# An Introduction to Quantum Reinforcement Learning (QRL)

Samuel Yen-Chi Chen
*Wells Fargo*
New York, NY, USA
yen-chi.chen@wellsfargo.com

*Abstract*—Recent advancements in quantum computing (QC) and machine learning (ML) have sparked considerable interest in the integration of these two cutting-edge fields. Among the various ML techniques, reinforcement learning (RL) stands out for its ability to address complex sequential decision-making problems. RL has already demonstrated substantial success in the classical ML community. Now, the emerging field of Quantum Reinforcement Learning (QRL) seeks to enhance RL algorithms by incorporating principles from quantum computing. This paper offers an introduction to this exciting area for the broader AI and ML community.

*Index Terms*—Quantum neural networks, Quantum machine learning, Variational quantum circuits, Quantum reinforcement learning, Quantum artificial intelligence

## I. INTRODUCTION

Quantum computing (QC) offers the potential for substantial computational advantages in specific problems compared to classical computers [1]. Despite the current limitations of quantum devices, such as noise and imperfections, significant efforts are being made to achieve quantum advantages. One prominent area of focus is quantum machine learning (QML), which leverages quantum computing principles to enhance machine learning tasks. Most QML algorithms rely on a hybrid quantum-classical paradigm, which divides the computational task into two components: quantum computers handle the parts that benefit from quantum computation, while classical computers process the parts they excel at.

Variational quantum algorithms (VQAs) [2] form the foundation of current quantum machine learning (QML) approaches. QML has demonstrated success in various machine learning tasks, including classification [3]–[6], sequential learning [7], [8], natural language processing [9]–[12], and reinforcement learning [13]–[19]. Among these areas, quantum reinforcement learning (QRL) is an emerging field where researchers are exploring the application of quantum computing principles to enhance the performance of reinforcement learning agents. This article provides an introduction to the concepts and recent developments in QRL.

## II. QUANTUM NEURAL NETWORKS

### A. Quantum Computing

A *qubit* represents the fundamental unit of quantum information processing. Unlike a classical *bit*, which is restricted to holding a state of either $0$ or $1$, a qubit can simultaneously encapsulate the information of both $0$ and $1$ due to the principle of superposition. A single qubit quantum state can be expressed as $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $|0\rangle = [1, 0]^T$ and $|1\rangle = [0, 1]^T$ are column vectors, and $\alpha$ and $\beta$ are complex numbers. In an $n$-qubit system, the state vector has a length of $2^n$. *Quantum gates* $U$ are utilized to transform a quantum state, represented as $|\Psi\rangle$, to another state $|\Psi'\rangle$ through the operation $|\Psi'\rangle = U|\Psi\rangle$. These quantum gates are *unitary transformations* that satisfy the condition $UU^\dagger = U^\dagger U = \mathbb{I}_{2^n \times 2^n}$, where $n$ denotes the number of qubits. It has been demonstrated that a small set of basic quantum gates is sufficient for universal quantum computation. One such set includes single-qubit gates $H$, $\sigma_x$, $\sigma_y$, $\sigma_z$, $R_x(\theta) = e^{-i\theta\sigma_x/2}$, $R_y(\theta) = e^{-i\theta\sigma_y/2}$, $R_z(\theta) = e^{-i\theta\sigma_z/2}$, and the two-qubit gate CNOT. In quantum machine learning (QML), rotation gates $R_x$, $R_y$, and $R_z$ are particularly crucial as their rotation angles can be treated as trainable or learnable parameters subject to optimization. For quantum operations on multi-qubit systems, the unitary transformation can be constructed via the tensor product of individual single-qubit or two-qubit operations, $U = U_1 \otimes U_2 \otimes \cdots \otimes U_k$. At the final stage of a quantum circuit, a procedure known as *measurement* is performed. A single execution of a quantum circuit generates a binary string. This procedure can be repeated multiple times to determine the probabilities of different computational bases (e.g., $|0, \cdots, 0\rangle$, $\cdots$, $|1, \cdots, 1\rangle$) or to calculate expectation values (e.g., Pauli $X$, $Y$, and $Z$).

### B. Variational Quantum Circuits

*Variational quantum circuits* (VQCs), also referred to as *parameterized quantum circuits* (PQCs), represent a specialized class of quantum circuits with trainable parameters. VQCs are extensively utilized within the current hybrid quantum-classical computing framework [2] and have demonstrated specific types of quantum advantages [20]–[22]. There are three fundamental components in a VQC: *encoding circuit*, *variational circuit* and the final *measurements*. As shown in Figure 1, the encoding circuit $U(\mathbf{x})$ transforms the initial

quantum state $|0\rangle^{\otimes n}$ into $|\Psi\rangle = U(\mathbf{x})|0\rangle^{\otimes n}$. Here the $n$ represents the number of qubits, $|0\rangle^{\otimes n}$ represents the $n$-qubit initial state $|0,\cdots,0\rangle$ and the $U(\mathbf{x})$ represents the unitary which depends on the input value $\mathbf{x}$. The measurement process extracts data from the VQC by assessing either a subset or all of the qubits, producing a classical bit sequence for further use. Running the circuit once yields a bit sequence such as "0,0,1,1." However, preparing and executing the circuit multiple times (shots) generates expectation values for each qubit. Most works mentioned in this survey focus on the evaluation of Pauli-$Z$ expectation values derived from measurements in VQCs. Generally, the mathematical expression of the VQC can be expressed as $\overrightarrow{f(\mathbf{x};\Theta)} = \left( \left\langle \hat{Z}_1 \right\rangle, \cdots, \left\langle \hat{Z}_n \right\rangle \right)$, where $\left\langle \hat{Z}_k \right\rangle = \left\langle 0 \left| U^\dagger(\mathbf{x}) W^\dagger(\Theta) \hat{Z}_k W(\Theta) U(\mathbf{x}) \right| 0 \right\rangle$. In the hybrid quantum-classical framework, the VQC can be integrated with other classical components, such as deep neural networks and tensor networks, or with other quantum components, including additional VQCs. The entire model can be optimized in an end-to-end manner using either gradient-based [4], [5] or gradient-free [14] methods. For gradient-based methods like gradient descent, the gradients of quantum components can be computed via the parameter-shift rules [3], [23], [24].
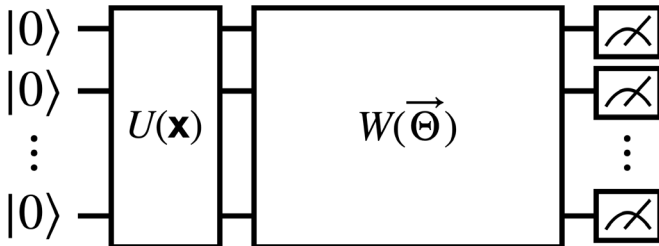


Fig. 1. **Generic Structure of a Variational Quantum Circuit (VQC).**

## III. QUANTUM REINFORCEMENT LEARNING

### A. Reinforcement Learning

*Reinforcement Learning* (RL) is a pivotal paradigm within machine learning, where an autonomous entity known as the *agent* learns to make decisions through iterative interactions with its environment [25]. The agent operates within a defined *environment*, represented as $\mathcal{E}$, over discrete time steps. At each time step $t$, the agent receives *state* or *observation* information, denoted as $s_t$, from the environment $\mathcal{E}$. Based on this information, the agent selects an *action* $a_t$ from a set of permissible actions $\mathcal{A}$, guided by its *policy* $\pi$. The policy $\pi$ acts as a function that maps the current state or observation $s_t$ to the corresponding action $a_t$. Notably, the policy can be stochastic, indicating that for a given state $s_t$, the action $a_t$ is determined by a probability distribution $\pi(a_t|s_t)$.

Upon executing action $a_t$, the agent transitions to the subsequent state $s_{t+1}$ and receives a scalar *reward* $r_t$. This cycle continues until the agent reaches a terminal state or fulfills a specified stopping condition, such as a maximum number of steps. We define an *episode* as the sequence beginning from an initial state, following the described process, and concluding either at the terminal state or upon meeting the

stopping criterion. The use of quantum neural networks for learning policy or value functions is referred to as quantum reinforcement learning (QRL). The idea of QRL is illustrated in Figure 2. For a comprehensive review of current QRL domain, refer to the review article [18].
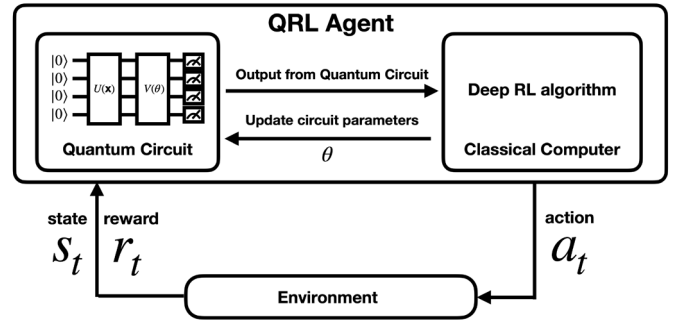


Fig. 2. **Concept of quantum reinforcement learning (QRL).**

### B. Quantum Deep Q-learning

$Q$-learning [25] is a fundamental model-free RL algorithm. It learns the optimal action-value function and operates *off-policy*. The process begins with the random initialization of $Q^\pi(s,a)$ for all states $s \in S$ and actions $a \in \mathcal{A}$, stored in a $Q$-table. The $Q^\pi(s,a)$ estimates are updated using the Bellman equation:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha \left[ r_t + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t) \right]. \quad (1)$$

The conventional $Q$-learning approach offers the optimal action-value function but is impractical for problems requiring extensive memory, especially with high-dimensional state ($s$) or action ($a$) spaces. In environments with continuous states, storing $Q(s,a)$ in a table is inefficient or impossible. To address this challenge, neural networks (NNs) are used to represent $Q^\pi(s,a), \forall s \in S, a \in \mathcal{A}$, leading to *deep Q-learning*. The network in this technique is known as a deep $Q$-network (DQN) [26]. To enhance the stability of DQN, techniques such as *experience replay* and the use of a *target network* are employed [26]. Experience replay stores *experiences* as transition tuples $s_t, a_t, r_t, s_{t+1}$ in a memory or buffer. After gathering sufficient experiences, the agent randomly samples a batch to compute the loss and update DQN parameters. Additionally, to reduce correlation between target and prediction, a *target network*, which is a duplicate of the DQN, is used. The DQN parameters $\theta$ are updated iteratively, while the target network parameters $\theta^-$ are updated periodically. The DQN training is done via minimizing the mean square error (MSE) loss function:

$$L(\theta) = \mathbb{E} \left[ (r_t + \gamma \max_{a'} Q(s_{t+1},a';\theta^-) - Q(s_t,a_t;\theta))^2 \right] \quad (2)$$

Other loss functions such as Huber loss or mean absolute error (MAE) can also be used. The first VQC-based QRL is described in the work [13] in which a VQC is designed to solve environments with discrete observations such as the

Frozen Lake and Cognitive-Radio. The design follows the original idea in classical deep $Q$-learning [26]. As shown in Figure 3, there are target network and experience replay in this quantum DQN. They are basically two sets of quantum circuit parameters. The quantum agent is optimized via gradient descent algorithm such as RMSProp in the hybrid quantum-classical manner. Later, more sophisticated efforts in the area of quantum DQN take into account continuous observation spaces like Cart-Pole [15], [16].
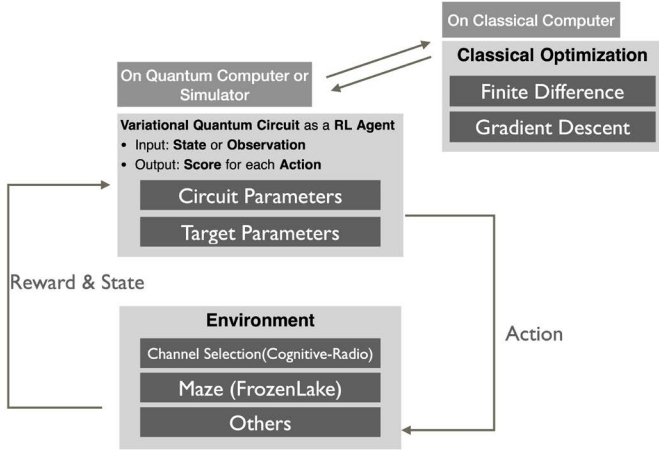


Fig. 3. **Quantum deep Q-learning.**

### C. Quantum Policy Gradient Methods

In contrast to *value-based* RL algorithms, such as $Q$-learning, which depend on learning a value function to guide decision-making at each time step, *policy gradient* methods aim to optimize a policy function directly. This policy function $\pi(a|s;\theta)$ is parameterized by $\theta$. The parameters $\theta$ are adjusted using gradient ascent on the expected total return, $\mathbb{E}[R_t]$. A prominent example of a policy gradient algorithm is the REINFORCE algorithm [27]. The policy function $\pi(a|s;\theta)$ can be implemented using a VQC, where the rotation parameters serve as $\theta$. In [28], the authors employ the REINFORCE algorithm to train a VQC-based policy. Their results demonstrate that VQC-based policies can achieve performance comparable to or exceeding that of classical DNNs on several standard benchmarks. In the traditional REINFORCE algorithm, parameter updates for $\theta$ are based on the gradient $\nabla_\theta \log \pi (a_t|s_t;\theta) R_t$, which provides an unbiased estimate of $\nabla_\theta \mathbb{E}[R_t]$. However, this gradient estimate can exhibit high variance, which may lead to difficulties or instability during training. To address this issue and reduce variance while preserving unbiasedness, a *baseline* term can be subtracted from the return. This baseline, $b_t(s_t)$, is a learned function of the state $s_t$. The update rule then becomes $\nabla_\theta \log \pi (a_t|s_t;\theta) (R_t - b_t(s_t))$. A typical choice for the baseline $b_t(s_t)$ in RL is an estimate of the value function $V^\pi(s_t)$. Employing this baseline generally leads to a reduction in the variance of the policy gradient estimate [25]. The term $R_t - b_t = Q(s_t, a_t) - V(s_t)$ represents the *advantage* $A(s_t, a_t)$ of taking action $a_t$ in state $s_t$. This advantage can be viewed as a measure of how favorable or unfavorable action $a_t$ is

compared to the average value of the state $s_t$. This method is referred to as the advantage actor-critic (A2C) approach, where the policy $\pi$ serves as the *actor* and the value function $V$ acts as the *critic* [25]. Similar to traditional policy gradient methods, the A2C algorithm can be implemented using VQCs. In [29], the authors utilize VQCs to construct both the actor (policy function) and the critic (value function). Their study demonstrates that, for comparable numbers of model parameters, a hybrid approach—where classical neural networks post-process the outputs from the VQC—achieves superior performance across the tested environments. The asynchronous advantage actor-critic (A3C) algorithm [30] is an enhanced variant of the A2C method that utilizes multiple concurrent actors to learn the policy through parallel processing. This approach involves deploying several agents across several instances of the environment, enabling them to experience a wide range of states simultaneously. By reducing the correlation between states or observations, this method improves the numerical stability of on-policy RL algorithms like actor-critic [30]. Moreover, asynchronous training eliminates the need for extensive replay memory, which helps in reducing memory usage [30]. A3C achieves high sample efficiency and robust learning performance, making it a favored choice in RL. In the context of quantum RL, asynchronous or distributed training can further boost sampling efficiency and leverage the capabilities of multiple quantum computers or quantum processing units (QPUs). In [31], the authors extend the A3C framework to quantum settings, showing that VQC actors and critics can outperform classical models when the sizes of the models are comparable.

### D. Quantum RL with Evolutionary Optimization

One of the significant challenges in current QML applications is the limitation of quantum computers or quantum simulation software in processing input dimensions. These systems can only handle inputs up to a certain level, which is insufficient for encoding larger vectors. In QRL, this constraint means the observation vector that the quantum agent can process from the environment is severely restricted. To address this issue, various dimensional reduction methods have been proposed. Among these, a hybrid quantum-classical approach that incorporates a classical learnable model with a VQC has shown promising results. In the work by Chen et al. [14], a quantum-inspired classical model based on a specific type of tensor network, known as a matrix product state (MPS), is integrated with a VQC to function as a learnable compressor [4] (see Figure 4). The hybrid architecture MPS-VQC, including the tensor network and VQC, is randomly initialized, and the entire model is trained in an end-to-end manner. Although gradient-based methods have achieved considerable success in RL, several challenges remain. Notably, these methods can become trapped in local minima or fail to converge to the optimal solution, particularly in sparse RL environments where the agent frequently receives zero rewards during episodes. Evolutionary optimization techniques have been proposed to address these challenges in classical RL and have demon-

strated significant success [32]. A similar approach can be applied to hybrid quantum-classical RL models. Specifically, a population $\mathcal{P}$ of $N$ agents, represented as parameter vectors $\Theta_i, i \in 1, \cdots, N$, is randomly initialized. In each *generation*, the top-performing agents are selected to serve as *parents* for generating the next generation of agents/parameter vectors. The update rules for the new parameters involve adding Gaussian noise to the parent parameters. This method has been shown to optimize MPS-VQC models effectively and to outperform NN-VQC in selected benchmarks [14].
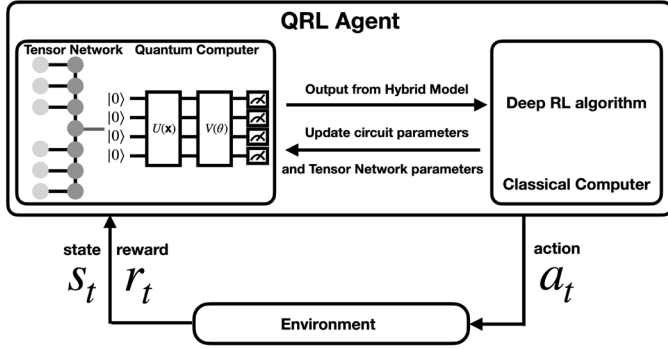


Fig. 4. **Hybrid Quantum-Classical RL with Tensor Networks.**

### E. Quantum RL with Recurrent Policies

The previously mentioned quantum RL methods primarily utilize various VQCs without incorporating recurrent structures. However, recurrent connections are essential in classical machine learning for retaining memory of past time steps. Certain RL tasks necessitate that agents have the capability to remember information from previous time steps to select optimal actions. For instance, environments with partial observability often require agents to make decisions based not only on information from the current time step but also on information accumulated from the past. In classical ML, recurrent neural networks (RNNs), such as long short-term memory (LSTM) [33], have been proposed to solve tasks with temporal dependencies. The quantum version of LSTM (QLSTM) has been designed by replacing classical neural networks with VQCs [7]. It has been shown that QLSTM can outperform classical LSTM in several time-series prediction tasks when the model sizes are similar [7]. To address RL environments with partial observability or those requiring temporal memories, QRL agents utilizing QLSTM as the value or policy function have been proposed in [17]. It has been demonstrated that QLSTM-based value or policy functions enable QRL agents to outperform classical LSTM models with a similar number of parameters.

While the QLSTM-based models achieve significant results in several benchmarks, there is at least one major challenge preventing such models from wide applications. The training of RNNs, both in quantum and classical, requires significant computational resources due to the requirement of performing *backpropagation-through-time* (BPTT). One might question whether it is possible to leverage the capabilities of QLSTM without the need for gradient calculations with respect to

the quantum parameters. Indeed, it has been demonstrated that a randomly initialized RNN can function as a *reservoir*, transforming input information into a high-dimensional space. The only part that requires training is the linear layer following the reservoir. Quantum RNNs, such as QLSTM, can also be utilized as reservoirs [8]. It has been shown that even without training, the QLSTM reservoir can achieve performance comparable to fully trained models [8]. To further enhance the performance of QLSTM-based QRL agents and reduce training resource requirements, a randomly initialized QLSTM can be employed as a reservoir in an RL agent [19]. Numerical simulations have demonstrated that the QLSTM reservoir can achieve performance comparable to, and sometimes superior to, fully trained QLSTM RL agents.

### F. Quantum RL with Fast Weight Programmers

An alternative approach for developing a QRL model that can memorize temporal or sequential dependencies without utilizing quantum RNNs is the *Quantum Fast Weight Programmers* (QFWP). The idea of *Fast Weight Programmers* (FWP) was originally proposed in the work of Schmidhuber [34], [35]. In this sequential learning model, two distinct neural networks (NN) are utilized: the *slow programmer* and the *fast programmer*. Here, the NN weights act as the model/agent's *program*. The core concept of FWP involves the slow programmer generating *updates* or *changes* to the fast programmer's NN weights based on observations at each time-step. This *reprogramming* process quickly redirects the fast programmer's attention to salient information within the incoming data stream. Notably, the slow programmer does not completely overwrite the fast programmer but instead applies updates or changes. This approach allows the fast programmer to incorporate previous observations, enabling a simple feed-forward NN to manage sequential prediction or control without the high computational demands of recurrent neural networks (RNNs). The idea of FWP can be further extended into the hybrid quantum-classical regime as described in the work [36]. In the work [36], classical neural networks are used to construct the *slow* networks, which generate values to update the parameters of the *fast* networks, implemented as a VQC.
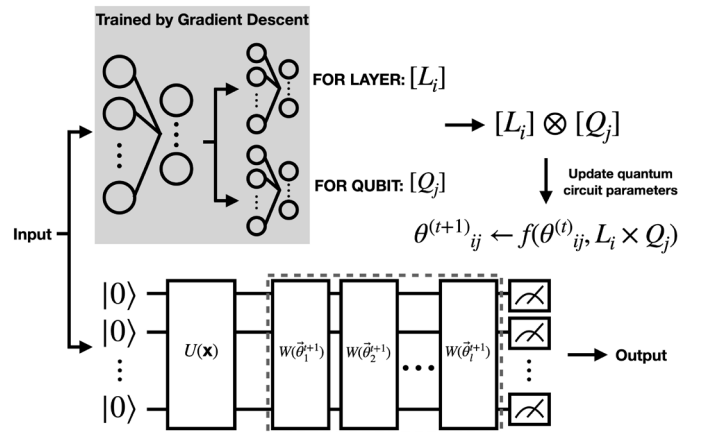


Fig. 5. **Quantum Fast Weight Programmers.**

As illustrated in Figure 5, the input vector $\vec{x}$ is first processed by a classical neural network encoder. The encoder's output is then fed into two additional neural networks. One network generates an output vector $[L_i]$ corresponding to the number of VQC layers, while the other produces an output vector $[Q_j]$ matching the number of qubits in the VQC. We then calculate the outer product of $[L_i]$ and $[Q_j]$. It can be written as $[L_i] \otimes [Q_j] = [M_{ij}] = [L_i \times Q_j] =$

$$\begin{bmatrix} L_1 \times Q_1 & L_1 \times Q_2 & \cdots & L_1 \times Q_n \\ L_2 \times Q_1 & L_2 \times Q_2 & \cdots & L_2 \times Q_n \\ \vdots & \ddots & & \vdots \\ L_l \times Q_1 & L_l \times Q_2 & \cdots & L_l \times Q_n \end{bmatrix}$$, where $l$ is the num-

ber of learnable layers in VQC and $n$ is the number of qubits. At time $t + 1$, the updated VQC parameters can be calculated as $\theta_{ij}^{t+1} = f(\theta_{ij}^t, L_i \times Q_j)$, where $f$ combines the previous parameters $\theta_{ij}^t$ with the newly computed $L_i \times Q_j$. In the time series modeling and RL tasks in [36], the *additive* update rule is used. The new circuit parameters are calculated as $\theta_{ij}^{t+1} = \theta_{ij}^t + L_i \times Q_j$. This method preserves information from previous time steps in the circuit parameters, influencing the VQC behavior with each new input $\vec{x}$. The output from the VQC can be further processed by components such as scaling, translation, or a classical neural network to refine the final results.

### G. Quantum RL with Quantum Architecture Search

While QRL has demonstrated effectiveness across various problem domains, the design of successful architectures is far from trivial. Developing VQC architectures tailored to specific problems requires substantial effort and expertise. The field of *quantum architecture search* (QAS) focuses on developing methods to identify high-performing quantum circuits for specific tasks. A QAS problem is formulated by specifying a particular goal (e.g., total returns in RL) and the constraints of the quantum device (e.g., maximum number of quantum operations, set of allowed quantum gates). QAS has been explored in the context of QRL. For instance, in [37], evolutionary algorithms are employed to search for high-performing circuits. The authors define a set of candidate VQC blocks, including entangling blocks, data-encoding blocks, variational blocks, and measurement blocks. The objective of the evolutionary search is to determine an optimal sequence of these blocks, given a constraint on the maximum number of circuit blocks. While this approach has shown effectiveness in the evaluated cases, scalability issues may arise as the search space expands. Differentiable quantum architecture search (DiffQAS) methods, as proposed in [38], draw inspiration from differentiable neural architecture search in classical deep learning to identify effective quantum circuits for RL. In [39], the authors apply DiffQAS to quantum deep $Q$-learning. They parameterize a probability distribution $P(k, \alpha)$ for circuit architecture $k$ using $\alpha$. During training, mini-batches of VQCs are sampled, and the weighted loss is calculated based on the distribution $P(k, \alpha)$. Both the architecture parameter $\alpha$ and the quantum circuit parameters $\theta$ are updated using conventional gradient-based methods. In [40], the authors extend

the DiffQAS framework to asynchronous QRL. This extension allows multiple parallel instances (a single instance is shown in Figure 6 ) to optimize their own structural weights (denoted as $w$ in Figure 6 ) alongside the VQC parameters. The gradients of these structural weights and quantum circuit parameters are shared across instances to enhance the training process.
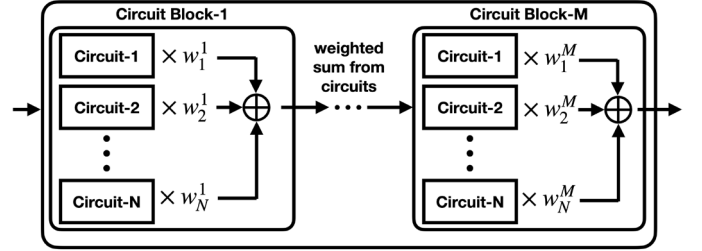


Fig. 6. **Differentiable Quantum Architecture Search (DiffQAS).**

## IV. QUANTUM RL APPLICATIONS AND CHALLENGES

QRL can be extended to multi-agent settings and applied in fields like wireless communication and autonomous control systems [41]. Additionally, as discussed in Section III-G, QAS involves sequential decision-making and can be addressed through RL. In [42], a QRL approach is developed to discover quantum circuit architectures that generate desired quantum states. In the NISQ era, a major challenge for QML applications is the limited quantum resources, which complicates both the training and inference phases. In [43], [44], the authors propose a method using a QNN to generate classical NN weights. For an $N$-qubit QNN, measuring the expectation values of individual qubits provides up to $N$ values. However, collecting the probabilities of all computational basis states $|00\cdots0\rangle, \cdots, |11\cdots1\rangle$ yields $2^N$ values. These values can be rescaled and used as NN weights. Thus, for an NN with $M$ weights, only $\lceil \log_2 M \rceil$ qubits are needed to generate the weights. Numerical simulations demonstrate that the quantum circuit can efficiently generate NN weights, achieving inference performance comparable to conventional training. Future research could further explore the trainability challenges in QRL models highlighted by Sequeira et al. [45], which are key to enhancing their practical performance.

## V. CONCLUSION AND OUTLOOK

This paper introduces the concept of quantum reinforcement learning (QRL), where variational quantum circuits (VQCs) are used as policy and value functions. It also explores advanced constructs, including quantum recurrent policies, quantum fast weight programmers, and QRL with differentiable quantum architectures. QRL holds the potential to offer quantum advantages in various sequential decision-making tasks.

## REFERENCES

[1] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*. Cambridge university press, 2010.

[2] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke *et al.*, "Noisy intermediate-scale quantum algorithms," *Reviews of Modern Physics*, vol. 94, no. 1, p. 015004, 2022.

[3] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, "Quantum circuit learning," *Physical Review A*, vol. 98, no. 3, p. 032309, 2018.

[4] S. Y.-C. Chen, C.-M. Huang, C.-W. Hsing, and Y.-J. Kao, "An end-to-end trainable hybrid classical-quantum classifier," *Machine Learning: Science and Technology*, vol. 2, no. 4, p. 045021, 2021.

[5] J. Qi, C.-H. H. Yang, and P.-Y. Chen, "Qtn-vqc: An end-to-end learning framework for quantum neural networks," *Physica Scripta*, vol. 99, 12 2023.

[6] S. Y.-C. Chen, T.-C. Wei, C. Zhang, H. Yu, and S. Yoo, "Quantum convolutional neural networks for high energy physics data analysis," *Physical Review Research*, vol. 4, no. 1, p. 013231, 2022.

[7] S. Y.-C. Chen, S. Yoo, and Y.-L. L. Fang, "Quantum long short-term memory," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 8622–8626.

[8] S. Y.-C. Chen, D. Fry, A. Deshmukh, V. Rastunkov, and C. Stefanski, "Reservoir computing via quantum recurrent neural networks," *arXiv preprint arXiv:2211.02612*, 2022.

[9] S. S. Li, X. Zhang, S. Zhou, H. Shu, R. Liang, H. Liu, and L. P. Garcia, "Pqlm-multilingual decentralized portable quantum language model," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[10] C.-H. H. Yang, J. Qi, S. Y.-C. Chen, Y. Tsao, and P.-Y. Chen, "When bert meets quantum temporal convolution learning for text classification in heterogeneous computing," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 8602–8606.

[11] R. Di Sipio, J.-H. Huang, S. Y.-C. Chen, S. Mangini, and M. Worring, "The dawn of quantum natural language processing," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 8612–8616.

[12] J. Stein, I. Christ, N. Kraus, M. B. Mansky, R. Müller, and C. Linnhoff-Popien, "Applying qnlp to sentiment analysis in finance," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 2. IEEE, 2023, pp. 20–25.

[13] S. Y.-C. Chen, C.-H. H. Yang, J. Qi, P.-Y. Chen, X. Ma, and H.-S. Goan, "Variational quantum circuits for deep reinforcement learning," *IEEE Access*, vol. 8, pp. 141 007–141 024, 2020.

[14] S. Y.-C. Chen, C.-M. Huang, C.-W. Hsing, H.-S. Goan, and Y.-J. Kao, "Variational quantum reinforcement learning via evolutionary optimization," *Machine Learning: Science and Technology*, vol. 3, no. 1, p. 015025, 2022.

[15] O. Lockwood and M. Si, "Reinforcement learning with quantum variational circuit," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 16, no. 1, 2020, pp. 245–251.

[16] A. Skolik, S. Jerbi, and V. Dunjko, "Quantum agents in the gym: a variational quantum algorithm for deep q-learning," *Quantum*, vol. 6, p. 720, 2022.

[17] S. Y.-C. Chen, "Quantum deep recurrent reinforcement learning," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[18] N. Meyer, C. Ufrecht, M. Periyasamy, D. D. Scherer, A. Plinge, and C. Mutschler, "A survey on quantum reinforcement learning," *arXiv preprint arXiv:2211.03464*, 2022.

[19] S. Y.-C. Chen, "Efficient quantum recurrent reinforcement learning via quantum reservoir computing," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 13 186–13 190.

[20] A. Abbas, D. Sutter, C. Zoufal, A. Lucchi, A. Figalli, and S. Woerner, "The power of quantum neural networks," *Nature Computational Science*, vol. 1, no. 6, pp. 403–409, 2021.

[21] M. C. Caro, H.-Y. Huang, M. Cerezo, K. Sharma, A. Sornborger, L. Cincio, and P. J. Coles, "Generalization in quantum machine learning from few training data," *Nature communications*, vol. 13, no. 1, pp. 1–11, 2022.

[22] Y. Du, M.-H. Hsieh, T. Liu, and D. Tao, "Expressive power of parametrized quantum circuits," *Physical Review Research*, vol. 2, no. 3, p. 033125, 2020.

[23] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, "Evaluating analytic gradients on quantum hardware," *Physical Review A*, vol. 99, no. 3, p. 032331, 2019.

[24] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, C. Blank, K. McKiernan, and N. Killoran, "Pennylane: Automatic differentiation of hy-

brid quantum-classical computations," *arXiv preprint arXiv:1811.04968*, 2018.

[25] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[27] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[28] S. Jerbi, C. Gyurik, S. Marshall, H. Briegel, and V. Dunjko, "Parametrized quantum policies for reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 28 362–28 375, 2021.

[29] M. Kölle, M. Hgog, F. Ritz, P. Altmann, M. Zorn, J. Stein, and C. Linnhoff-Popien, "Quantum advantage actor-critic for reinforcement learning," *arXiv preprint arXiv:2401.07043*, 2024.

[30] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[31] S. Y.-C. Chen, "Asynchronous training of quantum reinforcement learning," *Procedia Computer Science*, vol. 222, pp. 321–330, 2023, international Neural Network Society Workshop on Deep Learning Innovations and Applications (INNS DLIA 2023). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050923009365

[32] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," *arXiv preprint arXiv:1712.06567*, 2017.

[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[34] J. Schmidhuber, "Learning to control fast-weight memories: An alternative to dynamic recurrent networks," *Neural Computation*, vol. 4, no. 1, pp. 131–139, 1992.

[35] ——, "Reducing the ratio between learning complexity and number of time varying variables in fully recurrent nets," in *ICANN'93: Proceedings of the International Conference on Artificial Neural Networks Amsterdam, The Netherlands 13–16 September 1993 3*. Springer, 1993, pp. 460–463.

[36] S. Y.-C. Chen, "Learning to program variational quantum circuits with fast weights," *arXiv preprint arXiv:2402.17760*, 2024.

[37] L. Ding and L. Spector, "Evolutionary quantum architecture search for parametrized quantum circuits," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2022, pp. 2190–2195.

[38] S.-X. Zhang, C.-Y. Hsieh, S. Zhang, and H. Yao, "Differentiable quantum architecture search," *Quantum Science and Technology*, vol. 7, no. 4, p. 045023, 2022.

[39] Y. Sun, Y. Ma, and V. Tresp, "Differentiable quantum architecture search for quantum reinforcement learning," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 2. IEEE, 2023, pp. 15–19.

[40] S. Y.-C. Chen, "Differentiable quantum architecture search in asynchronous quantum reinforcement learning," *arXiv preprint arXiv:2407.18202*, 2024.

[41] C. Park, W. J. Yun, J. P. Kim, T. K. Rodrigues, S. Park, S. Jung, and J. Kim, "Quantum multiagent actor–critic networks for cooperative mobile access in multi-uav systems," *IEEE Internet of Things Journal*, vol. 10, no. 22, pp. 20 033–20 048, 2023.

[42] S. Y.-C. Chen, "Quantum reinforcement learning for quantum architecture search," in *Proceedings of the 2023 International Workshop on Quantum Classical Cooperative*, 2023, pp. 17–20.

[43] C.-Y. Liu, E.-J. Kuo, C.-H. A. Lin, J. G. Young, Y.-J. Chang, M.-H. Hsieh, and H.-S. Goan, "Quantum-train: Rethinking hybrid quantum-classical machine learning in the model compression perspective," *arXiv preprint arXiv:2405.11304*, 2024.

[44] C.-Y. Liu, C.-H. A. Lin, C.-H. H. Yang, K.-C. Chen, and M.-H. Hsieh, "Qtrl: Toward practical quantum reinforcement learning via quantum-train," *arXiv preprint arXiv:2407.06103*, 2024.

[45] A. Sequeira, L. P. Santos, and L. Soares Barbosa, "Trainability issues in quantum policy gradients," *Machine Learning: Science and Technology*, 2024.