# Quantum Machine Learning Architecture Search via Deep Reinforcement Learning

Xin Dai[1], Tzu-Chieh Wei[2], Shinjae Yoo[1], Samuel Yen-Chi Chen[1]
[1]Computational Science Initiative, Brookhaven National Laboratory
[2]C.N. Yang Institute for Theoretical Physics and Department of Physics and Astronomy, Stony Brook University
{xdai, sjyoo}@bnl.gov, tzu-chieh.wei@stonybrook.edu, ycchen1989@ieee.org

2024 IEEE International Conference on Quantum Computing and Engineering (QCE) | 979-8-3315-4137-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/QCE60285.2024.00179

***Abstract*—The rapid advancement of quantum computing (QC) and machine learning (ML) has given rise to the burgeoning field of quantum machine learning (QML), aiming to capitalize on the strengths of quantum computing to propel ML forward. Despite its promise, crafting effective QML models necessitates profound expertise to strike a delicate balance between model intricacy and feasibility on Noisy Intermediate-Scale Quantum (NISQ) devices. While complex models offer robust representation capabilities, their extensive circuit depth may impede seamless execution on extant noisy quantum platforms. In this paper, we address this quandary of QML model design by employing deep reinforcement learning to explore proficient QML model architectures tailored for designated supervised learning tasks. Specifically, our methodology involves training an RL agent to devise policies that facilitate the discovery of QML models without predetermined ansatz. Furthermore, we integrate an adaptive mechanism to dynamically adjust the learning objectives, fostering continuous improvement in the agent's learning process. Through extensive numerical simulations, we illustrate the efficacy of our approach within the realm of classification tasks. Our proposed method successfully identifies VQC architectures capable of achieving high classification accuracy while minimizing gate depth. This pioneering approach not only advances the study of AI-driven quantum circuit design but also holds significant promise for enhancing performance in the NISQ era.**

***Index Terms*—quantum machine learning, quantum neural networks, variational quantum circuits, quantum architecture search**

## I. Introduction

Quantum computing (QC) holds the potential to revolutionize computational tasks, offering distinct advantages over classical computers [1]. The convergence of advancements in quantum hardware and machine learning applications has sparked a growing interest in exploring the synergies between these cutting-edge technologies. Although existing quantum computers still suffer from noise, a promising solution lies in a hybrid quantum-classical framework. Here, computational tasks are divided into two parts: one executed on a quantum computer and the other on a classical computer [2], [3]. Central to this paradigm is the Variational Quantum Algorithm (VQA) [2], [3], which serves as the cornerstone of hybrid computing. Quantum machine learning (QML) algorithms heavily rely on VQAs, utilizing variational quantum circuits (VQCs) as trainable components akin to classical neural networks. QML has demonstrated remarkable success across various domains, including classification [4]–[8], time-series modeling [9], natural language processing [10]–[13], generative modeling [14]–[16], and reinforcement learning [17]–[24]. While existing QML models have shown promise, they often require expert knowledge to design effective quantum circuit architectures. For instance, the configuration of encoding and variational subcircuits within VQCs significantly influences model performance and the realization of potential quantum advantages [25]. Moreover, the vast search space of VQCs presents a challenge, given the multitude of possible circuit architectures. The necessity for expertise in quantum circuit design poses a barrier to the widespread adoption of QML techniques beyond the quantum computing community, limiting their application in other scientific domains.

In this paper, we address the challenge of designing QML models by introducing a novel approach called deep reinforcement learning with adaptive search of learning targets (RL-QMLAS). Our method, shown in Figure 1, focuses on classification tasks, wherein the RL agent's objective is to discover an optimal quantum gate sequence. Furthermore, we incorporate an adaptive learning threshold, dynamically adjusting the reward scheme during RL training to enhance the agent's learning process, enabling the model to acquire high-performing policies effectively. Through numerical simulations, we demonstrate that our proposed methods can effectively generate VQC architectures. These architectures achieve high classification accuracy without requiring prior physical knowledge and maintain a shallower circuit depth compared to manually crafted architectures. In addition, the adaptive learning target can enhance the agent learning process and reduce the need for a high pre-defined learning target. This paper is organized as follows: Section II provides a brief survey on current development of QAS. In Section III, we describe the concept of VQC, which is the core of existing QML models and the target the proposed framework is to search for. We formulate the QAS problem in Section IV and describe the RL techniques used in this work in Section V. We provide the details of simulation in Section VI and results in Section VII. Finally conclude in Section IX.

This work was supported by the U.S. DOE, Office of Science, Office of High Energy Physics under award DE-SC-0012704. This research used resources of the NERSC, under Contract No.DE-AC02-05CH11231 using NERSC award HEP-ERCAP0023403.

979-8-3315-4137-8/24/$31.00 ©2024 IEEE
DOI 10.1109/QCE60285.2024.00179

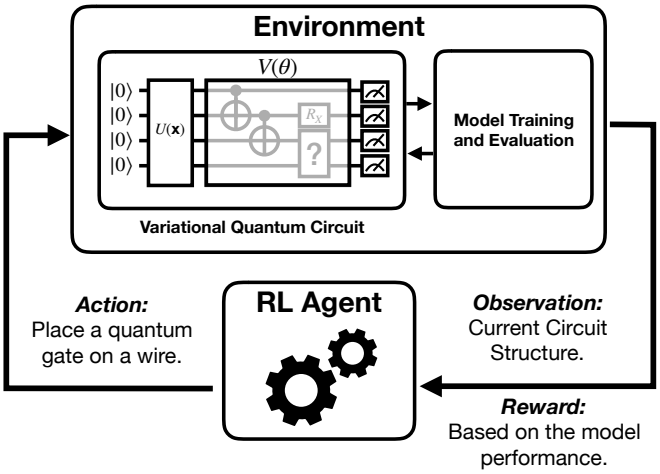


Fig. 1: **Overall scheme for RL-QMLAS.**

## II. Relevant Works

Machine learning techniques have been applied to tackle various quantum computing challenges such as quantum architecture search (QAS). The target task of a QAS might be generating a desired quantum state [26]–[36], finding an efficient circuit for solving chemical ground states [36]–[40], solving an optimization task [36], [38], [41]–[46], optimizing a given quantum circuit for a particular hardware architecture [47], compiling a circuit [48]–[50] or performing a machine learning task [42], [43], [51]–[57]. Various approaches are employed to find the optimal circuit for specified tasks. For example, the works [26]–[29], [31], [33], [34], [37], [41], [47], [50] consider the reinforcement learning based methods while the works [32], [51]–[53] works use different variants of evolutionary algorithms to search for the circuit. Differentiable QAS methods are also developed to leverage the highly successful gradient-based methods [43]–[45], [55]. Different ways of encoding the quantum circuit architecture are devised. For example, the works [39], [42] propose graph-based method while the work [47] consider the convolutional neural network based method to encode the quantum circuit architecture. Regarding the circuit performance metric, it can be a direct evaluation of the circuit performance on the particular task [37], [38], [51] or the closeness of the generated circuit to the actual circuit [26], [27], [42]. To reduce the computational resource required in direct evaluation, certain predictor-based methods are proposed to use neural network to predict the quantum model performance without direct circuit evaluation [40], [56]. The proposed method in this paper is to further generalize the concepts used in [26], [27] to more than finding a quantum circuit to synthesize a particular quantum state, but can actually perform a QML task. This paper further generalize the methods proposed in the work [37] to QML tasks. Our work is also different from previous works on quantum circuit optimization [47], since in our work, circuit ansatz are not provided. Though optimizing an existing circuit ansatz can decrease training time, it is not without cost; the ansatz necessitates input from quantum experts. Our approach investigates the prospect of whether, in the absence of a predefined ansatz, an agent can autonomously discover high-performing circuits.

## III. Variational Quantum Circuits

Variational quantum circuits (VQC), also known as parameterized quantum circuits (PQC) is a special kind of quantum circuit with trainable parameters which can be trained via gradient-based [4], [58], [59] or gradient-free [60] methods. This kind of circuits play a crucial role in the hybrid quantum-classical computing paradigm in which certain computing tasks are implemented on quantum computer while tasks not suitable for existing quantum computers are carried out by classical computers. Consider an $n$-qubit system. The fundamental components of a VQC (illustrated in Figure 2) include the *encoding* circuit $U(\vec{x})$, responsible for transforming the classical input vector $\vec{x}$ into a quantum state $U(\vec{x})\left|0\right\rangle^{\otimes n}$, the *variational* circuit $V(\vec{\theta})$, serving as the actual learning component with trainable parameters $\vec{\theta}$, and the final *measurement* operation, used to extract information from the circuit. The VQC used in this work can be expressed as $\overrightarrow{f(\vec{x};\vec{\theta})} = \left(\left\langle \hat{Z}_1 \right\rangle, \cdots, \left\langle \hat{Z}_n \right\rangle\right)$ , where $\left\langle \hat{Z}_k \right\rangle = \left\langle 0 \left| U^\dagger(\vec{x}) V^\dagger(\vec{\theta}) \hat{Z}_k V(\vec{\theta}) U(\vec{x}) \right| 0 \right\rangle$. The $Z$-expectation values can be derived via multiple sampling (shots) on real quantum devices or direct computation when using a simulation software. VQCs have been shown to provide certain advantages over classical neural networks [25], [61] and have demonstrated successful applications in various ML tasks [4], [5], [7], [9], [15], [17]–[19]. The variational circuit $V(\vec{\theta})$ requires special attention since the design of this circuit component will affect the QML model significantly. In general, several control gates and rotation gates are required in this circuit component, and there are several ansatzes which have been shown to be successful. However, these designs are not tailored for specific QML tasks, therefore may not be the optimal in terms of circuit depth.

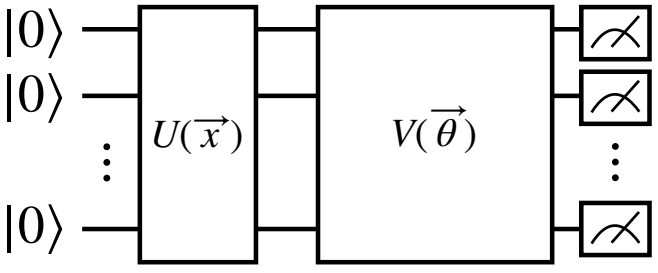


Fig. 2: **Generic variational quantum circuit (VQC) structure.**

## IV. Quantum Architecture Search

In this paper, we want to solve the following problem: Suppose we are given an initial quantum state $\left|0\right\rangle^{\otimes n}$, an

supervised learning dataset $\{(x_i, y_i)\}$ , and encoding circuit $U$, the maximum gate number $L$, the allowed gate set $\mathbb{G}$, an performance metric $\mathcal{M}$ (e.g. classification accuracy or the loss function $\mathcal{L}$), the goal is to find the quantum gate sequence $S$ such that the performance metric is maximized (or minimized).

**Definition IV.1** (QAS for Quantum Supervised Learning)**.** Given an $n$-qubit system with ground state initialization $|0\rangle^{\otimes n}$ and a predefined encoding circuit $U$, the QAS for quantum supervised learning is to find the gate sequence with length $< L$ composed from the allowed gate sets $\mathbb{G}$ to build the trainable circuit $V(\vec{\theta})$ such that, after the predefined training process, the quantum function $\overrightarrow{f(\vec{x};\vec{\theta})} = \left(\left\langle \hat{Z}_1 \right\rangle, \cdots, \left\langle \hat{Z}_m \right\rangle\right)$, where $\left\langle \hat{Z}_k \right\rangle = \left\langle 0 \left| U^\dagger(\vec{x}) V^\dagger(\vec{\theta}) \hat{Z}_k V(\vec{\theta}) U(\vec{x}) \right| 0 \right\rangle$ represents the $Z$ expectation value on $k$-th qubits and $m < n$ equals to the number of outputs, can minimize or maximize the given performance metric $\mathcal{M}(y_i, \hat{y}_i)$. Here the $y_i$ and $\hat{y}_i$ represent the ground truth and labels predicted by the quantum model, respectively. The predicted label $\hat{y}_i$ is derived from the $\overrightarrow{f(\vec{x};\vec{\theta})}$ and can be represented as $\hat{y}_i = g(\overrightarrow{f(\vec{x};\vec{\theta})})$, where $g$ is a post-processing function for $f$. Specifically, for our binary classification task we have $\mathcal{M}(\underrightarrow{y_i, \hat{y}_i}) = -\left(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right)$ and $g = (1 + f(\vec{x};\vec{\theta}))/2$

The allowed action (gate) set we consider for this particular problem is

$$\mathbb{G} = \bigcup_{i=1}^{n} \left\{ R_{X_i}, R_{Y_i}, R_{Z_i}, CNOT_{i,(i+1)(mod2)} \right\}, \quad (1)$$

where $R_{X_i}$, $R_{Y_i}$ and $R_{Z_i}$ represent rotations along $X$, $Y$ and $Z$ axis, respectively.

## V. Reinforcement Learning

*Reinforcement learning* (RL) stands as a machine learning approach where an *agent* learns decision-making by interacting with environments [62]. In this setup, the *agent* engages with an *environment* $\mathcal{E}$ over discrete time steps. At each time step $t$, the agent receives a current *state* or *observation* $s_t$ from the environment $\mathcal{E}$ and proceeds to select an *action* $a_t$ from a set of available actions $\mathcal{A}$ based on its governing *policy* $\pi$. This policy $\pi$ functions to map the state or observation $s_t$ to the action $a_t$. Typically, the policy may adopt a probabilistic nature, implying that given a state $s$, the action output can be a probability distribution $\pi(a_t|s_t)$ conditioned on $s_t$. Upon executing the action $a_t$, the agent encounters the subsequent state $s_{t+1}$ and receives a single *reward* $r_t$. This iterative process persists until the agent reaches a terminal state or meets predefined termination conditions (e.g., maximum steps allowed). An *episode* refers to the agent's journey from a randomly chosen initial state through to the terminal state or until it satisfies the stopping criteria.

We define the cumulative discounted reward from time step $t$ as $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$, where $\gamma$ is the discount factor within the range of $(0, 1]$. Essentially, $\gamma$ serves as a parameter set by the investigator to influence how future rewards impact decision-making. A higher $\gamma$ assigns greater importance to future rewards, while a lower $\gamma$ leads to more emphasis on immediate rewards, gradually discounting future ones. The agent's objective is to maximize the expected return from each state $s_t$ during the training phase. The *action-value function*, or *Q-value function*, $Q^\pi(s, a) = \mathbb{E}[R_t|s_t = s, a]$ represents the anticipated return for choosing action $a$ in state $s$ according to policy $\pi$. The optimal action-value function $Q^*(s, a) = \max_\pi Q^\pi(s, a)$ indicates the highest achievable action-value across all conceivable policies. Furthermore, the value of state $s$ under policy $\pi$, $V^\pi(s) = \mathbb{E}[R_t|s_t = s]$, represents the agent's expected return when adhering to policy $\pi$ from state $s$. Various reinforcement learning (RL) algorithms aim to identify the policy that maximizes the value function. Algorithms geared towards maximizing the value function are termed *value-based* RL algorithms. One of the notable example of value-based RL is the $Q$-learning [62].

### A. Q-Learning

$Q$-learning [62] stands out as one of the predominant and fundamental model-free algorithms in RL. In $Q$-learning, the agent acquires knowledge of the optimal action-value function and operates as an *off-policy* algorithm. The learning process starts with the random initialization of the value function $Q^\pi(s, a)$ for all states $s \in S$ and actions $a \in \mathcal{A}$, typically stored in a structured form known as the $Q$-table. The estimates for $Q^\pi(s, a)$ are then progressively updated according to the Bellman equation:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[ r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]. \quad (2)$$

### B. Double Deep Q-Learning

The conventional $Q$-learning approach, as previously elucidated, offers the theoretically optimal action-value function. However, it becomes impractical for problems necessitating extensive memory. Particularly, managing problems characterized by high-dimensional state ($s$) or action ($a$) spaces poses significant challenges. Moreover, in environments featuring continuous state values, the efficient storage of $Q(s, a)$ within a table is unclear. To circumvent this memory constraint, neural networks (NNs) are employed to effectively represent $Q^\pi(s, a) \forall s \in S, a \in \mathcal{A}$. This technique, termed *deep Q-learning*, utilizes NNs, with the network itself referred to as a deep $Q$-network (DQN) [63].

To enhance the stability of the deep DQN, methods such as *experience replay* and the integration of an auxiliary network referred to as the *target network* are employed [63]. Experience replay involves the agent storing encountered experiences during episodes in memory, preserving transition tuples, denoted as $s_t, a_t, r_t, s_{t+1}$. Upon accumulating a sufficient pool of experiences, the agent randomly selects a batch for computation of loss and subsequent update of DQN model parameters. Furthermore, to mitigate the correlation between target and prediction, a duplicate of the DQN, termed the *target network*, is utilized. The parameters $\theta$ of the DQN

are updated iteratively, whereas the parameters $\theta^-$ of the target network undergo updates at periodic intervals. The DQN training is done via minimizing the mean square error (MSE) loss function:

$$L(\theta) = \mathbb{E}\left[\left(r_t + \gamma \max_{a'} Q\left(s_{t+1}, a'; \theta^-\right) - Q\left(s_t, a_t; \theta\right)\right)^2\right] \quad (3)$$

Other loss functions such as Huber loss or mean absolute error (MAE) can also be used. Despite the considerable success achieved by DQN, instances arise where it tends to overestimate the action-value function [64]. As a remedy, an enhanced variant of DQN, known as *Double Deep Q-learning* (DoubleDQN), has been devised [64]. The essence of Double Deep $Q$-learning lies in deconstructing the max operation within the target $y_t^{DQN} = r_t + \gamma \max_{a'} Q\left(s_{t+1}, a'; \theta^-\right)$ into two distinct operations: *action selection* and *action evaluation*. Initially, action selection relies on the policy network, $\operatorname{argmax}_a Q\left(s_{t+1}, a; \theta\right)$, followed by the utilization of the target network to evaluate the action, $Q\left(s_{t+1}, \operatorname{argmax}_a Q\left(s_{t+1}, a; \theta\right), \theta^-\right)$. Consequently, the DoubleDQN target is reformulated as $y_t^{DDQN} = r_t + \gamma Q\left(s_{t+1}, \operatorname{argmax}_a Q\left(s_{t+1}, a; \theta\right), \theta^-\right)$.

The loss function $L(\theta)$ is therefore:

$$L(\theta) = \mathbb{E}\Big[\big(r_t + \gamma Q\left(s_{t+1}, \operatorname{argmax}_a Q\left(s_{t+1}, a; \theta\right), \theta^-\right) - Q\left(s_t, a_t; \theta\right)\big)^2\Big] \quad (4)$$

Then, $\theta$ is updated using the gradient descent method and every few iterations we update the target network $\theta^- \leftarrow \theta$.

### C. N-Step DDQN

The $N$-step DDQN extends the standard DDQN by considering a sequence (trajectory) of $N$ steps when updating the $Q$-values according to the following loss function,

$$L(\theta) = \mathbb{E}\left[\left(\sum_{k=0}^{N-1} \gamma^k r_{t+k+1} + \gamma^N Q\left(s_{t+N}, \operatorname{argmax}_a Q\left(s_{t+N}, a; \theta\right), \theta^-\right) - Q\left(s_t, a_t; \theta\right)\right)^2\right] \quad (5)$$

where $\gamma$ represents the discount factor, $\theta$ represents the policy net parameters, $\theta^-$ represents the target net parameters and $r_{t+k+1}$ is the reward received at timestep $t+k+1$. Note that here we use MSE loss as an example, however other kind of loss function can be used to fine-tune the model performance. In this work, we use the `Smooth_L1` loss. By considering multiple steps, the $N$-step DDQN provides a more informative signal for updates and allows the agent to consider the long-term consequences of its actions, potentially leading to faster convergence and improved performance compared to the standard DQN and DDQN.

## VI. Methods

In this work, we use the TensorCircuit [65] for constructing the variational quantum circuits and PyTorch [66] for building the deep reinforcement learning model.

### A. Experimental setup

In our experimental setup, we employ two types of datasets from scikit-learn [67] to generate data for the binary classification task. The primary dataset is generated using `sklearn.datasets.make_classification`. This function creates $n$-dimensional datasets where data points form normally distributed clusters (with a standard deviation of 1) around the vertices of an $n_{\text{informative}}$-dimensional hypercube. The remaining $n_{\text{redundant}} = n - n_{\text{informative}}$ features are random linear combinations of these informative features, adding complexity to the classification task. In addition to the `make_classification` dataset, we also utilize the `sklearn.datasets.make_moons` dataset. This dataset generates two interleaving half-moon shaped clusters, which are particularly useful for evaluating the model's ability to capture non-linear decision boundaries in binary classification tasks.

For each episode within our RL-QMLAS framework, we initialize the environment with an empty quantum circuit which is in the ground state $|0\rangle^{\otimes n}$. At every step, the agent selects a quantum gate and its location based on the output from the policy network. The set of permissible quantum gates includes rotation gates ($R_X$, $R_Y$ and $R_Z$) and the CNOT gate. To maintain a manageable action space, the agent is tasked with choosing the type of the rotation gate (either $X$, $Y$, or $Z$), while the specific rotation angle is optimized through a classical optimizer during the training of the quantum classifier. This approach balances the complexity of quantum gate selection with practical training considerations. Input data is encoded into the quantum circuit using an `arctan` embedding strategy, specifically, for each feature vector $f \in \mathbb{R}^n$, we compute angles $\theta_i = \arctan(f_i)$ and $\phi_i = \arctan(f_i^2)$ for $i \in \{1, 2, \ldots, n\}$. These angles are then used to apply rotation gates as follows:

$$\forall i \in \{1, 2, \ldots, n\}, \quad \text{apply } R_Y(\theta_i) \text{ and } R_Z(\phi_i) \text{ on qubit } i. \quad (6)$$

Upon the addition of a new gate to the circuit, the quantum classifier is trained for a fixed number of epochs, or until it reaches the desired accuracy level. This iterative process of gate selection and classifier training continues, evolving the quantum circuit step-by-step until the episode concludes, either by achieving the desired accuracy or by reaching the maximum limit of quantum gates. See Section VI-B2 for details about the reward scheme.

Significantly, this method of quantum architecture search through reinforcement learning negates the need for prior physical knowledge, enabling the algorithm to autonomously discover efficient and effective quantum circuits. It represents a novel approach where the intricacies of quantum computation are navigated and optimized through machine learning, rather than relying on pre-established physical *ansatz*.

### B. Hyperparameters of RL

*1) N-step Double Deep Q-Network (DDQN):* In this study, we applied an $N$-step Double Deep Q-Network (DDQN) [64]

to learn efficient quantum circuits for classification tasks. The discount factor $\gamma$ is set as $\gamma = 0.005^{1/L}$, where $L$ represents the maximum number of quantum gates allowed, promoting an approach that favors achieving tasks with minimal gate usage. To stabilize the learning process, we periodically synchronize the parameters of the target network with those of the policy network every 512 steps. An experience replay buffer with a capacity of 16384 transitions is used to break the correlation of sequential learning updates and enhance learning efficiency. The exploration strategy is governed by an $\epsilon$-greedy policy, with $\epsilon$ decaying from 1 to 0.1 over time to balance exploration and exploitation.

The architecture of the deep $Q$-network is a multilayer perceptron (MLP) consisting of a sequence of linear layers, each followed by a LeakyReLU activation function and dropout regularization. The input to the MLP is the state (observation) vector, which is a $4 \times L$ matrix representing the configuration of the quantum circuit at each step, where $L$ is the maximum number of layers. The first two elements of the state vector denote the locations of the control and NOT gates, while the third and fourth elements indicate the location of the rotation gate and the rotation axis, respectively. During training, the DDQN receives a flattened state vector. After each testing episode, the test accuracy is appended to the state vector, providing an additional input feature for the MLP. The output of the MLP corresponds to the $Q$-values for each action, guiding the agent's decision-making process in selecting the most promising gate configurations to explore.

*2) Reward Function:* The RL agent interacts with a quantum circuit environment, where the agent's actions involve selecting the control and target qubits for CNOT gates and the qubit and axis for rotation gates. The environment calculates the accuracy of the resulting quantum circuit and provides a reward signal based on the change in accuracy and the number of layers used. The reward function is defined as follows:

$$R(l) = \begin{cases} 0.2 \cdot \left(\frac{y_l}{y_{\text{target}}}\right) \cdot (L - l), \text{if } y_l \geq y_{\text{target}} \text{ and } l < L, \\ -0.2 \cdot \left(\frac{y_{\text{target}} - y_l}{y_{\text{target}}}\right) \cdot l, \text{if } y_l < y_{\min} \text{ and } l = L, \\ \text{clip}\left(\frac{y_l - y_{l-1}}{y_{l-1} + 1 \times 10^{-6}} - 0.01 \cdot l, -1.5, 1.5\right), \text{otherwise.} \end{cases} \tag{7}$$

Here $y_{\text{target}}$ is the target accuracy. The reward function encourages the agent to achieve or surpass the target accuracy with minimal gate usage. The scaling factor 0.2 moderates the reward magnitude to ensure stability. Moreover, the function imposes a penalty if $y_{\text{target}}$ is not reached when the maximum number of gates is used. Equation 7 also dynamically rewards small improvements in accuracy (the 3rd line), but this reward decreases as more gates are added, steering the agent towards more efficient solutions. To maintain numerical stability and prevent extreme values from skewing the agent's learning, the dynamic reward is constrained within the range $[-1.5, 1.5]$. This careful design of the reward function ensures an optimal trade-off between accuracy and efficiency.

*3) Adaptive Search:* One potential drawback of our previous approach is that the desired classification accuracy $y_{\text{target}}$ is determined *a priori*. If the $y_{\text{target}}$ is set too high, the agent will likely to fail in most of the cases and we need to manually increase $L$, which leads to slow convergence. On the other hand, a $y_{\text{target}}$ that's set too low allows the agent to quickly find simplistic solutions, hindering the development of more efficient and sophisticated quantum circuits.

To address the need for pre-selecting an appropriate classification accuracy target ($y_{\text{target}}$), we introduce an adaptive search strategy. This approach dynamically adjusts both $y_{\text{target}}$ and the exploration rate $\epsilon$ based on the agent's ongoing performance. During the training phase, $y_{\text{target}}$ increases incrementally by 0.01 whenever the agent consistently meets or exceeds this threshold across a specified number of episodes, such as 10 successes in 12 consecutive episodes. In the testing phase, a similar mechanism is in place. If the agent repeatedly achieves higher accuracies over 5 consecutive tests, the $y_{\text{target}}$ is further increased by 0.01, challenging the agent to refine its performance. Concurrently, $\epsilon$ is decreased to 95% of its value, shifting the agent's focus from exploration to exploitation. As a result, the agent increasingly relies on learned behaviors and experiences rather than random exploration, enhancing its proficiency.

## VII. Results

### A. Fixed target results

Our RL agent was first evaluated using the `make_classification` dataset. The outcomes are illustrated in Figure 3, where we observed a consistent improvement in classification accuracy (Figure 3a) and a decrease in the number of quantum gates required (Figure 3c). In the testing phase, the exploration rate $\epsilon$ was set to zero, ensuring that the agent's gate selection was entirely based on the learned policy. Notably, a stable and high testing accuracy (Figure 3b) coupled with a minimal number of gates (Figure 3d) was achieved after several hundred training episodes. This indicates the agent's capacity to learn and its efficiency in converging to an optimized quantum circuit structure over the course of training. Moreover, the rewards pattern in both training (Figure 3e) and testing (Figure 3f) further confirmed the agent's proficiency in balancing classification accuracy with circuit simplicity. For the `make_moons` dataset, similar trends were observed (Figure 4), signifying the robustness of our reinforcement learning approach. The agent not only maintained high classification accuracy but also continued to design quantum circuits with an optimized number of gates. This consistency across different datasets indicates the potential of our method to be applied broadly in quantum machine learning tasks without prior physical knowledge.

### B. Adaptive Search Results

The adaptive search strategy has been implemented to dynamically adjust the target accuracy $y_{\text{target}}$ and exploration rate $\epsilon$ during the training of our RL agent. This approach is designed to progressively challenge the agent, enhancing its

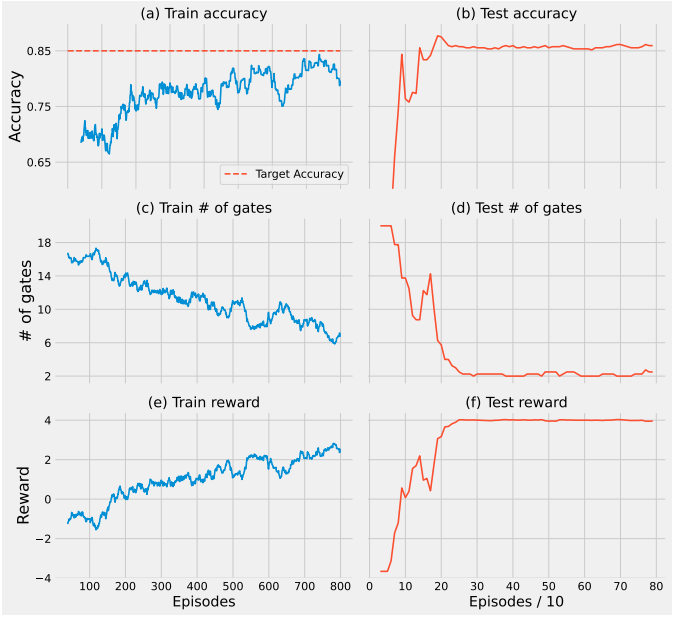


Fig. 3: Reinforcement learning performance metrics using the `make_classification` dataset with a fixed target accuracy of 0.85 and a maximum of 20 quantum gates for a total of 800 episodes. Training accuracy (a) and the number of gates (c) are smoothed with a 40-episode moving average. For testing accuracy (b) and gate count (d), a 4-episode moving average is applied. The maximum training epoch after each action is set to 15. The rewards patterns during training (e) and testing (f) further demonstrate the agent's learning.

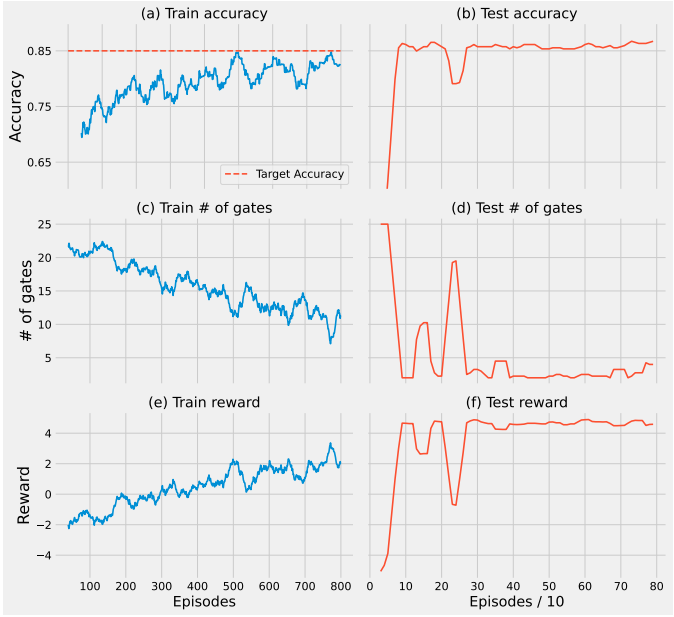


Fig. 4: Reinforcement learning performance metrics using the `make_moons` dataset with a target accuracy of 0.85 and a maximum of 25 quantum gates for a total of 800 episodes. Training accuracy (a) and the number of gates (c) are smoothed with a 40-episode moving average. For testing accuracy (b) and gate count (d), a 4-episode moving average is applied. The maximum training epoch after each action is set to 25.

ability to construct efficient quantum circuits. The results of this strategy are discussed below for two datasets.

Figure 5 presents the results for the `make_classification` dataset. In Figure 5a, the agent's training accuracy oscillates around the dynamic target accuracy, $y_{\text{target}}$, demonstrating continuous adjustment and learning in response to its evolution. To improve the stability and performance, we increased the training episodes from 800 (fixed $y_{\text{target}}$ scenario, Figure 3) to 1200. Figure 5c shows an initial increase in the number of gates, followed by stabilization, indicating the agent's attempt to balance the quantum circuit's complexity and efficiency. During the testing phase, as depicted in Figure 5b and d, we observe more significant performance fluctuations compared to the fixed $y_{\text{target}}$ scenario in Figure 3. Each increase in $y_{\text{target}}$ results in a temporary performance dip, followed by stabilization and alignment with the new $y_{\text{target}}$. After 1200 episodes, the agent efficiently utilizes 4 quantum gates to achieve a classification accuracy of 0.93.

Figure 7 showcases an example of quantum circuit discovered by the RL agent for the `make_classification` dataset using the adaptive search strategy, corresponding to the experiment shown in Figure 5. This circuit demonstrates the agent's capability to learn and optimize a quantum circuit tailored to the specific task requirements.

For the `make_moons` dataset, shown in Figure 6, the agent's performance is more variable due to the dataset's inherent complexity. Both training and testing phases indicate greater challenges in consistently meeting the dynamic target accuracy $y_{\text{target}}$, especially after it surpasses 0.85 (Figure 6b). This suggests a limitation in the agent's capability given the current resources, i.e., the number of qubits and the maximum circuit depth. Nonetheless, the agent shows adaptability and learning capacity, albeit with a less stable optimization process compared to the `make_classification` dataset.

### C. Comparison with classical machine learning

A comparative analysis with classical machine learning methods further elucidates the effectiveness of our quantum classifier. We conducted experiments using logistic regression (LR) and support vector machine (SVM) on the same datasets to benchmark performance.

For the `make_classification` dataset, both LR and SVM models achieved an accuracy of 90.3%. In contrast, for the `make_moons` dataset, the LR model attained an accuracy of 82.5%, while the SVM model, leveraging its RBF kernel, reached an accuracy of 100%. Notably, the 4-dimensional `make_classification` dataset highlights the efficiency of our quantum classifier. With merely 2 quantum gates, it paralleled the performance of the LR and SVM models. The LR model utilized 5 parameters (comprising 4 coefficients and 1 intercept), whereas the SVM model employed a total

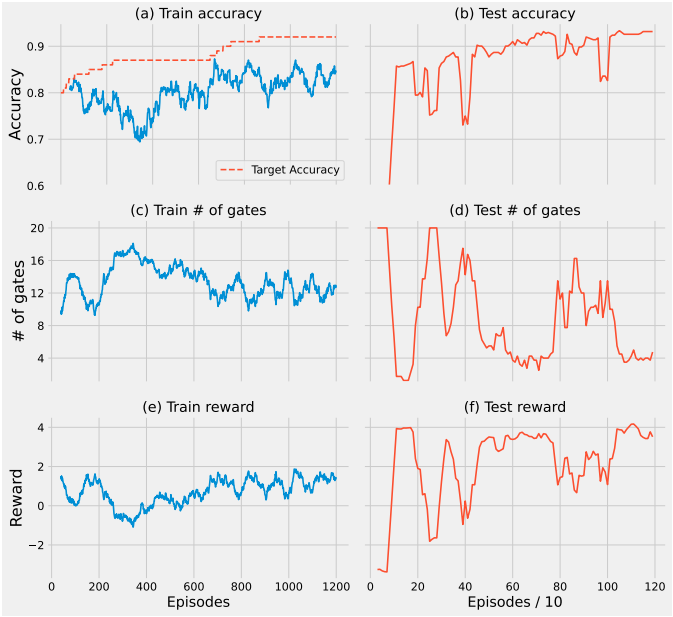


Fig. 5: Performance of the reinforcement learning agent on the `make_classification` dataset over 1200 episodes using a adaptive search strategy, starting from an initial target accuracy of 0.8. Panel (a) includes the dynamic target accuracy, which is adjusted by the adaptive search strategy, alongside the training accuracy. All other experimental conditions and metric smoothing methods align with those described for the fixed target experiments.

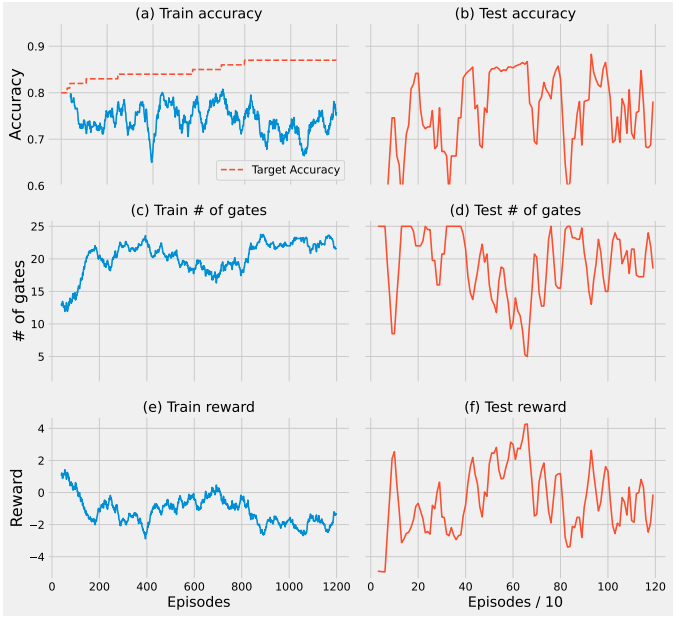


Fig. 6: Performance of the reinforcement learning agent on the `make_moons` dataset over 1200 episodes using a adaptive search strategy, starting from an initial target accuracy of 0.8. Panel (a) includes the dynamic target accuracy, which is adjusted by the adaptive search strategy, alongside the training accuracy. All other experimental conditions and metric smoothing methods align with those described for the fixed target experiments.

of 110 support vectors in this instance. In the case of the 2-dimensional `make_moons` dataset, the LR model required 3 parameters, and the SVM model used 44 support vectors. Here again, our quantum classifier demonstrated comparable accuracy, albeit with fewer parameters. This outcome is particularly noteworthy for the `make_classification` dataset, underscoring the potential of quantum classifiers in achieving high performance with reduced parameterization.

## VIII. Discussion

While the adaptive search strategy enables the agents to achieve better accuracy adaptively without relying on the initial guessing, the successful outcome of such strategy still relies partially on the specific target accuracy update scheme. For instance, if the target accuracy is increased too quickly, it is possible that the agents will get stuck and the resulting accuracy become worse. Further investigation will be required to establish the optimal schedule of changing the learning target as well as the connection between model performance and the complexity of the dataset. In more realistic scenarios, quantum devices are subject to noise and decoherence. Consequently, quantum circuit architectures identified through noise-free simulations may exhibit poor performance when directly implemented on real quantum devices. It is therefore compelling to explore the generalization of the proposed RL-QMLAS framework to noisy quantum devices or more

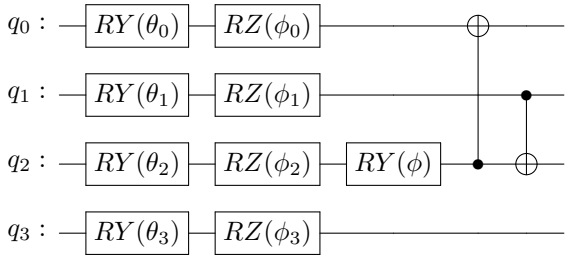


Fig. 7: An example of quantum circuit learned by the RL agent for the `make_classification` dataset using the adaptive search strategy, corresponding to the experiment shown in Figure 5 at episode 1200. The state vector for this circuit is $[[4,0,2,2],[2,0,4,0],[1,2,4,0],\ldots]$, where $\ldots$ denotes $[0,0,0,0]$ for the remaining $L-3$ layers, and $L$ is the maximum number of layers allowed. The initial $R_Z$ and $R_Y$ gates correspond to the `arctan` data embedding (Equation 6).

challenging conditions, such as fluctuating or drifting noise patterns. Another direction of investigation involves the scaling behavior of the proposed framework as the number of qubits increases. It is noted that as the number of qubits $n$ increases, the potential combinations of quantum gates scale up rapidly. For example, the number of allowed gates described in Equation 1 scales at $3n + n(n-1) = \Omega(n^2)$. This rapid scaling presents significant challenges in designing the deep neural

networks for the RL agents, as the number of output neurons would increase quickly. Further studies are required to design more efficient RL agents to generate plausible actions for constructing larger-scale quantum circuits.

## IX. Conclusion

In this paper, we present a framework that leverages deep reinforcement learning to construct quantum machine learning models tailored for classification tasks. Through extensive numerical simulations across various scenarios, our approach demonstrates the capability to develop high-performing QML models without the need for manually designing the learnable circuit based on prior physical knowledge. Furthermore, our models achieve commendable performance while utilizing a moderate number of quantum gates, making them suitable for implementation on existing noisy quantum devices. These findings offer a pioneering avenue for exploring the potential of automated QML in diverse application domains.

## Appendix

In this appendix, we present additional experimental results using the `make_classification` and `make_moons` datasets under varied conditions, as extensions to the main experiments described in the manuscript. These supplementary results Figs. 8, 9, 10 and 11 complement the primary findings by illustrating how variations in target accuracies and the implementation of adaptive search strategy influence the agent's learning trajectory and quantum circuit optimization.

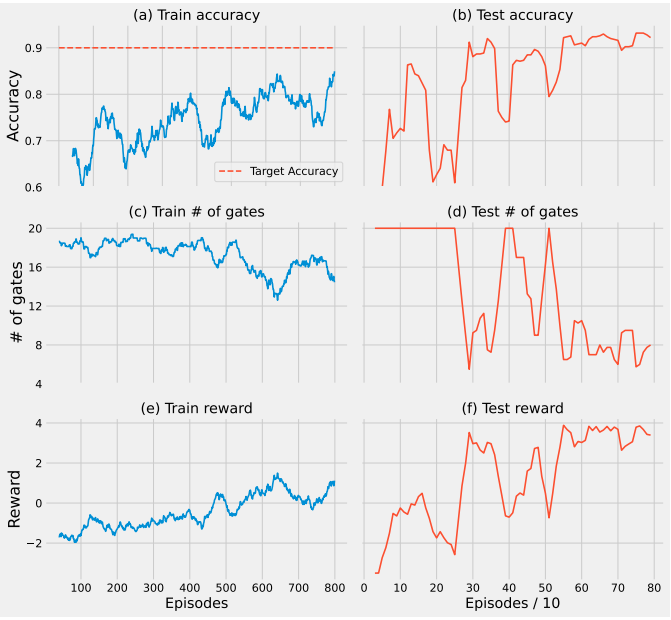


Fig. 8: Experimental results under the same conditions as in Figure 3, with the only difference being the target accuracy set at 0.90.

## References


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


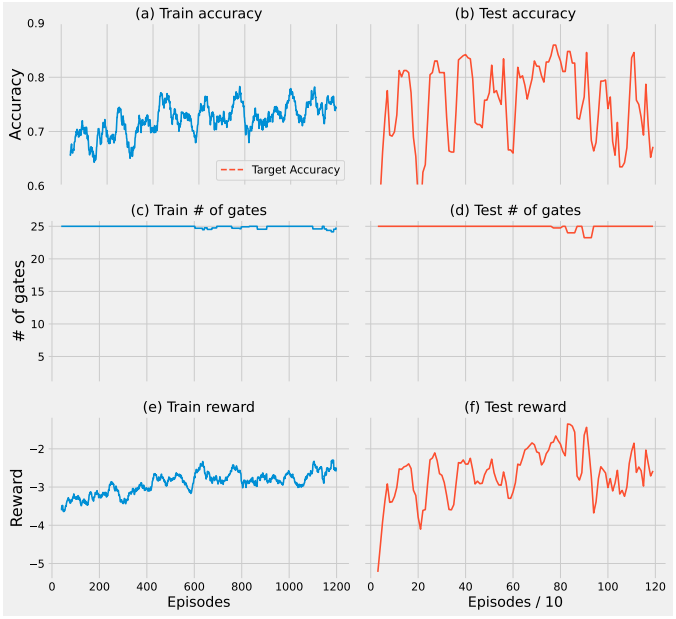


Fig. 9: Experimental results under the same conditions as in Figure 4, with the only difference being the target accuracy set at 0.90.

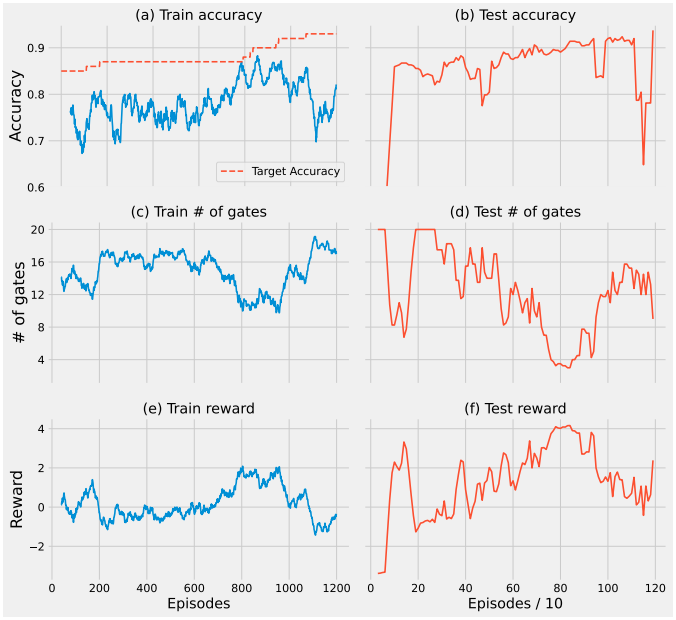


Fig. 10: Experimental results under the same conditions as in Figure 5, with the only difference being the initial target accuracy set at 0.85.


[2] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, *et al.*, "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021.

[3] 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.

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

[5] S. Y.-C. Chen, T.-C. Wei, C. Zhang, H. Yu, and S. Yoo, "Quantum convolutional neural networks for high energy physics data analysis,"


Fig. 11: Experimental results under the same conditions as in Figure 6, with the only difference being the initial target accuracy set at 0.85.

*Physical Review Research*, vol. 4, no. 1, p. 013231, 2022.

[6] 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.

[7] R. L'Abbate, A. D'Onofrio, S. Stein, S. Y.-C. Chen, A. Li, P.-Y. Chen, J. Chen, and Y. Mao, "A quantum-classical collaborative training architecture based on quantum state fidelity," *IEEE Transactions on Quantum Engineering*, 2024.

[8] J. Wu and Q. Li, "Poster: Scalable quantum convolutional neural networks for edge computing," in *2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)*, pp. 307–309, IEEE, 2022.

[9] 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)*, pp. 8622–8626, IEEE, 2022.

[10] 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)*, pp. 1–5, IEEE, 2023.

[11] 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)*, pp. 8602–8606, IEEE, 2022.

[12] 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)*, pp. 8612–8616, IEEE, 2022.

[13] 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, pp. 20–25, IEEE, 2023.

[14] S. A. Stein, B. Baheri, D. Chen, Y. Mao, Q. Guan, A. Li, B. Fang, and S. Xu, "Qugan: A quantum state fidelity based generative adversarial network," in *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp. 71–81, IEEE, 2021.

[15] M. Kölle, G. Stenzel, J. Stein, S. Zielinski, B. Ommer, and C. Linnhoff-Popien, "Quantum denoising diffusion models," *arXiv preprint arXiv:2401.07049*, 2024.

[16] C. Chu, G. Skipper, M. Swany, and F. Chen, "Iqgan: Robust quantum generative adversarial network for image synthesis on nisq devices," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, IEEE, 2023.

[17] 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. 141007–141024, 2020.

[18] 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. 28362–28375, 2021.

[19] 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.

[20] 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.

[21] 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)*, pp. 13186–13190, IEEE, 2024.

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

[23] W. J. Yun, J. Park, and J. Kim, "Quantum multi-agent meta reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 11087–11095, 2023.

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

[25] 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.

[26] E.-J. Kuo, Y.-L. L. Fang, and S. Y.-C. Chen, "Quantum architecture search via deep reinforcement learning," *arXiv preprint arXiv:2104.07715*, 2021.

[27] E. Ye and S. Y.-C. Chen, "Quantum architecture search via continual reinforcement learning," *arXiv preprint arXiv:2112.05779*, 2021.

[28] T. Kimura, K. Shiba, C.-C. Chen, M. Sogabe, K. Sakamoto, and T. Sogabe, "Quantum circuit architectures via quantum observable markov decision process planning," *Journal of Physics Communications*, vol. 6, no. 7, p. 075006, 2022.

[29] T. Sogabe, T. Kimura, C.-C. Chen, K. Shiba, N. Kasahara, M. Sogabe, and K. Sakamoto, "Model-free deep recurrent q-network reinforcement learning for quantum circuit architectures design," *Quantum Reports*, vol. 4, no. 4, pp. 380–389, 2022.

[30] X. Lu, K. Pan, G. Yan, J. Shan, W. Wu, and J. Yan, "Qas-bench: rethinking quantum architecture search and a benchmark," in *International Conference on Machine Learning*, pp. 22880–22898, PMLR, 2023.

[31] A. Kundu, P. Bedełek, M. Ostaszewski, O. Danaci, Y. J. Patel, V. Dunjko, and J. A. Miszczak, "Enhancing variational quantum state diagonalization using reinforcement learning techniques," *New Journal of Physics*, vol. 26, no. 1, p. 013034, 2024.

[32] L. Sünkel, D. Martyniuk, D. Mattern, J. Jung, and A. Paschke, "Ga4qco: genetic algorithm for quantum circuit optimization," *arXiv preprint arXiv:2302.01303*, 2023.

[33] X. Zhu and X. Hou, "Quantum architecture search via truly proximal policy optimization," *Scientific Reports*, vol. 13, no. 1, p. 5157, 2023.

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

[35] P. Selig, N. Murphy, D. Redmond, and S. Caton, "Deepqprep: Neural network augmented search for quantum state preparation," *IEEE Access*, 2023.

[36] Y. Sun, Z. Wu, Y. Ma, and V. Tresp, "Quantum architecture search with unsupervised representation learning," *arXiv preprint arXiv:2401.11576*, 2024.

[37] M. Ostaszewski, L. M. Trenkwalder, W. Masarczyk, E. Scerri, and V. Dunjko, "Reinforcement learning for optimization of variational quantum circuit architectures," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18182–18194, 2021.

[38] P. Wang, M. Usman, U. Parampalli, L. C. Hollenberg, and C. R. Myers, "Automated quantum circuit design with nested monte carlo tree search," *IEEE Transactions on Quantum Engineering*, 2023.

[39] Z. He, X. Zhang, C. Chen, Z. Huang, Y. Zhou, and H. Situ, "A gnn-based predictor for quantum architecture search," *Quantum Information Processing*, vol. 22, no. 2, p. 128, 2023.

[40] M. Deng, Z. He, S. Zheng, Y. Zhou, F. Zhang, and H. Situ, "A progressive predictor-based quantum architecture search with active

learning," *The European Physical Journal Plus*, vol. 138, no. 10, p. 905, 2023.
[41] J. Yao, H. Li, M. Bukov, L. Lin, and L. Ying, "Monte carlo tree search based hybrid optimization of variational quantum circuits," in *Mathematical and Scientific Machine Learning*, pp. 49–64, PMLR, 2022.
[42] T. Duong, S. T. Truong, M. Pham, B. Bach, and J.-K. Rhee, "Quantum neural architecture search with quantum circuits metric and bayesian optimization," in *ICML 2022 2nd AI for Science Workshop*, 2022.
[43] W. Wu, G. Yan, X. Lu, K. Pan, and J. Yan, "Quantumdarts: differentiable quantum architecture search for variational quantum algorithms," in *International Conference on Machine Learning*, pp. 37745–37764, PMLR, 2023.
[44] 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.
[45] Y. Sun, J. Liu, Y. Ma, and V. Tresp, "Differentiable quantum architecture search for job shop scheduling problem," *arXiv preprint arXiv:2401.01158*, 2024.
[46] C.-Y. Liu and H.-S. Goan, "Reinforcement learning quantum local search," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 2, pp. 246–247, IEEE, 2023.
[47] T. Fösel, M. Y. Niu, F. Marquardt, and L. Li, "Quantum circuit optimization with deep reinforcement learning," *arXiv preprint arXiv:2103.07585*, 2021.
[48] Z. He, C. Chen, L. Li, S. Zheng, and H. Situ, "Quantum architecture search with meta-learning," *Advanced Quantum Technologies*, vol. 5, no. 8, p. 2100134, 2022.
[49] Z. He, J. Su, C. Chen, M. Pan, and H. Situ, "Search space pruning for quantum architecture search," *The European Physical Journal Plus*, vol. 137, no. 4, p. 491, 2022.
[50] Q. Chen, Y. Du, Q. Zhao, Y. Jiao, X. Lu, and X. Wu, "Efficient and practical quantum compiler towards multi-qubit systems with deep reinforcement learning," *arXiv preprint arXiv:2204.06904*, 2022.
[51] L. Ding and L. Spector, "Evolutionary quantum architecture search for parametrized quantum circuits," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 2190–2195, 2022.
[52] A. Zhang and S. Zhao, "Evolutionary-based searching method for quantum circuit architecture," *Quantum Information Processing*, vol. 22, no. 7, p. 283, 2023.
[53] L. Ding and L. Spector, "Multi-objective evolutionary architecture search for parameterized quantum circuits," *Entropy*, vol. 25, no. 1, p. 93, 2023.
[54] O. Subasi, "Toward automated quantum variational machine learning," *arXiv preprint arXiv:2312.01567*, 2023.
[55] 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, pp. 15–19, IEEE, 2023.
[56] S.-X. Zhang, C.-Y. Hsieh, S. Zhang, and H. Yao, "Neural predictor based quantum architecture search," *Machine Learning: Science and Technology*, vol. 2, no. 4, p. 045027, 2021.
[57] Y. Du, T. Huang, S. You, M.-H. Hsieh, and D. Tao, "Quantum circuit architecture search for variational quantum algorithms," *npj Quantum Information*, vol. 8, no. 1, p. 62, 2022.
[58] 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.
[59] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, C. Blank, K. McKiernan, and N. Killoran, "Pennylane: Automatic differentiation of hybrid quantum-classical computations," *arXiv preprint arXiv:1811.04968*, 2018.
[60] 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.
[61] M. C. Caro, H.-Y. Huang, 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, p. 4919, 2022.
[62] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[63] 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.
[64] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, 2016.
[65] S.-X. Zhang, J. Allcock, Z.-Q. Wan, S. Liu, J. Sun, H. Yu, X.-H. Yang, J. Qiu, Z. Ye, Y.-Q. Chen, *et al.*, "Tensorcircuit: a quantum software framework for the nisq era," *arXiv preprint arXiv:2205.10091*, 2022.
[66] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.
[67] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.