Hindawi
Quantum Engineering
Volume 2023, Article ID 2451990, 16 pages
https://doi.org/10.1155/2023/2451990

Research Article

Investigating the Effects of Hyperparameters in Quantum-Enhanced Deep Reinforcement Learning

Getahun Fikadu Tilaye[1] **and Amit Pandey**[2]

[1]*College of Informatics, Software Engineering Department, Bule Hora University, Bule Hora, Ethiopia*
[2]*College of Informatics, Computer Science Department, Bule Hora University, Bule Hora, Ethiopia*

Correspondence should be addressed to Getahun Fikadu Tilaye; getahunf2020@gmail.com

Received 14 November 2022; Revised 20 January 2023; Accepted 20 February 2023; Published 14 March 2023

Academic Editor: Shi Hai Dong

Quantum machine learning uses quantum mechanical concepts of superposition of states to make the decision. In this work, we used these quantum advantages to enhance deep reinforcement learning (DRL). Our primary and foremost goal is to investigate and elucidate a way of representing and solving the frozen lake problems by using PennyLane which contains Xanadu's back-end quantum processing unit. This paper specifically discusses how to enhance classical deep reinforcement learning algorithms with quantum computing technology, making quantum agents get a maximum reward after a fixed number of epochs and realizing the effect of a number of variational quantum layers on the trainability of enhanced framework. We have analyzed that, as the number of layers increases, the ability of the quantum agent to converge to the optimal state also increases. For this work, we have trained the framework agent with 2, 3, and 5 variational quantum layers. An agent with 2 layers converges to a total reward of 0.95 after the training episode of 526. The other agent with layers converges to a total reward of 0.95 after the training episode of 397 and the agent which uses 5 quantum variational layers converges to a total reward of 0.95 after the training episode of 72. From this, we can understand that the agent with a more variational layer exploits more and converges to the optimal state before the other agent. We also analyzed our work in terms of different learning rate hyperparameters. We recorded every single learning epoch to demonstrate the outcomes of enhanced DRL algorithms with selected 0.1, 0.2, 0.3, and 0.4 learning rates or alpha values. From this result, we can conclude that the greater the learning rate values in quantum deep reinforcement learning, the fewer timesteps it takes to move from the start point to the goal state.

1. Introduction

Quantum machine learning (QML) is a subconcept of quantum computational data and information processing research in which the main target is developing the QML algorithms that can learn from data to improve the existing classical learning methods in machine learning [1]. Here, quantum computers can process data in the form of qubits rather than bits and this computer can have also its own architecture. For this architecture, it is impossible to apply classical machine learning methods directly to learn and process the data. A quantum machine learning algorithm can be implemented on a quantum computer; this computer can exploit the laws of the quantum mechanical model (quantum theory) to process the data and to predict the outcome. Aimed to investigate the applicability of quantum computing advantage to solve such problems, the researcher proposed this study and developed quantum circuits for DRL to analyze the performance of quantum agents with different learning rates and different quantum layers. This study used the advantage of PennyLane provided by Xanadu to create circuits and the open gym benchmark-like frozen lake problem was solved. The expected optimization of the best policy action selection of a quantized agent to get maximum reward is analyzed and then the learning performance is stated. This can be achieved by calculating the quantum expectation value or expected energy value.

The main basis of quantum machine learning is to use the essential benefits of quantum computing technology such as quantum entanglement, quantum superposition, and quantum parallelism to increase the performance of classical machine learning algorithms [2–5]. Although they utilized the quantum mechanics' concept of quantum entanglement and supper position, most of the quantum machine learning algorithms (QMLA) are borrowed from the basic concept and architecture of classical machine learning algorithms [6]. In other cases, some researchers used the hybrid classical-quantum architecture and proved that the use of classical-quantum hybrid systems can overcome the limitation of noise intermediate scale quantum (NISQ) computers [7, 8]. For example, Dunjko et al. [9] proposed the quantum approximate algorithm in which classical solution parameters to a problem can be used as an input and as the starting point for larger problems such as variational quantum eigen solvers. Recently, the experiment held by Parades et al. [10] proved that quantum machine learning has shown interesting results for machine learning applications where classical machine learning techniques are limited to perform because of inadequate training data and high dimensionality of the features. They proposed a PennyLane simulator-based hybrid classical-quantum model for transfer learning to enhance the classification of face images with a COVID-19 mask and resulted in an accuracy of 99.05% in classifying the exact protective masks. Lloyd et al. [11] also proposed quantum-inspiredk-means and nearest centroid algorithms through quantum distance approximation as we do have quantum access to the classical data. In quantum enhanced clustering algorithm, the execution time for a certain state of N vectors in feature space with dimension D of each iteration (by a distance approximation technique with error ε) is given by $O(kN \log D/\varepsilon)$. Generally, several works depict the incredible performance of the quantum-enhanced machine learning model for the problem with large feature spaces, which can be difficult for the classical machine learning algorithm to perform.

2. Related Works

By its nature, whether it is enhanced or not, deep reinforcement learning (DRL) is a machine learning in which the interaction takes place between an agent and the environment. In various real-worlddecision-making situations, there is no data on the best course of action. For such a scenario, the model must interfere with its surroundings in order to gather information and learn how to complete a task via its own experience. This learning method is called reinforcement learning (RL). For example, a video game character might learn a fruitful strategy by repetitively playing the game, examining the outcomes, and improving the future ways. In current studies of quantum enhanced RL [12, 13], variational quantum circuits (VQC) replace the policy of training the deep neural network (DNN) of existing deep reinforcement learning (DRL). At each experience in each episode, a quantum agent with particular state information decides its action from the policy of variational quantum circuits, and the classical optimizer such as RMsprop, Adam optimizer, and others are used to update the parameters. The action selection is based on the expected value of quantum measurement.

Though quantum deep reinforcement learning (QDRL) with designated parameterized quantum circuits (PQC) is a quite new field of study, some researchers such as Lockwood and Si [12] studied different encoding methods such as directional encoding and scaled-up encoding techniques and also pooled the parameterized circuit with quantum pooling operations to solve some reinforcement learning problems. This encoding technique is cost-efficient but requires a large number of quantum gates which are beyond the capabilities of actual quantum simulators.

The method used by Hu and Hu [14] uses common gates, but they investigated with less number of qubits. It needs some investigation to know what if the number of qubits increased. Here, the intention is to advance the intelligence of quantum agents, which can react with the environment and learn from it to reach some stated goal. In this sense, numerous works have provided suggestions in the last few years [9, 15–18]. Few of these deal with enhancing reward by quantizing an agent which interacts with a classical environment through Grover search. The others prove quantum speed up when both agents and environments had been quantized with quantum phenomena.

Table 1 shows the bird's eye view of quantum, quantum-classical, and classical deep reinforcement learning. The learning algorithm can be enhanced either by quantizing an agent or an environment, or both. Quantum computers may be available only in large companies such as IBM, D-Wave, and Google for research purposes [9, 19]. For the reason of unavailability of such resources, we find a way for developing the hybrid and running the learning algorithms on current classical computers such as the method used by [15, 20, 21].

For the scenario of "*quantum agent classical environment*," Dunjko et al. [9] suggested the Grovers iteration-based searching algorithm for unstructured search and investigates a potential quantum computing benefit when a quantized agent interacts with a classical environment and the study improves the quantum speedup over the classical computations. The other scenario of quantum-enhanced deep reinforcement learning comes with a "*quantum agent quantum environment*." The proposed study by Lamata [22] and Albarran et al. [23] works on this scenario by considering the quantum agent and quantum environment interactions to investigate the possibility of implementations by increasing the extent of system complexity. Another prospect would be to investigate classical agent and quantum environment interactions which are made by applying the "*classical agent-quantum environment*" scenario. The study proposed by Daoyi Dong et al. [24] investigates the way of mapping conventional reinforcement learning into quantum reinforcement learning, and the study focuses on linking the previous states with future quantum states. According to the study with different learning parameters, there is a quadratic quantum speedup compared to the classical temporal difference (TD) by applying superposition and quantum entanglement on the quantum algorithms.

Table 1: Quantum enhanced hybrid, quantum, and classical DRL concerning resource availability, information representation, applicable learning algorithm, and major application areas.

Ways of enhancing DRL	Agent	Environment	Information representation	Learning algorithm	Computational resource	Application area
Classical DRL	Classical	Classical	Bits	DNN, GBDT	Resource available	Broad application area
Hybrid C-Q DRL	Classical	Quantum	Bits and qubits	DNN + VQC	Limited resource	Area enhanced with partial QC advantage
	Quantum	Classical	Bits and qubits	DNN + VQC	Limited resources	Area enhanced with partial QC advantage
Pure quantum DRL	Quantum	Quantum	Qubits	Quantum algorithm	Limited resource	All areas which need QC advantage

Some other researchers [9, 15–18] study the applicability of quantum computing on classical reinforcement learning. According to the study, variational quantum circuits can replace the policy training of deep neural networks in existing classical DRL. The interaction between the environment and an agent takes place based on the policy of quantum circuits. For each and every episode, an agent decides its action from the policy of quantum circuits, and the classical optimizer such as Adam optimizer is used to update the parameter [19]. This paper uses the same scenarios with different circuit designs and different environments.

3. Methods

3.1. Deep Reinforcement Learning. Reinforcement learning (RL) is different from supervised and unsupervised machine learning, which is naturally about training input with its corresponding output and searching the structure hidden in a group of unlabeled examples. Even though one might think of reinforcement learning as a type of unsupervised learning for the reason that it does not depend on the instances of correct behavior, RL is a learning method to maximize the reward signal instead of finding the hidden structure of the dataset. Finding structure in an agent's experience can be advantageous in reinforcement learning, but the agent does not know the learning tricks of future activities.

All RL agents have obvious goals in that they pick actions to make an impact on their environments and based on the action the agent can be rewarded. Furthermore, it is typically supposed that from the starting point, the agent must try in any doubt about the environment it faces. When RL comprises of planning, it has to identify the relationship between planning and agents' ability of real-time action selection, and also the searching method of the environments model are developed and enhanced. By recall, the aim of RL is to solve the problem of serial decision-making in discrete and continuous state space and to take actions with extremely probable rewards. The agent must follow a policy in making a series of decisions by taking actions in a different environment. Estimating the goodness of policy in taking the decision is evaluated at the end of accumulated costs or rewards through epochs. Based on the evaluation of the prior policy, the next policy must be improved by the agent to make an increased probability of deciding with greater estimated rewards. In the iteration of every step, the agent uses trial and error to improve the policy till the policy reaches the best optimum, which is the Markov decision process (MDP). MDPs are a classical reinforcement learning in serial decision-making, where actions affect not just the immediate rewards but also the successive states and the future rewards.

Here, the interaction is between agents and the environment. The agent acts on an environment by selecting actions from the set of actions A. Selecting the best action is based on the policy π, which is a function used to map the environment state s_t to the action a_t (where t is the time steps an agent takes to receive state s_t). Here, the policy π can be stochastic, which depicts that, for every state, the result of an action can be a probability distribution. The agent obtains the next state s_{t+1}, and a scalar of reward r_t, after performing the action a_t, and the process is iteratively continued up to the end of episodes. The total discounted reward R for the time step t is calculated as

$$R_t = \sum_{t_{\text{next}}=t}^{T} d^{t_{\text{next}}-t}\left(r\left(t_{\text{next}}\right)\right). \tag{1}$$

Here, d represents the discounted factor and its value lies between (0 and 1] in principle and it is decided by the programmer to control the future returns. No matter what the discount rate is, the agent takes the future rewards into account when the programmer decides to take a large discount factor d. Conversely, the agent rapidly disregards the future rewards when the assumed discount factor d is small.

(i) Action-value function

Based on policy π, there is an expected reward returned to the agent for picking the action a in the environment of state s. The value of taking an action a in the state s following the policy π is represented by $q_\pi(s,a)$ as the probable return of reward starting from the state s, taking an action a, and then following the policy π which is represented as

$$\begin{aligned} q_\pi(s,a) &= E_\pi\left[R_t \middle| s_t = s, a\right], \\ &= E_\pi\left[\sum_{t_{\text{next}}=t}^{T} d^{t_{\text{next}}-t}\left(r\left(t_{\text{next}}\right)\right) \middle| s_t = s, a\right]. \end{aligned} \tag{2}$$

(ii) State value function

The function value of state s which is under the policy π is represented by $v_\pi(s)$ which is the probable return when the agent is initially in the state s and is following the policy π and this is represented as

$$\begin{aligned} v_\pi(s) &= E_\pi\left[R_t \middle| s_t = s\right], \\ &= E_\pi\left[\sum_{t_{\text{next}}=t}^{T} d^{t_{\text{next}}-t}\left(r\left(t_{\text{next}}\right)\right) \middle| s_t = s\right]. \end{aligned} \tag{3}$$

(iii) Optimal action-value function

There is one action value that is greater or equal to all other action values, and also at the minimum probability, there is permanently one policy that is greater than or equal to all other policies. This action value and policy are said to be optimal action value (denoted by q_*) and optimal policy (denoted by π_*) respectively.

$$q_*(s,a) = \max_\pi q_\pi(s,a). \tag{4}$$

This function $q_*(s,a)$ provides the probable return for taking action a in state s and then following the optimal policy π_*.

3.1.1. Q-Learning: Off-Policy Temporal Difference Control. According to Amelia [25], it is a type of model-free reinforcement learning algorithm in which the agent does not require to know the future environment state. In the beginning, Q is allocated to the random value. Here, regardless of the policy trained, the trained action-value function Q is an estimate q_* which is the optimum action-value function. At each timestep, the agent uses the ε-greedy policy resulting from Q to choose the action $a_{t,}$ and perceives a reward r_t and enters the new state $s_{t+1,}$ and with the learning rate ι, Q is updated. In this case, the learner is the off-policy learner in which the agent learns from previous experiences with different policies which is shown by the following equation:

$$Q(s_t, a_t) \longleftarrow Q(s_t, a_t) + \iota\left[r_t + d \max_a Q(s_{t+1}, a_t) - Q(s_t, a_t)\right]. \tag{5}$$

This bellman equation taken from [25] simplifies the investigation of an algorithm and is used to provide early convergence confirmations. However, this algorithm is applicable only for the problem with a small number of states in the state space and is represented by the tabular method and is difficult for the problem with a huge number of state spaces and action spaces. That is why Huang came up with deep Q-Learning [26].

3.1.2. The Deep Q-Learning. According to Huang [26], Q-learning is one type of reinforcement learning algorithm that solves the Markov's decision problem in the classical method. But, we recall that Q-learning is restricted to solving only problems having a discrete state. This means that it converges to optimal states by obtaining an optimal policy when the state or action space is small. Representing the action value function for the reinforcement problem with larger action and state by using Q-learning can be difficult. For this reason, the function approximators such as neural networks (NN) are used by researchers to represent the action-value function. Various researchers such as [27–29] have utilized NN to depict the Q-value function for the function approximators and they have gone over various reinforcement learning tasks such as playing a video game. Enhanced Q-Learning and deep Q-learning (DQN) are employed to learn the optimal policy of a problem containing a large state and action space.

DQN is a model-free RL algorithm in which the agent does not require to know just how the environment behaves and works. The two key features for training deep neural network (DNN) are the target network and experience replay. Every experience denoted by E which is $E_t = (s_t, a_t, R_{t+1}, s_{t+1})$ is stored in a buffer memory $M = (E_1, E_2, E_3, E_4, \ldots . E_T)$. The stored experience is sampled and resampled from time to time to restore and update parameters θ_j of policy π with loss function $L(\theta_j)$ at the j^{th} iteration of training. The main function of applying experience replay is to reduce the input correlations for training Q-functions. The loss function is defined as

$$L(\theta_j) = E\left[\left(r_t + d \max_{a_{\text{next}}} Q(s_{t+1}, a_{\text{next}}; \theta_j^n) - Q(s_t, a_t; \theta_j)\right)^2\right]. \tag{6}$$

Here, θ_j^n, is the parameter for the target network and θ_j is the parameter for the Q-network. In each defined timestep, the Q-network parameter θ_j is used to update the target network parameter. s_{t+1} is the state of the environment after the agent plays an action a_t at the environmental state s_t. The loss function is calculated from the batch sampled from the replay memory.

3.2. Froze Lake Environment. For this paper, we have selected frozen lake (FL) grid environment for its simplicity in discrete state space. It comprises of a 4×4 grid demonstrating a frozen surface, where the agent can select to move one step down, up, right, or left. The aim is to cross the lake from the starting point of the top left corner to the endpoint of the bottom right corner. Nevertheless, the particular grid point of the environment corresponds to holes in the ice, and when the agent steps on them, the iteration of the episode terminates and it has to restart from the initial state. One of the following conditions makes the terminations of iterative episode:

(1) The agent reaches the goal G

(2) The agent iterates and reaches a maximum of 3000-time steps

(3) The agent falls into a hole (red subgrid).

For each episode in which the goal is reached, the agent takes a reward of +1 for successfully reaching the goal, and −0.1 if the agent is stepping into one hole.

As shown in Figure 1 in frozen lake environment (extended from Hu and Hu [14]), it is expected from the quantum agent to move from the grid start location (S) found at the left-up corner to the right bottom corner goal location (G). The lake may not all be frozen which means there may be various holes (red color) on the path. It is the responsibility of the quantum agent to train and avoid walking into these hole locations, and if not, the quantum agent will get a huge negative reward, and then an episode iteration will be ended. To realize this, we have assigned a little negative return value to each wrong move. The frozen lake environment shown in Figure 1 has 16 eigen states (4 holes+ 12 frozen states) and 4 eigen actions (MoveTop, MoveDown, MoveLeft, and MoveRight). Without applying trainable input data weights, we encode each point on the grid as one of the computational basis states of a 4-qubit quantum system ($|0000\rangle \ldots |1111\rangle$).

From Table 2 , we have seen that most research studies have been performed in quantum reinforcement learning and they have realized the possibility of enhancing classical reinforcement learning with quantum computing. The literature depicts that the performance of the enhanced model was better than the classical one. Additionally, in the last five years, the study on the field of quantum computing has become significant and has grown as of frightening data

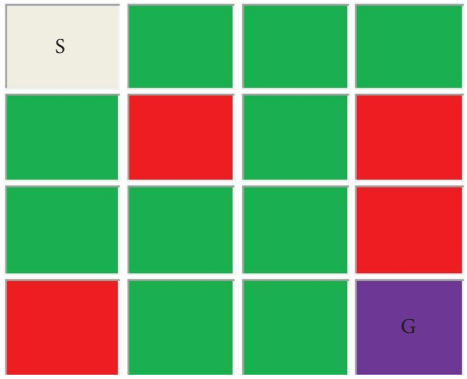

Figure 1: Schematic view of the standard frozen lake environment.

increment and the idea of promising quantum computing technologies will change today's computing world. This field got attention from various researchers of companies such as Google [34], IBM [35], and Microsoft [36], and they are racing to develop quantum computers and have realized the applicability of quantum computing technology in classical machine learning algorithms [37].

3.3. Variational Quantum Circuits. The parameterized quantum circuit **(PQC)** is a quantum circuit using tunable parameters to accomplish several mathematical tasks, such as classification, approximation, and optimization [38]. To perform such tasks, the operation of the quantum circuit model can have 4 simple steps.

3.3.1. Quantum State Preparation. This is the process of encoding input information into equivalent qubit quantum states, which can be processed in the quantum circuit later.

3.3.2. Applying Quantum Entanglement. This is the process of entangling qubit quantum states by controlled quantum gates such as CNOT gate and rotating qubits by parameterized quantum rotation gates. This process can be iterated after the update of parameters in a multilayer manner with other parameters, which attempts to improve the performance of the quantum circuit.

3.3.3. Quantum State Measurement. This is the process of decoding and evaluating the treated qubit states to a piece of proper output information.

3.3.4. Parameter Optimization. It is the last process conducted external to quantum circuits (QC) on a classical computer. The conventional classical optimizer algorithm, such as the Adam optimizer, can update the quantum circuit parameters in the way of optimizing the objective function of the algorithm. Then, the updated circuit with the new parameters has the ability to perform the calculation again from the 1st process with the next state input to the circuits. Similar to classical NN, the parameterized quantum circuit is known to estimate any continuous functions, and therefore, it has been broadly applied in QML research [39–42].

In the prototype of designed circuits shown in Figure 2, the agent is initially placed on the state $|0000\rangle$, and taking this encoded state as input, the unitary Pauli gates are applied to rotate the values on the arbitrary *x-axis* and *z*-axis. The entangling layer of the circuit entangles the values of unitary Pauli gates making the entangling between the qubits. The measurement layer measures each expectation value, which is related to the probability of measuring a certain state and how much does that state adds to the cost function.

3.4. Hybrid Classical-Quantum Deep Reinforcement Learning. The proposed hybrid framework with the designed VQC is viable on the existing noise intermediate scale quantum machine learning platform. Thanks to Xanadu for providing the PennyLane which consists of quantum machine learning libraries to simulate various tasks of quantum-enhanced machine learning, and the developed framework can resolve the VQC depth challenges by being hybridized with iterative parameter optimization (IPO) on a classical computer. Classical optimization takes place external to the quantum circuit and it provides the loss function and gradient calculations to update the parameters.

In our work, the loss function is calculated by mean squared error and RMsprop optimizers for providing an update for the parameters. Currently, PennyLane provides seven types of hardwired optimizers, which can work with the standard gradient descent with momentum, gradient descent, gradient descent with Nesterov momentum, Adam, Adagrad, and RMSprop. PyTorch and TensorFlow use optimizers that this library provides [33]. Here, we have selected *RMSprop* (the little explanation about the optimizers is found in T. Tieleman and G. Hinton, "Lecture 6.5_RmsProp: Divide the gradient by a running average of its recent magnitude") which is broadly utilized in deep reinforcement learning. The parameter used in this optimizer is the learning rate, alpha, and eps which are applied for gradient descent optimizers (GDO) only. For the frozen lake environment, the utilized strategy for the *ε-greedy* is given by

$$\varepsilon \longleftarrow \frac{100\varepsilon}{\text{episode} + 100}. \tag{7}$$

From Figure 3, the experience replay chooses *ε-greedy* action from the current state, executes it in the frozen lake environment, and returns a reward and then moves to the next state. It keeps this observation as a training sample of data. All previous experience replay observations are kept as training data. We now take the random batch of samples from this training data, so that it comprises of a combination of older and more recent samples. This batch of training data is then entered into both the quantum Q network and the target quantum network. The quantum Q network takes a current state and an action from the separate data sample and predicts the quantum Q value for that specific action which then becomes the predicted quantum Q value.

Table 2: Some quantum-enhanced ML algorithms we found from the literary review.

Related works of literature	OpenAI gym environment	Performance	Algorithm used by the researcher
Yueh Hsiao et al. [30]	Acrobot and LunarLander	Converged faster	Proximal policy optimization (PPO)
Lockwood and Si [31]	CartPole-v1 and Acrobot	Not defined	Policy gradient with baseline'
Lan [32]	Pendulum	Faster convergency	Soft actor-critic
Yun et al. [33]	CartPole-v0	Not defined	Q-Learning
Hu and Hu [14]	2×3 frozen lakes	Not defined	Deep actor-critic

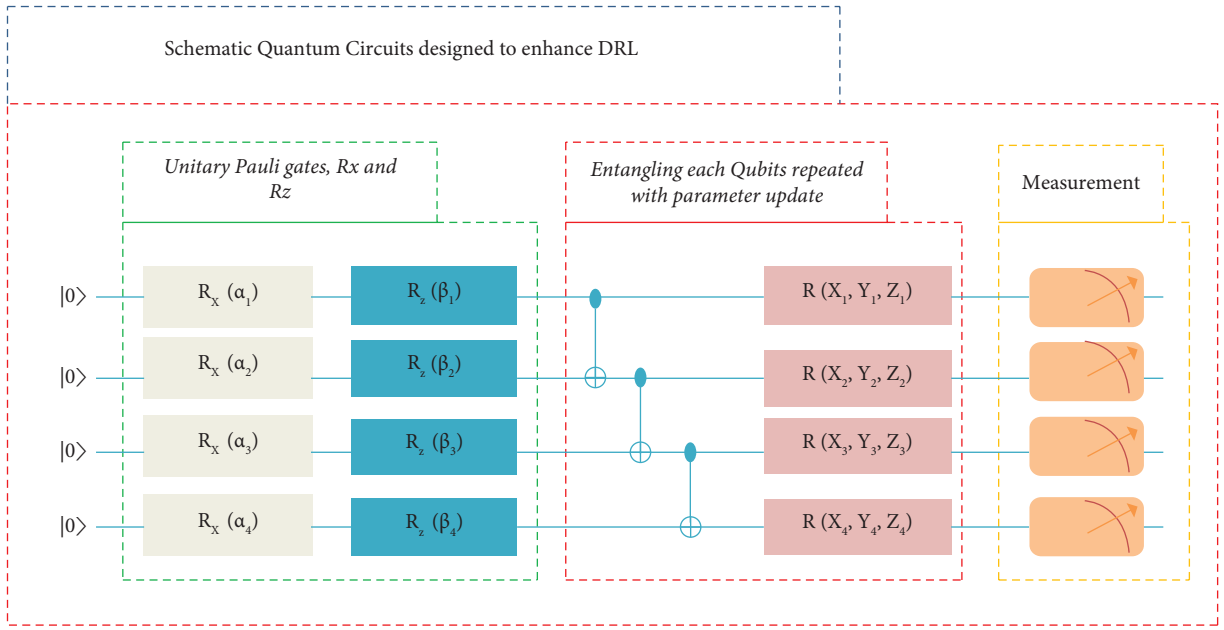

Figure 2: Variational quantum circuits.

The target quantum network takes the next state from each data sample and predicts the best Q value out of all actions that can be taken from that state and is called the target quantum Q value. The predicted quantum Q value, target Q value, and the observed return of reward from the data sample are used to calculate the loss to train the quantum Q network. Moreover, we take a broad view of VQC to the standard deep reinforcement learning for approximating the action-value function. Lastly, we examine a policy reward of different hyperparameters of deep reinforcement learning for investigating the performance of PennyLane-based quantum deep Q-learning. For the testing environment, we have selected the standard openAI gym [43] frozen-lake environment.

Algorithm 1 Quantum enhanced DQL sampled from Huang [26] and Chen et al. [13].

3.4.1. Action Selection in Quantum Enhanced Deep Q-Network. In deep reinforcement learning, the agent takes an action based on its policy. Selecting the best action among the other set of actions can provide a better reward for the agent. Here in the enhanced framework, selecting the action is determined via quantum expectation values of n number of quantum bits (Qubits). We have four action and four qubit systems which means that there are four expectation values after the measurement is applied on a four-qubit quantum system. Then, we represent each measurement of the qubit output wire (port) from zero to three. Then, the action with the index of the largest expectation value of the output wire is selected. For example, let us say our four qubit quantum system has the output wire with the set actions $A = (a1, a2, a3, \text{and } a4)$. If the set of numerical expectation values of all actions after the measurement is $e = (e\ (a1), e\ (a2), e\ (a3), \text{and } e\ (a4))$ and the expectation value of $e\ (a3)$ is greater than all four expectations, then the action that must be chosen is the index of action $a3$ which is 2. Now, action 2 is selected based on the expectation values and passed to the environment for testing. We recall that, in the FL environment, there are 4 actions in an agent's action space. These actions are MoveTop, MoveDown, MoveLeft, and MoveRight, and their output wires of four-qubit quantum systems are indexed with 0, 1, 2, and 3, respectively. If the output of wire indexed with 3 has the greatest value of expectation, then the action that should be chosen by the quantum agent is to go up one stage from a current state in a frozen lake (FL). Quantum simulators such as PennyLane [44] and IBM Qiskit [45] are used to calculate the expected value of quantum measurements on the current classical computer. The result of expectation values after the quantum

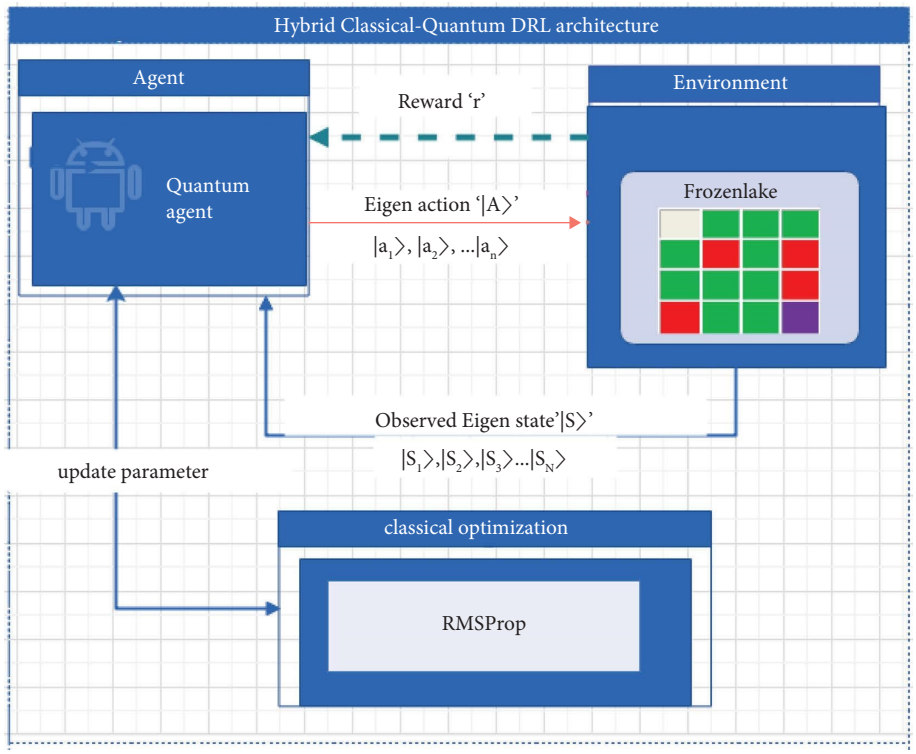

Figure 3: Proposed architecture.

Set replay memory M to state size N
Initialize action-value function quantum circuit Q with arbitrary parameters θ
For episode $e = 1, 2, 3, 4, \ldots\ldots E$ **do**
  Initialize State $s1$ from the set state S and encode it into
  the quantum state using basis encoding
  for the time step $t = 1, 2, 3, \ldots T$ do
    With probability ε, select a random action a_t
    otherwise, select the optimal action at $a_t = \max_a q_*(s_t, a; \theta)$ from the result of quantum circuit
    Execute the selected action a_t and see the reward r_t and the next state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in replay memory M
    Sample a random minibatch of transitions (s_b, a_b, r_b, s_{b+1}) from the replay memory M

$$y_b = \begin{cases} r_b & \text{for the terminal } s_{b+1} \\ r_b + d \max_{a_{\text{next}}} Q(s_{b+1}, a_{\text{next}}; \theta) & \text{for a non} - \text{terminal } s_{b+1} \end{cases}$$

    Perform a gradient descent step on $(y_b - Q(s_b, a_b; \theta))^2$
  end for
end for

Algorithm 1: Quantum enhanced deep Q-learning.

measurement is deterministic. Specifically, let us take the FL and let us say that the reinforcement learning quantum agent receives the state of 8 on the 4×4 grid which is 1000 in binary. The basis encoding with a two unitary Pauli gets are applied and this number must be converted to a quantum state such as $|1\rangle \otimes |0\rangle \otimes |0\rangle \otimes |0\rangle$. The encoded quantum state passes through the quantum circuit and at the end the expectation value will be measured. Here, the result after the measurement is either 0 or 1 if the system measures a first qubit $|1\rangle$ above, but the possibility of being 0 or 1 is stochastic or random. In random measurement, the agent selects only a single qubit randomly and uses that value for action selection. Each qubit of the prepared quantum state $|1\rangle \otimes |0\rangle \otimes |0\rangle \otimes |0\rangle$ can pass via blocks of the circuit and the measurement layer measure each qubit's expectation value. For each qubit state, there are multiple number of repeated measurements but the chance of becoming exactly in the state $|0\rangle$ and state$|1\rangle$ is a probability distribution. This means that, we can predict the probability of being in state $|0\rangle$ and $|1\rangle$ by averaging the expectation value of repeated measurement. For example, let us measure T times the state of the first qubit $|1\rangle$ and from this measurement, if K times the state of some qubit gives 1, then the probability $P(1)$and $P(0)$ is calculated as

$$P(1) = \frac{K}{T},$$
$$P(0) = \frac{T-K}{T}. \tag{8}$$

The average value can be produced after the measurement, and this average value is called the value of quantum expectation.

To be specific, let us take the following example. The quantum agent again wants to measure the expectation value of $|1\rangle \otimes |0\rangle \otimes |0\rangle \otimes |0\rangle$ (See Table 3) and let us say that each qubit has been measured 500 times.

Furthermore, each measurement on a quantum bit of n qubits can be executed concurrently in parallel.

Table 3: The method of calculating the expectation value for action selection (this is the assumption).

	$\|1\rangle$	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
Total number of repeated measurements	500	500	500	500
Total number of measurements which gives 1	330	400	350	190
Total number of measurements which gives 0	170	100	150	310
Probability of getting 1 or P (1)	0.66	0.8	0.7	0.38
Probability of getting 0 or P (0)	0.34	0.2	0.3	0.62
Expectation value	0.66	0.8	0.7	0.62

3.5. State Encoding. In the parameterized quantum circuits (PQC), to process classical data, data encoding is needed so that, classical data is converted into the quantum state. There are quite a lot of types of quantum encoding frequently used in quantum machine learning (QML) applications (see Schuld and Petruccione [46] for more information about quantum encoding). Some of them are amplitude encoding utilized by Antipov et al. [47], computational basis encoding, angle encoding, and directional encoding. Encoding classical state to quantum state is currently a hot research area and various methods of encoding can provide various quantum advantages for machine learning and natural language processing. Some of the encoding methods are currently not realized on real quantum computer hardware because they need a large number of quantum circuits made from complex quantum gate interaction.

3.5.1. Computational Basis Encoding for Frozen Lake Environment. Every quantum-enhanced machine learning investigation needs quantum computers (or quantum simulators) that are used to process classical data (states). The first step to processing classical data is encoding the classical data into quantum states. In fact, encoding the classical data is the most vital step in quantum data processing in a near-term quantum computer. The potential power of a quantum machine learning algorithm in the noise-intermediate scale quantum (NISQ) era is determined by the way we encode classical data [13, 48]. Our study mainly focused on hybrid quantum-classical computation and in hybrid quantum-classical computation, the basis encoding techniques are employed by [33, 49] and it is well suited for the DRL problem with fewer discrete states. In basis encoding, first, the state is represented by decimal numbers and then converted to a binary number. To be enhanced by quantum computation, these classical binary bits must be encoded to a quantum state to be processed by quantum circuits. The quantum state with n number of qubits is represented by the following equation:

$$|\psi = \sum_{[q_1,\ldots,q_n]\in\{0,1\}^n} \left(c_{q1\ldots qn}\right)|q_1\rangle \otimes |q_2\rangle \otimes |q_3\rangle \otimes \ldots \otimes |q_n\rangle. \tag{9}$$

The value of $C_{q1}\ldots C_{qn}$ is an element of complex number C and they are said to be the amplitude of quantum state, and the square sum of these complex numbers are measurement probabilities of finding each of the corresponding states and their sum must be equal to 1, and this is represented as

$$p\left(|q_1\rangle \otimes |q_2\rangle \otimes |q_3\rangle \otimes \ldots \otimes |q_n\rangle\right) = \sum_{[q_1,\ldots,q_n]\in\{0,1\}^n} \left(c_{q1\ldots qn}\right)^2 = 1. \tag{10}$$

Here, we have taken a frozen lake environment with four qubit systems. The four-qubit quantum system gives 2^4 or 16 possible basis states. The single-qubit unitary Pauli rotation operation is used to encode the classical bits into the 16 quantum states.

Table 4 shows the encoding of classical input state from the selected frozen lake environment into quantum states of the quantum circuit which needs the application of a single qubit unitary rotation method. The single quantum gates of arbitrary rotational angle ω, with any κ axis rotation is given by the following equation:

Table 4: The method of encoding binary state to quantum state.

Decimal number	Classical binary numbers	Representation of entangled quantum states
0	0000	$\|0\rangle \otimes \|0\rangle \otimes \|0\rangle \otimes \|0\rangle$
1	0001	$\|1\rangle \otimes \|0\rangle \otimes \|0\rangle \otimes \|1\rangle$
2	0010	$\|1\rangle \otimes \|0\rangle \otimes \|1\rangle \otimes \|0\rangle$
.....	.....	.....
13	1101	$\|1\rangle \otimes \|1\rangle \otimes \|0\rangle \otimes \|1\rangle$
14	1110	$\|1\rangle \otimes \|1\rangle \otimes \|1\rangle \otimes \|0\rangle$
15	1111	$\|1\rangle \otimes \|1\rangle \otimes \|1\rangle \otimes \|1\rangle$

$$R_\kappa(\omega) = e^{1/2\left[-i\omega\delta_\kappa\right]},$$

$$k = \begin{cases} \text{rotate along } x-\text{axis}, Rx(\omega) = \cos\left(\frac{\omega}{2}\right) * I - i * \sin\left(\frac{\omega}{2}\right), \\ \text{rotate along } y-\text{axis}, Ry(\omega) = \cos\left(\frac{\omega}{2}\right) * I - i * \sin\left(\frac{\omega}{2}\right), \\ \text{rotate along } z-\text{axis}, Rz(\omega) = \cos\left(\frac{\omega}{2}\right) * I - i * \sin\left(\frac{\omega}{2}\right). \end{cases} \tag{11}$$

Here, δ_x, δ_y, and δ_z are the Pauli matrixes of Pauli gates, which can be represented by $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, $\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$, and $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$, respectively. We have utilized the Pauli X gate and Pauli Z gate for the abovementioned quantum circuits (Figure 2) with α_i and β_i rotation angle, respectively. The rotation angle is shown by the following equation:

$$\begin{aligned} \alpha_i &= \Pi * b_i, \\ \beta_i &= \Pi * b_i, \end{aligned} \tag{12}$$

where i is the qubit index, b is a bit, and Π is a radian so that we can encode for each state.

From Table 5, we can see that the 14-state entangled with the CNOT gate, $|1\rangle \otimes |1\rangle \otimes |1\rangle \otimes |0\rangle$ can have α_1, α_2 , α_3, and α_4 values of $(\Pi, \Pi, \Pi, \text{and } 0)$ and the same for all β. From equation (11), we can get the following expression:

$$\begin{aligned} R_k(\Pi) &= -i\delta_k, \\ R_k(0) &= I, \\ Rx(\Pi) &= -i\delta_x, \\ Rz(\Pi) &= -i\delta_z. \end{aligned} \tag{13}$$

Now, from the given circuit (Figure 2), we have used Rx (α_i) and Rz (β_i) for encoding the state and both take $|0\rangle$ as the initial quantum state. The state after rotating the Pauli X gate and Pauli Z gate with Π rad is given by

$$\begin{aligned} RX(\Pi)RZ(\Pi)|0\rangle &= (-i\delta_x)(-i\delta_z)|0\rangle = |1\rangle, \\ \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}(\Pi)\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}(\Pi)|0\rangle &= (-i\delta_x)(-i\delta_z)|0\rangle = |1\rangle, \end{aligned} \tag{14}$$

and the state after rotating the Pauli X gate and Pauli Z gate with 0 degree is given by

$$\begin{aligned} RX(0)RZ(0)|0\rangle &= (I)(I)|0\rangle = |0\rangle, \\ \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}(0)\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}(0)|0\rangle &= (I)(I)|0\rangle = |0\rangle. \end{aligned} \tag{15}$$

These are applicable for all states and the result of the entangled qubits using a controlled CNOT gate (see Figure 2) is given by the following equation:

$$\begin{aligned} R(Xi, Yi, Zi) &= Rx(Xi)Ry(Yi)Rz(Zi), \\ R(Xi, Yi, Zi) &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}(Xi)\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}(Yi)\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}(Zi). \end{aligned} \tag{16}$$

The three parameters ($Xi, Yi, \text{and } Zi$) for this general single qubits unitary rotation operator are used to provide optimization.

4. Experimental Setup and Results

4.1. Experimental Setup. Quantum enhanced deep reinforcement learning (QDRL) framework is developed in the python programming language. Programming, experimental visualization, and all advance of this research study

Table 5: Entangling qubits.

State with the decimal number	Entangled quantum state	Value after applying Pauli gate of X gate and Y gate operators	
		α_i ($i = 1, 2, 3, 4$)	β_i ($i = 1, 2, 3, 4$)
0	$\|0\rangle \otimes \|0\rangle \otimes \|0\rangle \otimes \|0\rangle$	(0, 0, 0, 0)	(0, 0, 0, 0)
1	$\|0\rangle \otimes \|0\rangle \otimes \|0\rangle \otimes \|1\rangle$	$(0, 0, 0, \Pi)$	$(0, 0, 0, \Pi)$
…….	…….	…….	…….
14	$\|1\rangle \otimes \|1\rangle \otimes \|1\rangle \otimes \|0\rangle$	$(\Pi, \Pi, \Pi, 0)$	$(\Pi, \Pi, \Pi, 0)$
15	$\|1\rangle \otimes \|1\rangle \otimes \|1\rangle \otimes \|1\rangle$	(Π, Π, Π, Π)	(Π, Π, Π, Π)

was completed in Python 3. We employed Python 3.7.8 for accessing the PennyLane platform. The other program that we have employed to simply manage the package and control the environment is the anaconda. We utilized Jupyter Notebook as the development environment for writing and implementing the python code. The Matplotlib was utilized for the plotting and was mostly tied to NumPy in real-world applications and it offers functions for the plots such as line, pie charts, scatter, histogram, and others. Hence, the implementation procedures and the hyperparameters needed for the developed framework is discussed in the following table. At the last, the researcher discusses the overall outcome of the study with the experiment-based interpretations, and all hyper parameters used through this study is depicted in table 6.

Table 6: Some hyperparameters used in the study.

Hyperparameters	Value assigned
Number of qubits	4
Number of layers	2, 3, 5
Epsilon	1
Batch size	5
Maximum time step	3000
Maximum episode	1000
Gamma	0.99
Alpha	0.1, 0.2, 0.3, 0.4
Learning rate	0.01
eps	$1e-08$

4.2. Experimental Result. The frozen lake environment is used in our work to evaluate the performance of DRL algorithms on quantum circuits. Previous practitioners such as Hu and Hu [14] used a grid-based environment similar to the frozen lake but with smaller size (2×3) grids. For this study, we have used the standard frozen lake values of grid size (4×4) by increasing the number of qubits to 4 which can have 2^4 basis states. For the designed circuits, we have simulated numerically with the PennyLane [44]. A numeric simulator is used to discover the prospects of using quantum computing technology to solve DRL task and their optimality. To expand the former work of Hu and Hu [14], we present algorithms that improve upon the previous results. We have applied the learning techniques to OpenAI Gym [43] and frozen lake environments which are more complex with the number of states and layers than in the previous work [14]. To verify the trainability of the framework, we take 1000 episodes of the standard frozen lake experiment. The horizontal coordinate signifies an episode in the enhanced learning process, and the vertical axis represents the number of rewards returned to the agent for taking some actions on the environment. The experiment considers the trainability of enhanced DRL and the effect of the number of layers on the enhanced DRL framework. According to the experiment, the quantum enhanced DRL is trainable in all cases but, the time it takes to converge to the optimal value is different for the different quantum layers.

Figures 4(a)–4(c) show the learning ability of different quantum agents based on the different layers. From Figure 4(a), it is seen that the agent executed with two layers hit the maximum negative reward of −15.09 on episode 148. The agent tried and converges to the optimal Q value by taking the reward of 0.91 after the training episode of 525. The second agent from Figure 4(b) is executed with 3 layers and converges to the Q value with a total reward of 0.95 after the 397th episode. The third agent is executed with 5 layers and converges to the optimal Q-value with the reward of 0.95 after the 72^{nd} episode. Here we can conclude that, on the framework, the agent explores more if the framework uses a smaller number of layers and this means that, it exploits the optimal value when more layers are used. This means, when the quantum layer increases from 2 to 5, the framework converges faster without going for testing on various episodes. In the other scenario, the QDRL is better than classical DRL to reduce the parameters (weights in the classical case). Here, a number of parameters in PQC subjected to optimization can be calculated as

$$C(\theta) = \text{number of qubits} * [(\text{number of block in circuits} * \text{number of layers} + 1)], \tag{17}$$

where θ is the number of quantum layers, the number of circuit blocks used in this research study is 3, and the +1 is the added bias which is also subjected to optimizations [13]. Classically, four layers of NN which need 16 x 4 or 64 parameters are needed to represent the 4x 4 frozen lake problem [13]. In QDRL, we can represent this problem with

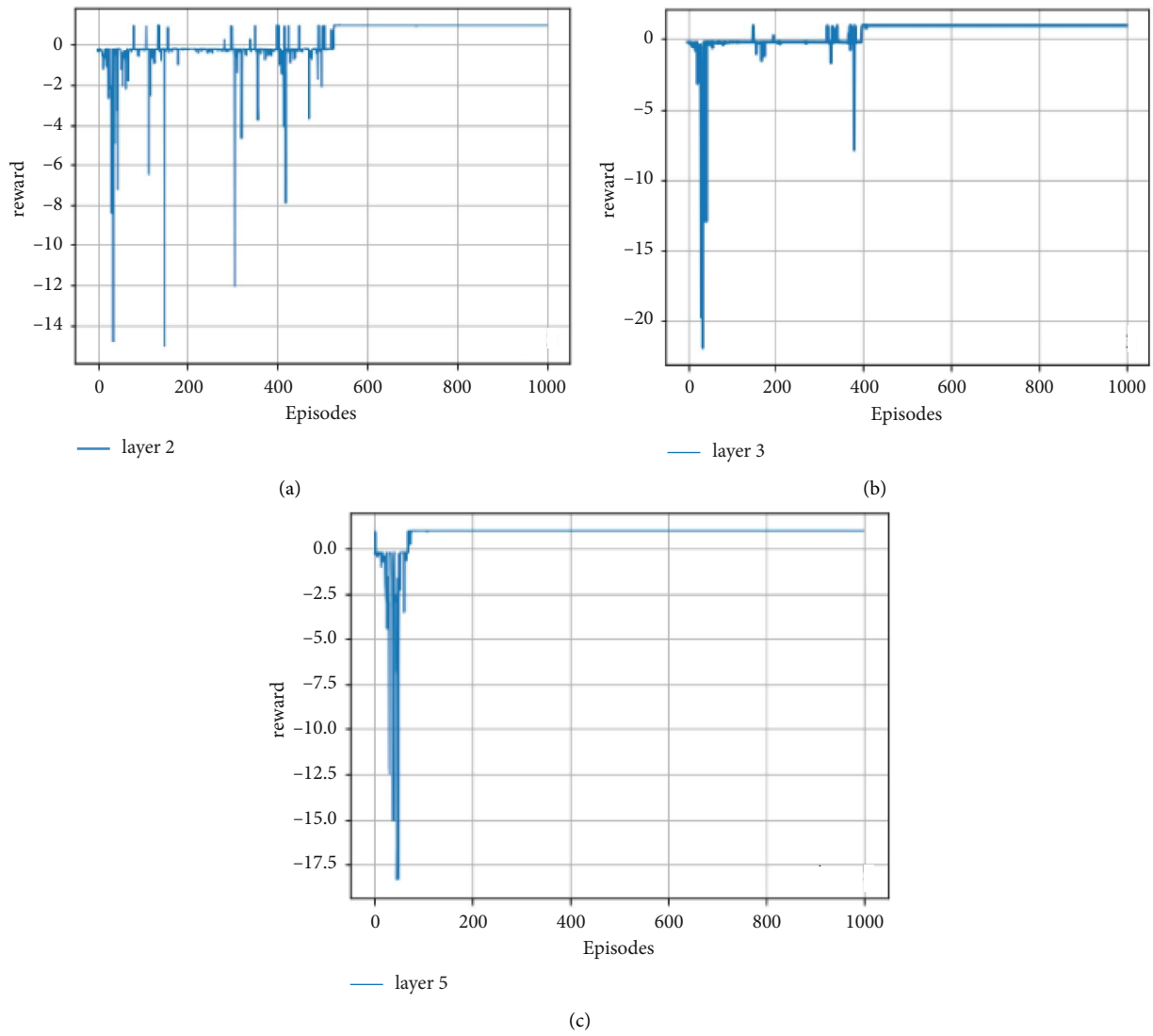

Figure 4: The agents reward with different quantum layers (a, b, and c): (a) agent with quantum layer 2, (b) agent with quantum layer 3, and (c) agent with quantum layer 5.

4x(3 x 4 + 1) or 52 parameters which reduces the number of parameters by 12. So, QDRL is a better choice for parameter reduction. On the other hand, the quantum-enhanced action selection strategy makes a good balance between exploration and exploitation using expectation values, which increases the quantum agent learning ability and promises to solve searching problems over entire state-action (s and a) space search in classical DRL.

In the case of seeing the performance in different learning rates (alpha), the agent running with 0.1 alpha value hits the maximum time steps of 2012. When the alpha value is 0.1 or smaller, the agent explores much more but it learns very slowly, so the training process converges to the optimal state very slowly.

Figure 5 gives a precise description of enhanced framework learning results. We record every single learning epoch to verify the outcomes of enhanced DRL algorithms with selected 0.1, 0.2, 0.3, and 0.4 learning rates or alpha values (see Figures 5(a)–5(d), respectively). From this result, we can conclude that the greater the learning rate values in quantum deep reinforcement learning, the fewer timesteps it takes to move from the start point to the goal state. The result of QDRL shows advantages with 0.4 learning rate or alpha values. The advantages are the use of quantum representation. The quantum representation uses the quantum superposition strategies of quantum mechanics in which the updating method is carried out via quantum parallelism. Quantum parallelism will be more useful in the near future when quantum enhanced device comes into use rather than simulating on conventional computers.

From Table 7, we can see that when the agent got the reward of 0.95, it costs the agent 6 timesteps which is the time step the agent takes to converge into the optimal state. Generally, the time steps of an agent with a learning rate from 0.2 to 0.4 for the last 200 episodes is similar which is 6 and the reward also converges to the optimal value of 0.95.

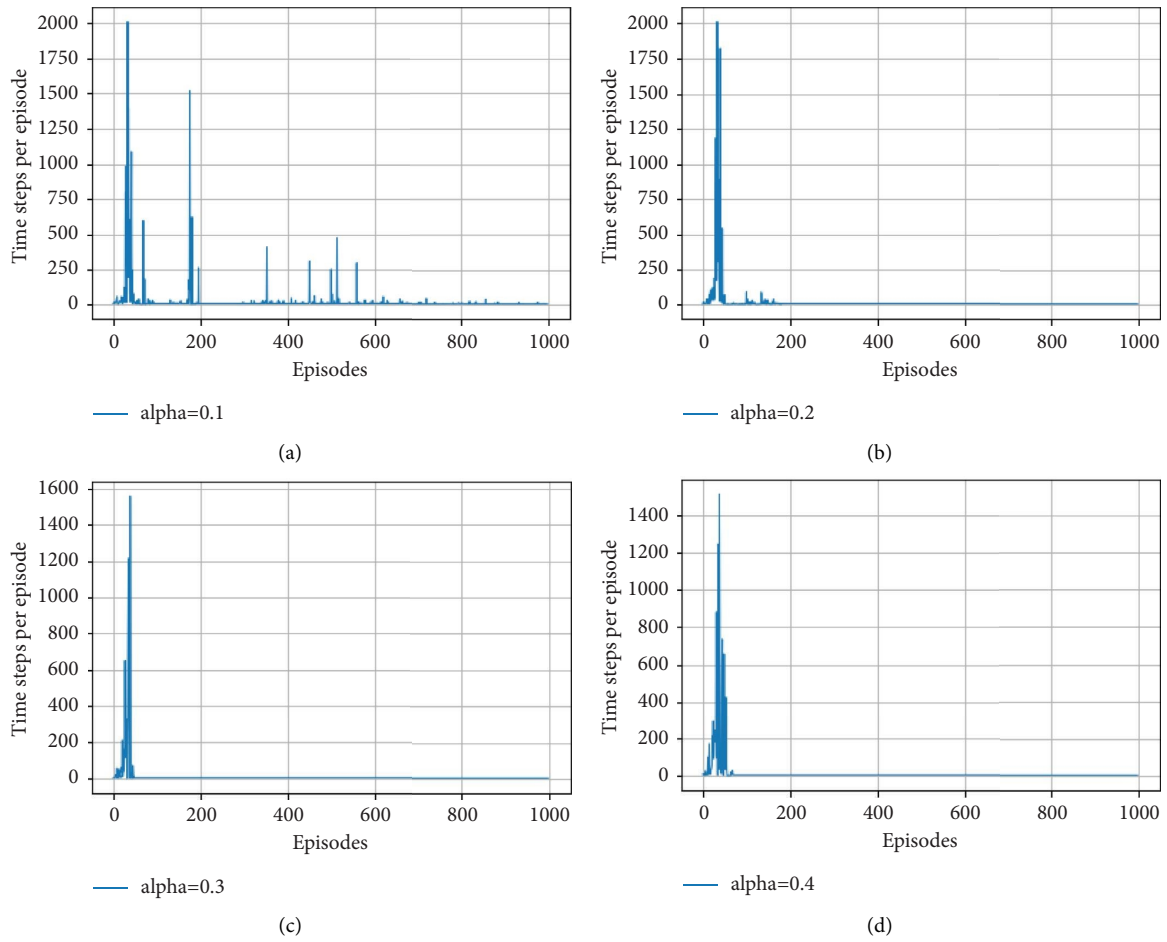

Figure 5: The timestep comparison for different learning rates: (a) time step with 0.1 alpha value, (b) time step with 0.2 alpha value, (c) time step with 0.3 alpha value, and (d) time step with 0.4 alpha value.

Table 7: The reward and timestep of the agent for the last 200 episodes.

Episode	799	849	899	949	999	Alpha
Reward	−0.24	−0.24	−0.26	0.95	0.95	←0.1
	0.95	0.95	0.95	0.95	0.95	←0.2
	0.95	0.95	0.95	0.95	0.95	←0.3
	0.95	0.95	0.95	0.95	0.95	←0.4
Timesteps	8	8	7	6	6	←0.1
	6	6	6	6	6	←0.2
	6	6	6	6	6	←0.3
	6	6	6	6	6	←0.4

5. Conclusions

Though the main focus of this study is enhancing DRL with quantum machine learning and analyzing the learning performance of that framework, we have studied the background of this technology and its applicability as well as and the need of applying this technology to current machine learning models. This can be seen from two sides which are the theoretical side and the technical side. On behalf of the theoretical study, we have studied that, the motivation behind this new machine learning paradigm is to obtain improved learning performance. The study also presents the state-of-the-art in the development of quantum science in the area of AI and encourages the enhancement of classical ML technology. Specifically, the representation of quantum computations is generally different from the current classical computations, and various features of quantum computation are prospectively evolving.

Machine learning is a feature that is currently influenced by the theory of quantum computation. There are various demonstrations and conformations that quantum mechanical phenomena such as superposition, entanglement, and quantum inference which can change the current machine learning algorithm such as supervised, unsupervised, and reinforcement learning for its betterment in providing sustainable speed up on processing data. In this study, we also analyzed the need for classical data and the method of encoding it to the quantum state to process it on cloud-based provided resources of quantum machine learning. To be accessed and processed on quantum processing units, the data need to be encoded into the quantum state. The type of encoding can be different from task to task for the specific problem that needs to be solved.

Generally, we have analyzed that knowing the encoding techniques that can handle the problems with that limited resource is expected from the researcher as the existing quantum devices are very small and intermediate scale. They only cover a small number of qubits. The other thing is considering the complexity of circuits. Minimizing the number of quantum circuits is needed to reduce the complexity and needs to minimize the number of parallel quantum actions required to grasp the quantum encoding. The well-known quantum data encoding is computational basis encoding which provides codification of classical data to a quantum state. Quantum phenomena such as superposition and entanglement can provide parallelization for all quantized states. Lastly, we have confirmed that the computational basis encoding is not efficient for the quantum deep reinforcement learning (QDRL) problem that needs a huge number of qubits but is efficient for tasks solved by applying a smaller number of qubits. The standard DRL task can be enhanced with a smaller number of qubits because the trainability of the framework can be investigated on a standard frozen lake which contains 4×4 matrices of state action spaces.

On behalf of the practical side, the outcomes of experiments validate the feasibility of an enhanced deep Q-learning framework and then verify its learning performance for the agent with different quantum layers and different learning rates. The experiment depicts that the value of a number of layers and the number of learning rates can affect the learning performance of an agent and as soon as a quantum-enhanced deep reinforcement learning turn on a real quantum computer, it can be efficiently used to improve the quantum robot learning for achieving some important tasks. Generally, we have verified the applicability of quantum computation to machine learning specifically deep reinforcement learning, and the further exciting results are what we expect after this enhanced deep Q-learning run on fully quantum computers in the nearby future.

6. Recommendations

Quantum-enhanced machine learning as a whole and quantum-enhanced deep reinforcement as specific is a newly quantum computing-inspired learning framework that has a theoretical and experimental gap. Most of the experimental gaps can be solved only when the quantum resource is available than today's resources. Most companies such as IBM, Google, and Xanadu are trying to handle these problems by providing cloud-based resources. Though the resource is enough for some simple problems, it is not efficient for complex problems such as the Atari games and Game of Go. Here, every activity such as state representation (preparation and encoding), policy evaluation and policy iteration, action selection, and optimization in reinforcement learning needs theoretical and experimental studies to enhance it with quantum computing technology. In the case of state representation, we have applied basis encoding techniques to encode the classical states into quantum states and we recommend that, by seeing the framework's performance applying other encoding techniques such as amplitude and angle encoding. In this paper, we mostly discussed the deep reinforcement problem with discrete state spaces. But the expected recommendation here is to extend this enhanced framework to DRL problems with continuous state spaces efficiently.

Data Availability

The data used to support the findings of the study can be obtained from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The research was funded by Bule Hora University, Informatics College.

References

[1] M. Schuld, F. Petruccione, S. Africa, and S. Africa, "Quantum machine learning," *Encyclopedia of Machine Learning and Data Mining*, vol. 549, pp. 1–10, 2016.

[2] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[3] C. Zhao and X. Gao, "Qdnn deep neural networks with quantum layers," *Quantum Machine Intelligence*, vol. 3, pp. 1–9, 2021.

[4] M. Cacciapuoti, R. Caleffi, L. Van Meter, and L. Hanzo, "When entanglement meets classical communications: quantum teleportation for the quantum internet," *IEEE Transactions on Communications*, vol. 68, no. 6, pp. 3808–3833, 2020.

[5] S. B. Ramezani and A. Sommers, "Machine learning algorithms in quantum computing a survey," in *Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN)*, vol. 2, Padua, Italy, July 2020.

[6] A. Ishtiaq and S. Mahmood, "Quantum machine learning: fad or future?," 2021, http://arxiv.org/abs/2106.10714.

[7] J. Preskill, "Quantum computing in the NISQ era and beyond," pp. 1–20, 2018, https://arxiv.org/abs/1801.00862.

[8] F. Leymann and J. Barzen, "The bitter truth about gate-based quantum algorithms in the NISQ era the bitter truth about gate-based quantum algorithms in the NISQ era," 2020, https://arxiv.org/abs/2006.02856.

[9] V. Dunjko, J. M. Taylor, and H. J. Briegel, "Quantum-enhanced machine learning," *Physical Review Letters*, vol. 117, no. 13, pp. 1–6, 2016.

[10] C. Soto-Paredes and J. Sulla-Torres, "Hybrid model of quantum transfer learning to classify face images with a COVID-19 Mask," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 10, pp. 826–836, 2021.

[11] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum algorithms for supervised and unsupervised machine learning," pp. 1–11, 2013, http://arxiv.org/abs/1307.0411.

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

[13] S. Y. Chen, C. H. Yang, J. U. N. Qi, P. Chen, X. Ma, and H. Goan, "Variational quantum circuits for deep reinforcement learning," *IEEE Access*, vol. 8, pp. 141007–141024, 2020.

[14] W. Hu and J. Hu, "Reinforcement learning with deep quantum neural networks," *Journal of Quantum Information Science*, vol. 9, no. 1, pp. 1–14, 2019.

[15] G. D. Paparo, V. Dunjko, A. Makmal, M. Angel Martin-Delgado, and H. J. Briegel, "Quantum speedup for active learning agents," *Physical Review X*, vol. 4, no. 3, pp. 1–14, 2014.

[16] M. Klusch, "Toward quantum computational agents," *Agents and Computational Autonomy*, Springer, Berlin, Germany, 2004.

[17] F. Neukart, D. Von Dollen, C. Seidel, and G. Compostella, "Quantum-enhanced reinforcement learning for finite-episode games with discrete state spaces," *Frontiers in Physics*, vol. 5, pp. 1–10, 2018.

[18] L. Lamata, J. C. Retamal, E. Solano, and F. A. Ca, "Multiqubit and multilevel quantum reinforcement learning with quantum technologies," pp. 1–25, 2018, https://arxiv.org/abs/1709.07848.

[19] B. E. Asenbeck, A. Hamann, T. Strömberg et al., "Experimental quantum speed-up in reinforcement learning agents," *Nature*, vol. 591, no. 7849, pp. 229–233, 2021.

[20] H. Zhao, L. Peng, and H. Yu, "Quantized model-free adaptive iterative learning bipartite consensus tracking for unknown nonlinear multi-agent systems," *Applied Mathematics and Computation*, vol. 412, Article ID 126582, 2022.

[21] D. Liu, "Adaptive discrete communication bottlenecks with dynamic vector quantization," 2022, http://arxiv.org/abs/2202.01334.

[22] L. Lamata, "Basic protocols in quantum reinforcement learning with superconducting circuits," *Scientific Reports*, vol. 7, no. 1, 2017.

[23] S. Albarrán, "Reconstruction of a photonic qubit state with reinforcement learning," *Advanced Quantum Technologies*, vol. 2, no. 7-8, pp. 1800074–1800078, 2019.

[24] D. Daoyi Dong, C. Chunlin Chen, H. Hanxiong Li, and T. J. Tzyh-Jong Tarn, "Quantum reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 5, pp. 1207–1220, 2008.

[25] D. Amelia, "Machine learning for quantum and complex systems," 2020, https://openresearch-repository.anu.edu.au/handle/1885/220395.

[26] Y. Huang, "Deep Q-networks," *Deep Reinforcement Learning*, vol. 18, pp. 135–160, 2020.

[27] V. Mnih, "Asynchronous methods for deep reinforcement learning," *International Conference on Machine Learning*, vol. 48, pp. 1928–1937, 2013, http://arxiv.org/abs/1301.3781.

[28] K. Kavukcuoglu, D. Silver, A. A. Rusu et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[29] Y. Zhang, I. Clavera, B. Tsai, and P. Abbeel, "Asynchronous methods for model-based reinforcement learning," 2019, http://arxiv.org/abs/1910.12453.

[30] U. The, H. P. C. Innovation, D. Through, and Q. Computing, "Untangling the HPC innovation dilemma through quantum computing executive summary," pp. 1–40, 2021, https://atos.net/wp-content/uploads/2021/11/atos-iqm-atos-state-of-quantum-hpc-research-2021.pdf.

[31] O. Lockwood and M. Si, "Reinforcement learning with quantum variational circuits," 2008, https://arxiv.org/abs/2008.07524.

[32] Q. Lan, "Variational quantum soft actor-critic," 2021, http://arxiv.org/abs/2112.11921.

[33] W. J. Yun, S. Jung, J. Kim, and J. Kim, "Ntroduction to quantum reinforcement learning theory and PennyLane-based implementation," 2021, https://arxiv.org/pdf/2108.06849.pdf.

[34] E. Parker, "An assessment of the U . S . and Chinese industrial bases in quantum technology," 2022, https://www.rand.org/pubs/research_reports/RRA869-1.html.

[35] S. Nishio, Y. Pan, T. Satoh, H. Amano, and R. Meter, "Extracting success from IBM's 20-qubit machines using error-aware compilation," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 16, no. 3, pp. 1–25, 2020.

[36] K. Srivastava and P. R. Srivastava, "Quantum algorithm for quicker clinical prognostic analysis: an application and experimental study using CT scan images of COVID-19 patients," *BMC Medical Informatics and Decision Making*, vol. 21, no. 1, pp. 1–14, 2021.

[37] M. Mohseni, P. Read, and H. Neven, "Commercialize early quantum technologies," 2017, https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45919.pdf.

[38] E. J. Kuo, Y. L. L. Fang, and S. Y. C. Chen, "Quantum architecture search via deep reinforcement learning," pp. 1–31, 2021, http://arxiv.org/abs/2104.07715.

[39] C. H. Yang, J. Qi, S. M. Siniscalchi, S. Y. Chen, and P. Chen, "Decentralizing feature extraction with quantum convolutional neural network for automatic speech recognition," *Xiaoli Ma School of Electrical and Computer Engineering, Georgia Institute of Technology, USA Brookhaven National Laboratory, NY, USA and 3 IB*, vol. 78, pp. 6523–6527, 2021.

[40] J. Choi and J. Kim, "A tutorial on quantum convolutional neural networks (QCNN)," pp. 2019–2022, 2020, https://arxiv.org/abs/2009.09423.

[41] P. Wang, "Qgan quantized generative adversarial networks," 2018, https://arxiv.org/abs/1901.08263.

[42] Y. U. Gong, S. Li, Y. A. N. Ding et al., "QCNN inspired reconfigurable keyword spotting processor with hybrid data-weight reuse methods," *IEEE Access*, vol. 8, pp. 205878–205893, 2020.

[43] G. Brockman, "OpenAI gym," pp. 1–4, 2016, http://arxiv.org/abs/1606.01540.

[44] V. Bergholm, "PennyLane: automatic differentiation of hybrid quantum-classical computations," pp. 1–12, 2018, http://arxiv.org/abs/1811.04968.

[45] M. Telahun, "Exploring information for quantum machine learning models," 2020, https://ir.library.louisville.edu/etd/3433.

[46] M. Schuld and F. Petruccione, *Supervised Learning With Quantum Computers*, Springer, Heidelberg, Germany, 2018.

[47] A. V. Antipov, E. O. Kiktenko, and A. K. Fedorov, "Efficient realization of quantum primitives for Shor's algorithm using PennyLane library," pp. 1–10, 2022, http://arxiv.org/abs/2201.05426.

[48] L. Wossnig, "Quantum machine learning for classical data," 2021, http://arxiv.org/abs/2105.03684.

[49] Q. Cai, C. Cui, Y. Xiong, W. Wang, Z. Xie, and M. Zhang, "A survey on deep reinforcement learning for data processing and analytics," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, p. 1, 2022.