When Reinforcement Learning meets Quantum Computing

Presenter: Samuel Yen-Chi Chen

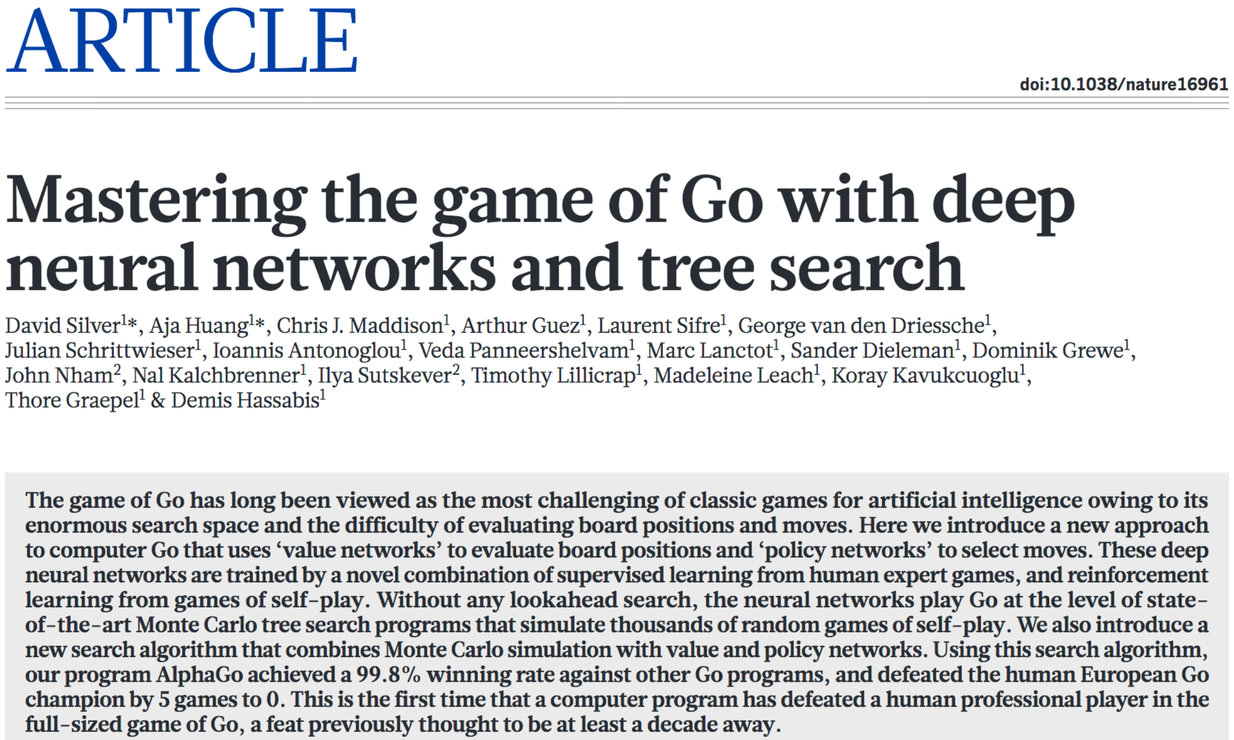
BNL AI/ML Working Group Seminar

Date 2022-03-01

I. Introduction
II. Reinforcement Learning (RL)
III. Quantum Computing (QC)
IV. Quantum RL
V. RL for QC
VI. Conclusion and Outlook

I. **Introduction**
II. Reinforcement Learning (RL)
III. Quantum Computing (QC)
IV. Quantum RL
V. RL for QC
VI. Conclusion and Outlook

2016 AlphaGo


ARTICLE

doi:10.1038/nature16961

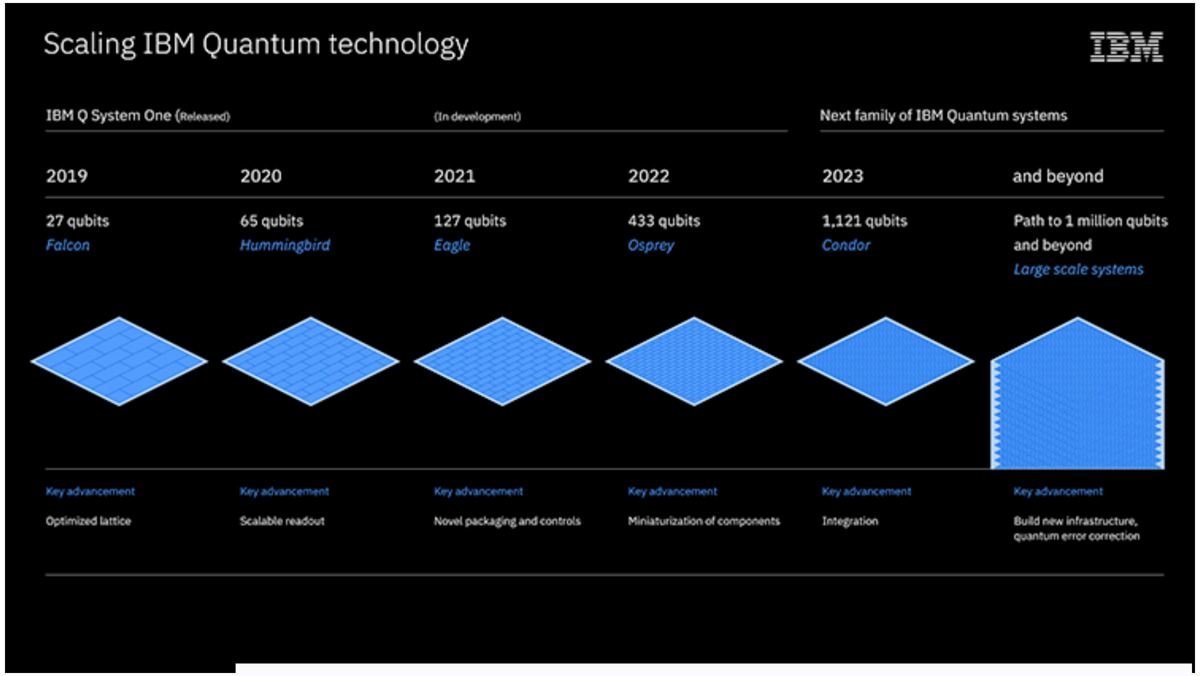
Mastering the game of Go with deep neural networks and tree search

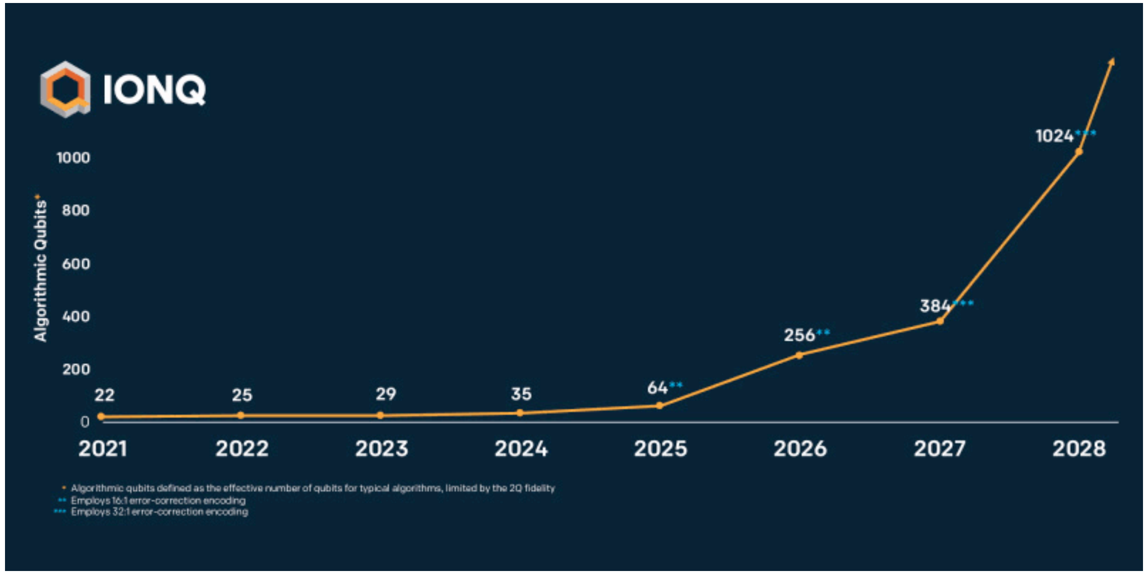
David Silver[1]*, Aja Huang[1]*, Chris J. Maddison[1], Arthur Guez[1], Laurent Sifre[1], George van den Driessche[1], Julian Schrittwieser[1], Ioannis Antonoglou[1], Veda Panneershelvam[1], Marc Lanctot[1], Sander Dieleman[1], Dominik Grewe[1], John Nham[2], Nal Kalchbrenner[1], Ilya Sutskever[2], Timothy Lillicrap[1], Madeleine Leach[1], Koray Kavukcuoglu[1], Thore Graepel[1] & Demis Hassabis[1]

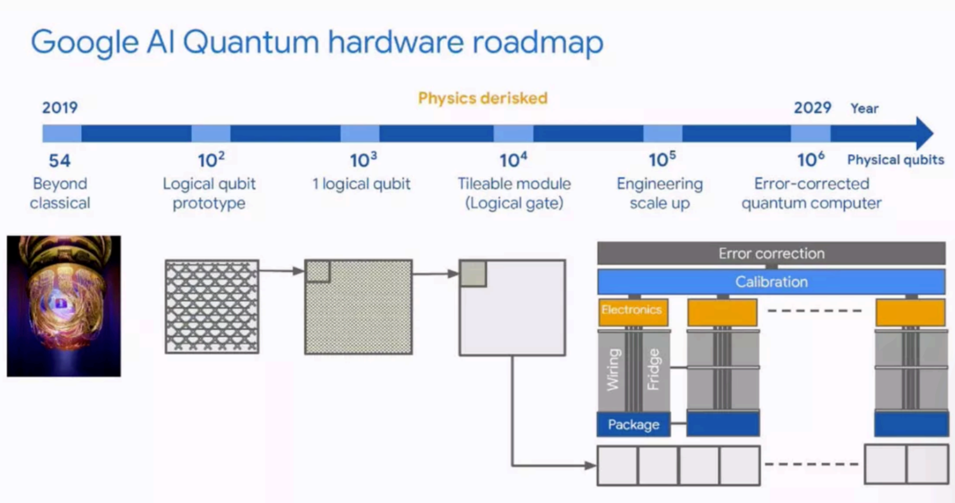
The game of Go has long been viewed as the most challenging of classic games for artificial intelligence owing to its enormous search space and the difficulty of evaluating board positions and moves. Here we introduce a new approach to computer Go that uses 'value networks' to evaluate board positions and 'policy networks' to select moves. These deep neural networks are trained by a novel combination of supervised learning from human expert games, and reinforcement learning from games of self-play. Without any lookahead search, the neural networks play Go at the level of state-of-the-art Monte Carlo tree search programs that simulate thousands of random games of self-play. We also introduce a new search algorithm that combines Monte Carlo simulation with value and policy networks. Using this search algorithm, our program AlphaGo achieved a 99.8% winning rate against other Go programs, and defeated the human European Go champion by 5 games to 0. This is the first time that a computer program has defeated a human professional player in the full-sized game of Go, a feat previously thought to be at least a decade away.

Brookhaven National Laboratory

2017~ QC hardware







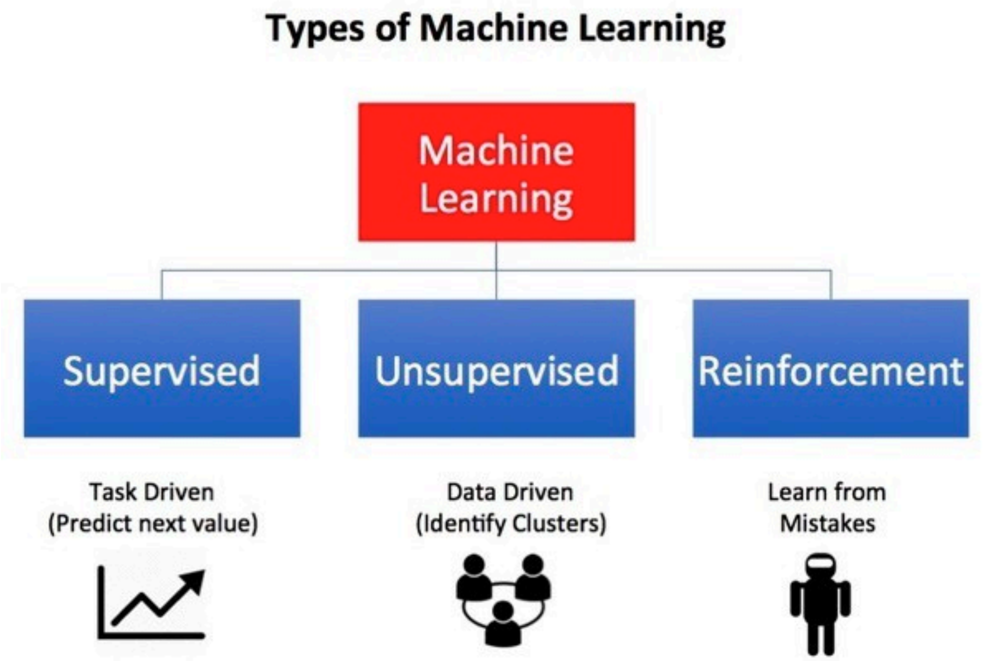
Brookhaven National Laboratory

I. Introduction
II. **Reinforcement Learning (RL)**
III. Quantum Computing (QC)
IV. Quantum RL
V. RL for QC
VI. Conclusion and Outlook

Brookhaven National Laboratory

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.

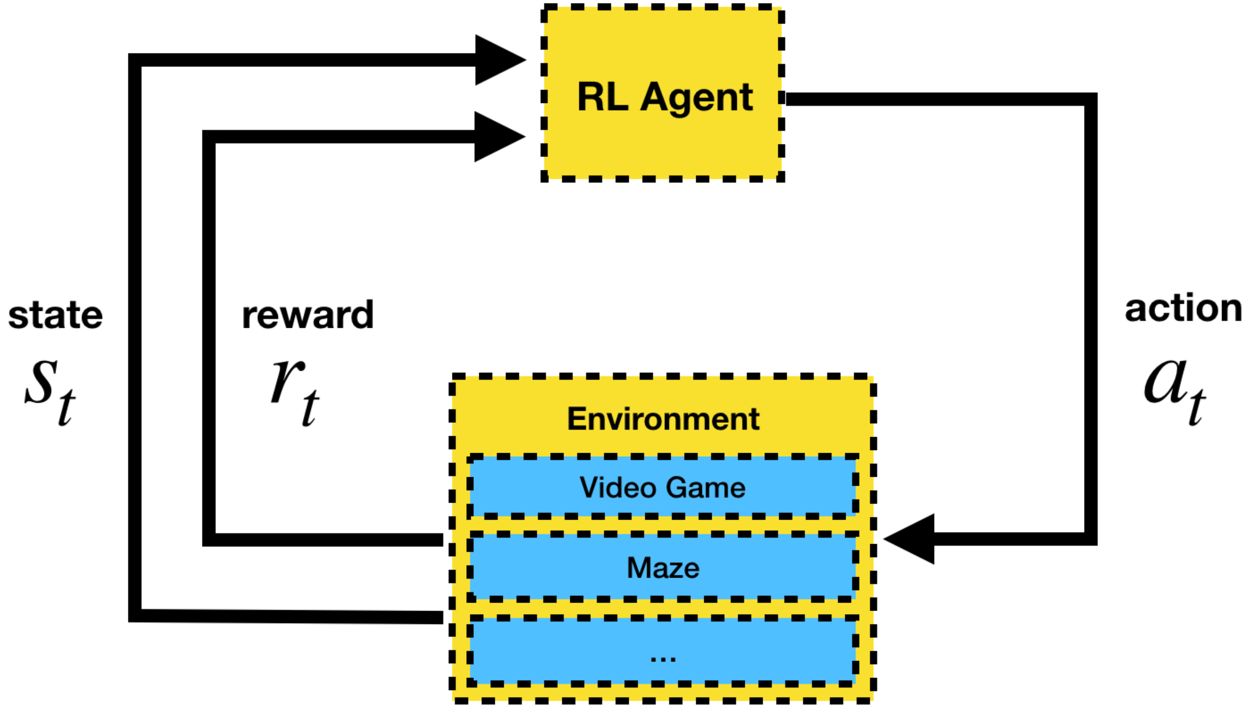


Brookhaven National Laboratory

Reinforcement Learning (RL)

- RL: An agent interacts with an ***environment*** $\mathscr{E}$ over a number of discrete time steps.
- The agent receives ***state*** or ***observation*** s_t and then chooses an ***action*** a_t from a set of actions $\mathscr{A}$ according to its ***policy*** π.
- Goal: Maximize the ***total discounted***

return $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$



Brookhaven National Laboratory

I. Introduction
II. Reinforcement Learning (RL)
III. Quantum Computing (QC)
IV. Quantum RL
V. RL for QC
VI. Conclusion and Outlook

Quantum Computing

- Classical computers: Classical bits 0 vs 1
- Quantum computers: Quantum bits (qubit) $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$ where α and β are complex numbers $\mathbb{C}$
- Quantum entanglements: A unique property of quantum physics —> No analog in the classical computer
- Famous algorithms:
    - Shor's algorithm: Can be used to break the state-of-the-art public key cryptography systems such as RSA
    - Grover's algorithm: Quadratic speedup in unstructured search

- Designing a quantum algorithm is non-trivial task
- Even harder in the noisy quantum machines

Brookhaven National Laboratory

Quantum States

Single Qubit State

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Two Qubit State

$$|0\rangle \otimes |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

N **Qubit State**

$$\underbrace{|0\rangle \otimes |0\rangle \otimes \cdots \otimes |0\rangle}_{N} = \underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{N}$$

Quantum States

Density Operators

$$\rho = \sum_j p_j \left|\psi_j\right\rangle \left\langle\psi_j\right|$$

$\left|\psi_j\right\rangle$ **Basis state**

p_j **Probability**

Examples:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \langle 0| = [1 \quad 0]$$

↓

$$|0\rangle\langle 0| = \begin{bmatrix} 1 \\ 0 \end{bmatrix} [1 \quad 0] = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

Quantum Operations

X $$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Y $$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

Z $$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

H $$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Example:

$|0\rangle$ —— **X**

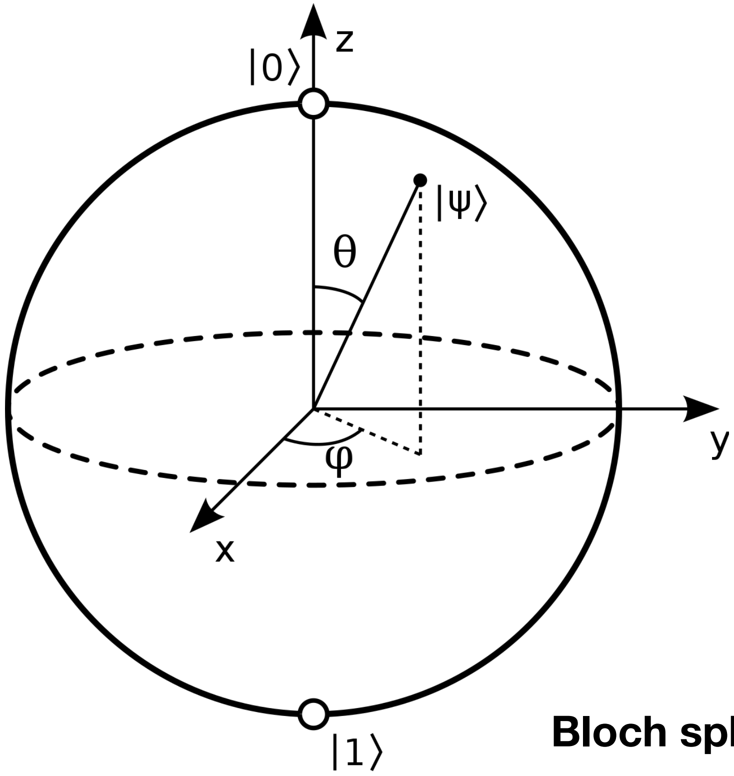
$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$$

Brookhaven National Laboratory

Quantum Operations

$R(\phi, \theta, \omega)$

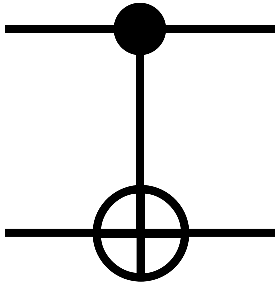
$$\begin{bmatrix} e^{-i(\phi+\omega)/2}\cos(\theta/2) & e^{-i(\phi-\omega)/2}\sin(\theta/2) \\ e^{-i(\phi-\omega)/2}\sin(\theta/2) & e^{i(\phi+\omega)/2}\cos(\theta/2) \end{bmatrix}$$



Bloch sphere

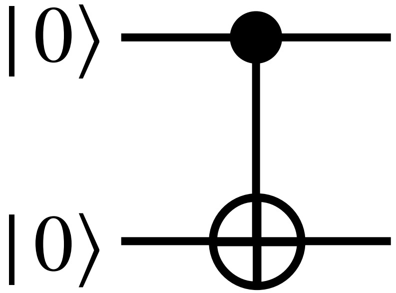
Quantum Operations

CNOT



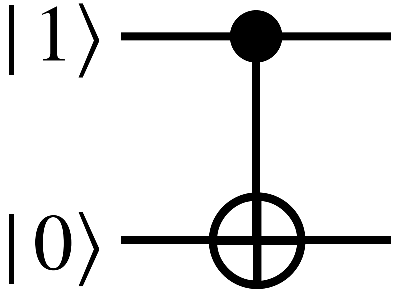
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Result



$|0\rangle$

$|0\rangle$

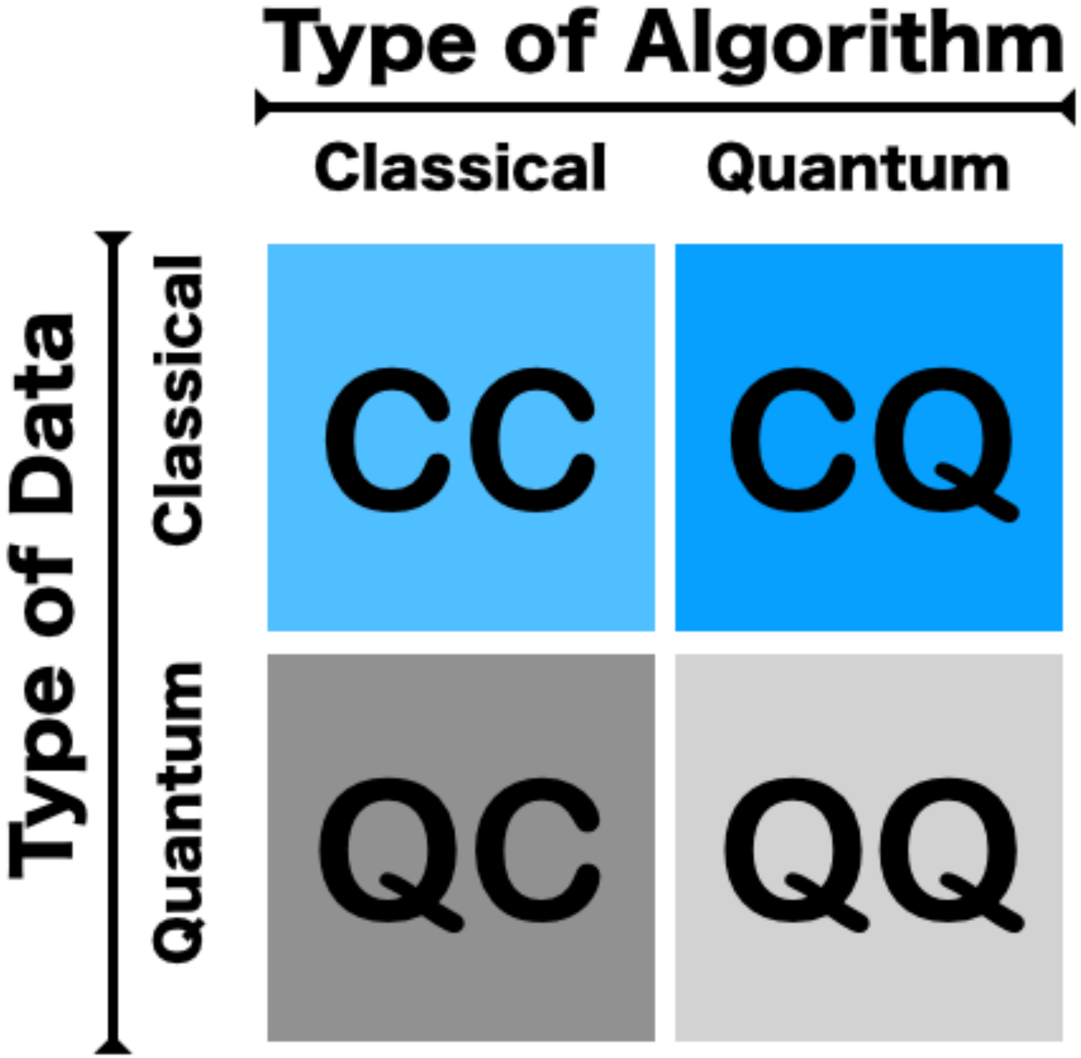


$|1\rangle$

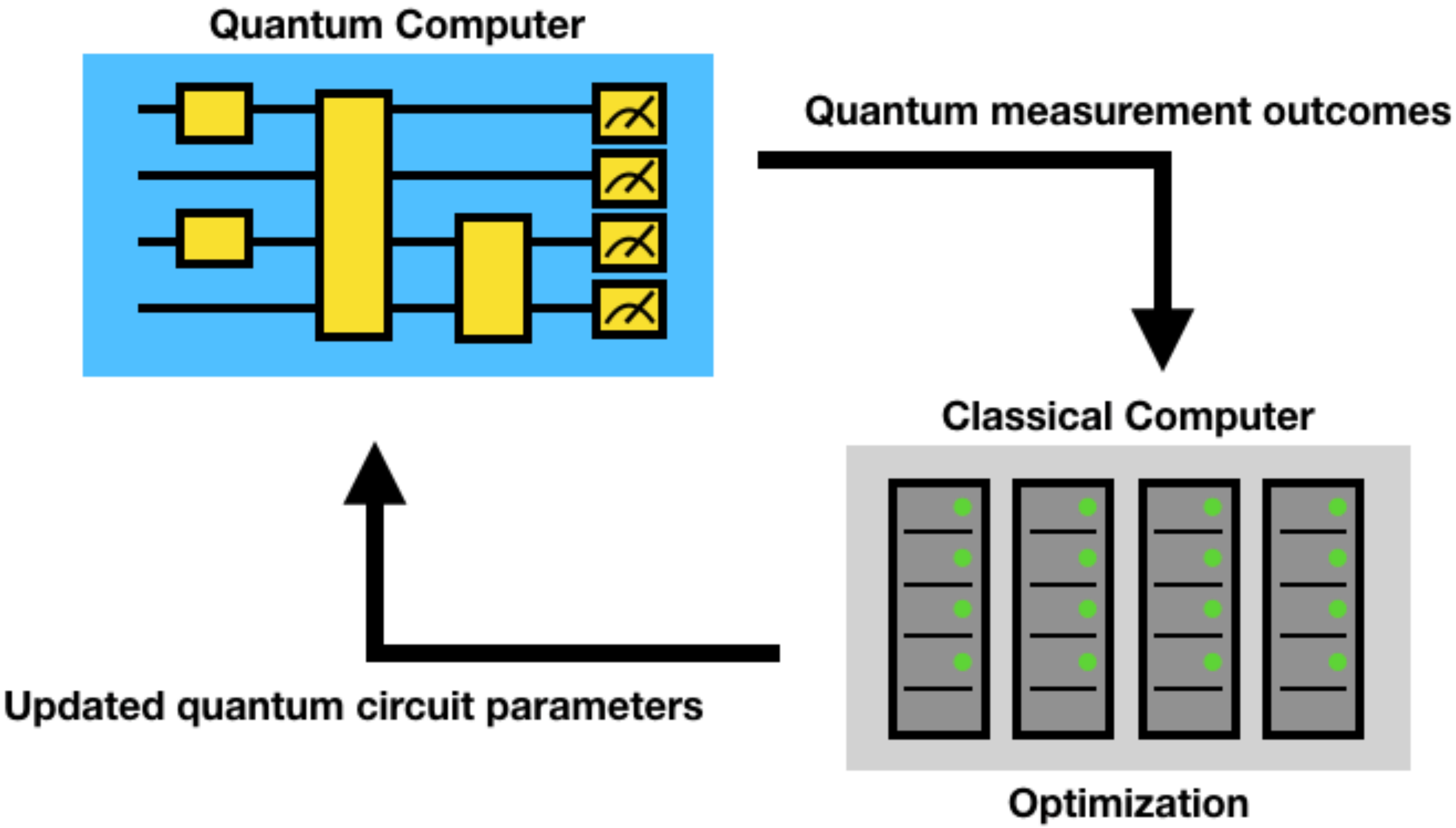
$|1\rangle$

Brookhaven National Laboratory

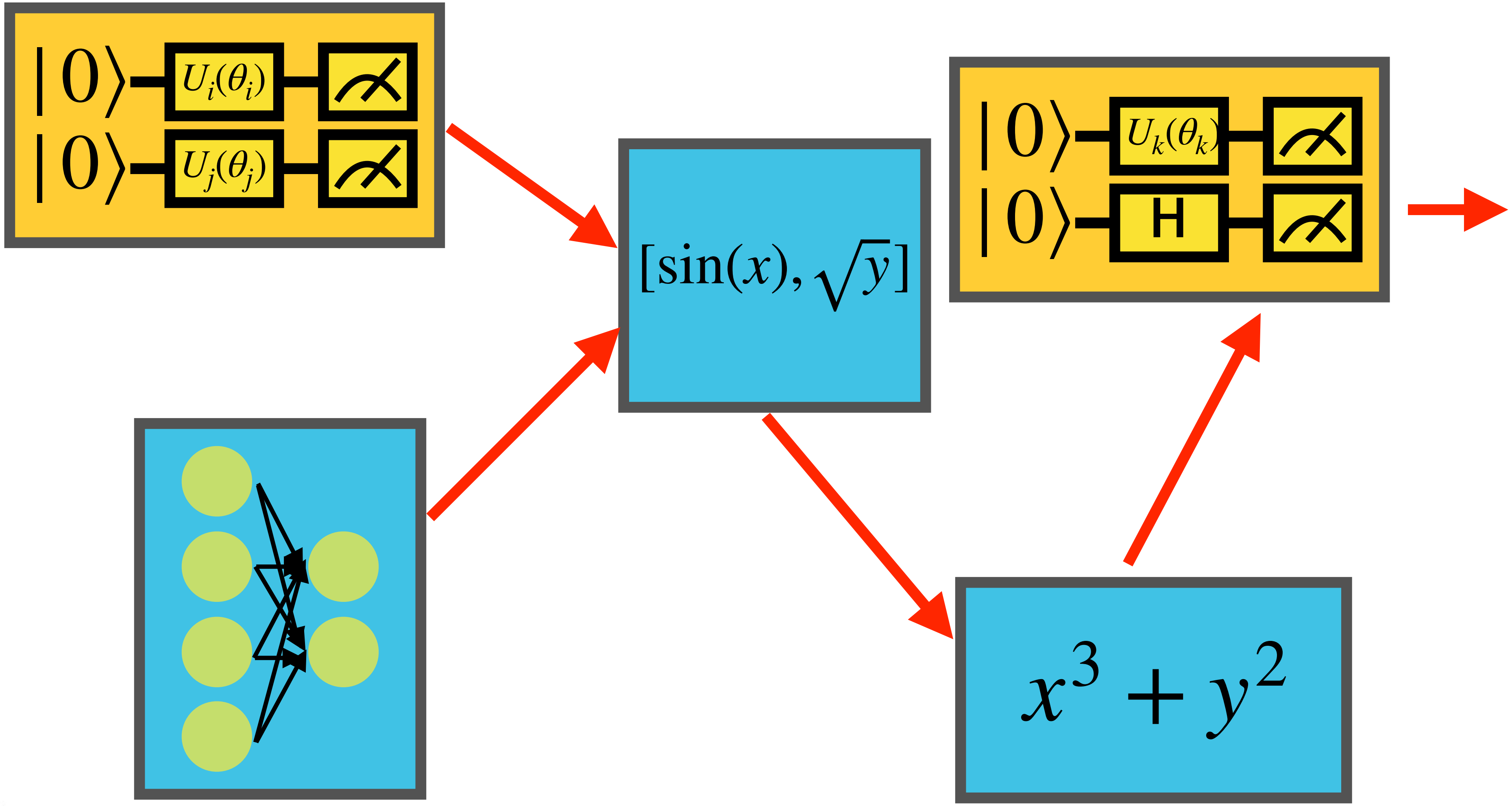
Quantum Machine Learning



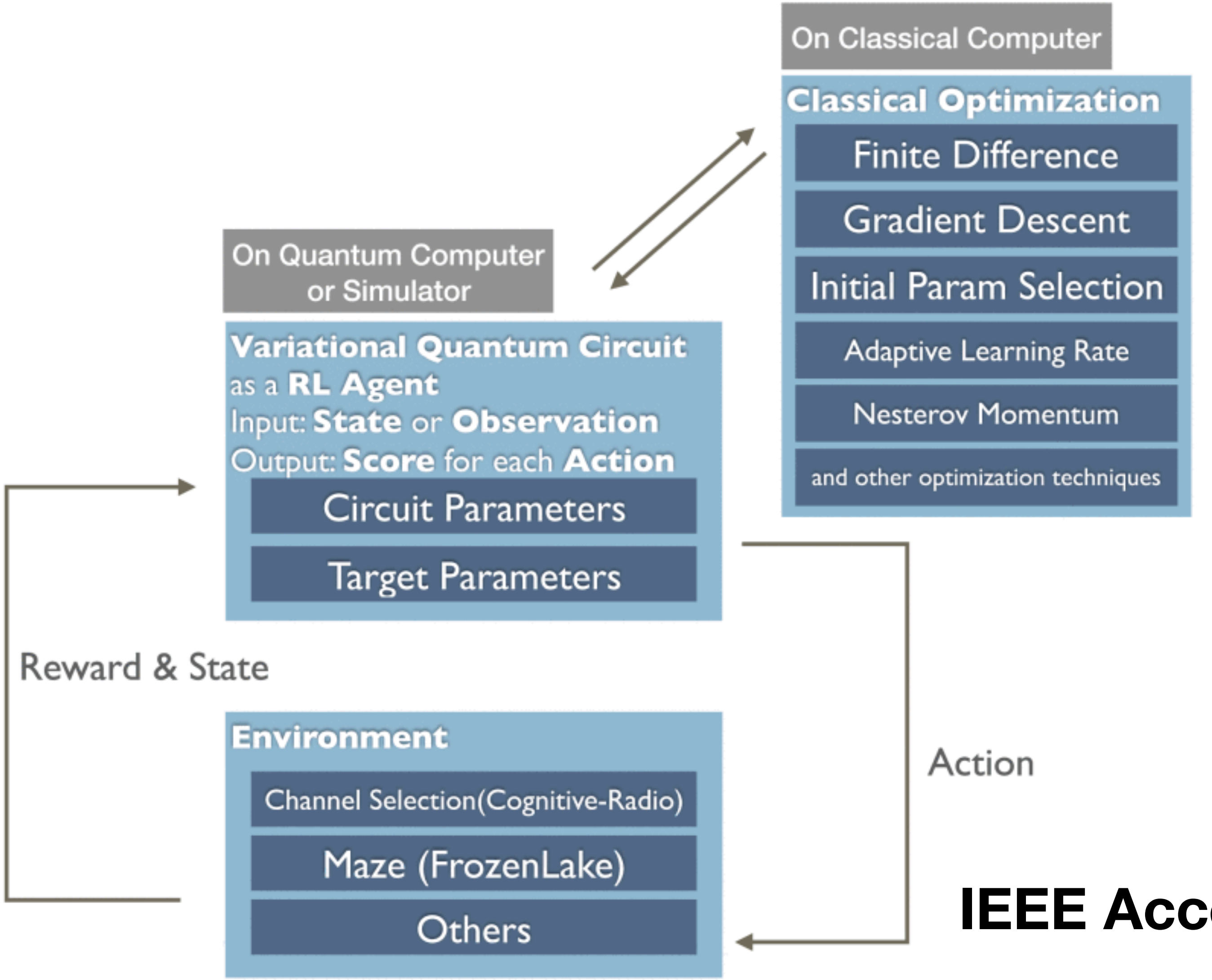
Hybrid Quantum-Classical Paradigm



Interfacing with Classical ML



Quantum Reinforcement Learning

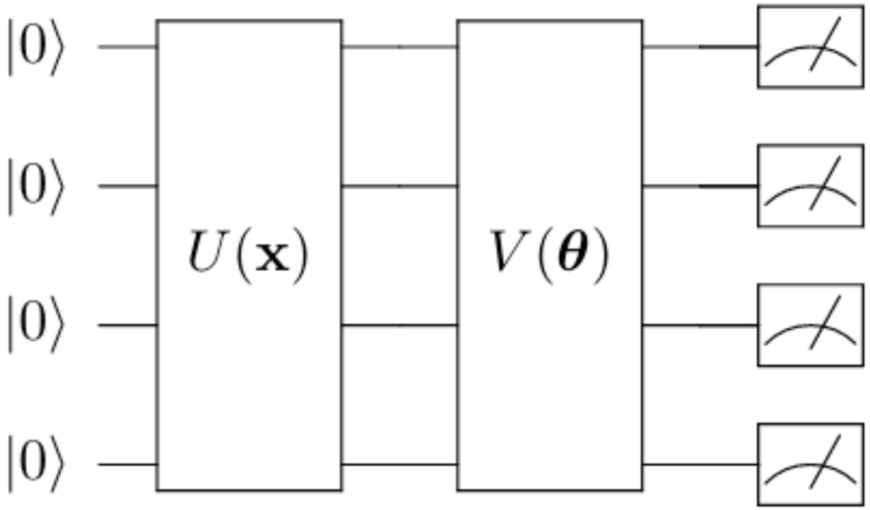


IEEE Access 8, 141007-141024

Brookhaven National Laboratory

Variational Quantum Circuits (VQC)

- Quantum circuits with tunable parameters.
- Subject to iterative optimization procedures.
- $U(\mathbf{x})$: encoding circuit
- $V(\theta)$: variational circuit
-  : measurement



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: $\sum_{(q_1,\cdots,q_N)\in\{0,1\}} \| c_{q_1,\cdots,q_N} \|^2 = 1$

Brookhaven National Laboratory

Quantum Encoding and State Preparation

Amplitude Encoding

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

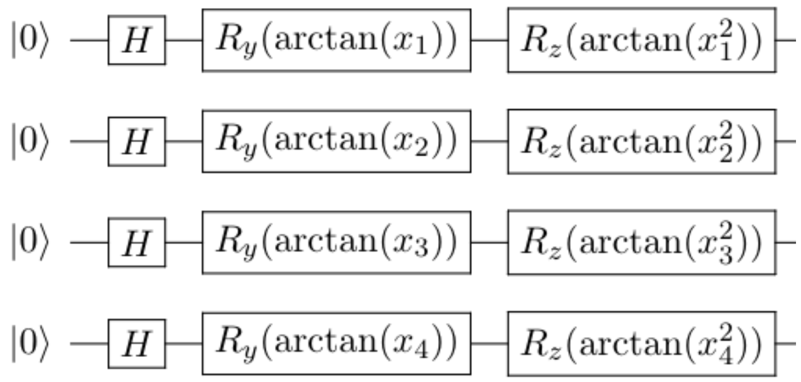
$$|\Psi\rangle = \alpha_0 |00\cdots0\rangle + \cdots + \alpha_{2^n-1} |11\cdots1\rangle$$

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

Brookhaven National Laboratory

Quantum Deep Q-Learning

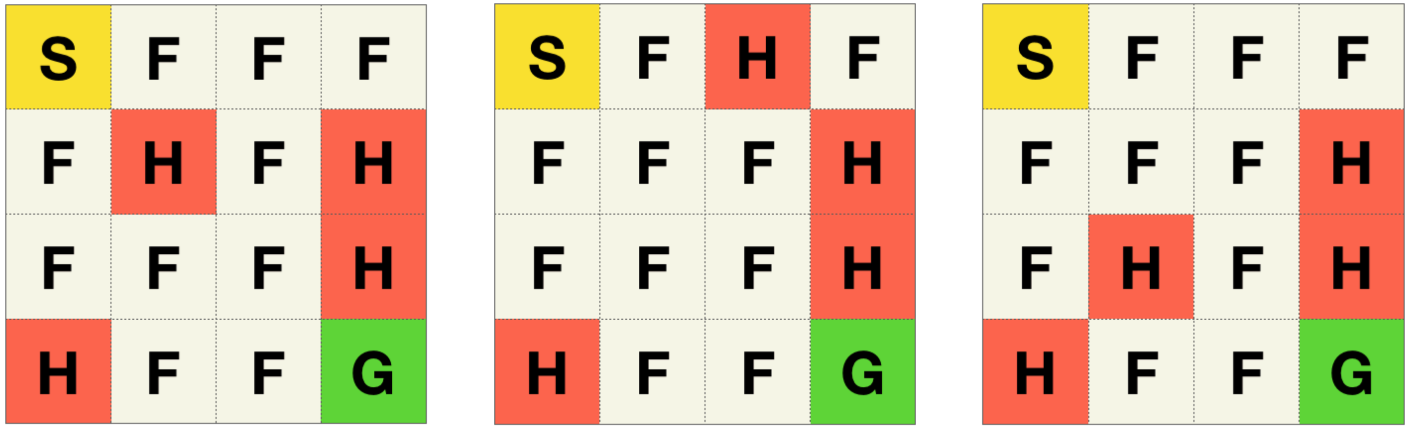
Algorithm 1 Variational Quantum Deep Q Learning

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 $\left(s_j, a_j, r_j, s_{j+1}\right)$ 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

IEEE Access 8, 141007-141024

Quantum Deep Q-Learning

- Env: FrozenLake
- 16 discrete states
- 4 actions



(a) (b) (c)

Location	Reward
HOLE	-0.2
GOAL	1.0
OTHER	-0.01

Environment with 16 states.

States numbered as 0~15

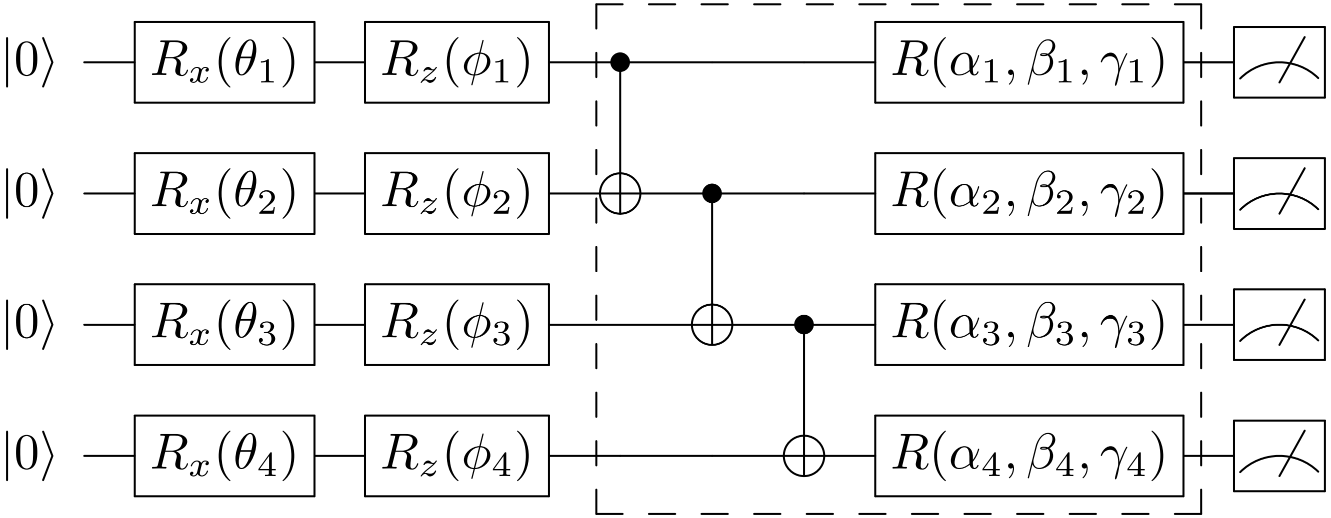
Example: State 12 : 1100 ->1,1,0,0

Rotation :

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

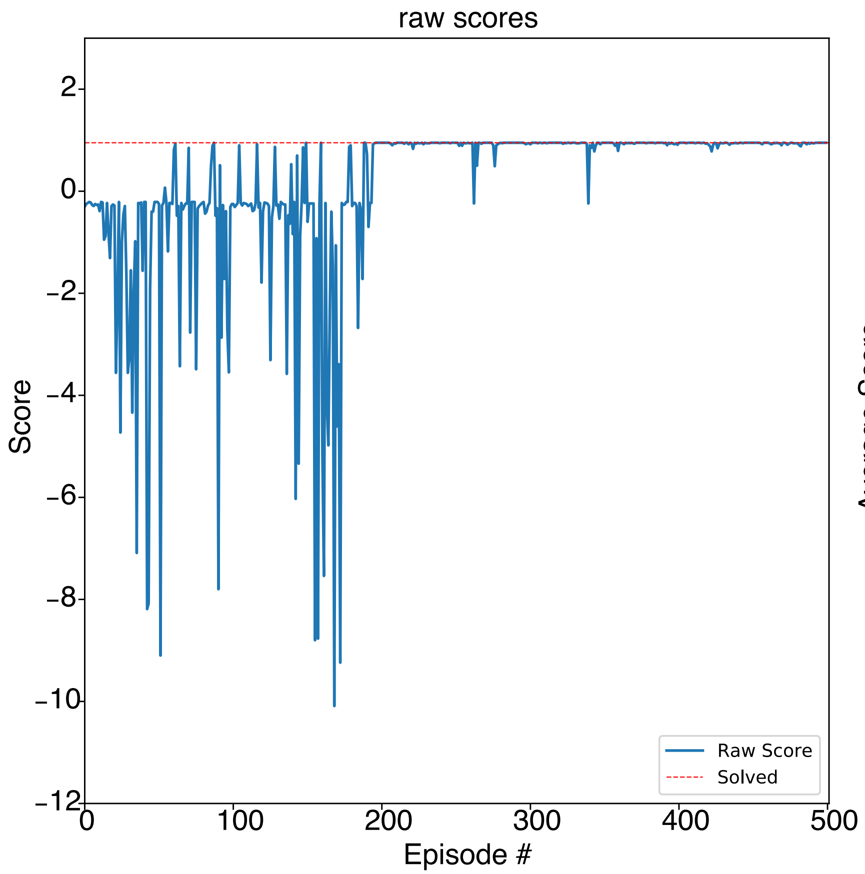
$$\phi_i = \pi \times b_i$$

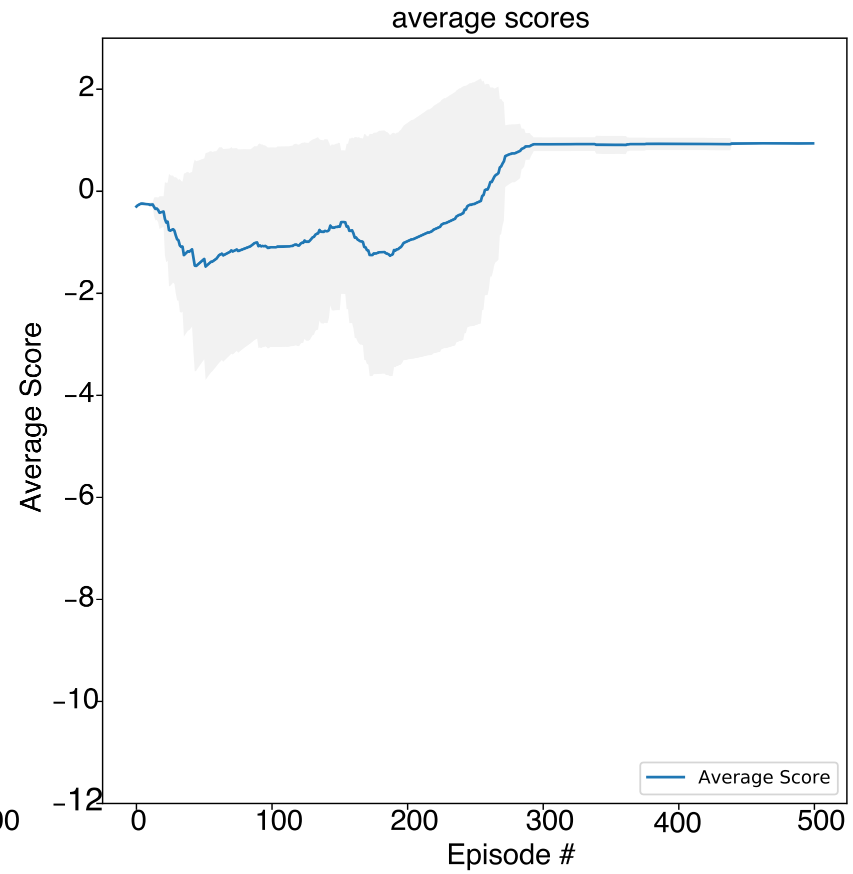
Result : $|1\rangle \otimes |1\rangle \otimes |0\rangle \otimes |0\rangle$

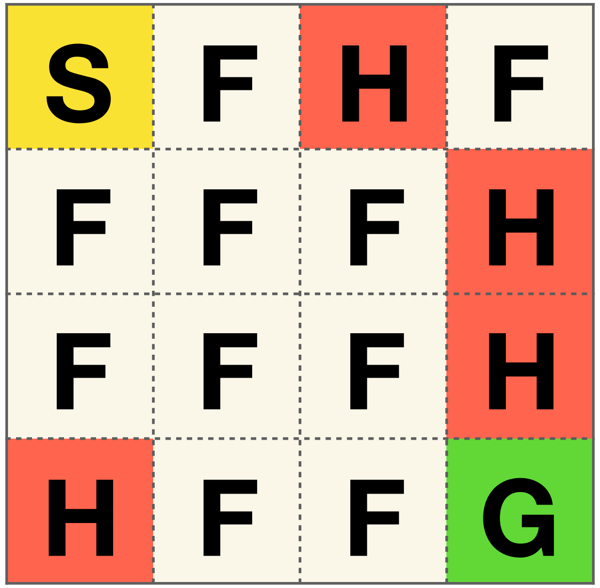


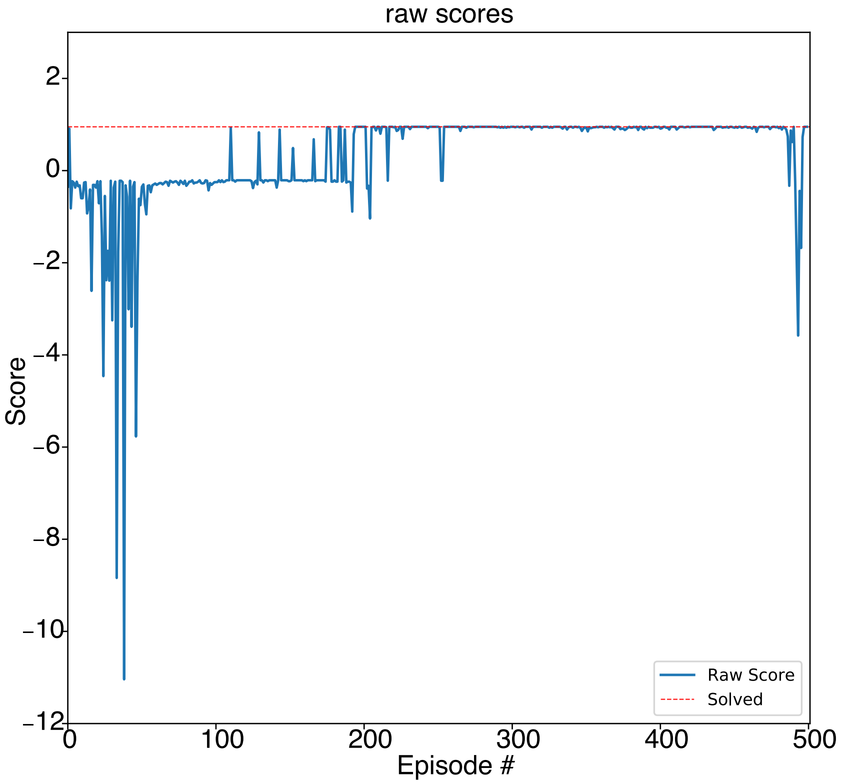
S	F	F	F
F	H	F	H
F	F	F	H
H	F	F	G

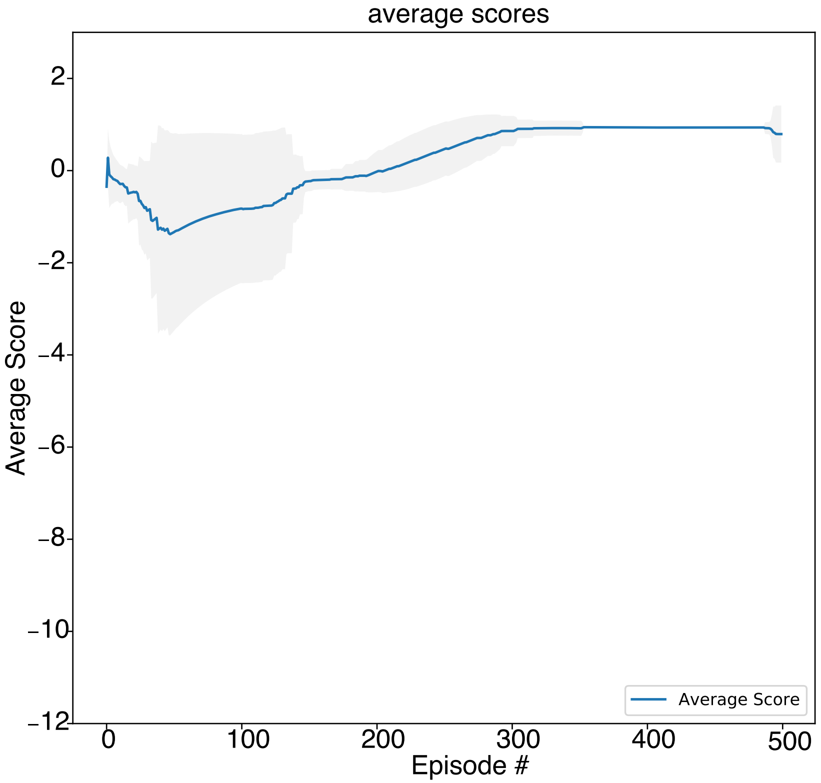

S F F F
F H F H
F F F H
H F F G


raw scores
Score
Episode #
Raw Score
Solved
2
0
−2
−4
−6
−8
−10
−12
0
100
200
300
400
500


average scores
Average Score
Episode #
Average Score
2
0
−2
−4
−6
−8
−10
−12
0
100
200
300
400
500

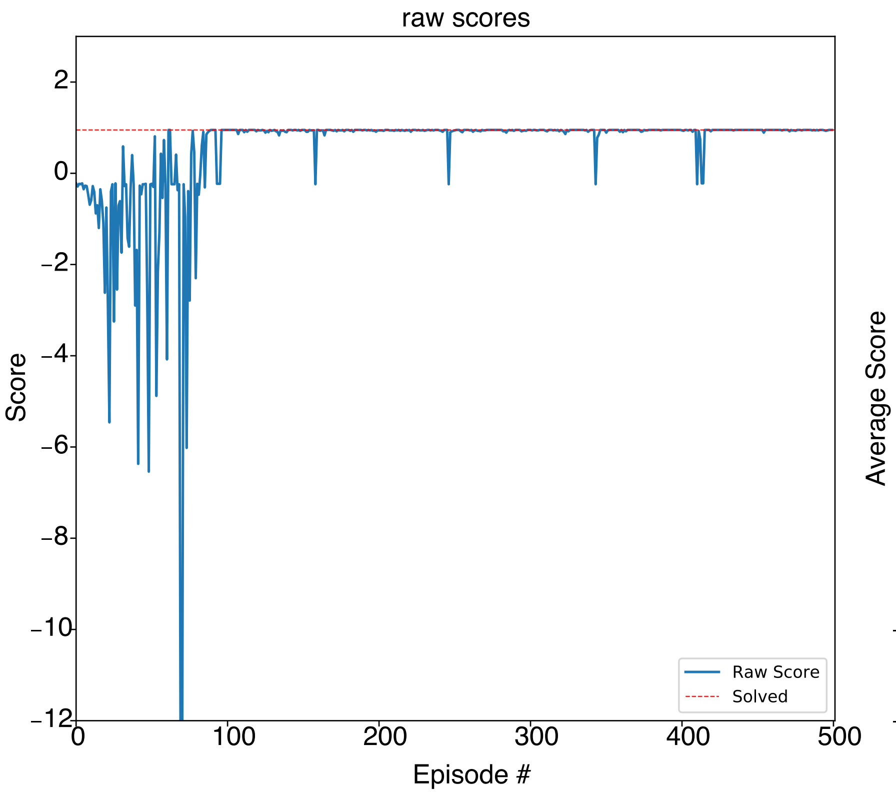


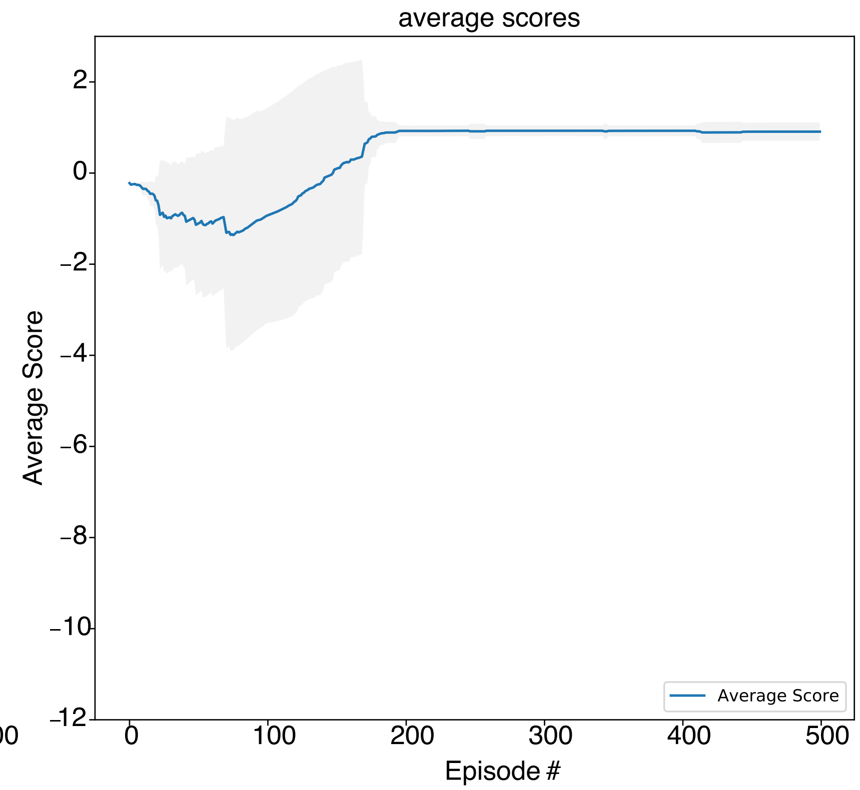




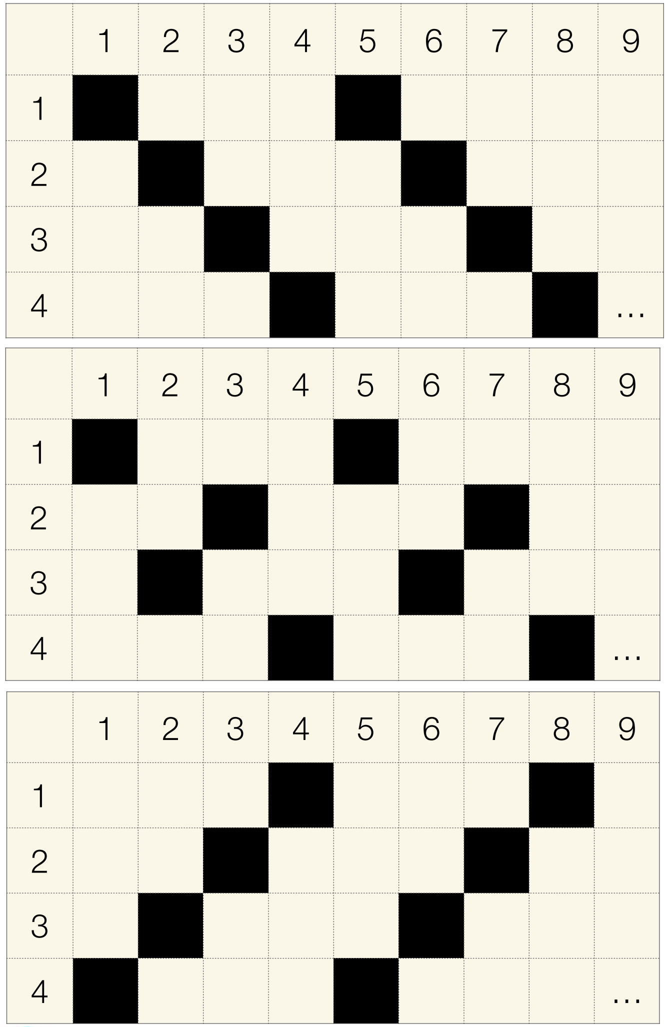
Brookhaven National Laboratory


S F F F
F F F H
F H F H
H F F G


raw scores
Score
Episode #
Raw Score
Solved
2
0
−2
−4
−6
−8
−10
−12
0
100
200
300
400
500


average scores
Average Score
Episode #
Average Score
2
0
−2
−4
−6
−8
−10
−12
0
100
200
300
400
500

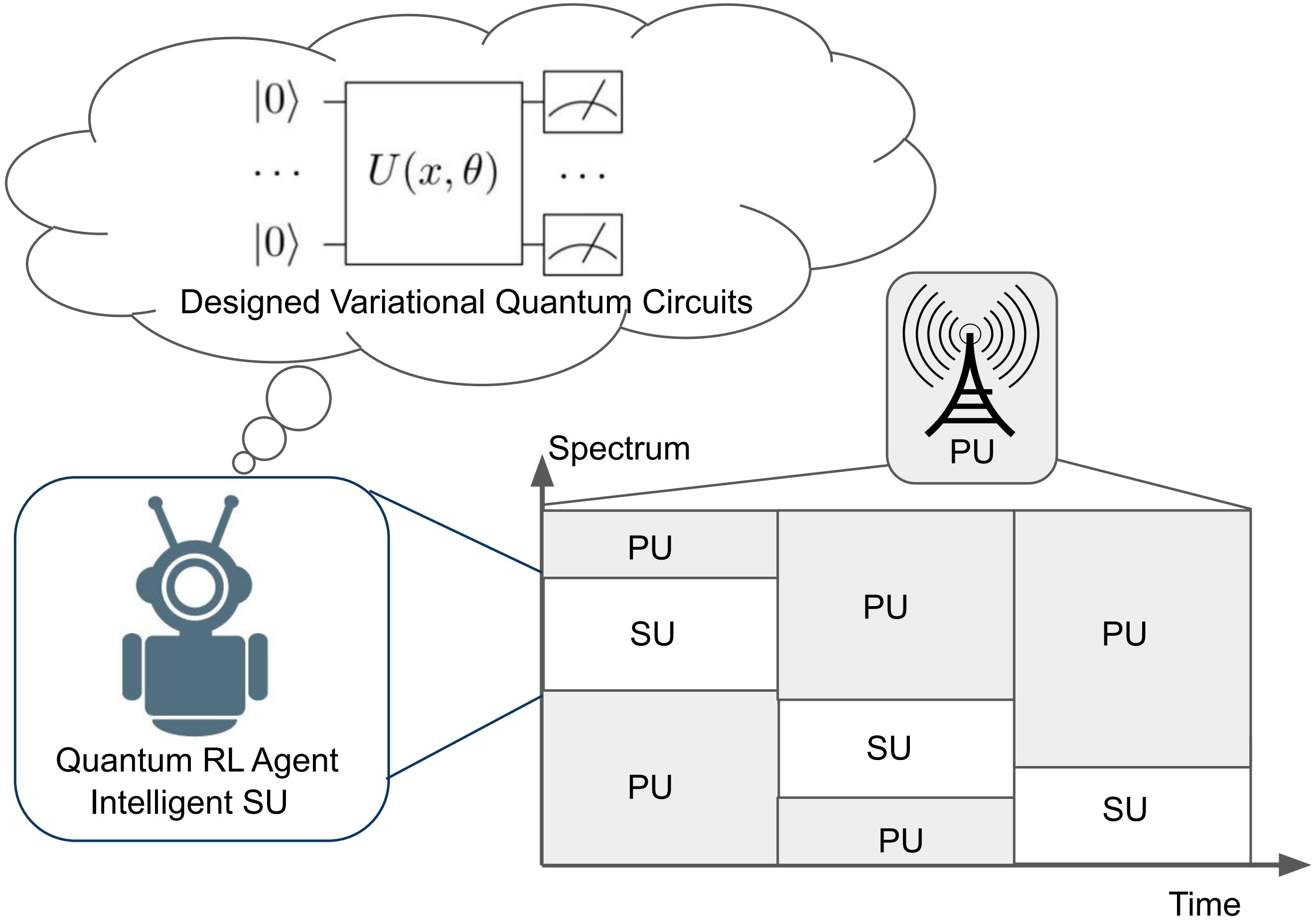
- Env: CognitiveRadio

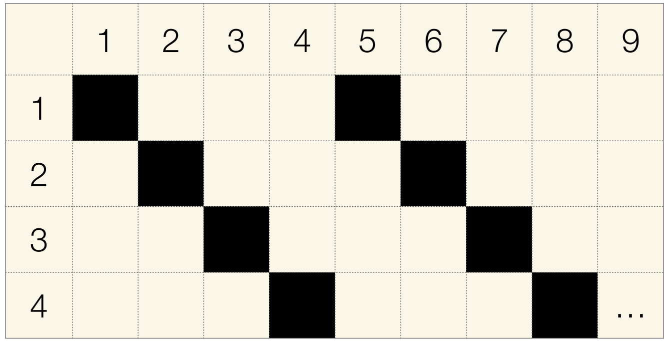


(a)

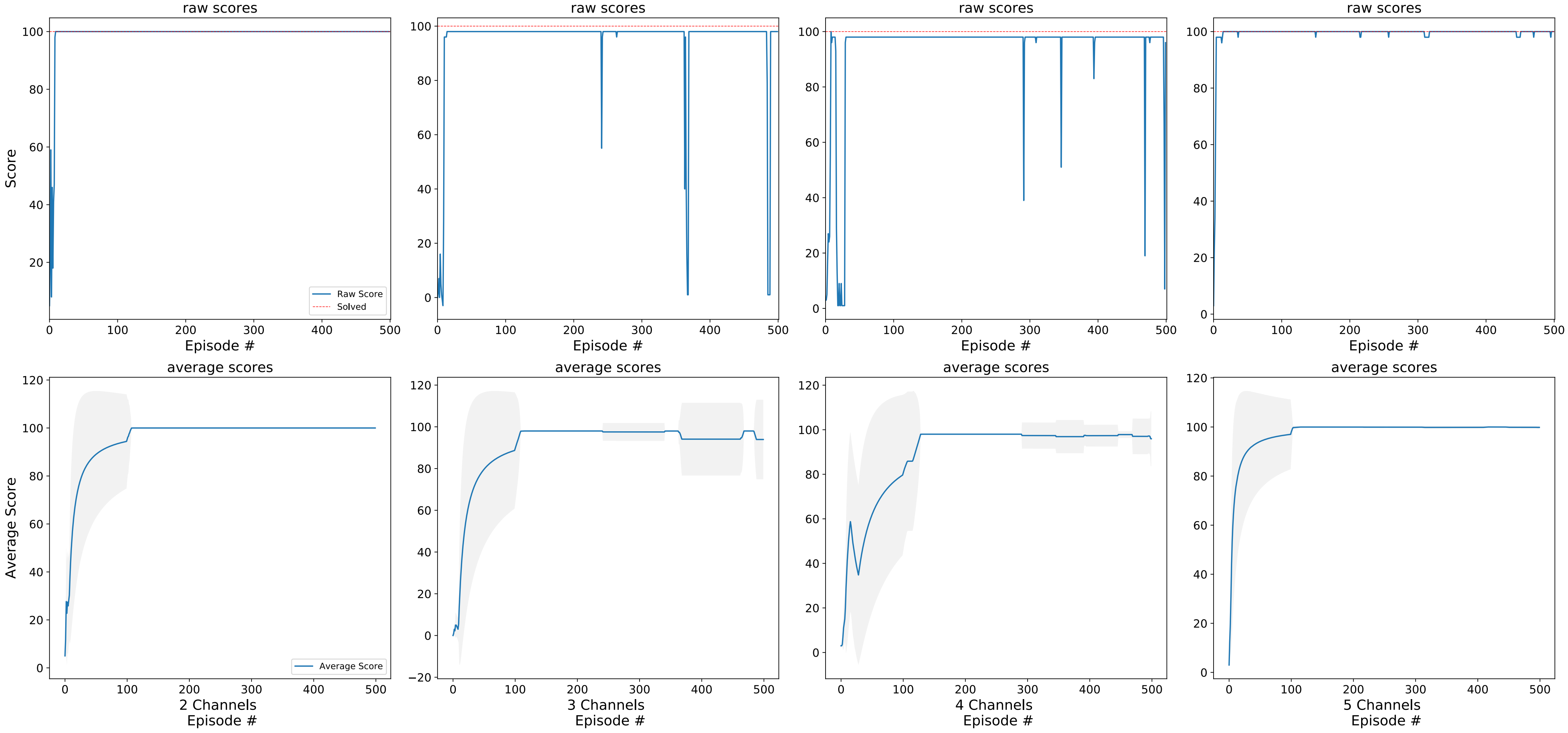
(b)

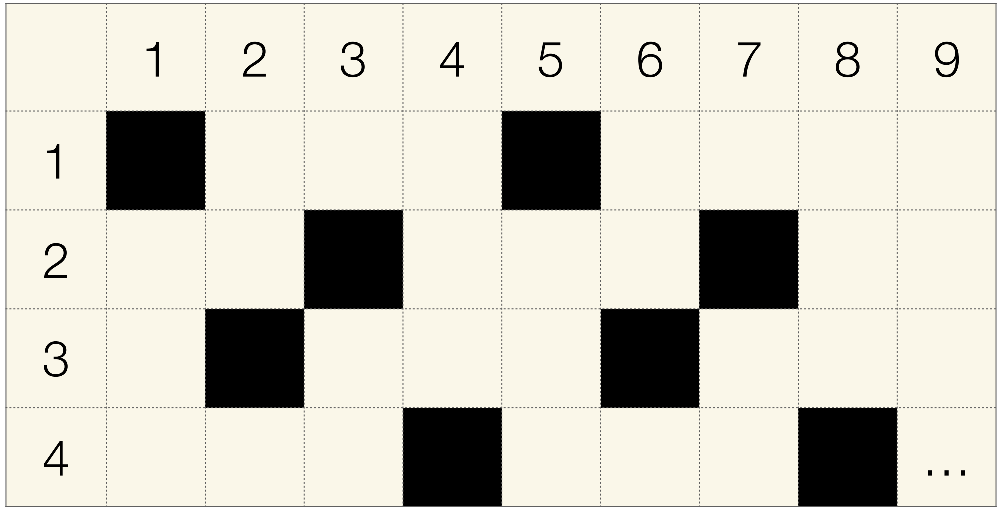
(c)

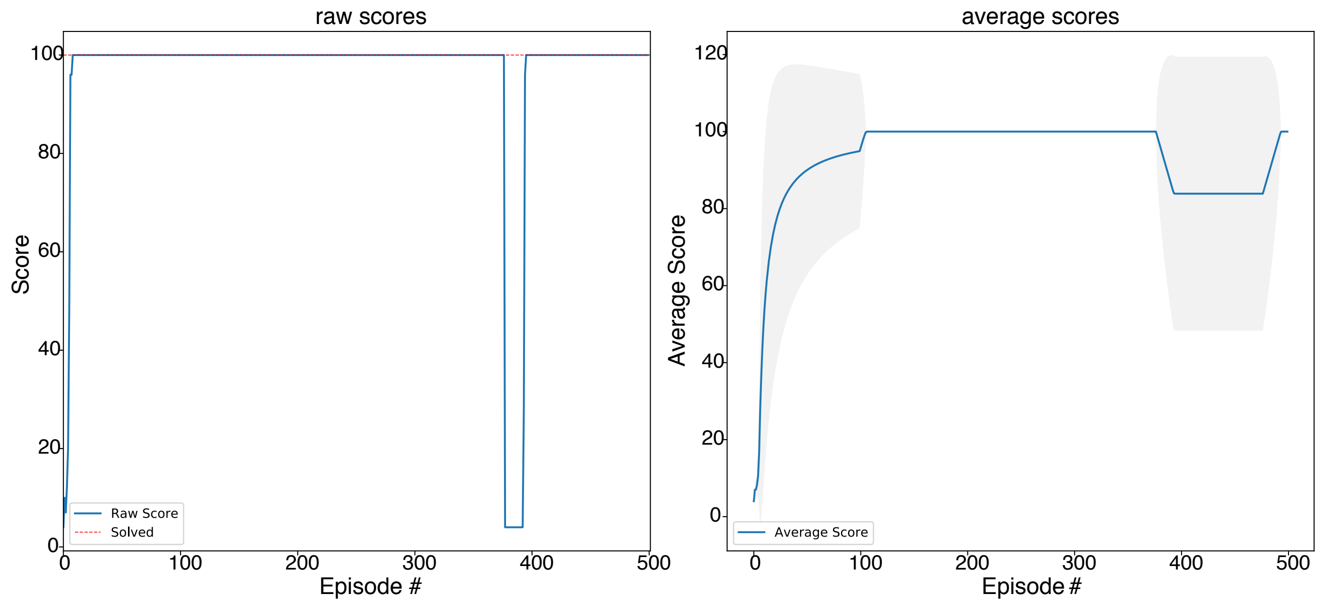



1
2
3
4
5
6
7
8
9
1
2
3
4
…

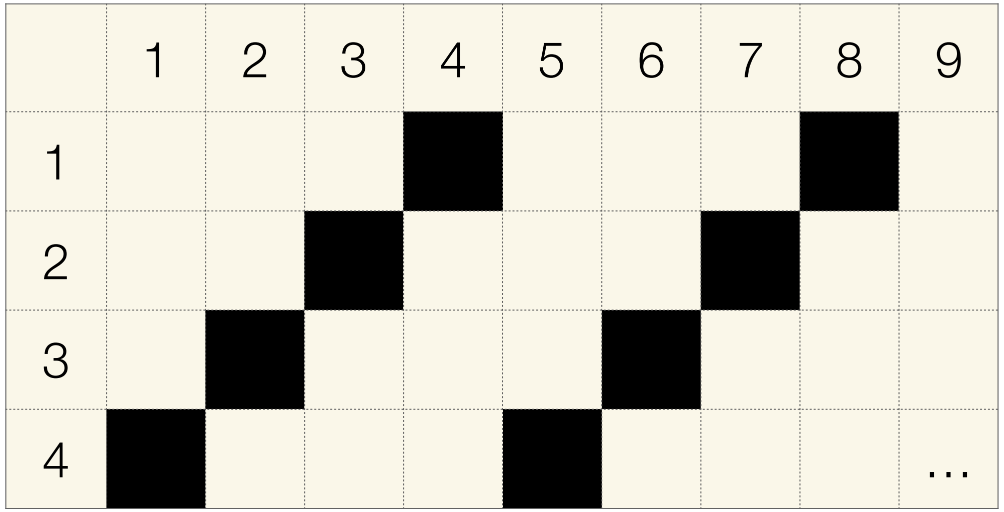
(a)

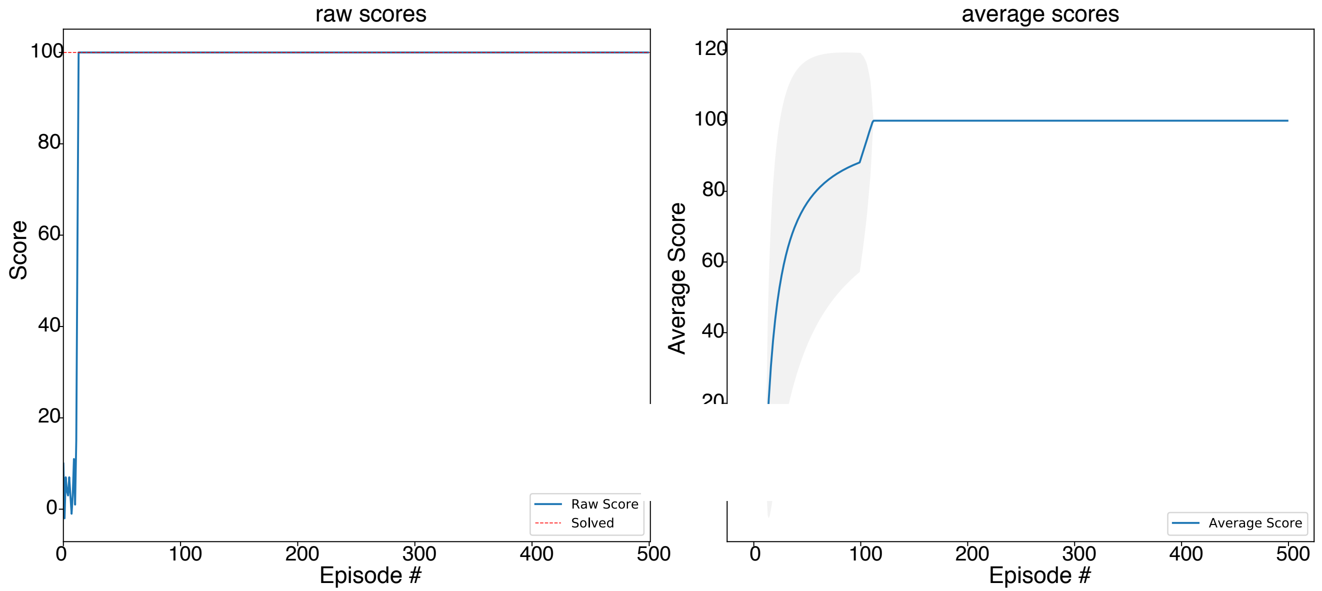

raw scores
Score
Episode #
Raw Score
Solved
average scores
Average Score
Average Score
2 Channels
Episode #
3 Channels
Episode #
4 Channels
Episode #
5 Channels
Episode #





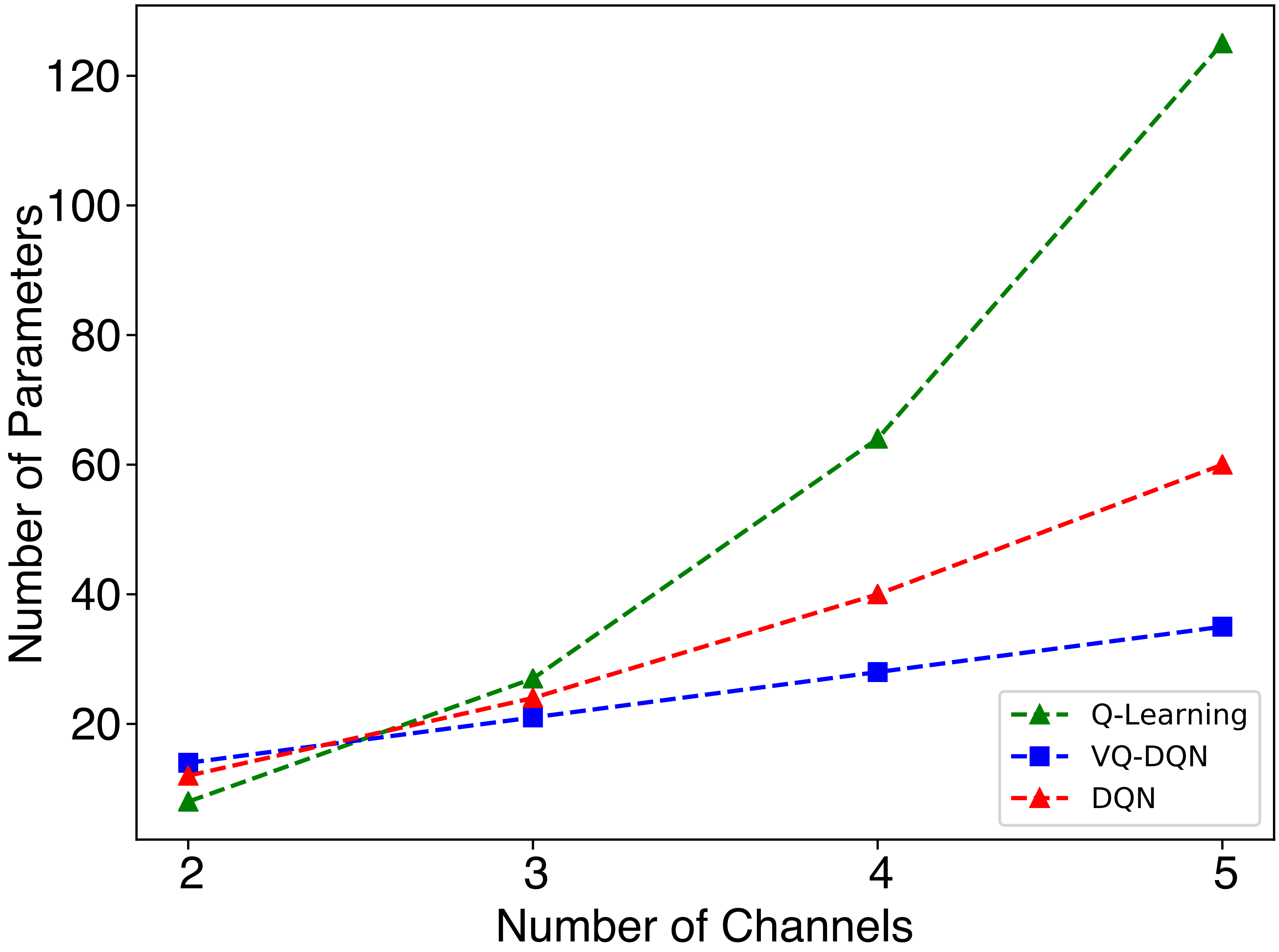
(b)





Brookhaven National Laboratory

(c)



Brookhaven National Laboratory

Evolutionary Quantum RL

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

Chen, S. Y. C., Huang, C. M., Hsing, C. W., Goan, H. S., & Kao, Y. J. (2022). Variational quantum reinforcement learning via evolutionary optimization. *Machine Learning: Science and Technology*, *3*(1), 015025.

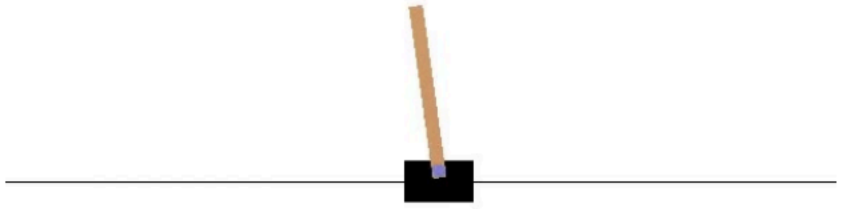
Evolutionary Optimization

- **Initialization:**

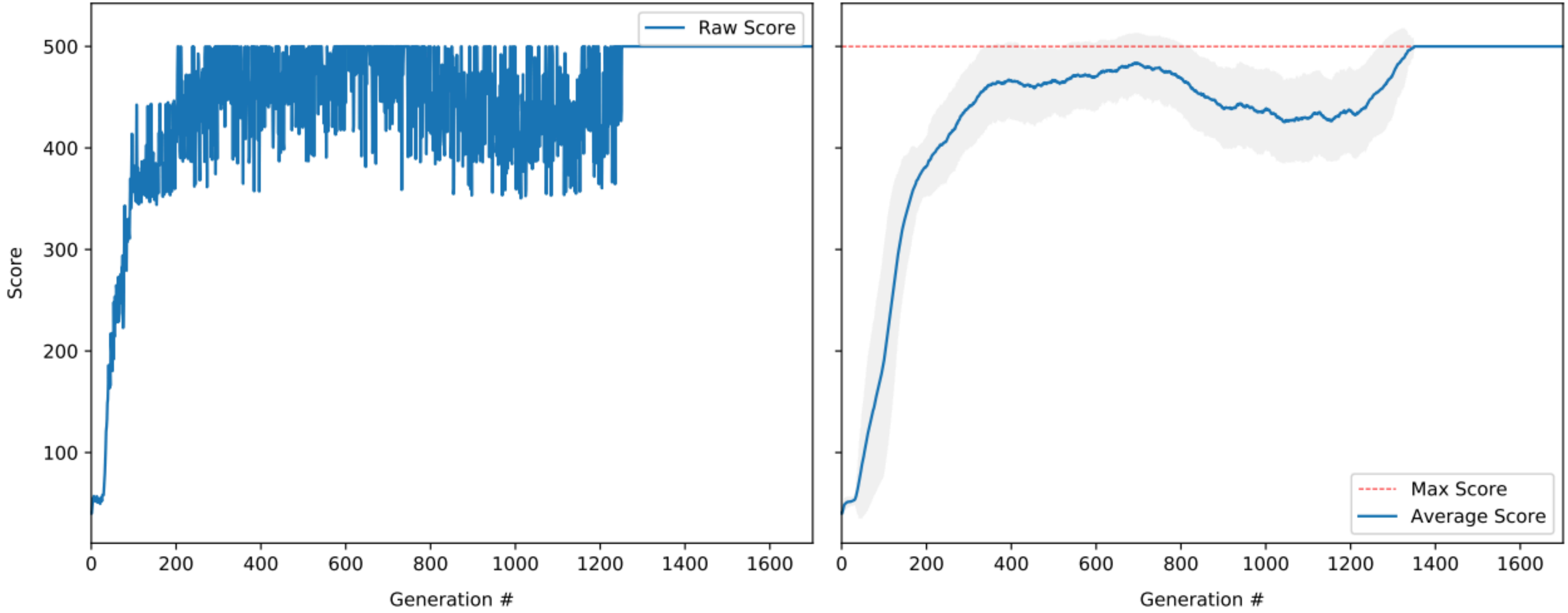
  Initialize the population $\mathscr{P}$ of N agents with each of them given randomly generated initial parameters θ, which are sampled from $\mathcal{N}(0,I)$
- **Running and evaluating the agents:**
  - Each agent plays the game R_1 times and get the average score $S_i^{avg} = \frac{1}{R_1}\sum_{r=1}^{R_1} S_{i,r}$
  - 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.



Brookhaven National Laboratory

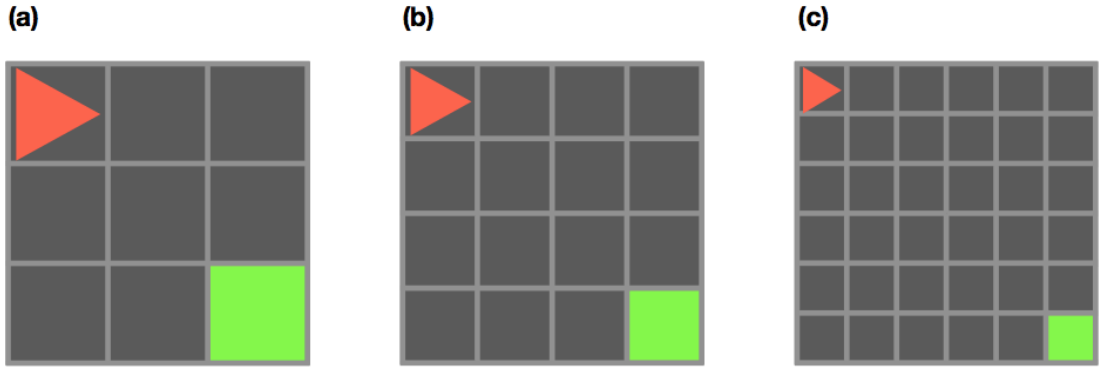


Chen, S. Y. C., Huang, C. M., Hsing, C. W., Goan, H. S., & Kao, Y. J. (2022). Variational quantum reinforcement learning via evolutionary optimization. *Machine Learning: Science and Technology*, *3*(1), 015025.

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:

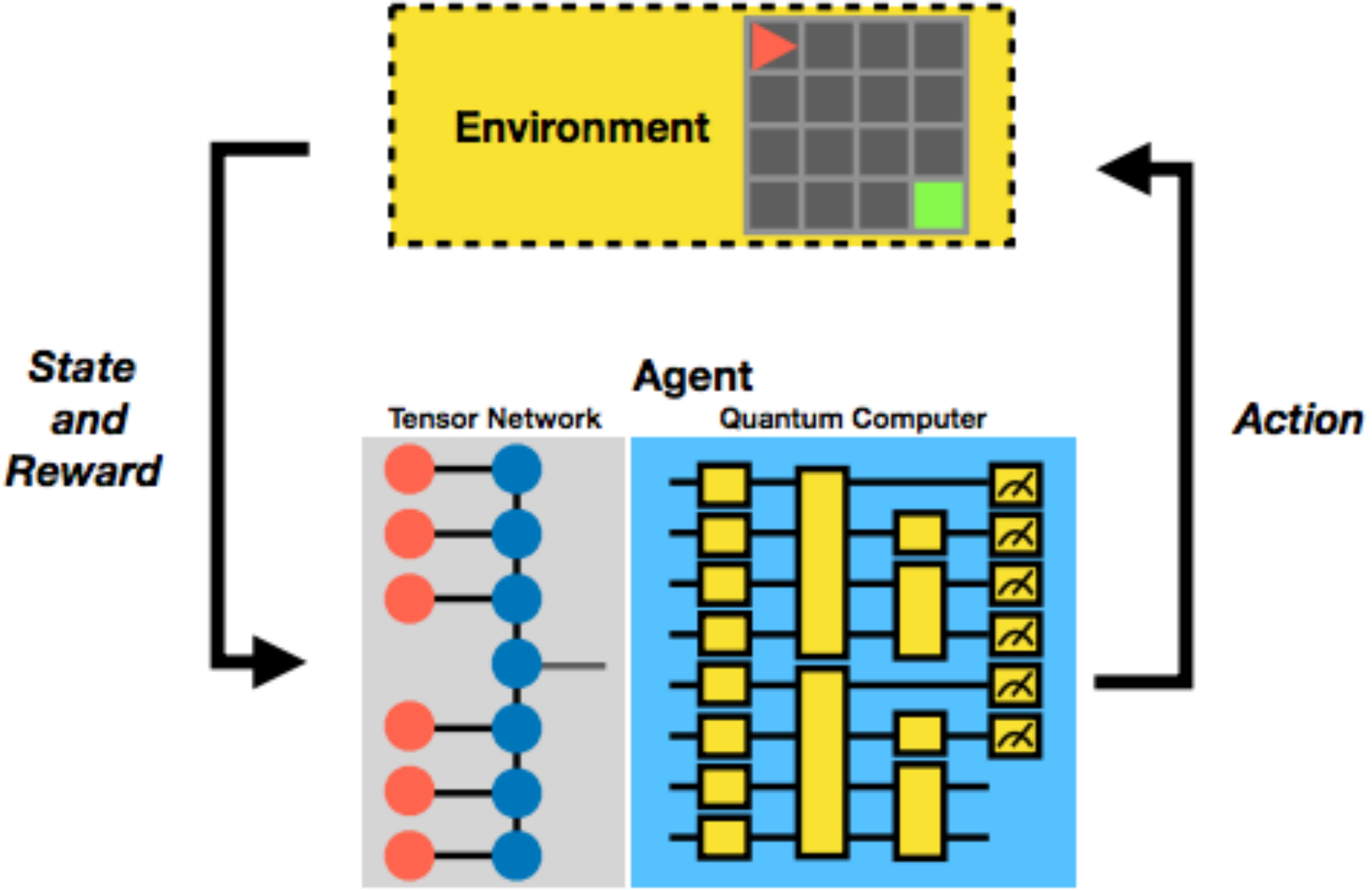
  1 − 0.9 × (*number of steps/max steps allowed*)



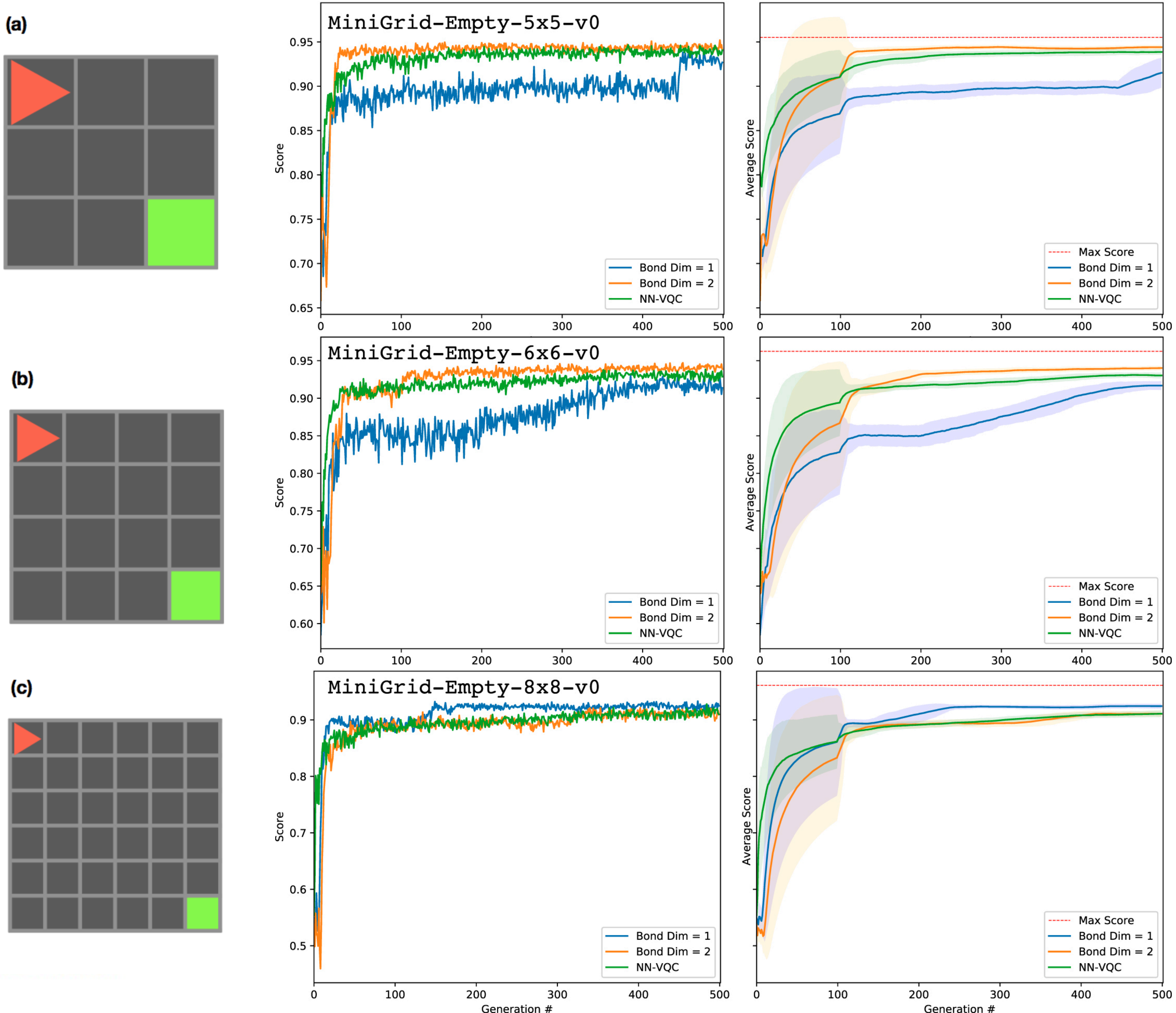
Brookhaven National Laboratory

Chen, S. Y. C., Huang, C. M., Hsing, C. W., Goan, H. S., & Kao, Y. J. (2022). Variational quantum reinforcement learning via evolutionary optimization. *Machine Learning: Science and Technology*, *3*(1), 015025.

Hybrid TN-VQC model



Chen, S. Y. C., Huang, C. M., Hsing, C. W., Goan, H. S., & Kao, Y. J. (2022). Variational quantum reinforcement learning via evolutionary optimization. *Machine Learning: Science and Technology*, *3*(1), 015025.



(a)
(b)
(c)
MiniGrid-Empty-5x5-v0
MiniGrid-Empty-6x6-v0
MiniGrid-Empty-8x8-v0
Score
Average Score
Generation #
Max Score
Bond Dim = 1
Bond Dim = 2
NN-VQC

Follow-up works

TITLE	CITED BY
Variational quantum circuits for deep reinforcement learning SYC Chen, CHH Yang, J Qi, PY Chen, X Ma, HS Goan IEEE Access 8, 141007-141024	70

- Lockwood, O., & Si, M. (2020, October). **Reinforcement learning with quantum variational circuit.** In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (Vol. 16, No. 1, pp. 245-251).

- Jerbi, S., Trenkwalder, L. M., Nautrup, H. P., Briegel, H. J., & Dunjko, V. (2021). **Quantum enhancements for deep reinforcement learning in large spaces.** PRX Quantum, 2(1), 010328.

- Wu, S., Jin, S., Wen, D., & Wang, X. (2020). **Quantum reinforcement learning in continuous action space.** arXiv preprint arXiv:2012.10711.

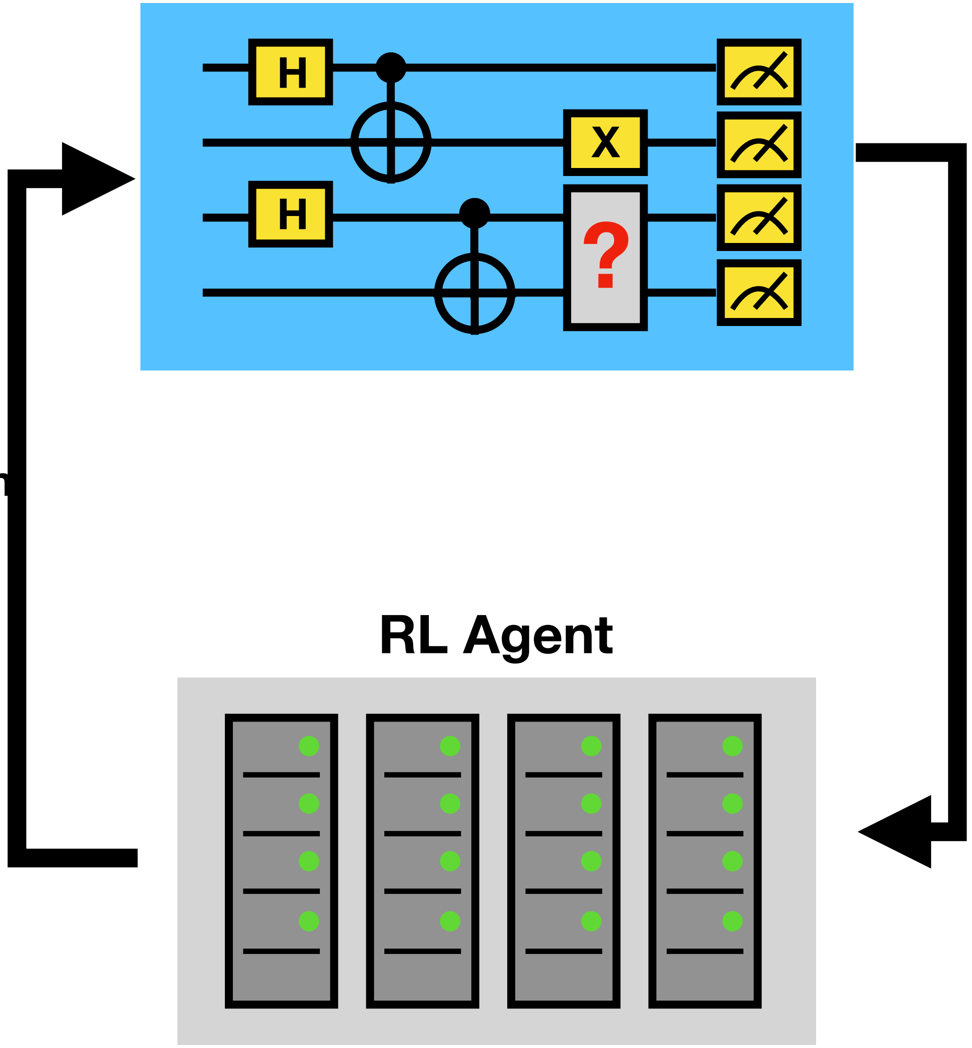
- Jerbi, S., Gyurik, C., Marshall, S., Briegel, H. J., & Dunjko, V. (2021). **Variational quantum policies for reinforcement learning.** arXiv preprint arXiv:2103.05577.

- Skolik, A., Jerbi, S., & Dunjko, V. (2021). **Quantum agents in the Gym: a variational quantum algorithm for deep Q-learning.** arXiv preprint arXiv:2103.15084.

I. Introduction
II. Reinforcement Learning (RL)
III. Quantum Computing (QC)
IV. Quantum RL
V. RL for QC
VI. Conclusion and Outlook

RL for Quantum Architecture Search (QAS)

Quantum Computer



RL Agent

Action:
Place a quantum gate on a wire.

Reward:
(a) -0.01 for each step.
(b) +0.99 when reaching fidelity > 0.99.

Observation:
Pauli-X,Y,Z expectation values.

Brookhaven National Laboratory

Kuo, E. J., Fang, Y. L. L., & Chen, S. Y. C. (2021). Quantum Architecture Search via Deep Reinforcement Learning. *arXiv preprint* ***arXiv:2104.07715***.

RL for QAS

- Goal:
  - Find the specific quantum circuit for a desired quantum states
  - Use as few gates as possible
- Algorithms:
  - Deep Q-learning
  - Policy Gradients

Policy Gradient

- Q-learning or deep Q-learning: **value-based RL->** learns the **value function** and use it as the reference to generate the decision on each time-step.

- Policy gradient -> the policy function $\pi(a \mid s; \theta)$ is parameterized with the parameters θ

- REINFORCE algorithm: parameters θ are updated along the direction $\nabla_\theta \log \pi \left(a_t \mid s_t; \theta\right) R_t$

- To reduce the variance, **baseline** function is introduced $\nabla_\theta \log \pi \left(a_t \mid s_t; \theta\right) \left(R_t - b_t\left(s_t\right)\right)$

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, *8*(3), 229-256.

Brookhaven National Laboratory

Advantage Actor-Critic (A2C)

- **Advantage** function $A\left(s_t, a_t\right)$ is defined to be $R_t - b_t = Q\left(s_t, a_t\right) - V\left(s_t\right)$
- *How good or bad* the action a_t compared to the average value at this state $V(s_t)$

Proximal Policy Optimization (PPO)

- Provide more stable policy gradient training through limiting the policy update step size at each training step.

- $q_t(\theta) = \dfrac{\pi\left(a_t \mid s_t; \theta\right)}{\pi\left(a_t \mid s_t; \theta_{\text{old}}\right)}$

- $L_{\text{policy}}(\theta) = \mathbb{E}_t\left[q_t(\theta) A_t\right] = \mathbb{E}_t\left[\dfrac{\pi\left(a_t \mid s_t; \theta\right)}{\pi\left(a_t \mid s_t; \theta_{\text{old}}\right)} A_t\right]$

- $L_{\text{policy}}(\theta) = \mathbb{E}_t\left[-\min\left(q_t A_t, \operatorname{clip}\left(q_t, 1 - C, 1 + C\right) A_t\right)\right]$

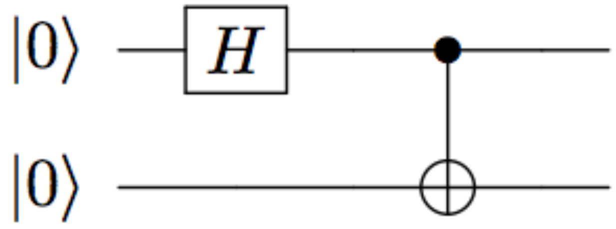
Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

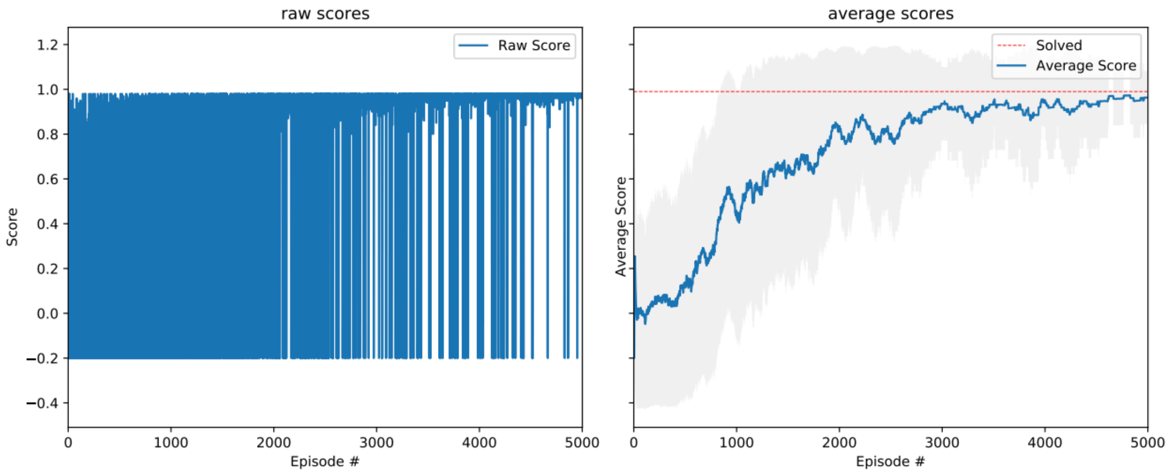
2-qubit Bell state

$$|\ \textbf{Bell}\ \rangle = \frac{|0\rangle^{\otimes 2} + |1\rangle^{\otimes 2}}{\sqrt{2}} = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

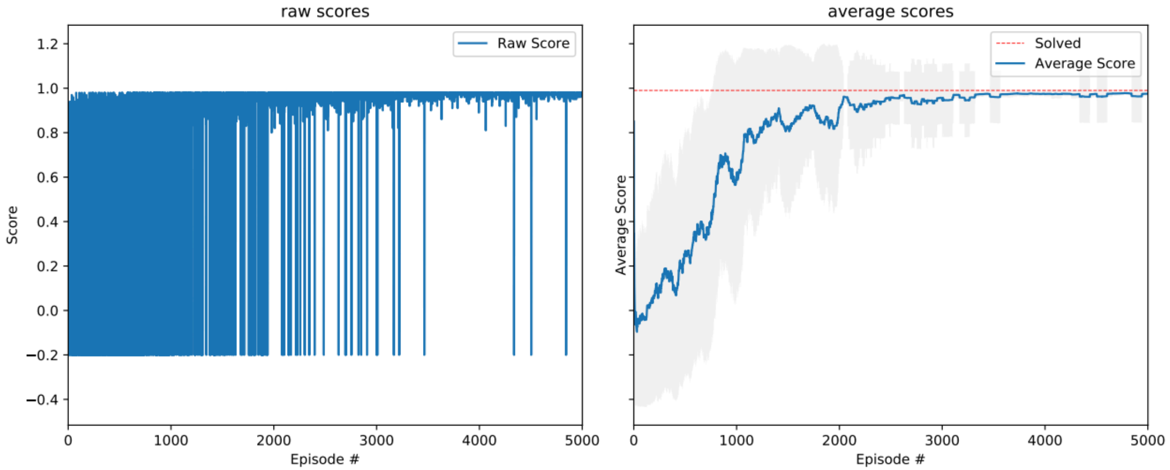
$$\mathbb{G} = \bigcup_{i=1}^{n} \left\{ U_i(\pi/4), X_i, Y_i, Z_i, H_i, CNOT_{i,(i+1)(mod 2)} \right\}$$

- For $n = 2$, there are 12 actions.





(a) **A2C for noise-free two-qubit system.**



(b) **PPO for noise-free two-qubit system.**

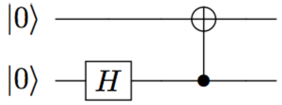


FIG. 4: **Quantum circuit for the Bell state generated by the DRL(PPO) agent.**

Brookhaven National Laboratory

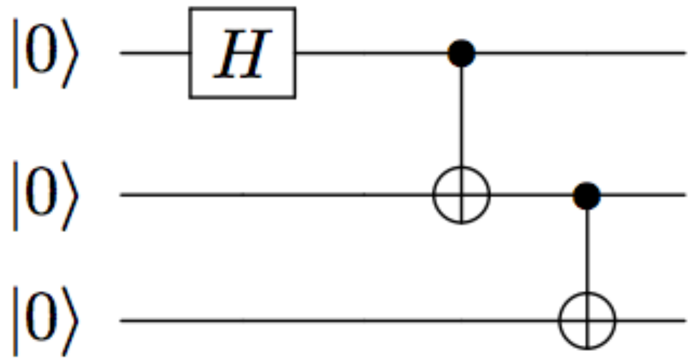
Kuo, E. J., Fang, Y. L. L., & Chen, S. Y. C. (2021). Quantum Architecture Search via Deep Reinforcement Learning. *arXiv preprint* ***arXiv:2104.07715***.

3-qubit GHZ state

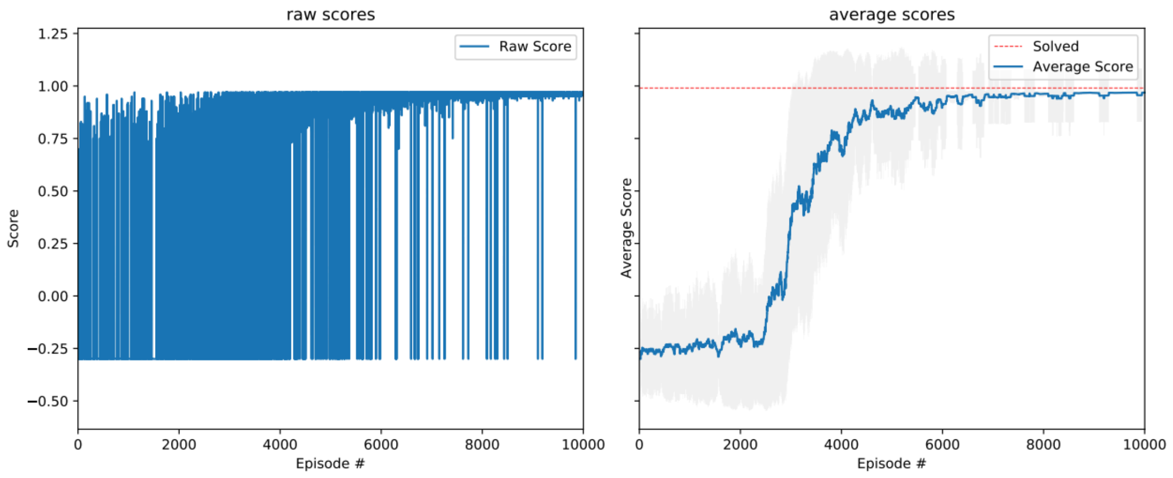
$$|\text{GHZ}\rangle = \frac{|0\rangle^{\otimes 3} + |1\rangle^{\otimes 3}}{\sqrt{2}} = \frac{|000\rangle + |111\rangle}{\sqrt{2}}$$

$$\mathbb{G} = \bigcup_{i=1}^{n} \left\{ U_i(\pi/4), X_i, Y_i, Z_i, H_i, CNOT_{i,(i+1)(mod2)} \right\}$$

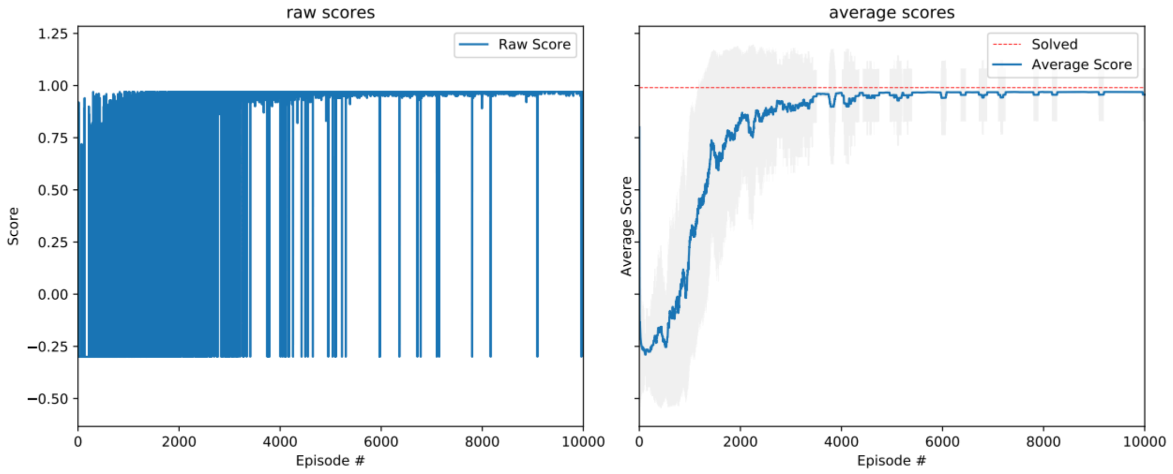
- For $n = 3$, there are 21 actions.



Brookhaven National Laboratory



(a) **A2C for noise-free three-qubit system.**



(b) **PPO for noise-free three-qubit system.**

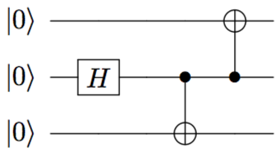


FIG. 6: **Quantum circuit for the GHZ state generated by the DRL(PPO) agent.**

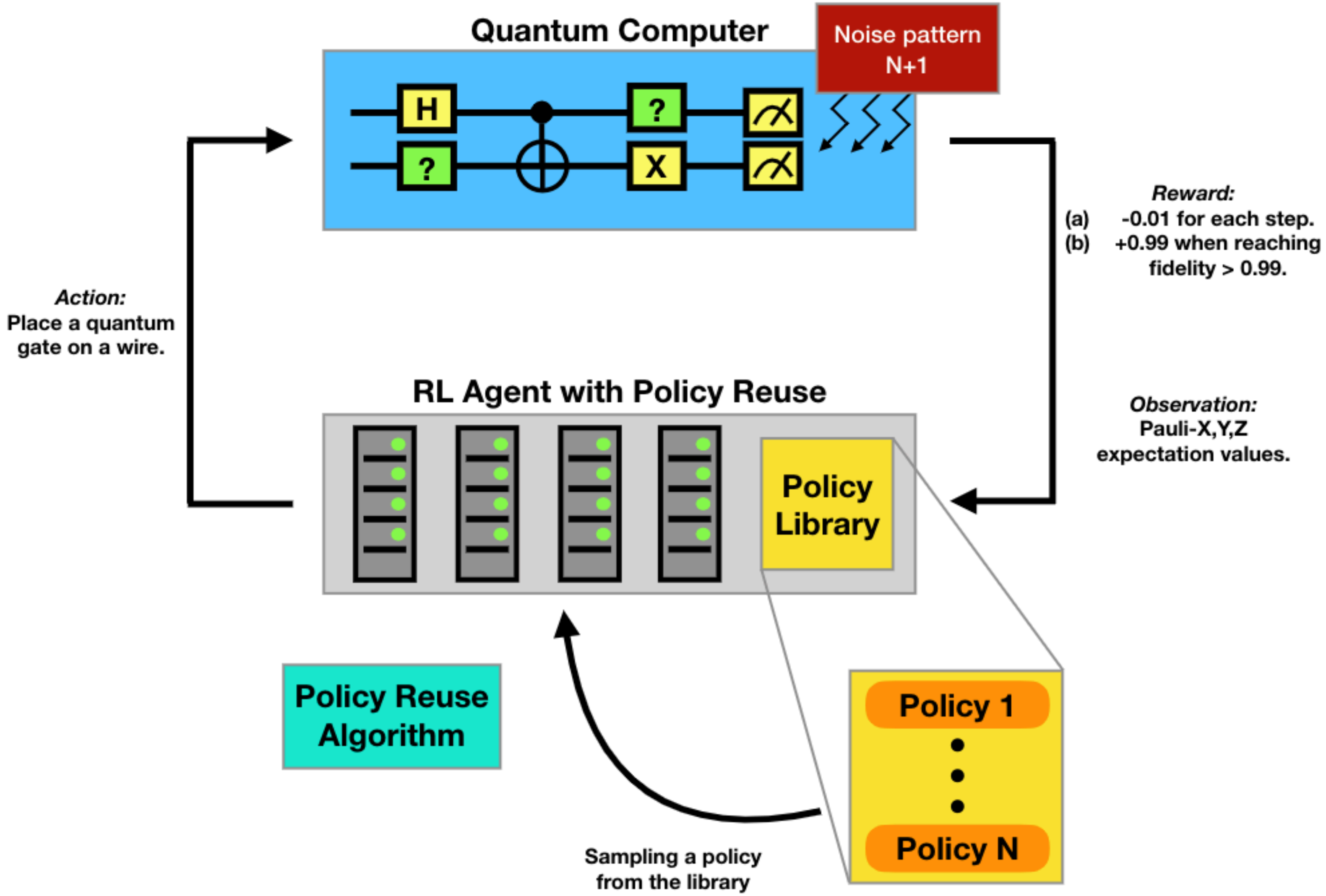
Brookhaven National Laboratory

Kuo, E. J., Fang, Y. L. L., & Chen, S. Y. C. (2021). Quantum Architecture Search via Deep Reinforcement Learning. *arXiv preprint* ***arXiv:2104.07715***.

Continual RL

- Noise pattern in the quantum device may change
- RL policies trained previously may not perform well with new environments.
- Training a new policy from scratch is computationally expensive and time-consuming.

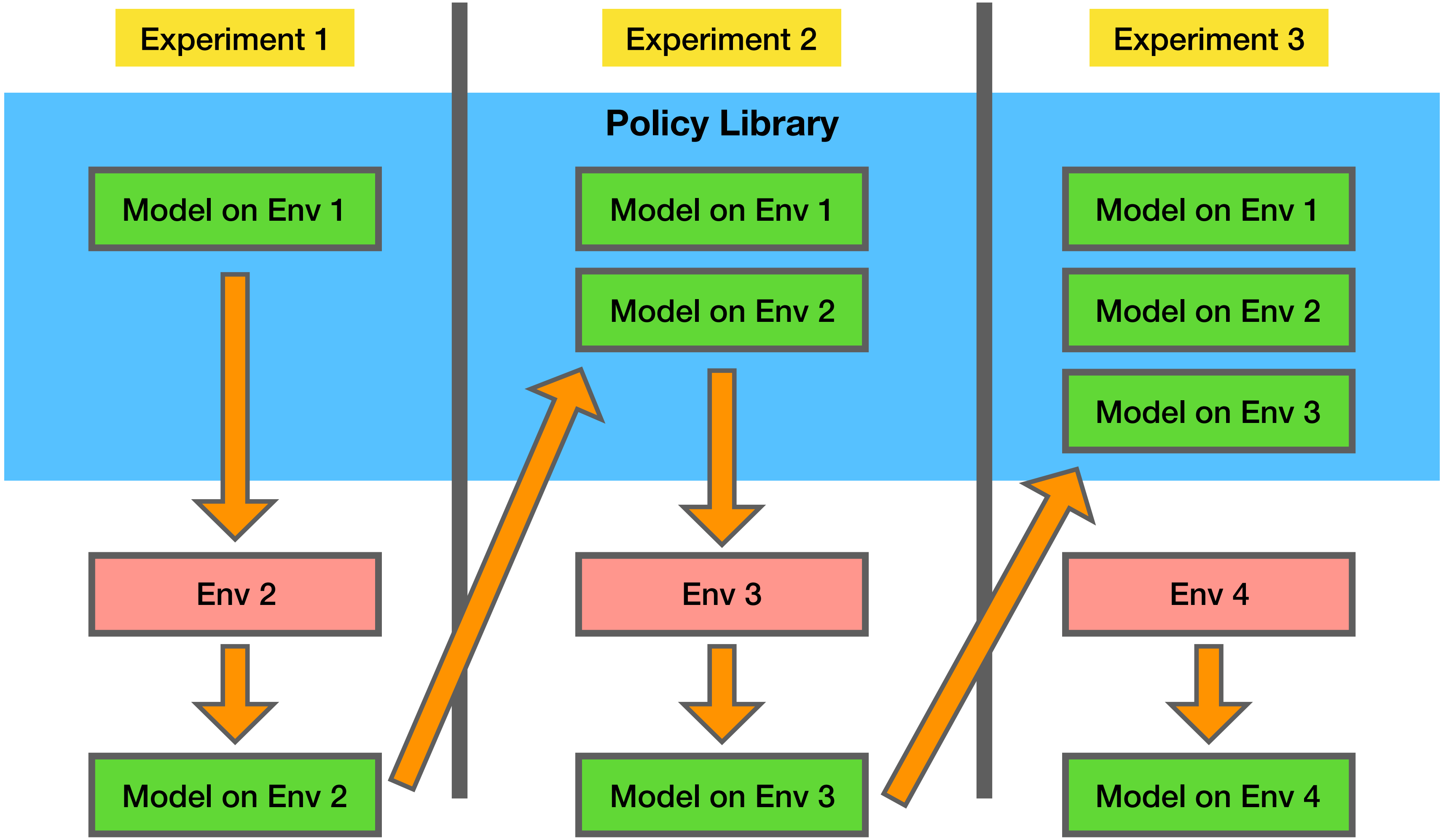
Continual RL for QAS



Brookhaven National Laboratory

Ye, E., & Chen, S. Y. C. (2021). Quantum Architecture Search via Continual Reinforcement Learning. *arXiv preprint* ***arXiv:2112.05779***.

Probabilistic Policy Reuse



Brookhaven National Laboratory

Ye, E., & Chen, S. Y. C. (2021). Quantum Architecture Search via Continual Reinforcement Learning. *arXiv preprint* ***arXiv:2112.05779***.

Probabilistic Policy Reuse

- Given:
    1. A new task Ω we want to solve
    2. A Policy Library $L = \{\Pi_1, \ldots, \Pi_n\}$
    3. An initial value of the temperature parameter, τ, and an incremental size, $\Delta\tau$, for the Boltzmann policy selection strategy
    4. A maximum number of episodes to execute, K
    5. A maximum number of steps per episode, H
    6. The parameters ψ and υ for the π-exploration strategy
    7. The parameters γ and α for the Q-learning update equation

Brookhaven National Laboratory

Fernández, F., & Veloso, M. (2006, May). Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems* (pp. 720-727).

Probabilistic Policy Reuse

- Initialize:

1. $Q_\Omega(s,a) = 0, \forall s \in \mathcal{S}, a \in \mathcal{A}$
2. Initialize W_Ω to 0
3. Initialize W_i to 0
4. Initialize the number of episodes where policy Π_Ω has been chosen, $U_\Omega = 0$
5. Initialize the number of episodes where policy Π_i has been chosen, $U_i = 0, \forall i = 1, \ldots, n$

Brookhaven National Laboratory

Fernández, F., & Veloso, M. (2006, May). Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems* (pp. 720-727).

Probabilistic Policy Reuse

- For $k = 1$ to K do
  - Choose an action policy, Π_k, assigning to each policy the probability of being selected computed by the following equation (equation 4):

$$P(\Pi_j) = \frac{e^{\tau W_j}}{\sum_{p=0}^{n} e^{\tau W_p}}$$

    where W_0 is set to W_Ω
  - Execute the learning episode k
    * If $\Pi_k = \Pi_\Omega$, execute a Q-Learning episode following a fully greedy strategy
    * Otherwise, use the π-reuse exploration strategy to reuse Π_k, i.e. call π-reuse$(\Pi_k, 1, H, \psi, \upsilon)$
    * In any case, receive the reward obtained in that episode, say R, and the updated Q function, $Q_\Omega(s, a)$
  - Set $W_k = \frac{W_k U_k + R}{U_k + 1}$
  - Set $U_k = U_k + 1$
  - Set $\tau = \tau + \Delta\tau$
- Return the policy derived from $Q_\Omega(s, a)$

Brookhaven National Laboratory

Fernández, F., & Veloso, M. (2006, May). Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems* (pp. 720-727).

Probabilistic Policy Reuse

π-reuse $(\Pi_{past}, K, H, \psi, \upsilon)$.

Initialize $Q^{\Pi_{new}}(s,a) = 0, \forall s \in \mathcal{S}, a \in \mathcal{A}$
For $k = 0$ to $K-1$
  Set the initial state, s, randomly.
  Set $\psi_1 \leftarrow \psi$
  for $h = 1$ to H
    With a probability of ψ_h, $a = \Pi_{past}(s)$
    With a probability of $1 - \psi_h$, $a = \epsilon\text{-greedy}(\Pi_{new}(s))$
    Receive the next state s', and reward, $r_{k,h}$
    Update $Q^{\Pi_{new}}(s,a)$, and therefore, Π_{new}:

$$Q^{\Pi_{new}}(s,a) \leftarrow (1-\alpha)Q(s,a)^{\Pi_{new}} + \alpha[r + \gamma \max_{a'} Q^{\Pi_{new}}(s',a')]$$

    Set $\psi_{h+1} \leftarrow \psi_h \upsilon$
    Set $s \leftarrow s'$

$W = \frac{1}{K}\sum_{k=0}^{K}\sum_{h=0}^{H} \gamma^h r_{k,h}$
Return W, $Q^{\Pi_{new}}(s,a)$ and Π_{new}

Brookhaven National Laboratory

Fernández, F., & Veloso, M. (2006, May). Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems* (pp. 720-727).

Probabilistic Policy Reuse with DQN

- Extend the Probabilistic Policy Reuse with deep neural networks
- Experience Replay and Target Networks

```
## --- Neural Net --- ##

class DQN(nn.Module):
    def __init__(self):
        super(DQN, self).__init__()        # self.mps = MPS(
        self.fc_1 = nn.Linear(OBSERVATION_DIM,HIDDEN_DIM)
        self.fc_2 = nn.Linear(HIDDEN_DIM,HIDDEN_DIM)
        self.fc_3 = nn.Linear(HIDDEN_DIM,ACTION_DIM)

    def forward(self, x):
        x = F.relu(self.fc_1(x))
        x = F.relu(self.fc_2(x))
        x = torch.tanh(self.fc_3(x))
        return x
```

Ye, E., & Chen, S. Y. C. (2021). Quantum Architecture Search via Continual Reinforcement Learning. *arXiv preprint* ***arXiv:2112.05779***.

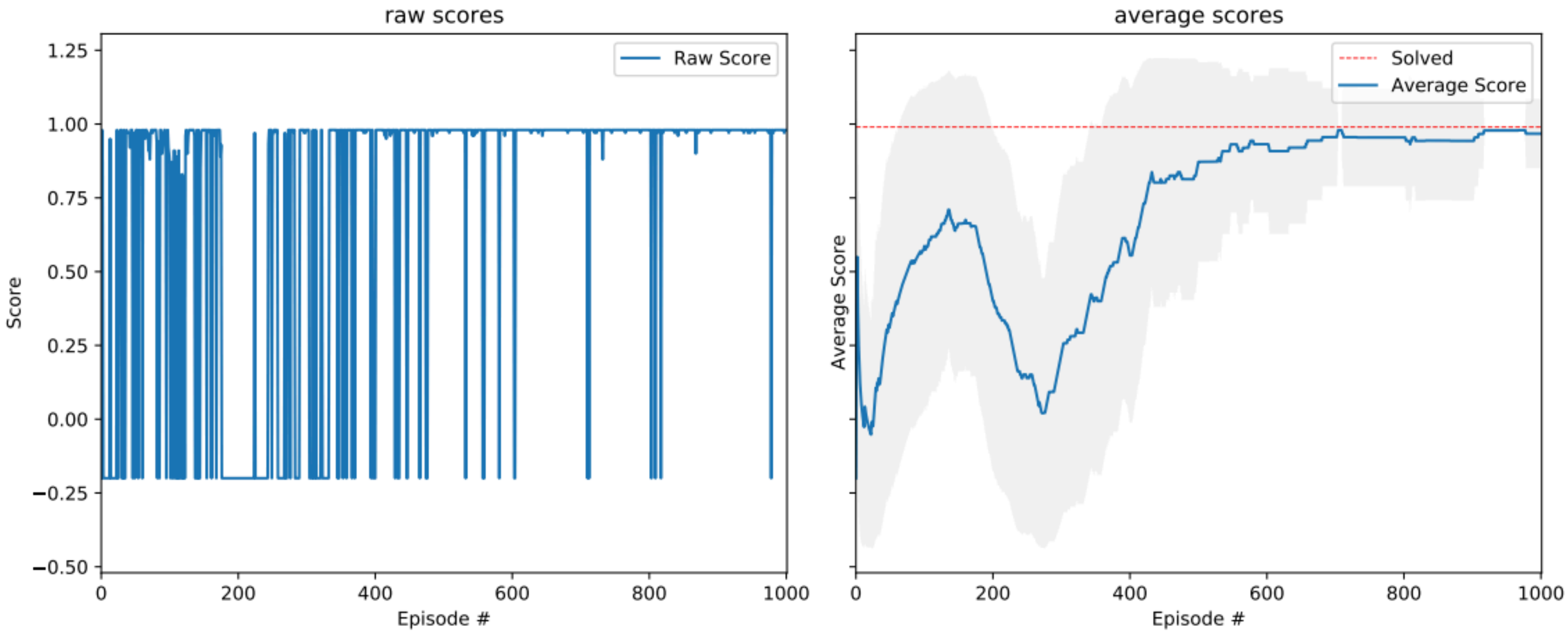
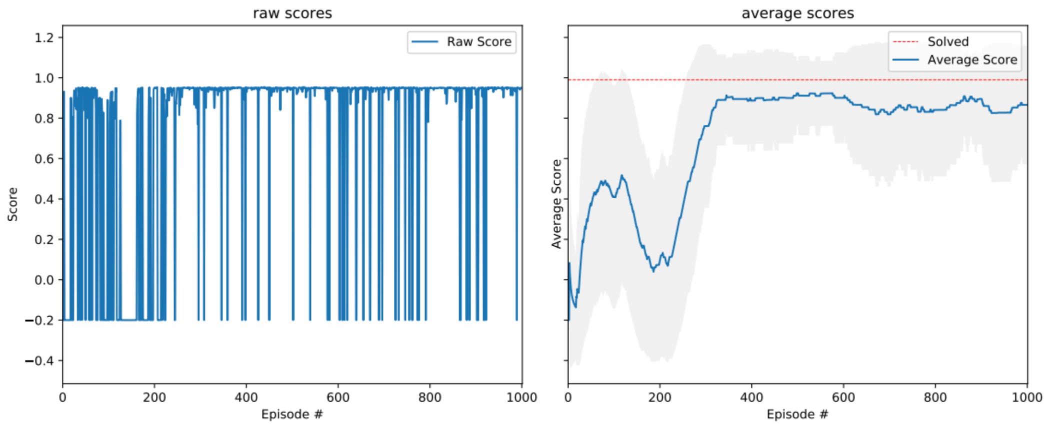


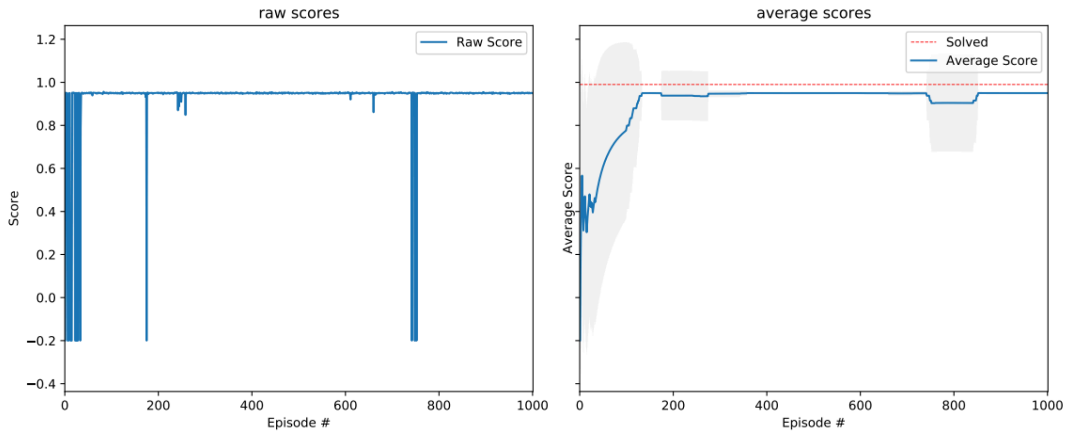
FIG. 3: Environment-0 (Noise-Free) training-from-scratch simulation

Brookhaven National Laboratory

Results: Noise on X gate



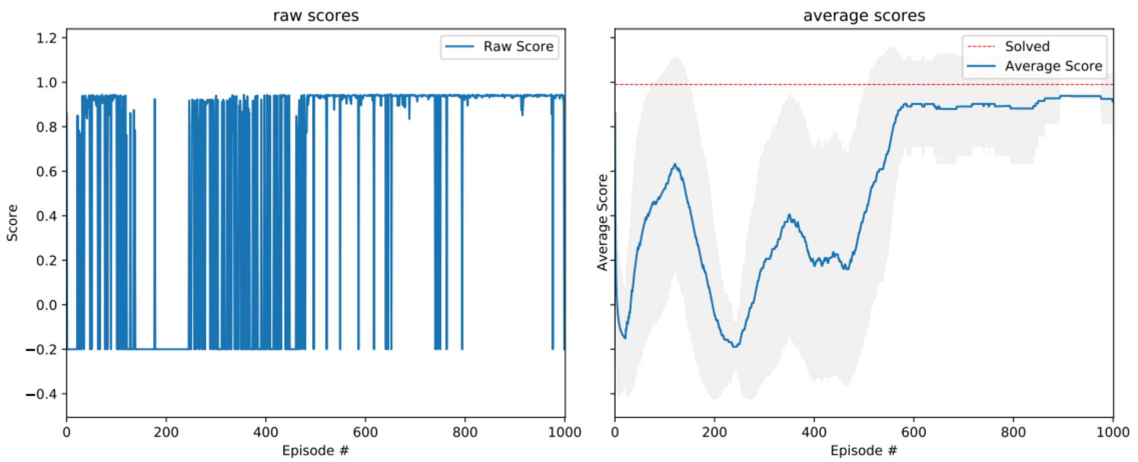
(a) **Training-from-scratch simulation for** Environment-1.



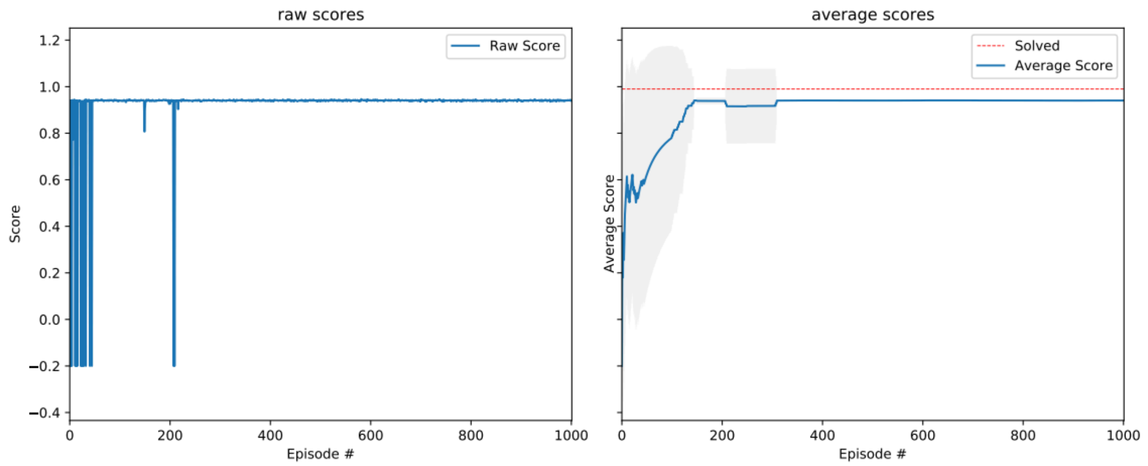
(b) **Policy reuse simulation result for** Environment-1.
Starting Policy Library: Simulation from ***Environment-0***.

Ye, E., & Chen, S. Y. C. (2021). Quantum Architecture Search via Continual Reinforcement Learning. *arXiv preprint* ***arXiv:2112.05779***.

Results: Noise on X, H gates



(a) Training-from-scratch simulation for `Environment-2`.



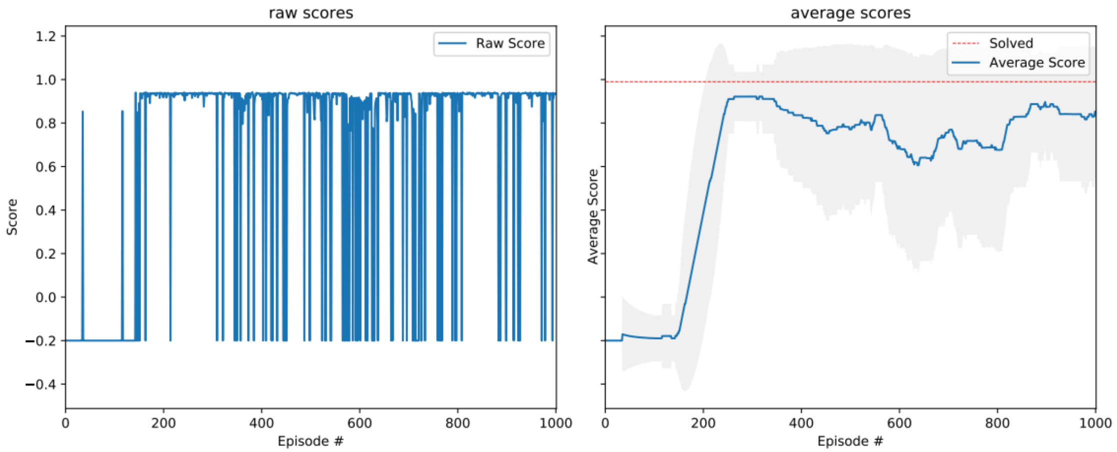
(b) Policy reuse simulation result for `Environment-2`.

*Starting Policy Library: from Scratch - **Environment-0**, Policy Reuse - **Environment-1**.*

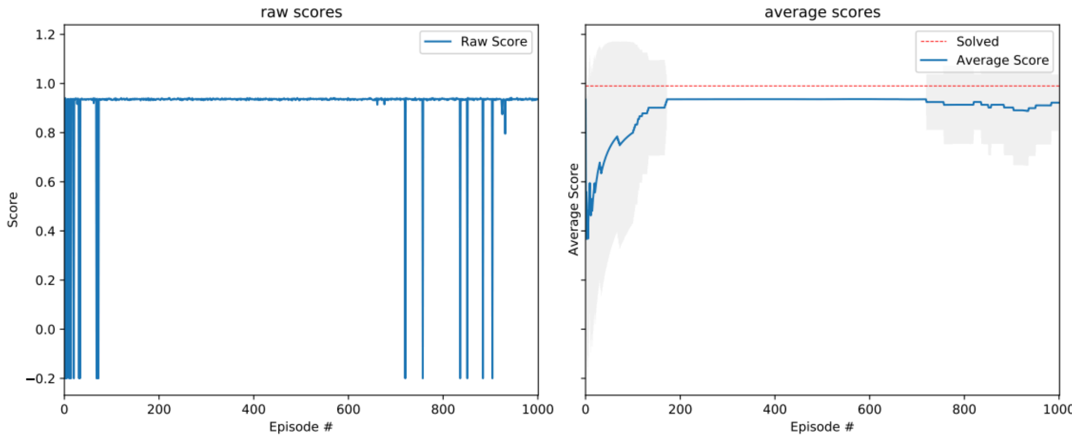
Brookhaven National Laboratory

Ye, E., & Chen, S. Y. C. (2021). Quantum Architecture Search via Continual Reinforcement Learning. *arXiv preprint* ***arXiv:2112.05779***.

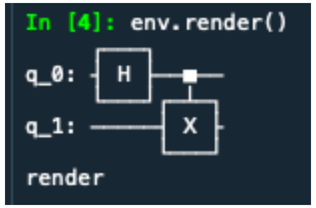
Results: Noise on X, CNOT gates



(a) Training-from-scratch simulation for `Environment-3`.



(b) Policy reuse simulation result for `Environment-3`.
Starting Policy Library: from Scratch - ***Environment-0****, Policy Reuse -* ***Environment-1****, Policy Reuse -* ***Environment-2****.*



Ye, E., & Chen, S. Y. C. (2021). Quantum Architecture Search via Continual Reinforcement Learning. *arXiv preprint* ***arXiv:2112.05779***.

I. Introduction
II. Reinforcement Learning (RL)
III. Quantum Computing (QC)
IV. Quantum RL
V. RL for QC
VI. Conclusion and Outlook

Brookhaven National Laboratory

- 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
- 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 environments.