IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 4 September 2023, accepted 17 September 2023, date of publication 22 September 2023,
date of current version 28 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3318173

RESEARCH ARTICLE

Efficient Dimensionality Reduction Strategies for Quantum Reinforcement Learning

EVA ANDRÉS, M. P. CUÉLLAR, AND G. NAVARRO
Department of Computer Science and Artificial Intelligence. University of Granada, 18014 Granada, Spain

Corresponding author: Eva Andrés (e.evaandres@go.ugr.es)

This work was supported in part by the Project Quanergy under Grant TED2021-129360B-I00; in part by the Ecological and Digital Transition Research and Development Projects Call 2022 funded by MCIN/AEI/10.13039/501100011033 and European Union NextGenerationEU/PRTR; and in part by the Research and Development Projects Call 2020, Ministry of Science and Innovation, Spain, under Project PID2020-112495RB-C21.

ABSTRACT Quantum neural networks constitute one of the most promising applications of Quantum Machine Learning, as they leverage both the capabilities of classical neural networks and the unique advantages of quantum mechanics. Moreover, quantum mechanics has demonstrated its ability to detect atypical patterns in data that are challenging for classical approaches to recognize. However, despite their potential, there are still open questions such as barren plateau phenomenon and the challenges of scalability and the curse of dimensionality, which become particularly relevant in Reinforcement Learning (RL) when working in environments with high-dimensional state and action spaces. This study delves into the critical realm of representing classical data as quantum states, a topic of keen interest across the scientific community. The aim is to construct streamlined circuits for efficient execution on quantum computers and simulators using minimal qubits and entanglement gates to evade barren plateau phenomena and reducing computational times. Our investigation examines and validates the efficacy of three strategies for data management and dimensionality reduction in real-world, large-scale environments for Quantum Reinforcement Learning, particularly in energy efficiency scenarios. The techniques encompass amplitude encoding, linear layer preprocessing, and data reuploading, supplemented by trainable parameters. This research sheds light on the potential of quantum machine learning in enhancing real-world environments, including energy efficiency scenarios and showcases the capabilities of quantum neural networks in the reinforcement learning landscape.

INDEX TERMS Energy efficiency, quantum neural networks, quantum encoding, quantum reinforcement learning, variational quantum circuits.

I. INTRODUCTION

Quantum machine learning (QML) [1], [2] is one of the key fields that could reap advantages from near-term quantum devices, alongside optimization and quantum chemistry. Quantum Neural Networks (QNNs) [3], [4], are of particular interest to this manuscript owing to their significant potential in this area, and their ability to compete with current state-of-the-art classical algorithms based on artificial neural networks (ANNs) [5].

QNNs are models inspired by both neural networks and quantum mechanics. They are implemented as Parametrized Quantum Circuits (PQCs) or Variational Quantum Circuits (VQCs). QNNs have shown success in various domains, including supervised learning [6], [7], [8] and unsupervised learning [9]. While research on Quantum Reinforcement Learning (QRL) is still in its early stages, recent studies have demonstrated that QNNs can outperform classical models in Reinforcement learning (RL) scenarios, achieving better cumulative rewards with fewer parameters to be learned [10].

However, being compared to classic Artificial Neural Networks (ANNs) that have led to significant advancements in various fields, such as AlphaGo [11] and other remarkable

The associate editor coordinating the review of this manuscript and approving it for publication was Byung-Gyu Kim.

achievements [12], [13], quantum approaches to RL still have ample room for improvement. Ongoing advancements in deep reinforcement learning (DRL) and quantum computing technologies, such as Google's Cirq, IBM's Qiskit, and Xanadu's Pennylane, have contributed to the increasing interest in QML and QRL among researchers [14], [15].

VQC refers to a class of hybrid classical-quantum algorithm, which are based on a quantum circuit with learnable parameters optimized using classical algorithms. These models have the capability to approximate any continuous function, similar to classical neural networks [4], [16]. In Quantum Reinforcement Learning, VQCs can be used as function approximators for the agent's policy $\pi(a|s)$ or Q-values $Q(s, a)$. The trainable parameters are optimized using either gradient-based or gradient-free methods.

The VQC model consists of three main components: 1) An encoding layer responsible for translating classical data into quantum states to be managed by the quantum system; 2) A variational layer (ansatz) in charge of the approximation of an objective function (cost function), whose parameters are trained with classical optimizers, and 3) The definition of observables that will be considered as circuit outputs. The selection of the encoding strategic is crucial for the construction of efficient and accurate quantum neural networks. The use of an appropriate ansatz with the minimum number of qubits, layers and entanglement gates is essential to deal with noise and barren plateaus phenomenon. In QRL, these considerations are particularly important when working with feature-rich environments.

Prior research has showcased the capability of variational quantum circuits to approximate deep Q-value functions for decision-making and policy selection [17], [18]. These studies have leveraged techniques such as Grover's algorithm for encoding the policy probability distribution [19] and have even extended the approach to encompass multi-agent scenarios [20].

In the field of energy efficiency and energy management, there is a wide range of approaches in the state-of-the-art of RL. In datacenter environments, RL has been applied to tasks such as efficient resource allocation/deallocation in response to workload variations [21], [22], task scheduling [23], [24], predictive and efficient routing schemes [25], and minimizing power usage of resources [26].

In the case of energy efficiency in buildings, RL has been used for climate control in commercial buildings, such as the application of the Zap Q-learning algorithm to maintain temperature comfort within the desired range while reducing energy consumption [27]. Wang et al. proposed an RL controller based on actor/critic models using recurrent neural networks (LSTM) for HVAC control in buildings [28]. A comprehensive survey on RL applied to energy efficiency in buildings can be found in [29].

Despite the reported success of RL and Deep Reinforcement Learning (DRL) in various energy efficiency and management scenarios, most of the approaches rely on the development of simulation environments that implement digital twins of real devices, often using historical data. Simulation tools like EnergyPlus [30] and Modelica [31] play a fundamental role in training and testing models before deploying them in real energy systems. RL environments are built on top of the simulation software, as for example, Sinergym [32], Gym-EPlus [33], or EnerGym [34], which model energy consumption and control in buildings, simulate energy management in electrical vehicles (EVs) [35], [36], and simulate EV charging stations [37]. Various environments are described and analyzed in [32].

RL algorithms typically operate in large-dimensional action and state spaces, which presents a major challenge for quantum and classical computing implementations. Past studies have endeavored to address this concern by incorporating diverse encoding and dimensionality reduction strategies within the context of OpenAI Gym challenges. These strategies encompass techniques like data re-upload [38], amplitude encoding [39], and the transformation of pixel inputs into quantum data [4]. Building upon these foundations, we extend our investigation to more expansive environments, thereby assessing the scalability of Quantum Reinforcement Learning within real-world scenarios.

In this article, we show that using quantum circuits with the minimum possible number of qubits can not only accurately represent the required information, but also achieve superior performance metrics compared to circuits that require more qubits. This finding is particularly relevant as qubits are a limited resource subject to noise in quantum computers and time consuming in simulators. Furthermore, circuits with more qubits are more susceptible to falling into barren plateaus, emphasizing the importance of optimizing quantum circuits for reinforcement learning. Additionally, across all conducted experiments, it consistently emerges that a quantum model surpasses the classical model.

The presented manuscript tackles these challenges by introducing and comparing three strategies for managing classical data and reducing dimensionality in the context of Quantum Reinforcement Learning. The first strategy involves incorporating a classical linear layer before the encoding circuit, where the number of input neurons matches the variables in the state space and the number of output neurons matches the actions. The second strategy is a variant of the first, incorporating data re-uploading and scaling techniques with trainable parameters in the encoding circuit. Finally, we propose an efficient circuit that accurately represents classical information by utilizing a logarithmic scaling of the number of variables in the state space, rather than using a classical layer.

The manuscript is organized as follows: Sections II provide an overview of Artificial Neural Networks, Quantum Neural Networks, and Reinforcement Learning, respectively. Section III explains the proposed methods used to build QRL agents. Section IV describes the experimental setup and analysis of the results. Finally, he discussion and conclusions of the proposal and their results in QRL energy efficiency scenarios are presented in Sections V and VI, respectively.

II. BACKGROUND

For article self-completeness, this section provides a brief introduction to Artificial Neural Networks (ANNs), Quantum Neural Networks (QNNs), and Reinforcement Learning (RL), which are the main topics covered in this work. Firstly, Section II-A, describes the fundamentals of Artificial Neural Networks. This includes an overview of the basic structure and functioning of ANNs. Then, Section II-B, introduces Quantum Neural Networks, and finally Section II-C presents the basics of Deep Reinforcement Learning (DRL) to learn optimal policies for sequential decision-making problems.

A. ARTIFICIAL NEURAL NETWORKS

Artificial Neural Networks (ANNs) [5] are computational models inspired by the structure and functioning of the human brain. They are used for a wide variety of tasks, including classification, regression, and pattern recognition. ANNs can be viewed mathematically as a directed graph, where each node represents an artificial neuron, and each edge is assigned with a weight that determines the strength of the connection between two nodes. Based on the connection pattern or architecture, ANNs can be classified into two broad categories: feed-forward networks and recurrent networks. In feed-forward networks, the graphs have no loops and the information flows in only one direction, from input to output. Recurrent networks, on the other hand, have loops because of feedback connections, which allow them to handle time-series data and sequential processing [40].

In the most common family of feed-forward networks, called multilayer perceptron (MLP) whose graphical representation is shown in Fig. 1, neurons are organized into layers that have unidirectional connections between them. Typically, there are three or more layers: an input layer where data are presented to the network through an input buffer, an output layer with a buffer that holds the output response to a given input, and one or more intermediate or ''hidden'' layers [41].

B. QUANTUM NEURAL NETWORKS

Quantum Machine Learning (QML) models constitute one of the most promising applications of near-term quantum computers [4], [42]. One of the main tools in QML are Quantum Neural Networks (QNNs), which can be implemented as Variational Quantum Circuits (VQCs). VQCs are a class of hybrid algorithm based on a quantum circuit with trainable parameters optimized using classical algorithms. They are able to approximate any continuous function like a classical neural network [4], [16], therefore, they can perform optimization, approximation and classification tasks. The trainable parameters are optimized using gradient-based or gradient-free methods. Fig. 2 shows the general schema of a VQC. This hybrid methodology encompasses the following steps [1]:

1) **Pre-processing (CPU)**: In this step we start from a classical input data x, and targets at preparing the classical data before its encoding into a quantum state. The step encompasses classical preprocessing techniques such as normalizations, changes of scale, etc., with a classical neural network (Fig. 5).
2) **Quantum Embedding (QPU)**: The objective of the encoding layer, also known as quantum embedding, is to map the classical data to quantum states $|x\rangle$ by means of an encoding circuit using parameterized quantum gates. There are different encoding strategies depending on the problem nature, such as the basis encoding, amplitude encoding, or tensor product/angle encoding, among others. In this manuscript, we use the tensor product encoding technique [43] and probability encoding, as an efficient and novel encoding technique that uses a reduced number of qubits.
3) **Variational Layer (QPU)**: The quantum state $|x\rangle$ is the input of the variational layer (ansatz), which implements the internal hebabiour of Quantum neural Networks and it is composed of entanglement and rotation gates parameterized with learnable parameters θ. A classical optimization algorithm will optimize these parameters, aiming at minimizing a given cost function. The choice of the encoding strategy, as well as the construction of this layer, are crucial to obtain optimal results. In this work we use the same composition of rotation and entanglement gates in all experiments, and it is inspired by previous works [10], [44], [45].
4) **Measurement Process (QPU/CPU)**: In this final step, the quantum state provided by the ansatz is measured and decoded to obtain the desired output. The critical point in quantum measurement is to find an optimal way to associate outputs of the observations with target classes. The selection of the observables used to read out information from the quantum model is crucial to achieve a good performance. In this work, we calculate as output the expectation of the σ_z observable. The value of this observable is in the range $[-1, +1]$, where the bound $+1$ means that ket $|0\rangle$ is always returned, and -1 means that ket $|1\rangle$ is always returned as output. In addition, we use for the second proposal the state-vector, since the number of qubits is less than the number of outputs.
5) **Post-processing (CPU)**. This step gathers the outputs returned by the QPU and make required transformation to the data before returning the outcomes to the user and to the cost function during the learning process.
6) **Learning (CPU)**. This last step computes the cost function optimizes the ansatz parameters θ with a classic optimization algorithm (Adam, SGD, etc.). We remark that gradient-free approaches such as COLYBA or SPSA can also be used to approximate the update direction of the parameters.

The selection of the coding strategy and the observables are essential for the construction of efficient and accurate models. Building expressive but simple circuits plays a fundamental role to deal with noise and barren plateaus phenomenon, which refers to gradients that vanish exponentially, similar to what happens in classical neural networks, but which are becoming

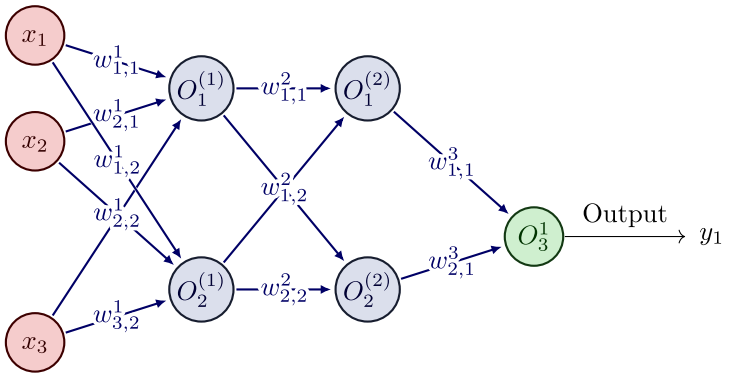


FIGURE 1. Example of a Multilayer Perceptron with 3 inputs, 2 layers with 2 neurons each, and 1 output.

even more present in VQC, and increasing exponentially with the number of qubits, layers, and entanglement gates. This effect is specially relevant when the number of input features or variables is large. In this work, we propose a compact circuit design as a potential solution to mitigate the risk of falling into a barren plateau, as well as to reduce the model complexity.

C. DEEP REINFORCEMENT LEARNING

Reinforcement learning [46] is a type of machine learning inspired by behavioral psychology. A decision-making entity, named the agent, will modify its behavior considering the rewards and penalties it receives from its interaction with an unkwnown environment. RL provides the fundamental basis to describe how intelligent agents learn autonomously to take actions in unknown environments in order to maximize the notion of cumulative reward. DRL unifies classic RL algorithms together with neural networks, and its general schema is shown in Fig. 3: When an Agent interacts with an environment, the state of the latter is unknown to the former except for the environment observation it can perceive. At a given time instant t, the environment observation is written as s_t. Then, the agent selects an action a_t from an available action set, and performs it over the environment. After that, the environment evolves to the next state and returns the new state observation s_{t+1} and a reward r_t to the agent. The reward informs the agent how good the choice of the conditioned action has been; and it is used to enhance its performance in the next interaction.

This sequential process is modeled with a Markov Decision Process (MDP). An MDP is characterized by the tuple $\langle S, A, P, r\rangle$, where S and A are the set of states and actions respectively, P is the probability of state transition $P(s'|s,a) = P\left[s_{t+1} = s'|s_t = s, a_t = a\right]$; i.e., the probability of transitioning from the state s_t at time t to state s_{t+1} at time $t+1$ using action a_t selected at time t, and $r(s_t, a_t, s_{t+1})$ is the reward function for executing action a_t at state s_t and then performing a transition from s_t to s_{t+1}.

The objective of the agent is to maximize its total reward over a sequence of agent–environment interactions τ starting at time t_0, called the return and defined in Equation (1), where γ is a hyperparameter to establish how important recent or older rewards are for the learning, known as *discount factor*. In order to maximize $R(\tau)$, it is necessary that the agent learns the best action a to be performed at a given state s, i.e., the *policy* $\pi(a|s)$, this function models the agent's behavior in the environment, and gives the probability of taking action a in a given state s. In RL we consider two important functions: the value of a state–action pair $Q(s,a)$ in Equation (2) (the expected return obtained in the trajectory starting from state s and action a), and the value of a state $V(s)$ in Equation (3) (the expected return obtained in the trajectory starting from state s). A third relevant concept is the advantage of the state–action pair $Adv(s,a)$ (Equation (4)), which returns the advantage of choosing action a in state s with respect to the other actions available in the action set for the same state s.

$$R(\tau) = \sum_{t=t_0}^{\infty} r_t \gamma^{t-t_0} \tag{1}$$

$$\begin{aligned} Q(s,a) &= \mathbb{E}_{\tau\sim\pi}\left(R(\tau)|s_t = s, a_t = a\right) \\ &= \sum_{s'} p(s'|s,a)(r(s,a,s') + \gamma \sum_{a'} \pi(a'|s')Q(s',a')) \end{aligned} \tag{2}$$

$$V(s) = \mathbb{E}_{\tau\sim\pi}\left(R(\tau)|s_t = s\right) = \sum_i R(\tau_i)\pi(a_i|s) \tag{3}$$

$$Adv(s,a) = Q(s,a) - V(s) \tag{4}$$

Deep reinforcement learning attempts to use a (deep) artificial neural network to learn the optimal policy. In the literature, two main families of algorithms have been highlighted in the last few years: deep Q-networks (DQN) [48] and policy gradient [49]. DQN approximates the function $Q(s,a)$ by training an artificial neural network, while policy gradient directly learns an optimal policy.

In this paper, we have utilized the DQN algorithm, its training is inspired by the classic Q-learning method, and aims to minimize the loss function in Equation (5), where $\hat{Q}(s,a)$ represents the a-th output value of the neural network given input s. This algorithm employs two separate networks to facilitate the learning process. The first is the DQN-deep

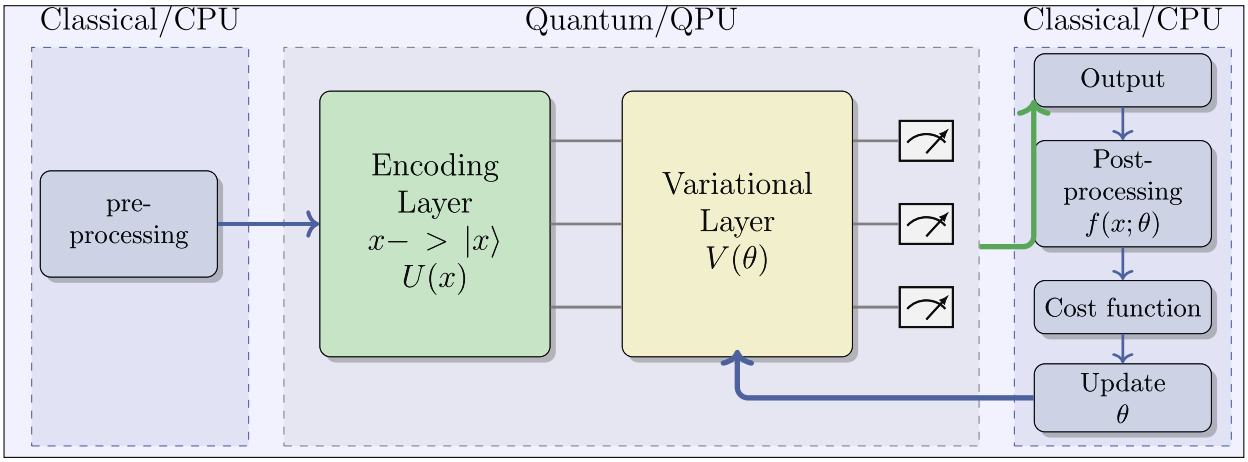


FIGURE 2. General VQC Schema. The dashed gray line encompasses the steps executed in a Quantum Processing Unit (QPU) and the dashed blue line shows the steps executed in a CPU.

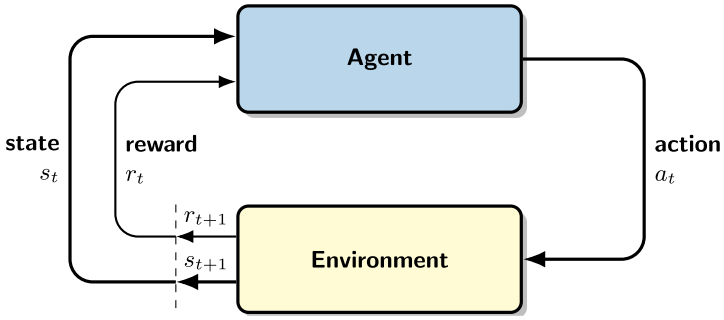


FIGURE 3. General reinforcement Learning diagram [47]. The agent at time t perceived the state s_t and considering this state selects an action a_t. The environment evolves to a state s_{t+1} returning to this agent this state and a reward r_{t+1}.

neural network, which takes the current state s as input and generates an approximate action-value function (Q-value) for each possible action. The second network is called the target network, which is a copy of the original network and is updated with a delayed frequency to reduce correlations with the target. Additionally, the algorithm employs experience replay, a biologically inspired mechanism that randomizes data, eliminating correlations in the observation sequence and smoothing over changes in the data distribution. This approach helps stabilize the learning process, which has several causes: the correlations present in the sequence of observations, the fact that small updates to Q may significantly change the policy and therefore change the data distribution, and the correlations between the action-values (Q) and the target values $r + \gamma max_{a'} Q(s', a')$ [48].

$$MSE = \sum_t \big(r(s_t, a_t, s_{t+1}) + \gamma max_{a_{t+1}} \left[\hat{Q}(s_{t+1}, a_{t+1})\right] - \hat{Q}(s_t, a_t)\big)^2 \quad (5)$$

III. METHODOLOGY

In this manuscript we adopt a hybrid quantum-classical schema for the agent with a set of parameters $\theta = (\theta_1, \ldots, \theta_k)$ that are learned through interaction with a classical (non-quantum) environment, as depicted in Fig. 4. At the beginning of an episode, the quantum–classical hybrid agent receives the environment state s_t and uses this information to select an action a_t based on its policy $\pi_\theta(a|s)$. Subsequently, the agent applies this action to the environment perceiving a reward r_t and the next environment state s_{t+1}. The policy π_θ is given by a VQC that runs on a quantum processing unit (QPU). The optimization of this policy is achieved by updating the parameters θ, which is performed by a classical learning algorithm running on a central processing unit (CPU), with the aim of minimizing a cost function [10], [45].

In this study, we propose three different QRL architectures for analysis and comparison against classical and compacted models. The architectures are illustrated in Fig. 5, Fig. 6 and Fig. 7, and the aim is to evaluate their performance relative to classical models, while demonstrating the potential of compacted models requiring fewer qubits than other hybrid architectures. The primary hypothesis is that compacted models offer superior performance due to their ability to map classical input data accurately without loss of information and their reduced susceptibility to barren plateaus problems.

Through comprehensive experiments and analyses, we aim to demonstrate the efficacy and trade-offs of these architectures and provide valuable insights into their practical

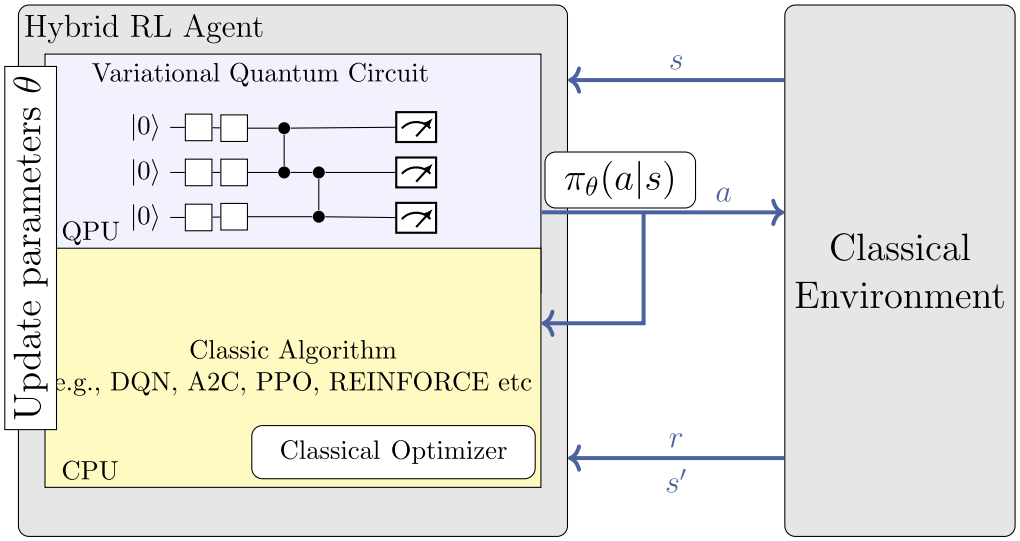


FIGURE 4. QRL schema. Example of hybrid agent and classical/quantum interaction to optimize the trainable parameters of the policy using a classical learning algorithm.

application in real-world settings. Specifically, we present three hybrid quantum-classical architectures.

The first model utilizes a hybrid architecture for the agent's policy, using a linear layer that serves to reduce the dimensionality of the state space. The number of input neurons of the linear layer corresponds to the dimension of the state space, while the number of output neurons corresponds to the dimension of the action space. As a result, the quantum circuit that follows this layer contains a number of qubits equal to the dimension of the action space, as depicted in Fig. 5. Tensor product encoding is employed to convert the observation s_t into a quantum state $|s_t\rangle$. In this work, we calculate as output the expectation of the σ_z observable, i.e., the operation $\langle\phi|\,\sigma_z\,|\phi\rangle$, for each qubit. The value of this observable is in the range $[-1, +1]$, where the bounds $+1$ means that ket $|0\rangle$ is always returned, and -1 means that ket $|1\rangle$ is always returned as output.

The second model builds upon the first one by incorporating the data-reuploading and scaling technique. As discussed in section II-B, an important aspect of designing VQCs is how classical data can be encoded. While several encoding methods have been proposed in the literature [43], recent studies on the expressiveness of quantum circuits suggest the advantage of repeated encodings also referred to as data-reuploading [50], [51], in a similar way that a classic ResNet architecture works [5]. The use of this technique, in conjunction with trainable weights on inputs, can lead to the creation of richer models that have the potential to approximate the objective function more accurately. Quantum models have a natural ability to learn periodic functions in data, then, using repeated gate encoding can increase the frequency spectrum of the model, allowing it to represent a wider range of data. However, it is important to ensure that the data is scaled appropriately to fit within the period of the function being learned, thereby enabling the model to learn and represent the data effectively. Furthermore, classical neural networks take several times the same input when processing the data within hidden layers, therefore and considering the non-cloning theorem, the use of data-reuploading makes sense [50], [51], [52]. This technique enhances the model's ability to represent classical data accurately and approximate the objective function effectively. It involves multiple re-uploadings of the encoding layer, where each re-uploading utilizes trainable parameters to improve the data encoding process. This model is structured with interleaved encoding layers that consist of R_x gates, and variational layers that consist of R_y, R_z, and C_z gates. The layout of this model is depicted in Fig. 6. Each layer is comprised of one encoding circuit and one ansatz circuit, repeated for the number of layers specified.

Finally, the third model uses a variational quantum circuit with $log_2 N$ qubits to encode N features without loss of information, in contrast to a linear layer. This encoding technique is known as Amplitud encoding, which is achieved applying cRy and Ry rotations over the qubits. To be more specific, a combination of gates is employed to map $|0\ldots.0\rangle \rightarrow |x\rangle = \sum_i a_i\,|i\rangle$ where $|i\rangle$ is the *ith* entry of the computational basis. The data used in this approach must be normalized, such that $\sum_i |a_i|^2 = 1$. Fig. 8 depicts an example of the circuit encoding for a system with two qubits. As an example, four features can be represented as follows:

$$|00\rangle \rightarrow a_{00}\,|00\rangle + a_{01}\,|01\rangle + a_{10}\,|10\rangle + a_{11}\,|11\rangle \quad (6)$$

With two qubits previously initialized to $|0\rangle$ and applying a single qubit rotation gate $U(\theta)$ defined in Equation (7) on the fist qubit, we OBTAIN:

$$U(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & -\cos\theta \end{pmatrix} \quad (7)$$

$$U(\theta_1)\,|0\rangle \otimes |0\rangle \quad (8)$$

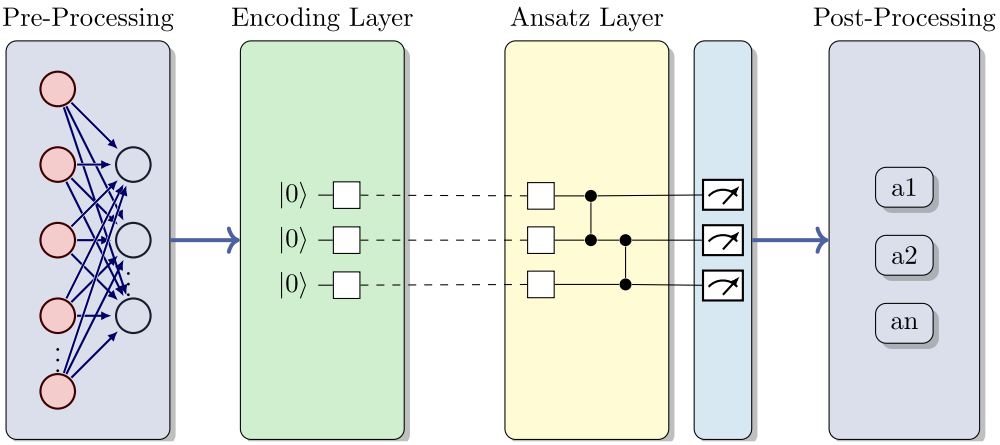


FIGURE 5. First proposed architecture. Hybrid classical/quantum neural network. The pre-processing stage utilizes a linear layer, for dimensionality reduction purposes, that is dimensioned with as many input neurons as the observation space dimension and as many output neurons as the action space dimension. This ensures that the expected output dimension is consistent with the dimension used in the quantum circuit. The output of the pre-processing stage is then fed into the encoding circuit, followed by the variational layers. The final output is obtained by measuring each qubit in the computational z-basis and post-processed using a Softmax function.

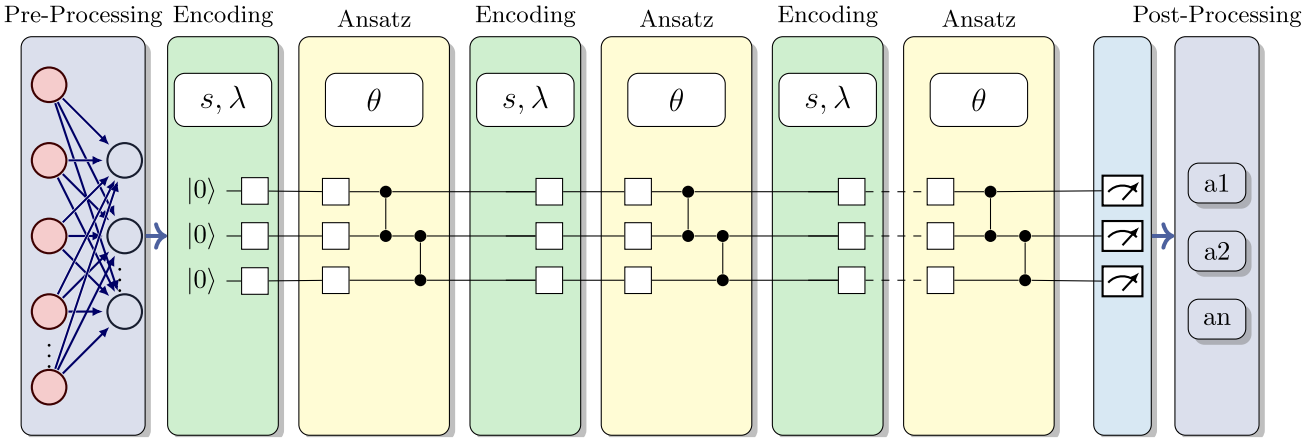


FIGURE 6. Second proposed architecture. Hybrid classical/quantum neural network. This architecture incorporates a linear layer to reduce the dimensionality of the input data, along with data re-uploading and scaling techniques. The encoding circuit of this architecture is composed of several unitary gates that are interleaved with the ansatz circuits. The λ-input scaling parameters and the θ-weights are the learnable parameters for the encoding and ansatz circuits, respectively. It is worth noting that the number of encoding and ansatz circuits is determined by the number of layers defined in the model, which is three layers in the example shown in this figure.

$$= \sqrt{1 - (|a_{10}|^2 + |a_{11}|^2}\,|00\rangle + \sqrt{|a_{10}|^2 + |a_{11}|^2}\,|10\rangle \tag{9}$$

$$= \sqrt{|a_{00}|^2 + |a_{01}|^2}\,|00\rangle + \sqrt{|a_{10}|^2 + |a_{11}|^2}\,|10\rangle \tag{10}$$

where θ_1 is given by:

$$\arcsin\left(\sqrt{|a_{10}|^2 + |a_{11}|^2}\right) \tag{11}$$

Then, applying a controlled-rotation $U(\theta_2)$ on the second qubit, with as control the fist qubit, where θ_2 is chosen as:

$$\cos(\theta_2) = \frac{a_{00}}{\sqrt{|a_{00}|^2 + |a_{01}|^2}}, \sin(\theta_2) = \frac{a_{01}}{\sqrt{|a_{00}|^2 + |a_{01}|^2}} \tag{12}$$

Finally, applying controlled-rotation $U(\theta_3)$ on the second qubit, with the first qubit being 1 as control. Choosing θ_3 such that:

$$\cos(\theta_3) = \frac{a_{10}}{\sqrt{|a_{10}|^2 + |a_{11}|^2}}, \sin(\theta_3) = \frac{a_{11}}{\sqrt{|a_{10}|^2 + |a_{11}|^2}} \tag{13}$$

Therefore, the complete operation will be:

$$\begin{aligned}&\sqrt{|a_{00}|^2 + |a_{01}|^2}\,|0\rangle \otimes U(\theta_2)\,|0\rangle \\ &\quad + \sqrt{|a_{10}|^2 + |a_{11}|^2}\,|1\rangle \otimes U(\theta_3)\,|0\rangle \\ &= a_{00}\,|00\rangle + a_{01}\,|01\rangle + a_{10}\,|10\rangle + a_{11}\,|11\rangle\end{aligned} \tag{14}$$

The diagram in Fig. 9 depicts an example of the encoding process for 2 qubits and 4 features. The general algorithm used for amplitude encoding is outlined in Algorithm 1.

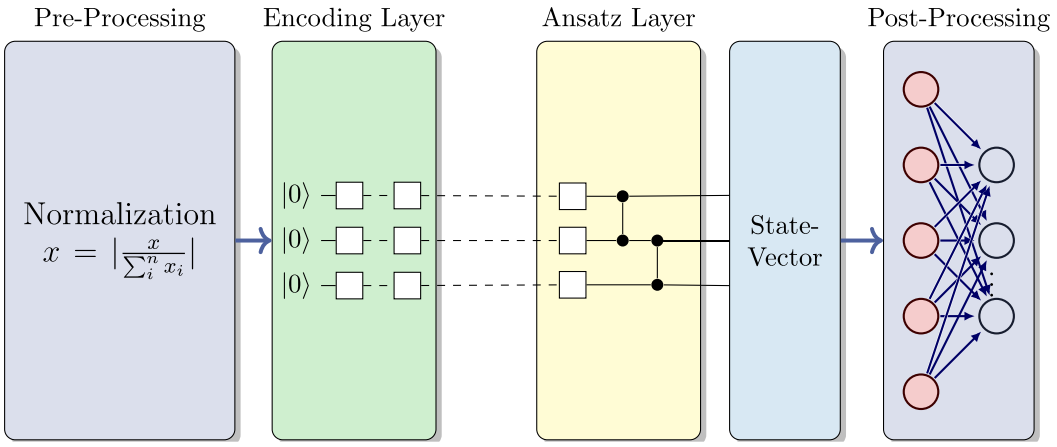


FIGURE 7. Third proposed architecture. This architecture involves a pre-processing step in which the classical data is normalized, a necessary requirement for the subsequent encoding strategy. Then, the encoding circuit illustrated in Fig. 8 is applied, which employs only $log_2 N$ qubits, where N is the dimension of the observation space, in order to obtain the corresponding quantum state that will feed the ansatz circuit. Finally, the resulting state-vector of this quantum system is subject to post-processing using a classical linear layer, which transforms the dimension obtained with 2^n where n is the number of qubits, into the action space dimension.

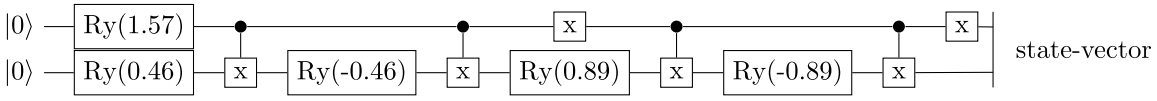


FIGURE 8. Encoding circuit for the data n = 0.2, 0.3, 0.4, 0.1). Using the decomposition of cRy gates.

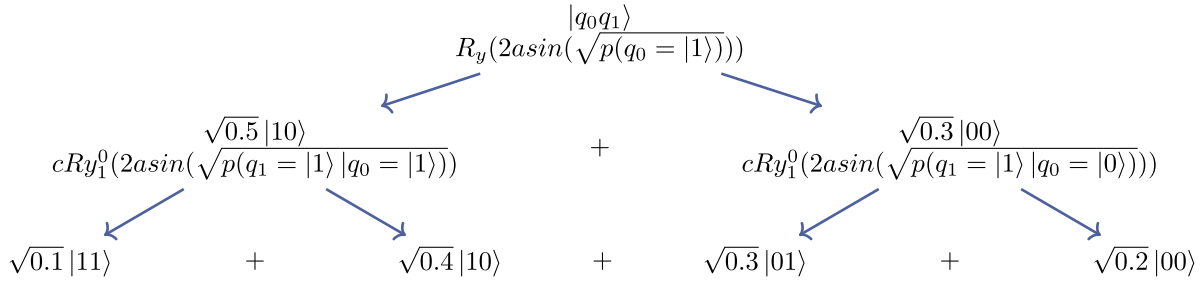


FIGURE 9. Example of amplitude embedding and probability amplitudes decomposition for the data n = (0.2, 0.3, 0.4, 0.1).

Regarding the ansatz layer, it is worth noting that all models employ a combination of rotation and entanglement gates, which have demonstrated favorable outcomes in various algorithms [44]. The primary aim is to construct circuits with a high level of expressivity, enabling a close approximation of the objective function. This is achieved by employing rotation and entanglement gates, as well as increasing the number of layers in the circuit. However, there is a point of diminishing returns, where increasing these components beyond a certain limit can actually have a negative impact on the convergence of the algorithm. We remark that the selection of encoding and read-out strategies significantly impact the convergence of the algorithm. In our first proposed approach, we employed PauliZ gates on all qubits for measurement, aligning the number of qubits with the number of outputs. In contrast, for the second approach, where the number of outputs exceeds the number of qubits, we opted for a different strategy. We leveraged the state-vector in the measurement process and employed a linear layer to ensure the desired output dimensionality. Specifically, we employed 2^N neurons for N qubits, and matched the number of output neurons to the dimensionality of the action space.

IV. EXPERIMENTATION

In the literature we can find different energy-efficiency and management scenarios. Simulation tools such as Energy Plus [30] or Modelica [31] play a fundamental role in the training and testing of models before deploying them in real energy systems. RL environments are built on top of this simulation software, for example, Sinergym [32], Gym-EPlus [33], or EnerGym [34] to model energy consumption and control in buildings.

In this section, we will test quantum reinforcement learning algorithms in various energy efficiency scenarios.

Algorithm 1 Amplitude Encoding Algorithm

Input: $NQubits, x$ ▷ x is the classicData x, NQubits is the number of qubits
Output: $StateVector$
Require: $x = |\frac{x}{\sum_i^n x_i}|$ ▷ Normalization of ClassicData
$ConditionList \leftarrow \{\epsilon, s | s \in (1|0)|(1|0)\{2\}|(1|0)\{3\} \ldots |(1|0)\{NQubits - 1\}\}$ ▷ for 2 qubits: ['','1','11','10','0','01','00']
for *basis* in [0 to $len(x)$] **do**
  $basisStates \leftarrow binary(basis, NQubits)$ ▷ Building the basis States for ClassicData
end for
for *conditioning* in *ConditionList* **do**
  $currentq \leftarrow len(conditioning)$
  $sum \leftarrow 0$
  $prob1 \leftarrow 0$
  for *state* in *basisStates* **do**
    if state startsWith(conditioning) **then**
      $p \leftarrow x[state]$ ▷ the value of x normalized in state (basisState)
      $sum \leftarrow sum + p$
      if state[currentq] = '1' **then**
        $prob1 \leftarrow prob1 + p$
      end if
    end if
  end for
  $prob1 \leftarrow prob1/sum$ ▷ After iterating over the basis states for the current conditioning, we compute the angles.
  $circuit \leftarrow \epsilon$
  if conditioning = ϵ **then** ▷ Ry over first angle and qubit 0
    $circuit \leftarrow circuit + Ry(2asin(\sqrt{prob1}), qubit = 0)$
  else
    for *j* in *conditioning* **do**
      if $conditioning[j] =' 0'$ **then**
        $circuit \leftarrow circuit + PauliX(qubit = j)$
      end if
    end for
    $controls \leftarrow [0, 1, \ldots, currentq]$ ▷ cRy gates implementation for the rest of angles
    $circuit \leftarrow circuit + Ry(asin(\sqrt{prob1}, qubit = currentq)$
    $circuit \leftarrow circuit + ControlledPauliX(controlQubits = controls, targetQubit = currentq)$
    $circuit \leftarrow circuit + Ry(-asin(\sqrt{prob1}, qubit = currentq)$
    $circuit \leftarrow circuit + ControlledPauliX(controlQubits = controls, targetQubit = currentq)$
    for *j* in *conditioning* **do**
      if $conditioning[j] =' 0'$ **then**
        $circuit \leftarrow circuit + PauliX(qubit = j)$
      end if
    end for
  end if
end for

Our goal is to compare the results of all proposed models, including the classical MLP, and verify whether QRL and quantum technologies are appropriate for energy efficiency in environments with a large number of variables. Through this experiments, we aim to demonstrate that efficient encoding strategies and hybrid techniques can successfully handle high dimensionality, enabling us to achieve better results than their classical counterparts. We evaluate the influence of different design choices on learning performance through numerical simulations in three classical benchmarking environments related to energy efficiency. These environments are available in *Sinergym*, an EnergyPlus-supported simulator that contains reference problems for energy efficiency and HVAC control in buildings and facilities. Section IV-A addresses a use case of classic control of HVAC energy-saving in buildings; Section IV-B solves a datacenter environment; and, finally, Section IV-C addresses a warehouse environment with a small office.

Lastly, using cumulative reward as the basic metric for analysis and comparison of QRL models, which are trained with the same DQN algorithm, allows us to assess QRL performance uniformly and fairly across all three use cases, and also enables us to compare QRL with classical RL regardless of the specific use case, making it easier to analyze the results in a more general and comprehensive manner.

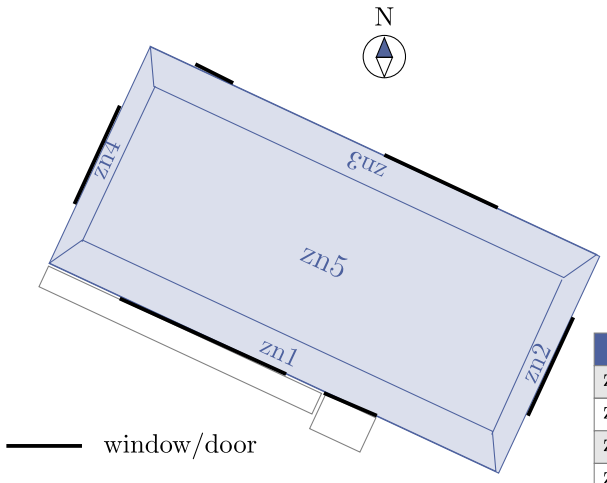
A. USE CASE 1: 5-ZONES-BUILDING

1) PROBLEM STATEMENT

The target building in this study is the environment *Eplus-demo-v1* and focuses on the scenario *5ZoneAutoDx* [32]. This building is based in Pittsburgh, PA, USA, consists of a single-floor rectangular of 100 ft with five zones, four exterior and one interior, regularly occupied by office workers; oriented 30 degrees east of north (see Fig. 10). The overall building height is 10 feet; there are windows on all four facades, single and double panel constructions with 3mm and 6mm glass and argon or 6mm or 13mm air gap; the window-to-wall ratio is approximately 0.29. The south and north facades have glass doors. The walls are wooden shingles over plywood, R11 insulation, and Gypboard; The south wall and door have overhangs. The roof is a gravel built-up roof with R-3 mineral board insulation and plywood sheathing and finally, the floor area is 463.6 m^2 (5000 ft^2).

A state s in this environment represents a sequence of historical observations of this building (e.g., outdoor air temperature and room temperatures). The goal is to maximize an aggregation of KPIs (key performance indicators) regarding energy usage and user comfort. The detailed features are described below:

- **State space:** The state space contains 20 features; 16 are described in Table 1 and 4 are reserved for the environment.
- **Action space:** The action space contains a set of 10 discrete actions described in Table 2. The bounds for



	Area (m^2)	Hight (m^2)
zn1	100	2.4
zn2	43	2.4
zn3	100	2.4
zn4	43	2.4
zn5	186	2.4

FIGURE 10. Zone building plan.

TABLE 1. Observation variables for Use case 1, comprising 16 variables, with an additional 4 empty variables reserved for cases where they are necessary based on the specific requirements of the problem.

Variable Name	Units
Site Outdoor Air Drybulb Temperature	°C
Site Outdoor Air Relative Humidity	%
Site Wind Speed	m/s
Site Wind Direction	degree from north
Site Diffuse Solar Radiation Rate per Area	W/m^2
Site Direct Solar Radiation Rate per Area	W/m^2
Zone Thermostat Heating Setpoint Temperature	°C
Zone Thermostat Cooling Setpoint Temperature	°C
Zone Air Temperature	°C
Zone Thermal Comfort Mean Radiant Temperature	°C
Zone Air Relative Humidity	%
Zone Thermal Comfort Clothing Value	Icl (clo)
Zone Thermal Comfort Fanger Model PPD	%
Zone People Occupant Count	count
People Air Temperature	°C
Facility Total HVAC Electricity Demand Rate	W

heating and cooling setpoint temperature are [15, 22.5] and [21, 30] respectively.

- **Reward function:** Reward is always negative. This means that the perfect reward would be 0, indicating perfect power consumption and temperature comfort. In addition, there are two temperature comfort ranges (winter and summer) and a weight of energy and comfort. Therefore, the total reward is calculated as the sum of reward of energy and comfort multiplied by their respective relevance weights, given by W_{energy}, as a hyperparameter. The reward function is the same for all three use cases but can be customized and added to the environment.

$$reward = W_{energy} \times reward_{energy} + (1.0 - W_{energy}) \times reward_{comfort} \quad (15)$$

In this experimentation we design a classic multilayer perceptron (MLP) neural network agent with 20 inputs

TABLE 2. Action variables.

Variable Name	Heating Setpoint	Cooling Setpoint
0	15	30
1	16	29
2	17	28
3	18	27
4	19	26
5	20	25
6	21	24
7	22	23
8	22	22
9	21	21

(environment state dimension) and 10 outputs (environment action dimension), training the agent in the *Eplus-demo-v1* environment with the Deep Q-Network algorithm (DQN) [48]. After that, we also train three different quantum agents designed with the methodologies described in Section III, and configure each model as described in the following section. Finally, the settings for the environment and DQN algorithm remain the same for the classic baseline and quantum agents, to make a fair performance comparison.

2) EXPERIMENTAL SETTINGS

In this section we evaluate the three proposed architectures for quantum agents, comparing them to the classical feedforward multilayer perceptron (MLP). The first implementation for the agent's policy is a hybrid architecture with one linear layer and one variational quantum circuit 5; the second proposal using the same architecture adding data-reuploading and scaling methods 6 and finally the third approach using encoding strategies achieves to use a compacted and simplest model but very expressive representing all the information included in the features of the environment, showed in Fig. 7.

In the present study, we employed the DQN RL method to train our agents. To ensure the validity and robustness of

our results, we conducted 10 independent runs, each with distinct initial random seeds. The DQN algorithm utilizes two distinct networks to facilitate the learning process. The first network is the DQN-deep neural network, which takes the current state s as input and generates an approximate action-value function (Q-value) for each possible action. The second network is known as the target network, and it is parameterized by a copy of the original network. This target network is initialized with the same weights as the DQN network and is updated with a delayed frequency, thereby reducing correlations with the target. Additionally, uses a biologically inspired mechanism termed experience replay that randomizes over the data, thereby removing correlations in the observation sequence and smoothing over changes in the data distribution. This approach helps stabilize the learning process, which has several causes: the correlations present in the sequence of observations, the fact that small updates to Q may significantly change the policy and therefore change the data distribution, and the correlations between the action-values (Q) and the target values $r + \gamma max_{a'} Q(s', a')$ [48].

The configuration details for all models are summarized in Table 3. In the first solution, the quantum agent utilizes a total of 10 qubits, with each qubit corresponding to a possible action selection. Initially, the environment state, consisting of 20 values, is inputted to a linear layer with 20 input neurons employed to compute a linear combination, enabling dimensionality reduction and facilitating subsequent processing with 10 qubits. This is followed by a ReLU activation function. Subsequently, the input information is scaled to the range of $[-\pi/2, \pi/2]$ prior to being fed into the quantum embedding process. The encoded information from the input layer is then passed through five consecutive quantum variational layers, as described in Section III. Finally, the expectation value of the σ_z operator is separately measured for each qubit, yielding the Q-value associated with each action. In this particular case, the Q-networks employed in the algorithm consist of a total of 300 parameters. This parameter count is calculated as follows: $20 \times 10 = 200$ (pre-processing layer) + 5 (number of layers) $\times$ 2 (learning parameters used in Ry and Rz) $\times$ 10 (number of qubits). The second solution, which is a variant of the previous approach, incorporates data-reuploading and scaling methods. In this case, the Q-networks consist of a total of 350 parameters: $20 \times 10 = 200$ (pre-processing layer) + 5 (number of layers) $\times$ 1 (learning parameter used in Rx for encode the data) $\times$ 10 (number of qubits) + 5 (number of layers) $\times$ 2 (learning parameters used in Ry, Rz for ansatz circuit) $\times$ 10. In the third solution, the architecture consists of encoding and ansatz circuits composed of 5 qubits, where the number of qubits is determined by $log_2 N$, with N representing the number of features in the observation space, in this particular environment, N is equal to 20. The total number of parameters in the third solution's architecture is 225: 15 (number of layers) $\times$ 3 (learning parameters used in Rx, Ry and Rz for ansatz circuit) $\times$ 5 (number of qubits). Finally, the classical network consists of an input layer with 20 neurons, a hidden layer of 100 neurons, and an output layer with 10 neurons. The total number of parameters in this case is 3110: 20×100 (weights) + 100 (bias) + 10×100 (weights) + 10 (bias).

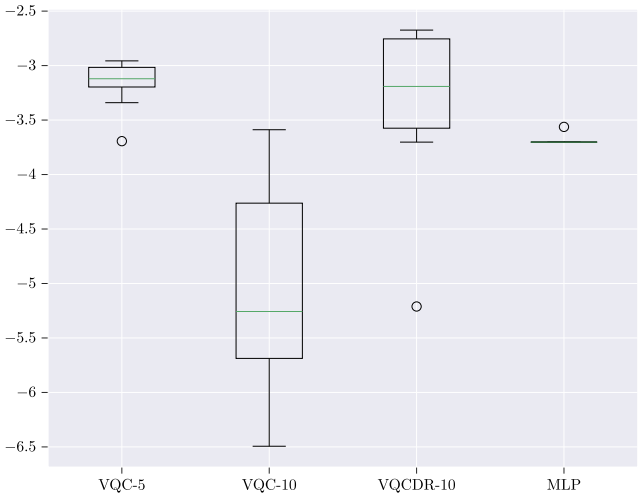


FIGURE 11. Boxplots illustrating the distribution of average total reward obtained from four different approaches in Use case 1, arranged from left to right as follows: a VQC with Amplitude encoding (5 qubits), VQC with linear layer pre-processing, VQC with data-reuploading and scaling methods, and MLP.

FIGURE 12. Learning curves obtained from four distinct methods in Use Case 1. The approaches are: VQC with Amplitude Encoding using 5 qubits, VQC with a linear layer at pre-processing, a variant of the latter approach which incorporates data-reuploading and scaling methods, and the use of MLP.

It is worth highlighting that the total number of parameters needed for the quantum approach is significantly lower than the number of parameters required to train the classical agent. This observation suggests that, in theory, the quantum agent exhibits lower complexity compared to its classical counterpart. The reduced parameter count in the quantum agent implies a potential advantage in terms of computational efficiency and resource utilization.

3) RESULTS

In this section we present the results from the four models. Table 4 provides a comprehensive summary of the average,

TABLE 3. Structure of DQN classical/quantum networks used in use case 1.

	MLP	VQC	VQC (Trainable input weight -Data-Reuploading)	VQC (Amplitude Encoding)
No. of qubits	-	10	10	5
Inputs	20	20. Scaled to $[-\pi/2, \pi/2]$	20. Scaled with λ	20. Normalized to 1
Pre-processing	-	Linear (20 units, 10 units)	Linear (20 units, 10 units)	-
Quantum embedding	-	Tensor product encoding	Tensor product encoding	Amplitude encoding (5 qubits)
Intermediate Layers	Linear (20 units, 100 units) ReLU	Variational Layer (10 qubits) Tahn	Variational Layer (10 qubits) Tahn	Variational Layer (5 qubits) ReLU
Output	Linear (100 units, 10 units)	$\sigma_z(q_i)$ for each qubit q_i	$\sigma_z(q_i)$ for each qubit q_i	state-vector
Post-processing	-	-	Softmax	Linear($2^{Nqubits}$, 10)+Softmax
Training algorithm	RMSProp (lr=0.001)	RMSProp (lr=0.009)	RMSProp ($lr[1 \times 10^{-2}, 1 \times 10^{-4}]$)	RMSProp (lr=0.01)

TABLE 4. Results obtained by MLP and VQC agents in the use case 1. Column 1: MLP agent (classic) and the different VQC agents proposed (quantum); Column 2: Average total reward after training; Column 3: Best total reward after training; Column 4: Worst total reward after training; Column 5: Computational time in seconds; Column 6: Total reward (test with deterministic policy).

Model	Avg. Total Reward	Best Total Reward	Worst Total Reward	Time (s)	Test Reward
VQC-10	−5.03	−3.59	−6.49	947.37	−5.03
VQC-10 (trainable input weight and Data-Reuploading)	−3.32	−2.67	−5.21	2214.63	−3.32
VQC-5 (Amplitude Encoding)	−3.16	−2.96	−3.69	3489.55	−3.16
MLP (Classic)	−3.69	−3.56	−3.70	661.88	-3.69

best, and worst total accumulated reward obtained by the MLP and the VQC models, along with the corresponding computational time required for each experiment, measured in seconds. The evaluation of the results is based on the total accumulated reward obtained by the agent in each episode, which encompasses the summation of the reward function described in Equation (15) over all steps of an episode. The average accuracy is defined as the mean total accumulated reward calculated across the 10 different runs conducted.

One significant observation derived from the outcomes presented in Table 4 is consistently, at least one quantum agent outperforms the MLP model across all evaluated categories. However, it is noteworthy that the computational time for the VQC models was comparatively higher due to the utilization of simulators instead of real quantum computers. Despite this trade-off in computation time, the quantum agents demonstrated superior performance across all evaluated metrics, solidifying their effectiveness and potential in this domain.

The results clearly indicate that use of amplitude encoding, such as the third quantum approach, outperform those that do not utilize this technique in terms of average, worst and test reward. This finding highlights the significant potential of amplitude encoding as coding strategy and underscores the importance of matching the use of compacted and expressive circuits to the learning task at hand.

Based on the boxplot analysis presented in Fig. 11, several conclusions can be drawn. Firstly, the VQC with amplitude encoding, as indicated by its narrow interquartile range, demonstrates high robustness. Although it shows one outlier, this model consistently performs well across different experimental runs. Secondly, the MLP model exhibits the presence of an outlier and a concentration of values in a single region, indicating its robustness even in the presence of an outlier. The fact that most of the data points are clustered together suggests that the MLP model is relatively stable and consistent in its performance. Despite this outlier, both models demonstrate resilience and remain robust in their performance. It is closely followed by VQC with trainable input weights and Data-Reuploading. The VQC with 10 qubits, it is characterized by a higher variance in the data, exhibiting the poorest performance among the models considered. The data points for this model have a wider dispersion compared to the previous models, indicating a larger range of variability probably. This could be attributed to the model's sensitivity to initial conditions and hyperparameters, leading to different outcomes in each experimental run.

Finally, from Fig. 12, it is noticeable that the model utilizing amplitude encoding displays the best learning curve. This is indicated by a decreasing standard deviation of total rewards across episodes in the final iterations, suggesting a more stable and consistent learning process. Following closely is the Data-reuploading and scaling methods, which also demonstrate a relatively favorable learning curve, albeit with a slightly higher standard deviation. These findings highlight the effectiveness of amplitude encoding and the potential benefits of incorporating data-reuploading and scaling techniques in improving the learning performance of quantum agents.

B. USE CASE 2: DATACENTER

1) PROBLEM STATEMENT

The target building in this study is the *Eplus-datacenter-mixed-discrete-v1* environment, which is one of the environments available in the *Sinergym* simulator [32]. Specifically, the study focuses on the scenario *2ZoneDataCenterHVAC*.

This building model is based in NYC, USA. It comprises two zones: the West zone, with a dimension of 232.26 m^2, and the East zone, with a dimension of 259.08 m^2. The building structure consists of mass walls, a regular slab floor, and a regular roof without windows. The IT equipment in the data center is air-cooled. Each zone within the building is served

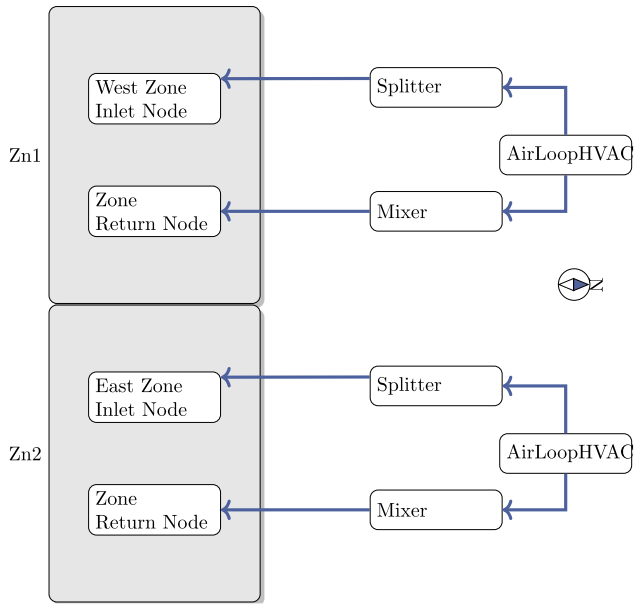


FIGURE 13. The figure illustrates the HVAC components used in the Zone Datacenter. The components include Air Economizer, Direct Evaporative Cooler, Indirect Evaporative Coolers, Single Speed DX Cooling Coil, Chilled Water Coil, and VAV with No Reheat Air Terminal Unit. These components play a crucial role in the data center's cooling system, providing different cooling mechanisms and control options to maintain optimal temperature conditions.

by its own airloopHVAC system. The air loops are equipped with a DX cooling coil and a chilled water cooling coil, which are responsible for the cooling process in the respective zones (see Fig. 13).

This specific building configuration provides a realistic representation of a data center environment, allowing for the analysis and evaluation of energy efficiency and HVAC system performance in such settings.

The state space is presented in Table: 5, while the action space remains consistent with Use case 1 as described in Section IV-A. The reward function is defined by equation (15).

In this particular use case, we developed a classic multilayer perceptron (MLP) neural network agent with 29 input nodes representing the environment state, and 10 output nodes. The agent is trained in the Eplus-datacenter-mixed-discrete-v1 environment. Additionally, we train three quantum agents using the methodologies outlined in Section III. It is important to note that the environment settings and DQN algorithm parameters are kept consistent across all agents to ensure a fair performance comparison.

2) EXPERIMENTAL SETTINGS

For this experiment, we employ four models with identical architectures as those used in the previous use case, as described in Section III. The first model is a hybrid model, which incorporates a linear layer for dimensionality reduction. In this model, the Q-networks comprise a total of 390 parameters:$29 \times 10 = 290$ (pre-processing layer) + 5 (number of layers) $\times$ 2 (learning parameters used in Ry and Rz) $\times$ 10 (number of qubits). The second solution, a variant of the previous model, incorporates data-reuploading and scaling techniques. In this model, the Q-networks are composed of 440 parameters: $29 \times 10 = 290$ (pre-processing layer) + 5 (number of layers) $\times$ 1 (learning parameter used in Rx for encode the data) $\times$ 10 (number of qubits) + 5 (number of layers) $\times$ 2 (learning parameters used in Ry, Rz for ansatz circuit) $\times$ 10. The third solution utilizes an architecture composed of encoding and ansatz circuits, with 5 qubits corresponding to the logarithm base 2 of the number of features in the observation space (N = 29 in this environment). The number of parameters in this model is 250: 25 (number of layers) $\times$ 2 (learning parameters used in Rx, Ry for ansatz circuit) $\times$ 5. Finally, the classical network consists of an input layer with 29 neurons, a hidden layer with 100 neurons, and an output layer with 10 neurons. Consequently, the total number of parameters in this classical network is 4010: 29×100 (weights) + 100 (bias) + 10×100 (weights) + 10 (bias). The configuration details for these models are presented in Table 9.

At this stage, we can infer that the quantum proposal requires significantly fewer total parameters compared to classical agent. Consequently, the VQC models and, therefore, the quantum agents exhibit a more compact structure when compared to their classical counterparts.

3) RESULTS

The performance results are presented in Table 7, which provides an overview of the average, best, and worst total accumulated reward achieved by the three quantum agents and the MLP. Additionally, the table includes the computation time of each experiment, measured in seconds. The average accuracy refers to the average total accumulated reward obtained from 10 independent runs, where different but fixed seeds were used to ensure reproducibility.

One notable observation from the results presented in Table 7 is the consistent performance of at least one quantum agent compared to the MLP model across all evaluated categories. Despite the higher computation time associated with VQC due to the use of simulators instead of real quantum computers, the quantum agents demonstrate superior performance. In particular, the VQC model with 5 qubits, utilizing efficient encoding with amplitude encoding, excels for execution time, mirroring the findings from the previous use case. This model showcases the ability to maintain expressiveness and effectively approximate the objective function even with a smaller number of qubits ($log_2 N$ for N features). As a result, it proves to be a highly competitive and promising option for the given task.

Continuing with the analysis and examining the boxplot in Fig. 14, we can observe that the VQC agent with 5 qubits exhibits a relatively moderate interquartile range, indicating a reasonable range of variability in its performance. Although the range is not extremely narrow, the absence of outliers and the proximity of the median to the center suggest a consistent

TABLE 5. Observation Variables for Use case 2, comprising 25 variables, with an additional 4 empty variables included to accommodate specific usage scenarios that require them.

Variable Name	Units
Site Outdoor Air Drybulb Temperature (Environment)	°C
Site Outdoor Air Relative Humidity (Environment)	%
Site Wind Speed (Environment)	m/s
Site Wind Direction	degree (Environment)
Site Diffuse Solar Radiation Rate per Area (Environment)	W/m^2
Site Direct Solar Radiation Rate per Area (Environment)	W/m^2
Zone Thermostat Heating Setpoint Temperature (West Zone)	°C
Zone Thermostat Cooling Setpoint Temperature (West Zone)	°C
Zone Air Temperature (West Zone)	°C
Zone Thermal Comfort Mean Radiant Temperature (West Zone People)	°C
Zone Air Relative Humidity (West Zone)	%
Zone Thermal Comfort Clothing Value (West Zone People)	Icl (clo)
Zone Thermal Comfort Fanger Model (West Zone People) PPD	%
Zone People Occupant Count	count (West Zone)
People Air Temperature (West Zone People)	°C
Zone Thermostat Heating Setpoint Temperature (East Zone)	°C
Zone Thermostat Cooling Setpoint Temperature (East Zone)	°C
Zone Air Temperature (East Zone)	°C
Zone Thermal Comfort Mean Radiant Temperature (East Zone People)	°C
Zone Air Relative Humidity (East Zone)	%
Zone Thermal Comfort Clothing Value (East Zone People)	Icl (clo)
Zone Thermal Comfort Fanger Model PPD (East Zone People)	%
Zone People Occupant Count (East Zone)	count
People Air Temperature (East Zone People)	°C
Facility Total HVAC Electricity Demand Rate (Whole building)	W

TABLE 6. Structure of DQN classical/quantum networks used in Use case 2.

	MLP	VQC	VQC (Trainable input weight-Data-Reuploading)	VQC (Amplitude Encoding)
No. of qubits	-	10	10	5
Inputs	29	29. Scaled to $[-\pi/2, \pi/2]$	29. Scaled with λ	29. Normalized to 1
Pre-processing	-	Linear (29 units, 10 units)	Linear (29 units, 10 units)	-
Quantum embedding	-	-	Tensor product encoding	Amplitude encoding
Intermediate Layers	Linear (10 units, 100 units)	Variational Layer (10 qubits)	Variational Layer (10 qubits)	Variational Layer (5 qubits)
	ReLU	Tahn	Tahn	ReLU
Output layer	Linear (100 units, 10 units)	$\sigma_z(q_i)$ for each qubit q_i	$\sigma_z(q_i)$ for each qubit q_i	state-vector
Post-processing	-	Softmax	Softmax	Linear($2^{Nqubits}$, 10)+Softmax
Training algorithm	RMSProp (lr=0.001)	RMSProp (lr=0.01)	RMSProp ($lr[1 \times 10^{-2}, 1 \times 10^{-4}]$)	RMSProp(lr=0.03)

and stable performance. This suggests that this agent achieved the best result in robustness. Following closely is the VQC agent with 10 qubits, although it shows two outliers. The MLP model performs relatively well, with a moderate interquartile range and no outliers. However, the median of the MLP model is located above the box in the boxplot, indicating a slightly higher performance compared to the center of the interquartile range. Lastly, the VQC agent with 10 qubits and the data-reuploading and scaling technique displays a wider dispersion of data points, indicating a larger range of variability in its performance.

Finally, to conclude the analysis of results and regarding learning, in the Fig. 15 we can observe a good evolution of learning with very similar curves for all the agents. Notably, there is a decrease in variance in the later episodes, indicating increased stability and convergence in learning. This suggests that the chosen methodologies and architectures successfully capture the environment's patterns and dynamics. Overall, the learning curves exhibit reliable and consistent learning performance, highlighting the agents' capability to adapt to the task.

C. USE CASE 3: WAREHOUSE

1) PROBLEM STATEMENT

The focus of this use case is the *Eplus-warehouse-mixed-discrete-v1* environment, which is one of the environments available in the *Sinergym* simulator [32]. Specifically, this study concentrates on the *ASHRAE9012016-Warehouse-Denver* scenario.

The building in this study is located in Denver, USA, and has a total area of 49,495 square feet (330 ft x 150 ft). It consists of three zones: bulk storage, fine storage and office. The office area, which is the only section with windows, is enclosed on two sides and at the top by the fine storage zone. The exterior walls are constructed with a metal surface, wall insulation and gypsum board. The roof is also made of metal surface with roof insulation. Additionally, there are skylights present over the bulk and fine storage areas (see Fig. 16).

The state space is represented in Table: 8, which provides an overview of the variables included in the state space for this use case. The action space is the same as the previous use cases IV-A and the reward function is defined in Equation (15).

TABLE 7. Results obtained by MLP and VQC agents in the Use case 2. Column 1: MLP agent (classic) and the different VQC agents proposed (quantum); Column 2: Average total reward after training; Column 3: Best total reward after training; Column 4: Worst total reward after training; Column 5: Computational time in seconds; Column 6: Total reward (test with deterministic policy).

Model	Avg. Total Reward	Best Total Reward	Worst Total Reward	Time (s)	Test Reward
VQC-10	−4.43	−3.94	−5.15	1989.94	−4.43
VQC-10 (trainable input weight and Data-Reuploading)	−4.05	−3.48	−4.92	2435.74	−4.05
VQC-5 (Amplitude Encoding)	−4.02	−3.48	−4.49	4965.20	−4.02
MLP (Classic)	−4.46	−4.36	−4.73	1267.49	−4.46

TABLE 8. Observation Variables for Use case 3. Comprising 19 variables, with an additional 4 empty variables included to accommodate specific usage scenarios that require them.

Variable Name	Units
Site Outdoor Air Drybulb Temperature (Environment)	°C
Site Outdoor Air Relative Humidity (Environment)	%
Site Wind Speed (Environment)	m/s
Site Wind Direction	degree (Environment)
Site Diffuse Solar Radiation Rate per Area (Environment)	W/m^2
Site Direct Solar Radiation Rate per Area (Environment)	W/m^2
Zone Thermostat Heating Setpoint Temperature (Zone1 Office)	°C
Zone Thermostat Cooling Setpoint Temperature (Zone1 Office)	°C
Zone Air Temperature (Zone1 Office)	°C
Zone Air Relative Humidity (Zone1 Office)	%
Zone People Occupant Count	count (Zone1 Office)
Zone Thermostat Heating Setpoint Temperature (Zone2 Fine Storage)	°C
Zone Thermostat Cooling Setpoint Temperature (Zone2 Fine Storage)	°C
Zone Air Temperature (Zone2 Fine Storage)	°C
Zone Air Relative Humidity (Zone2 Fine Storage)	%
Zone Thermostat Heating Setpoint Temperature (Zone3 Bulk Storage)	°C
Zone Air Temperature (Zone3 Bulk Storage)	°C
Zone Air Relative Humidity (Zone3 Bulk Storage)	%
Facility Total HVAC Electricity Demand Rate (Whole building)	W

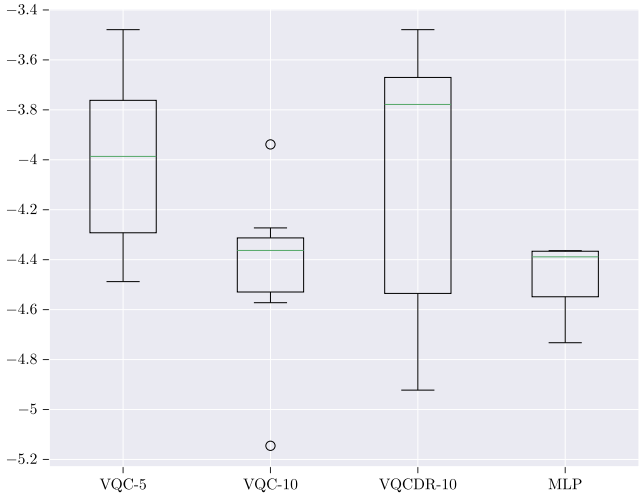


FIGURE 14. Boxplots illustrating the distribution of average total reward obtained from four different approaches in Use case 2, arranged from left to right as follows: a VQC with Amplitude encoding (5 qubits), VQC with linear layer pre-processing, VQC with data-reuploading and scaling, and MLP.

FIGURE 15. Learning curves obtained from four distinct methods in Use case 2. The approaches are: VQC with Amplitude Encoding using 5 qubits, VQC with a linear layer at pre-processing, a variant of the latter approach which incorporates data-reuploading and scaling methods, and the use of MLP.

2) EXPERIMENTAL SETTINGS

For this experiment, we utilize four models with an identical architecture, as detailed in Section III. The first model, referred to as the VQC-10 qubits model, incorporates a linear layer for dimensionality reduction. The Q-networks within this model are composed of a total of 330 parameters:$23 \times 10 = 230$ (pre-processing layer) + 5 (number of layers) × 2 (learning parameters used in Ry and Rz) × 10 (number of qubits). And the second solution, a variant of the previous model, incorporates data-reuploading and scaling techniques to the

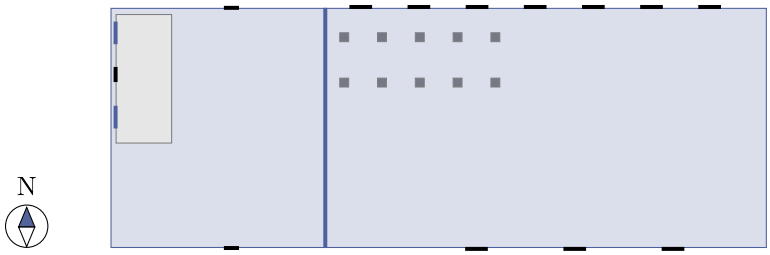


FIGURE 16. Warehouse plan. Illustrating bulk storage, fine storage, and an adjacent office area within the fine storage section.

solution. This variant consists of a total of 380 parameters: $23 \times 10 = 230$ (pre-processing layer) + 5 (number of layers) × 1 (learning parameter used in Rx for encode the data) × 10 (number of qubits) + 5 (number of layers) × 2 (learning parameters used in Ry, Rz for ansatz circuit) × 10. For the third solution, the architecture consists of encoding and ansatz circuits with 5 qubits, where the number of qubits is determined by $log_2 N$ with N being the number of features in the observation space, which is 23 in this environment. The total number of parameters in this model is 100: 10 (number of layers) × 2 (learning parameters used in Ry, Rz for ansatz circuit) × 5. Finally, the classical network used in this experiment comprises an input layer with 23 neurons, a hidden layer with 150 neurons and an output layer with 10 neurons. Consequently, the total number of parameters in this classical network is 5110: 23× 150 (weights) +150 (bias) + 10 × 150 (weights) + 10 (bias). The detailed configuration of these models can be found in Table: 9.

Once again, it is evident that the quantum proposal requires significantly fewer parameters compared to the classical agent. This indicates that the quantum agent exhibits a lower complexity compared to its classical counterpart. The reduced number of parameters in the quantum agent suggests a more efficient and compact representation of the model, which can have advantages in terms of training efficiency and computational resources required.

3) RESULTS

The performance results for the third use case are summarized in Table 10. This table presents the average, best and worst accumulated reward achieved by the three quantum agents and the MLP model. The computation time for each experiment is also included, measured in seconds. Furthermore, the average accuracy, defined as the average total accumulated reward over 10 different runs with fixed seeds for reproducibility, is reported.

Once again, the results highlight the superior performance of the quantum models compared to the classical model. The quantum agent with data-reuploading and trainable input weights achieved the highest average and test reward values, indicating its effectiveness in maximizing rewards. On the other hand, the quantum agent utilizing amplitude encoding demonstrated the best and worst total reward values, showcasing its potential achieving both exceptional and suboptimal outcomes.

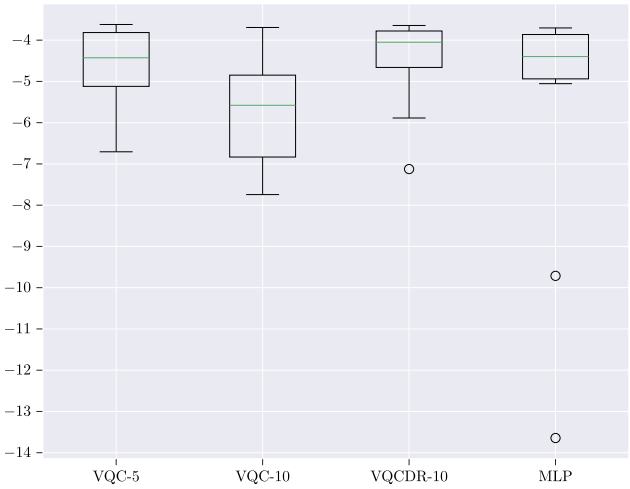


FIGURE 17. Boxplots illustrating the distribution of average total reward obtained from four different approaches in Use case 3, arranged from left to right as follows: a VQC with Amplitude encoding (5 qubits), VQC with linear layer pre-processing, VQC with data-reuploading and scaling methods, and MLP.

Furthermore, the robustness analysis presented in Fig. 17 demonstrates that the agent utilizing amplitude encoding exhibits the highest level of robustness among all the models. This is evident from the absence of outliers and the positioning of its median in the middle of the box, indicating consistent and stable performance. The MLP model also performs relatively well in terms of robustness, with a narrower interquartile range. However, it does have two outliers, suggesting some variability in its performance. Finally, the VQC model with 10 qubits and VQC with data-reuploading and trainable input weights show a similar level of robustness. The former, although without outliers and with the median very close to the center, exhibits a larger interquartile range, while the latter has a narrower interquartile range but a higher median and an outlier.

To conclude the analysis, Fig. 18 demonstrates that all the models exhibit good learning curves, with a decrease in variance in the later epochs. It is worth noting that the MLP model shows a relatively poorer learning curve compared to the other models.

TABLE 9. Structure of DQN classical/quantum networks used in use case 3.

	MLP	VQC-10	VQC-10 (Trainable input weight and Data-Reuploading)	VQC-5 (Amplitude Encoding)
No. of qubits	-	10	10	5
Inputs	23	23. Scaled to $[-\pi/2, \pi/2]$	23. Scaled with λ	23. Normalized to 1
Pre-processing	-	Linear (10 units)	Linear (10 units)	-
Quantum embedding	-	-	Tensor product encoding	Amplitude encoding
Intermediate Layers	Linear (23 units, 150 units) ReLU	Variational Layer (10 qubits) Tahn	Variational Layer (10 qubits) Tahn	Variational Layer (5 qubits) ReLU
Output layer	Linear (150 units, 10 units)	$\sigma_z(q_i)$ for each qubit q_i	$\sigma_z(q_i)$ for each qubit q_i	state-vector
Post-processing	Softmax	Softmax	Softmax	Linear($2^{Nqubits}$, 10)+Softmax
Training algorithm	RMSProp (lr=0.001)	RMSProp (lr=0.009)	RMSProp ($lr[1 \times 10^{-2}, 1 \times 10^{-4}]$)	RMSProp (lr=0.01)

TABLE 10. Results obtained by MLP and VQC agents in the Use case 3. Column 1: MLP agent (classic) and the different VQC agents proposed (quantum); Column 2: Average total reward after training; Column 3: Best total reward after training; Column 4: Worst total reward after training; Column 5: Computational time in seconds; Column 6: Total reward (test with deterministic policy).

Model	Avg. Total Reward	Best Total Reward	Worst Total Reward	Time (s)	Test Reward
VQC-10	−5.75	−3.69	−7.74	4745.92	−5.75
VQC-10 (trainable input weight and Data-Reuploading)	−4.53	−3.64	−7.12	4497.74	−4.53
VQC-5 (Amplitude Encoding)	−4.60	−3.62	−6.71	4929.64	−4.60
MLP (Classic)	−5.70	−3.70	−13.64	3327.15	−5.70

FIGURE 18. Learning curves obtained from four distinct methods in Use case 3. The approaches are: VQC with Amplitude Encoding using 5 qubits, VQC with a linear layer at pre-processing, a variant of the latter approach which incorporates data-reuploading and scaling methods, and the use of MLP.

V. DISCUSSION

We introduced three distinct architectures for quantum agents in our study, each designed to address the challenge of dimensionality reduction. These architectures encompassed approaches involving a classical layer for dimensionality reduction, as well as methods employing efficient encoding techniques to effectively manage the high-dimensional feature space. These three proposed architectures were applied to three distinct use cases focusing on efficient energy efficiency management. In order to evaluate their performance, we compared them against state-of-the-art classical RL methods.

An overarching insight derived from the analysis of the results across the three use cases is the feasibility and effectiveness of employing quantum reinforcement learning approaches in decision-making processes. These quantum approaches demonstrate the ability to achieve superior performance compared to their classical counterparts, while also exhibiting reduced complexity (requiring fewer parameters during the training). Additionally, the quantum reinforcement learning approaches investigated in this study offer various strategies for addressing the challenge of dimensionality reduction, achieving excellent results, particularly through the utilization of compact circuits with the minimum number of required qubits. This highlights the versatility and efficiency of quantum algorithms in effectively handling high-dimensional observation spaces, enabling the agents to make informed decisions while minimizing computational complexity. The combination of superior performance and reduced dimensionality further reinforces the potential of quantum reinforcement learning as a powerful tool for tackling complex decision-making problems. During the training process, all the agents demonstrated stability with lower standard deviation in terms of the total reward. However, consistently across all use cases, the VQC agent utilizing amplitude encoding emerged as the top-performing agent. This agent not only achieved the highest performance in terms of total reward but also showcased superior robustness compared to other agents. Its ability to consistently outperform other models in various use cases highlights the effectiveness and versatility of quantum reinforcement learning with amplitude encoding.

On the other hand, it is important to acknowledge that the computation time of quantum reinforcement learning approaches is typically longer compared to their classical counterparts. This is primarily due to the utilization of quantum simulators instead of real quantum computers. Although quantum models offer the advantage of reduced model complexity, the simulation of quantum systems introduces additional computational overhead. Furthermore, it is worth noting that the proposed quantum models were not evaluated on contemporary noisy intermediate-scale quantum (NISQ) hardware. Accessing and conducting large-scale

experiments on current quantum devices presents challenges, and the inherent noise in these systems can impact the performance and reliability of the models. These limitations underscore the need for ongoing advancements in quantum hardware and improved accessibility to facilitate real-world testing and validation of quantum reinforcement learning algorithms.

VI. CONCLUSION AND FUTURE WORK

In this work, we conducted an investigation into the potential of quantum deep reinforcement learning for energy efficiency scenarios. Our study revealed that the hybrid quantum-classical algorithmic setup offers superior performance compared to the classical deep reinforcement learning baseline. The quantum models showed improved performance, with the compact model utilizing amplitude encoding standing out. However, it is important to note that the training duration of the quantum models was longer compared to the classical baseline due to the utilization of quantum simulators, as explained in the previous section.

The selection of an appropriate encoding strategy plays a crucial role in the convergence of the quantum reinforcement learning algorithm. In our study, we employed data-reuploading and scaling techniques [50], [51] and amplitude encoding. These strategies proved to be effective, yielding promising results across the three quantum architectures. Notably, the agent utilizing amplitude encoding consistently outperformed the other architectures, exhibiting superior performance across all the use cases studied in this work.

Addressing the scaling behavior of quantum reinforcement learning methods in real-world environments with a large number of variables has been a significant challenge. Most of the existing research and benchmarking in this field have focused on small-scale OpenAI environments. Our work complements the existing body of research by not only addressing the challenges encountered in small-scale environments but also paving the way for the analysis of intricate and large-scale and real-world scenarios, we make three key contributions.

Firstly, we demonstrate the feasibility and effectiveness of efficient encoding techniques in quantum deep reinforcement learning. This involved employing circuits with fewer qubits, enabling dimensionality reduction without sacrificing model effectiveness or essential information retention. This successful endeavor addressed challenges related to scalability, thus facilitating the application of quantum reinforcement learning in high-dimensional scenarios.

Secondly, we highlight the potential application of quantum reinforcement learning in energy efficiency and management scenarios, which has been previously demonstrated in our earlier work [10]. By leveraging the power of quantum algorithms and combining them with reinforcement learning techniques, we showcase the capabilities of quantum agents in addressing complex decision-making problems in the field of energy efficiency.

And finally, we have addressed critical technical challenges, such as barren plateaus [53], potencially leading to prolonged convergence times and, in some cases, impeded optimization altogether, We have successfully mitigated the barren plateau issue through the application of higher or adaptive learning rate techniques, allowing us to overcome the stagnation commonly associated with flat regions of the optimization landscape. Additionally, the strategic selection of quantum circuits with optimal qubit counts played a pivotal role in alleviating barren plateau-related convergence issues. Our investigation revealed that by carefully designing circuits with the appropriate number of qubits, we were able to circumvent the most severe instances of plateau-induced optimization difficulties. These solutions have not only improved convergence but have also laid the foundation for more stable and efficient quantum neural network training. However, despite these solutions, further research is necessary to comprehensively address the barren plateau challenge in quantum neural networks. Exploring alternative parameter initialization methods and novel optimization strategies remains an open area of investigation.

It is important to note that the inherent instability of the DQN algorithm, which in these use cases has been solved using adaptive learning rates and/or a higher number of layers, highlights the need for alternative neural network models that offer greater stability. Brain-inspired neural network models, for example, hold promise as they bridge the fields of neuroscience, artificial intelligence, and quantum computing. Research efforts should be directed towards exploring these models and investigating their potential in quantum deep reinforcement learning scenarios.

Another important consideration is that all our experiments were conducted using quantum simulators, which allowed us to isolate the noise typically present in current quantum hardware. However, a noteworthy challenge arises from the substantial time required for simulations. The complexity of emulating quantum operations on classical hardware contributes to these extended execution times. To address this challenge, we have adopted an approach that strategically employs compact quantum circuits with fewer qubits, layers, and entanglements or hybrid models with classical layers to reduce the number of qubits. This optimization allows us to mitigate the computational demands of simulations, effectively reducing execution times. Future research should focus on optimizing libraries and simulators, as well as exploring techniques for efficiently combining quantum simulators and quantum hardware in training setups. This would enable more practical and efficient experimentation, ultimately paving the way for real-world applications.

Lastly, it is our strong belief that the progress made in the areas of deep reinforcement learning (DRL), quantum neural networks (QNN), and future developments in quantum computing hardware will play a pivotal role in advancing the field of quantum machine learning and quantum reinforcement learning. The synergy between DRL and QNN holds great promise for addressing real-world challenges that are beyond

the capabilities of classical approaches. By harnessing the power of quantum computing and combining it with the sophisticated learning capabilities of DRL, we can expect significant advancements in various domains, including optimization, pattern recognition, and decision-making. Quantum machine learning and quantum reinforcement learning have the potential to unlock new insights, uncover hidden patterns, and provide innovative solutions in areas such as drug discovery, financial modeling, logistics optimization, and more.

In conclusion, future research in quantum machine learning and quantum reinforcement learning should focus on addressing the challenges posed by barren plateaus, exploring alternative neural network models, optimizing simulation techniques, and leveraging advancements in quantum hardware. By pursuing these avenues, we can anticipate significant advancements in the fields of quantum machine learning and quantum reinforcement learning, with broad implications for various real-world applications.

REFERENCES

[1] P. Wittek, *Quantum Machine Learning: What Quantum Computing Means to Data Mining*. Amsterdam, The Netherlands: Elsevier, 2014.

[2] C. Ciliberto, M. Herbster, A. Ialongo, M. Pontil, A. Rocchetto, S. Severini, and L. Wossnig, "Quantum machine learning: A classical perspective," *Proc. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 474, p. 0551, Jul. 2017.

[3] D. Pastorello, *Quantum Neural Networks*. Singapore: Springer Nature, 2023, pp. 101–123, doi: 10.1007/978-981-19-6897-6_9.

[4] A. Macaluso, L. Clissa, S. Lodi, and C. Sartori, "A variational algorithm for quantum neural networks," in *Computational Science–ICCS 2020*. Berlin, Germany: Springer, 2020, pp. 591–604.

[5] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[6] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe, "Circuit-centric quantum classifiers," *Phys. Rev. A, Gen. Phys.*, vol. 101, no. 3, Mar. 2020, Art. no. 032308.

[7] N. Wiebe, A. Kapoor, and K. M. Svore, "Quantum nearest-neighbor algorithms for machine learning," *Quantum Inf. Comput.*, vol. 15, pp. 318–358, Jan. 2015.

[8] D. Anguita, S. Ridella, F. Rivieccio, and R. Zunino, "Quantum optimization for training support vector machines," *Neural Netw.*, vol. 16, nos. 5–6, pp. 763–770, Jun. 2003.

[9] E. Aïmeur, G. Brassard, and S. Gambs, "Quantum speed-up for unsupervised learning," *Mach. Learn.*, vol. 90, no. 2, pp. 261–287, Feb. 2013.

[10] E. Andrés, M. P. Cuéllar, and G. Navarro, "On the use of quantum reinforcement learning in energy-efficiency scenarios," *Energies*, vol. 15, no. 16, p. 6034, Aug. 2022.

[11] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[12] (2017). *Advances in Neural Information Processing Systems: 31st Annual Conference on Neural Information Process*. [Online]. Available: https://books.google.es/books?id=KrTxuQEACAAJ

[13] M. Botvinick, J. X. Wang, W. Dabney, K. J. Miller, and Z. Kurth-Nelson, "Deep reinforcement learning and its neuroscientific implications," *Neuron*, vol. 107, no. 4, pp. 603–616, Aug. 2020.

[14] D. Niraula, J. Jamaluddin, M. M. Matuszak, R. K. T. Haken, and I. E. Naqa, "Quantum deep reinforcement learning for clinical decision support in oncology: Application to adaptive radiotherapy," *Sci. Rep.*, vol. 11, no. 1, p. 23545, Dec. 2021, doi: 10.1038/s41598-021-02910-y.

[15] E. A. Cherrat, S. Raj, I. Kerenidis, A. Shekhar, B. Wood, J. Dee, S. Chakrabarti, R. Chen, D. Herman, S. Hu, P. Minssen, R. Shaydulin, Y. Sun, R. Yalovetzky, and M. Pistoia, "Quantum deep hedging," 2023, *arXiv:2303.16585*.

[16] C. Zhao and X.-S. Gao, "QDNN: Deep neural networks with quantum layers," *Quantum Mach. Intell.*, vol. 3, no. 1, p. 15, Jun. 2021.

[17] S. Y. Chen, C. 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] O. Lockwood and M. Si, "Reinforcement learning with quantum variational circuit," in *Proc. AAAI Conf. Artif. Intell. Interact. Digit. Entertainment*, Oct. 2020, vol. 16, no. 1, pp. 245–251. [Online]. Available: https://ojs.aaai.org/index.php/AIIDE/article/view/7437

[19] A. Sannia, A. Giordano, N. L. Gullo, C. Mastroianni, and F. Plastina, "A hybrid classical-quantum approach to speed-up Q-learning," *Sci. Rep.*, vol. 13, no. 1, p. 3913, Mar. 2023, doi: 10.1038/s41598-023-30990-5.

[20] W. J. Yun, Y. Kwak, J. P. Kim, H. Cho, S. Jung, J. Park, and J. Kim, "Quantum multi-agent reinforcement learning via variational quantum circuit design," in *Proc. IEEE 42nd Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2022, pp. 1332–1335.

[21] Z. Liao, J. Peng, Y. Chen, J. Zhang, and J. Wang, "A fast Q-learning based data storage optimization for low latency in data center networks," *IEEE Access*, vol. 8, pp. 90630–90639, 2020.

[22] X. Dutreilh, S. Kirgizov, O. Melekhova, J. Malenfant, N. Rivierre, and I. Truck, "Using reinforcement learning for autonomic resource allocation in clouds: Towards a fully automated workflow," in *Proc. 7th Int. Conf. Autonomic Auton. Syst.*, May 2011, pp. 1–8.

[23] S. Swarup, E. M. Shakshuki, and A. Yasar, "Task scheduling in cloud using deep reinforcement learning," *Proc. Comput. Sci.*, vol. 184, pp. 42–51, 2021.

[24] J. Wan, J. Zhou, and X. Gui, "Intelligent rack-level cooling management in data centers with active ventilation tiles: A deep reinforcement learning approach," *IEEE Intell. Syst.*, vol. 36, no. 6, pp. 42–52, Nov. 2021.

[25] Q. Fu, E. Sun, K. Meng, M. Li, and Y. Zhang, "Deep Q-learning for routing schemes in SDN-based data center networks," *IEEE Access*, vol. 8, pp. 103491–103499, 2020.

[26] A. H. Khalaj and S. K. Halgamuge, "A review on efficient thermal management of air- and liquid-cooled data centers: From chip to the cooling system," *Appl. Energy*, vol. 205, pp. 1165–1188, Nov. 2017.

[27] N. S. Raman, A. M. Devraj, P. Barooah, and S. P. Meyn, "Reinforcement learning for control of building HVAC systems," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2020, pp. 2326–2332.

[28] Y. Wang, K. Velswamy, and B. Huang, "A long-short term memory recurrent neural network based reinforcement learning controller for office heating ventilation and air conditioning systems," *Processes*, vol. 5, no. 3, p. 46, Aug. 2017. [Online]. Available: https://www.mdpi.com/2227-9717/5/3/46

[29] Q. Fu, Z. Han, J. Chen, Y. Lu, H. Wu, and Y. Wang, "Applications of reinforcement learning for building energy efficiency control: A review," *J. Building Eng.*, vol. 50, Jun. 2022, Art. no. 104165. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352710222001784

[30] D. Crawley, L. Lawrie, C. Pedersen, and F. Winkelmann, "EnergyPlus: Energy simulation program," *ASHRAE J.*, vol. 42, no. 4, pp. 49–56, 2000.

[31] S. E. Mattsson and H. Elmqvist, "Modelica—An international effort to design the next generation modeling language," *IFAC Proc. Volumes*, vol. 30, no. 4, pp. 151–155, Apr. 1997.

[32] J. Jiménez-Raboso, A. Campoy-Nieves, A. Manjavacas-Lucas, J. Gómez-Romero, and M. Molina-Solana, "Sinergym: A building simulation and control framework for training reinforcement learning agents," in *Proc. 8th ACM Int. Conf. Syst. Energy-Efficient Buildings, Cities, Transp.* New York, NY, USA: Association for Computing Machinery, 2021, pp. 319–323, doi: 10.1145/3486611.3488729.

[33] Z. Zhang and K. P. Lam, "Practical implementation and evaluation of deep reinforcement learning control for a radiant heating system," in *Proc. 5th Conf. Syst. Built Environments*. New York, NY, USA: Association for Computing Machinery, Nov. 2018, pp. 148–157, doi: 10.1145/3276774.3276775.

[34] P. Scharnhorst, B. Schubnel, C. F. Bandera, J. Salom, P. Taddeo, M. Boegli, T. Gorecki, Y. Stauffer, A. Peppas, and C. Politi, "Energym: A building model library for controller benchmarking," *Appl. Sci.*, vol. 11, no. 8, p. 3518, Apr. 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/8/3518

[35] R. Lian, J. Peng, Y. Wu, H. Tan, and H. Zhang, "Rule-interposing deep reinforcement learning based energy management strategy for power-split hybrid electric vehicle," *Energy*, vol. 197, Apr. 2020, Art. no. 117297.

[36] J. Peng, Y. Fan, G. Yin, and R. Jiang, "Collaborative optimization of energy management strategy and adaptive cruise control based on deep reinforcement learning," *IEEE Trans. Transport. Electrific.*, vol. 9, no. 1, pp. 34–44, Mar. 2023. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85130815207

[37] S. Wang, S. Bi, and Y. A. Zhang, "Reinforcement learning for real-time pricing and scheduling control in EV charging stations," *IEEE Trans. Ind. Informat.*, vol. 17, no. 2, pp. 849–859, Feb. 2021.

[38] M. Periyasamy, M. Hölle, M. Wiedmann, D. D. Scherer, A. Plinge, and C. Mutschler, "Batch quantum reinforcement learning," 2023, *arXiv:2305.00905*.

[39] S. Y.-C. Chen, C.-M. Huang, C.-W. Hsing, H.-S. Goan, and Y.-J. Kao, "Variational quantum reinforcement learning via evolutionary optimization," *Mach. Learning: Sci. Technol.*, vol. 3, no. 1, Feb. 2022, Art. no. 015025, doi: 10.1088/2632-2153/ac4559.

[40] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, Mar. 1996, doi: 10.1109/2.485891.

[41] R. E. Uhrig, "Introduction to artificial neural networks," in *Proc. 21st Annu. Conf. IEEE Ind. Electron. (IECON)*, vol. 1, Sep. 1995, pp. 33–37.

[42] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, "Parameterized quantum circuits as machine learning models," *Quantum Sci. Technol.*, vol. 4, no. 4, Nov. 2019, Art. no. 043001, doi: 10.1088/2058-9565/ab4eb5.

[43] M. Weigold, J. Barzen, F. Leymann, and M. Salm, "Expanding data encoding patterns for quantum algorithms," in *Proc. IEEE 18th Int. Conf. Softw. Archit. Companion (ICSA-C)*, 2021, pp. 95–101.

[44] 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, May 2022.

[45] Y. Kwak, W. J. Yun, S. Jung, J.-K. Kim, and J. Kim, "Introduction to quantum reinforcement learning: Theory and Pennylane-based implementation," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, 2021, pp. 416–420.

[46] H. Dong, Z. Ding, S. Zhang, H. Yuan, H. Zhang, J. Zhang, Y. Huang, T. Yu, H. Zhang, and R. Huang, *Deep Reinforcement Learning: Fundamentals, Research, and Applications*, H. Dong, Z. Ding, and S. Zhang, Eds. Berlin, Germany: Springer Nature, 2020. [Online]. Available: http://www.deepreinforcementlearningbook.org

[47] R. S. Sutton and A. G. Barto, *The Reinforcement Learning Problem*. Cambridge, MA, USA: MIT Press, 1998, pp. 51–85. [Online]. Available: https://ieeexplore.ieee.org/document/6282968

[48] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[49] K. Shao, D. Zhao, Y. Zhu, and Q. Zhang, "Visual navigation with actor-critic deep reinforcement learning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2018, pp. 1–6.

[50] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, "Data re-uploading for a universal quantum classifier," *Quantum*, vol. 4, p. 226, Feb. 2020, doi: 10.22331/q-2020-02-06-226.

[51] M. Schuld, R. Sweke, and J. J. Meyer, "Effect of data encoding on the expressive power of variational quantum-machine-learning models," *Phys. Rev. A, Gen. Phys.*, vol. 103, no. 3, Mar. 2021, Art. no. 032430.

[52] S. Jerbi, C. Gyurik, S. C. Marshall, H. J. Briegel, and V. Dunjko, "Parametrized quantum policies for reinforcement learning," 2021, *arXiv:2103.05577*.

[53] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, "Barren plateaus in quantum neural network training landscapes," *Nature Commun.*, vol. 9, no. 1, p. 4812, Nov. 2018, doi: 10.1038/s41467-018-07090-4.

EVA ANDRÉS received the degree in computer science from the University of Granada, specializing in artificial intelligence, and the business master's degree from Camilo Jose Cela University, Madrid, in 2015. She is currently pursuing the Ph.D. degree in parametrized quantum circuits for reinforcement learning. Her professional career has developed in an international environment, dealing with critical services and complex projects that require a customer-centric approach. Since 2018, she has been working on innovation projects focusing on time series, NLP, and computer vision, among others. Her research interests include artificial neural networks, reinforcement learning, evolutionary algorithms, and quantum machine learning.

M. P. CUÉLLAR received the bachelor's and master's degrees in computer science, in 2003 and 2006, respectively. He has been working in time series forecasting and neural networks, since 2004. Since 2012, he has been a tenured Professor with the University of Granada, Spain. He is currently with the Department of Computer Science and Artificial Intelligence. His main research interests include neural networks, evolutionary computation, metaheuristics, ambient intelligence, reinforcement learning, or sensor data gathering and modeling for AI applications, among others. His research focuses on quantum machine learning and quantum neural networks and their application to different machine learning fields.

G. NAVARRO received the Ph.D. degree in mathematics from the University of Granada (UGR), Granada, Spain, in 2006. He is currently an Associate Professor with the Department of Computer Sciences and Artificial Intelligence, UGR. He is also a member of the Research Centre for Information and Communications Technologies (CITIC), UGR. His current research interests include coding theory, computational algebra, and fuzzy and rough sets.