77378094
4	0.77378094	0.81450625	0	0.73509189
5	0	0	0	0
6	0	0.9025	0	0.81450625
7	0	0	0	0
8	0.81450625	0	0.857375	0.77378094
9	0.81450625	0.9025	0.9025	0
10	0.857375	0.95	0	0.857375
11	0	0	0	0
12	0	0	0	0
13	0	0.9025	0.95	0.857375
14	0.9025	0.95	1	0.9025
15	0	0	0	0

Quantum Machine Learning:

Quantum Machine Learning (QML) *is an interdisciplinary field combining quantum computing and machine learning. It aims to leverage the unique properties of quantum systems, such as* ***superposition***, ***entanglement***, *and* ***quantum interference***, *to improve computational efficiency and enable novel algorithms for processing data.*

CC (Classical Algorithm, Classical Data)

Description: Classical algorithms processing classical data.
Example: Traditional machine learning algorithms like linear regression, decision trees, etc.

CQ (Classical Algorithm, Quantum Data)

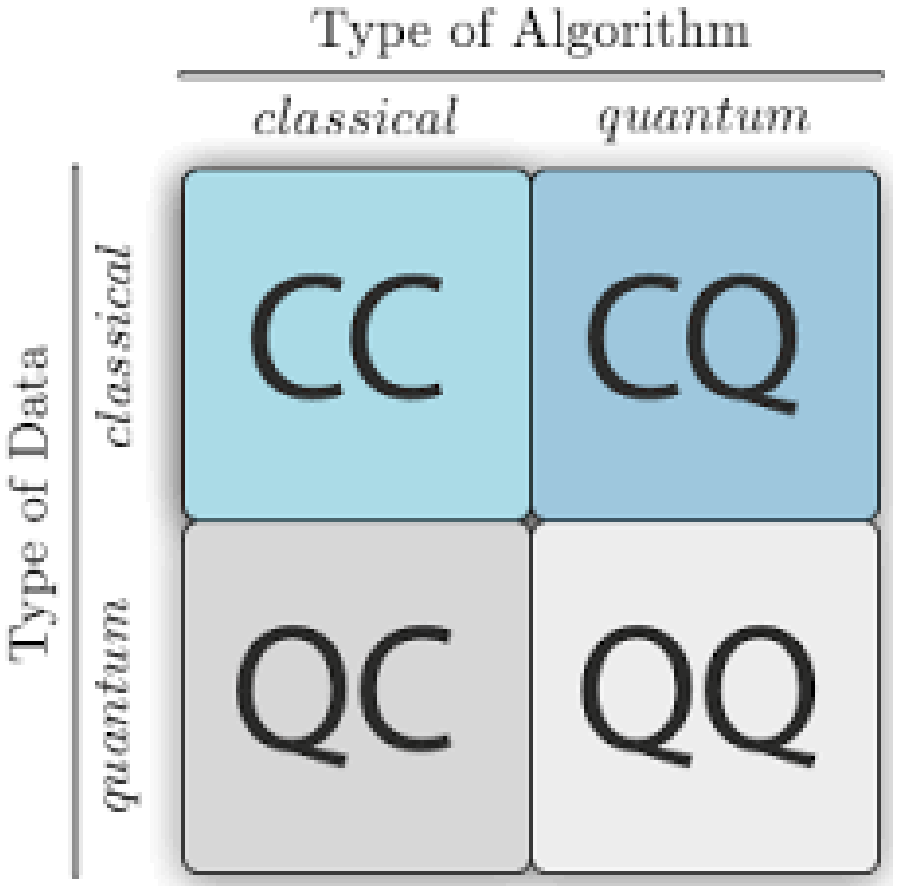
Description: Classical algorithms processing quantum data.
Example: Using classical analysis to extract insights from quantum states or quantum-generated data.

QC (Quantum Algorithm, Classical Data)

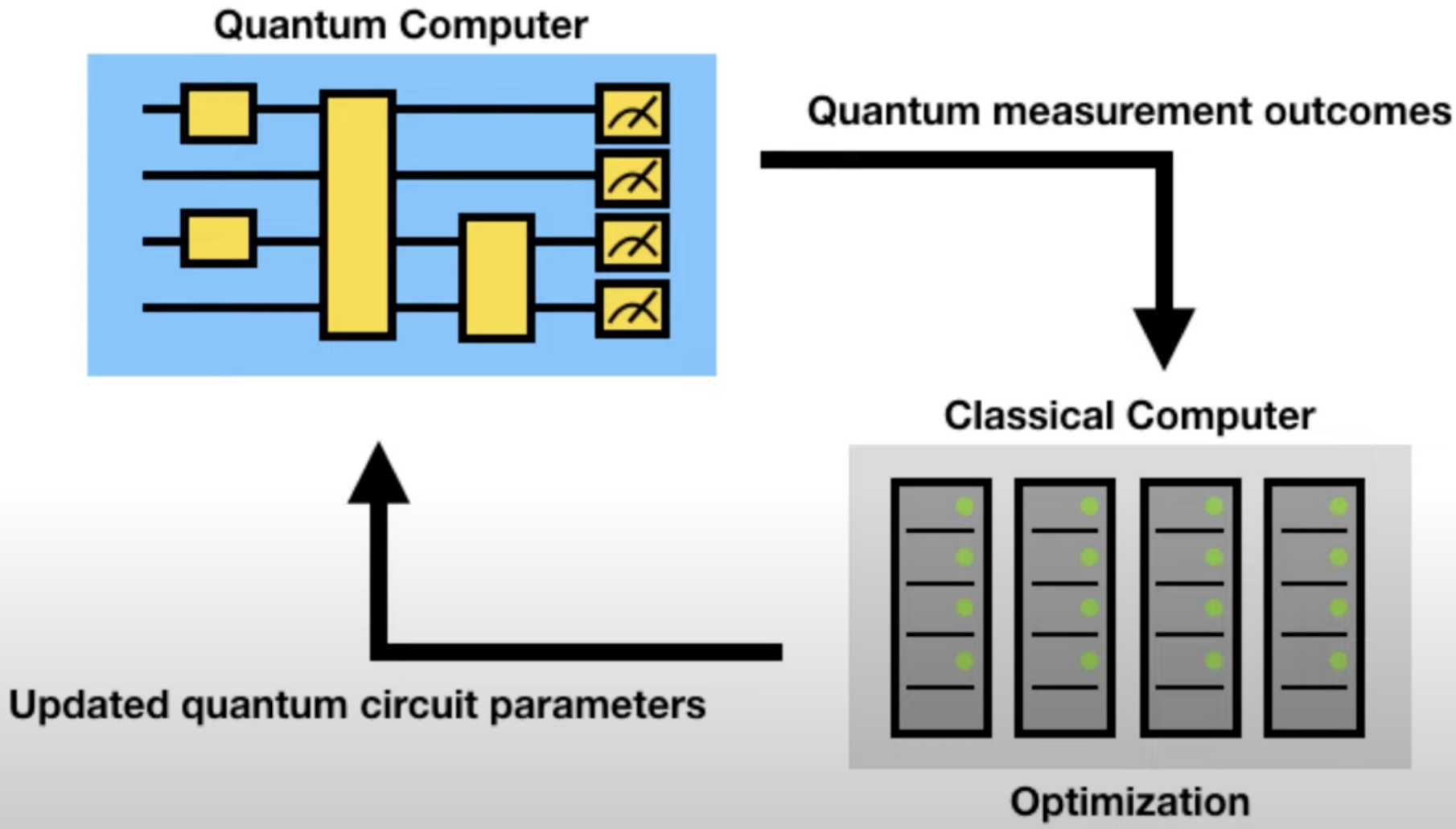
Description: Quantum algorithms processing classical data.
Example: Algorithms like Grover's search or quantum Fourier transform

QQ (Quantum Algorithm, Quantum Data)

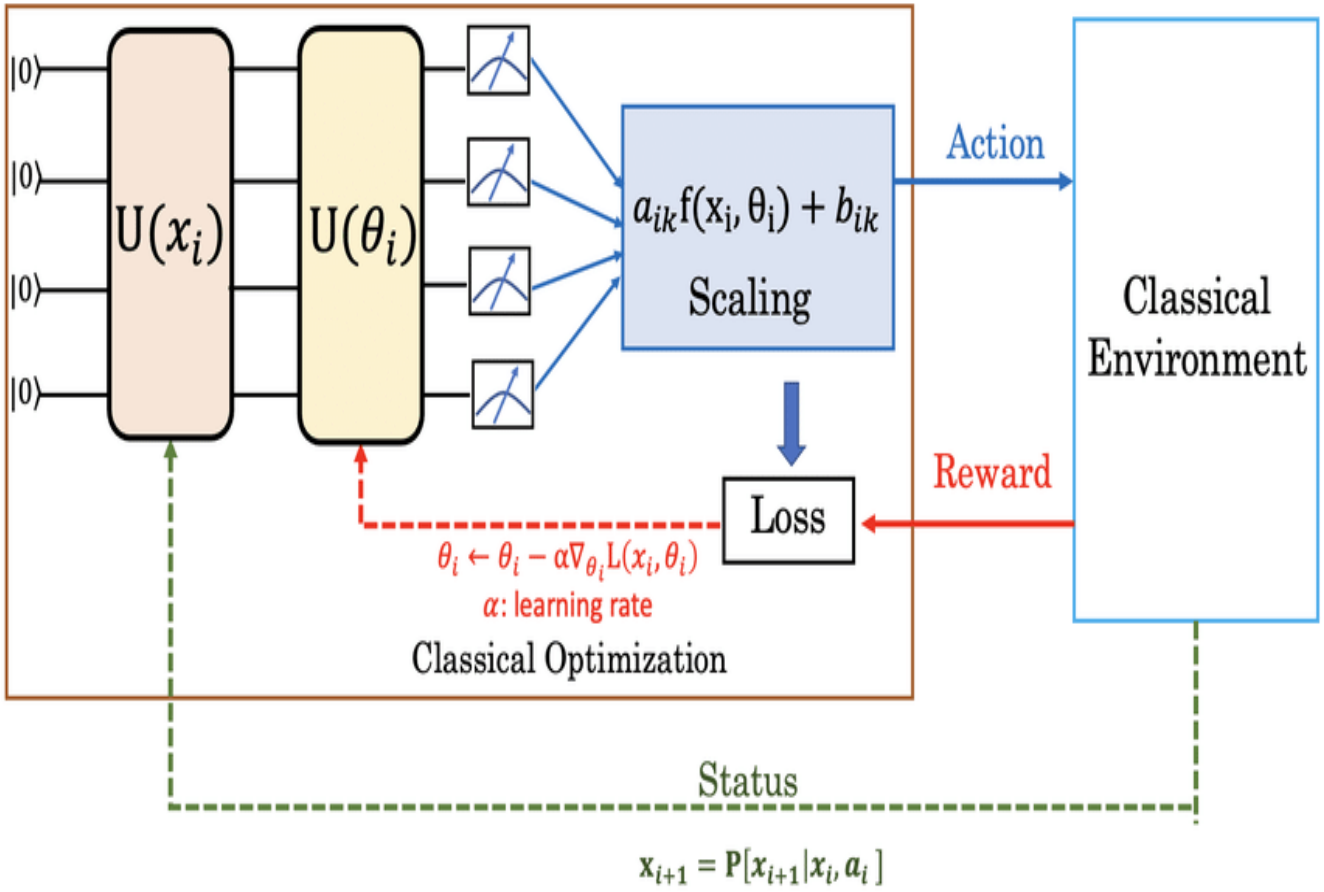
Description: Quantum algorithms processing quantum data.
Example: Quantum error correction, quantum simulations, or variational quantum eigen solvers (VQE) that process quantum states.



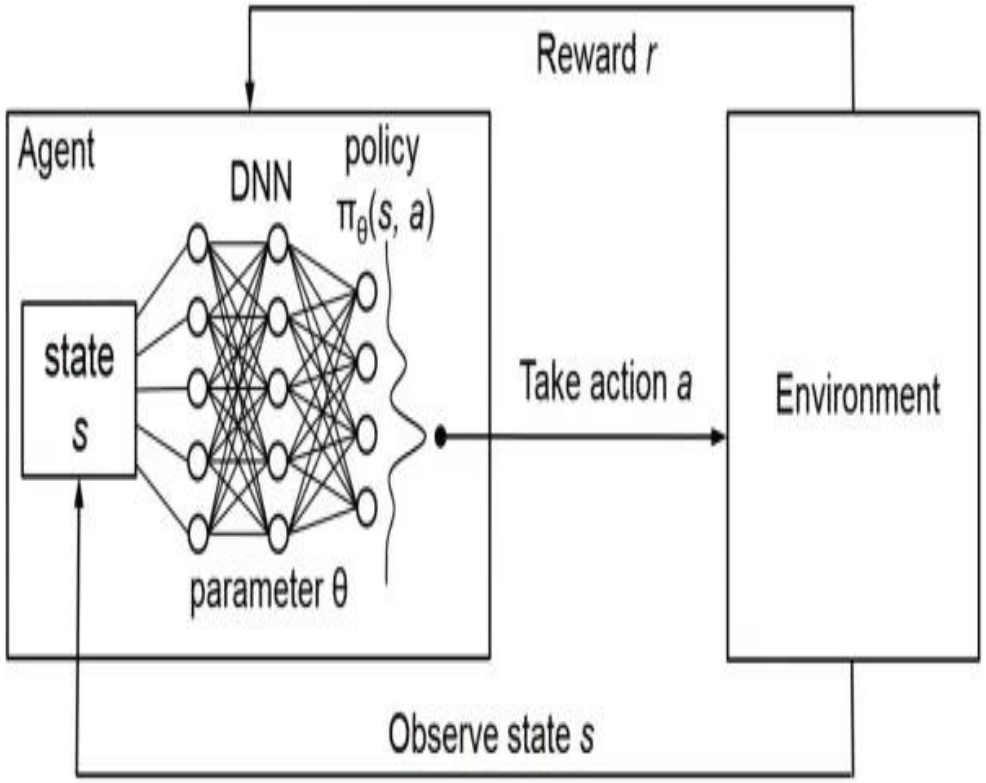
Hybrid Quantum-Classical Paradigm



Hybrid Quantum-Classical Paradigm continued:



variational-based quantum reinforcement learning (VQRL):



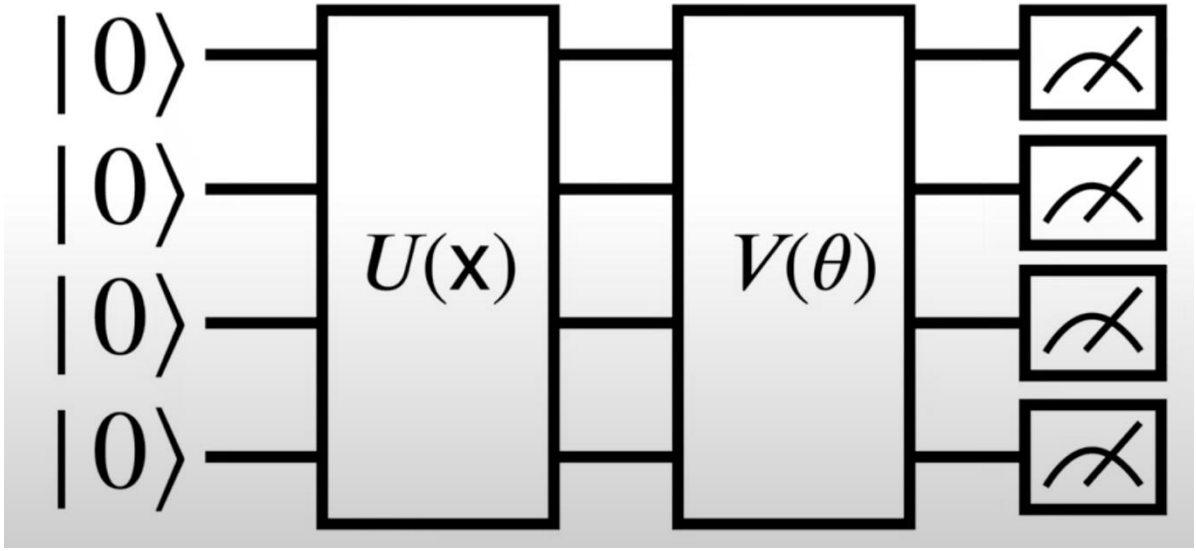
Deep Q-Networks

Variational Quantum circuit (VQC):

Parameterized Quantum Circuit:

•A quantum circuit composed of gates whose parameters (e.g rotation angles) can be tuned.

•These gates define the quantum state evolution.



Classical Optimization Loop:

•The parameters are iteratively adjusted using classical optimization algorithms such as gradient descent or genetic algorithms.

•The optimization is based on minimizing a cost function derived from measurements of the quantum system.

Hybrid Nature:

•***Quantum Part:*** Handles state preparation, entanglement, and measurements.

•***Classical Part:*** Processes the measurement results and updates the circuit parameters.

Quantum Encoding and State Preparation:

A general N qubit quantum state can be represented as:

$$|\psi\rangle = \sum_{(q_1,q_2,\cdots,q_N)\in\{0,1\}} c_{q_1,q_2,\cdots,q_N} \left|q_1\right\rangle \otimes \left|q_2\right\rangle \otimes \cdots \otimes \left|q_N\right\rangle$$

where $c_{q_1,\cdots,q_N} \in \mathbb{C}$ is the complex amplitude for each basis state and each $q_i \in \{0,1\}$

The total probability is equal to 1: $\sum_{(q_1,\cdots,q_N)\in\{0,1\}} \| c_{q_1,\cdots,q_N} \|^2 = 1$

Quantum Encoding and State Preparation, Continued

Amplitude Encoding

Encode a vector $\left(\alpha_0, \cdots, \alpha_{2^n-1}\right)$ into a n-qubit quantum state:

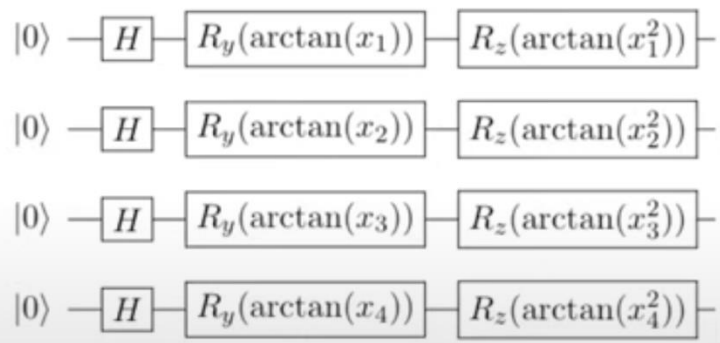
$$|\Psi\rangle = \alpha_0 |00\cdots0\rangle + \cdots + \alpha_{2^n-1} |11\cdots1\rangle$$

where α_i are real numbers and $\left(\alpha_0, \cdots, \alpha_{2^n-1}\right)$ is normalized

N-dimensional vector will require only $\log_2(N)$ qubits to encode

Variational Encoding

Input numbers $x_1 \cdots x_n$ are used as quantum rotation angles



Simpler implementation than amplitude encoding

Quantum Encoding and State Preparation, Continued:

Example: Amplitude Encoding a Vector

Input:

Classical vector $\mathbf{x} = [1, 2, 3, 4]$.

Steps:

1. **Normalize the Data:**

$$\|\mathbf{x}\| = \sqrt{1^2 + 2^2 + 3^2 + 4^2} = \sqrt{30}$$

Normalized $\mathbf{x} = \left[\frac{1}{\sqrt{30}}, \frac{2}{\sqrt{30}}, \frac{3}{\sqrt{30}}, \frac{4}{\sqrt{30}}\right]$.

2. **Map to a Quantum State:**

$$|\psi\rangle = \frac{1}{\sqrt{30}}|00\rangle + \frac{2}{\sqrt{30}}|01\rangle + \frac{3}{\sqrt{30}}|10\rangle + \frac{4}{\sqrt{30}}|11\rangle$$

3. **Circuit Construction:**

   - Use controlled rotations to prepare the amplitudes $\frac{1}{\sqrt{30}}, \frac{2}{\sqrt{30}}, \ldots$.

Challenges of Amplitude Encoding:

1. State Preparation Complexity:

- Preparing a general amplitude-encode state requires ***O(N)*** gates.

2. **Hardware Constraints:**

- Noisy intermediate-scale quantum(NISQ) devices have limited qubits and shallow depth, making large-scale encoding challenging.

3. Normalization:

- Data must be preprocessed to satisfy normalization, which can computationally be expensive for large for datasets.

Application: Quantum Support Vector Machine(QSVM), Quantum Simulations, and Quantum Search(Grover's Algorithm)

Quantum Encoding and State Preparation, Continued:

Angle encoding:

In angle encoding, the features of a classical data vector are mapped to the rotation angles of single-qubit rotation gates (R_x, R_y, or R_z). These gates rotate the state of the qubits around the corresponding axis of the Bloch sphere.

Mathematical Representation

Given a classical data vector $\mathbf{x} = [x_0, x_1, \ldots, x_{n-1}]$, the state preparation using angle encoding applies rotations to n qubits as follows:

$$|0\rangle \xrightarrow{R_y(2\pi x_i)} |\psi\rangle = \cos(\pi x_i)|0\rangle + \sin(\pi x_i)|1\rangle$$

where x_i is a feature of the classical data normalized to $[0, 1]$, and $R_y(\theta)$ is the rotation gate around the y-axis:

$$R_y(\theta) = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix}$$

X-axis, denoted as $R_x(\theta)$, is represented as:

$$R_x(\theta) = e^{-i\theta X/2} = \begin{bmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{bmatrix}$$

Z-axis denoted as $R_z(\theta)$

$$R_z(\theta) = e^{-i\theta Z/2} = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}$$

Example: Angle Encoding

Input:

Classical data vector $\mathbf{x} = [0.2, 0.4, 0.6]$.

Steps:

1. **Normalize the Data**: Assume $\mathbf{x}$ is already normalized to $[0, 1]$.
2. **Circuit Construction**:
   - Initialize 3 qubits in $|0\rangle$.
   - Apply the following rotations:
     - $R_y(2\pi \cdot 0.2)$ to the first qubit.
     - $R_y(2\pi \cdot 0.4)$ to the second qubit.
     - $R_y(2\pi \cdot 0.6)$ to the third qubit.
3. **Resulting State**: The quantum state becomes:

$$|\psi\rangle = \cos(0.4\pi)|000\rangle + \sin(0.4\pi)|001\rangle + \ldots$$

Challenges:

- *Requires one qubit for features, which can be limiting for high dimensional.*
- *Each qubit can only encode a single feature, so it may require additional layers of the circuit (e.g., data re-uploading) to process complex data.*
- *Accurate encoding depends on proper normalization of the input data.*

Quantum Encoding and State Preparation, Continued:

Basis Encoding :

This is the most straightforward form of quantum encoding. In this method, classical bits are directly mapped to quantum bits (qubits) using the computational basis states $|0\rangle$ and $|1\rangle$. The information is stored in the amplitudes of these states. A classical number 101 can be represented as the quantum state $|101\rangle$:

$$|101\rangle = |1\rangle \otimes |0\rangle \otimes |1\rangle.$$

In terms of amplitudes:

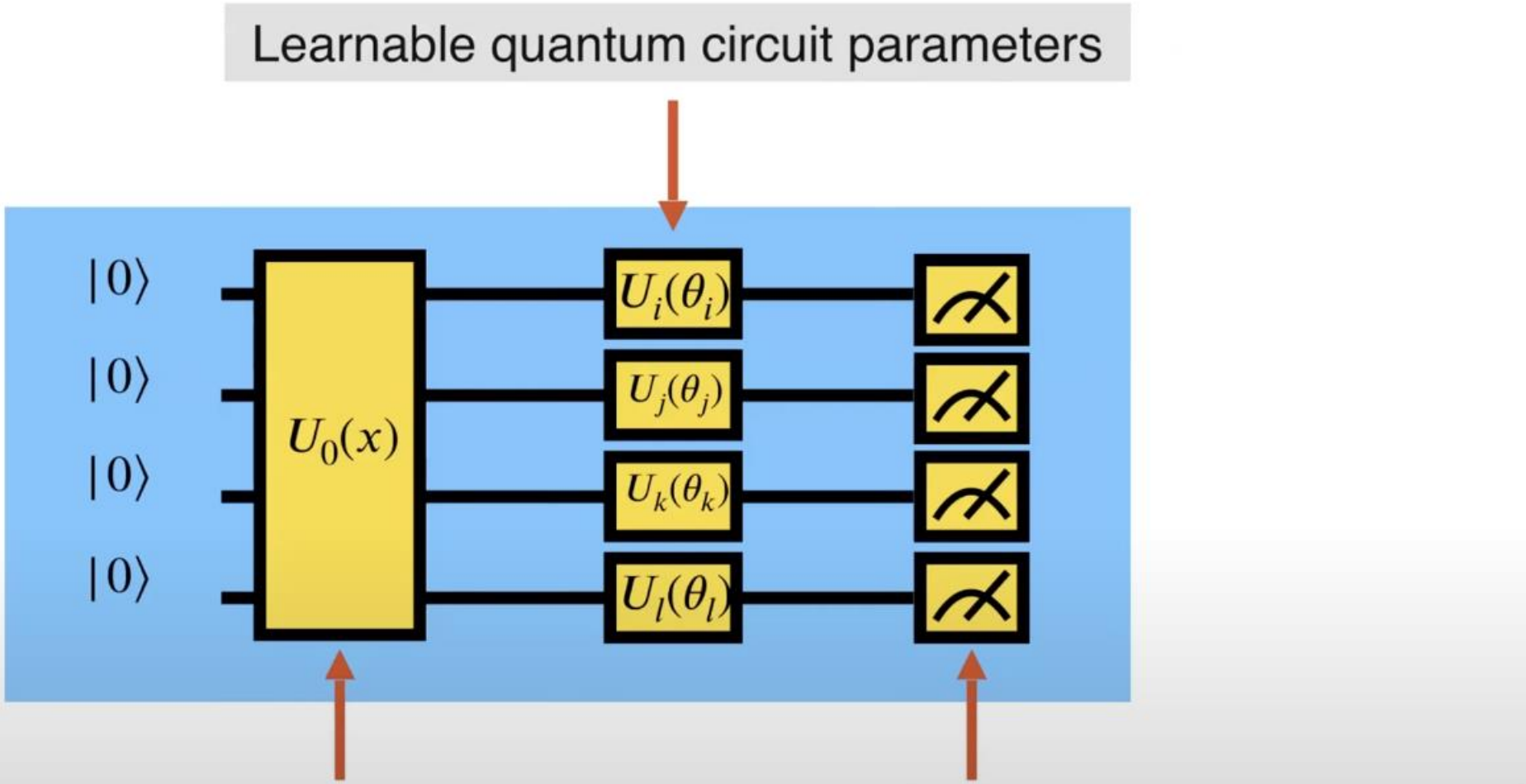
$$|101\rangle = \alpha|1\rangle \otimes \beta|0\rangle \otimes \gamma|1\rangle,$$

where α, β, γ are the probability amplitudes associated with each qubit state. The tensor product symbol $\otimes$ indicates that each qubit is considered independently.
The basis encoding of the word "hello" is given by the quantum states: $|h\rangle = |1101000\rangle$, $|e\rangle = |1100101\rangle$, $|l_1\rangle = |1101100\rangle$, $|l_2\rangle = |1101100\rangle$, $|o\rangle = |1101111\rangle$ To perform basis encoding of the word "hello," each ASCII character is converted into its binary representation and then encode each bit using quantum states.ASCII for 'h': 104 (binary: 1101000), ASCII for 'e': 101 (binary: 1100101), ASCII for 'l': 108 (binary: 1101100), ASCII for 'l': 108 (binary: 1101100), ASCII for 'o': 111 (binary: 1101111). This encoding scheme is suitable for discrete data represented as binary or integer values.

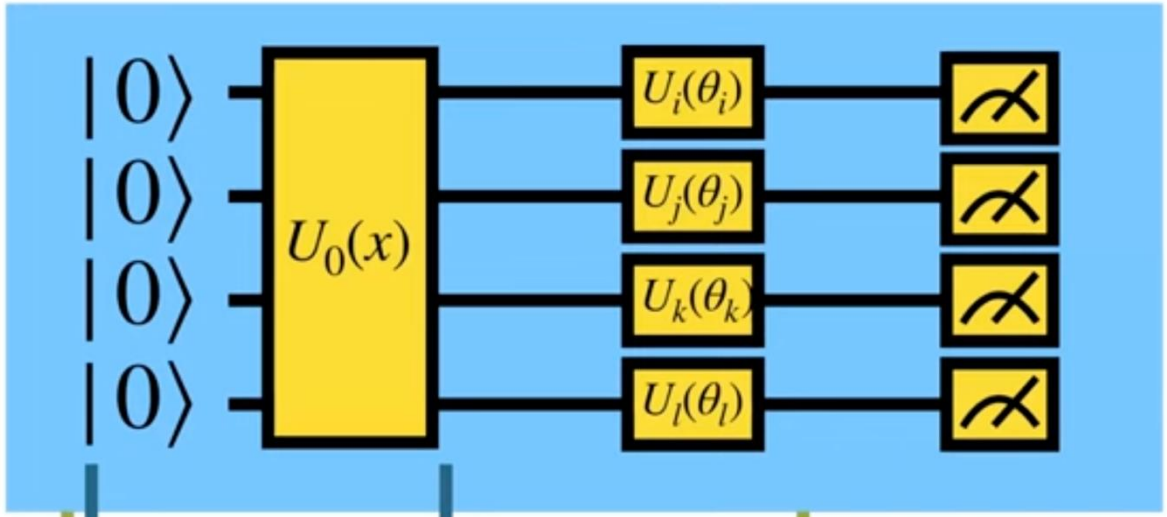
$$|hello\rangle = \frac{1}{\sqrt{5}}|1101000\rangle + \frac{1}{\sqrt{5}}|0010011\rangle + \frac{1}{\sqrt{5}}|1011011\rangle + \frac{1}{\sqrt{5}}|1100110\rangle + \frac{1}{\sqrt{5}}|1101111\rangle$$

Encoding Method	Description	Usage	Example
Amplitude Encoding	Classical data is encoded into the amplitudes of a quantum state. It represents data in a superposition of basis states.	Efficient for storing large amounts of classical data in quantum systems. Complex to prepare the quantum state.	A classical bitstring "101" might correspond to a quantum state with appropriate amplitudes.
Basis Encoding	Classical data is directly mapped to the computational basis states of the quantum system.	Simple and direct. Each classical value corresponds to a distinct quantum state.	A 3-bit string "110" directly corresponds to the quantum state (
Angle Encoding	Classical data is used to parameterize the angles of quantum gates, typically rotation gates, which affect the quantum state.	Often used in quantum machine learning. Classical data is mapped to the angles for rotation gates.	A classical value of 0.5 might encode a rotation gate $R_z(0.5)$, changing the quantum state.

Quantum Gradients



Quantum Gradients, Continued



$$U_0(x)\,|0\rangle$$

$$U_i\left(\theta_i\right)\,U_0(x)\,|0\rangle$$

$$\left\langle 0 \left| U_0^{\dagger}(x) U_i^{\dagger}\left(\theta_i\right) \hat{B} U_i\left(\theta_i\right)\, U_0(x) \right| 0 \right\rangle$$

Quantum Gradients, Continued

$$f\left(x;\theta_i\right) = \left\langle 0 \left| U_0^{\dagger}(x) U_i^{\dagger}\left(\theta_i\right) \hat{B} U_i\left(\theta_i\right) U_0(x) \right| 0 \right\rangle = \left\langle x \left| U_i^{\dagger}\left(\theta_i\right) \hat{B} U_i\left(\theta_i\right) \right| x \right\rangle$$

x: input value

$U_0(x)$: encoding circuit

i: circuit parameter index

$U_i(x_i)$: single-qubit rotation generated by the Pauli operators

Mitarai, K., Negoro, M., Kitagawa, M., & Fujii, K. (2018). Quantum circuit learning. *Physical Review A*, *98*(3), 032309.
Schuld, M., Bergholm, V., Gogolin, C., Izaac, J., & Killoran, N. (2019). Evaluating analytic gradients on quantum hardware. *Physical Review A*, *99*(3), 032331.

Quantum Gradients, Continued

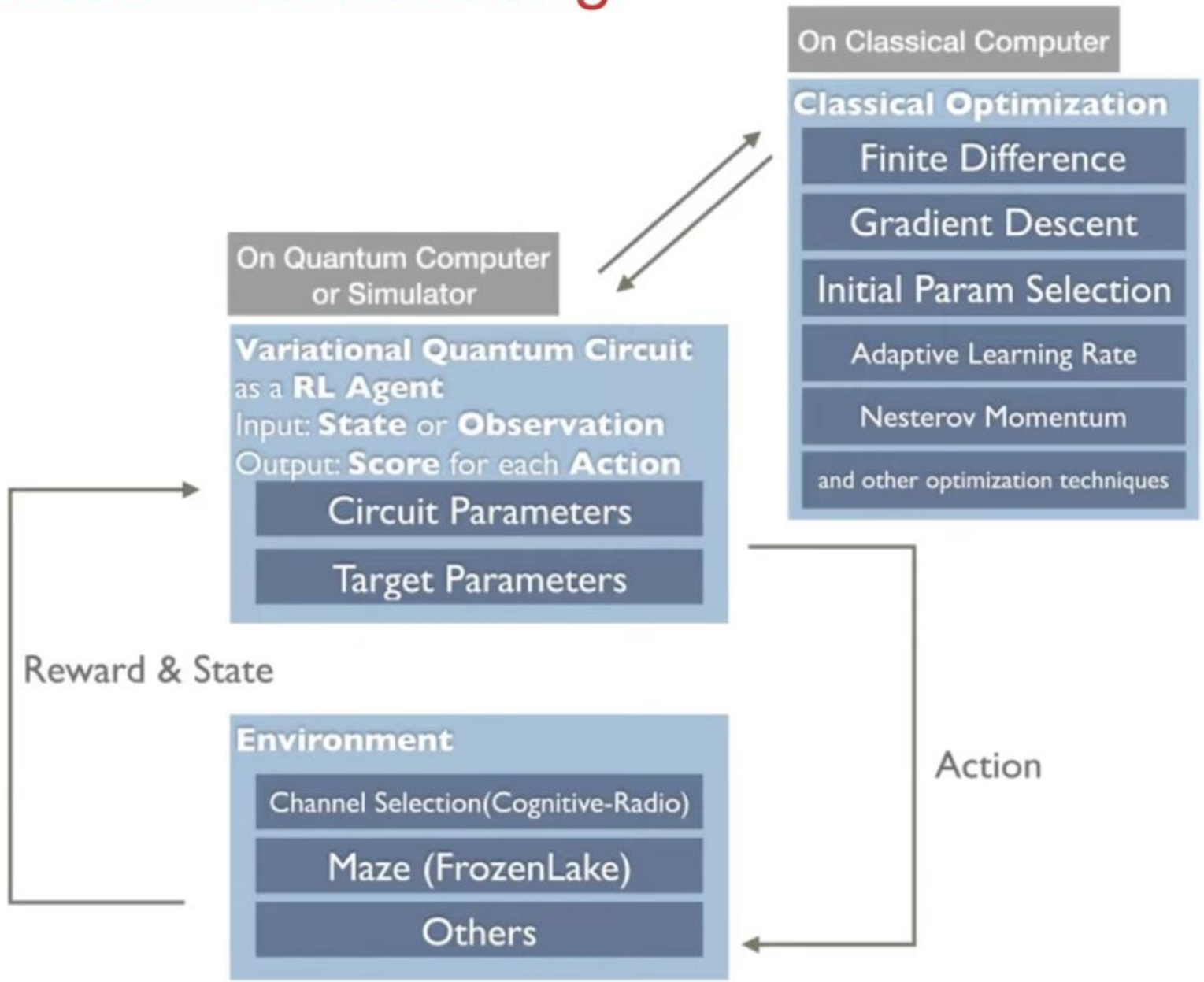
The gradient of f with respect to the parameter θ_i is:

$$\nabla_{\theta_i} f(x;\theta_i) = \frac{1}{2}\left[f\left(x;\theta_i + \frac{\pi}{2}\right) - f\left(x;\theta_i - \frac{\pi}{2}\right)\right]$$

This value can be calculated via running two quantum circuits with shifted parameters, the so-called *parameter-shift* rule.

Mitarai, K., Negoro, M., Kitagawa, M., & Fujii, K. (2018). Quantum circuit learning. *Physical Review A*, *98*(3), 032309.
Schuld, M., Bergholm, V., Gogolin, C., Izaac, J., & Killoran, N. (2019). Evaluating analytic gradients on quantum hardware. *Physical Review A*, *99*(3), 032331.

Quantum Reinforcement Learning



Quantum Deep Q-Learning

Algorithm 1 Variational Quantum Deep Q Learning

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function quantum circuit Q with random parameters
for episode $= 1, 2, \ldots, M$ **do**
  Initialise state s_1 and encode into the quantum state
  for $t = 1, 2, \ldots, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(s_t, a; \theta)$ from the output of the quantum circuit
    Execute action a_t in emulator and observe reward r_t and next state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
    Sample random minibatch of transitions (s_j, a_j, r_j, s_{j+1}) from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } s_{j+1} \\ r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta) & \text{for non-terminal } s_{j+1} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q(s_j, a_j; \theta)\right)^2$
  end for
end for

Quantum Deep Q-Learning, Continued

- Env: FrozenLake
- 16 discrete states
- Four actions
  - Up
  - Down
  - Right
  - Left

(a)

(b)

(c)

Location	Reward
HOLE	-0.2
GOAL	1.0
OTHER	-0.01

Quantum Deep Q-Learning, Continued

- Environment with 16 states
- States numbered as 0-15
- Example:

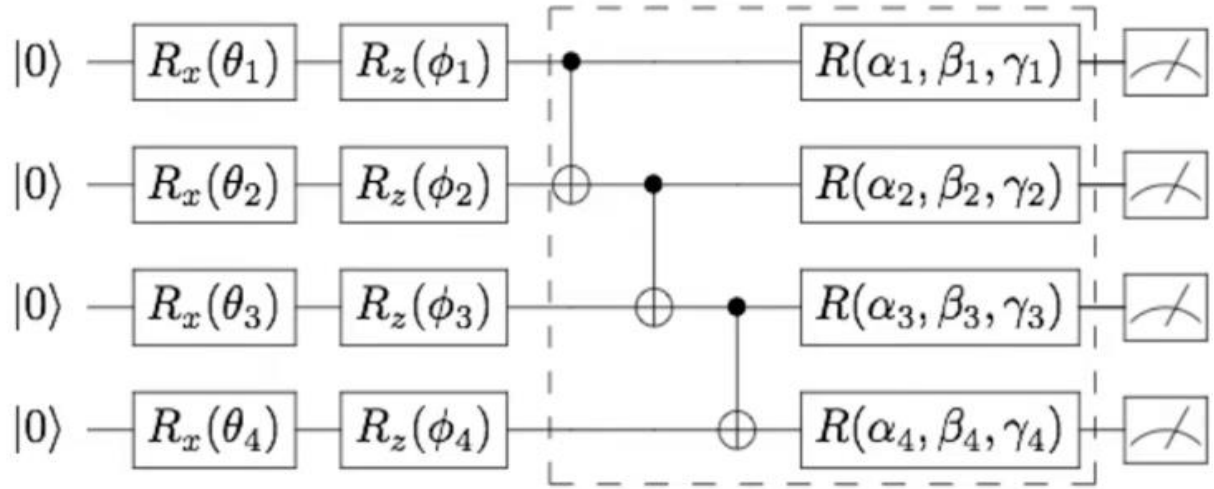
State 12: 1100 -> 1,0,0,0

Rotation:

$$\theta_i = \pi \times b_i$$

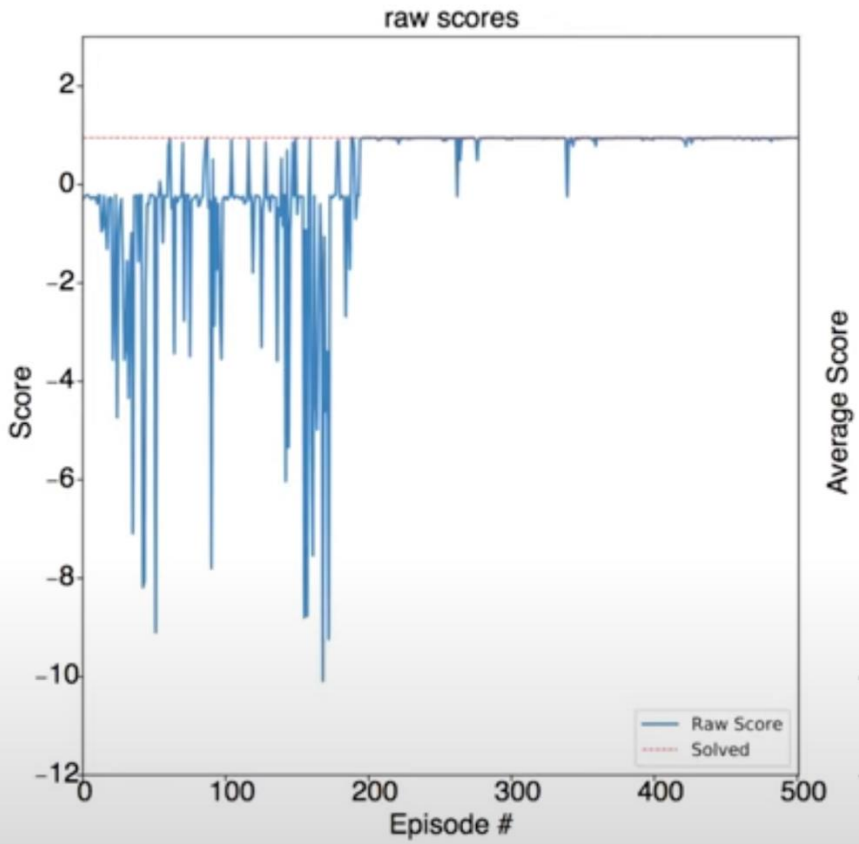
$$\phi_i = \pi \times b_i$$

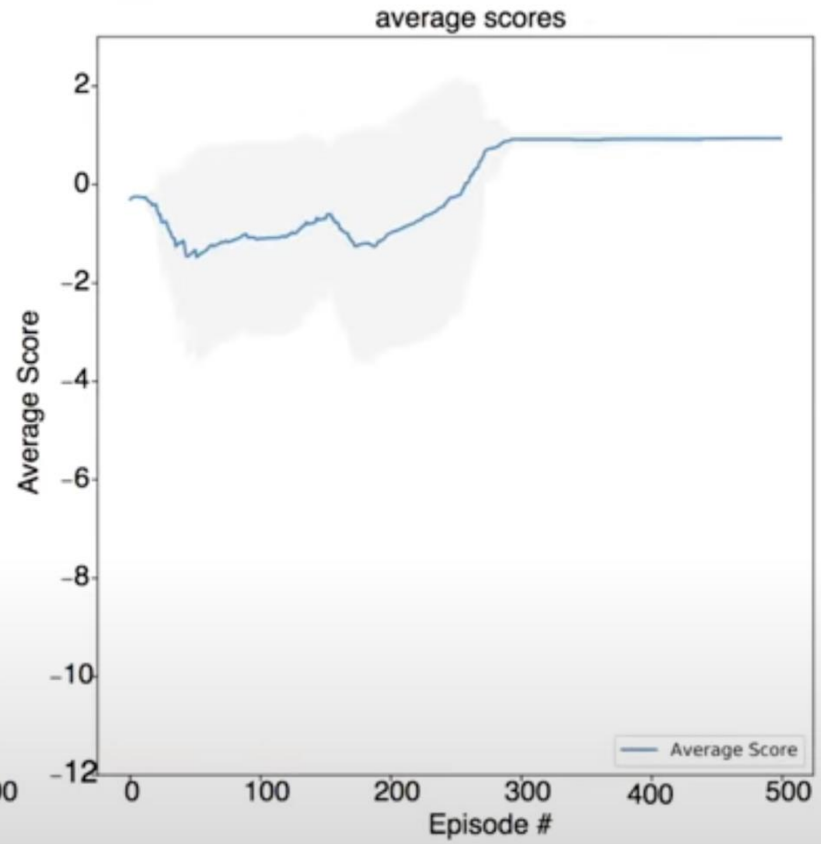
Result: $|1\rangle \otimes |1\rangle \otimes |0\rangle \otimes |0\rangle$



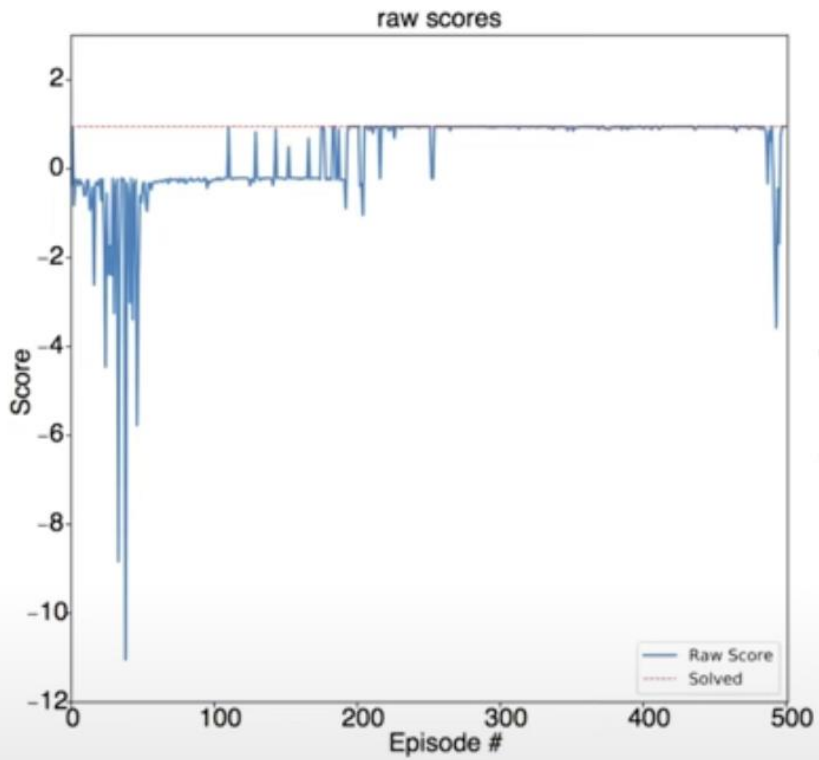
Chen, S. Y. C., Yang, C. H. H., Qi, J., Chen, P. Y., Ma, X., & Goan, H. S. (2020). Variational quantum circuits for deep reinforcement learning. *IEEE Access*, *8*, 141007-141024.

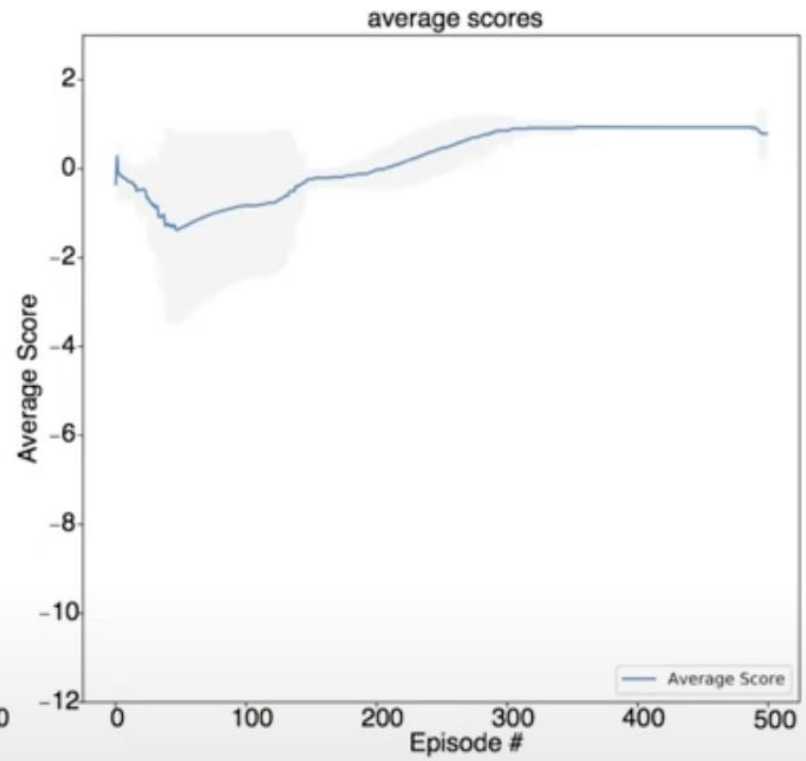
Quantum Deep Q-Learning, Continued





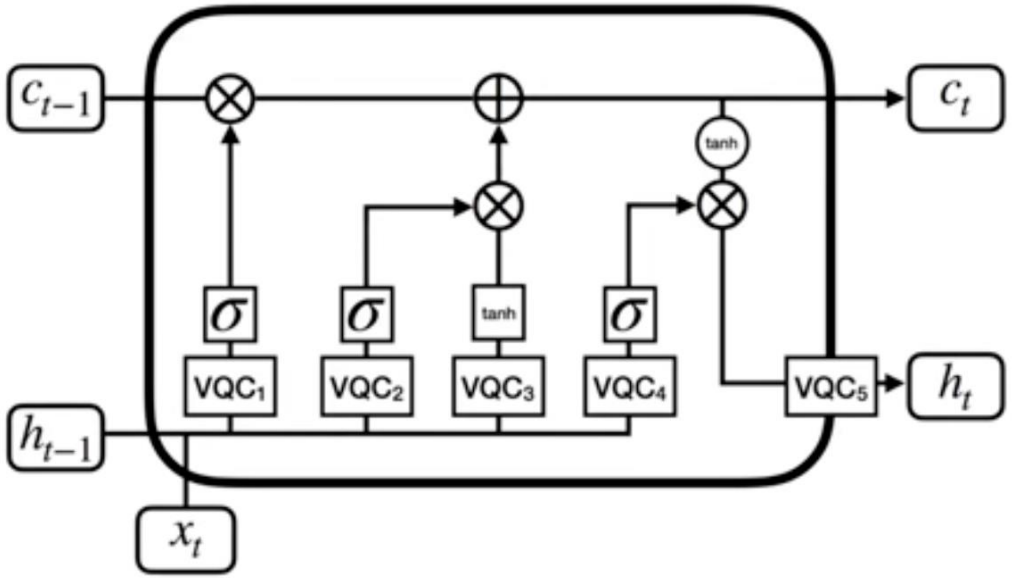
Quantum Deep Q-Learning, Continued

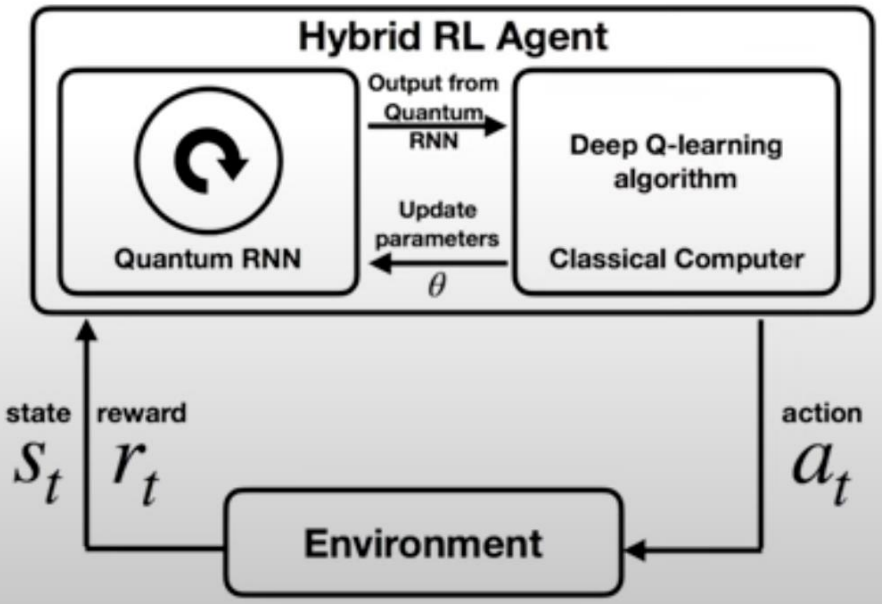




Quantum Recurrent RL

- **Motivation:** Many real-world environments are only **partially observable.** The AI can only receive partial information of the world.
- **Challenges:** Existing QRL architectures do not have the capabilities to **memorize** previous time steps.
- **Approach:** Could quantum recurrent neural nets (QRNN) be helpful in QRL?





Chen, S. Y. C., Yoo, S., & Fang, Y. L. L. (2022, May). Quantum long short-term memory. In ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 8622-8626). IEEE.

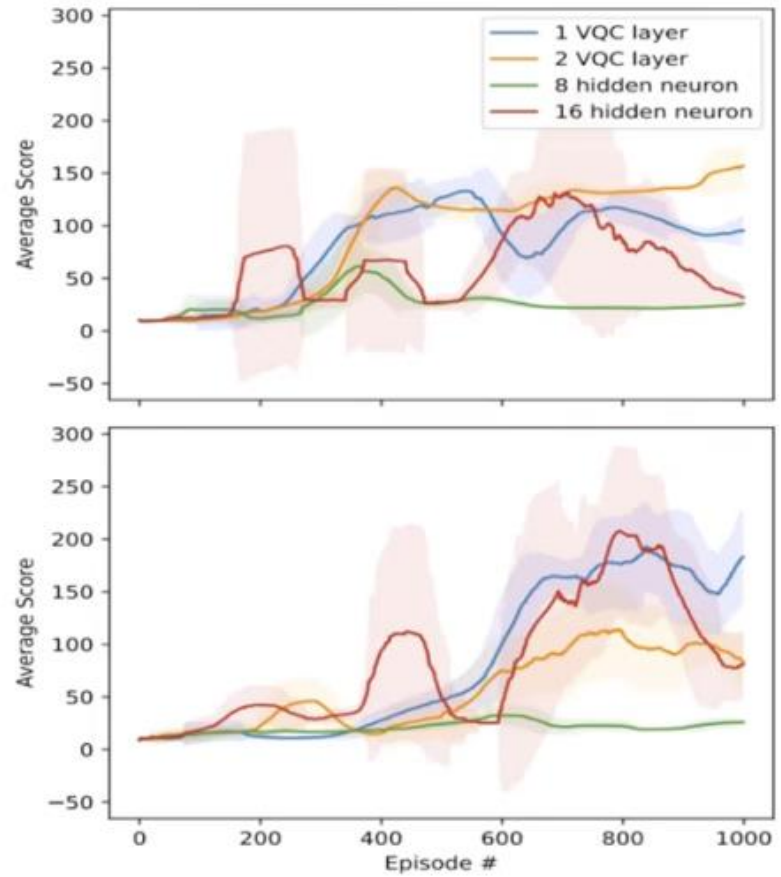
Quantum Recurrent RL, Continued

Algorithm 1 Quantum deep recurrent Q-learning

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function dressed QLSTM Q with random parameters θ
Initialize target dressed QLSTM Q with $\theta^- = \theta$
for episode $= 1, 2, \ldots, M$ **do**
  Initialize the episode record buffer $\mathcal{M}$
  Initialise state s_1 and encode into the quantum state
  Initialize h_1 and c_1 for the QLSTM
  for $t = 1, 2, \ldots, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(s_t, a; \theta)$ from the output of the QLSTM
    Execute action a_t in emulator and observe reward r_t and next state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{M}$
    Sample random batch of trajectories $\mathcal{T}$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } s_{j+1} \\ r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta) & \text{for non-terminal } s_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(s_j, a_j; \theta^-))^2$
    Update the target network θ^- every S steps.
  end for
  Store episode record $\mathcal{M}$ to $\mathcal{D}$
  Update ϵ
end for

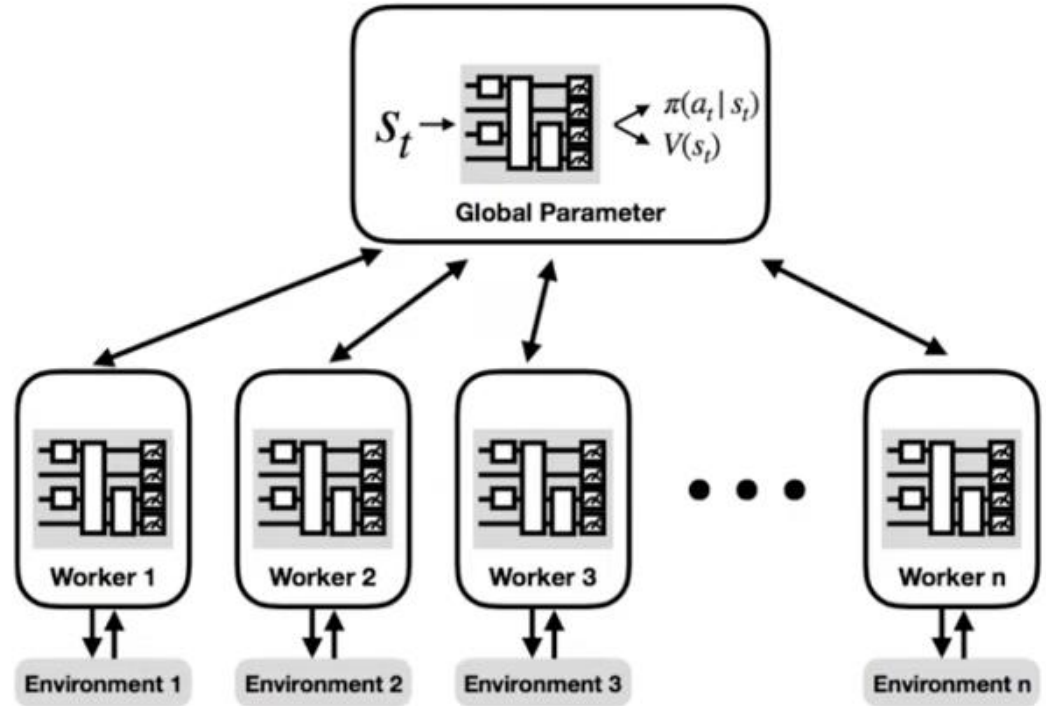
	QLSTM-1	QLSTM-2	LSTM-8	LSTM-16
Full	150	270	634	2290
Partial	146	266	626	2274

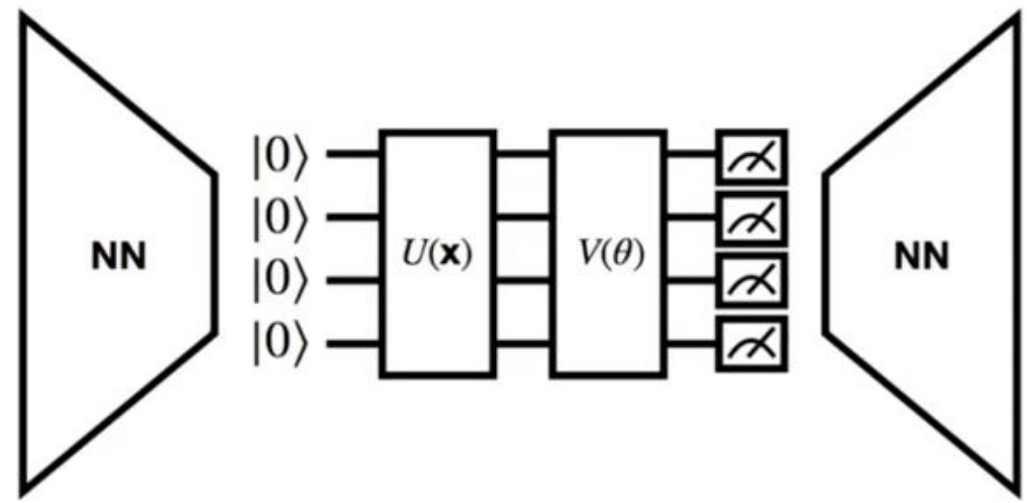
Table 1. Number of parameters.



Asynchronous quantum RL

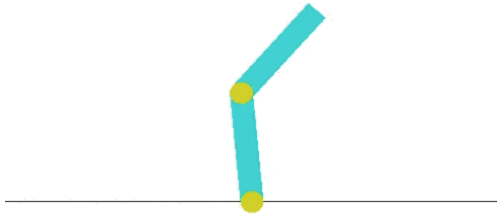
- **Motivation:** VQC based quantum RL models have been shown to perform well in a variety of benchmark tasks.
- **Challenges:** Training of VQC-based quantum RL requires long training time, limiting the exploration of application possibilities.
- **Approach:** Could we further improve the training speed via asynchronous or parallel training?





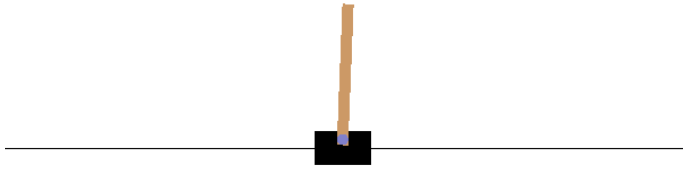
Asynchronous quantum RL continued:

Acrobot Environment:



- **State (s)**:
  - The state vector consists of the following components: `[cos(θ1), sin(θ1), cos(θ2), sin(θ2), θ̇1, θ̇2]`. These represent the angular positions and velocities of the two links of the Acrobot system.
- **Action**:
  - **Action 0**: Apply **negative torque** (counterclockwise).
  - **Action 1**: Apply **no torque** (neutral).
  - **Action 2**: Apply **positive torque** (clockwise).
- **Q-values**:
  - The Q-values (`Q(s, a)`) represent the expected future reward for taking a specific action in a given state. These values are updated during training and are used by the agent to decide which actions to take.

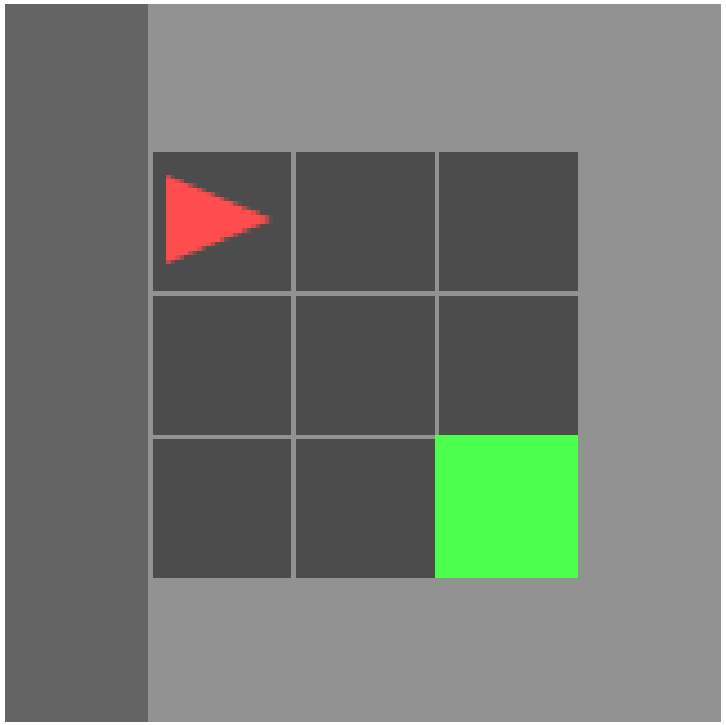
Cartpole Environment:



- **State Space**: 4 continuous variables:
  1. Cart position (`x`)
  2. Cart velocity (`x_dot`)
  3. Pole angle (`theta`)
  4. Pole angular velocity (`theta_dot`)
- **Action Space**: 2 discrete actions:
  1. **Action 0**: Move the cart to the left
  2. **Action 1**: Move the cart to the right

Asynchronous quantum RL continued:

MiniGrid Environment:



State for MiniGrid:

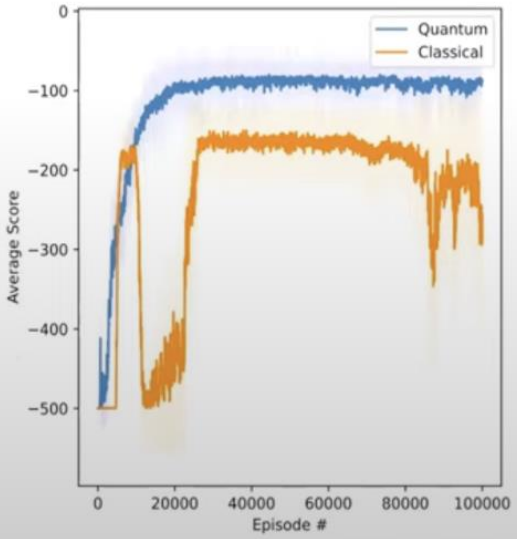
1. **Agent's Position**: The (x, y) coordinates of the agent in the grid.
2. **Agent's Direction**: The direction the agent is facing (e.g., "Up", "Down", "Left", "Right").
3. **Objects in the Grid**: The objects located in the environment (e.g., walls, doors, keys, goal).
4. **Agent's Inventory**: The items the agent is carrying (e.g., a key, box).
5. **Observation Grid**: A small region around the agent's current position (e.g., 3x3 grid of objects).
6. **Task-Specific Attributes**: Additional attributes relevant to the task (e.g., key status, goal location).

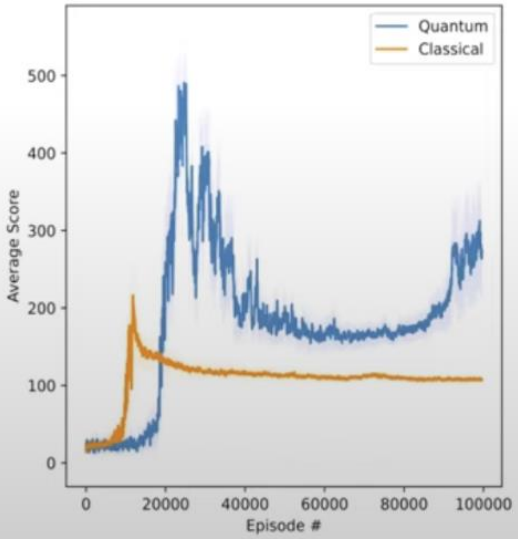
Action for MiniGrid:

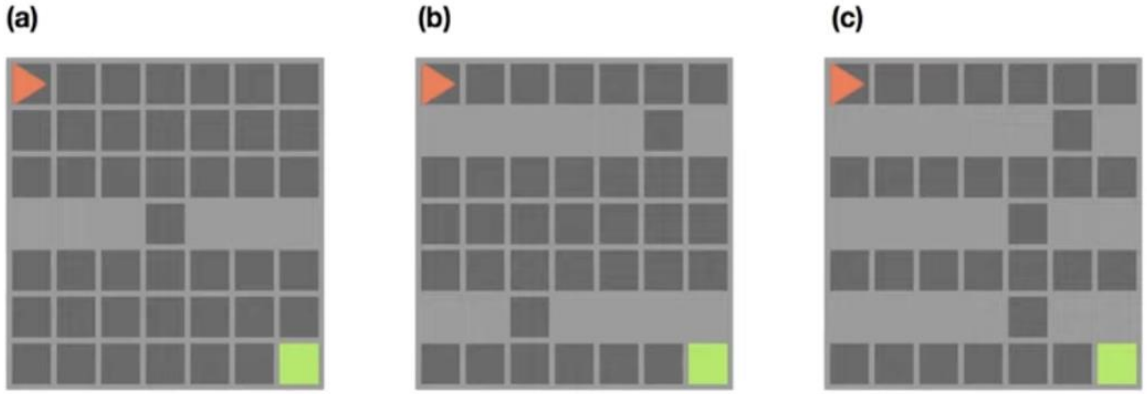
1. **Move Forward**: Move one step forward in the direction the agent is facing.
2. **Turn Left**: Rotate the agent 90 degrees counter-clockwise.
3. **Turn Right**: Rotate the agent 90 degrees clockwise.
4. **Pick Up Object**: Pick up an object if it's in the current cell.
5. **Use Object**: Use an object from the agent's inventory (e.g., use a key to open a door).

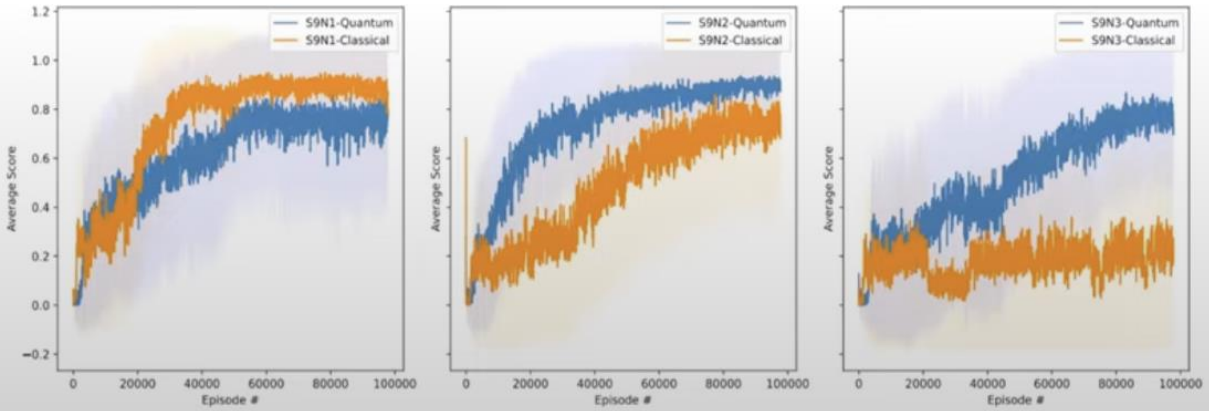
Asynchronous quantum RL, Continued

	QA3C			Classical
	Classical	Quantum	Total	Total
Acrobot	148	96	244	292
Cart-Pole	107	96	203	251
SimpleCrossing	2431	96	2527	2575









Evolutionary Quantum RL

Why?

- Gradient-based methods may suffer from local optima.
- Certain QRL models are difficult to train through gradient-based methods.
- In classical RL, evolutionary optimization can beat gradient-based methods in some hard tasks.

Evolutionary Optimization

- **Initialization:**

  Initialize the population $\mathscr{P}$ of N agents with each of them given randomly generated initial parameters θ, which are sampled from $\mathcal{N}(0, I)$.

- **Running and evaluating the agents:**

  - Each agent plays the game R_1 times and get the average score $S_i^{avg} = \frac{1}{R_1} \sum_{r=1}^{R_1} S_{i,r}$.
  - Top T agents age selected to be the *parents* to generate the next generation.

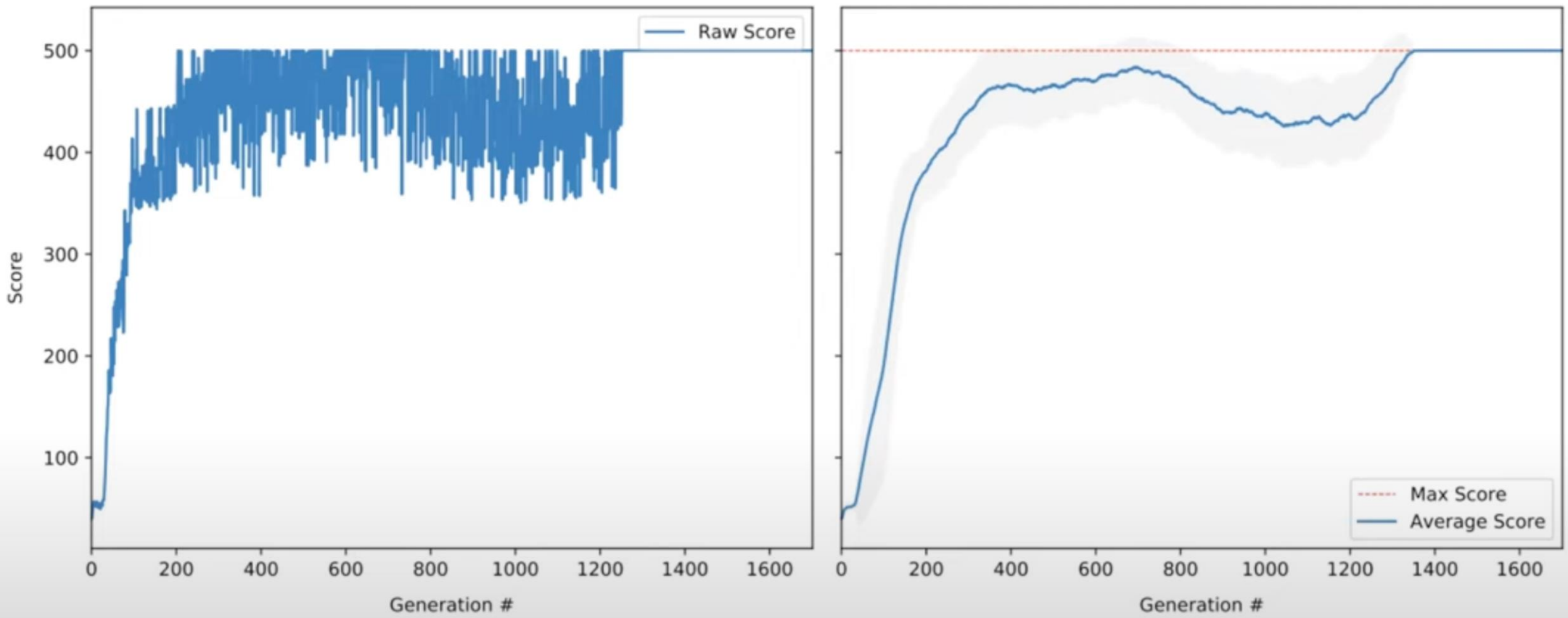
- **Mutation and the next generation:**

  - $N - 1$ children: Each child is generated via a randomly selected agent from the parent group and slightly mutated according to $\theta \leftarrow \theta + \sigma\epsilon$ where σ is the mutation power and ϵ is the Gaussian noise.
  - The *elite* or N^{th}- child is the best performing from the parent group.

Environments - CartPole

- **Observation:** A four dimensional vector s_t comprising values of the cart position, cart velocity, pole angle and pole velocity at the top.
- **Action:** There are two actions: pushing to the *right* or *left*.
- **Reward:** A reward +1 is given for every time step where the pole close to being upright.

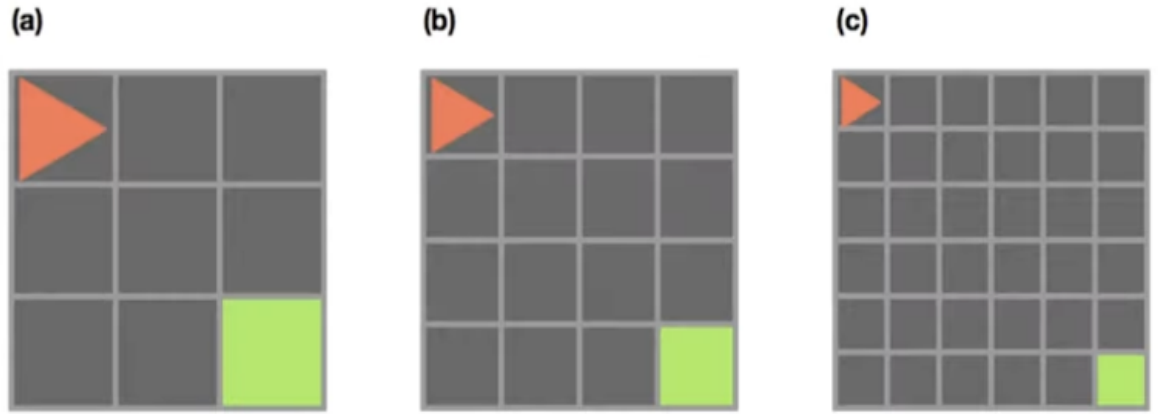
Results - CartPole



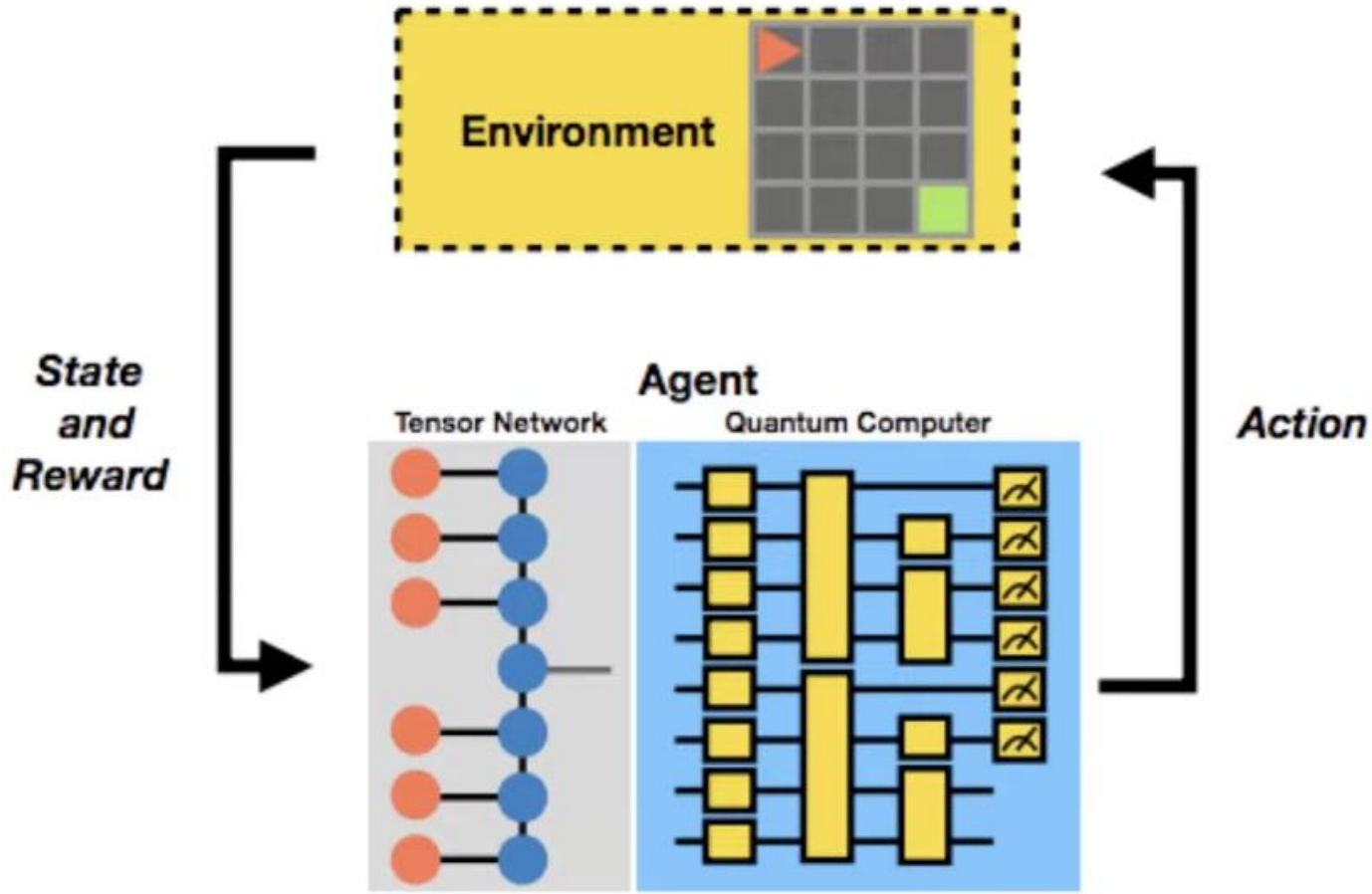
Environments - MiniGrid

- **Observation:** A 147 dimensional vector s_t
- **Action:** There are 6 actions:
  - Turn left
  - Turn right
  - Move forward
  - Pick up an object
  - Drop the object
  - Toggle
- **Reward:** A reward of 1 is given when the agent reaches the goal. A penalty is subtracted from the reward according to:

  1 – 0.9 × (*number of steps/max steps allowed*)



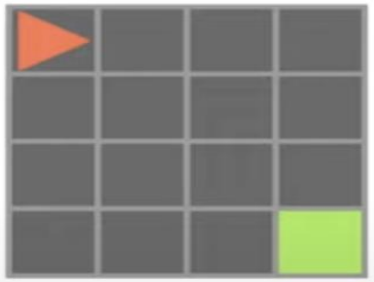
Hybrid TN-VQC model



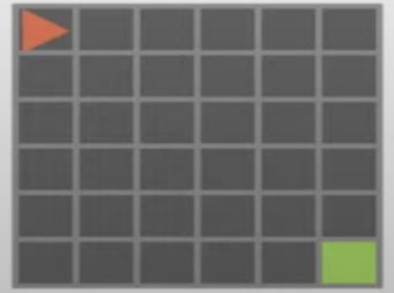
Results - MiniGrid

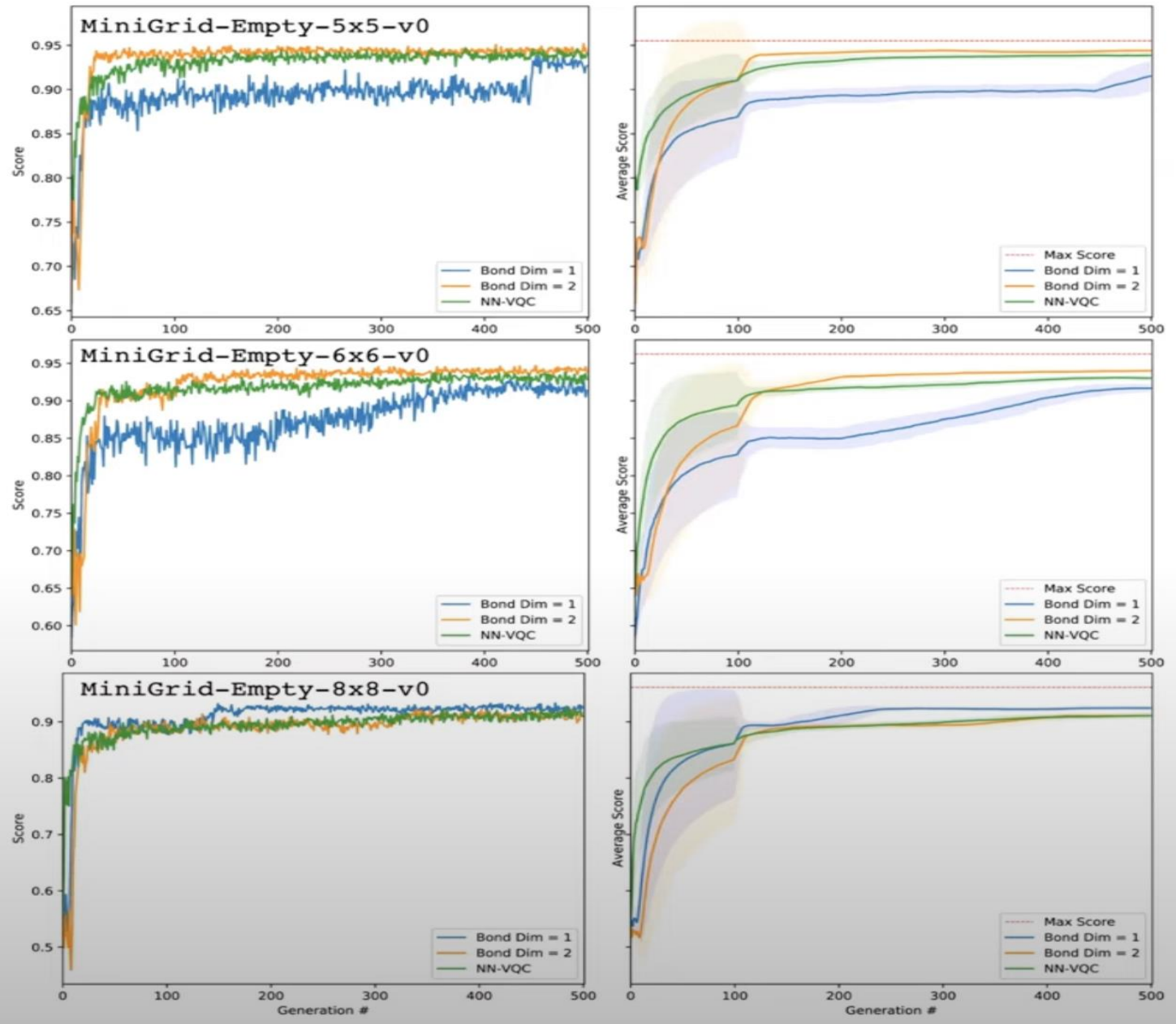
(a)

(b)



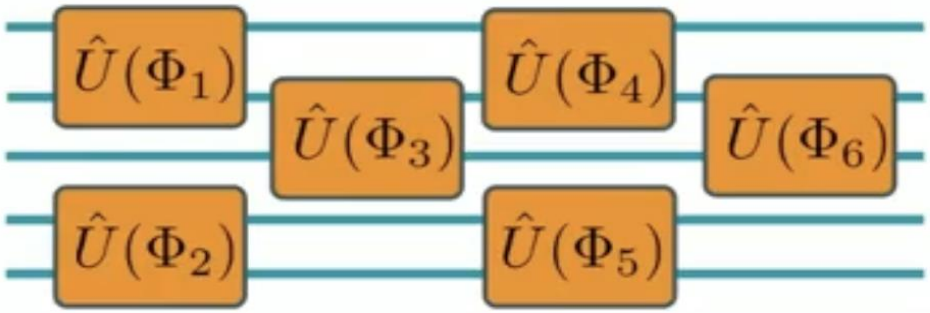
(c)



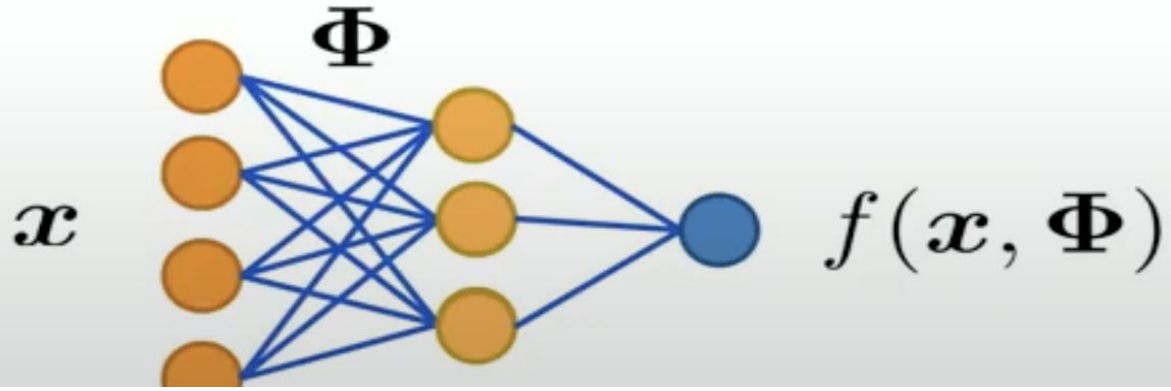


TensorFlow Quantum –Cirq:

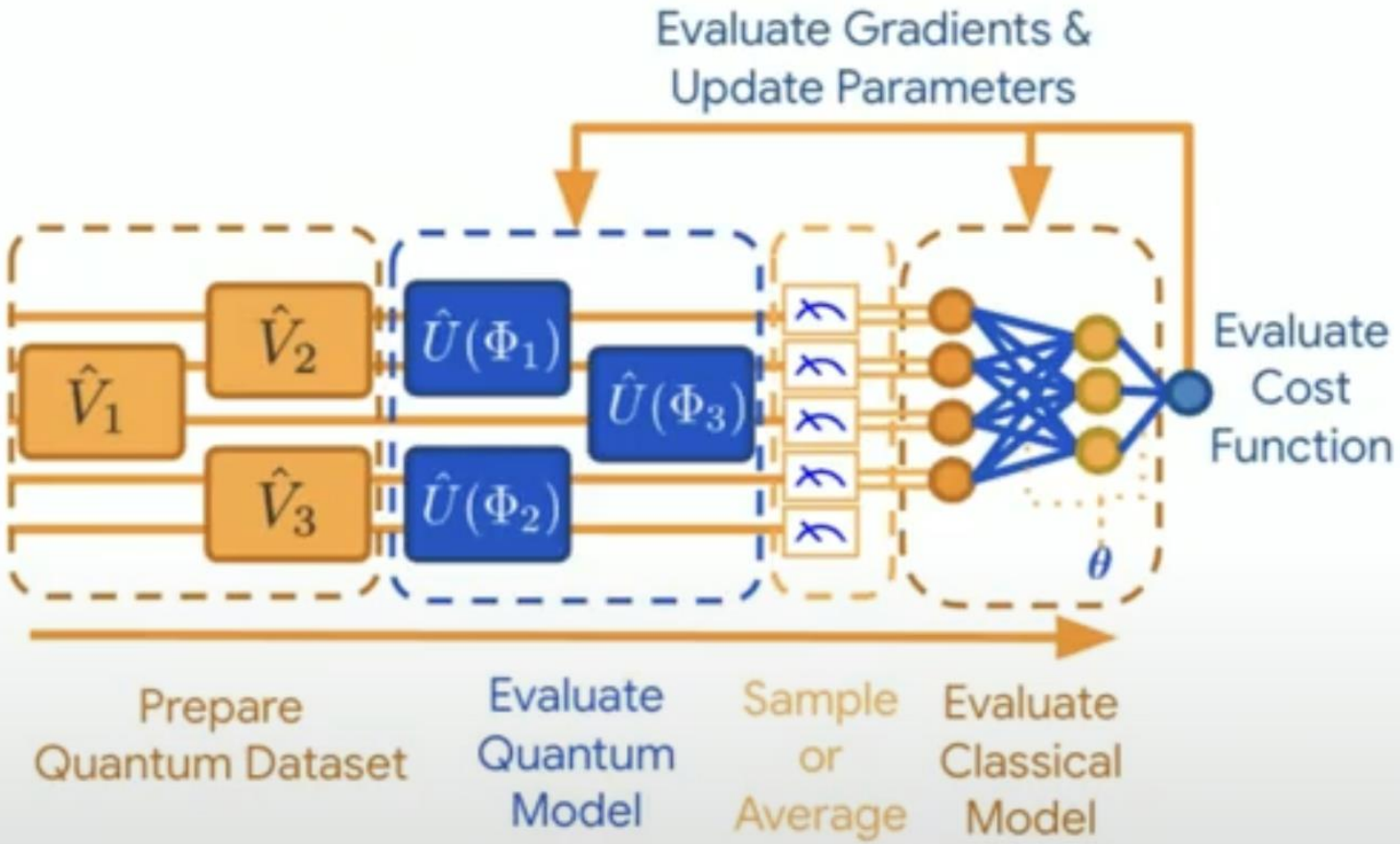
Parameterized Quantum Circuits



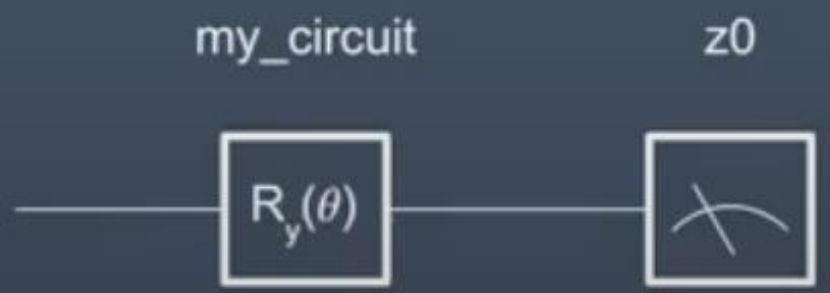
- Sequence of continuously-parameterized “rotations”
- Forms a parameterized quantum circuit, also known as a ***quantum neural network***



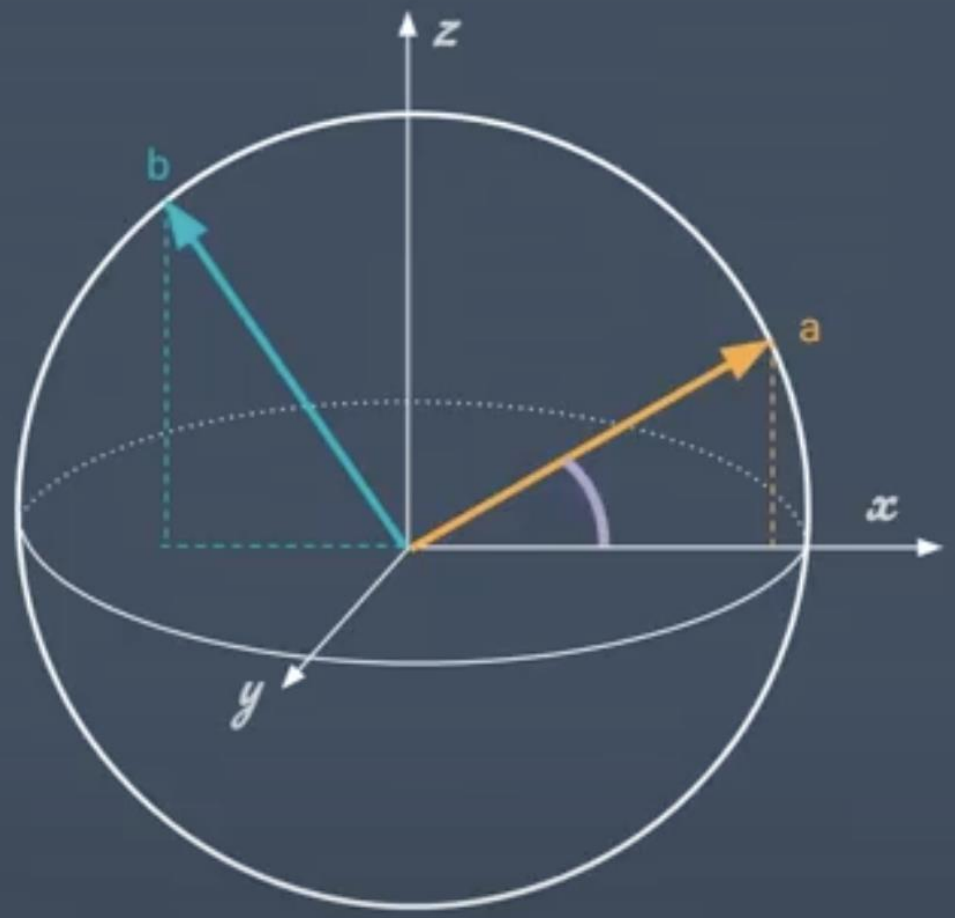
TFQ pipeline for a hybrid discriminative model



```python
import cirq
import sympy

q0 = cirq.GridQubit(0, 0)

theta = sympy.Symbol('theta')
my_circuit = cirq.Circuit(cirq.Ry(theta)(q0))

param_resolver={theta: 1.2}
state_vector = cirq.Simulator().simulate(
     my_circuit, param_resolver).final_state

z0 = cirq.Z(q0)
z0.expectation_from_wavefunction(
     state_vector, qubit_map={q0: 0}).real
```

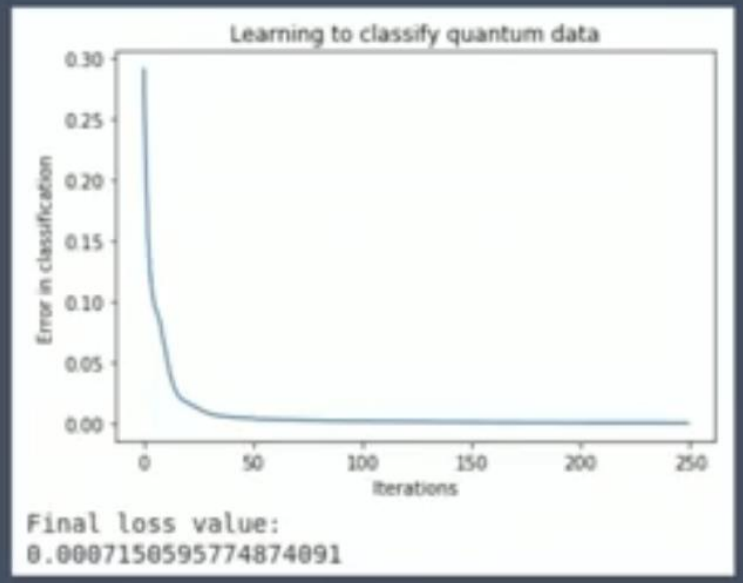


```
import cirq, random, sympy
import numpy as np
import tensorflow as tf
import tensorflow_quantum as tfq

qubit = cirq.GridQubit(0, 0)

# Quantum data labels
expected_labels = np.array([[1, 0], [0, 1]])

# Random rotation of X and Z axes
angle = np.random.uniform(0, 2 * np.pi)

# Build the quantum data

a = cirq.Circuit(cirq.Ry(angle)(qubit))
b = cirq.Circuit(cirq.Ry(angle + np.pi/2)(qubit))
quantum_data = tfq.convert_to_tensor([a, b])
```




```
# Build the quantum model
q_data_input = tf.keras.Input(shape=(), dtype=tf.dtypes.string)

theta = sympy.Symbol('theta')
q_model = cirq.Circuit(cirq.Ry(theta)(qubit))

expectation = tfq.layers.PQC(q_model, cirq.Z(qubit))
expectation_output = expectation(q_data_input)

# Attach the classical SoftMax classifier
classifier = tf.keras.layers.Dense(2, activation=tf.keras.activations.softmax)
classifier_output = classifier(expectation_output)
```


```
# Train the hybrid model
model = tf.keras.Model(inputs=q_data_input,
     outputs=classifier_output)
model.compile(
     optimizer=tf.keras.optimizers.Adam(learning_rate=0.1),
     loss=tf.keras.losses.CategoricalCrossentropy())
history = model.fit(x=quantum_data, y=expected_labels,
     epochs=250, verbose=0)
```




```
Final loss value:
0.0007150595774874091
```

```
# Check inference on noisy quantum datapoints
noise = np.random.uniform(-0.25, 0.25, 2)
test_data = tfq.convert_to_tensor([
     cirq.Circuit(
          cirq.Ry(random_angle + noise[0])(qubit)),
     cirq.Circuit(
          cirq.Ry(random_angle + noise[1] + np.pi/2)(qubit))
])
predictions = model.predict(test_data)
```

```
Noisy element from a:
prob(0)=0.9995, prob(1)=0.0005
Noisy element from b:
prob(0)=0.0025, prob(1)=0.9975
```

Comparative Analysis:

Factor	Quantum RL (QRL)	Classical RL
State Representation	Handles high-dimensional state spaces efficiently via superposition.	Struggles with exponential state spaces in large problems.
Speed	Potential quadratic speedup for search and sampling.	Linear or polynomial scaling, but hardware is faster.
Hardware Limitations	Constrained by qubit count, noise, and decoherence.	Highly optimized and scalable on classical systems.
Optimization Landscape	Faces barren plateaus but benefits from quantum gradients.	Stable, well-studied optimization methods.
Applications	May excel in quantum-native problems (e.g., quantum control, combinatorial RL).	Performs well across most classical domains (e.g., games).

Conclusion and Outlook

- Quantum encoding / embedding methods are critical.
- Trainable quantum-inspired classical architectures help data compression.
- With careful design, quantum reinforcement learning can learn a similar task with fewer model parameter.
- Gradient-based and gradient-free algorithms for quantum RL.
- **Quantum recurrent neural networks (QRNN)** can be used to further enhance the QRL.
- **Asynchronous training** can be used to further reduce the training time of QRL.
- Reinforcement learning can help finding quantum circuit architecture under various patterns.
- Previously-learned policies can be used to help the training of RL agent for unseen environme (**changing noise environments**).
- RL applications: **quantum architecture search**, **quantum compiler**, **quantum error correction**

References:

1. **Variational Quantum Algorithms (VQAs)**
   Cerezo, M., Arrasmith, A., Babbush, R., Benjamin, S. C., Endo, S., Fujii, K., ... & Coles, P. J. (2021). Variational quantum algorithms. *Nature Reviews Physics*, vol. 3, no. 9, pp. 625-644.
   DOI: 10.1038/s42254-021-00348-9
   Comprehensive review of VQAs, covering their applications, including optimization problems in RL.

2. **Quantum Reinforcement Learning (QRL)**
   Jerbi, S., Marsh, B., Nguyen, L., Heya, K., & Sornborger, A. (2023). Quantum reinforcement learning with parameterized quantum policies. *Quantum Information Processing*, vol. 22, no. 4, pp. 123.
   DOI: 10.1007/s11128-023-03754-5
   Introduces parameterized quantum policies for reinforcement learning and highlights potential advantages.

3. **Quantum Speedup in RL**
   Dong, D., Chen, C., Li, H., & Tarn, T. J. (2008). Quantum reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 5, pp. 1207-1220.
   DOI: 10.1109/TSMCB.2008.925743
   One of the foundational works in QRL, demonstrating quantum speedup in specific reinforcement learning tasks.

4. **Challenges in Quantum Optimization**
   McClean, J. R., Boixo, S., Smelyanskiy, V. N., Babbush, R., & Neven, H. (2018). Barren plateaus in quantum neural network training landscapes. *Nature Communications*, vol. 9, pp. 4812.
   DOI: 10.1038/s41467-018-07090-4
   Explores the barren plateau phenomenon, a critical challenge for training quantum RL models.