Brain-inspired agents for Quantum Reinforcement Learning

E. Andrés , Manuel Pegalajar Cuellar * , Gabriel Navarro

Posted Date: 18 March 2024

doi: 10.20944/preprints202403.0843.v1

Keywords: Quantum Reinforcement Learning; Quantum Neural Networks; Quantum Spiking Neural Network; Quantum Long Short-Term Memory; Brain-inspired models.

Article

Brain-Inspired Agents for Quantum Reinforcement Learning

E. Andrés, M. P. Cuéllar * and G. Navarro

Department of Computer Science and Artificial Intelligence, ETSI Informática y de Telecomunicación, Universidad de Granada, C/. Pdta Daniel Saucedo Aranda sn, 18014 Granada, Spain

* Correspondence: manupc@decsai.ugr.es

Abstract: In recent years, advancements in brain science and neuroscience have significantly influenced the field of computer science, particularly in the domain of reinforcement learning (RL). Drawing insights from neurobiology and neuropsychology, researchers have leveraged these findings to develop novel mechanisms for understanding intelligent decision-making processes in the brain. Concurrently, the emergence of quantum computing has opened new frontiers in artificial intelligence, leading to the development of quantum machine learning (QML). This study introduces a novel model that integrates Quantum Spiking Neural Networks (QSNN) and Quantum Long Short-Term Memory (QLSTM) architectures, inspired by the complex workings of the human brain. Specifically designed for reinforcement learning tasks in energy-efficient environments, our approach progresses through two distinct stages mirroring sensory and memory systems. In the initial stage, analogous to the brain's hypothalamus, low-level information is extracted to emulate sensory data processing patterns. Subsequently, resembling the hippocampus, this information is processed at a higher level, capturing and memorizing correlated patterns. We conduct a comparative analysis of our model against existing quantum models such as Quantum Neural Networks and their classical counterparts, elucidating its unique contributions. Through empirical results, we aim to underscore the effectiveness of our approach in optimizing energy consumption in practical scenarios.

Keywords: quantum reinforcement learning; Quantum Neural Networks; Quantum Spiking Neural Network; Quantum Long Short-Term Memory; Brain-inspired models

1. Introduction

Recent progress in artificial intelligence (AI) has spurred the emergence of brain-inspired AI, an interdisciplinary field blending principles from neuroscience, psychology, and computer science to enhance the development of more robust systems [1]. This interdisciplinary fusion has driven innovation, particularly in reinforcement learning (RL), where insights from neurobiology and neuropsychology have revolutionized algorithm development, leading to a surge in research activity [2].

Several works in Brain-inspired Reinforcement Learning (RL) have been presented, inspired by diverse cognitive mechanisms. There is a consensus that the prefrontal cortex (PFC) and basal ganglia (BG) are key structures involved in RL. Previous neurophysiological experiments have revealed that the medial prefrontal cortex (mPFC) contributes to regulating RL parameters such as learning rate and exploration rate. Additionally, representations in the entorhinal and ventromedial prefrontal cortex (vmPFC) play a significant role in generalizing the framework of RL problems. While various computational models of information processing have been developed for the BG, the most prominent one is the actor-critic model of RL, which closely aligns with the neural architecture of the basal ganglia [2–8]. The actor-critic implements two computational modules: the critic, responsible for learning state values and potentially implemented in the ventral striatum (possibly in conjunction with the amygdala and orbitofrontal cortex); and the actor, responsible for learning stimulus-response (S-R) associations and potentially implemented in the dorsal striatum. Both the critic and the actor utilize dopamine-signaled prediction errors to update their estimates [9–11].

Inspired by the hippocampus, hierarchical state-space structures have been employed in grid-world simulations, while meta-learning has been utilized to mimic neurotransmitter dynamics [12,13]. Another significant area of research involves the emulation of theory of mind capabilities, achieved

 doi:10.20944/preprints202403.0843.v1

through Spiking Neural Networks for actor Networks and Artificial Neural Networks for critic networks [14]. Attentional reinforcement learning techniques have also found applications in various domains [15–20].

Quantum machine learning has arisen as an intriguing focal point within the scientific community. Notably, Variational Quantum Circuits (VQC) also know as Quantum Neural Networks (QNN) have exhibited success across various domains, including unsupervised learning [21] and supervised learning [22–24]. Although research on Quantum Reinforcement Learning (QRL) is still in its nascent stages, recent studies have showcased that QNNs can surpass classical models in Reinforcement learning (RL) for energy-efficiency scenarios. They achieve superior cumulative rewards while requiring fewer parameters to be learned [25,26].

Moreover, numerous researchers argue that quantum probability theory receives greater emphasis in quantum cognition compared to its physical counterpart. This is because quantum probability theory, functioning as a generalized probability theory, offers a more robust representation for tasks and internal states. Permitting internal states to exist in an indefinite state prior to an action being taken [27]. Several examples of Quantum brain-inspired works exists, such as the study of Li et al [28] which demonstrated the efficacy of quantum reinforcement learning (QRL) in mimicking human decision-making. Their research involved comparing two QRL and twelve classical RL models using the Iowa Gambling Task with healthy and cigarette-smoking subjects. They contended that human decision-making exhibits quantum-like characteristics, wherein choices can impact the subjective values of alternatives owing to the superposition of quantum states.

Drawing inspiration from a theory of prefrontal cortex and hippocampus function, our work delves into the intricate roles of these brain regions. While the hippocampus is involved in the formation and recall of specific memories, the prefrontal cortex accumulates features of related memories to shape the 'context' of interconnected experiences [29]. Moreover, our study is motivated by the widely accepted memory model proposed by Atkinson and Shifrin [30], which delineates the short-term store (STS) for transient memory storage and the long-term store (LTS) for maintaining memories over extended periods [31,32].

Supporting this view, our model mimics the prefrontal cortex using Quantum Spiking Neural Networks (QSNN) and the hippocampus using Quantum Long Short-Term Memory (QLSTM). The QSNN effectively filters out noisy and infrequent events while strengthening information with stronger space-time correlations. Subsequently, we access QLSTM to retrieve specific information, engage in processing and memorization phases, and transform it from short-term to long-term storage. Given the temporal nature of QLSTM, we preserve both temporal and spatial information throughout the learning process.

In this manuscript, we will undertake a comparative study focusing on energy efficiency. Previous research has delved into this scenario of energy efficiency in buildings, both with classical models (e.g., [33–35]) and quantum models (e.g., [25,26]).

We will evaluate three different architectures, Artificial Neural Networks (ANN), along with quantum architectures including Quantum Neural Networks (QNN) and a novel model comprising Quantum spiking Neural Network (QSNN) and Quantum Long short-term memory (QLSTM). Our research will focus on determining which of these architectures exhibits optimal performance.

2. Background

To comprehensively cover the topics addressed in this manuscript, this section presents a concise introduction to Spiking Neural Networks (SNNs), Long Short-Term Memory, and Quantum Neural Networks (QNNs), and Deep Reinforcement Learning (DRL), which constitute the principal subjects addressed in this research.

At the outset, in Section 2.1, we delve into the essentials of Spiking Neural Networks. This section provides an introduction to the fundamental structure and functionality of SNNs, along with a discussion of Hebbian Theory.

 doi:10.20944/preprints202403.0843.v1

Subsequently, in Section 2.2, we explore the architecture and operation of Long Short-Term Memory networks followed by the Section 2.3, which introduces reinforcement learning principles and their integration with (deep) artificial neural networks.

Lastly, in Section 2.4, we introduce Quantum Neural Networks, elucidating their key concepts and principles.

2.1. Spiking Neural Networks

SNNs draw inspiration from the neural communication patterns observed in the brain, resembling the encoding and retention processes of working memory or short-term memory in the prefrontal cortex, along with the application of the Hebbian plasticity principle.

The Hebbian theory is a neuroscientific concept that describes a fundamental mechanism of synaptic plasticity. According to this theory, the strength of a synaptic connection increases when neurons on both sides of the synapse are repeatedly activated simultaneously. Introduced by Donald Hebb in 1949, it is known by various names, including Hebb's rule, Hebbian learning postulate, or Cell Assembly Theory. The theory suggests that the persistence of repetitive activity or a signal tends to induce long-lasting cellular changes that enhance synaptic stability. When two cells or systems of cells are consistently active at the same time, they tend to become associated, facilitating each other's activity. This association leads to the development of synaptic terminals on the axon of the first cell in contact with the soma of the second cell, as depicted in Figure 2 [36].

Despite DNNs being historically inspired by the brain, there exist fundamental differences in their structure, neural processing, and learning mechanism when compared to biological brains. One of the most significant distinctions lies in how information is transmitted between their units. This observation has lead to the emergence of spiking neural networks (SNNs). In the brain, neurons communicate by transmitting sequences of potentials or spike trains to downstream neurons. These individual spikes are temporally sparse, imbuing each spike with significant information content. Consequently, SNNs convey information through spike timing, encompassing both latencies and spike rates. In a biological neuron, a spike occurs when the cumulative changes in membrane potential, induced by pre-synaptic stimuli, surpass a threshold. The rate of spike generation and the temporal pattern of spike trains carry information about external stimuli and ongoing computations.

ANNs communicate using continuous-valued activations and, for that reason, SNNs are more efficient. This efficiency stems from the temporal sparsity of spike events, as elaborated bellow. Additionally, SNNs posses the advantage of being inherently attuned to the temporal dynamics of information transmission observed in biological neural systems. The precise timing of every spike is highly reliable across various brain regions, indicating a pivotal role in neural encoding, particularly in sensing information-processing areas and neural-motor regions [37,38].

SNNs find utility across various domains of pattern recognition, including visual processing, speech recognition, and medical diagnosis. Deep SNNs represent promising avenues for exploring neural computation and diverse coding strategies within the brain. Although the training of deep spiking neural networks is still in its nascent stages an important open question revolves around their training, such as enabling online learning while mitigating catastrophic forgetting [39].

Spiking neurons operate by summing weighted inputs. Instead of passing this result through a sigmoid or ReLU non-linearity, the weighted sum contributes to the membrane potential $U(t)$ of the neuron. If the neuron becomes sufficiently excited by this weighted sum and the membrane potential reaches a threshold θ, it will emit a spike to its downstream connections. However, most neuronal inputs consist of brief bursts of electrical activity known as spikes. It is highly unlikely for all input spikes to arrive at the neuron body simultaneously. This suggests the existence of temporal dynamics that maintain the membrane potential over time. Figure 1.

 doi:10.20944/preprints202403.0843.v1

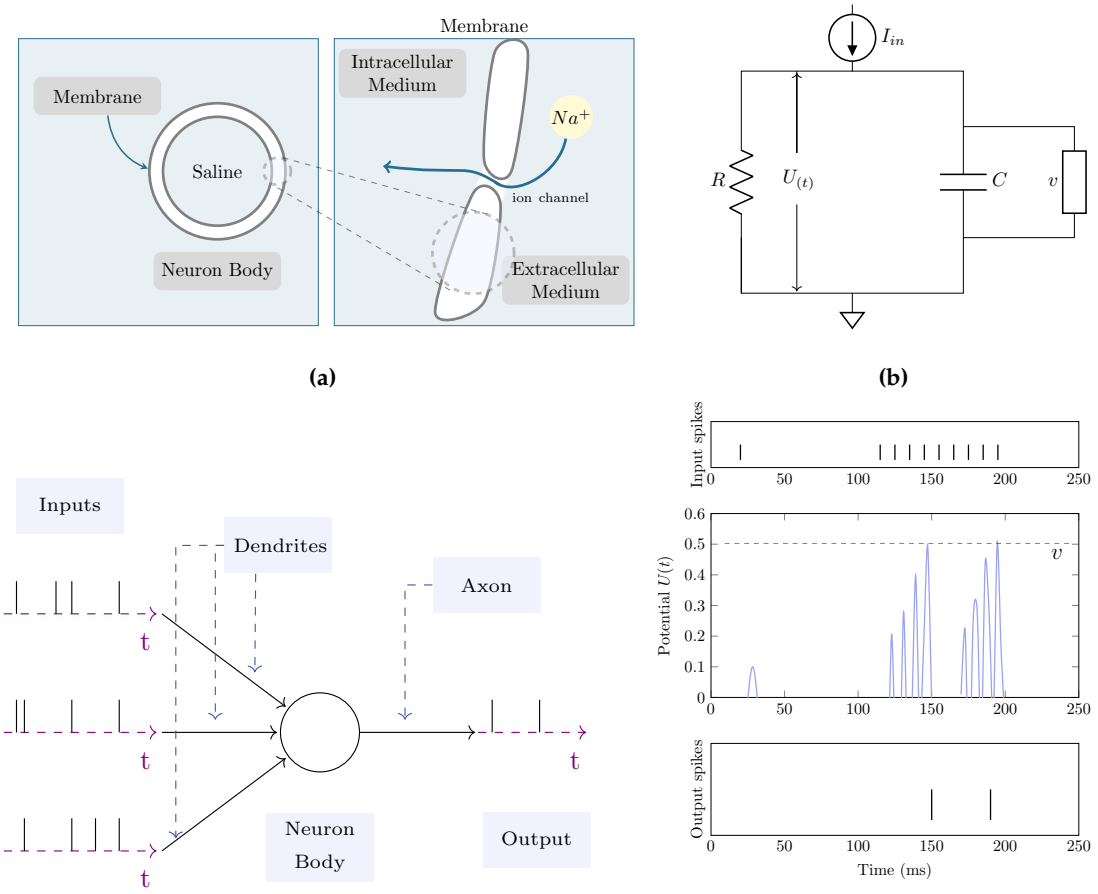


Figure 1. The Leaky Integrate-and-Fire Neuron Model [37] involves an insulating lipid bilayer membrane that separates the interior and exterior environments. Gated ion channels enable the diffusion of charge carriers, such as Na+, through the membrane.1a. This neuron function is often modeled using an RC circuit. When the membrane potential exceeds the threshold, a spike is generated 1b. Input spikes are transmitted to the neuron body via dendritic branches. Accumulation of sufficient excitation triggers spike emission at the output 1c. A simulation depicting the membrane potential $U(t)$ reaching a threshold of $\theta = 0.5V$, leading to the generation of output spikes 1d

Louis Lapicque [40] observed that a spiking neuron can be analogously likened to a low-pass filter circuit, comprising a resistor (R) and a capacitor (C). This concept is referred to as the leaky integrate-and-fire neuron model. Even in the present, that idea remains valid. Physiologically, the neuron's capacitance arises via the insulating lipid bilayer constituting its membrane, while the resistance is a consequence of gated ion channels regulating the flow of charged particles across the membrane (see Figure 1b).

The characteristics of this passive membrane can be elucidated through an RC circuit, in accordance with Ohm's Law. This law asserts that the potential across the membrane, measured between the input and output of the neuron, is proportional to the current passing through the conductor [37].

$$I_{in}(t) = \frac{U(t)}{R} \tag{1}$$

The behavior of the passive membrane, which is simulated using an RC circuit, can be depicted as follows:

$$\tau \frac{dU(t)}{dt} = -U(t) + I_{in}(t)R \tag{2}$$

 doi:10.20944/preprints202403.0843.v1

where $\tau = RC$, representing the time constant of the circuit. Following the Euler method, $dU(t)/dt$ without $\Delta t \to 0$:

$$\tau \frac{U(t+\Delta t) - U(t)}{\Delta t} = -U(t) + I_{in}(t)R \tag{3}$$

Extracting the membrane potential in the subsequent step:

$$U(t+\Delta t) = \left(1 - \frac{\Delta t}{\tau}\right) U(t) + \frac{\Delta t}{\tau} I_{in}(t)R \tag{4}$$

To isolate the dynamics of the leaky membrane potential, let's assume there is no input current $I_{in} = 0$:

$$U(t+\Delta t) = \left(1 - \frac{\Delta t}{\tau}\right) U(t) \tag{5}$$

The parameter $\beta = U(t+\Delta t)/U(t)$ represents the decay rate of the membrane potential, also referred to as the inverse of the time constant. Based on the preceding equation, it follows that $\beta = 1 - \Delta t/\tau$.

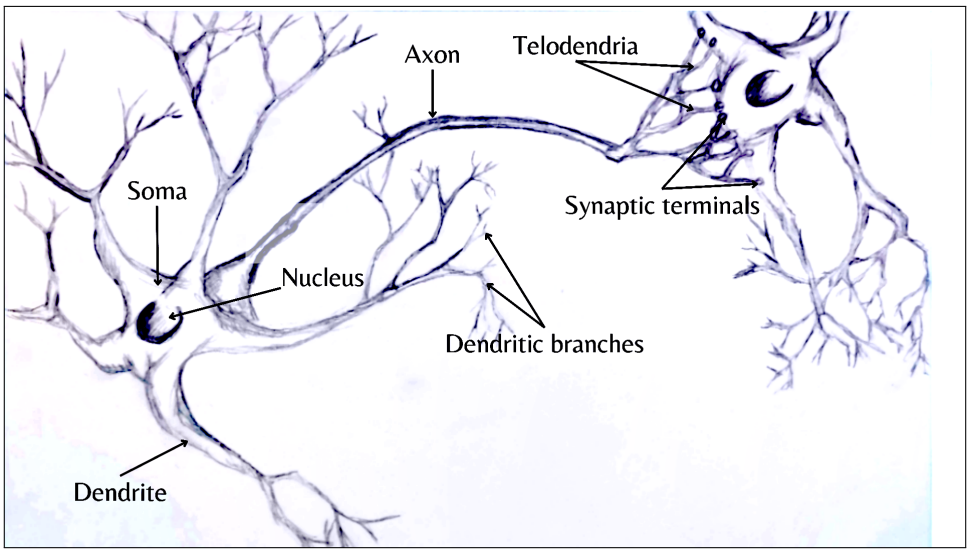


Figure 2. The standard structure of a neuron includes a cell body, also known as soma, housing the nucleus and other organelles; dendrites, which are slender, branched extensions that receive synaptic input from neighboring neurons; an axon; and synaptic terminals.

Let's assume that time t is discretised into consecutive time steps, such that $\Delta t = 1$. To further minimize the number of hyperparameters, let's assume $R = 1\,\Omega$. Then

$$\beta = 1 - \frac{1}{\tau} \Rightarrow U(t+1) = \beta U(t) + (1-\beta) I_{in}(t+1) \tag{6}$$

When dealing with a constant current input, the solution to this can be obtained as

$$U(t) = I_{in}(t)R + [U_0 - I_{in}(t)R]e^{-t/\tau} \tag{7}$$

This demonstrates the exponential relaxation of $U(t)$ towards a steady-state value following current injection, with U_0 representing the initial membrane potential at $t = 0$

$$U(t) = U_0 e^{-t/\tau} \tag{8}$$

If we compute Eq.(8) at discrete intervals of t, $(t+\Delta t)$, $(t+2\Delta t)$..., then we can determine the ratio of the membrane potential between two consecutive steps using:

 doi:10.20944/preprints202403.0843.v1

$$\beta = \frac{U_0 e^{-(t+\Delta t)/\tau}}{U_0 e^{-t/\tau}} = \frac{U_0 e^{-(t+2\Delta t)/\tau}}{U_0 e^{-(t+\Delta t)/\tau}} = \ldots \Rightarrow \beta = e^{-\Delta t/\tau} \tag{9}$$

This equation for β offers greater precision compared to $\beta = (1 - \Delta t/\tau)$, which is accurate only under the condition that $\Delta t \ll \tau$.

Another non-physiological assumption is introduced, wherein the effect of $(1 - \beta)$ is assimilated into a learnable weight W (in deep learning, the weight assigned to an input is typically a parameter that can be learned):

$$WX(t) = I_{in}(t) \tag{10}$$

$X(t)$ represents an input voltage, spike, or unweighted current, which is scaled by the synaptic conductance W to produce a current injection to the neuron. This generates the following outcome:

$$U(t+1) = \beta U(t) + WX(t+1) \tag{11}$$

by decoupling the effects of W and β, simplicity is prioritized over biological precision. Lastly, a reset function is added, which is triggered each time an output spike occurs:

$$U(t) = \underbrace{\beta U(t-1)}_{\text{decay}} + \underbrace{WX(t)}_{\text{input}} - \underbrace{S_{out}(t-1)\theta}_{\text{reset}} \tag{12}$$

where $S_{out}(t) \in \{0, 1\}$ is the output spike, 1 in case of activation and 0 in otherwise. In the first scenario, the reset term subtracts the threshold θ from the membrane potential, whereas in the second scenario, the reset term has no impact.

A spike occurs when the membrane potential exceeds the threshold:

$$S_{out}(t) = \begin{cases} 1, & \text{if } U(t) > \theta \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

Various techniques exist for training SNN's [37], with one of the more commonly utilized approaches being backpropagation using spikes, also known as backpropagation through time (BPTT). Starting from the final output of the network and moving backwards, the gradient propagates from the loss to all preceding layers. The objective is to train the network utilizing the gradient of the loss function concerning the weights, thus updating the weights to minimize the loss. The backpropagation algorithm accomplishes this by utilizing the chain rule:

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial S} \underbrace{\frac{\partial S}{\partial U}}_{\{0,\infty\}} \frac{\partial U}{\partial I} \frac{\partial I}{\partial W} \tag{14}$$

Nevertheless, the derivative of the Heaviside step function from (13) is the Dirac Delta function, which equates to 0 everywhere except at the threshold $U_{\text{thr}} = \theta$, where it tends to infinity. Consequently, the gradient is almost always nullified to zero (or saturated if U precisely sits at the threshold), rendering learning ineffective. This phenomenon is referred to as the dead neuron problem. The common approach to address the dead neuron problem involves preserving the Heaviside function during the forward pass, but substituting it with a continuous function, $tilde(S)$, during the backward pass. The derivative of this continuous function is then employed as a substitute for the Heaviside function's derivative, denoted as $\partial S/\partial U \leftarrow \partial \tilde{S}/\partial U$, and is termed the surrogate gradient. In this manuscript, we utilize snntorch library, which defaults to using the arctangent function [37].

The structure of QSNNs follows the hybrid-architecture formed by Classical linear layers and VQC for QLIF implementation (Quantum Leaky integrate and fired Neuron) and trained using gradient

 doi:10.20944/preprints202403.0843.v1

descent method. The Figure 3 shown the general pipeline for this model for a classification task and the detailed architecture is defined in section 3.

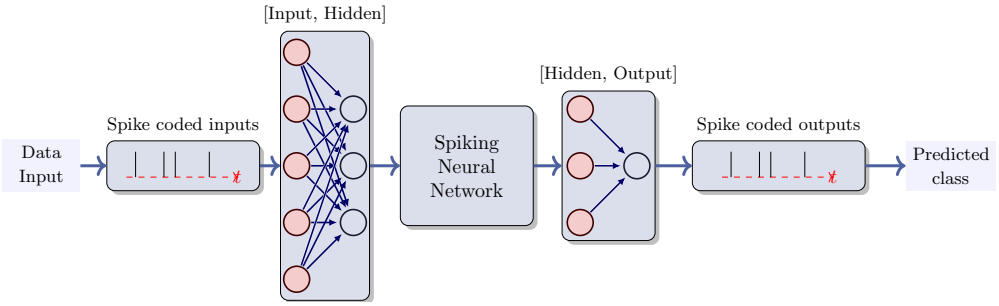


Figure 3. SNN pipeline. The input data for an SNN may undergo transformation into a firing rate or alternative encodings to produce spikes. Subsequently, the model is trained to accurately predict the correct class by employing encoding strategies, such as the highest firing rate or firing first, amongst other options

Previous works have been made inspiring in brain functionality emulating for classification tasks such as MINST dataset classification using SNN and Hyperdimensional computing [41] or in the decoding and understanding of muscle activity and kinematics from electroencephalography signals [42], utilizing Hyperdimensional Computing (HDC) and SNN for the MNIST classification problem [41]. Other works have explored the application of Reinforcement Learning for navigation in dynamic and unfamiliar environments, supporting neuroscience-based theories that consider grid cells as crucial for vector-based navigation [43].

2.2. Long Short-Term Memory

Long Short-Term Memory (LSTM) networks belong to a class of recurrent neural networks that have the ability to learn order dependence in sequence prediction problems. These networks are crafted to address the challenges encountered in training RNNs (Recurrent Neural Networks). Retro-propagated gradients often exhibit substantial growth or decay over time due to their dependency not only on the current error but also on past errors. The accumulation of these errors impedes the memorization of long-term dependencies. Consequently, Long Short-Term Memory neural networks (LSTM) are employed to tackle these issues. LSTMs incorporate a series of mechanisms to determine which information should be retained and which should be discarded [44]. Furthermore, standard RNNs have a limited ability to access contextual information in practice. The impact of a specific input on the hidden layer and, consequently, on the network output, diminishes or amplifies exponentially as it circulates through the recurrent connections of the network. This phenomenon is known as the vanishing gradient problem, which represents the second challenge to overcome using LSTM [45,46].

The behavior of this model is essential in complex problem domains such as machine translation, speech recognition and time-series analysis, among others.

These networks consist of LSTM modules, which are a specialized type of recurrent neural network introduced in 1997 by Hochreiter and Schmidhuber [47]. They consist of three internal gates, known as input, forget and output gates, detailed in Figure 4.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 March 2024 doi:10.20944/preprints202403.0843.v1

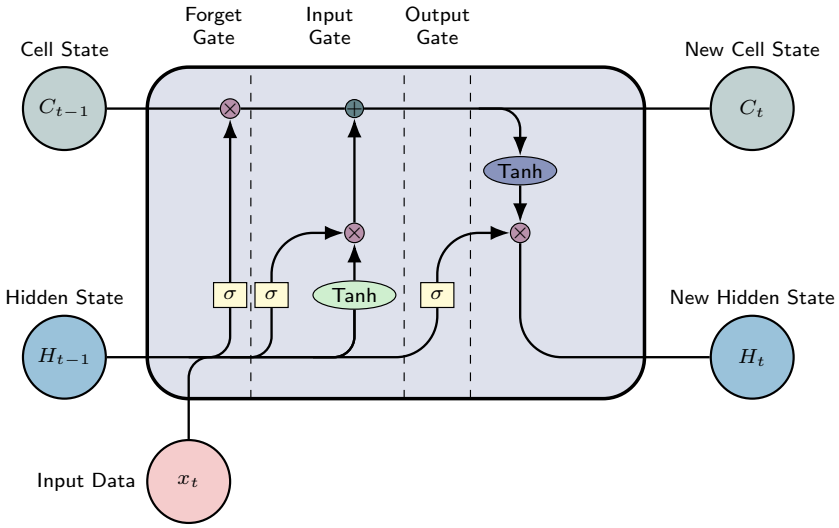


Figure 4. LSTM Cell Architecture: Featuring three essential gates (forget, input and output gates). The *tanh* and σ blocks symbolize the hyperbolic tangent and sigmoid activation functions, correspondingly. x_t represents the input at time t, h_t denotes the hidden state, and c_t signifies the cell state. The symbols $\otimes$ and $\oplus$ denote element-wise multiplication and addition, respectively.

These gates are filters and each of them have its own neural-network. At a given moment, the output of an LSTM relies on three factors:

- Cell state: The network's current long-term memory
- Hidden state: The output from the preceding time step
- The input data in the present time step

The internal gates mentioned above can be described as follows [48]:

- **Forget Gate**: This gate decides which information from the cell state is important, considering both the previous hidden state and the new input data. The neural network that implements this gate is build to produce an output closer to 0 when the input data is considered unimportant, and closer to 1 otherwise. To achieve this, we employ the sigmoid activation function. The output values from this gate are then passed upwards and undergo pointwise multiplication with the previous cell state. This pointwise multiplication implies that components of the cell state identified as insignificant by the forget gate network will be multiplied by a value approaching 0, resulting in reduced influence on subsequent steps.

  To summarize, the forget gate determines which portions of the long-term memory should be disregarded (given less weight) based on the previous hidden state and the new input data.
- **Input gate**: Determines the integration of new information into the network's long-term memory (cell state), considering the prior hidden state and incoming data. The same inputs are utilized, but now with the introduction of a hyperbolic tangent as the activation function. This hyperbolic tangent has learned to blend the previous hidden state with the incoming data, resulting in a newly updated memory vector. Essentially, this vector encapsulates information from the new input data within the context of the previous hidden state. It informs us about the extent to which each component of the network's long-term memory (cell state) should be updated based on the new information.

  It should be noted that the utilization of the hyperbolic tangent function in this context is deliberate, owing to its output range confined to [-1,1]. The inclusion of negative values is imperative for this methodology, as it facilitates the attenuation of the impact associated with specific components.

 doi:10.20944/preprints202403.0843.v1

- **Output gate**: the objective of this gate is to decide the new hidden state by incorporating the newly updated cell state, the prior hidden state, and the new input data. This hidden state has to contain the necessary information while avoiding the inclusion of all learned data. To achieve this, we employ the sigmoid function.

This architecture is replicated for each time step considered in the prediction. The ultimate layer of this model is a linear layer responsible for converting the hidden state into the ultimate prediction.

The quantum counterpart of this neural network is constructed with a VQC model for each gate as Figure 9 shows.

Finally, we summarise the Lstm implementation steps as follows:

- The initial step involves determining which information to discard or preserve at the given moment in time. This process is facilitated by the utilisation of the sigmoid function. It examines both the preceding state h_{t-1} and the present input x_t, computing the function accordingly:

$$f_t = \sigma(W_f \cdot v_t + bf) \tag{15}$$

Where $v_t = [x_t, h_{t-1}]$ and w_f and b_f are weights and biases.

- In this step, the memory cell content undergoes an update by choosing new information for storage within the cell state. The subsequent layer, known as the input gate, comprises two components: the sigmoid function and the hyperbolic tangent (tanh). The sigmoid layer decides which values to update; a value of 1, indicates no change, while a value of 0 results in exclusion. Subsequently, a tanh layer generates a vector of new candidate values, assigning weights to each value based on its significance within the range of -1 to 1. These two components are then combined to update the state:

$$\begin{aligned} i_t &= \sigma(W_i \cdot v_t + b_i) \\ \widetilde{C} &= tanh(W_c \cdot v_t + b_c) \end{aligned} \tag{16}$$

- The third step consists of updating the previous cell state, C_{t-1} with the new cell state, C_t, through two operations: forgetting irrelevant information by scaling the previous state by f_t and incorporating new information from the candidate $\widetilde{C}_t$:

$$C_t = f_t \cdot c_{t-1} + i_t \cdot \widetilde{C}_t \tag{17}$$

- Ultimately, the output is calculated through a two-step process. Initially, a sigmoid layer is employed to determine which aspects of the cell state are pertinent for transmission to the output.

$$o_t = \sigma(W_o \cdot v_t + b_0) \tag{18}$$

Subsequently, the cell state undergoes processing via the tanh layer to normalize values between -1 and 1, followed by multiplication with the output of the sigmoid gate.

$$h_t = tanh(C_t) \cdot o_t \tag{19}$$

2.3. Deep Reinforcement Learning

Reinforcement learning (RL) [49] is a branch of machine learning inspired by behavioral psychology. In RL, an entity known as the agent adjusts its behavior based on the rewards and penalties it receives from interacting with an unkwnown environment. RL serves as the foundational framework for elucidating how autonomous intelligent agents acquire the ability to navigate unfamiliar environments and optimize cumulative rewards through decision-making. Deep Reinforcement Learning (DRL) combines traditional RL algorithms with neural networks. The general schema of DRL is

 doi:10.20944/preprints202403.0843.v1

illustrated in Figure 5: When an agent interacts with an environment, it has no knowledge of the environment´s state except for the observations it receives. At time t, the observation of the environment is denoted as s_t. The agent then selects an action a_t from the set of the available actions and executes in the environment. Subsequently, the environment transitions to a new state and provides the agent with the new observation s_{t+1} and a reward r_t. The reward indicates the quality of the action taken by the agent and is utilized to improve its performance in subsequent interactions.

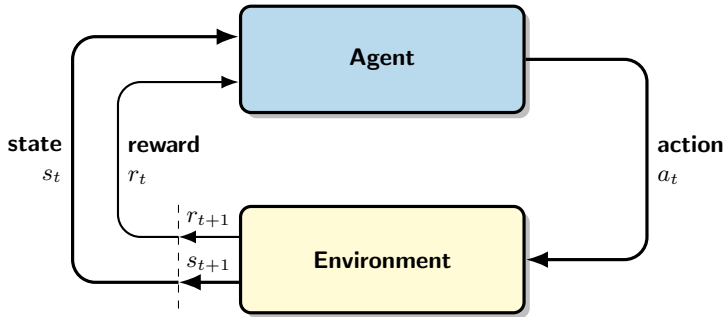


Figure 5. General reinforcement Learning diagram [50]. At time t, the agent perceives state s_t and, based on this state, selects an action a_t.The environment then transitions to a new state s_{t+1} and provides the agent with this new state along with a reward r_{t+1}.

This sequential process is described using a Markov Decision Process (MDP). which consists of the tuple $\langle S, A, P, r \rangle$, where S and A represent the sets of states and actions, respectively. P denotes the probability of state transition, defined as

$$P(s'|s,a) = P[s_{t+1} = s'|s_t = s, a_t = a] \tag{20}$$

indicating the likelihood of transitioning from state s_t at time t to state s_{t+1} at time $t+1$ when action a_t is taken at time t. Additionally, $r(s_t, a_t, s_{t+1})$ represents the reward function associated with executing action a_t in state s_t and transitioning to state s_{t+1}.

The agent's goal is to maximize its cumulative reward through a series of interactions with the environment τ, beginning at time t_0. This cumulative reward, referred to as the return and defined in Equation (21), is influenced by the hyperparameter γ, which determines the relative significance of recent versus past rewards in the learning process. γ is commonly known as the *discount factor*. To maximize $R(\tau)$, the agent must acquire knowledge about the optimal action a to take in a given state s, known as the *policy* $\pi(a|s)$. This policy function charasterizes the agent's behavior within the environment, providing the probability of selecting action a in state s. In RL we consider two key functions: the value of a state–action pair $Q(s,a)$ (as defined in Equation (22), representing the expected return obtained from starting at state s and taking action a), and the value of a state $V(s)$ (as defined in Equation (23), representing the expected return obtained from starting at state s). Additionally, another relevant concept is the advantage of a state–action pair $Adv(s,a)$ (as defined in Equation (24)), which quantifies te benefit of selecting action a in state s compared to the other available actions in the same state s.

$$R(\tau) = \sum_{t=t_0}^{\infty} r_t \gamma^{t-t_0} \tag{21}$$

$$\begin{aligned} Q(s,a) = \mathbb{E}_{\tau\sim\pi}(R(\tau)|s_t = s, a_t = a) = \\ \sum_{s'} p(s'|s,a)(r(s,a,s') + \gamma \sum_{a'} \pi(a'|s')Q(s',a')) \end{aligned} \tag{22}$$

 doi:10.20944/preprints202403.0843.v1

$$V(s) = \mathbb{E}_{\tau\sim\pi}(R(\tau)|s_t = s) = \sum_i R(\tau_i)\pi(a_i|s) \tag{23}$$

$$Adv(s,a) = Q(s,a) - V(s) \tag{24}$$

Deep reinforcement learning aims to use a (deep) artificial neural network to learn the optimal policy $\pi(a|s)$. This policy takes the state s as input and outputs either a chosen action a (in the case of a deterministic policy) or the probability distribution of selection of action a in the state s (in the case of a stochastic policy). In recent years, the literature has emphasized two main families of algorithms: deep Q-networks (DQN) [51] and policy gradient [52]. The former focuses on training an artificial neural network to approximate the function $Q(s,a)$, while the latter directly approximate $\pi(a|s)$. DQN training draws inspiration from the classic Q-learning approach and aims to minimize the loss function described in Equation (26). Here, $\hat{Q}(s,a)$ represents the output value corresponding to action a generated by the neural network when provided with input s. A deep Q-network consists of input neurons equal to the dimension of the state and output neurons equal to the number of actions available in the action set.

On the contrary, actor–critic policy gradient models necessitate the utilization of at least two neural networks types for training: one (the actor) shares a structure resembling that of a DQN, but its a-th output aims to yield $\pi(a|s)$. The other type endeavors to approximate $V(s)$, mirroring the actor in terms of number of inputs and featuring a single output value.

Various approaches have been developed enhance DQN training and policy gradient methods. For additional information, we direct readers to the following references: [51,52]. In this manuscript, we employ the Advantage Actor–Critic (A2C) training algorithm [52], which is characterized by its designed loss function outlined in Equation (25).

$$LogLoss = -\sum_t Adv(s_t,a_t) log\pi(a_t|s_t) \tag{25}$$

$$MSE = \sum_t \left(r(s_t,a_t,s_{t+1}) + \gamma max_{a_{t+1}}\left[\hat{Q}(s_{t+1},a_{t+1})\right] - \hat{Q}(s_t,a_t)\right)^2 \tag{26}$$

2.4. Quantum Neural Networks

Quantum Neural Networks (QNNs) play a pivotal role in Quantum Machine Learning (QML) [53,54], utilizing Variational Quantum Circuits (VQCs). These VQCs are hybrid algorithms, combining quantum circuits with learnable parameters optimized utilising classical techniques. They have the ability to approximate continuous functions [53,55], enabling tasks such as optimization, approximation, and classification.

Figure 6 depicts the overall structure of a VQC. This hybrid approach involves the following stages [56]:

The VQC workflow comprises several steps:

1. **Pre-processing (CPU)**: Initial classical data preprocessing, which includes normalization and scaling.
2. **Quantum Embedding (QPU)**: Encoding classical data into quantum states through parameterized quantum gates. Various encoding techniques exist, such as angle encoding, also known as tensor product encoding, and amplitude encoding, among others [57].
3. **Variational Layer (QPU)**: This layer embodies the functionality of Quantum Neural Networks through the utilization of rotations and entanglement gates with trainable parameters, which are optimized using classical algorithms.
4. **Measurement Process (QPU/CPU)**: Measuring the quantum state and decoding it to derive the expected output. The selection of observables employed in this process is critical for achieving optimal performance.

 doi:10.20944/preprints202403.0843.v1

5. **Post-processing (CPU)**: Transformation of QPU outputs before feeding them back to the user and integrating them into the cost function during the learning phase.
6. **Learning (CPU)**: Computation of the cost function and optimization of ansatz parameters using classic optimization algorithms, such as Adam or SGD. Gradient-free methods, such as SPSA or COLYBA, are also capable of estimating parameter updates.

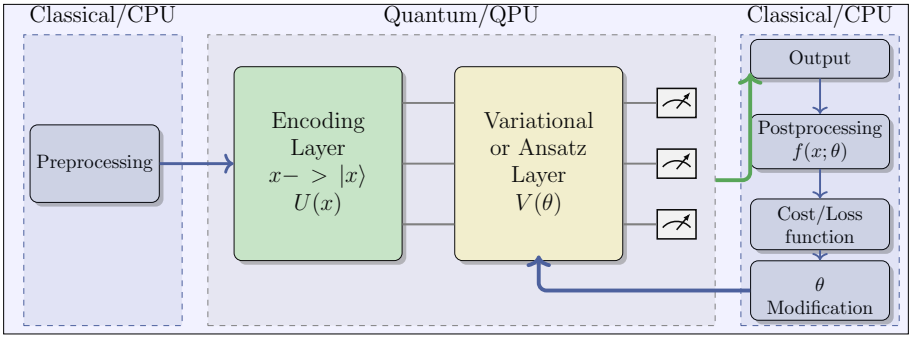


Figure 6. General VQC Schema. The dashed gray line delineates the process carried out within a Quantum Processing Unit (QPU), while the dashed blue line illustrates the processes executed in a CPU.

3. Methodology

In this manuscript, we conducted an extensive benchmarking analysis, comparing various quantum-classical networks with their classical counterparts trained using gradient-descent. These include Quantum Neural Network (QNN), Quantum Spiking Neural Network (QSNN), Quantum Long Short-Term Memory (QLSTM), and a novel model designed to closely emulate brain functionality by integrating QSNN and QLSTM, establishing a scalable and robust cognitive learning system. To achieve this, the QSNN component captures information at a low-level perspective, mirroring the role of the hypothalamus. Subsequently, the QLSTM processes this information at a higher level, identifying and memorizing correlated patterns while reinforcing long-term memorization, emulating hippocampus and mitigating the risk of catastrophic forgetting [37]. Catastrophic forgetting, which occurs when new information causes the network to lose previously learned knowledge [58], poses a significant challenge in real-time systems where the batch-size is small, often 1. Various strategies have been proposed to address catastrophic forgetting in continual learning. These include employing higher-dimensional synapses [59], using ensembles of networks [60], implementing pseudo-replay techniques [61], and penalizing weights that exhibit excessively rapid changes [62]. In summary, this innovative approach combines the advantages of QSNN for new learning with QLSTM's ability to retain and build upon previous knowledge.

Considering the diminishing learning rate as network parameters approach optimal values, wherein future data has less influence than past data, our model proactively aims to prevent catastrophic forgetting by preserving previously acquired knowledge, adding a QLSTM model to the existing and trained QSNN. This proactive approach ensures a more comprehensive and stable learning system that balances the integration of new information with the retention of valuable past knowledge.

The initial model presented is a quantum neural network leveraging amplitude encoding to minimize qubits usage. This strategy is chosen because it necessitates only $log_2 n$ for n features. Figure 7 illustrates this architecture.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 March 2024 doi:10.20944/preprints202403.0843.v1

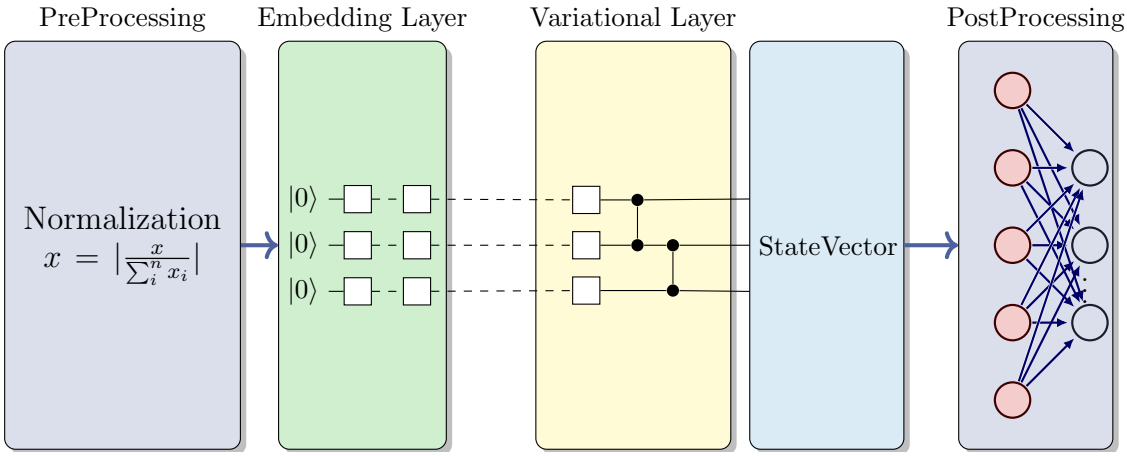


Figure 7. QNN architecture. This architecture involves several steps. First, classical data is normalized as a preprocessing step, which is necessary for the subsequent encoding strategy. Then, the amplitud encoding algorithm is applied, using only $log_2 N$ qubits, where N corresponds to the number of independent variables or the space dimension in the case of RL. This process generates the corresponding quantum state, which is then fed into the ansatz circuit. Finally, the resulting state-vector of this quantum system undergoes post-processing through a classical linear layer. This layer transforms the dimension obtained with 2^n, where n is the number of qubits, into the expected output dimension.

The subsequent model is a Quantum Spiking Neural Network (QSNN), implemented using the Python package snnTorch [63]. The architecture of the QSNN is showed in Figure 10. We enhanced the snnTorch library by incorporating a quantum version of a Leaky Integrate-and-Fire (LIF) neuron. In this quantum version, a VQC is employed to initialize the membrane potential, replacing the conventional tensor approach. The VQC consists of an encoding circuit that utilizes amplitude embedding to minimize qubit usage, and an Ansatz composed of rotation gates along each axis and a controlled-Z gate.

The third model is a Quantum Long Short-Term memory (QLSTM), that utilizes VQCs for its forget, input, update, and output gates, the Quantum Lstm cell is detailed in Figure 8 and the entire structure in Figure 9. The encoding circuit employs Angle encoding after a Classical linear layer to transform concat size (input and hidden dimension) to number of qubits. Additionally, the ansatz circuits incorporate Basic Entangling Layers, which uses Rx and Cx gates.

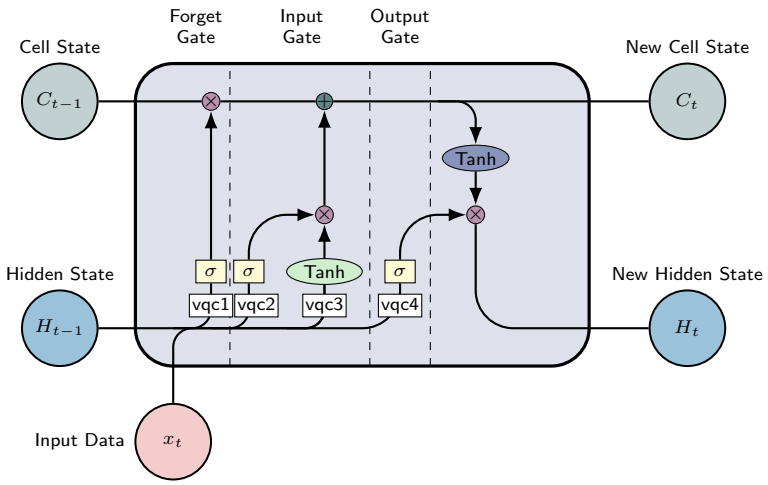


Figure 8. QLSTM Cell. Each VQC box follows the structure outlined in Figure 6. The σ and *tanh* blocks denote the sigmoid and the hyperbolic tanget activation functions, respectively. x_t represents the input at time t, h_t denotes the hidden state and c_t signifies the cell state. Symbols $\otimes$ and $\oplus$ signify element-wise multiplication and addition, respectively.

 doi:10.20944/preprints202403.0843.v1

The last Quantum model is an innovative architecture that combines a Quantum Spiking Neural Network (QSNN) and Quantum Long Short-Term Memory (QLSTM), with the QSNN and QLSTM architecture shown in Figures 10 and 9 respectively. It undergoes co-training, updating gradients simultaneously through a multi-optimizer with distinct learning rates for each network's parameters. The training process of this model is illustrated in Figure 12

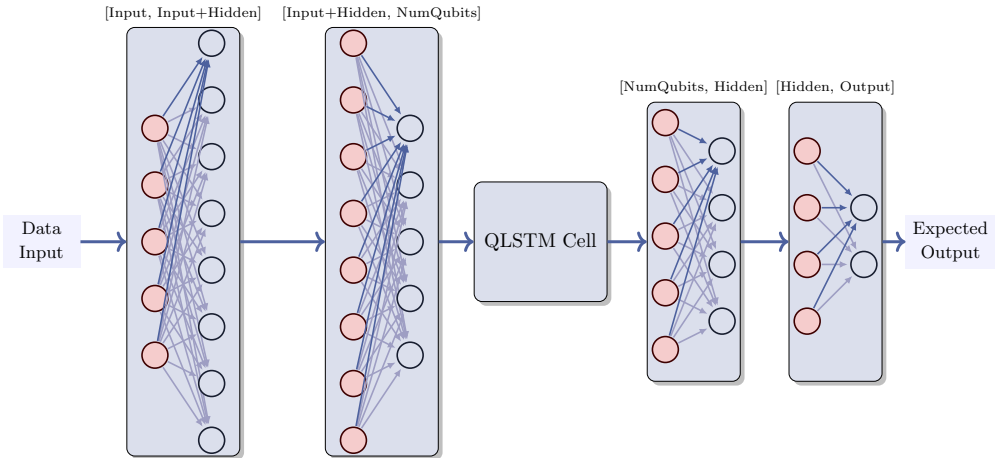


Figure 9. QLSTM Architecture. The input data passes trough an initial classical layer, which receives the inputs data and produces the concatenated size formed by the input dimension and hidden size. This output then passes trough a second classical layer, which outputs the same size as the number of qubits expected by the Quantum Qlstm cell, whose architecture is detailed in Figure 6. Subsenquently, this output is received by another classical layer that transform it into output of the hidden size. Finally, this output is futher transformed into the expected output.

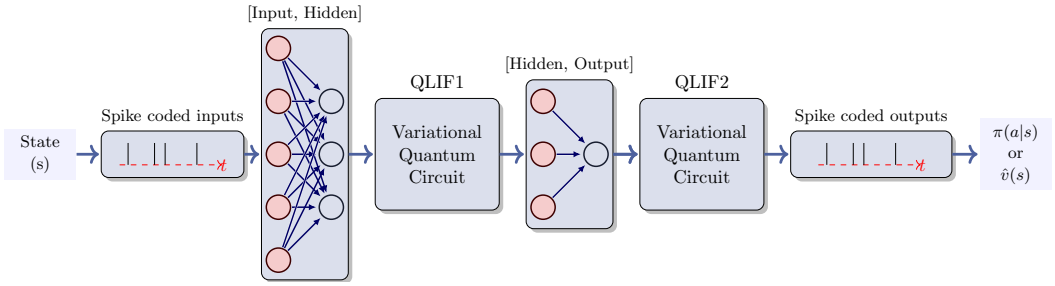


Figure 10. QSNN Architecture. This architecture involves an encoding step where classical data is translated into spikes. The network comprises two LIFs orchestrated by VQC and is trained using gradient descent. Various encoding strategies can be utilized, including the utilization of the highest firing rate or firing first, among others.

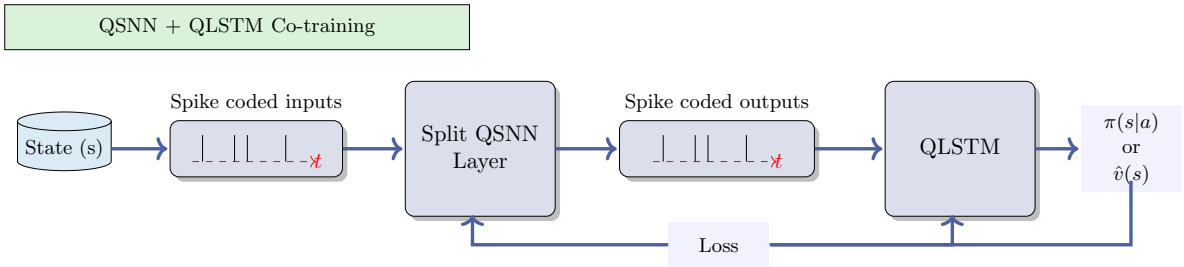


Figure 11. QSNN-QLSTM co-training.

Figure 12. Training Process of QSNN-QLSTM. An instance of QSNN (see Figure 10) is combined with a QLSTM instance (refer to Figure 9) for joint training. The loss computed from the output of QLSTM is utilized for the concurrent update of QSNN and QLSTM.

 doi:10.20944/preprints202403.0843.v1

4. Experimentation

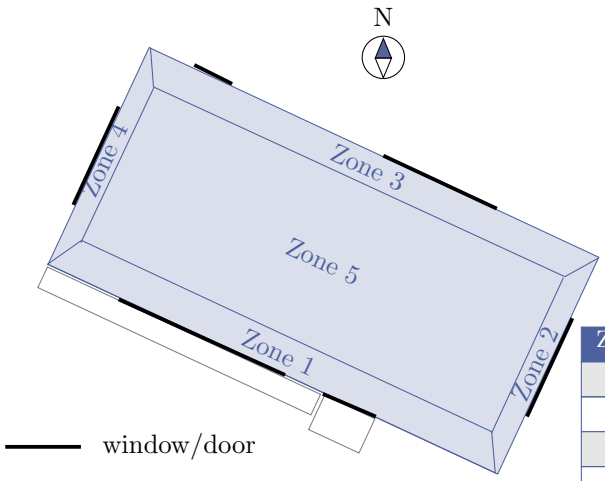
In the literature, various scenarios related to energy efficiency and management are explored. Simulation tools like Energy Plus [64] or Modelica [65] are crucial for training and validating models before their deployment in real-world energy systems. RL environments are build upon such simulation software; for instance, Gym-EPlus [66], Sinergym [67] or EnerGym [68] are utilized for simulating energy consumption and control in buildings. The environment utilized in our experimentation is accessible within *Sinergym* framework. It includes reference problems designed specifically to improve energy efficiency and HVAC control in buildings and facilities.

In this section, we conduct experiments to evaluate various models described in the previous section in a scenario related to energy efficiency for QRL. Our objective is to contrast the performance of each proposed architecture trained with the same advanced actor-critic algorithm and determine whether the novel model composed of QSNN and QLSTM achieves superior results. We performed 10 individual executions, each initiated with distinct random seeds to facilitate reproducibility.

All implemented models make use of the PyTorch, PennyLane, SNNtorch and Sinergym libraries.

4.1. Problem Statement

The focus of this study is the building environment named *Eplus-5zone-hot-discrete-v1*, specifically targeting the scenario *5ZoneAutoDx* [67]. Situated in Arizona, USA characterized by subtropical weather and desert heat, the building comprises a single-floor rectangular structure measuring 100 feet in length. It encompasses five zones, including four exterior and one interior, typically occupied by office workers. The building is oriented 30 degrees east of north (as depicted in Figure 13). With an overall height of 10 feet, all four facades feature windows, constructed with single and double panel of 3mm and 6mm glass, along with argon or air gap of 6mm or 13mm, resulting in a window-to-wall ratio of approximately 0.29. Glass doors adorn the south and north facades, while the walls consist of wooden shingles over plywood, R11 insulation, and Gypboard. Additionally, the south wall and door feature overhangs. The roof is composed of a gravel built-up structure with R-3 mineral board insulation and plywood sheathing. The total floor area spans 463.6 m^2 (5000 $feet^2$).



Zone	Area (m^2)	Hight (m^2)
1	100	2.4
2	43	2.4
3	100	2.4
4	43	2.4
5	186	2.4

Figure 13. Zone building plan.

A state s within this environment encapsulates a series of past observations concerning the building (such as room temperatures or outdoor air temperature). The aim is to optimize a set of KPIs (Key Performance Indicators) related to energy consumption and overall comfort. The specific features are outlined below:

 doi:10.20944/preprints202403.0843.v1

- **State space:** The state space encompasses 17 attributes; with 14 detailed in Table 1, while the remaining 3 are reserved in case new customized features need to be added.
- **Action space:** The action space comprises a collection of 10 discrete actions as outlined in Table 2. The temperature bounds for heating and cooling are [12, 23.5] and [21.5, 40] respectively.
- **Reward function:** The reward function is formulated as multi-objective, where both energy consumption and thermal discomfort are normalized and added together with different weights. The reward value is consistently non-positive, signifying that optimal behavior yields a cumulative reward of 0. Notice also that there are two temperature comfort ranges defined, one for the summer period and other for the winter period. The weights of each term in the reward allow to adjust the importance of each aspect when environments are evaluated. Finally, the reward function is customizable and can be integrated into the environment

$$r_t = -\omega\lambda_P P_t - (1-\omega)\lambda_T(|T_t - T_{up}| + |T_t - R_{low}|) \tag{27}$$

Where P_t denotes power consumption; T_t is the current indoor temperature; T_{up} and T_{low} are the imposed comfort range limits (penalty is 0 if T_t is within this range); ω represents the weight assigned to power consumption (and consequently, $1-\omega$, the comfort weight), and λ_P and λ_T are scaling constants for consumption and comfort, respectively [67].

Table 1. The observation variables consist of 14 variables, with an additional 3 empty variables reserved for specific problem requirements if necessary.

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-Air-Relative-Humidity	%
Zone-People-Occupant-Count	count
Environmental-Impact-Total-CO2-Emissions-Carbon-Equivalent-Mass	Kg
Facility-Total-HVAC-Electricity-Demand-Rate	W
Total-Electricity-HVAC	W

Table 2. Action variables

Name	Heating Target Temperature	Cooling Target Temperature
0	13	37
1	14	34
2	15	32
3	16	30
4	17	30
5	18	30
6	19	27
7	20	26
8	21	25
9	21	24

In this experiment, a classic multilayer perceptron (MLP) neural network agent was developed, featuring 17 inputs (environment state dimension) and 10 outputs (environment action dimension) for the actor, along with 1 output for the critic. This agent underwent training within the *Eplus-5zone-hot-discrete-v1* environment using the Advantage Actor–Critic (A2C) algorithm [52]. Additionally,

 doi:10.20944/preprints202403.0843.v1

we trained a classic spiking neural network (SNN) and a classic long short-term memory (LSTM) both described in Section 3. Subsequently, four different quantum agents were trained using the methodologies described in Section 3, with each model configured according to the specifications detailed in the aforementioned section. Importantly, the environment settings and A2C algorithm parameters remained consistent for both classic and quantum agents to ensure an equitable comparison of performance.

4.2. Experimental Settings

We conducted seven types of experiments. Initially, we constructed the agent's policy using a classic feedforward multilayer perceptron (MLP). This was followed by the implementation of a classical Spiking Neural Network in the subsequent experiment, and then the utilization of a classical Long Short-Term Memory (LSTM) network in another. In the fourth experiment, we utilized a variational quantum circuit for the agent's policy, while in the fifth, we employed a Quantum Spiking Neural Network (QSNN). The sixth experiment introduced a Quantum Long-Short Term Memory (QLSTM). Finally, we developed a novel model by combining QSNN and QLSTM. We employed the advanced actor-critic RL method to train the agent for these experiments. To this end, we concurrently executed five environments, each with a maximum of 15 steps. The algorithm was set to terminate after completing 100 episodes, serving as the stopping criterion. Furthermore, in the final episode, the agent's performance was evaluated using a deterministic policy that selected the action with the highest probability, thereby assessing the agent's ability to interact effectively with the environment. To validate the experiments, we conducted 10 runs with distinct initial random seeds, recording the total accumulated reward in each execution to summarize the average, best, and worst total accumulated reward obtained by the models. Tables 3 and 4 provide an overview of the configuration details of both classical and quantum models, including the corresponding hyperparameters used.

The initial quantum model, denoted as the QNN model, consists of encoding and ansatz circuits, each utilizing five qubits. The number of qubits is determined by $log_2 N$ where N represents the number of features. Additionally, it incorporates a single linear layer that transforms 2^N into the corresponding output dimension. The actor model comprises a total of 405 parameters, calculated as follows: 5 (layers) x 3 (rotations) x 5 (qubits) + 32 x 10 (weights) + 10 (bias). On the other hand, the critic model shares the same composition, except for the output layer, which has 1 neuron. Therefore, the overall parameter count for the critic model is 108.

The classical counterpart's actor includes an input layer with 17 neurons, a hidden layer with 450 neurons, and an output layer with 10 neurons. This classical network has a total parameter count of 12,610, calculated as follows: 17 × 450 (weights) + 450 (bias) + 450 × 10 (weights) + 10 (bias). Similarly, the critic shares the same composition, except for the output layer, which has 1 neuron. Therefore, the total parameters for the critic model is 8.551.

The second Quantum model, QSNN as described in Section 3, consists of 2 QLIF cells with 15 layers and 15 neurons in the hidden layer, and 5 qubits. This results in a total of 880 parameters, calculated as follows: 17 x 15 (weights) +15 (bias) + 15 (layers) x 3 (learning parameters used in the three rotation gates for the ansatz circuit) x 5 (qubits) x 2 (number of QLIF cells) + 15 x 10 (weights) + 10 (bias). Similarly, the critic shares the same composition, except for the output layer, which has 1 neuron. Therefore, the total parameters for the critic model are 736.

For its classical counterpart, the actor is composed of a linear layer with 17 input neurons and 15 output neurons (hidden), two LIF neurons, and a final linear layer with 15 input neurons and ten output neurons (number of classes). Consequently, the total of parameters is 430, calculated as follows: 17 x 15 (weights) +15 (bias) + 15 x 10 (weights) + 10 (bias). Similarly, the critic shares the same composition, except for the output layer, which has 1 neuron. Therefore, the total parameters for the critic model are 286.

The third Quantum model, QLSTM, as described in Section 3, involves an actor component consisting of 1.481 parameters. This model includes a Linear layer that transforms an input size of

 doi:10.20944/preprints202403.0843.v1

17 into 42 output neurons (the sum of input size and hidden size), and a subsequent linear layer that converts 42 into 5 (the number of qubits):17 x 42 (weights) + 42 (bias) + 42 x 5 (weights) + 5 (bias). It then incorporates the parameters for the corresponding variational quantum circuits (VQCs): 4 x [5 (layer) x 5 (qubits) x 1 (learning parameter employed in the Rotation x gate for the ansatz circuit)]. Following this, another Linear layer transforms the number of qubits into the hidden size: 5 x 25 (weights) + 25 (bias). The final layer is a Linear transformation from hidden size to output dimension: 25 x 10 (weights) + 10 (bias). Similarly, the critic shares the same composition, except for the output layer consists of only 1 neuron. Thus, the total parameter count for the critic model is 1.247.

For its classical counterpart, the actor is defined by a total of 25.460 parameter. It includes a Linear layer with 42 input neurons and 25 output neurons, followed by two LSTM units, and a final layer transitioning from 25 input neurons to 10 output neurons (action space dimension). 4 x (17 + 25 + 1) x 25 + 4 x hidden (for the first LSTM layer) + [4 x (25 + 25 +1) x 25 + 4 x hidden (for the rest of LSTM layers)]*4 + 25 *10 +10 (hidden layer to output layer). The critic model is similarly structured, but the final output layer comprises just 1 neuron, leading to a total of 25.226 parameters.

The last agent, QSNN-QLSTM, as described in Section 3, consists of 5 layers of QSNN and 5 layers of QLSTM for the actor component. Each QSNN layer contains 2 QLIF cells, 15 neurons for the hidden layer and 5 qubits. Additionally, it incorporates a QLSTM module with 125 neurons for the hidden layer. In total, the model comprises 5.635 parameters, calculates as follows: [(17 x 15 + 15) + (5 (layers) x 3 (learning parameters utilized in the three rotation gates within the ansatz circuit) x 5 (qubits) x 2 (number of QLIF cells)) + (15 x 10 +10)] + [(15 x 140 (concat size) + 140) + (140 x 5 +5) + (5 layers x 4 (VQCs) x 1 (rotation gate) x 5 qubits) + (5 x 125 + 125 (for a Linear layer)) + (125 x 10 + 10 (for the output layer))]. Similarly, the critic shares the same composition, except for the output layer, which has 1 neuron. Therefore, the total parameters for the critic model are 4.357.

Table 3. Configuration of Classical Models. Hyperparameters and settings for the Artificial Neural Network, Spiking Neural Network, and Long Short-Term Memory models, respectively.

	MLP	**SNN**	**LSTM**
Optimizer	Adam(lr=1e-4)	Adam(lr=1e-3)	Adam(lr=1e-3)
Batch Size	32	16	32
BetaEntropy	0.01	0.01	0.01
Discount Factor	0.98	0.98	0.98
Steps	-	15	-
Hidden	-	15	25
Layers	Actor: [Linear[17, 450], ReLU Linear[450, 10]] Critic: [Linear[17, 450], ReLU Linear[450, 1]]	Actor: [Linear[17, 15] Lif1, Lif2 Linear[15, 10]] Critic: [Linear[17, 15] Lif1, Lif2 Linear[15, 1]]	Actor:[LSTM(17, 25, layers=5) Linear[25, 10]] Critic: [LSTM(17, 25, layers=5) Linear[25, 1]]

 doi:10.20944/preprints202403.0843.v1

Table 4. Configuration of Quantum Models. Hyperparameters and settings for the Quantum Neural Network, Quantum Spiking Neural Network, Quantum Long Short-Term Memory, and the novel model composed of the combination of the last two networks.

	QNN	**QSNN**	**QLSTM**	**QSNN-QLSTM**
Optimizer	Adam(lr=1e-2)	Adam(lr=1e-3)	Adam(lr=1e-3)	[(Adam(QSNN.parameters,lr= 1e-2), Adam(QLSTM.parameters, lr=1e-2))]
Batch Size	128	16	128	128
Pre-processing	Normalization	Normalization	-	Normalization
Post-processing	-	-	-	-
BetaEntropy	0.01	0.01	0.01	0.01
Discount Factor	0.98	0.98	0.98	0.98
Steps	-	15	15	15
Hidden	-	15	25	15 (QSNN), 125 (QLSTM)
Layers	Actor: [[5 QNN] ReLU Linear[2^N, 10]] Critic: [[5 QNN ReLU Linear[2^N, 1]]	Actor: [Linear[17, 15] 15 QSNN Linear[15, 10]] Critic: [Linear[17, 15] 15 QSNN Linear[15, 1]]	Actor: [Linear[17, 42] Linear[42, 5] 4 VQCs Linear[25, 5] Linear[5, 25] Linear[25, 10]] Critic: [Linear[17, 42] Linear[42, 5] 4 VQC´S Linear[25, 5] Linear[5, 25] Linear[25, 1]]	5 QSNN 5 QLSTM
Qubits	5	5	5	5
Encoding Strategy	Amplitud Encoding	Amplitud Encoding	Angle Encoding	Amplitud Encoding

 doi:10.20944/preprints202403.0843.v1

4.3. Results

The results of this use case are outlined in Table 5, which provides a summary of the average, best, and worst total accumulated rewards obtained by the four quantum agents and the three classical agents. In addition, the table also includes the computational time for each experiment, measured in seconds, along with the average accuracy, which represents the mean total accumulated reward obtained from 10 independent runs. Various unchanging seeds were utilized to ensure reproducibility..

In our results analysis, we observed the excellent performance of the novel agent QSNN-QLSTM, which emerged as the top-performing model in terms of average total reward, followed by the LSTM, QLSTM and QNN, QSNN and MLP models, in this order. Similarly, in the category of best total reward, the QSNN-QLSTM agent consistenly achieved the highest result, followed by the LSTM and MLP. Nevertheless, it's worth mentioning that the performance of the MLP appears as an outlier in this context. Consequently, the third position is occupied by the SNN. In the worst total reward category (as illustrated in Figure 14), after removing the outlier of the QSNN-QLSTM agent, once again this agent outperformed the others, with the QLSTM and QNN agents achieving the second and third positions, respectively. Furthermore, when considering the test rewards, the QSNN-QLSTM agent consistently obtained the highest rewards. Importantly, the computation time for quantum agents is longer compared to their classical counterparts. This is primarily due to the use of quantum simulators rather than real quantum computers, as simulating quantum operations on classical hardware incurs significant computational costs.

Upon further the analysis and examination of the boxplot shown in Figure 14, it becomes apparent that the QNN and QSNN-QLSTM models exhibit better robustness results than the others, with the MLP model showing the worst results.

Finally, considering the learning curves shown in Figure 15, we can witness a positive progression in learning for QSNN-QLSTM agent. Significantly, there is a reduction in variance during the later iterations, suggesting improved consistency and progress in learning. The remaining agents display similar curves, except for MLP, which shows the worst performance.

Table 5. Results obtained by Classical and Quantum models. Column 1: Classical models followed by Quantum versions and the novel quantum model proposed; Column 2: Average Total Reward post-training; Column 3: Best Total Reward post-training; Column 4: Worst Total Reward post-training; Column 5: Total reward (evaluation with deterministic policy); Column 6: Computational time in seconds.

	Average Tot.Reward	**Best Tot.Reward**	**Worst Tot.Reward**	**Test Reward**	**Time (s)**
MLP	-13.75	-9.67	-14.88	-13.25	297.8
SNN	-11.05	-9.79	-12.15	-13.31	307.6
LSTM	-10.56	-9.43	-12.11	-12.67	302.8
QNN	-10.92	-9.90	-11.75	-12.87	326.1
QSNN	-11.12	-9.89	-12.39	-13.32	467.4
QLSTM	-10.72	-10.14	-11.14	-12.19	335.42
QSNN-QLSTM	-9.40	-8.26	-12.73	-11.83	962.5

 doi:10.20944/preprints202403.0843.v1

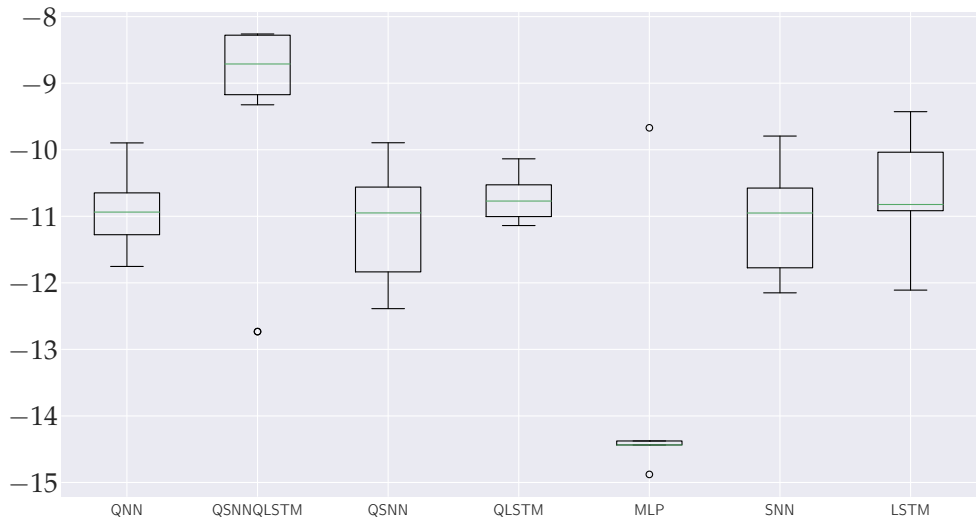


Figure 14. Boxplots that depict the distribution of average total reward achieved by quantum (QNN, QSNN, QLSTM and QSNN-QLSTM) and classical models (MLP, SNN and LSTM).

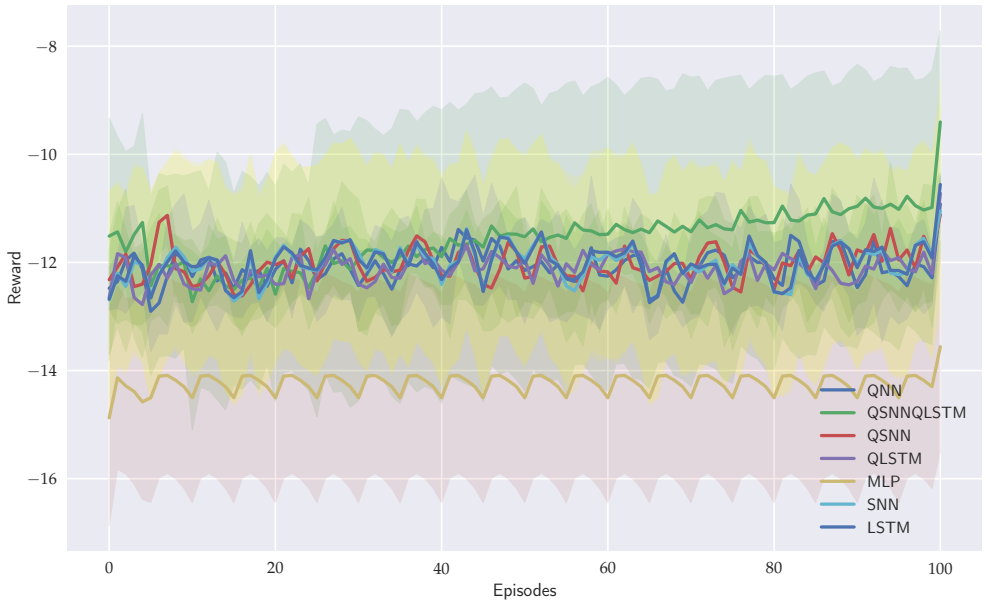


Figure 15. Learning curves obtained from seven models. Three classical models: MLP, SNN and LSTM. Four quantum models: QNN, QSNN, QLSTM and new brain-inpired model QSNN-QLSTM.

5. Discussion

We presented three classical models and four quantum models, one of which incorporates a novel approach inspired by brain function to address the scenario of reinforcement learning in the context of energy optimization. A notable observation from the results analysis is the effectiveness of utilizing quantum models inspired by the brain, which outperform classical approaches and other quantum models. Additionally, these quantum models demonstrate reduced complexity, requiring fewer parameters during training compared to classical counterparts. However, it's worth noting that, since they have been executed in simulators, the runtime is impacted due to the computational load

 doi:10.20944/preprints202403.0843.v1

they carry. Conducting and implementing large-scale experiments on existing quantum devices poses challenges, and the intrinsic noise in these systems may impact the effectiveness and dependability of the models. These constraints emphasize the necessity for continual advancements in both quantum hardware and simulators to enable decreased computation times.

6. Conclusions and Future Work

Recent advancements in neuroscience-inspired learning have extended Hebbian plasticity by incorporating rewards through the exploration of the neuromodulator dopamine. Within the realm of reinforcement learning (RL), review articles on neo-Hebbian learning have emerged, addressing diverse topics including the consolidation of associative memories and the connections among specific neo-Hebbian formulations and their counterparts in computational RL theory [69]. Moreover, it is widely acknowledged that quantum probability theory plays a more prominent role in quantum cognition compared to its classical analog. This is attributed to the fact that quantum probability theory, being a generalized form of probability theory, offers a more robust representation for tasks and internal states (for instance, allowing internal states to exist in an indefinite state prior to action) [27].

Our investigation has explored the integration of these theories into quantum agents. Unveiling the potential benefits of infusing neuroscientific insights into quantum reinforcement learning could pave the way for exciting advancements in the field, yielding new perspectives and opening novel avenues. Our hypothesis, based on mimicking the functionalities of the prefrontal cortex and hippocampus using QSNN and QLSTM respectively, for achieving enhanced performance, memory retention, and experience retrieval while mitigating catastrophic forgetting, has been validated with the consideration of the results. Thus, it is intriguing to continue researching the intersection of Neuroscience, Quantum Computing and Artificial Intelligence, as it could offer mutual insights to comprehend the physiological processes underlying in the human memory, particularly regarding what to retain and what to forget.

Additionally, it could address other challenges in deep RL such as generalization, adaptability to change, and navigating uncertain environments, among others. Deep RL systems have yet to demonstrate the ability to match humans in terms of flexible adaptation through structured inference, drawing on a substantial repository of background knowledge. If these systems can bridge this gap remains an unanswered and compelling question. Recent research suggests that under certain conditions, deep RL systems can efficiently leverage previous learning to systematically adapt to new and seemingly unfamiliar situations [70]. Nevertheless, this capability does not always occur [71], and exploring the distinctions is of significance to both AI and neuroscience [72].

Another unresolved question concerns how to represent the intricate interaction mechanisms among multiple brain areas using suitable RL models. This complexity arises from the fact that the decision-making process is distributed across various brain regions, which may dynamically change depending on specific task demands. Furthermore, there is currently a lack of a mathematical theory capable of elucidating all the remarkable discoveries in human brain learning. Hence, it is important for future research to advance an integrative theory and develop computational models capable of harmonizing various types of brain-inspired RL [2].

7. Patents

Not applicable.

Author Contributions: Eva Andrés conducted the conceptualization and implementation. All authors participated in the review process, and approved the final version.

Funding: This article was funded by the project QUANERGY (Ref. TED2021-129360B-I00), Ecological and Digital Transition R&D projects call 2022 by MCIN/AEI/10.13039/501100011033 and European Union NextGeneration EU/PRTR.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not required.

 doi:10.20944/preprints202403.0843.v1

Data Availability Statement: The scenario tested in the experimentation were originally proposed in [67] (with the simulator publicly available at https://github.com/ugr-sail/sinergym, accessed on 27 July 2022).

Acknowledgments: This article was funded by the project QUANERGY (Ref. TED2021-129360B-I00), Ecological and Digital Transition R&D projects call 2022 by MCIN/AEI/10.13039/501100011033 and European Union NextGeneration EU/PRTR.

Conflicts of Interest: The authors have no conflicts of interest to disclose.

Abbreviations

Abbreviations utilized in this manuscript comprise:

A2C	Advantage Actor–Critic
BPTT	backpropagation through time
BG	Basal Ganglia
CPU	Central Processing Unit
DQN	Deep Q-network
DRL	Deep reinforcement learning
HDC	Hyperdimensional Computing and Spiking
HVAC	Heating, ventilation, air-conditioning
KPI	Key performance indicator
LIF	Leaky integrate and fired Neuron
LSTM	Long/short-term memory
LTS	Long-term store
mPFC	Medial prefrontal cortex
MDP	Markov decision process
MLP	Multilayer perceptron
NISQ	Noisy, intermediate-scale quantum era
PFC	Prefronal Cortex
QC	Quantum computing
QLIF	Quantum Leaky integrate and fired Neuron
QLSTM	Quantum long/short-term memory
QML	Quantum machine learning
QNN	Quantum neural network
QSNN	Quantum spiking neural network
QPU	Quantum Processing Unit
QRL	Quantum reinforcement learning
RNN	Recurrent Neural Networks
RL	Reinforcement learning
SNN	Spiking neural network
S-R	Stimulus-response
STS	short-term store
vmPFC	Ventromedial prefrontal cortex
VQC	Variational quantum circuit

References

1. Zhao, L.; Zhang, L.; Wu, Z.; Chen, Y.; Dai, H.; Yu, X.; Liu, Z.; Zhang, T.; Hu, X.; Jiang, X.; et al. When brain-inspired AI meets AGI. *Meta-Radiology* **2023**, *1*, 100005. https://doi.org/10.1016/j.metrad.2023.100005.
2. Fan, C.; Yao, L.; Zhang, J.; Zhen, Z.; Wu, X. Advanced Reinforcement Learning and Its Connections with Brain Neuroscience. *Research* **2023**, *6*, 0064, [https://spj.science.org/doi/pdf/10.34133/research.0064]. https://doi.org/10.34133/research.0064.
3. Domenech, P.; Rheims, S.; Koechlin, E. Neural mechanisms resolving exploitation-exploration dilemmas in the medial prefrontal cortex. *Science* **2020**, *369*, eabb0184, [https://www.science.org/doi/pdf/10.1126/science.abb0184]. https://doi.org/10.1126/science.abb0184.

4. Baram, A.B.; Muller, T.H.; Nili, H.; Garvert, M.M.; Behrens, T.E.J. Entorhinal and ventromedial prefrontal cortices abstract and generalize the structure of reinforcement learning problems. *Neuron* **2021**, *109*, 713–723.e7. https://doi.org/10.1016/j.neuron.2020.11.024.
5. Bogacz, R.; Larsen, T. Integration of Reinforcement Learning and Optimal Decision-Making Theories of the Basal Ganglia. *Neural computation* **2011**, *23*, 817–51. https://doi.org/10.1162/NECO_a_00103.
6. Houk, J.; Adams, J.; Barto, A. A Model of How the Basal Ganglia Generate and Use Neural Signals that Predict Reinforcement. *Models of Information Processing in the Basal Ganglia* **1995**, *Vol. 13*.
7. Joel, D.; Niv, Y.; Ruppin, E. Actor–critic models of the basal ganglia: new anatomical and computational perspectives. *Neural Netw.* **2002**, *15*, 535–547.
8. Collins, A.G.E.; Frank, M.J. Opponent actor learning (OpAL): Modeling interactive effects of striatal dopamine on reinforcement learning and choice incentive. *Psychol. Rev.* **2014**, *121*, 337–366.
9. Maia, T.V.; Frank, M.J. From reinforcement learning models to psychiatric and neurological disorders. *Nat. Neurosci.* **2011**, *14*, 154–162.
10. Maia, T.V. Reinforcement learning, conditioning, and the brain: Successes and challenges. *Cogn. Affect. Behav. Neurosci.* **2009**, *9*, 343–364.
11. O'Doherty, J.; Dayan, P.; Schultz, J.; Deichmann, R.; Friston, K.; Dolan, R.J. Dissociable Roles of Ventral and Dorsal Striatum in Instrumental Conditioning. *Science* **2004**, *304*, 452–454, [https://www.science.org/doi/pdf/10.1126/science.1094285]. https://doi.org/10.1126/science.1094285.
12. Chalmers, E.; Contreras, E.B.; Robertson, B.; Luczak, A.; Gruber, A. Context-switching and adaptation: Brain-inspired mechanisms for handling environmental changes. In Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), 2016, pp. 3522–3529. https://doi.org/10.1109/IJCNN.2016.7727651.
13. Robertazzi, F.; Vissani, M.; Schillaci, G.; Falotico, E. Brain-inspired meta-reinforcement learning cognitive control in conflictual inhibition decision-making task for artificial agents. *Neural Networks* **2022**, *154*, 283–302. https://doi.org/10.1016/j.neunet.2022.06.020.
14. Zhao, Z.; Zhao, F.; Zhao, Y.; Zeng, Y.; Sun, Y. A brain-inspired theory of mind spiking neural network improves multi-agent cooperation and competition. *Patterns* **2023**, *4*, 100775. https://doi.org/10.1016/j.patter.2023.100775.
15. Zhang, K.; Lin, X.; Li, M. Graph attention reinforcement learning with flexible matching policies for multi-depot vehicle routing problems. *Physica A: Statistical Mechanics and its Applications* **2023**, *611*, 128451. https://doi.org/10.1016/j.physa.2023.128451.
16. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019, [arXiv:cs.CL/1810.04805].
17. Rezayi, S.; Dai, H.; Liu, Z.; Wu, Z.; Hebbar, A.; Burns, A.H.; Zhao, L.; Zhu, D.; Li, Q.; Liu, W.; et al. ClinicalRadioBERT: Knowledge-Infused Few Shot Learning for Clinical Notes Named Entity Recognition. In Proceedings of the Machine Learning in Medical Imaging; Lian, C.; Cao, X.; Rekik, I.; Xu, X.; Cui, Z., Eds., Cham, 2022; pp. 269–278.
18. Liu, Z.; He, X.; Liu, L.; Liu, T.; Zhai, X. Context Matters: A Strategy to Pre-train Language Model for Science Education. In Proceedings of the Artificial Intelligence in Education. Posters and Late Breaking Results, Workshops and Tutorials, Industry and Innovation Tracks, Practitioners, Doctoral Consortium and Blue Sky; Wang, N.; Rebolledo-Mendez, G.; Dimitrova, V.; Matsuda, N.; Santos, O.C., Eds., Cham, 2023; pp. 666–674.
19. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners, 2020, [arXiv:cs.CL/2005.14165].
20. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, 2021, [arXiv:cs.CV/2010.11929].
21. Aïmeur, E.; Brassard, G.; Gambs, S. Quantum speed-up for unsupervised learning. *Machine Learning* **2013**, *90*, 261–287.
22. Schuld, M.; Bocharov, A.; Svore, K.M.; Wiebe, N. Circuit-centric quantum classifiers. *Phys. Rev. A* **2020**, *101*, 032308.
23. Wiebe, N.; Kapoor, A.; Svore, K.M. Quantum Nearest-Neighbor Algorithms for Machine Learning. *Quantum Information and Computation* **2015**, *15*, 318–358.

24. Anguita, D.; Ridella, S.; Rivieccio, F.; Zunino, R. Quantum optimization for training support vector machines. *Neural networks* **2003**, *16*, 763–770.
25. Andrés, E.; Cuéllar, M.P.; Navarro, G. On the Use of Quantum Reinforcement Learning in Energy-Efficiency Scenarios. *Energies* **2022**, *15*.
26. Andrés, E.; Cuéllar, M.P.; Navarro, G. Efficient Dimensionality Reduction Strategies for Quantum Reinforcement Learning. *IEEE Access* **2023**, *11*, 104534–104553. https://doi.org/10.1109/ACCESS.2023.3318173.
27. Busemeyer, J.R.; Bruza, P.D. *Quantum Models of Cognition and Decision*; Cambridge University Press, 2012.
28. Li, J.A.; Dong, D.; Wei, Z.; Liu, Y.; Pan, Y.; Nori, F.; Zhang, X. Quantum reinforcement learning during human decision-making. *Nature Human Behaviour* **2020**, *4*, 294–307. https://doi.org/10.1038/s41562-019-0804-2.
29. Miller, E.K.; Cohen, J.D. An Integrative Theory of Prefrontal Cortex Function. *Annual Review of Neuroscience* **2001**, *24*, 167–202. https://doi.org/10.1146/annurev.neuro.24.1.167.
30. Atkinson, R.; Shiffrin, R. Human Memory: A Proposed System and its Control Processes. In *Human Memory: A Proposed System and its Control Processes.*; Spence, K.W.; Spence, J.T., Eds.; Academic Press, 1968; Vol. 2, *Psychology of Learning and Motivation*, pp. 89–195. https://doi.org/10.1016/S0079-7421(08)60422-3.
31. Andersen, P. *The hippocampus book*; Oxford university press, 2007.
32. Olton, D.S.; Becker, J.T.; Handelmann, G.E. Hippocampus, space, and memory. *Behavioral and Brain sciences* **1979**, *2*, 313–322.
33. Raman, N.S.; Devraj, A.M.; Barooah, P.; Meyn, S.P. Reinforcement Learning for Control of Building HVAC Systems. In Proceedings of the 2020 American Control Conference (ACC), 2020, pp. 2326–2332. https://doi.org/10.23919/ACC45564.2020.9147629.
34. Wang, Y.; Velswamy, K.; Huang, B. A Long-Short Term Memory Recurrent Neural Network Based Reinforcement Learning Controller for Office Heating Ventilation and Air Conditioning Systems. *Processes* **2017**, *5*. https://doi.org/10.3390/pr5030046.
35. Fu, Q.; Han, Z.; Chen, J.; Lu, Y.; Wu, H.; Wang, Y. Applications of reinforcement learning for building energy efficiency control: A review. *Journal of Building Engineering* **2022**, *50*, 104165. https://doi.org/10.1016/j.jobe.2022.104165.
36. Hebb, D. *The Organization of Behavior: A Neuropsychological Theory*; Taylor & Francis, 2005.
37. Eshraghian, J.K.; Ward, M.; Neftci, E.; Wang, X.; Lenz, G.; Dwivedi, G.; Bennamoun, M.; Jeong, D.S.; Lu, W.D. Training Spiking Neural Networks Using Lessons From Deep Learning, 2021.
38. Tavanaei, A.; Ghodrati, M.; Kheradpisheh, S.R.; Masquelier, T.; Maida, A. Deep learning in spiking neural networks. *Neural Networks* **2019**, *111*, 47–63.
39. Lobo, J.L.; Del Ser, J.; Bifet, A.; Kasabov, N. Spiking Neural Networks and online learning: An overview and perspectives. *Neural Networks* **2020**, *121*, 88–100. https://doi.org/10.1016/j.neunet.2019.09.004.
40. Lapicque, L. Recherches quantitatives sur l'excitation electrique des nerfs. *J. Physiol. Paris.* **1907**, *9*, 620–635.
41. Zou, Z.; Alimohamadi, H.; Zakeri, A.; Imani, F.; Kim, Y.; Najafi, M.H.; Imani, M. Memory-inspired spiking hyperdimensional network for robust online learning. *Scientific Reports* **2022**, *12*, 7641. https://doi.org/10.1038/s41598-022-11073-3.
42. Kumarasinghe, K.; Kasabov, N.; Taylor, D. Brain-inspired spiking neural networks for decoding and understanding muscle activity and kinematics from electroencephalography signals during hand movements. *Scientific Reports* **2021**, *11*, 2486. https://doi.org/10.1038/s41598-021-81805-4.
43. Banino, A.; Barry, C.; Uria, B.; Blundell, C.; Lillicrap, T.; Mirowski, P.; Pritzel, A.; Chadwick, M.J.; Degris, T.; Modayil, J.; et al. Vector-based navigation using grid-like representations in artificial agents. *Nature* **2018**, *557*, 429–433. https://doi.org/10.1038/s41586-018-0102-6.
44. Bengio, Y.; Simard, P.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* **1994**, *5*, 157–166.
45. Graves, A.; Liwicki, M.; Fernández, S.; Bertolami, R.; Bunke, H.; Schmidhuber, J. A Novel Connectionist System for Unconstrained Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2009**, *31*, 855–868.
46. Graves, A.; Schmidhuber, J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural networks : the official journal of the International Neural Network Society* **2005**, *18*, 602–610.
47. Hochreiter, S.; Schmidhuber, J. LSTM can solve hard long time lag problems. *Advances in Neural Information Processing Systems* **1996**, *9*, 473–479.

48. Triche, A.; Maida, A.S.; Kumar, A. Exploration in neo-Hebbian reinforcement learning: Computational approaches to the exploration–exploitation balance with bio-inspired neural networks. *Neural Networks* **2022**, *151*, 16–33. https://doi.org/10.1016/j.neunet.2022.03.021.
49. Dong, H.; Ding, Z.; Zhang, S.; Yuan, H.; Zhang, H.; Zhang, J.; Huang, Y.; Yu, T.; Zhang, H.; Huang, R. *Deep Reinforcement Learning: Fundamentals, Research, and Applications*; Springer Nature, 2020. http://www.deepreinforcementlearningbook.org.
50. Sutton, R.S.; Barto, A.G., The Reinforcement Learning Problem. In *Reinforcement Learning: An Introduction*; MIT Press, 1998; pp. 51–85.
51. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. https://doi.org/10.1038/nature14236.
52. Shao, K.; Zhao, D.; Zhu, Y.; Zhang, Q. Visual Navigation with Actor-Critic Deep Reinforcement Learning. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 1–6. https://doi.org/10.1109/IJCNN.2018.8489185.
53. Macaluso, A.; Clissa, L.; Lodi, S.; Sartori, C. A Variational Algorithm for Quantum Neural Networks. In Proceedings of the Computational Science – ICCS 2020. Springer International Publishing, 2020, pp. 591–604.
54. Benedetti, M.; Lloyd, E.; Sack, S.; Fiorentini, M. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology* **2019**, *4*, 043001. https://doi.org/10.1088/2058-9565/ab4eb5.
55. Zhao, C.; Gao, X.S. QDNN: deep neural networks with quantum layers. *Quantum Machine Intelligence* **2021**, *3*, 15.
56. Wittek, P. *Quantum Machine Learning: What Quantum Computing means to data mining*; Elsevier, 2014.
57. Weigold, M.; Barzen, J.; Leymann, F.; Salm, M. Expanding Data Encoding Patterns For Quantum Algorithms. In Proceedings of the 2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C), 2021, pp. 95–101.
58. McCloskey, M.; Cohen, N.J. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In *Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem*; Bower, G.H., Ed.; Academic Press, 1989; Vol. 24, *Psychology of Learning and Motivation*, pp. 109–165. https://doi.org/10.1016/S0079-7421(08)60536-8.
59. Zenke, F.; Poole, B.; Ganguli, S. Continual Learning Through Synaptic Intelligence. In Proceedings of the Proceedings of the 34th International Conference on Machine Learning; Precup, D.; Teh, Y.W., Eds. PMLR, 06–11 Aug 2017, Vol. 70, *Proceedings of Machine Learning Research*, pp. 3987–3995.
60. Rusu, A.A.; Rabinowitz, N.C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R.; Hadsell, R. Progressive Neural Networks. *CoRR* **2016**, *abs/1606.04671*, [1606.04671].
61. Shin, H.; Lee, J.K.; Kim, J.; Kim, J. Continual Learning with Deep Generative Replay. *CoRR* **2017**, *abs/1705.08690*, [1705.08690].
62. Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.C.; Veness, J.; Desjardins, G.; Rusu, A.A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. Overcoming catastrophic forgetting in neural networks. *CoRR* **2016**, *abs/1612.00796*, [1612.00796].
63. Eshraghian, J.K.; Ward, M.; Neftci, E.; Wang, X.; Lenz, G.; Dwivedi, G.; Bennamoun, M.; Jeong, D.S.; Lu, W.D. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE* **2023**, *111*, 1016–1054.
64. Crawley, D.; Pedersen, C.; Lawrie, L.; Winkelmann, F. EnergyPlus: Energy Simulation Program. *Ashrae Journal* **2000**, *42*, 49–56.
65. Mattsson, S.E.; Elmqvist, H. Modelica - An International Effort to Design the Next Generation Modeling Language. *IFAC Proceedings Volumes* **1997**, *30*, 151–155. 7th IFAC Symposium on Computer Aided Control Systems Design (CACSD '97), Gent, Belgium, 28-30 April, https://doi.org/10.1016/S1474-6670(17)43628-7.
66. Zhang, Z.; Lam, K.P. Practical Implementation and Evaluation of Deep Reinforcement Learning Control for a Radiant Heating System. In Proceedings of the Proceedings of the 5th Conference on Systems for Built Environments, New York, NY, USA, 2018; BuildSys '18, p. 148–157. https://doi.org/10.1145/3276774.3276775.

 doi:10.20944/preprints202403.0843.v1

67. Jiménez-Raboso, J.; Campoy-Nieves, A.; Manjavacas-Lucas, A.; Gómez-Romero, J.; Molina-Solana, M. Sinergym: A Building Simulation and Control Framework for Training Reinforcement Learning Agents. In Proceedings of the Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, New York, NY, USA, 2021; p. 319–323. https://doi.org/10.1145/3486611.3488729.
68. Scharnhorst, P.; Schubnel, B.; Fernández Bandera, C.; Salom, J.; Taddeo, P.; Boegli, M.; Gorecki, T.; Stauffer, Y.; Peppas, A.; Politi, C. Energym: A Building Model Library for Controller Benchmarking. *Applied Sciences* **2021**, *11*. https://doi.org/10.3390/app11083518.
69. Triche, A.; Maida, A.S.; Kumar, A. Exploration in neo-Hebbian reinforcement learning: Computational approaches to the exploration–exploitation balance with bio-inspired neural networks. *Neural Networks* **2022**, *151*, 16–33. https://doi.org/10.1016/j.neunet.2022.03.021.
70. Hill, F.; Lampinen, A.; Schneider, R.; Clark, S.; Botvinick, M.; McClelland, J.L.; Santoro, A. Environmental drivers of systematicity and generalization in a situated agent, 2020, [arXiv:cs.AI/1910.00571].
71. Lake, B.M.; Baroni, M. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks, 2018, [arXiv:cs.CL/1711.00350].
72. Botvinick, M.; Wang, J.X.; Dabney, W.; Miller, K.J.; Kurth-Nelson, Z. Deep Reinforcement Learning and Its Neuroscientific Implications. *Neuron* **2020**, *107*, 603–616. https://doi.org/10.1016/j.neuron.2020.06.014.