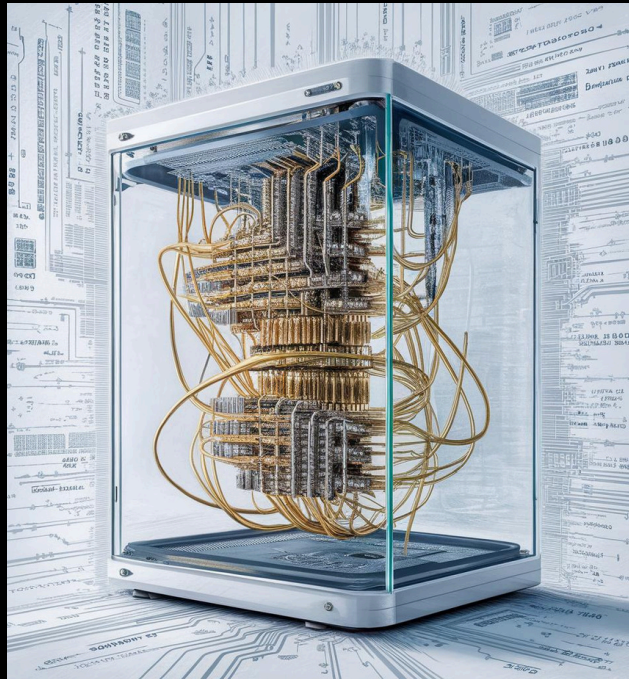


# Parametrized Quantum Circuits for Reinforcement Learning



*Doctoral Thesis*

*Eva Andrés Núñez*



UNIVERSIDAD  
DE GRANADA



**UNIVERSIDAD  
DE GRANADA**

# Parametrized Quantum Circuits for Reinforcement Learning

Eva Andrés Núñez

Tesis doctoral realizada dentro del programa:  
Doctorado en Tecnologías de la Información y la Comunicación

Directores:  
Manuel Pegalajar Cuéllar  
Gabriel Navarro Garulo

Escuela Técnica Superior de Ingeniería Informática y Telecomunicaciones  
Departamento de Ciencias de la Computación e Inteligencia Artificial

2024

Editor: Universidad de Granada. Tesis Doctorales  
Autor: Eva María Andrés Núñez  
ISBN: 978-84-1195-560-7  
URI: <https://hdl.handle.net/10481/97436>

Hoy antes del alba, subí a la colina, miré los cielos  
apretados de luminarias y le dije a mi espíritu:  
cuando conozcamos todos esos mundos y el placer y  
la sabiduría de todas las cosas que contienen,  
¿estaremos tranquilos y satisfechos? Y mi espíritu  
dijo: no, ganaremos esas alturas sólo para seguir  
adelante.

---

*Walt Whitman*



# Contents

|          |                                                                                           |           |
|----------|-------------------------------------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                                                                       | <b>22</b> |
| 1.1      | Introduction . . . . .                                                                    | 24        |
| 1.2      | Objectives . . . . .                                                                      | 27        |
| <b>2</b> | <b>State of the Art</b>                                                                   | <b>29</b> |
| 2.1      | Background . . . . .                                                                      | 31        |
| 2.1.1    | Quantum Computing . . . . .                                                               | 31        |
| 2.1.2    | Quantum Neural Networks . . . . .                                                         | 34        |
| 2.1.3    | Deep Reinforcement Learning . . . . .                                                     | 36        |
| 2.1.4    | Quantum Reinforcement Learning . . . . .                                                  | 39        |
| 2.1.5    | Spiking neural network and QSNN . . . . .                                                 | 40        |
| 2.1.6    | LSTM and QLSTM . . . . .                                                                  | 45        |
| <b>3</b> | <b>Methodology</b>                                                                        | <b>48</b> |
| 3.1      | Methodology . . . . .                                                                     | 50        |
| <b>4</b> | <b>Contributions</b>                                                                      | <b>53</b> |
| 4.1      | Contributions . . . . .                                                                   | 55        |
| 4.1.1    | QRL in the energy consumption optimization problem:<br>Benefits and Limitations . . . . . | 55        |
| 4.1.2    | The problem of scalability . . . . .                                                      | 60        |
| 4.1.3    | The problem of catastrophic forgetting . . . . .                                          | 62        |
| <b>5</b> | <b>Results</b>                                                                            | <b>68</b> |
| 5.1      | Results . . . . .                                                                         | 70        |
| 5.1.1    | QRL in the energy consumption optimization problem:<br>benefits and limitations . . . . . | 70        |
| 5.1.2    | The problem of scalability . . . . .                                                      | 71        |

|          |                                                                                               |           |
|----------|-----------------------------------------------------------------------------------------------|-----------|
| 5.1.3    | The problem of catastrophic forgetting . . . . .                                              | 72        |
| <b>6</b> | <b>Conclusions and future work</b>                                                            | <b>73</b> |
| 6.1      | Conclusions and future work . . . . .                                                         | 75        |
| 6.2      | Conclusiones y trabajo futuro . . . . .                                                       | 77        |
| <b>7</b> | <b>Publications</b>                                                                           | <b>80</b> |
| 7.1      | On the Use of Quantum Reinforcement Learning in Energy<br>Efficiency Scenarios . . . . .      | 82        |
| 7.1.1    | Introduction . . . . .                                                                        | 83        |
| 7.1.2    | Background . . . . .                                                                          | 86        |
|          | Artificial Neural Networks . . . . .                                                          | 87        |
|          | Reinforcement Learning . . . . .                                                              | 88        |
|          | Quantum Computing and Variational Quantum Circuits                                            | 91        |
| 7.1.3    | Methodology . . . . .                                                                         | 94        |
| 7.1.4    | Experiments . . . . .                                                                         | 98        |
|          | Use Case 1: HVAC Control . . . . .                                                            | 99        |
|          | Problem Statement . . . . .                                                                   | 99        |
|          | Experimental Settings . . . . .                                                               | 102       |
|          | Results . . . . .                                                                             | 105       |
|          | Use Case 2: Energy Management in Electrical Vehicles                                          | 108       |
|          | Problem Statement . . . . .                                                                   | 108       |
|          | Experimental Settings . . . . .                                                               | 109       |
|          | Results . . . . .                                                                             | 112       |
|          | Use Case 3: Profit Optimization of Charging Stations .                                        | 115       |
|          | Problem Statement . . . . .                                                                   | 115       |
|          | Experimental Settings . . . . .                                                               | 117       |
|          | Results . . . . .                                                                             | 120       |
| 7.1.5    | Discussion . . . . .                                                                          | 124       |
| 7.1.6    | Conclusions and Future Work . . . . .                                                         | 124       |
| 7.2      | Efficient Dimensionality Reduction Strategies for Quantum<br>Reinforcement Learning . . . . . | 126       |
| 7.2.1    | Introduction . . . . .                                                                        | 127       |
| 7.2.2    | Background . . . . .                                                                          | 130       |
|          | Artificial Neural Networks . . . . .                                                          | 131       |
|          | Quantum Neural Networks . . . . .                                                             | 132       |
|          | Deep Reinforcement Learning . . . . .                                                         | 134       |
| 7.2.3    | Methodology . . . . .                                                                         | 136       |

|          |                                                                  |            |
|----------|------------------------------------------------------------------|------------|
| 7.2.4    | Experimentation . . . . .                                        | 144        |
|          | Use Case 1: 5-Zones-Building . . . . .                           | 145        |
|          | Problem Statement . . . . .                                      | 145        |
|          | Experimental Settings . . . . .                                  | 147        |
|          | Results . . . . .                                                | 151        |
|          | Use Case 2: Datacenter . . . . .                                 | 155        |
|          | Problem Statement . . . . .                                      | 155        |
|          | Experimental Settings . . . . .                                  | 156        |
|          | Results . . . . .                                                | 160        |
|          | Use Case 3: Warehouse . . . . .                                  | 164        |
|          | Problem Statement . . . . .                                      | 164        |
|          | Experimental Settings . . . . .                                  | 164        |
|          | Results . . . . .                                                | 168        |
|          | Discussion . . . . .                                             | 170        |
|          | Conclusions and Future work . . . . .                            | 172        |
| 7.3      | Brain-inspired agents for Quantum Reinforcement Learning . .     | 176        |
| 7.3.1    | Introduction . . . . .                                           | 177        |
| 7.3.2    | Background . . . . .                                             | 180        |
|          | Spiking Neural Networks . . . . .                                | 181        |
|          | Long Short-Term Memory . . . . .                                 | 187        |
|          | Deep Reinforcement Learning . . . . .                            | 191        |
|          | Quantum Neural Networks . . . . .                                | 194        |
| 7.3.3    | Methodology . . . . .                                            | 195        |
| 7.3.4    | Experimentation . . . . .                                        | 197        |
|          | Problem Statement . . . . .                                      | 198        |
|          | Experimental Settings . . . . .                                  | 203        |
|          | Results . . . . .                                                | 208        |
| 7.3.5    | Discussion . . . . .                                             | 210        |
| 7.3.6    | Conclusions and Future Work . . . . .                            | 212        |
| <b>8</b> | <b>Brain-inspired for enhanced pattern recognition: Unifying</b> |            |
|          | <b>Quantum Spiking Neural Networks and QLSTM</b>                 | <b>214</b> |
| 8.0.1    | Introduction . . . . .                                           | 216        |
| 8.0.2    | Background . . . . .                                             | 218        |
|          | Spiking Neural Networks . . . . .                                | 219        |
|          | Long Short-Term Memory . . . . .                                 | 225        |
|          | Quantum Neural Networks . . . . .                                | 229        |
| 8.0.3    | Methodology . . . . .                                            | 230        |

|       |                                       |     |
|-------|---------------------------------------|-----|
| 8.0.4 | Experimentation . . . . .             | 232 |
|       | Problem Statement . . . . .           | 233 |
|       | Experimental Settings . . . . .       | 234 |
|       | Results . . . . .                     | 237 |
| 8.0.5 | Discussion . . . . .                  | 239 |
| 8.0.6 | Conclusions and Future Work . . . . . | 240 |

# List of Figures

|      |                                                                              |     |
|------|------------------------------------------------------------------------------|-----|
| 2.1  | Quantum Circuit . . . . .                                                    | 33  |
| 2.2  | General VQC Schema . . . . .                                                 | 34  |
| 2.3  | General Reinforcement Learning diagram . . . . .                             | 36  |
| 2.4  | Spiking Neural Networks . . . . .                                            | 42  |
| 2.5  | Structure of a biological neuron . . . . .                                   | 43  |
| 2.6  | SNN pipeline . . . . .                                                       | 44  |
| 2.7  | LSTM Cell Architecture . . . . .                                             | 46  |
| 4.1  | QRL schema . . . . .                                                         | 58  |
| 4.2  | Preprocessing with classical layer . . . . .                                 | 59  |
| 4.3  | Example of a 7-qubit VQC structure . . . . .                                 | 59  |
| 4.4  | Data-reuploading example . . . . .                                           | 61  |
| 4.5  | QSNN-QLSTM cotraining Annomaly Detection . . . . .                           | 64  |
| 4.6  | QLSTM model . . . . .                                                        | 66  |
| 4.7  | QNN architecture utilizing amplitud encoding . . . . .                       | 67  |
| 4.8  | QSNN architecture . . . . .                                                  | 67  |
| 4.9  | QSNN-QLSTM co-training process QDRL . . . . .                                | 67  |
| 7.1  | MultiLayer Perceptron . . . . .                                              | 87  |
| 7.2  | General Reinforcement Learning diagram . . . . .                             | 89  |
| 7.3  | Example of a quantum circuit with two qubits. . . . .                        | 92  |
| 7.4  | Example of a Variational Quantum Circuit with 4 qubits. . . . .              | 93  |
| 7.5  | QRL Schema . . . . .                                                         | 94  |
| 7.6  | General hybrid quantum–classical agent structure. . . . .                    | 96  |
| 7.7  | VQC. Preprocessing with input data manipulation . . . . .                    | 96  |
| 7.8  | VQC. Preprocessing with hybrid Classical/Quantum Neural<br>Network . . . . . | 97  |
| 7.9  | VQC Circuit Example . . . . .                                                | 98  |
| 7.10 | Zone building plan. . . . .                                                  | 100 |

|      |                                                                                      |     |
|------|--------------------------------------------------------------------------------------|-----|
| 7.11 | Boxplots of average total reward obtained from VQC and MLP<br>in use case 1. . . . . | 106 |
| 7.12 | Learning curve obtained from VQC and MLP in use case 1. .                            | 106 |
| 7.13 | Boxplots of average total reward obtained from VQC and MLP<br>in use case 2. . . . . | 114 |
| 7.14 | Learning curve obtained from VQC and MLP in use case 2. .                            | 115 |
| 7.15 | Boxplots of average total reward obtained from VQC and MLP<br>in use case 3. . . . . | 122 |
| 7.16 | Learning curve obtained from VQC and MLP in use case 3. .                            | 123 |
| 7.17 | MultiLayer Perceptron . . . . .                                                      | 131 |
| 7.18 | General VQC Schema . . . . .                                                         | 134 |
| 7.19 | General Reinforcement Learning diagram . . . . .                                     | 135 |
| 7.20 | QRL Schema . . . . .                                                                 | 137 |
| 7.21 | First proposed architecture . . . . .                                                | 138 |
| 7.22 | Second proposed architecture . . . . .                                               | 139 |
| 7.23 | Third proposed architecture . . . . .                                                | 140 |
| 7.24 | Encoding circuit . . . . .                                                           | 141 |
| 7.25 | Amplitude embedding tree . . . . .                                                   | 141 |
| 7.26 | <b>Zone building plan.</b> . . . .                                                   | 146 |
| 7.27 | Building-5Zones Boxplots VQC-5, VQC-10, VQCDR-10 and<br>MLP . . . . .                | 153 |
| 7.28 | Building-5Zones Learning Curves VQC-5, VQC-10, VQCDR-10<br>and MLP . . . . .         | 154 |
| 7.29 | Datacenter structure . . . . .                                                       | 156 |
| 7.30 | Datacenter Boxplots VQC-5, VQC10, VQCDR-10 and MLP .                                 | 161 |
| 7.31 | Datacenter Learning curves MLP and VQC . . . . .                                     | 162 |
| 7.32 | Warehouse scheme . . . . .                                                           | 164 |
| 7.33 | Warehouse Boxplots VQC-5, VQC-10, VQCDR-10 and MLP .                                 | 170 |
| 7.34 | Warehouse Learning curves VQC-5, VQC-10, VQCDR-10 and<br>MLP . . . . .               | 171 |
| 7.35 | LIF neuron model . . . . .                                                           | 185 |
| 7.36 | Biological neuron . . . . .                                                          | 186 |
| 7.37 | SNN pipeline . . . . .                                                               | 187 |
| 7.38 | LSTM Cell Architecture . . . . .                                                     | 188 |
| 7.39 | General Reinforcement Learning diagram . . . . .                                     | 192 |
| 7.40 | General VQC Schema . . . . .                                                         | 194 |
| 7.41 | QNN architecture . . . . .                                                           | 197 |
| 7.42 | QLSTM Cell . . . . .                                                                 | 198 |

|      |                                                                           |     |
|------|---------------------------------------------------------------------------|-----|
| 7.43 | QLSTM Architecture . . . . .                                              | 199 |
| 7.44 | QSNN Architecture . . . . .                                               | 200 |
| 7.45 | QSNN-QLSTM Training Process . . . . .                                     | 200 |
| 7.46 | <b>Zone building plan.</b> . . . .                                        | 201 |
| 7.47 | Boxplots QNN, QSNN, QLSTM, QSNN-QLSTM, MLP, SNN,<br>LSTM . . . . .        | 210 |
| 7.48 | Learning curves QNN, QSNN, QLSTM, QSNN-QLSTM, MLP,<br>SNN, LSTM . . . . . | 211 |
| 8.1  | LIF Neuron Model . . . . .                                                | 223 |
| 8.2  | Biological Neuron . . . . .                                               | 224 |
| 8.3  | SNN Pipeline . . . . .                                                    | 225 |
| 8.4  | LSTM Cell Architecture . . . . .                                          | 226 |
| 8.5  | General VQC Schema . . . . .                                              | 229 |
| 8.6  | QNN Architecture . . . . .                                                | 231 |
| 8.7  | QLSTM Cell . . . . .                                                      | 233 |
| 8.8  | QLSTM Architecture . . . . .                                              | 234 |
| 8.9  | QSNN Architecture . . . . .                                               | 235 |
| 8.10 | QSNN-QLSTM Training Process . . . . .                                     | 236 |
| 8.11 | Anomaly Detection Boxplots . . . . .                                      | 240 |
| 8.12 | Anomaly Detection Learning Curves . . . . .                               | 241 |



## Agradecimientos

Quiero agradecer sobre todo a mis directores de tesis, con quienes he trabajado muy agusto durante estos años. Siempre estuvieron dispuestos a brindarme su ayuda y orientación, lo que contribuyó a avivar mi pasión por la investigación. En particular, deseo agradecer a Manolo Pegalajar por su confianza, al otorgarme la libertad necesaria para desarrollar y disfrutar mi propia tesis. Aprecio enormemente tu constante apoyo, tu cercanía y la valiosa enseñanza que he recibido de ti.

Asimismo, quiero dedicar un agradecimiento especial a mi marido Oscar, cuyo apoyo y ánimo inquebrantables fueron fundamentales en este camino.

# Abstract

Quantum Computing (QC) is currently undergoing significant research despite being in its early stage and facing limitations during the NISQ (Noisy Intermediate-Scale Quantum) era, where simulators and emulators are being extensively utilized. Particularly, Quantum Machine Learning (QML) is gaining increasing interest area in the scientific community. However, Reinforcement Learning (RL) has received comparatively less attention in this field.

To address this gap, this study explores RL using Parametrized Quantum Circuits (PQC), with a specific focus on energy efficiency scenarios. Although PQCs have demonstrated success in various domains, including supervised learning and unsupervised learning, their application as quantum algorithm has been controversial due to challenges such as Barren Plateaus, which resemble the vanishing gradient phenomenon, as well as issues related to generalization and scalability, many of which are shared with classical models.

We will compare their performance with classical approaches and propose solutions for their limitations.

In parallel, recent advancements in artificial intelligence (AI) have led to the emergence of brain-inspired AI, an interdisciplinary field blending principles from neuroscience, psychology, and computer science to enhance the development of Artificial General Intelligence (AGI). For instance, current language models such as ChatGPT and GPT-4 employ Reinforcement Learning with human feedback (RLHF) to align their behavior with human values.

This interdisciplinary fusion has driven innovation, particularly in RL, where insights from neurobiology and neuropsychology have revolutionized algorithm development, sparking a surge in research activity. Researchers have leveraged these insights to develop novel mechanisms for understanding intelligent decision-making processes in the brain, such as meta-learning to mimic neurotransmitter dynamics and the emulation of theory of mind capabilities through Spiking Neural Networks (SNN) for actor Networks and Artificial Neural Networks (ANN) for critic networks. Attentional Reinforcement Learning techniques have also found applications in various domains such as the capacitated vehicle routing problem [1].

Consequently, integrating of neuroscientific theories into quantum agents could not only help solve existing challenges, such as noise interference, infrequent events, and mitigating catastrophic forgetting, but also provide

valuable insights that could open up new research avenues in this field. In this context, we introduce an innovative model inspired by the hypothalamus and hippocampus to address challenges such as catastrophic forgetting, noise susceptibility, Barren Plateaus, and improve convergence.

This research represents a novel integration of neuroscientific principles into Quantum Computing.

## Resumen

La Computación Cuántica (QC) está actualmente experimentando una investigación significativa, a pesar de encontrarse en sus etapas iniciales y enfrentar limitaciones durante la era NISQ, donde los simuladores y emuladores se utilizan ampliamente. En particular, el Aprendizaje Automático Cuántico (QML) está ganando gran interés en la comunidad científica, aunque el Aprendizaje por Refuerzo (RL) ha recibido comparativamente menos atención en este ámbito.

Para abordar este gap, este estudio explora RL utilizando Circuitos Cuánticos Parametrizados (PQC), con un enfoque específico en escenarios de eficiencia energética. Aunque los PQC han demostrado éxito en varios dominios, incluidos el aprendizaje supervisado y no supervisado, su aplicación como algoritmo cuántico ha sido controvertida debido a desafíos como las mesetas estériles (Barren Plateau), que se asemejan al fenómeno del gradiente desvaneciente, así como problemas relacionados con la generalización y la escalabilidad, muchos de los cuales son compartidos con los modelos clásicos.

En paralelo, los recientes avances en inteligencia artificial (AI) han dado lugar al surgimiento de una AI inspirada en el cerebro, un campo interdisciplinario que combina principios de neurociencia, psicología e informática para mejorar el desarrollo de la Inteligencia Artificial General (AGI). Por ejemplo, los modelos de lenguaje actuales como ChatGPT y GPT-4 emplean aprendizaje por refuerzo con retroalimentación humana (RLHF) para alinear su comportamiento con los valores humanos.

Esta fusión interdisciplinaria ha impulsado la innovación, especialmente en RL, donde los conocimientos de neurobiología y neuropsicología han revolucionado el desarrollo de algoritmos, provocando un aumento en la actividad investigativa. Los investigadores han aprovechado estos conocimientos para desarrollar mecanismos novedosos que permitan comprender los procesos de toma de decisiones inteligentes que ocurren en el cerebro, como el meta-aprendizaje para imitar la dinámica de los neurotransmisores y la emulación de las capacidades de teoría de la mente a través de redes neuronales de impulsos (SNN) para redes de actores y redes neuronales artificiales (ANN) para redes de críticos. Las técnicas de aprendizaje por refuerzo atencional también han encontrado aplicaciones en diversos dominios como el problema de enrutamiento de vehículos con capacidad.

Como resultado, la integración de teorías neurocientíficas en agentes cuánticos podría no solo ayudar a resolver los desafíos existentes, como

la interferencia de ruido, eventos poco frecuentes y la mitigación del olvido catastrófico, sino también proporcionar ideas valiosas que podrían abrir nuevas vías de investigación en este campo. En este contexto, presentamos un modelo innovador inspirado en el hipotálamo e hipocampo para abordar desafíos como el olvido catastrófico, la susceptibilidad al ruido, las mesetas estériles y mejorar la convergencia.

Esta investigación representa una integración novedosa de principios neurocientíficos en la computación cuántica.

## Thesis Motivation

Quantum Computing has attracted significant attention in recent years; however, Quantum Reinforcement Learning (QRL) remains relatively under-explored compared to other branches of QML. This research gap presents an intriguing opportunity for investigation, as understanding the potential of QRL is crucial. Furthermore, the limitations of Quantum Neural Networks (QNNs) have sparked debates within the scientific community. While they have demonstrated promising results in both supervised and unsupervised learning, they face challenges similar to classical models, such as Barren Plateaus, the latter akin to the vanishing gradient problem, which may have contributed to the early winter of Deep Neural Networks.

This thesis aims to explore the potential and limitations of Variational Quantum Circuits (VQCs) also known as Parametrized Quantum Circuits (PQCs) or Quantum Neural Networks (QNNs) by comparing them with classical models. Our objective is to contribute to the state-of-the-art by proposing strategies or models that can help mitigate or resolve these limitations, not only for quantum models but also for classical ones. Specifically, we will focus on the use of QNNs in Reinforcement Learning for various energy efficiency use cases and supervised learning in anomaly detection. We will explore their potential, compare them with classical models, and propose strategies and models to address challenges such as Barren Plateaus, scalability to circuits with a larger number of qubits, noise filtering, handling infrequent events, and mitigating catastrophic forgetting.

Among other proposals, we will present a new model based on VQCs inspired by neuroscience, which emulate the functionality of the hypothalamus and the hippocampus. This approach is motivated by the widely accepted memory model proposed by Atkinson and Shiffrin [2], which outlines short-term storage (STS) for transient memory storage and long-term storage (LTS) for maintaining memories over extended periods. QSNN (Quantum Spiking Neural Network) component captures information at a low-level perspective. Subsequently, the QLSTM (Quantum Long Short-Term Memory) processes this information at a higher level, identifying and memorizing correlated patterns while reinforcing long-term memorization.



## Compilation of Published Works

This doctoral thesis consists of a compilation of four research papers published in high-impact scientific journals, one of which is currently under review. The references for these papers are provided below:

- Andrés, E.; Cuéllar, M.P.; Navarro, G. On the Use of Quantum Reinforcement Learning in Energy-Efficiency Scenarios. *Energies* 2022, 15, 6034. <https://doi.org/10.3390/en15166034>
- E. Andrés, M. P. Cuéllar and G. Navarro. Efficient Dimensionality Reduction Strategies for Quantum Reinforcement Learning in *IEEE Access*, vol. 11, pp. 104534-104553, 2023, doi: 10.1109/ACCESS.2023.3318173.
- Eva Andrés, Manuel Pegalajar Cuéllar, and Gabriel Navarro. Brain-inspired agents for Quantum Reinforcement Learning. *Mathematics*, 12(8), 2024.

Additionally, given its importance and to complete the objectives, the work utilizing a variation of the novel brain-inspired model for an anomaly detection use case, which is currently under review by the journal *Quantum Machine Intelligence* (Springer Nature), is included. This work helps to verify the suitability of this model, providing important insights into its adaptability and potential use in different scenarios, such as reinforcement learning and anomaly detection. This work is presented in Section 8:

- Andrés, E.; Cuellar, M.P.; Navarro, G. Brain-inspired for enhanced pattern recognition: Unifying Quantum Spiking Neural Networks and QLSTM (peer-review)



# Chapter 1

## Introduction

Me parece haber sido sólo un niño jugando en la orilla del mar, divirtiéndose y buscando una piedra más lisa o una concha más bonita de lo normal, mientras el gran océano de la verdad yacía ante mis ojos con todo por descubrir.

---

*Isaac Newton*



## 1.1 Introduction

In the past decade, Deep Reinforcement Learning (DRL) has made significant strides, achieving superhuman performance in complex games such as Chess, Go [3], Atari games [4], robotics [5], autonomous driving [6], recommender systems, and language models [7, 8], among others. Additionally, the field of artificial intelligence has seen remarkable advancements, notably with the emergence of brain-inspired artificial intelligence, which integrates insights from neuroscience, psychology, and computer science to develop more efficient and powerful AI systems [9]. 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 [10].

Recent advancements in neuroscience-inspired learning have extended Hebbian plasticity by incorporating rewards through the exploration of the neuromodulator dopamine. In this context, research on neo-Hebbian learning has resulted in several comprehensive review articles, addressing diverse topics such as the consolidation of associative memories and the relationships between specific neo-Hebbian formulations and their counterparts in computational RL theory [11].

Concurrently, Quantum Computing has emerged as a captivating paradigm, blending principles from computer science, physics, and engineering. Rooted in quantum mechanics phenomena like superposition and entanglement, Quantum Computing offers the promise of significant advancements over classical algorithms due to its exploitation of these quantum phenomena. Early theoretical works showcase the potential of quantum algorithms, including Grover’s algorithm for searching unsorted lists in  $O(\sqrt{N})$  time [12], Shor’s algorithm for breaking cryptosystems like RSA (Rivest, Shamir and Adleman) in polynomial time [13], and simulating quantum systems [14].

Quantum Machine Learning (QML) has emerged as a compelling area of interest, a discipline at the intersection of Quantum Computing and machine learning, aiming to leverage quantum computers to accelerate computations or solve more complex problems efficiently. While strides have been made in supervised and unsupervised learning tasks, Quantum Reinforcement Learning has received less attention in the QML community. Recent approaches involve Grover Iterations [15] and Parametrized Quantum Circuits (PQCs), also known as Variational Quantum Circuits (VQCs) or Quantum Neural Networks (QNNs) [16]. Nevertheless, ongoing advancements in Deep Reinforcement

Learning (DRL) and Quantum Computing technologies, such as Google’s Cirq, IBM’s Qiskit, and Xanadu’s PennyLane, are fueling growing interest in QML and QRL among researchers [17, 18]. QNNs play a pivotal role in Quantum Machine Learning [19, 20], they are hybrid algorithms, combining quantum circuits with trainable parameters optimized using classical techniques. They are capable of approximating continuous functions [21, 19], enabling tasks such as optimization, approximation, and classification. Although, these models have shown success in various domains, including supervised learning [22, 23, 24] and unsupervised learning [25], their application as quantum algorithm has been controversial, as they face challenges such as the phenomenon of Barren Plateaus (BP), generalization, or scalability, many of which are shared with classical models.

This thesis aims to explore the application of VQC in various RL energy efficiency environments, and a small portion dedicated to supervised learning for anomaly detection, thus addressing the aforementioned challenges. Furthermore, the integration of neuroscientific theories into quantum agents could not only help solve these and other issues, such as ignoring noise, infrequent events, and mitigating catastrophic forgetting but also provide valuable insights that could pave the way for new research avenues in this field.

The inclusion of supervised learning in this thesis is justified to examine the overall behavior of Quantum Neural Networks and our proposed brain-inspired model in this context. This will allow us to gain a better understanding of their capability and applicability, in addition to contributing to our overall assessment of their potential and limitations.

It is worth to note that this thesis comprises two clearly differentiated parts: First, the PhD dissertation; and second, the scientific contributions published. Section 1.2 outlines the objectives of this thesis. Section 2.1 introduces a general overview of the state of the art and the concepts supporting this research. Section 3.1 describes the methodology used in the development of this work, and Section 4.1 provides a description of the main publications derived from this thesis, including an analysis of the results obtained during the research period in 5.1. The conclusions and future work are presented in Section 6.1. The second part of the present manuscript comprises the publications in scientific journals, as outlined in Section 7, along with the outgoing work, currently under review by the journal Quantum Machine Intelligence, presented in Section 8.

- Andrés, E.; Cuéllar, M.P.; Navarro, G. "On the Use of Quantum

Reinforce- ment Learning in Energy-Efficiency Scenarios". *Energies* 2022, 15, 6034. <https://doi.org/10.3390/en15166034>.

- E. Andrés, M. P. Cuéllar and G. Navarro, "Efficient Dimensionality Reduction Strategies for Quantum Reinforcement Learning". *IEEE Access*, vol. 11, pp. 104534-104553, 2023, doi: 10.1109/ACCESS.2023.3318173.
- E. Andrés, M. P. Cuéllar, and G. Navarro, "Brain-inspired agents for Quantum Reinforcement Learning". *Mathematics*, vol. 12, no. 8, 2024. <https://www.mdpi.com/2227-7390/12/8/1230>. doi:10.3390/math12081230.

Furthermore, the study employing a novel brain-inspired model variation for anomaly detection, currently under review by Quantum Machine Intelligence (Springer Nature), is included to fulfill objectives. It verifies the model's suitability, offering insights into its adaptability and potential uses in various scenarios, including reinforcement learning and anomaly detection. This is detailed in Section 8:

- E. Andrés, M. P. Cuéllar, and G. Navarro, "Brain-inspired for enhanced pattern recognition: Unifying Quantum Spiking Neural Networks and QLSTM" (peer-review).

## 1.2 Objectives

The main objective of this doctoral thesis is to investigate Parametrized Quantum Algorithms in Machine Learning, especially in RL. This involves adapting classical algorithms such as DQN (Deep Q-Network) or A2C (Advantage Actor-Critic), among others, for use with quantum agents, as well as developing new proposals that utilize these circuit variations to address some of the challenges in both quantum and classical RL. To achieve this purpose, a comprehensive analysis of the limitations of the current state of the art will be carried out, such as Barren Plateaus problems, as well as the selection of encoding strategy and ansatz. Therefore, the main objective is divided into the following specific objectives:

1. Investigate the state of the art to appropriately evaluate existing solutions and limitations in the use of PQCs in Quantum Reinforcement Learning: conduct a thorough review of the scientific literature and recent developments in the field of Parametrized Quantum Circuits (PQCs) applied to Quantum Reinforcement Learning. This will involve identifying the strengths and weaknesses of current methodologies, with a special focus on unresolved issues and areas with potential for new contributions.
2. Explore data encoding techniques and ansatz layers: analyze and compare various data encoding strategies and ansatz configurations to optimize the performance of quantum algorithms in reinforcement learning contexts. This objective will include the implementation of different encoding techniques and the evaluation of their impact on the scalability, efficiency and accuracy of the models.
3. Analyze and apply various strategies to address Barren Plateaus problems: investigate the causes and proposed solutions for Barren Plateaus problems that affect the optimization of quantum algorithms. This will include implementing current mitigation techniques and experimenting with novel methods to improve the convergence and performance of quantum algorithms in the presence of these challenges.
4. Implement and analyze different use cases with RL algorithms for classical and quantum agents, comparing their performance: develop and implement case studies involving reinforcement learning algorithms for

both classical and quantum agents. Conduct experiments to compare the performance of quantum agents with their classical counterparts, evaluating key metrics such as learning speed, robustness, and higher accumulated rewards.

5. Propose and develop new solutions that contribute to improving the state of the art in the field of Machine Learning, especially in Reinforcement Learning: based on previous findings and analyses, design and test new algorithmic proposals and techniques that leverage the unique capabilities of quantum systems. These solutions will aim to overcome or mitigate current limitations providing significant improvements in the performance of Quantum Reinforcement Learning algorithms.
6. Establish a standard process for effectively addressing quantum machine learning solutions, especially in Deep Reinforcement Learning: develop a methodological framework that guides the implementation and evaluation of quantum machine learning solutions in the context of Deep Reinforcement Learning. This standard process will include practical recommendations, tools, and best practices to facilitate the adoption and development of quantum technologies in reinforcement learning applications.

# Chapter 2

## State of the Art

Nature isn't classical, dammit, and if you want to  
make a simulation of nature, you'd better make it  
quantum mechanical.

---

*Richard Feynman*



## 2.1 Background

This section covers the main concepts used in this thesis as well as a brief review of the state of the art in each of them. Section 2.1.1 will introduce the fundamental concepts of Quantum Computing. Next, we will introduce the concepts of Quantum Neural Networks, Spiking Neural Networks, and Long Short-Term Memory (LSTM) networks along with their quantum counterparts in Sections 2.1.2, 2.1.5, 2.1.6, respectively. Finally, Section 2.1.3 will present the basic concepts of Deep Reinforcement Learning.

### 2.1.1 Quantum Computing

The most basic unit of information in classical computing is the bit, and the underlying mathematical model for working with bits is  $\mathbb{Z}_2$ , the field with two elements. Unlike the classical model, the fundamental unit of information in Quantum Computing is the qubit, and the underlying mathematical model is the complex plane  $\mathbb{C}^2$ , where the standard orthonormal basis vectors are usually denoted as  $|0\rangle$  and  $|1\rangle$ . Hence, the value (state) of a qubit  $|\psi\rangle$  is a linear combination of these two vectors.

Subject to the constraint that  $|\alpha_0|^2 + |\alpha_1|^2 = 1$ , as the Born rule indicates that upon measuring this qubit, we will obtain  $|0\rangle$  with a probability of  $|\alpha_0|^2$  and  $|1\rangle$  with a probability of  $|\alpha_1|^2$ .

When adding a new bit to a classical system, the underlying state space increases by one dimension, as the resulting space is the Cartesian product of the previous one with  $\mathbb{Z}_2$ . Therefore, in systems of  $n$  bits, the state space is  $\times^n \mathbb{Z}_2 = \mathbb{Z}_2^n$ . On the other hand, when adding a new qubit to a quantum system, the underlying space doubles its dimension, as the resulting space is the tensor product of the previous one with  $\mathbb{C}^2$ , leading to exponential growth in dimension. In a system of  $n$  qubits, the underlying model is  $\mathbb{C}^{2^n} = \otimes^n \mathbb{C}^2$ , i.e., the tensor product of  $\mathbb{C}^2$ ,  $n$  times. For example, the vector space  $\mathbb{C}$  of  $\mathbb{C}^{2^2}$  has a basis formed by the vectors  $|0\rangle \otimes |0\rangle$ ,  $|0\rangle \otimes |1\rangle$ ,  $|1\rangle \otimes |0\rangle$ , and  $|1\rangle \otimes |1\rangle$ , which, for simplicity, are denoted as  $|00\rangle$ ,  $|01\rangle$ ,  $|10\rangle$ , and  $|11\rangle$ , respectively.

Classical computers manipulate bits using logic gates such as AND, OR, NOT, NAND, XOR, among others. Similarly, quantum computers manipulate qubits using quantum gates. A quantum gate is represented as a unitary matrix that acts on the state of the qubit system. All quantum operations must be reversible (except for measurement). Some examples of quantum gates include single-qubit rotations, such as Pauli-X ( $\sigma_x$ ), Pauli-Y ( $\sigma_y$ ), and

Pauli-Z ( $\sigma_z$ ) rotations with a phase of  $\pi$ , as well as their general rotation gates  $R_x(\theta)$ ,  $R_y(\theta)$ , and  $R_z(\theta)$ , which rotate a qubit around the axis specified by the given angle  $\theta$ . Other relevant gates are the entanglement gates, such as the CNOT (Controlled-NOT), which operates on two qubits. The unitary matrices of these gates are shown in Equations (2.1), (2.2), (2.3), and (2.4) respectively.

$$R_x(\theta) = \begin{pmatrix} \cos(\frac{\theta}{2}) & -i\sin(\frac{\theta}{2}) \\ -i\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix} \quad (2.1)$$

$$R_y(\theta) = \begin{pmatrix} \cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix} \quad (2.2)$$

$$R_z(\theta) = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix} \quad (2.3)$$

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (2.4)$$

A *quantum program*, implemented as a *quantum circuit*, consists of a sequence of quantum gates applied to one or more qubits. An example of this representation is shown in Figure 2.1, where a quantum circuit with two qubits is presented. In this example, a Hadamard (H) single-qubit quantum gate is first applied individually to the first qubit  $q_0$ . Then, a two-qubit CNOT gate is applied to both qubits. Finally, this quantum system is measured, resulting in one of the bell states  $[\frac{1}{\sqrt{2}}, 0, 0, \frac{1}{\sqrt{2}}]$ , which can also be represented as  $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ .

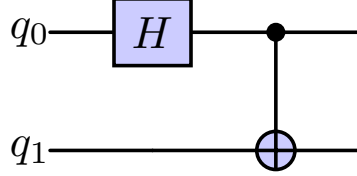


Figure 2.1: Example of a quantum circuit with two qubits.

The output of a quantum system is obtained using the *measurement* operator, which is used to perform measurements and determine the value of a specific observable. In the formalism of quantum mechanics, observables are represented by linear operators, which correspond to Hermitian matrices. When measuring an observable in a quantum system, we obtain one of the possible eigenvalues associated with that observable. The expectation or expected value of an observable in a given quantum state is the average value expected to be obtained when performing multiple measurements of the observable in that state. Mathematically, the expected value of an observable  $\langle A \rangle$  in a state  $|\psi\rangle$  is calculated as the inner product  $\langle\psi|A|\psi\rangle$ .

A commonly used observable is the Pauli-Z matrix, whose unitary matrix,  $\sigma_z$ , is shown in Equation (2.5). This matrix has two principal eigenvectors,  $|0\rangle$  and  $|1\rangle$ , with eigenvalues  $+1$  and  $-1$  respectively. For example, if the quantum state is  $|\psi\rangle = |0\rangle$ , then the expectation of the observable returns the value  $1$ , and if the quantum state is  $|\psi\rangle = |1\rangle$ , then the expectation of the observable returns the value  $-1$ . Any other intermediate value in the range  $[-1, 1]$  is related to the probabilities of  $\alpha_0$  and  $\alpha_1$  described previously [26].

$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (2.5)$$

In the current state of the art, recent advancements in enabling technologies are accelerating the timeline to achieve significant quantum advantage. For instance, in 2021, improvements in algorithms, error mitigation techniques, and hardware led to a x120 acceleration in computation and enhanced precision. In June 2023, IBM, in collaboration with Lawrence Berkeley National Laboratory and Purdue University, demonstrated that quantum computers can produce precise results, competing with high-performance simulations of certain material properties. These advancements not only inspired new

classical algorithms for material simulation in High-Performance Computing (HPC) but also underscore the importance of advancing simultaneously in the fields of HPC and Quantum Computing.

In the longer term, researchers are exploring methods to directly input quantum information into qubits from quantum sensors and sources, expanding the potential applications of Quantum Computing.

In the mid-term, the development of a computer with tens of thousands of qubits and sufficient error correction would move us beyond the NISQ era into the fault-tolerant era. Such devices could execute algorithms like Shor’s algorithm on a scale capable of addressing larger problem sets, such as breaking 2000-bit RSA asymmetric cryptosystem [27].

### 2.1.2 Quantum Neural Networks

Quantum Neural Networks (QNNs) play a pivotal role in Quantum Machine Learning (QML) [19, 20], utilizing Variational Quantum Circuits (VQCs). These VQCs are hybrid algorithms, combining quantum circuits with trainable parameters optimized using classical techniques. They are capable of approximating continuous functions [21, 19], enabling tasks such as optimization, approximation, and classification.

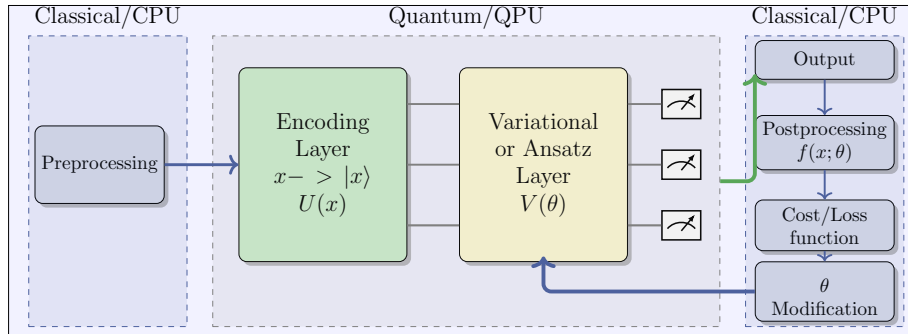


Figure 2.2: **General VQC Schema.** The dashed gray line delineates the process carried out within a Quantum Processing Unit (QPU), while the dashed blue line illustrates the processes executed in a CPU.

Figure 2.2 depicts the overall structure of a VQC. This hybrid approach involves the following stages [28]:

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 [29]. The selection of the encoding strategy affects to model expressive and therefore to fix better to the objective function [30].
3. **Variational Layer (QPU)**: This layer embodies the functionality of QNNs 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.
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 (Stochastic Gradient Descent). Gradient-free methods, such as SPSA (Simultaneous Perturbation Stochastic Approximation) or COBYLA (Constrained Optimization BY Linear Approximation), are also capable of estimating parameter updates.

Several studies have been conducted in QNN, such as [31], where the authors demonstrated that QNN can exhibit a favorable Fisher information spectrum, enabling faster training and expression of more functions compared to classical and quantum models, thus, enhancing generalization capability. Additionally, the utilization of QNNs have been explored in supervised learning [22, 23, 24], unsupervised learning [25], and for generative tasks [32, 33]. Other works have shown better expressiveness in QNNs models compared to classical neural networks [31, 34]. A comprehensive review of this model is conducted, also considering its execution on Quantum Hardware, in [35].

### 2.1.3 Deep Reinforcement Learning

Reinforcement learning (RL) [36] 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 unknown 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 illustrated in Figure 2.3: 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.

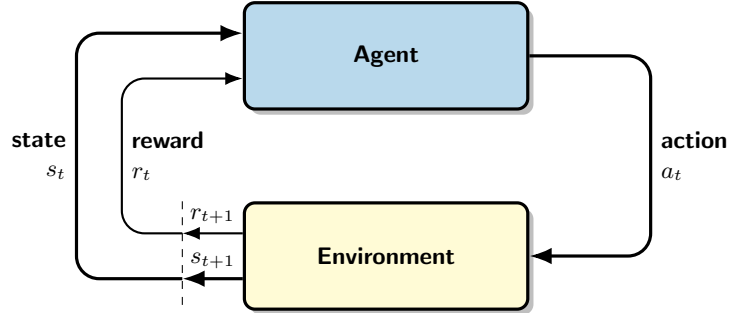


Figure 2.3: **General Reinforcement Learning diagram** [37]. At time  $t$ , the agent perceives state  $s_t$  and, based on this state, selects an action  $a_t$ . The environment then transitions to a new state  $s_{t+1}$  and provides the agent with this new state along with a reward  $r_{t+1}$ .

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

$$P(s'|s, a) = P[s_{t+1} = s' | s_t = s, a_t = a] \quad (2.6)$$

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  $\tau$ , beginning at time  $t_0$ . This cumulative reward, referred to as the return and defined in Equation (2.7), is influenced by the hyperparameter  $\gamma$ , which determines the relative significance of recent versus past rewards in the learning process.  $\gamma$  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 characterizes 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 (2.8), 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 (2.9), 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 (2.10)), which quantifies the benefit of selecting an action  $a$  in state  $s$  compared to the other available actions in the same state  $s$ .

$$R(\tau) = \sum_{t=t_0}^{\infty} r_t \gamma^{t-t_0} \quad (2.7)$$

$$Q(s, a) = \mathbb{E}_{\tau \sim \pi} (R(\tau) | s_t = s, a_t = a) = \sum_{s'} p(s' | s, a) (r(s, a, s') + \gamma \sum_{a'} \pi(a' | s') Q(s', a')) \quad (2.8)$$

$$V(s) = \mathbb{E}_{\tau \sim \pi} (R(\tau) | s_t = s) = \sum_i R(\tau_i) \pi(a_i | s) \quad (2.9)$$

$$Adv(s, a) = Q(s, a) - V(s) \quad (2.10)$$

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) [38] and policy gradient [39]. 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 (2.12). Here,  $\hat{Q}(s, a)$  represents the output value corresponding to an 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: [38, 39]. In this work, we employ the Advantage Actor-Critic (A2C) training algorithm [39], which is characterized by its designed loss function outlined in Equation (2.11).

$$LogLoss = - \sum_t Adv(s_t, a_t) \log \pi(a_t | s_t) \quad (2.11)$$

$$MSE = \sum_t \left( r(s_t, a_t, s_{t+1}) + \gamma \max_{a_{t+1}} [\hat{Q}(s_{t+1}, a_{t+1})] - \hat{Q}(s_t, a_t) \right)^2 \quad (2.12)$$

Over the past 10 years, various applications of DRL have emerged, including industrial process control [40], autonomous vehicles [41], robotics [42], traffic signal control [43], recommender systems [44], computer vision [45], medical [46, 10], combinatorial optimization problems [1] among others.

In the field of energy efficiency and energy management, there is a wide range of approaches in the state-of-the-art of RL. In datacenter environments, RL has been applied to tasks such as efficient resource allocation/deallocation in response to workload variations [47, 48], task scheduling [49, 50], predictive and efficient routing schemes [51], and minimizing power usage of resources [52]. Additionally, for energy efficiency in buildings, RL has been used for climate control in commercial buildings, such as the application of the Zap Q-learning algorithm to maintain temperature comfort within the desired range while reducing energy consumption [53]. Wang et al. proposed an

RL controller based on actor/critic models using Recurrent Neural Networks (LSTM) for HVAC control in buildings [54]. A comprehensive survey on RL applied to energy efficiency in buildings can be found in [55].

#### 2.1.4 Quantum Reinforcement Learning

Quantum Reinforcement Learning (QRL) is a relatively nascent research area. Due to limited global access to actual quantum hardware capable of running the learning process, researchers often resort to quantum computer simulators operating on classical computers. As a result, existing QRL research, such as that highlighted in [15, 19, 56, 57], tends to focus on well-tested, simplistic environments, predominantly sourced from the OpenAI Gym RL platform [58]. Despite these constraints, current QRL methodologies have yielded significant insights into applying RL principles within a quantum framework. Notable techniques include leveraging Grover iterations [15] to provide a quantum implementation of the classic Q-learning algorithm and employing Variational Quantum Circuits (VQC) [16] as quantum agents in deep Q-learning or policy gradient methods [19, 56, 57].

VQCs have been proposed as function approximators, replacing their classical counterparts in RL algorithms in the policy space, value space, or both. Typically, a classical learning heuristic is replaced by one with a quantum component. The typical hybrid pipeline, originally proposed for Q-function approximation by Chen et al. [59] is very similar to Figure 4.1. Where the DQN algorithm is generalized to a hybrid quantum reinforcement learning method by replacing the neural network approximator with a VQC. This approach was extended to policy approximation by Jerbi et al. [57]. Additionally, an extension to environments with continuous action space is proposed in [60].

We can find other applications related to robotics and similar control tasks are discussed in [61, 62, 63, 64]. Planning tasks of different form are the focus of [65, 66, 67]. Collaborative environments are addressed with multi-agent methods [68, 69, 70]. in the field of finances we can find some examples such as [18, 71].

The deployment of agents trained with VQC components presents several challenges, including scalability, generalization, and the barren plateau problem, among others. These challenges, along with their implications for various application domains, are currently under active investigation within the research community.

### 2.1.5 Spiking neural network and QSNN

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 2.5 [72].

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 led 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 below. Additionally, SNNs possess 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 [73, 74].

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 [75].

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  $\theta$ , 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 2.4.

Louis Lapicque [76] 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 2.4b).

A spike occurs when the membrane potential  $U(t)$  exceeds the threshold, defined as:

$$S_{out}(t) = \begin{cases} 1, & \text{if } U(t) > \theta \\ 0, & \text{otherwise} \end{cases} \quad (2.13)$$

Additionally, a reset function is triggered each time an output spike is generated. When a neuron receives a spike from external sources or from a preceding neuron, its membrane potential increases and begins to decay exponentially until a subsequent spike arrives.

$$U(t) = \underbrace{\beta U(t-1)}_{\text{decay}} + \underbrace{WX(t)}_{\text{input}} - \underbrace{S_{out}(t-1)\theta}_{\text{reset}} \quad (2.14)$$

where  $\beta$  represents the decay rate and  $S_{out}(t) \in \{0, 1\}$  is the output spike, 1 in case of activation and 0 in otherwise. In the first scenario, the reset term

subtracts the threshold  $\theta$  from the membrane potential, whereas in the second scenario, the reset term has no impact.

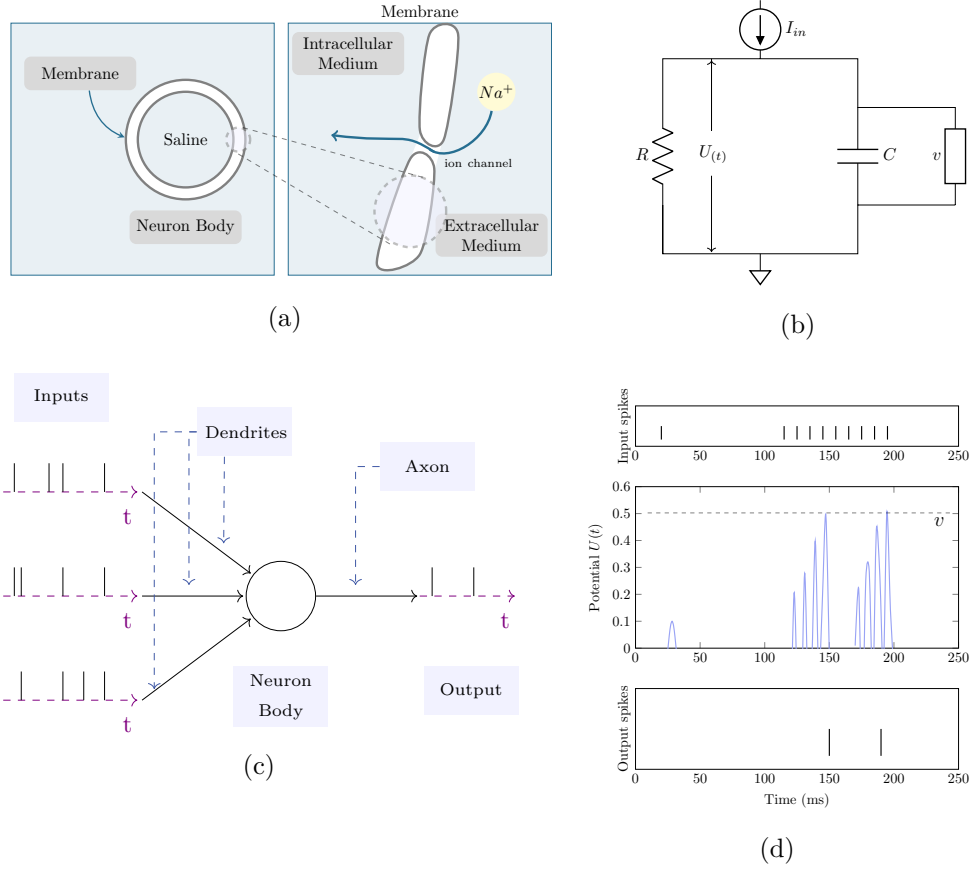


Figure 2.4: **The Leaky Integrate-and-Fire Neuron Model.** [73] Involves an insulating lipid bilayer membrane that separates the interior and exterior environments. Gated ion channels enable the diffusion of charge carriers, such as  $Na^+$ , through the membrane. 2.4a. This neuron function is often modeled using an RC circuit. When the membrane potential exceeds the threshold, a spike is generated 2.4b. Input spikes are transmitted to the neuron body via dendritic branches. Accumulation of sufficient excitation triggers spike emission at the output 2.4c. A simulation depicting the membrane potential  $U(t)$  reaching a threshold of  $\theta = 0.5V$ , leading to the generation of output spikes 2.4d

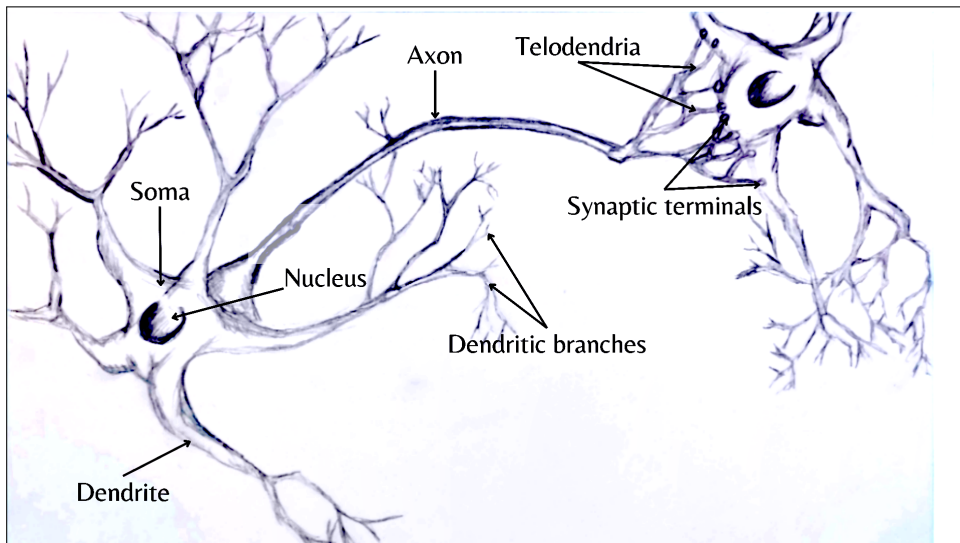


Figure 2.5: The standard structure of a neuron includes a cell body, also known as soma, housing the nucleus and other organelles; dendrites, which are slender, branched extensions that receive synaptic input from neighboring neurons; an axon; and synaptic terminals.

Various techniques exist for training SNN's [73], 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:

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial S} \underbrace{\frac{\partial S}{\partial U}}_{\{0, \infty\}} \frac{\partial U}{\partial I} \frac{\partial I}{\partial W} \quad (2.15)$$

Nevertheless, the derivative of the Heaviside step function from (2.13) 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 thesis, we utilize snntorch library, which defaults to using the arctangent function [73].

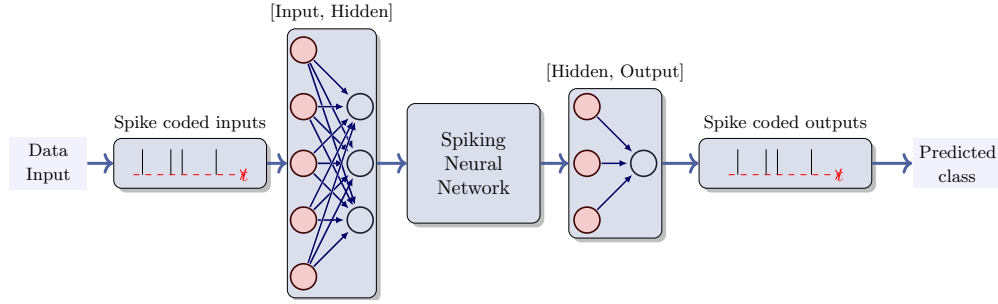


Figure 2.6: **SNN pipeline.** The input data for an SNN may undergo transformation into a firing rate or alternative encodings to produce spikes. Subsequently, the model is trained to accurately predict the correct class by employing encoding strategies, such as the highest firing rate or firing first, amongst other options

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 descent method. Figure 2.6 shows the general pipeline for this model for a classification task and the detailed architecture is defined in Section 4.1.3.

Previous works have been made inspiring in brain functionality emulating for classification tasks such as MINST dataset classification using SNN and HyperDimensional Computing (HDC) [77] or in the decoding and understanding of muscle activity and kinematics from electroencephalography signals [78], utilizing HDC and SNN for the MNIST classification problem [77]. Other works have explored the application of RL for navigation in dynamic and unfamiliar environments, supporting neuroscience-based theories that consider grid cells as crucial for vector-based navigation [79].

Finally, in relation to QSNN models found in the literature [80], the authors propose an algorithm to optimize the forward process using a circuit designed to calculate unsigned vector inner products for the MNIST dataset.

Another model is proposed in [81], which utilizes a combination of classical spiking neural network and variational quantum circuit for feature extraction, also targeted at the MNIST dataset.

### 2.1.6 LSTM and QLSTM

Long Short-Term Memory (LSTM) networks, a subset of Recurrent Neural Networks (RNNs), excel in learning order dependence in sequential prediction tasks. They address key challenges encountered in training RNNs, such as the exploding or vanishing gradient problem and the difficulty in retaining long-term dependencies. LSTMs achieve this by incorporating mechanisms to selectively retain or discard information [82, 83], making them indispensable in domains like machine translation, speech recognition, and time-series analysis [84, 85].

Introduced by Hochreiter and Schmidhuber in 1997 [86], LSTM modules feature three internal gates—input, forget, and output gates. These gates regulate the flow of information, enabling LSTMs to capture and retain essential patterns over extended sequences (Figure 2.7).

These gates are filters and each of them have its own neural-network, to control the flow of information at each time step. At any given moment, the output of an LSTM depends on three main factors: the cell state (the network’s current long-term memory), the hidden state (the output from the previous time step), and the input data at the current time step. These internal gates are described as follows:

- **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

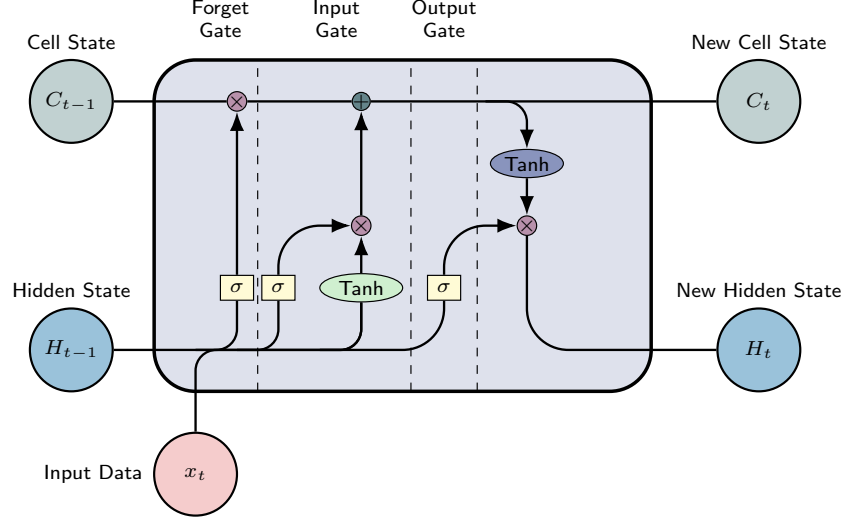


Figure 2.7: **LSTM Cell Architecture:** Featuring three essential gates (forget, input and output gates). The  $\tanh$  and  $\sigma$  blocks symbolize the hyperbolic tangent and sigmoid activation functions, correspondingly.  $x_t$  represents the input at time  $t$ ,  $h_t$  denotes the hidden state, and  $c_t$  signifies the cell state. The symbols  $\otimes$  and  $\oplus$  denote element-wise multiplication and addition, respectively.

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.

- **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 in the prediction, and the final layer of this model is a linear layer responsible for converting the hidden state into the ultimate prediction.

However, training these models, can be computationally intensive, requiring significant memory and processing power [87]. Hence, neural networks such as gated recurrent unit (GRU) [88], minimal gated unit (MGU) [89], and their variations [90, 91] were developed with the aim of attaining comparable performance to more complex models while utilizing fewer parameters, thereby reducing the computational expenses associated with training. Utilizing QLSTM could help alleviate these issues. Several models for QLSTM have been proposed in the literature, such as the one by Chen et al. [92], which employs six VQCs for QLSTM cell construction. Additionally, recent work demonstrates that QLSTM requires fewer epochs to converge in time series processing compared to classical LSTM [93].

# Chapter 3

## Methodology

The methods of theoretical physics should be applicable to all those branches of thought in which the essential features are expressible with numbers.

---

*Paul Dirac*



## 3.1 Methodology

In order to undertake this thesis, a rigorous methodology within the traditional scientific framework is essential. Given the significant experimental and theoretical components of this research, the following steps are proposed to achieve the objectives:

- **Analysis and Observation:** Studying and reviewing the state of the art in classic Reinforcement Learning, Artificial Neural Networks, as well as Quantum Reinforcement Learning and Parametrized Quantum Circuits, with the aim of gaining a deep understanding of the currently available techniques and their limitations.
- **Hypothesis formulation:** Design of new techniques and optimize state-of-the-art of algorithms to address or mitigate limitations observed, specially tailored to our specific use cases. Particularly, whether these novel quantum models or techniques can effectively solve or mitigate issues encountered in classical approaches. In our first study, we aimed to establish a procedure for comparing classical and quantum models, and to analyze the feasibility of applying Quantum Reinforcement Learning (QRL) in energy efficiency scenarios. Our hypothesis was that it is possible to implement QRL in such environments with at least the same level of convergence as classical methods. This involved evaluating the performance and potential advantages of QRL in optimizing energy consumption and management, while also developing a method for rigorous comparison between classical and quantum approaches.

In the second study, our focus was on scalability, aiming to facilitate the application of QRL to real-world problems characterized by a large number of variables. Our hypothesis posits that through the utilization of encoding techniques, which efficiently map classical data to a quantum state without loss of information, alongside the integration of classical layers capable of processing data in a reduced dimensionality, we can effectively address high-dimensional problems while maintaining performance and convergence. This approach seeks to demonstrate the ability of QRL to be seamlessly scaled for the management of complex, large-scale environments, thereby expanding its practical utility.

In our third and fourth studies, our objective was to investigate the potential benefits of incorporating neuroscientific theories into Quantum

Reinforcement Learning (QRL) algorithms. Our hypothesis posited that integrating insights from neuroscience would improve algorithm convergence and mitigate challenges such as catastrophic forgetting and noise. By leveraging neuroscientific principles, our aim was to bolster the robustness and adaptability of QRL algorithms, thereby ensuring sustained performance across different environmental conditions and over extended periods.

- **Experimentation:** Analysis of the results obtained by applying the the current state of the art models and techniques, as well as the novel models/techniques proposed, and comparing them to their classical counterparts. This analysis will focus particularly on use cases with large variables, specially in energy-efficiency environments or anomaly detection datasets.
- **Hypothesis test:** Comparison and validation of the results obtained with the proposed models and techniques against those reported in the literature. In our study, we assumed a hybrid quantum-classical QRL scenario, where the VQC-agent, characterized by a set of trainable parameters, interacts with a classical (non-quantum) environment. In the first work’s three use cases (building, electric vehicle and charge station), we employed a classical linear layer for both the actor and critic VQC. This layer aims to reduce the dimensionality of the feature space, converting the number of features into a manageable number of qubits. In our second study, we applied dimensionality reduction techniques such as amplitude encoding, linear layer preprocessing, and data-reuploading across three different use cases (building, data center, and data warehouse). These techniques aimed to enhance convergence by creating expressive yet simplified models that also approximate the objective function well, while avoiding barren plateaus. In our third and fourth studies, we developed a new model inspired by the human brain, composed of Quantum Spiking Neural Networks (QSNN) and Quantum Long Short-Term Memory (QLSTM). The QSNN mimics the hypothalamus, capturing basic information, while the QLSTM functions like the hippocampus, identifying patterns and preventing memory loss. In Quantum Reinforcement Learning (QRL), we utilized an Actor-Critic algorithm and gradient descent method to optimize energy efficiency in a building scenario. For anomaly detection, the composite model

underwent training in two phases. Initially, an independent instance of QSNN was trained without the influence of QLSTM, using the gradient descent method. Subsequently, the model was partitioned and combined with a new instance of the QLSTM model for co-training. The loss from the QLSTM output was used to simultaneously update both QSNN and QLSTM.

- **Hypothesis validation or refutation:** Verify whether the conclusions obtained align with the proposed objectives and hypotheses. If the results do not meet the initial expectations, we revert back to step 2 and formulate a new hypothesis. In our research, we conducted comparisons with the current classic state of the art because there were no existing Quantum Reinforcement Learning (QRL) approaches in energy scenarios when we began. The same applies to the brain-inspired model, as there were no findings in the state of the art except for classical proposals. Therefore, comparisons were made with classical models.

The metrics employed covered accumulated rewards across different categories, including best, worst, average, and the reward achieved in the last episode of testing. Additionally, we evaluated the robustness of the algorithms, analyzed the learning curve, assessed the number of parameters used by the models, and evaluated computational time. For anomaly detection, we chose to utilize the f1-score instead of accumulated reward.

- **Thesis extraction or scientist theory:** Extracting, formalizing, and embracing the conclusions derived from the entire research process to formulate a theory that elucidates and substantiates the techniques developed in the experimentation. Each of the models developed throughout this research will comprise the present doctoral thesis.

The results and main conclusions extracted from this thesis are presented in 5.1 and 6.1, respectively. These have been primarily derived from the text of the articles comprising the thesis.

# Chapter 4

## Contributions

Nothing in life is to be feared, it is only to be understood.  
Now is the time to understand more, so that we may fear less.

---

*Marie Curie*



## 4.1 Contributions

This section encapsulates the key ideas pertaining to the scientific works of the present doctoral thesis. Each of the subsequent subsections provides a summary of each individual works, and in section 5.1, the results and conclusions derived will be outlined. These works are enumerated in the following lines:

- Andrés, E.; Cuéllar, M.P.; Navarro, G. "On the Use of Quantum Reinforcement Learning in Energy-Efficiency Scenarios". *Energies* 2022, 15, 6034. <https://doi.org/10.3390/en15166034>.
- E. Andrés, M. P. Cuéllar and G. Navarro, "Efficient Dimensionality Reduction Strategies for Quantum Reinforcement Learning". *IEEE Access*, vol. 11, pp. 104534-104553, 2023, doi: 10.1109/ACCESS.2023.3318173.
- E. Andrés, M. P. Cuéllar, and G. Navarro, "Brain-inspired agents for Quantum Reinforcement Learning". *Mathematics*, vol. 12, no. 8, 2024. <https://www.mdpi.com/2227-7390/12/8/1230>. doi:10.3390/math12081230.

Additionally, given its importance and to complete the objectives, the work utilizing a variation of the novel brain-inspired model for an anomaly detection use case, which is currently under review by the journal *Quantum Machine Intelligence* (Springer Nature), is included. This work helps to verify the suitability of this model, providing important insights into its adaptability and potential use in different scenarios, such as reinforcement learning and anomaly detection:

- E. Andrés, M. P. Cuéllar, and G. Navarro, "Brain-inspired for enhanced pattern recognition: Unifying Quantum Spiking Neural Networks and QLSTM" (peer-review).

### 4.1.1 QRL in the energy consumption optimization problem: Benefits and Limitations

In this preliminary study, our aim was to analyze and explore the benefits and limitations within energy efficient environments. Previous studies had primarily focused on small-scale environments such as Frozen Lake or Cart-Pole, among others from the OpenAI Gym RL suite. We sought to expand

this research by investigating larger environments and real-world scenarios, comparing them against classical models to understand the integration process of a variational quantum circuit into Reinforcement Learning algorithms. Our experimentation aimed to explore phenomena such as Barren Plateaus, the curse of dimensionality, convergence improvements, and the attainment of enhanced cumulative rewards.

Initially, we replicated the findings of the preliminary studies conducted in OpenAI environments [94, 56, 57], utilizing quantum inspired approach (Grover iteration) or gradient-based methods. To broaden our experimentation, we incorporated novel approaches such as genetic algorithms and Particle Swarm Optimization, with the goal of gaining insights into the current state of the art techniques. Subsequently, guided by the aforementioned objectives, we identified three energy efficiency use cases: optimizing energy consumption in buildings without compromising user comfort, managing energy consumption in electric vehicles, and maximizing profits for electric vehicle charging stations.

We opted to utilize gradient-based methods, reserving gradient-free approaches or new techniques discovered during these experiments for cases we encounter Barren Plateaus. However, some studies have shown that gradient-free algorithms (except evolutionary algorithms) do not effectively avoid Barren Plateaus [95, 96].

We configured the development environments to accommodate the existing setups. Our choice of algorithm included the Advantage Actor-Critic (A2C), alongside leveraging simulation software libraries such as Synergym to access biometric data from the building, the electric car, and the charging station. Development was carried out using PyTorch with PennyLane, as well as TensorFlow with Cirq and TensorFlow Quantum.

We assumed a hybrid quantum–classical QRL scenario, the agent, characterized by a set of parameters  $\theta = (\theta_1, \dots, \theta_k)$  interacts with a classical (non-quantum) environment, as shown in Figure 4.1. In the beginning of an episode, the quantum–classical hybrid agent receives a state information from the environment, determines its action from its policy  $\pi_\theta(a|s)$ , and perceives a reward  $r$  and the next environment state after applying the action to the environment. The policy  $\pi_\theta$  is given by a VQC running on a Quantum Processing Unit (QPU). The optimization of this policy is carried out by updating the parameters  $\theta$ , which is performed by a classical learning algorithm in a Central Processing Unit (CPU) with the objective of reducing the cost function [94].

For the three use cases, we utilized a classical linear layer for both the actor and critic VQC. This layer serves to reduce the dimensionality of the feature space, transforming the number of features into a manageable number of qubits (see Figure 4.2), with tensor product encoding technique.

As previously noted, VQCs, akin to classical neural networks, are prone to encountering the vanishing gradient problem, commonly known as the Barren Plateau phenomenon [16, 97]. Much like the vanishing gradient descent observed in classical computing, wherein the gradient exponentially diminishes with the increase in the number of qubits. Historically, vanishing gradients may have contributed to the early winter of Deep Neural Networks [98, 99, 100].

The study of the Barren Plateau (BP) phenomenon is an active area of research and is of great interest to the scientific community, as this problem poses a limitation on achieving superiority over classical computing. Currently, it is known that BPs are prone to emerge when exponentially increasing the complexity of the Hilbert space. This is because Levy’s lemma implies that the variance of measurements will decrease exponentially with the number of qubits, resulting in the average gradient of the objective function becoming zero over vast regions of quantum space. The probability of escaping from such a region, which is a plateau rather than a local minimum, is exponentially small in the number of qubits. However, the usage and parameterization of the space also play a fundamental role [97].

Another cause is the intrinsic nature of quantum models, as the gradient saturates exponentially with the number of qubits due to the normalization of the output state. However, this phenomenon also occurs in classical models using functions like softmax or sigmoid. Although, the relationships between the classical and quantum phenomena are still under study [97]. Furthermore, another cause of this phenomenon is the non-deterministic nature of quantum models, which also occurs in Reinforcement Learning. Lastly, another type of BP emerges from hardware noise in the system [101]. As previously mentioned, such effects are not mitigated by adopting gradient-free methods [95, 96], although it has not been demonstrated with evolutionary or bio-inspired algorithms.

Several techniques have been proposed to alleviate this problem. It is recommended to avoid random initializations in circuits to explore the space of quantum states. Initializing the ansatz following a beta distribution rather than a uniform one is suggested, as it performs better with random distributions and demonstrates less degradation of the gradient variance. Additionally,

noise or perturbations can be incorporated into the parameters (angles) before backpropagation [102]. Another approach involves using a reduced number of qubits and layers, as excessive expressibility in deep quantum models may lead to decreased circuit depth. Alternatively, restricting either the direction or angle of rotations, and reducing the number of entanglements could also prove effective [103, 104]. Another possibility is to use local cost functions. Pre-training segment by segment, is also recommended. Additionally, detecting these issues using classical shadows has been suggested [105].

In our studies, to mitigate BP, we typically utilized simple ansatz circuits with minimal layers and basic entanglement gates, as illustrated in Figure 4.3. Although in subsequent studies, we will explore new strategies to address this challenge.

Returning to the focus of this initial study, we presented a comparison between the quantum model and the MLP for all three use cases.

The publication associated to this study is:

---

Andrés, E.; Cuéllar, M.P.; Navarro, G. "On the Use of Quantum Reinforcement Learning in Energy-Efficiency Scenarios". *Energies* 2022, 15, 6034. <https://doi.org/10.3390/en15166034>.

---

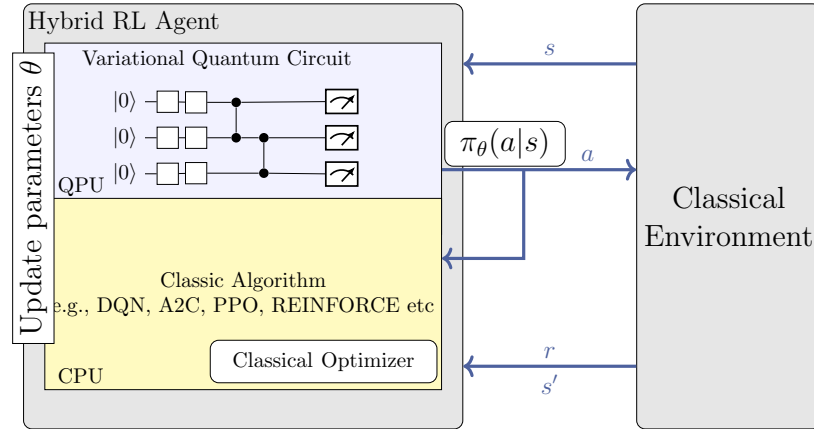


Figure 4.1: **QRL schema**. Example of hybrid agent and classical/quantum interaction to optimize the trainable parameters of the policy using a classical learning algorithm.

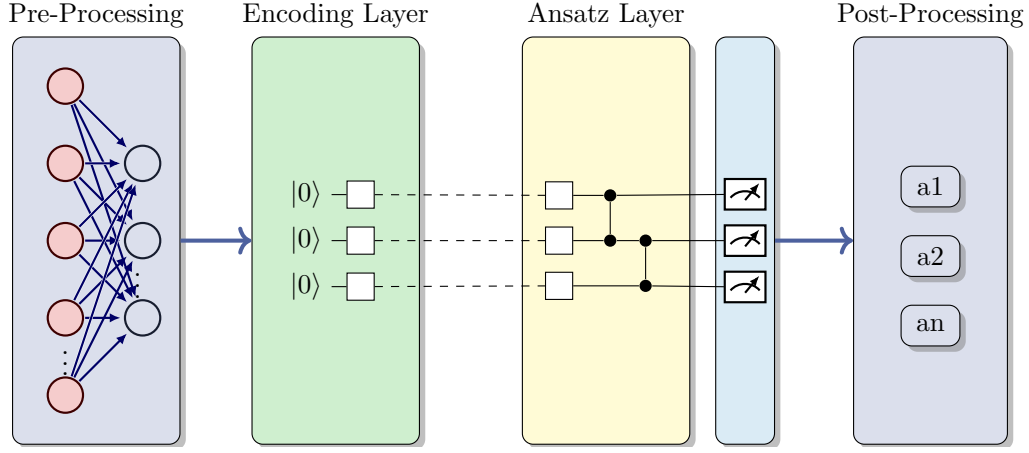


Figure 4.2: Preprocessing with a hybrid classical/Quantum Neural Network. The output of preprocessing is fed to the encoding circuit and then to the variational layers.

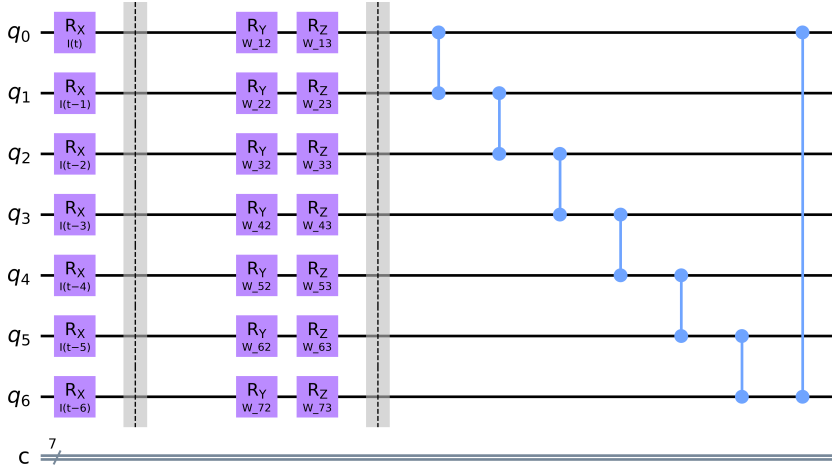


Figure 4.3: Example of a 7-qubit VQC with the structure used in the experiments. Left: encoding layer; middle: variational gates; right: entanglement CNOT gates organized in a ring.

### 4.1.2 The problem of scalability

classical Reinforcement Learning faces the challenge of the curse of dimensionality, especially in scenarios involving numerous environmental variables. When considering the utilization of quantum systems with a significant number of qubits, not only does this impact execution times, but it also increases the likelihood of encountering Barren Plateaus, alongside the inherent challenge of stabilizing a larger number of qubits on a real quantum computer. Consequently, we investigated and experimented with dimensionality reduction strategies and encodings. These enable effective operation in high-dimensional real-world environments while tracking these obstacles.

We examined and validated the efficacy of three strategies for data management and dimensionality reduction in real-world, large-scale environments, focusing on energy efficiency scenarios (a data warehouse, data center and a 5-Zone Building). The techniques encompass amplitude encoding, linear layer preprocessing, and data-reuploading, supplemented by trainable parameters [106, 107].

In summary, the objective was to explore the potential of quantum machine learning in enhancing real-world environments, particularly in energy efficiency use cases, by utilizing VQCs and employing DQN algorithms.

The three techniques used for comparison, both among themselves and with the classical MLP (MultiLayer Perceptron), were the following:

- The Amplitud Encoding strategy, which requires only  $\log_2 N$  qubits, where  $N$  is the state space dimension. This architecture is illustrated in Figure 4.7.
- Classical linear layer prior to the encoding ansatz, as depicted in Figure 4.2.
- Data-reuploading and scaling techniques with trainable parameters in the encoding circuit. We utilized a variation of the previous model but incorporating these techniques. Utilizing this technique alongside trainable input weights enhances model richness, facilitating a more accurate approximation of the objective function. Quantum models inherently excel at learning periodic functions in data, with repeated gate encoding expanding the frequency spectrum and enabling representation of a broader dataset. Proper scaling ensures effective data learning and

representation within the model's period. Additionally, employing data-reuploading in light of the non-cloning theorem optimizes processing efficiency, given classical neural networks' repeated input handling within hidden layers. The architecture of this model is presented in Figure 4.4.

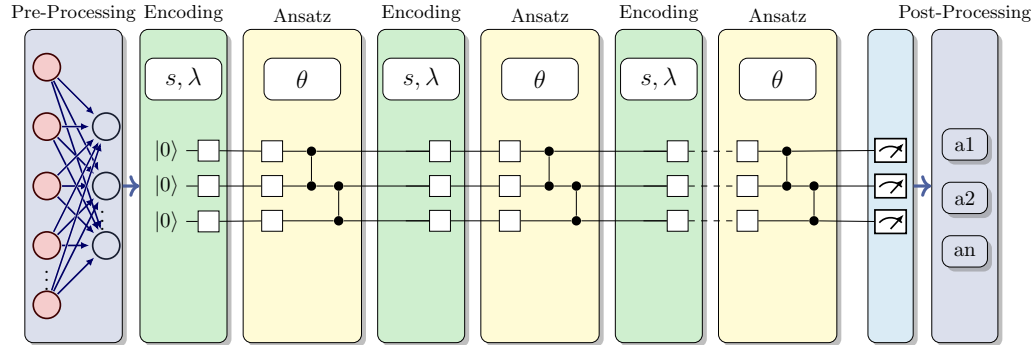


Figure 4.4: This architecture features a linear layer for input data dimensionality reduction, incorporating data-reuploading and scaling techniques. The encoding circuit consists of interleaved unitary gates and ansatz circuits. Learnable parameters include  $\lambda$ -input scaling parameters for encoding circuits and  $\theta$ -weights for ansatz circuits. The number of circuits is determined by the model's defined layers; in this example, three layers are depicted.

This work resulted in the publication of the following paper:

---

E. Andrés, M. P. Cuéllar and G. Navarro, "Efficient Dimensionality Reduction Strategies for Quantum Reinforcement Learning". IEEE Access, vol. 11, pp. 104534-104553, 2023, doi: 10.1109/ACCESS.2023.3318173.

---

### 4.1.3 The problem of catastrophic forgetting

With the objective of optimize convergence, avoid Barren Plateaus, mitigate noise effects in quantum computers, and address issues like catastrophic forgetting in classical Reinforcement Learning. We drew inspiration from neurobiology and psychology to emulate the intricate workings of the brain, particularly regarding memory.

Exploring previous investigations related to interactions between Reinforcement Learning and neuroscience, the most prominent theory aligns closely with the actor-critic model and the neural architecture of the Basal Ganglia [10, 108, 109, 110, 111, 112, 113]. The critic is responsible for learning state values and may be implemented in the ventral striatum, possibly in conjunction with the amygdala and orbitofrontal cortex. The actor, on the other hand, is responsible for learning Stimulus-Response (S-R) associations and may be implemented in the dorsal striatum. Both the critic and the actor utilize dopamine-signaled prediction errors to update their estimates [114, 115, 116].

In support of this perspective, and employing Actor-Critic algorithm, we developed a novel model that emulates 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 enhancing information with stronger space-time correlations, capturing 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, thus emulating the hippocampus and mitigating the risk of catastrophic forgetting [73]. Catastrophic forgetting, characterized by the loss of previously learned knowledge when new information is introduced, presents a significant challenge in real-time systems with small batch sizes, often limited to one [117].

In essence, our innovative approach combines the strengths of QSNN for novel learning with QLSTM’s capacity to preserve and build upon existing knowledge. Furthermore, the integration of QSNN and QLSTM not only aligns with the biological architecture of the human brain but also provides a comprehensive solution to the challenge of catastrophic forgetting in continual learning scenarios. It ensures a seamless transition from short-term to long-term memory, preserving both temporal and spatial information throughout the learning process and enabling the system to adapt to new knowledge

without erasing previously acquired insight.

We evaluated 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). The aim of this work is determining which of these architectures exhibits optimal performance and gathering insights about their behavior during the learning process. The structure of the Quantum models are illustrated in Figures 4.7 4.8, 4.6 and 4.9.

Furthermore, we aimed to assess the effectiveness of this innovative model across various applications, particularly its ability to memorize and detect patterns in imbalanced datasets with limited data. This evaluation seeks to reduce execution times and improve convergence. To achieve this, we conducted experiments in the anomaly detection use case.

In this experiment, we performed 10 independent runs, each initialized with distinct random seeds to ensure the validity and robustness of our results. For each execution, we recorded the rates of true positives, true negatives, false positives and false negatives to calculate f1-score, along with the training and validation losses, enabling a comprehensive analysis of their respective curves.

For classical models, we utilized a larger dataset consisting of 390 fraud cases paired with 5000 non-fraud cases for training, and 999 instances of non-fraud paired with 101 instances of fraud for test dataset. In the case of quantum models, we utilised 390 fraud cases along with 1000 non-fraud cases for training dataset, mirroring the size of classical model's test dataset. This highlights that quantum models require fewer data points to achieve favorable metrics compared to classical models, showcasing their ability to discern non-trivial patterns that classical models struggle to identify, necessitating more data for effective learning. Lastly, we allocated 20% of training data for validation.

In this scenario, the training process differs slightly from the DRL use case. The model is trained in two phases: firstly, the QSNN is trained independently, without the influence of QLSTM, using the gradient descent method. Then, the model is split and joined to QLSTM for co-training. The loss from the output of QLSTM is utilized to simultaneously update both QSNN and QLSTM, as illustrated in Figure 4.5.

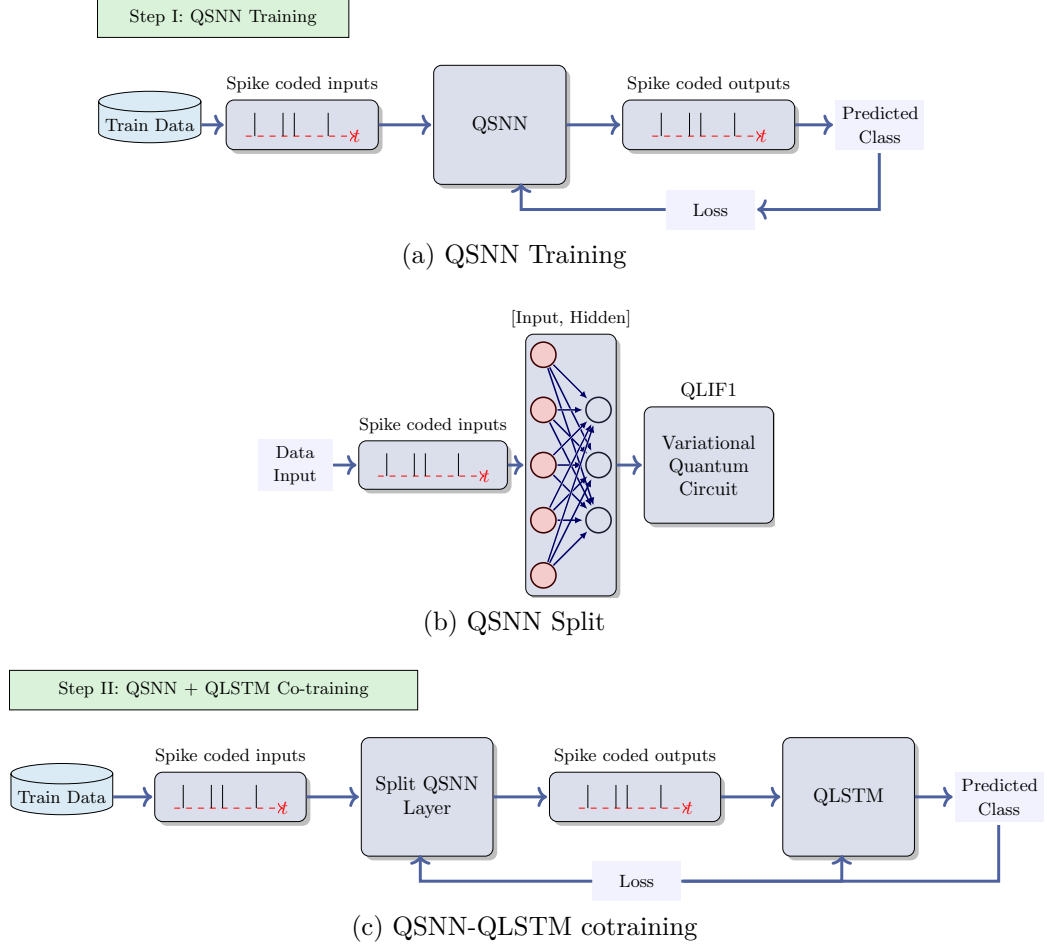


Figure 4.5: **Training Process of QSNN-QLSTM.** Step I involves the iterative training of QSNN. An original instance of QSNN (see Fig. 4.8) is trained independently without the influence of QLSTM, utilizing the gradient descent method. The model is then split (see Fig. 4.5b). In Step II, the split QSNN model is joined with a new instance of the QLSTM model for co-training. The loss from the output of QLSTM is used to simultaneously update both QSNN and QLSTM.

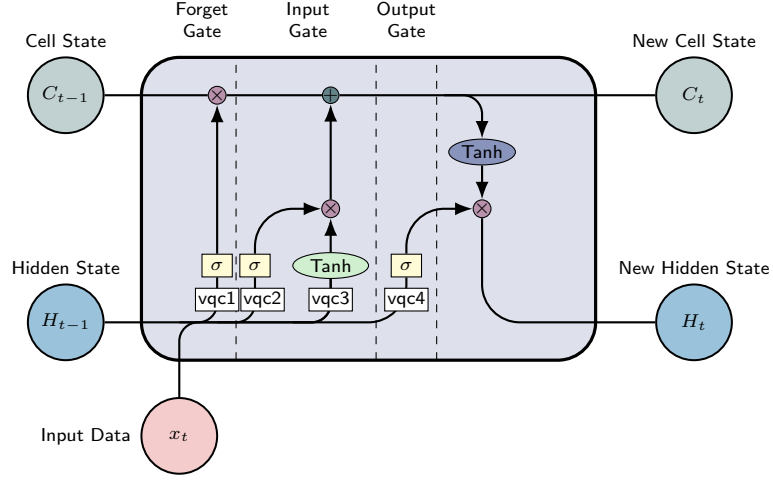
As a result of our investigation, we have produced two papers, the latter of which is currently undergoing peer review:

---

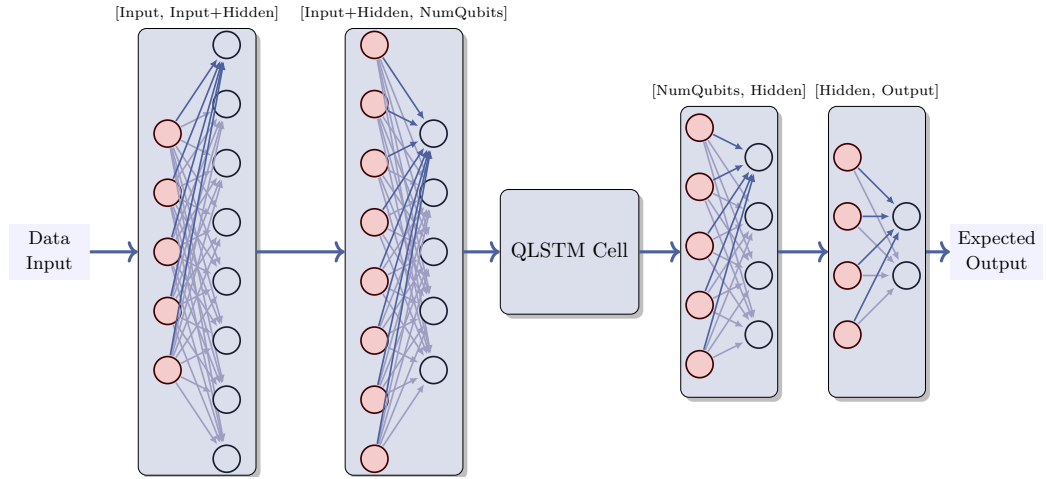
E. Andrés, M. P. Cuéllar, and G. Navarro, "Brain-inspired agents for Quantum Reinforcement Learning". Mathematics, vol. 12, no. 8, 2024.  
<https://www.mdpi.com/2227-7390/12/8/1230>. doi:10.3390/math12081230.

E. Andrés, M. P. Cuéllar, and G. Navarro, "Brain-inspired for enhanced pattern recognition: Unifying Quantum Spiking Neural Networks and QLSTM" (peer-review).

---



(a) QLSTMCell



(b) QLSTM Architecture: The input data undergoes initial processing through a classical layer, concatenating input and hidden dimensions. It then passes through a second classical layer to match the qubit count for the Quantum QLSTM cell. Afterward, another classical layer transforms it into the hidden size output, further processed to match the expected output.

Figure 4.6: QLSTM model

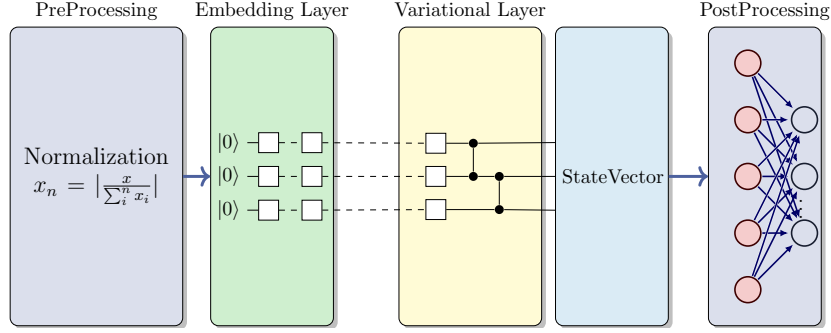


Figure 4.7: QNN architecture. Classical data normalization precedes amplitude encoding to generate a quantum state using  $\log_2 N$  qubits.

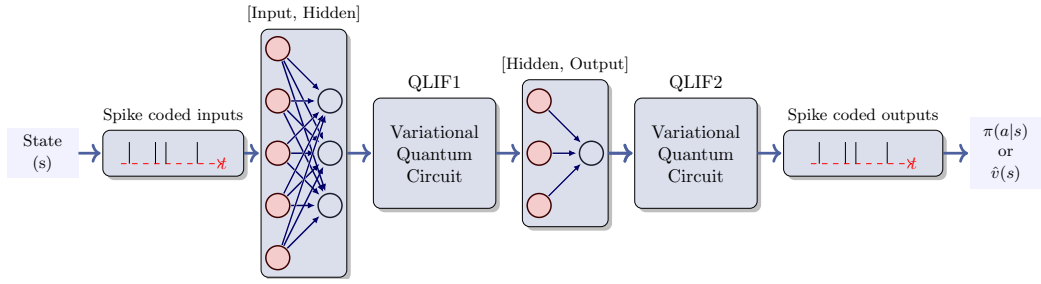


Figure 4.8: QSNN architecture. Classical data is encoded into spikes, processed through two LIFs implemented via VQC.

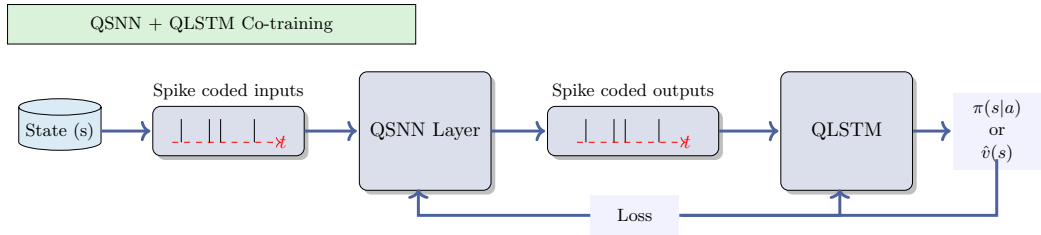


Figure 4.9: **Training process of QSNN-QLSTM.** An instance of QSNN is combined with a QLSTM instance for joint training. The loss computed from the output of QLSTM is utilized for the concurrent update of QSNN and QLSTM.

# Chapter 5

## Results

All outstanding work, in art as well as in science,  
results from immense zeal applied to a great idea.

---

*Santiago Ramón y Cajal*



## 5.1 Results

In this section we discuss the results obtained in the publications covered in this thesis.

### 5.1.1 QRL in the energy consumption optimization problem: benefits and limitations

After constructing the model outlined in Section 4.1.1, we conducted 10 executions for each experiment with different seeds for reproducibility purposes. Additionally, we defined metrics and figures of merit to enable a clear comparison between the quantum model and its classical counterpart. To achieve this, we recorded the total accumulated reward of the agent in each episode, the validation reward of the last episodes, and computational time. This allowed us to obtain the average, worst, and best accumulated reward, considering the 10 different runs performed. Additionally, we generated learning curves and boxplots to gain insights into its robustness and learning process.

A comprehensive analysis of the findings across three distinct use cases elucidates that the application of a Quantum Reinforcement Learning approach, demonstrates the potential for decision-making processes with reduced complexity compared to its classical counterpart. Notably, quantum models outperform classical MLP models across all investigated scenarios. However, they require more time to converge than conventional neural networks (particularly evident in use case 2) and higher computational time due to the use of quantum simulators instead of real quantum computers, even when the model complexity is lower than using MLP. Additionally, a limitation is that the proposal could not be tested in real contemporary NISQ quantum hardware due to the limitations in accessing such computers for large experiments and their inherent noise in measurement.

Furthermore, quantum agents exhibit greater stability during training, indicated by lower standard deviation in total rewards compared to the MLP model, suggesting a higher robustness of the quantum approach (especially in use cases 1 and 3), where MLP is more sensitive to initial conditions (weights, bias, and discount factor).

Finally, through this experimentation, we have not only compared classical and quantum models across three distinct use cases, understanding their benefits and limitations, but also established a methodology for comparing classical models with quantum ones.

### 5.1.2 The problem of scalability

The configuration of the experiments and the metrics used have been the same as those already explained in the first problem 5.1.1. Throughout the training process, all agents exhibited stability with lower standard deviation in terms of total reward. However, consistently across all use cases, the VQC agent employing amplitude encoding emerged as the top performer. Not only did this agent achieve the highest total reward, but it also demonstrated superior robustness compared to other agents. Its consistent outperformance highlights the effectiveness and versatility of Quantum Reinforcement Learning with amplitude encoding.

Additionally, we successfully addressed the Barren Plateau issue by implementing higher or adaptive learning rate techniques, enabling us to overcome the stagnation often encountered in flat regions (Barren Plateaus) of the optimization landscape. Furthermore, strategic selection of quantum circuits with optimal qubit counts played a crucial role in mitigating Barren Plateau-related convergence issues.

Our investigation revealed that careful circuit design with the appropriate number of qubits allowed us to circumvent severe instances of plateau-induced optimization difficulties. These solutions not only enhanced convergence but also established a basis for more stable and efficient Quantum Neural Network training.

Lastly, it is important to note that the inherent instability of the DQN algorithm in these use cases was mitigated using adaptive learning rates and/or a higher number of layers.

### 5.1.3 The problem of catastrophic forgetting

A notable observation from the results analysis is the effectiveness of utilizing quantum models inspired by the brain. This model emerged as the top-performer in terms of average, best total, worst and test reward (the latter removing the outlier), along with exhibiting the best learning curve and robustness.

Our investigation explored the integration of these theories into quantum agents, unveiling the potential benefits of infusing neuroscientific insights into Quantum Reinforcement Learning. 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.

As mentioned in Section 4.1.3, the brain-inspired model is used in a classification use case to gain further insight into its behavior in a different problem domain. We want to verify if it detects patterns that elude other models, thus improving the rates of false positives and negatives, even when dealing with large and imbalanced datasets. In this particular use case, the QSNN-QLSTM model delivered superior overall performance, requiring fewer iterations as an additional advantage. Furthermore, all models obtained a good evolution of learning with very similar curves, indicating that the selected architectures successfully captured the dataset’s patterns. Lastly, the quantum models had the advantage of working with less data compared to classical ones.

Note that, the configuration of the experiments and the metrics used have been the same as those already explained in the first problem 5.1.1, but using f1-score for the latest use case.

# Chapter 6

## Conclusions and future work

The mind that opens to a new idea never returns to  
its original size.

---

*Albert Einstein*



## 6.1 Conclusions and future work

This section summarizes the conclusions derived from the research conducted in this doctoral thesis and outlines potential avenues for future work.

Throughout this thesis, we have addressed the utilization of VQCs in the field of DRL. One of the objectives was to establish a methodological process for tackling such projects and enabling standardized comparisons among the proposed models. This objective has been achieved by following this methodology in all conducted studies.

Another aim was to explore and provide solutions to current limitations of Quantum Neural Networks. One of these limitations is the issue of Barren Plateaus. To address this problem, our experiences led us to employ various mechanisms, such as the use of higher or adaptive learning rates, reducing circuit expressiveness by employing fewer layers, reducing entanglement, and minimizing the number of qubits using compact circuits that effectively represent information without loss or the use of classical layers. These strategies have proven effective, as evidenced by the analyzed use cases. However, further exploration of new models and optimization methodologies remains a possibility in studying this phenomenon.

To address scalability challenges, we proposed the use of both classical layers and encoding strategies to minimize the required number of qubits. We tested these proposals in environments with a large number of variables (more than 20 environmental variables) and confirmed their effectiveness.

Furthermore, with the objective of reducing convergence times, filtering noise (which is also useful when executing on quantum computers and Barren Plateaus induced by noise), as well as handling infrequent events and memorizing decisions that maximize rewards, we developed a model that mimics the functionality of the hypothalamus and hippocampus, a QSNN-QLSTM composition. Here, the QSNN captures information at a low-level perspective, while the QLSTM processes this information at a higher level, identifying and memorizing correlated patterns while reinforcing long-term memorization. Thus, contributing to the state of the art with this model.

However, there remain several intriguing lines of research to explore. One such area is Online Learning (OL), where a single sample is provided to the learning algorithm at each time instant. There is a critical need for novel algorithms that can adapt swiftly to changes while maintaining high performance. OL, particularly in the presence of concept drift, has garnered substantial interest in recent years and continues spark active debate within

the research community [118, 119, 120]. Although this thesis does not delve into OL learning, demonstrating the adaptability of the proposed model to real-time situations could be a valuable pursuit. This would leverage the potential of Quantum Spiking Neural Network for real-time gradients calculation [73, 75], representing an interesting avenue for future research.

Additionally, further exploration of the intersection between Neuroscience, Quantum Computing and Artificial Intelligence, holds promise for mutual insights into understanding the physiological processes underlying human memory, particularly regarding what to retain and what to forget.

Other interesting challenges include generalization, adaptability to change, and navigation in uncertain environments, among others. Deep RL systems have yet to match human-level flexible adaptation through structured inference, drawing on a substantial repository of background knowledge. Whether 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 adapt to new situations systematically [121]. Nevertheless, this capability does not always manifest [122], and exploring the distinctions is of significance to both AI and neuroscience [123].

Additionally, an 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, future research could aim to advance an integrative theory and develop computational models capable of harmonizing various types of brain-inspired RL [10].

Another line of work would involve executing and comparing these proposed models on actual quantum computers.

Finally, besides exploring Variational Quantum Circuits (VQCs), future research could also investigate alternative variational algorithms that do not rely solely on VQCs. Developing and evaluating new variational algorithms that provide unique advantages or address current challenges in quantum machine learning tasks.

## 6.2 Conclusiones y trabajo futuro

Esta sección resume las conclusiones derivadas de la investigación realizada en esta tesis doctoral y describe posibles líneas de trabajo futuro.

A lo largo de esta tesis, hemos abordado la utilización de Circuitos Cuánticos Variacionales (VQCs) en el campo del aprendizaje profundo por refuerzo. Uno de los objetivos era establecer un proceso metodológico que facilitara la comparación entre los modelos propuestos. Una vez definida esta metodología, la aplicamos de manera consistente en todos los estudios subsiguientes.

Otro objetivo fue explorar y proporcionar soluciones a las limitaciones actuales de las redes neuronales cuánticas. Una de estas limitaciones es el problema de las mesetas estériles, para abordar este problema, nuestras experiencias nos llevaron a emplear varios mecanismos, como el uso de tasas de aprendizaje más altas o adaptativas, reducir la expresividad del circuito mediante el uso de menos capas, reducir el entrelazamiento y minimizar el número de qubits utilizando circuitos compactos que representen efectivamente la información sin pérdidas o el uso de capas clásicas. Estas estrategias han demostrado ser efectivas, como lo demuestran los casos de uso analizados. Sin embargo, la exploración adicional de nuevos modelos y metodologías de optimización sigue siendo una posibilidad en el estudio de este fenómeno.

Para abordar los desafíos de escalabilidad, propusimos el uso tanto de capas clásicas como de estrategias de codificación para minimizar el número requerido de qubits. Probamos estas propuestas en entornos con un gran número de variables (más de 20 variables de entorno) y confirmamos su efectividad.

Además, con el objetivo de reducir los tiempos de convergencia, filtrar el ruido (lo cual también es útil al ejecutar en ordenadores cuánticos), gestionar eventos infrecuentes y memorizar decisiones que maximizan las recompensas, desarrollamos un modelo que imita la funcionalidad del hipotálamo y el hipocampo: una composición QSNN-QLSTM. Aquí, el QSNN captura información a bajo nivel, mientras que el QLSTM procesa esta información a alto nivel, identificando y memorizando patrones correlacionados mientras refuerza la memorización a largo plazo. Este modelo representa una contribución significativa al estado del arte.

Sin embargo, aún quedan líneas de investigación interesantes por explorar. Una de ellas es el aprendizaje en línea (OL), donde se proporciona una sola muestra al algoritmo de aprendizaje en cada instante de tiempo. Existe una

necesidad crítica de algoritmos novedosos que puedan adaptarse rápidamente a los cambios mientras mantienen un alto rendimiento. El OL, especialmente en presencia de cambios conceptuales, ha suscitado un interés en los últimos años y continúa generando debate activo dentro de la comunidad científica [118, 119, 120]. Aunque esta tesis no profundiza en el aprendizaje en línea, demostrar la adaptabilidad del modelo propuesto a situaciones en tiempo real podría ser una búsqueda valiosa, aprovechando el potencial de las QSNN para el cálculo de gradientes en tiempo real [73, 75]. Por lo tanto, esto representa una línea de investigación futura interesante.

Explorar más a fondo la intersección entre Neurociencia, Computación Cuántica e Inteligencia Artificial abre la puerta a la promesa de obtener perspectivas mutuas para comprender mejor los procesos fisiológicos subyacentes en la memoria humana, especialmente en lo que respecta a qué retener y qué olvidar.

Otros desafíos interesantes incluyen la generalización, la adaptabilidad al cambio y la navegación en entornos inciertos, entre otros. Aunque, los sistemas de DRL aún tienen que igualar la adaptación flexible a nivel humano a través de la inferencia estructurada, basándose en un gran repositorio de conocimientos previos. Si estos sistemas pueden cerrar esta brecha sigue siendo una pregunta muy interesante y sin respuesta. La investigación reciente sugiere que, bajo ciertas condiciones, los sistemas Deep RL pueden aprovechar eficientemente el aprendizaje previo para adaptarse sistemáticamente a situaciones nuevas [121]. Sin embargo, esta capacidad no siempre se manifiesta [122] y explorar estas distinciones es de gran importancia tanto para la IA como para la neurociencia [123].

Además, persiste una pregunta sin resolver sobre cómo representar los intrincados mecanismos de interacción entre múltiples áreas cerebrales utilizando modelos de RL adecuados. Esta complejidad surge del hecho de que el proceso de toma de decisiones está distribuido en diversas regiones cerebrales, que pueden cambiar dinámicamente dependiendo de las demandas específicas de la tarea. Además, actualmente falta una teoría matemática capaz de explicar todos los descubrimientos significativos en el aprendizaje cerebral humano. Por lo tanto, la investigación futura podría aspirar a avanzar en una teoría integradora y desarrollar modelos computacionales capaces de armonizar diversos tipos de RL inspirados en el cerebro [10].

Otra posible línea de trabajo potencial implicaría la ejecución y comparación de los modelos propuestos en ordenadores cuánticos reales.

Finalmente, además de explorar los Circuitos Cuánticos Variacionales

(VQCs), la investigación futura también podría investigar algoritmos variacionales alternativos que no dependan únicamente de los VQCs. Se podrán desarrollar y evaluar nuevos algoritmos variacionales que ofrezcan ventajas únicas o aborden los desafíos actuales en las tareas de aprendizaje automático cuántico.

# Chapter 7

## Publications

Acercaos al abismo, les dijo. Tenemos miedo,  
respondieron. Acercaos al abismo, les dijo.  
Se acercaron.  
Él los empujó...  
y salieron volando.

---

*Apollinaire*



## 7.1 On the Use of Quantum Reinforcement Learning in Energy Efficiency Scenarios

|                                                                                                                                                                                                                                                |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Reference</b>                                                                                                                                                                                                                               |
| E. Andrés, M. P. Cuéllar, and G. Navarro.<br>On the Use of Quantum Reinforcement Learning in Energy-Efficiency Scenarios.<br>Energies 2022, 15, 6034.<br><a href="https://doi.org/10.3390/en15166034">https://doi.org/10.3390/en15166034</a> . |
| <b>Status</b>                                                                                                                                                                                                                                  |
| Published                                                                                                                                                                                                                                      |
| <b>Impact factor</b>                                                                                                                                                                                                                           |
| IF: 3.2<br>H-Index: 152                                                                                                                                                                                                                        |
| <b>Category</b>                                                                                                                                                                                                                                |
| Engineering (Q1); Energy Engineering and Power Technology (Q2); Energy (Q2), Control and Optimization (Q2)<br>SJR: 0.651                                                                                                                       |
| <b>DOI</b>                                                                                                                                                                                                                                     |
| 10.3390/en15166034                                                                                                                                                                                                                             |
| <b>Journal-Editorial</b>                                                                                                                                                                                                                       |
| Energies-MDPI                                                                                                                                                                                                                                  |
| <b>Logo</b>                                                                                                                                                                                                                                    |
|  <b><i>energies</i></b><br>an Open Access Journal by MDPI                                                                                                   |

## *On the Use of Quantum Reinforcement Learning in Energy Efficiency Scenarios*

**Abstract:**In the last few years, Deep Reinforcement Learning has been proposed as a method to perform Online Learning in energy-efficiency scenarios such as HVAC control, electric car energy management, or building energy management, just to mention a few. On the other hand, quantum machine learning was born during the last decade to extend classic machine learning to a quantum level. In this work, we propose to study the benefits and limitations of Quantum Reinforcement Learning to solve energy-efficiency scenarios. As a testbed, we use existing energy-efficiency-based Reinforcement Learning simulators and compare classic algorithms with the quantum proposal. Results in HVAC control, electric vehicle fuel consumption, and profit optimization of electrical charging stations applications suggest that Quantum Neural Networks are able to solve problems in Reinforcement Learning scenarios with better accuracy than their classical counterpart, obtaining a better cumulative reward with fewer parameters to be learned.

**keyword**Quantum Neural Networks; Variational Quantum Circuits; Quantum Reinforcement Learning; Energy Efficiency.

### 7.1.1 Introduction

Reinforcement Learning (RL) [36] is a type of machine learning inspired by behavioral psychology. It provides the fundamental basis to describe how intelligent agents learn autonomously to take actions in unknown environments in order to maximize the notion of cumulative reward. Deep Reinforcement Learning (DRL) [36] integrates RL algorithms together with neural networks, and this combination has attracted great attention in the scientific literature and has been growing at an astounding speed in the last decade, due to its success in solving massively complex problems such as games (chess, Go [3], and Atari games [4]), robotics [5], or autonomous driving [6], just to mention a few. In the last few years, it has also attracted the attention of researchers in energy-efficiency methods such as [124, 125], due to its ability to perform Online Learning and the capacity of adaptation to new changes in an environment. Neural network models used in DRL range from the

MultiLayer Perceptron or radial basis function networks, to more complex recurrent models such as Elman, Jordan, LSTM, or GRU networks, as the most sound examples.

With regards to the state of the art of RL for energy efficiency and energy management, there is a wide range of approaches, such as the efficient allocation/deallocation of the resources in response to the workload variations [47, 48] or task scheduling [49, 50], providing predictive and efficient routing schemes [51], or minimizing resources power usage [52]. Raman et al. [53] developed the Zap Q-learning algorithm, aimed at climate control of a commercial building. Apart from the temperature comfort, the proposal attempted to minimize energy consumption while maintaining comfort regarding humidity. As another example, Wei et al. [126] used deep Q-learning to control building automation control systems. The DQN algorithm was compared with a rule-based system and a Q-learning algorithm. Their contribution proved to be able to maintain temperature comfort in the desired range while reducing the energy consumption. Lissa et al. [127], using the same neural network model, developed a methodology to find the bounds of indoor temperature comfort, and their proposal was able to save energy more efficiently than a rule-based system, while comfort was not affected. Finally, Wang et al. [54] proposed a Reinforcement Learning controller based on actor/critic models to building HVAC control, using Recurrent Neural Networks (LSTM). Experiments were performed on a simulated environment containing a digital twin of an office building, and their results were able to save energy while maintaining the user comfort. A complete survey about RL applied for energy efficiency in buildings can be found in [55]. Despite the success of DRL that has been reported in the literature in different energy-efficiency and management scenarios, the main limitation of RL methods lies on the fact that, to be able to learn a correct behavior, the agent must interact with the environment, making good actions but also making failures until the optimal operation is learned. For this reason, researchers rely on the development of simulation environments that implement digital twins of real devices, which in some cases use historical data also. Simulation tools such as Energy Plus [128] or Modelica [129] play a fundamental role in the training and testing of models before deploying them in real energy systems. RL environments are built on top of this simulation software, for example, Sinergym [130], Gym-EPlus [131], or EnerGym [132] to model energy consumption and control in buildings, the simulation of energy management in electrical vehicles (EVs) [133, 134], or the simulation of EV charging stations [135]. Different environments are described and analyzed in

[130].

Contemporaneously with the advances in DRL, Quantum Computing (QC) [26] and quantum machine learning (QML) [28, 136] in particular, have evolved rapidly as research areas thanks to the advances in quantum hardware and the creation of new quantum hardware simulators such as Google’s Cirq, IBM’s Qiskit, Xanadu’s PennyLane, etc. The concept of Quantum Computing was originally presented in 1982 by Richard Feynman [14] following an observation of the exponential complexity involved in modeling the behavior of a quantum system with the existing knowledge of classical computing. It is based on quantum mechanics properties such as superposition and entanglement, which also enable quantum parallelism. The promise of QC in general, and QML in particular, is a huge speedup to obtain solutions to complex problems that could not be solved in reasonable time using classic computers. For example, early theoretical works demonstrated the massive potential of quantum algorithms solving complex problems more efficiently than their classical counterparts, such as Grover’s search [12], or integer factorization using Shor’s algorithm [13].

On the other hand, quantum machine learning is the intersection of machine learning and Quantum Computing, and it attempts to create machine learning models able to run in quantum computers with the final goal of achieving a significant speedup in their execution, or solving more complex problems with fewer resources than their classical counterpart. Quantum Reinforcement Learning (QRL) is a very recent research field, and the lack of worldwide access to a real quantum hardware able to run the learning process forces researchers to use quantum computer simulators running in classical computers. We believe that, for that reason, the existing QRL research works in the literature, such as [15, 19, 56, 57], focus their experimentation on well-tested simple environments, mainly from the OpenAI Gym RL platform [58]. Despite these limitations, the existing QRL approaches have provided relevant insights into how RL can be adopted by a quantum system, such as the use of Grover iterations [15] to provide a quantum implementation of the classic Q-learning algorithm, or Variational Quantum Circuits (VQC) [16] as quantum agents in deep Q-learning or policy gradient methods [19, 56, 57]. The VQC is a class of hybrid classical-quantum algorithm based on a quantum circuit with learnable parameters optimized using classical algorithms so that, in the case of QRL, VQCs are used as function approximators for the agents’ policy. The trainable parameters would be optimized using gradient or gradient-free methods.

In this manuscript, we apply QRL to solve problems in the field of energy efficiency and energy management using quantum Artificial Neural Networks. We hypothesize that QRL technologies could solve the problems using a more compact model than their classical counterparts thanks to inherent nonlinear data management and quantum parallelization. To the best of our knowledge, this article is the first attempt to address energy-efficiency problems with quantum technologies. For that reason, we study different types of use cases where classical RL has been successfully applied before: the Sinergym [130] project, which is a simulator backed up by EnergyPlus and contains reference problems of energy efficiency and HVAC control in buildings and facilities; the Prius [133] simulator of electrical vehicles, aimed at reducing the fuel and battery usage of electrical vehicles with the Toyota Prius car as testbed; and a simulator of electrical vehicle charging stations [135] whose objective is to maximize the station’s profit in a real scenario, considering the workload and the price of energy. Thus, the main contributions of the article can be summarized as follows: (1) the proposal of QRL methodology for energy-efficiency and management scenarios; (2) the design of hybrid quantum/classical agents to solve QRL energy-related environments; and (3) a wide experimental study with an analysis of pros and cons of contemporary QRL technologies in the field of energy efficiency and management.

The remainder of the manuscript is structured as follows: Section 7.1.2 provides a background in Reinforcement Learning and Quantum Computing. After that, Section 7.1.3 explains the approach to build hybrid quantum/-classical QRL agents. Section 7.1.4 describes the experimentation and the analysis of results, and finally Section 7.1.6 provides conclusions.

## 7.1.2 Background

For article self-completeness and notation description, this section summarizes the main concepts required to describe the proposal of the manuscript in Section 7.1.3. Firstly, Section 7.1.2 describes the fundamentals of Artificial Neural Networks. Then, Section 7.1.2 introduces the field of Reinforcement Learning and its integration with (deep) Artificial Neural Networks. Finally, Section 7.1.2 provides an introduction to Quantum Computing and quantum variational circuits methods.

## Artificial Neural Networks

Artificial Neural Networks [137] are computational models inspired by connectionist models of human and animal brains. A *feedforward* neural network is formalized as a weighted, non-cyclic, directed graph where nodes represent computation units called *neurons*, and edges are *synapses*. The most known neural network model is the *MultiLayer Perceptron* (MLP), where neurons are stacked in layers, and its graphical representation is shown in Figure 7.1.

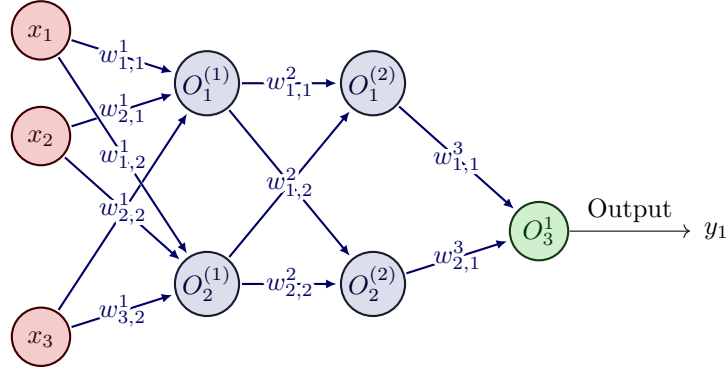


Figure 7.1: Example of a MultiLayer Perceptron with 3 inputs, 2 layers with 2 neurons each, and 1 output.

The dynamic of an MLP is as follows: Assuming an input data pattern  $O^0 = X$  composed of  $N_0$  values,  $X = (x_1, x_2, \dots, x_{N_0})$ , for which an output response  $Y = (y_1, y_2, \dots)$  is expected, each neuron  $i$  at layer  $l$  calculates its output  $O_i^l$  as a function  $f$  of the weighted sum of its inputs (Equation (7.1)), where  $w_{ji} \in \mathbb{R}$  represents the synaptic strength (weight) of the connection between output of neuron  $j$  at layer  $l - 1$  to neuron  $i$  at layer  $l$ ;  $b_i^l \in \mathbb{R}$  is a neuron intercept also called *bias*, and  $f$  is the neuron activation function. The network output  $Y$  corresponds to the returned values of neurons located at the last layer. Usual activation functions are sigmoid, hyperbolic tangent, rectified linear units (ReLUs), etc. In this work, we use MLP as baseline models to compare Quantum Neural Networks behavior in the experimentation. The ReLU activation function (Equation (7.2)) is used as activation function.

$$O_i^l = f\left(\sum_{j=1}^{N_{l-1}} w_{ji}^l O_j^{l-1} + b_i^l\right) \quad (7.1)$$

$$f(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases} \quad (7.2)$$

Training a neural network encompasses finding the correct values for weights  $w_{ij}^l$  and biases  $b_i^l$  so that the network output  $\tilde{Y}$  matches a desired output  $Y$ . This problem is usually formulated as a minimization problem over a defined cost function, also called *loss*, for instance, in Equation (7.3), where  $Y(p)$  is the desired output for a given input pattern  $X(p)$ , and  $\tilde{Y}(p)$  is the output provided by the network for pattern  $X(p)$ . Different algorithms such as backpropagation, gradient descent, ADAM, etc., are used to optimize both weights and biases. For more information about neural networks, we believe that the reference [137] is a good, detailed book for both novices and experts in the field.

$$[w_{ij}^{l*}, b_i^{l*}] = \underset{\{w_{ij}^l, b_i^l\}}{\operatorname{argmin}} \left\{ \sum_p (\tilde{Y}(p) - Y(p)) \right\} \quad (7.3)$$

## Reinforcement Learning

The main model of RL encompasses two main components: an environment (whose internal dynamics are unknown) and an intelligent agent that interacts with the environment over time. At any time instant, the environment is at an unknown state, but the agent can *observe* the environment to obtain some information about such state, e.g., using sensors. For the sake of simplicity, in this manuscript we name as  $s_t$  the environment observation perceived by the agent at time  $t$ . Then, the agent selects an action  $a_t$  from an available action set considering the observed state  $s_t$ , and performs the action over the environment. After that, the environment evolves from state  $s_t$  to a state  $s_{t+1}$  as a consequence of the agent–environment interaction, and returns  $s_{t+1}$  and a reward  $r_{t+1}$  to the agent. This reward is a scalar value that informs the agent about how good/bad the selection was of action  $a_t$  conditioned to state  $s_t$ . This process is repeated in a loop indefinitely (Figure 7.2).

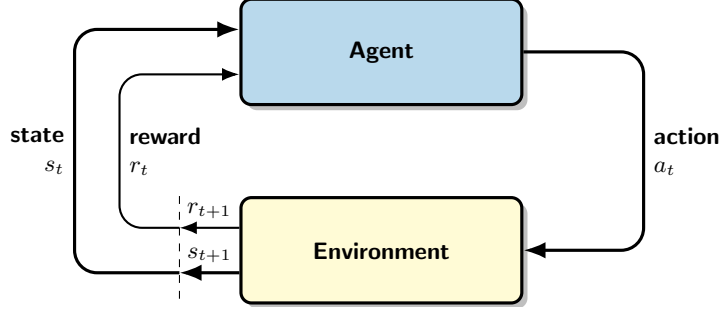


Figure 7.2: General Reinforcement Learning diagram

The process of sequential agent–environment interactions (also known as trajectories or episodes) involving state transitions, actions, and rewards in the RL problem can be formulated as a Markov Decision Process (MDP). An MDP is characterized by the tuple  $\langle S, A, P, r \rangle$ , where  $S$  is the set of states,  $A$  is the set of actions,  $P$  is the probability of state transition  $P(s'|s, a) = P[s_{t+1} = s' | s_t = s, a_t = a]$ ; i.e., the probability of transitioning from the state  $s_t$  at time  $t$  to state  $s_{t+1}$  at time  $t + 1$  using action  $a_t$  selected at time  $t$ , and  $r(s_t, a_t, s_{t+1})$  is the reward function for executing action  $a_t$  at state  $s_t$  and then performing a transition from  $s_t$  to  $s_{t+1}$ .

The goal of an agent in RL is to maximize the total cumulative reward, or *return*, of a sequence of agent–environment interactions  $\tau$  starting at time  $t_0$  defined in Equation (7.4), where  $\gamma$  is a hyperparameter to establish how important recent or older rewards are for the learning, known as *discount factor*.

$$R(\tau) = \sum_{t=t_0}^{\infty} r_t \gamma^{t-t_0} \quad (7.4)$$

In order to maximize  $R(\tau)$ , it is necessary that the agent learns the best action  $a$  to be performed at a given state  $s$ , i.e., the *policy*  $\pi(a|s)$ , which is the probability to select action  $a$  in state  $s$ . Two fundamental concepts arise from this problem formulation: the value of a state–action pair  $Q(s, a)$  in Equation (7.5) (the expected return obtained in the trajectory starting from state  $s$  and action  $a$ ), and the value of a state  $V(s)$  in Equation (7.6) (the expected return obtained in the trajectory starting from state  $s$ ). A third relevant concept is the advantage of the state–actionpair  $Adv(s, a)$  (Equation (7.7)), which returns the advantage of choosing action  $a$  in state  $s$  with respect

to the other actions available in the action set for the same state  $s$ .

$$Q(s, a) = \mathbb{E}_{\tau \sim \pi} (R(\tau) | s_t = s, a_t = a) = \sum_{s'} p(s' | s, a) \left( r(s, a, s') + \gamma \sum_{a'} \pi(a' | s') Q(s', a') \right) \quad (7.5)$$

$$V(s) = \mathbb{E}_{\tau \sim \pi} (R(\tau) | s_t = s) = \sum_i R(\tau_i) \pi(a_i | s) \quad (7.6)$$

$$Adv(s, a) = Q(s, a) - V(s) \quad (7.7)$$

Deep Reinforcement Learning attempts to use a (deep) Artificial Neural Network to learn the optimal policy  $\pi(a|s)$ , having the state  $s$  as input and, as output, a selected action  $a$  (deterministic policy), or the probability distribution of selection of action  $a$  in the state  $s$  (stochastic policy). In the literature, two main families of algorithms have been highlighted in the last few years: Deep Q-Networks (DQN) [38] and policy gradient [39]. The former family attempts to train an artificial neural network to approximate the function  $Q(s, a)$ , while the latter attempts to approximate  $\pi(a|s)$  directly. DQN training is inspired by the classic Q-learning method, and attempts to minimize the loss function in Equation (7.8), where  $\hat{Q}(s, a)$  is the  $a$ -th output value of the neural network for the input  $s$ . A Deep Q-Network contains as many input neurons as the dimension of the state, and as many output neurons as actions available in the action set.

On the other hand, actor-critic policy gradient models require at least two types of neural networks for training: one type of network (actor) has a similar structure to a DQN, but the  $a$ -th network output is designed to return  $\pi(a|s)$ . The second type of network attempts to return an approximation of  $V(s)$ , so that it contains the same number of inputs as the actor, and a single output value.

$$MSE = \sum_t \left( r(s_t, a_t, s_{t+1}) + \gamma \max_{a_{t+1}} [\hat{Q}(s_{t+1}, a_{t+1})] - \hat{Q}(s_t, a_t) \right)^2 \quad (7.8)$$

Different strategies have been devised to improve DQN training and policy gradient methods and, for further reading, we refer the reader to references [38, 39]. In this work, we use the Advantage Actor-Critic (A2C) training algorithm [39], whose designed loss function is described in Equation (7.9).

$$LogLoss = - \sum_t Adv(s_t, a_t) \log \pi(a_t | s_t) \quad (7.9)$$

## Quantum Computing and Variational Quantum Circuits

The most basic information unit in classic computing is the bit, and the underlying mathematical model to work with bits is  $\mathbb{Z}_2$ , the field with two elements. Unlike the classic model, the fundamental information unit in Quantum Computing is the qubit, and the underlying mathematical model is the complex plane  $\mathbb{C}^2$ , where the standard orthonormal basis vectors are usually denoted by  $|0\rangle$  and  $|1\rangle$ . Hence, the value (state) of a qubit  $|\psi\rangle$  is a linear combination of these vectors:

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle; \alpha_0, \alpha_1 \in \mathbb{C} \quad (7.10)$$

with the constraint that  $|\alpha_0|^2 + |\alpha_1|^2 = 1$ , since Born's rule tells us that if this qubit is measured, we will obtain  $|0\rangle$  with probability  $|\alpha_0|^2$  and  $|1\rangle$  with probability  $|\alpha_1|^2$ .

When a new bit is included into a classical system, the underlying state space increases by one dimension, since the resultant space is the Cartesian product of the older one by  $\mathbb{Z}_2$ . Thus, in  $n$ -bit systems, the state space is  $\times^n \mathbb{Z}_2 = \mathbb{Z}_2^n$ . On the other hand, when a new qubit is added to a quantum system, the underlying space doubles its dimension, since the resultant space is the tensor product of the older one by  $\mathbb{C}^2$ , yielding an exponential growth of the dimension. In an  $n$ -qubit system, the underlying model is  $\mathbb{C}^{2^n} = \bigotimes_n \mathbb{C}^2$ , i.e., the tensor product of  $\mathbb{C}^2$ ,  $n$  times. For instance, the  $\mathbb{C}$ -vector space  $\mathbb{C}^{2^2}$  has a basis formed by the vectors  $|0\rangle \otimes |0\rangle$ ,  $|0\rangle \otimes |1\rangle$ ,  $|1\rangle \otimes |0\rangle$ , and  $|1\rangle \otimes |1\rangle$ , which, for simplicity, are denoted by  $|00\rangle$ ,  $|01\rangle$ ,  $|10\rangle$ , and  $|11\rangle$ , respectively.

Classical computers manipulate bits using logic gates such as AND, OR, NOT, NAND, XOR, etc. Likewise, quantum computers manipulate qubits using quantum gates. A quantum gate is modeled as a unitary matrix that multiplies the system qubit state, and all quantum operations must be reversible (except measurement). Some examples of quantum gates are the Pauli-X ( $\sigma_x$ ), Pauli-Y ( $\sigma_y$ ), and Pauli-Z ( $\sigma_z$ ) single qubit rotations with a phase of  $\pi$ , or their generalization rotation gates  $R_x(\theta)$ ,  $R_y(\theta)$ , and  $R_z(\theta)$  that rotate a qubit in the specified axis by the given angle  $\theta$ . Another relevant quantum gate for the proposal described in this work is the CNOT gate over two qubits. The unitary matrices of the gates  $R_x(\theta)$ ,  $R_y(\theta)$ , and  $R_z(\theta)$ , CNOT

are shown in Equations (7.11), (7.12), (7.13), and (7.14), respectively.

$$R_x(\theta) = \begin{pmatrix} \cos(\frac{\theta}{2}) & -i\sin(\frac{\theta}{2}) \\ -i\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix} \quad (7.11)$$

$$R_y(\theta) = \begin{pmatrix} \cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix} \quad (7.12)$$

$$R_z(\theta) = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix} \quad (7.13)$$

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (7.14)$$

A *quantum program*, implemented as a *quantum circuit*, is a sequence of quantum gates applied over one or more qubits. An example of a graphical representation of a quantum circuit with two qubits is shown in Figure 7.3: First, two single-qubit Hadamard (H) quantum gates are applied separately over qubits  $q_0$  and  $q_1$ . Then, a two-qubit SWAP quantum gate is applied on both qubits, followed by a CNOT gate over  $q_1$  controlled by  $q_0$ . Finally, both qubits are measured to obtain the result, which can be any quantum state in  $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ .

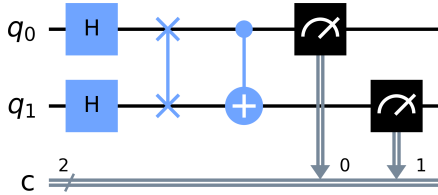


Figure 7.3: Example of a quantum circuit with two qubits.

A Variational Quantum Circuit is a quantum circuit containing parameterized gates, such as the aforementioned  $R_x, R_y, R_z$  gates with parameter  $\theta$ . Since VQCs are universal approximators and are able to approximate any continuous function similar to a classical neural network [21, 19], they

are also known in the quantum machine learning community as Quantum Neural Networks. An example of a VQC is shown in Figure 7.4, where three rotations  $R_x, R_y, R_z$  gates are applied over each qubit (12 total parameters), and then different non-parameterized gates transfer the qubit information. If the reader is interested in extending the fundamentals of QC and QML, we suggest the references [26, 28].

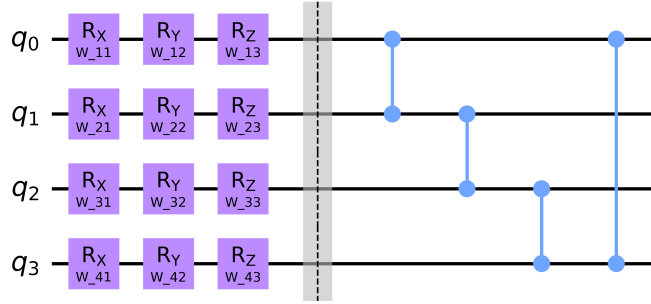


Figure 7.4: Example of a Variational Quantum Circuit with 4 qubits.

The output of a quantum system is obtained using the *measurement* operator, which makes a qubit collapse to a basis vector  $|0\rangle$  or  $|1\rangle$ . In Quantum Computing hardware it is required to measure outputs with regards to an *observable*, i.e., quantum properties that can be observed. It is also common to measure expectations of these observables [138], and this last technique is the one we use in our experimentation. The expectation of an observable whose unitary matrix is  $U$ , over a quantum state  $|\phi\rangle$ , is denoted as  $\langle\phi|U|\phi\rangle$ . In our case, we use the  $Z$  observable whose unitary matrix  $\sigma_z$  is the Pauli-Z shown in Equation (7.15). As an example, if the quantum state is  $|\phi\rangle = |0\rangle$ , then the expectation of the observable returns value 1, and if the quantum state is  $|\phi\rangle = |1\rangle$ , then the expectation of the observable returns value  $-1$ . Any other intermediate value in  $[-1, 1]$  is linked to the probabilities  $\alpha_0$  and  $\alpha_1$  described previously.

$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (7.15)$$

### 7.1.3 Methodology

We assume a QRL scenario of a hybrid quantum–classical agent with a set of parameters  $\theta = (\theta_1, \dots, \theta_k)$  that learns by interacting with a classical (non-quantum) environment, as shown in Figure 7.5. In the beginning of an episode, the quantum–classical hybrid agent receives a state information from the environment, determines its action from its policy  $\pi_\theta(a|s)$ , and perceives a reward  $r$  and the next environment state after applying the action to the environment. The policy  $\pi_\theta$  is given by a VQC running on a Quantum Processing Unit (QPU). The optimization of this policy is carried out by updating the parameters  $\theta$ , which is performed by a classical learning algorithm in a Central Processing Unit (CPU) with the objective of reducing the cost function [94]. In our work, the cost function is described in Equation (7.9) for A2C (actor network) and Equation (7.16) for the critic network, where  $V^\tau(s)$  is the true value function of state  $s$  in a given trajectory  $\tau$ , and  $\hat{V}^\tau(s)$  is the value network output.

$$MSELoss = \sum_{\tau} \sum_{s \in \tau} \left( \hat{V}^\tau(s) - V^\tau(s) \right)^2 \quad (7.16)$$

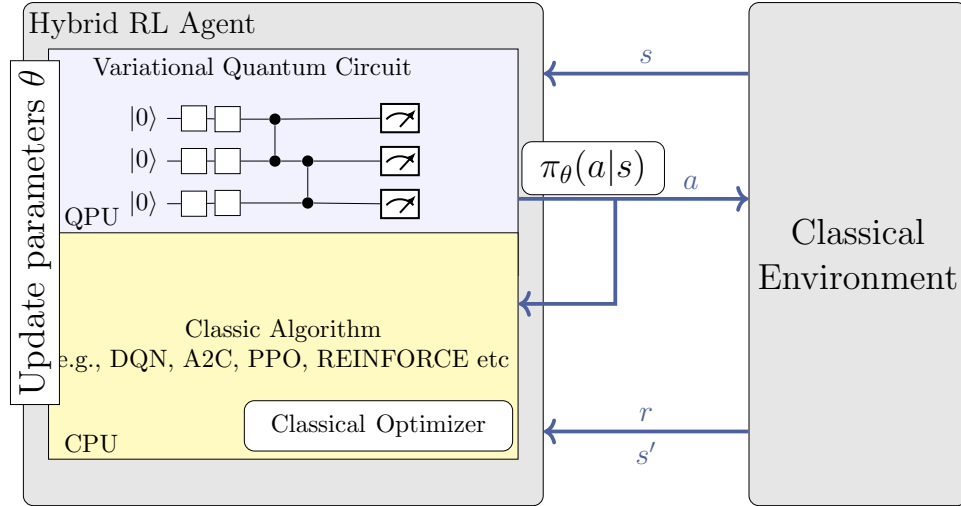


Figure 7.5: QRL schema. Example of hybrid agent and classical/quantum interaction to optimize the trainable parameters of the policy using a classical learning algorithm.

The general hybrid methodology to train a quantum agent in QRL is illustrated in Figure 7.6, and encompasses the following steps:

1. **Preprocessing (CPU)**: This step has the environment state  $s$  as input, and performs data preprocessing to prepare classical data before encoding the environment state into a quantum state. This preprocessing could be a simple operation of translation and/or change of scale (Figure 7.7), or more complex data processing such as a change of space (projection, injection) with a classical (non-quantum) neural network layer (Figure 7.8).
2. **Quantum Embedding (QPU)**: The objective of the encoding layer, also named quantum embedding, is to map the classic data to the qubits' Hilbert space to be manipulated afterward. This layer is composed of rotation and entanglements gates, and the tensor product and the entanglement generated in those encoding circuits capture the existing nonlinearities of the data. The information of the environment state  $s$ , after the previous preprocessing step, must be encoded into a quantum state  $|s\rangle$  by means of a quantum circuit with the appropriate sequence of quantum gates. In this work, we use the tensor product encoding technique [29] using  $R_x(\theta)$  rotation gates, so that each feature of state  $s$  is matched with a qubit. The value of the feature is the rotation angle of the  $R_x$  gate. Thus, we must ensure that all features in  $s$  are in the range  $[-\pi/2, \pi/2]$ , which is the domain of the parameter of  $R_x(\theta)$ .
3. **Variational Layer (QPU)**: The quantum state  $|s\rangle$  is the input of the VQC in charge of modeling  $\pi_\theta(a|s)$  as a quantum state  $|\phi\rangle$ . Usually, VQCs are composed of entanglement and rotation gates parameterized with learnable parameters  $\theta$ . A classic optimization algorithm will optimize these parameters, aiming at minimizing a given cost function. The choice of the encoding strategy, as well as the construction of the VQC, are crucial to obtain optimal results. In this work we use the same VQC in all experiments, and it is inspired by previous works [56, 94].
4. **Measurement Process (QPU/CPU)**: In this final step, the quantum state provided by the VQC is measured and decoded to obtain the desired output. The critical point in the quantum measurement is to find an optimal way to associate outputs of the observations with target classes. The selection of the observables used to read out information

from the quantum model is crucial to achieve a good performance. In this work, we calculate as output the expectation of the  $\sigma_z$  observable, i.e., the operation  $\langle \phi | \sigma_z | \phi \rangle$ , for each qubit. The value of this observable is in the range  $[-1, +1]$ , where the bounds  $+1$  means that ket  $|0\rangle$  is always returned, and  $-1$  means that ket  $|1\rangle$  is always returned as output.

5. **Postprocessing (CPU)**. This step gathers the outputs returned by the QPU, i.e., the policy  $\pi_\theta(a|s)$ , and performs post-processing operations over the outputs, if necessary.
6. **Learning (CPU)**. This last step computes the loss function of the VQC to optimize its parameters  $\theta$ . In general QRL approaches, and also in this work, this is performed by a traditional optimization algorithm such as Adam, RMSProp, etc.

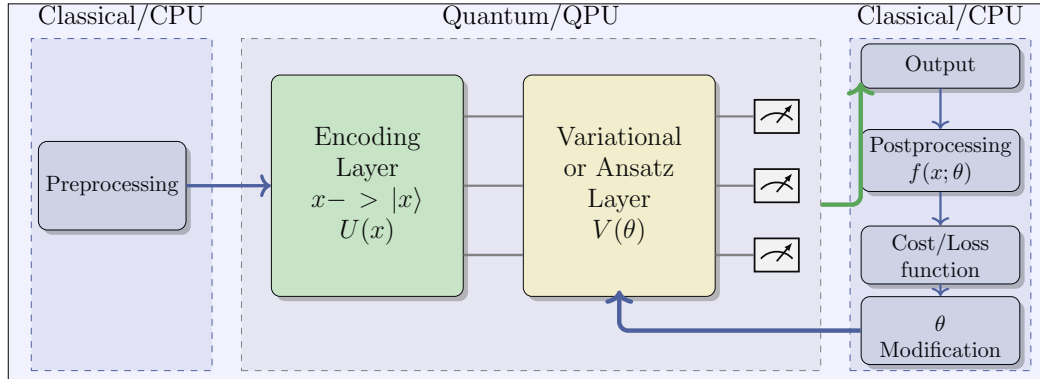


Figure 7.6: General hybrid quantum-classical agent structure.

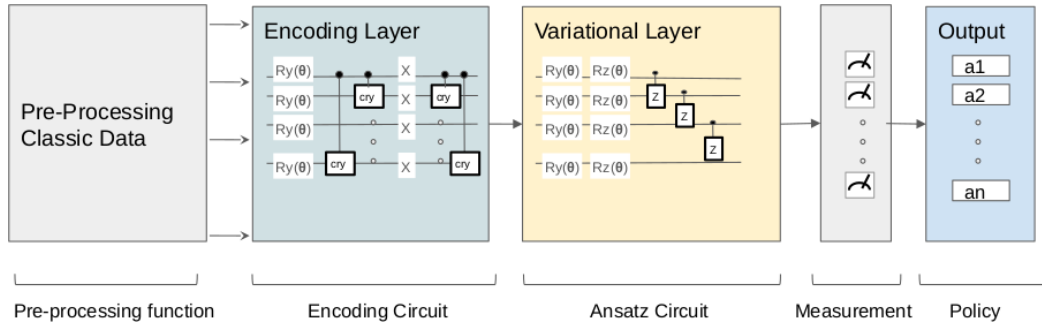


Figure 7.7: Preprocessing with input data manipulation. The output of preprocessing is fed to the encoding circuit and then to the variational layers.

As happens in classical neural networks, VQCs also suffer from the gradient vanishing phenomenon, also known as Barren Plateau [16, 97]. In general, a VQC is able to solve a problem using fewer parameters than its classical counterpart, although there is an inflection point at which the excessive number of qubits, entanglements, or layers used in the construction of the circuit makes this phenomenon more prone to happen. For the case study of Section 7.1.4, we selected a circuit made up of  $R_y, R_z$  rotation gates, followed by entanglement gates similar to circuits used in other previous QRL algorithms that provided promising results in simple RL problems [56, 94]. Figure 7.9 shows an example of the VQC structure used in this work for a quantum program containing 7 qubits. A multilayer VQC is generated in our experiments by means of creating a sequence of VQCs with this structure. Thus, we may see that the number of learnable parameters in a VQC increases linearly, which is an advantage when compared with classical (non-quantum) neural networks.

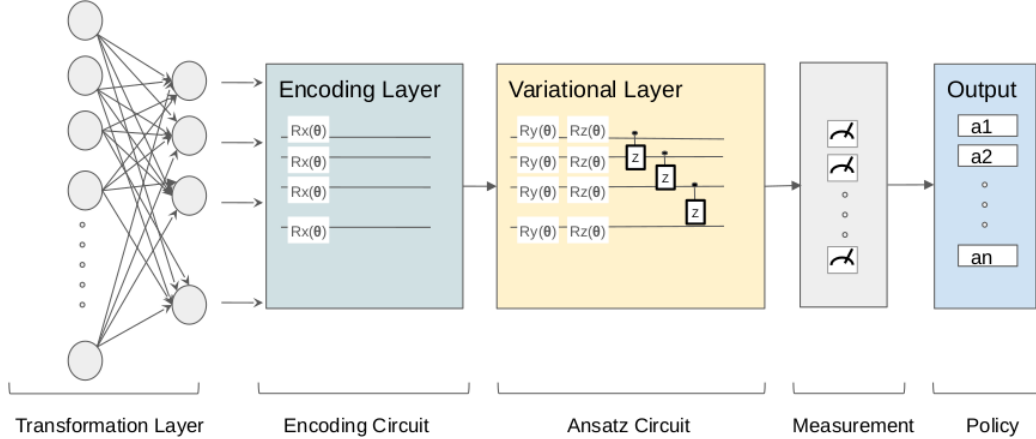


Figure 7.8: Preprocessing with a hybrid classical/Quantum Neural Network. The output of preprocessing is fed to the encoding circuit and then to the variational layers.



Figure 7.9: Example of a 7-qubit VQC with the structure used in the experiments. Left: encoding layer; middle: variational gates; right: entanglement CNOT gates organized in a ring.

### 7.1.4 Experiments

In this section, we test Quantum Reinforcement Learning algorithms in different scenarios related to energy efficiency, and compare the results with their corresponding classical counterparts. The goal is to verify if QRL and quantum technologies are appropriate in the field of energy efficiency. To that end, three different use cases with different constraints and nature are used. Section 7.1.4 addresses a use case of classic control of HVAC energy-saving in buildings; Section 7.1.4 solves an environment regarding energy management strategies in hybrid electric vehicles; and, finally, Section 7.1.4 addresses a mixed environment regarding optimal control of electric vehicle charging stations.

To avoid biases and to favor baseline comparison and analysis of results among different types of problems, the ground metric used in the analysis is the cumulative reward (return) of episodes, which eases comparison between different types of problems and environments. The QRL models are compared with state-of-the-art baseline approaches, which are classic actor/critic MultiLayer Perceptron networks whose structure can be easily adapted for the three use cases. Both classic RL and QRL models are trained using the same A2C algorithm, to avoid training biases between quantum and classical

models.

With the previous decisions, we are able to compare QRL with classic RL independently of the use case for their peculiarities, therefore simplifying the global analysis of results among different types of problems.

## Use Case 1: HVAC Control

**Problem Statement** This use case is powered by the *Sinergym* [130] simulator. The target building in this study is the environment *Eplus-demo-v1* and focuses on the scenario *5ZoneAutoDx* [130], Pittsburgh, PA, USA. It is a single-floor rectangular building 100 ft  $\times$  50 ft with five zones (four exterior, one interior) and all zones were regularly occupied by office workers. The overall building height is 10 feet. There are windows on all four facades; the south and north facades have glass doors. The building is oriented 30 degrees east of north (see Figure 7.10). The walls are woodshingle over plywood, R11 insulation, and Gypboard. The roof is a gravel built-up roof with R-3 mineral board insulation and plywood sheathing. The windows are of various single- and double-pane construction with 3 mm and 6 mm glass and either 6 mm or 13 mm argon or air gap. The window-to-wall ratio is approximately 0.29. The south wall and door have overhangs. Finally, the floor area is 463.6 m<sup>2</sup> (5000 ft<sup>2</sup>). A state  $s$  in the Reinforcement Learning environment represents a sequence of historical observations (e.g., outdoor air temperature and room temperatures) from the buildings. The goal of this environment is to maximize an aggregation of KPIs (key performance indicators) regarding energy usage and user comfort. The selected environment has the following features:

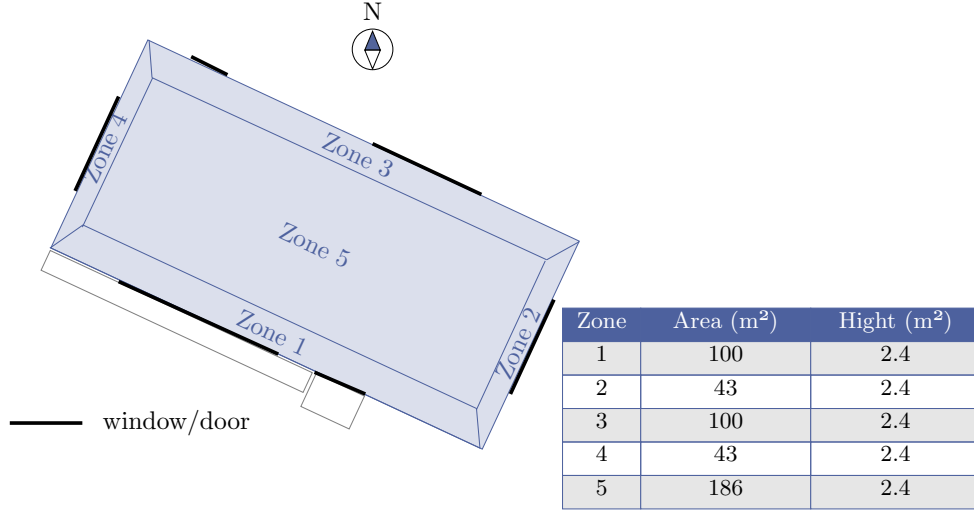


Figure 7.10: Zone building plan.

- **State space:** The state space contains 20 features; 16 are described in Table 7.2 and 4 are reserved for the environment.
- **Action space:** The action space contains a set of 10 discrete actions described in Table 7.3. The bounds for heating setpoint temperature is  $[15, 22.5]$  and for cooling setpoint temperature is  $[21, 30]$ .
- **Reward function:** Reward is always negative. This means that perfect reward would be 0 (perfect power consumption and perfect temperature comfort); there are two temperature comfort ranges (winter and summer) and a weight of energy and comfort. Therefore, the reward is calculated as the sum of reward of energy and comfort multiplied by their respective relevance weights, given by  $W_{energy}$ , as a hyperparameter.

$$reward = W_{energy} * reward_{energy} + (1.0 - W_{energy}) * reward_{comfort} \quad (7.17)$$

Table 7.2: Observation variables.

| 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 <sup>2</sup>  |
| Site Direct Solar Radiation Rate per Area     | W/m <sup>2</sup>  |
| Zone Thermostat Heating Setpoint Temperature  | °C                |
| Zone Thermostat Cooling Setpoint Temperature  | °C                |
| Zone Air Temperature                          | °C                |
| Zone Thermal Comfort Mean Radiant Temperature | °C                |
| Zone Air Relative Humidity                    | %                 |
| Zone Thermal Comfort Clothing Value           | Icl (clo)         |
| Zone Thermal Comfort Fanger Model PPD         | %                 |
| Zone People Occupant Count                    | count             |
| People Air Temperature                        | °C                |
| Facility Total HVAC Electricity Demand Rate   | W                 |

Table 7.3: Action variables.

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

In our experimentation we design a classic MultiLayer Perceptron (MLP) neural network agent with 20 inputs (environment state) and 10 outputs and train the agent in the *Eplus-demo-v1* environment with the A2C algorithm [39]. After that, we also train a quantum agent designed with the methodology described in Section 7.1.3 and trained with the same A2C algorithm. The settings for the environment and A2C algorithm remain the same for the classic baseline and quantum agents, to make a fair performance comparison.

For the actions' control, the proposed framework provides continuous and discrete action space, and the RL agent controllers were trained and evaluated using a discrete action space environment.

**Experimental Settings** Two types of experiments were performed: The first one models the agent’s policy with a classic feedforward MultiLayer Perceptron (MLP) neural network; the second one implements the agent’s policy as a hybrid architecture with one linear layer and one Variational Quantum Circuit. In these experiments we used the advanced actor critic RL method to train the agent. To that end, we ran four environments in parallel to generate batches of five interactions between the agent and each environment, using a discount factor  $\gamma = 0.98$  to calculate the total return of each episode. The stopping criterion was set so that the algorithm stops after 100 episodes were finished. Additionally, the agent performance was tested in the last episode with a deterministic policy that selected the action with maximum probability to check how the agent learned to interact with the environment. Finally, we performed 10 different runs with different initial random seeds to validate the experimentation.

The algorithm A2C requires two networks to perform the learning: a policy network (actor) which inputs a state  $s$  and outputs  $\pi(a|s)$ , i.e., the probability of selection of action  $a$  in state  $s$ , and a value network (critic) which inputs a state  $s$  and outputs an estimation of the value of the state  $V(s)$ . The structure of the policy and value networks for the classical and the quantum agents are described in Table 7.4. Different stages of information processing can be distinguished in Table 7.4: firstly, the quantum agents define the use of 10 qubits (one for each possible action selection). Then, the information of a state of the environment (20 values) is provided to each network (in the case of the quantum VQCs, a scale to the range  $[-\pi/2, \pi/2]$  is required prior to the quantum embedding process). After that, the intermediate network layers of the MLP agent contain 50 units that compute a linear combination of the previous inputs followed by an ReLU activation function. In the case of the quantum agent, the encoded information of the input layer is fed to five sequential quantum variational layers, as explained in Section 7.1.3. The last step in the policy MLP network calculates the probability of selection of each action using a linear layer containing 10 outputs using a softmax activation function, and the policy VQC agent measures the expectation of the  $\sigma_z$  operator over all qubits separately, followed by a softmax activation function to transform the measured data into a probability distribution.

In the case of the policy MLP (actor), the number of parameters is  $20 \times 150 + 150 \times 10$  (weights) +  $150 + 10$  (bias) = 4660; and the number of parameters of the value network (critic) is  $20 \times 150 + 150 \times 1$  (weights) +  $150 + 1$  (bias) = 3301. If we focus on the policy VQC, each qubit of every quantum

variational layer contains two trainable parameters: one for qubit rotation in the y axis, and one for rotation in the z axis. The value network requires two additional parameters to perform the output scale and translation. Then, the total number of parameters is 300 for the policy network:  $20 \times 10 = 200$  (preprocessing layer) + 5 (number of layers)  $\times$  2 (learning parameters used in Rx and Ry)  $\times$  10 (number of qubits). For the value VQC network, the number of parameters is  $20 \times 10 + 5 \times 2 \times 10 + 2 \times 10 = 320$ . We remark that the number of total parameters required by the quantum proposal is much lower than the number of parameters required to train the classical agent so that, in theory, the complexity of the quantum agent is lower than the complexity of its classical counterpart.

Table 7.4: Structure of policy and value classical/quantum networks used in use case 1.

|                     | Policy MLP                    | Value MLP                    | Policy VQC                                                 | Value VQC                                                             |
|---------------------|-------------------------------|------------------------------|------------------------------------------------------------|-----------------------------------------------------------------------|
| No. of qubits       | -                             | -                            | 10                                                         | 10                                                                    |
| Inputs              | 20                            | 20                           | 20. Scaled to $[-\pi/2, \pi/2]$                            | 20. Scaled to $[-\pi/2, \pi/2]$                                       |
| Preprocessing       | -                             | -                            | Linear (10 units)                                          | Linear (10 units)                                                     |
| Quantum embedding   | -                             | -                            | Tanh                                                       | Tanh                                                                  |
| Intermediate Layers | Linear (150 units)<br>ReLU    | Linear (150 units)<br>ReLU   | Tensor product encoding<br>Variational Layer (10 qubits)   | Tensor product encoding<br>Variational Layer (10 qubits)              |
| Output layer        | Linear (10 units)             | Linear (1 unit)              | -                                                          | -                                                                     |
| Post-processing     | softmax activation            | -                            | $\sigma_z(q_i)$ for each qubit $q_i$<br>softmax activation | Product $\sigma_z(q_i)$ for all qubits $q_i$<br>Scale and translation |
| Training algorithm  | RMSProp ( $\lambda = 0.001$ ) | RMSProp ( $\lambda = 0.03$ ) | RMSProp ( $\lambda = 0.01$ )                               | RMSProp ( $\lambda = 0.03$ )                                          |

**Results** The performance results are shown in Table 7.5 summarizing the average, best, and worst total accumulated reward obtained by the MLP and the VQC, as well as the computational time required to finish each experiment (in seconds) and the validation using a deterministic policy that selects the action with maximum probability. The results consider the total accumulated reward of the agent in an episode, i.e., the sum of the reward function described in Equation (7.17) in all steps of an episode. We refer to the average accuracy as the average total accumulated reward in the 10 different runs performed.

The first thing we notice in the results shown in Table 7.5 is that the quantum agent achieves the best result in average, in the best and in the worst total reward value, although the computation time is higher for VQC because of the use of simulators instead of real quantum computers. The boxplots of the total reward distribution for both VQC and MLP are shown in Figure 7.11. We remark on the high standard deviation of the classical agents during the whole process of learning, and this fact suggests that the MLP can achieve suitable results, but it depends highly on the initial parameter values (weights and biases). It also makes the classical MLP less robust than the quantum VQC.

To conclude with the analysis of results, Figure 7.12 plots the learning curves of both quantum VQC and classical MLP agents during the learning process from episode 1 to episode 100. We can observe the better learning in the case of VQC and its higher standard deviation of total reward among episodes of different runs in the first episodes of learning than in the last learning episodes, when it has converged, as should be expected. However, the standard deviation of the total reward in the classical agent remains higher during the whole process of learning, as mentioned above. In addition, the classic MLP becomes trapped in a local optima which does not reach the optimal total reward; therefore, it is not able to learn the optimal policy.

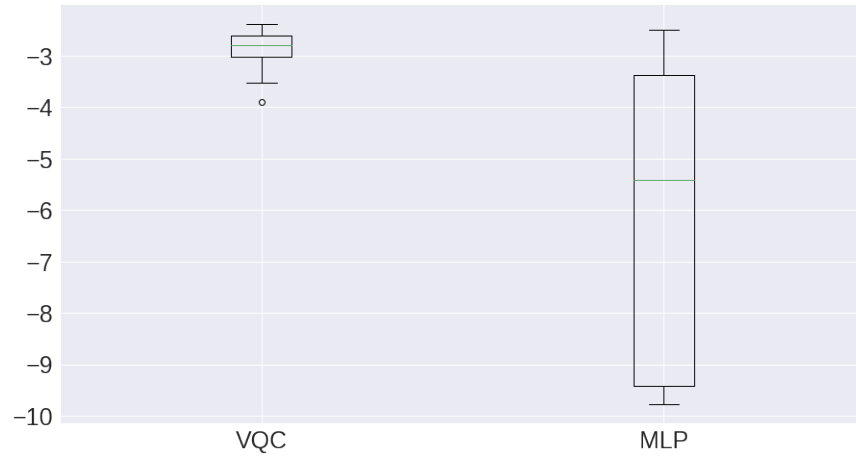


Figure 7.11: Boxplots of average total reward obtained from VQC and MLP in use case 1.

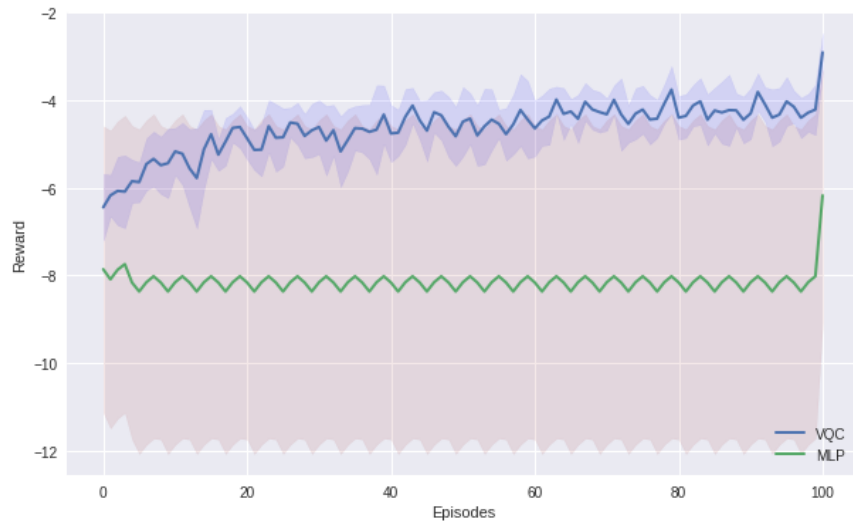


Figure 7.12: Learning curve obtained from VQC and MLP in use case 1.

Table 7.5: Results obtained by MLP and VQC agents in the use case 1 environment. Column 1: MLP agent (classic) and VQC agent (quantum); Column 2: Average total reward after training; Column 3: Best total reward after training; Column 4: Worst total reward after training; Column 5: Computational time in seconds; Column 6: Total reward (test with deterministic policy).

| Model | Avg. Total Reward | Best Total Reward | Worst Total Reward | Time(s) | Test Reward |
|-------|-------------------|-------------------|--------------------|---------|-------------|
| VQC   | -2.91             | -2.38             | -3.90              | 1365.46 | 0.0         |
| MLP   | -6.17             | -2.50             | -9.77              | 737.93  | 0.0         |

## Use Case 2: Energy Management in Electrical Vehicles

**Problem Statement** The environment tested in this use case was first described in [133], and the authors used Deep Reinforcement Learning (DRL) to improve the energy management strategy in hybrid electrical vehicles. More specifically, they developed algorithms based on the deep deterministic policy gradient method to learn optimal energy management policies for the Toyota Prius car. As described in [133], the vehicle is equipped with a gasoline engine, a traction motor, a generator, and a small Ni-MH battery. The authors gathered real data measured experimentally, and they used these data to create a simplified simulation Reinforcement Learning environment with the following features:

- **State space:** The state space contains three features (*SoC*, *velocity*, and *acceleration*). The *state of charge (SoC)* refers to the battery state, while the other features *velocity*, *acceleration* refer to the current state of the vehicle.
- **Action space:** The action space contains a single continuous action, *engine power*, which is a real value that describes the power that the engine must supply. The authors also provided a discretized simulation of the action space, containing 14 different actions that update the required engine power by increasing/decreasing the power by +0 kW,  $\pm 1$  kW,  $\pm 2$  kW,  $\pm 4$  kW,  $\pm 6$  kW,  $\pm 8$  kW, and  $\pm 10$  kW (13 actions), plus an additional action that resets the engine power to value 0 kW. In our experiments, we use the discretized action space.
- **Reward function:** The work [133] attempts to make multi-objective optimization of both the fuel and the battery required by the vehicle. Equation (7.18) describes the aggregated reward function which considers both objectives, where  $\alpha, \beta$  are parameters to set the relevance of fuel/battery consumption in the aggregated objective,  $fuel(t)$  is the fuel consumption at time  $t$ ,  $SoC_{ref}$  is the reference value for the battery state of charge, and  $SoC(t)$  is the battery state of charge at time  $t$ . The goal in the environment is to maximize the accumulated value of Equation (7.18) among the episode, so that the required energy consumption of the vehicle is minimized.

$$r = -\left(\alpha fuel(t) + \beta(SoC_{ref} - SoC(t))^2\right) \quad (7.18)$$

The environment dynamics are as follows: at the beginning of the episode, the agent receives the car current state as the tuple (*SoC*, *velocity*, and *acceleration*). Then, the agent decides which engine power should be provided according to this state, and sends the selected action to the simulator. Finally, the simulator updates the internal state of the car according to the experimental data and the internal simplified Prius modeling, and returns the next state and the current reward to the agent. This is repeated in a loop until the episode ends, which occurs when the real dataset ends.

The authors of the work [133] proposed a neural network agent to solve both continuous and discrete action set environments. In this manuscript, our interest is not to propose a new approach to improve their findings, but to use the work carried out as a ground test scenario to check whether Quantum Reinforcement Learning can be used as a tool to solve problems related to energy efficiency. To do so, in our experimentation we design a classic MultiLayer Perceptron (MLP) neural network agent with 3 inputs (environment state) and 14 outputs (available discrete actions to be selected), and train the agent in the *Prius* environment with the Advantage Actor–Critic (A2C) algorithm. This agent is used as baseline. After that, we also train a quantum agent designed with the methodology described in Section 7.1.3 and trained with the same A2C algorithm. The settings for the *Prius* environment and A2C algorithm remain the same for the classic baseline and quantum agents, to make a fair performance comparison.

**Experimental Settings** As in use case 1, we also tested the quantum agent implemented as VQCs against the classical MLP, using the same conditions and learning algorithm. As in use case 1, four environments were running in parallel to generate batches of 10 interactions between the agent and each environment, using a discount factor  $\gamma = 0.99$  to calculate the total return of each episode. The stopping criterion was set so that the algorithm stops after 100 episodes were finished. In addition, the agent performance was tested every five episodes with a deterministic policy that selected the action with maximum probability to check how the agents were learning to interact with the environment. Finally, we performed 10 different runs with different initial random seeds to validate the experimentation.

The experimental settings for the classical and quantum actor/critic networks are shown in Table 7.6. In the case of the policy MLP, as the linear

combination layers are dense, the number of network parameters is  $3 \times 30 + 30 \times 14$  (weights)  $+ 30 + 14$  (bias)  $= 554$ , and the value MLP has  $3 \times 30 + 30 \times 1$  (weights)  $+ 30 + 1$  (bias)  $= 151$  parameters. If we focus on the policy VQC, each qubit of every quantum variational layer contains two trainable parameters for  $R_y, R_z$  gates. The total number of parameters is  $3 \times 14$  (preprocessing layer)  $+ 5 \times 2 \times 14 = 182$  parameters. The value VQC network requires three qubits only (one for each input), plus two more parameters to perform the output scale and translation. In this case, the value quantum network has  $5 \times 3 \times 2 + 2 = 32$  trainable parameters. It is easy to verify that the number of total parameters required by the quantum proposal is much lower than the number of parameters required to train the classical agent, as happens in use case 1.

Table 7.6: Structure of policy and value classical/quantum networks used in use case 2.

|                     | Policy MLP                    | Value MLP                    | Policy VQC                                                     | Value VQC                                                          |
|---------------------|-------------------------------|------------------------------|----------------------------------------------------------------|--------------------------------------------------------------------|
| No. of qubits       | -                             | -                            | 14                                                             | 3                                                                  |
| Inputs              | 3                             | 3                            | 3. Scaled to $[-\pi/2, \pi/2]$                                 | 3. Scaled to $[-\pi/2, \pi/2]$                                     |
| Preprocessing       | -                             | -                            | Linear (14 units)                                              | Linear (3 units)                                                   |
| Quantum embedding   | -                             | -                            | Tanh                                                           | -                                                                  |
| Intermediate Layers | Linear (30 units)<br>ReLU     | Linear (30 units)<br>ReLU    | Tensor product encoding<br>Variational Layer ( $\times 5$ )    | Tensor product encoding<br>Variational Layer ( $\times 5$ )        |
| Output layer        | Linear (14 units)             | Linear (1 unit)              | -                                                              | -                                                                  |
| Post-processing     | softmax activation            | -                            | Exp $\sigma_z(q_i)$ for each qubit $q_i$<br>softmax activation | Prod $\sigma_z(q_i)$ for all qubits $q_i$<br>Scale and translation |
| Training algorithm  | RMSProp ( $\lambda = 0.001$ ) | RMSProp ( $\lambda = 0.01$ ) | RMSProp ( $\lambda = 0.01$ )                                   | RMSProp ( $\lambda = 0.01$ )                                       |

Finally, we highlight that the previous experimental settings were obtained after a long pre-experimentation in order to find the best configuration for each model.

**Results** We describe the performance results considering the total accumulated reward of the agent in an episode, i.e., the sum of the reward function described in Equation (7.18) in all steps of an episode. As in use case 1, we call average accuracy the average total accumulated reward in the 10 different executions performed. Table 7.7 summarizes the average, best, and worst total accumulated reward obtained by the MLP and the VQC, the computational time required to finish each experiment (in seconds), and the validation using a deterministic policy that selects the action with maximum probability. The boxplots of the total reward distribution for both VQC and MLP are shown in Figure 7.13.

Table 7.7: Results obtained by MLP and VQC agents in the use case 2 environment. Column 1: MLP agent (classic) and VQC agent (quantum); Column 2: Average total reward after training; Column 3: Best total reward after training; Column 4: Worst total reward after training; Column 5: Computational time in seconds; Column 6: Total reward (test with deterministic policy).

| Model | Avg. Total Reward | Best Total Reward | Worst Total Reward | Time (s.) | Test Reward |
|-------|-------------------|-------------------|--------------------|-----------|-------------|
| VQC   | -2.58             | -0.99             | -4.22              | 5675.24   | 0.0         |
| MLP   | -4.28             | -1.79             | -8.69              | 337.04    | 0.0         |

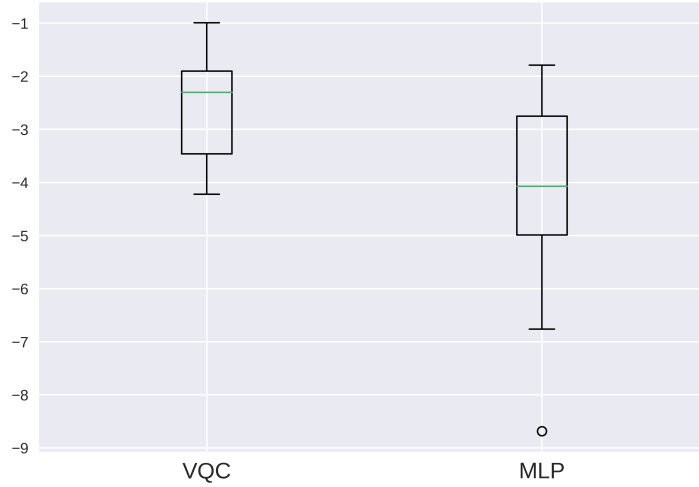


Figure 7.13: Boxplots of average total reward obtained from VQC and MLP in use case 2.

According to Table 7.7 and Figure 7.13, the quantum VQC agent accomplished the task of energy management better than its classic MLP counterpart. In fact, all the average, best, and worst total rewards with a stochastic policy performed better than MLP, and are near the optimum value of 0.0. On the other hand, the clear limitation of the quantum proposal lies in the computational time required to learn the policy, which is near to 1 h 30' per execution against the approximately 5'40'' required by the classical agent. We validated these results using a paired t-test with 95% confidence level, obtaining a  $p$ -value equal to 0.005, which suggests clear statistical differences between the average total reward of VQC and MLP. In any case, neither VQC nor MLP stochastic policies are optimal due to the inherent probabilistic nature of such policies. However, if we transform the stochastic policies to deterministic ones by mean of selecting the action with the highest probability at each episode step, the test rewards of both algorithms perform the same with no statistical differences and are able to learn the optimal policy.

Finally, we analyze the learning curves performed by both agents from episode 1 to episode 100 in Figure 7.14. As is usual in any Deep Reinforcement Learning problem, the standard deviation of the total reward is higher during the execution of the first episodes in both agents, and it is lower as the agent

learns the policy. In Figure 7.14, we also observe that the MLP is able to obtain a nearly optimal policy in fewer episodes than VQC, and that the quantum model is able to overcome the classic agent just in the last episodes. This is consistent with VQC training problems such as the Barren Plateau [16, 97], which makes the gradient have small values, therefore requiring more iterations to converge. To conclude with the analysis of results, Quantum Reinforcement Learning might compensate a tradeoff between accuracy, model complexity (measured as learnable parameters), and learning time, being able to overcome classical ANNs in terms of long-term learning.

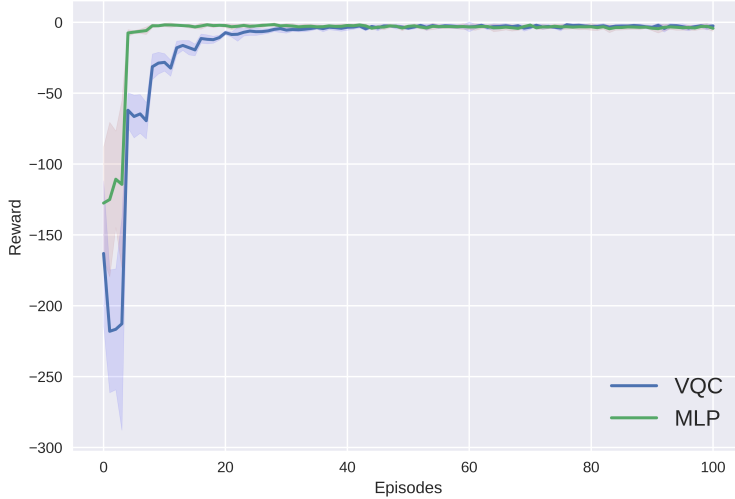


Figure 7.14: Learning curve obtained from VQC and MLP in use case 2.

### Use Case 3: Profit Optimization of Charging Stations

**Problem Statement** This use case differs from the previous experiments since it is not directly targeted at solving energy-efficiency problems. In contrast, it assumes a scenario of scheduling control and real-time pricing in electrical vehicle (EV) charging stations. The Reinforcement Learning environment used in this experimentation was proposed in [135], and considers the operation of an EV charging station over a discretized period of  $T$  time

instants. The objective is to develop an optimal policy that maximizes the profit of the charging station, knowing that there are customers that arrive at a certain time slot, then request service, and the scheduler must send them to the service zone. The price of the service also depends on the electricity price at each time slot. The authors of the article [135] formalized the problem as a Markov Decision Process and created a simulated Reinforcement Learning environment to provide the scheduler with Online Learning. The designed environment has the following features:

- **State space:** The original state space in [135] contains eight features, where four of them are directly acquired from a real experimentation over a working charging station, and the remaining four are aggregated features calculated as a function of the raw experimental measurements, reward design, and action design, to obtain the best performance. In this manuscript, we use the first four raw features as input, namely, the set of EVs which are using the charging station at the current time instant, the expected residual charging demand, the parking time of EVs waiting for service, and the online electricity price.
- **Action space:** At each time slot  $t$ , the agent must make two decisions: (1) the charging price for the EV that arrived at time  $t$ ; and (2) the charging rate of each EV in the service zone. Thus, this environment differs substantially from classic Reinforcement Learning environments, since there are two different action spaces which must be sampled at each time step by the agent to select the two required actions. Both action spaces are discrete: the charging rate can be set to a value in  $\{0 \text{ kWh}, 20 \text{ kWh}, 40 \text{ kWh}, 60 \text{ kWh}, 80 \text{ kWh}\}$ , and the charging price is a value in  $\{1, 2, 3, 4, 5, 6\}$ , which indicates a multiplicative factor over the electricity price to offer the service to the customer arriving at the current time slot.
- **Reward function:** The environment sets the charging station profit as the sole criterion to be optimized. Thus, the goal is to maximize the difference between the payment it collects from customers and the charge it has to pay to the utility company to consume the required electricity. This reward is shown in Equation (7.19), where  $T_t$  is the set of EV arrivals at time  $t$ ,  $p_t$  is the charging price selected at time  $t$ ,  $D_i(p_t)$  is the demand of the  $i$ -th EV in response to the price  $p_t$ ,  $e_t$  is

the charging rate of the station (service zone) at time  $t$ , and  $c_t$  is the electricity price the station has to pay at time  $t$ .

$$r = \sum_{i \in T_t} p_t D_i(p_t) - c_t e_t \quad (7.19)$$

The authors of [135] proposed a single artificial neural network that outputs an action from both action sets simultaneously, in response to a current input state. This strategy has the advantage that the internal feature space learned by the network from the input data is shared by the decision layers of both action spaces. However, in this work, we design the agent containing two different networks, one targeted at the selection of an action from each action space. This decision is justified because of the high computational power required to simulate a Quantum Neural Network with the characteristics of [135]. In our case, we have two separate networks with a reduced size which can be run in a quantum program simulation software in affordable time. In fact, by adopting this strategy, we reduce the quantum search space from  $\mathbb{C}^{2^{11}}$  to two search spaces in  $\mathbb{C}^{2^5}$  and  $\mathbb{C}^{2^6}$ , respectively, which reduces the simulation computational complexity substantially.

**Experimental Settings** As in use case 2, we trained two agents (classical MLP and quantum VQC) under the same Reinforcement Learning algorithm with the same settings. We also used the A2C algorithm to learn stochastic policies. The A2C configuration used was the same as in use case 2, since in a preliminary battery of experiments it was found that these settings could provide optimal policies. With regards to the policy/value classical/quantum networks, we recall that the environment has two different action sets, and we used two different policy networks (one for each action set). Table 7.8 shows each network structure, similarly to how it was explained in the previous use case 2.

In the case of the policy MLP for the first action set, the number of network parameters is  $4 \times 100 + 100 \times 100 + 100 \times 5$  (weights)  $+ 2 \times 100 + 5$  (bias) = 11,105; the number of parameters for the policy MLP in the second action set is  $4 \times 100 + 100 \times 100 + 100 \times 6$  (weights)  $+ 2 \times 100 + 6$  (bias) = 11,206; and the value MLP has  $4 \times 100 + 100 \times 100 + 100 \times 1$  (weights)  $+ 2 \times 100 + 1$  (bias) = 10,701 parameters. If we consider the policy VQC network for the first action set, it contains  $5 \times 2 \times 5 = 50$  parameters; the policy VQC network for the second action set contains  $6 \times 2 \times 6 = 72$

parameters, and the value VQC network has  $4 \times 4 \times 4 = 64$  parameters. As happens in the previous use cases, the number of total parameters to be trained by the quantum proposal is on orders of magnitude smaller than the number of parameters of the classic networks.

Table 7.8: Structure of policy and value classical/quantum networks used in use case 2.

|                     | Policy MLP1                   | Policy MLP2                   | Value MLP                    | Policy VQC1                              | Policy VQC2                          | Value VQC                                 |
|---------------------|-------------------------------|-------------------------------|------------------------------|------------------------------------------|--------------------------------------|-------------------------------------------|
| No. of qubits       | -                             | -                             | -                            | 5                                        | 6                                    | 4                                         |
| Inputs              | 4                             | 4                             | 4                            | 4. Scaled to $[-\pi/2, \pi/2]$           | 4. Scaled to $[-\pi/2, \pi/2]$       | 4. Scaled to $[-\pi/2, \pi/2]$            |
| Preprocessing       | -                             | -                             | -                            | Linear (5 units)                         | Linear (6 units)                     | Linear (4 units)                          |
| Quantum embedding   | -                             | -                             | -                            | Tensor product encoding                  | Tensor product encoding              | Tensor product encoding                   |
| Intermediate Layers | Linear (100 units)            | Linear (100 units)            | Linear (100 units)           | Variational Layer ( $\times 5$ )         | Variational Layer ( $\times 6$ )     | Variational Layer ( $\times 4$ )          |
|                     | ReLU                          | ReLU                          | ReLU                         | -                                        | -                                    | -                                         |
|                     | Linear (100 units)            | Linear (100 units)            | Linear (100 units)           | -                                        | -                                    | -                                         |
|                     | ReLU                          | ReLU                          | ReLU                         | -                                        | -                                    | -                                         |
| Output layer        | Linear (5 units)              | Linear (6 units)              | Linear (1 unit)              | Exp $\sigma_z(q_i)$ for each qubit $q_i$ | $\sigma_z(q_i)$ for each qubit $q_i$ | Prod $\sigma_z(q_i)$ for all qubits $q_i$ |
| Post-processing     | softmax activation            | softmax activation            | -                            | softmax activation                       | softmax activation                   | Scale and translation                     |
| Training algorithm  | RMSProp ( $\lambda = 0.001$ ) | RMSProp ( $\lambda = 0.001$ ) | RMSProp ( $\lambda = 0.01$ ) | RMSProp ( $\lambda = 0.01$ )             | RMSProp ( $\lambda = 0.01$ )         | RMSProp ( $\lambda = 0.01$ )              |

The experimental settings described in Table 7.8 were obtained after a preliminary experimentation was performed to find the best configuration for each model.

**Results** The performances of both VQC and MLP agents are described in Table 7.9, as the total accumulated reward of episodes following the reward function described in Equation (7.19). As in the previous use cases, Table 7.9 contains the average, best, and worst total accumulated reward obtained by both quantum and classic agents after the learning process, together with the computational time required to finish each experiment (in seconds) and the total reward over a test episode using a deterministic policy that selects the agent’s action with maximum probability. Boxplots in Figure 7.15 help to support the analysis of the results of the table.

Table 7.9: Results obtained by MLP and VQC agents in the use case 3 environment. Column 1: MLP agent (classic) and VQC agent (quantum); Column 2: Average total reward after training; Column 3: Best total reward after training; Column 4: Worst total reward after training; Column 5: Computational time in seconds; Column 6: Total reward (test with deterministic policy).

| Model         | Avg. Total Reward | Best Total Reward | Worst Total Reward | Time (s) | Test Reward |
|---------------|-------------------|-------------------|--------------------|----------|-------------|
| VQC (quantum) | 4832.00           | 4832.00           | 4832.00            | 865.47   | 4832.00     |
| MLP (Classic) | 4312.00           | 4752.00           | 3846.11            | 168.83   | 4704.78     |

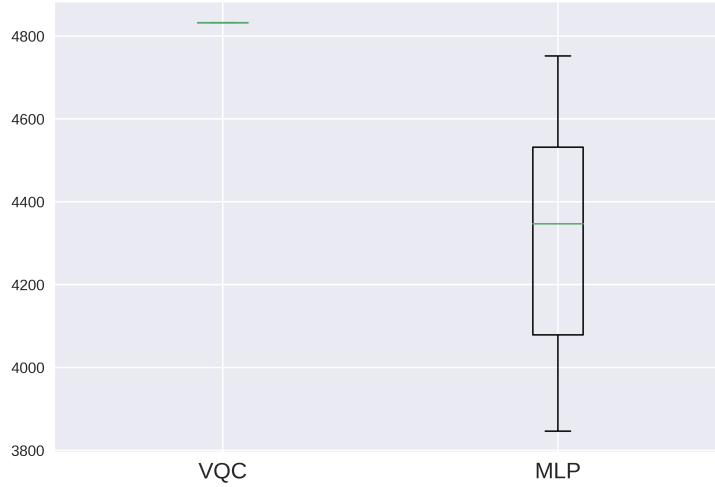


Figure 7.15: Boxplots of average total reward obtained from VQC and MLP in use case 3.

The first thing we notice in the results shown in Table 7.9 is that the quantum agent always converges to the same solution, which in turn is the optimal policy, since the average, best, and worst results print the same value of total reward. Additionally, the execution of the VQC in a test episode using the deterministic policy returns the same value of total reward, which supports this statement.

On the other hand, the classical MLP agent is not able to learn the optimal policy, with a total reward of 4752.00 in the best execution and an average total reward of 4312.00 over all executions. Figure 7.13 shows the average total reward data distributions of both agents at the end of learning, and the previous statement can be verified graphically. In this use case, the quantum VQC agent is also able to overcome its classical counterpart. We validated this analysis using a paired  $t$ -test with 95% confidence level, obtaining a  $p$ -value under  $10^{-6}$ , which indicates that there are significant differences between the results of executions for VQC and MLP. However, as happens in all use cases, the current limitation of the quantum agent relates to the high computational time required to simulate the quantum agent in a classic computer. In this use case, it is almost five times higher (approximately 14'30") than training the MLP (approximately 3').

To conclude the analysis of results, Figure 7.16 plots the learning curves of both quantum VQC and classical MLP agents during the learning process from episode 1 to episode 100. It is worth mentioning two facts from this figure: firstly, the classic MLP shows a faster convergence than the quantum VQC agent, achieving an average reward over 4000.00 in only a few episodes of learning. However, it becomes trapped in a local optima which does not reach the optimal total reward. As happened in the previous use cases, the quantum agent learning is slower, but it is able to achieve better solutions than the classical agent. Secondly, the standard deviation of total reward among episodes of different runs is higher in the first episodes of learning of the quantum agent than in the last learning episodes, when it has converged, as should be expected. However, the standard deviation of the total reward in the classical agent remains higher during the whole process of learning. This fact suggests that the model is highly sensitive to initial conditions (weights and bias network initialization values), which makes the classical MLP less robust than the quantum VQC.

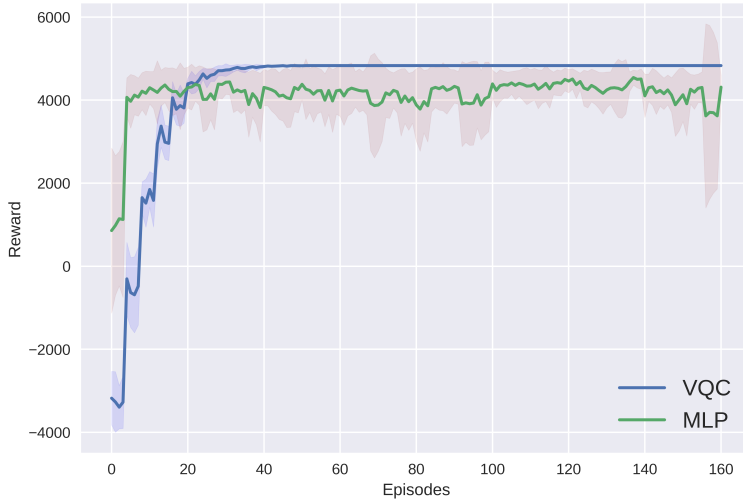


Figure 7.16: Learning curve obtained from VQC and MLP in use case 3.

### 7.1.5 Discussion

We proposed a hybrid classical/quantum model with a classical layer for preprocessing purposes, allowing a reduction of the dimension of the input features by a Variational Quantum Circuit. After the variational layer, a measurement process was performed by the selection of the observables to obtain the expected values and, finally, post-processing using a softmax activation function was performed to transform the measured data into a probability distribution. This structure was applied to three different use cases regarding energy efficiency and management, and it was compared with state-of-the-art classical (non-quantum) RL methods.

A global insight that arises from the study of the results in the three use cases is that the Quantum Reinforcement Learning approach proves the feasibility of implementing decision-making processes with less complexity than their classical counterpart, obtaining better performance than the classical MLP in all use cases studied in this work. On the other hand, they require more time to converge than the classical neural networks, and their computing time is higher than the classical counterpart because of the use of quantum simulators instead of real quantum computers, even when the model complexity is lower than using MLP. In addition, another limitation is that the proposal could not be tested in real contemporary NISQ quantum hardware due to the limitations in accessing these types of computers for large experiments nowadays, and their inherent noise in measurement.

In addition, the quantum agents showed more stability during the training with a lower standard deviation of the total reward, suggesting that the MLP model is more sensitive to initial conditions (weights, bias, and discount factor), which makes the classical MLP less robust than the quantum VQC (especially in use cases 1 and 3).

### 7.1.6 Conclusions and Future Work

In this work, we carried out a study about the suitability of using quantum decision-making models for Deep Reinforcement Learning in different energy-efficiency and management scenarios. The results suggest that the best performance can be obtained by the quantum agent, although the learning process is slower, requiring more interactions with the environment than classical MLP neural network agents, because of the Barren Plateau phenomenon, which refers to gradients that vanish exponentially, similarly to

classical neural networks, but are becoming even more present in VQC.

On the other hand, the complexity of the quantum agents, measured as the number of free model parameters to be optimized, is orders of magnitude smaller than the number of parameters of the classic networks. This suggests that quantum technologies can be suitable to reduce the size and complexity of decision-making processes, although at the cost of a higher computational time to operate in quantum simulation software.

With regards to the quantum agent, we illustrated the importance of encoding and selecting the observables of a quantum model that can represent the observation spaces and the expected outputs. This task becomes even more important when working with environments with a large number of features. Therefore, as a future work, we would have to investigate how to encode environments with a large number of features and how to handle these observables to obtain the expected output more efficiently using a reduced number of qubits. Another challenge in training quantum agents is avoiding the Barren Plateaus, which become even more present when increasing the number of qubits, layers, and entanglement gates. Therefore, building expressive but simple circuits plays a fundamental role in Quantum Reinforcement Learning and should be a key aspect to be studied in the future.

It is also worth mentioning the slow learning speed observed, especially in more complex problems with a high number of dimensions in the state space or the action space. In these cases, quantum agents may provide better results than their classical counterpart, although the simulation on classical computers may become a limitation in terms of resources required for computation and the large size of quantum operations, expressed as complex matrix operations of size  $2^n \times 2^n$ .

In summary, this work is indicative of the potential impact of the relationship between Quantum Computing and Reinforcement Learning in energy-efficiency and management scenarios, demonstrating that quantum agents can solve complex environments even better than their classical counterparts.

## 7.2 Efficient Dimensionality Reduction Strategies for Quantum Reinforcement Learning

|                                                                                                                                                                              |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Reference</b>                                                                                                                                                             |
| E. Andrés, M. P. Cuéllar, and G. Navarro.<br>Efficient Dimensionality Reduction Strategies for Quantum Reinforcement Learning. IEEE Access, vol. 11, pp. 104534-104553, 2023 |
| <b>Status</b>                                                                                                                                                                |
| Published                                                                                                                                                                    |
| <b>Impact factor</b>                                                                                                                                                         |
| IF: 3.9<br>H-Index: 242                                                                                                                                                      |
| <b>Category</b>                                                                                                                                                              |
| Computer Science (Q1), Engineering (Q1)<br>SJR: 0.96                                                                                                                         |
| <b>DOI</b>                                                                                                                                                                   |
| doi: 10.1109/ACCESS.2023.3318173                                                                                                                                             |
| <b>Journal-Editorial</b>                                                                                                                                                     |
| IEEE Access-IEEE Xplore                                                                                                                                                      |
| <b>Logo</b>                                                                                                                                                                  |
|                                                                                          |

## *Efficient Dimensionality Reduction Strategies for Quantum Reinforcement Learning*

**abstract** Quantum Neural Networks constitute one of the most promising applications of Quantum Machine Learning, as they leverage both the capabilities of classical neural networks and the unique advantages of quantum mechanics. Moreover, quantum mechanics has demonstrated its ability to detect atypical patterns in data that are challenging for classical approaches to recognize. However, despite their potential, there are still open questions such as Barren Plateau phenomenon and the challenges of scalability and the curse of dimensionality, which become particularly relevant in Reinforcement Learning (RL) when working in environments with high-dimensional state and action spaces. This study delves into the critical realm of representing classical data as quantum states, a topic of keen interest across the scientific community. The aim is to construct streamlined circuits for efficient execution on quantum computers and simulators using minimal qubits and entanglement gates to evade Barren Plateau phenomena and reducing computational times.

Our investigation examines and validates the efficacy of three strategies for data management and dimensionality reduction in real-world, large-scale environments for Quantum Reinforcement Learning, particularly in energy efficiency scenarios. The techniques encompass amplitude encoding, linear layer preprocessing, and data-reuploading, supplemented by trainable parameters. This research sheds light on the potential of quantum machine learning in enhancing real-world environments, including energy efficiency scenarios and showcases the capabilities of Quantum Neural Networks in the Reinforcement Learning landscape.

**keywords:** Energy Efficiency, Quantum Neural Networks, Quantum Encoding, Quantum Reinforcement Learning, Variational Quantum Circuits.

### 7.2.1 Introduction

Quantum machine learning (QML) [28, 136] is one of the key fields that could reap advantages from near-term quantum devices, alongside optimization and quantum chemistry. Quantum Neural Networks (QNNs) [16, 19], are of particular interest to this manuscript owing to their significant potential in

this area, and their ability to compete with current state-of-the-art classical algorithms based on Artificial Neural Networks (ANNs) [137].

QNNs are models inspired by both neural networks and quantum mechanics. They are implemented as Parametrized Quantum Circuits (PQCs) or Variational Quantum Circuits (VQCs). QNNs have shown success in various domains, including supervised learning [22, 23, 24] and unsupervised learning [25]. While research on Quantum Reinforcement Learning (QRL) is still in its early stages, recent studies have demonstrated that QNNs can outperform classical models in Reinforcement learning (RL) scenarios, achieving better cumulative rewards with fewer parameters to be learned [139].

However, being compared to classic Artificial Neural Networks (ANNs) that have led to significant advancements in various fields, such as AlphaGo [140] and other remarkable achievements [141, 123], quantum approaches to RL still have ample room for improvement. Ongoing advancements in Deep Reinforcement Learning (DRL) and Quantum Computing technologies, such as Google’s Cirq, IBM’s Qiskit, and Xanadu’s PennyLane, have contributed to the increasing interest in QML and QRL among researchers [17, 18].

VQC refers to a class of hybrid classical–quantum algorithm, which are based on a quantum circuit with learnable parameters optimized using classical algorithms. These models have the capability to approximate any continuous function, similar to classical neural networks [21, 19]. In Quantum Reinforcement Learning, VQCs can be used as function approximators for the agent’s policy  $\pi(a|s)$  or Q-values  $Q(s, a)$ . The trainable parameters are optimized using either gradient-based or gradient-free methods.

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

Prior research has showcased the capability of Variational Quantum Circuits to approximate deep Q-value functions for decision-making and policy selection [59],[142]. These studies have leveraged techniques such as Grover’s

algorithm for encoding the policy probability distribution [143] and have even extended the approach to encompass multi-agent scenarios [68].

In the field of energy efficiency and energy management, there is a wide range of approaches in the state-of-the-art of RL. In datacenter environments, RL has been applied to tasks such as efficient resource allocation/deallocation in response to workload variations [47, 48], task scheduling [49, 50], predictive and efficient routing schemes [51], and minimizing power usage of resources [52].

In the case of energy efficiency in buildings, RL has been used for climate control in commercial buildings, such as the application of the Zap Q-learning algorithm to maintain temperature comfort within the desired range while reducing energy consumption [53]. Wang et al. proposed an RL controller based on actor/critic models using Recurrent Neural Networks (LSTM) for HVAC control in buildings [54]. A comprehensive survey on RL applied to energy efficiency in buildings can be found in [55].

Despite the reported success of RL and Deep Reinforcement Learning (DRL) in various energy efficiency and management scenarios, most of the approaches rely on the development of simulation environments that implement digital twins of real devices, often using historical data. Simulation tools like EnergyPlus [128] and Modelica [129] play a fundamental role in training and testing models before deploying them in real energy systems. RL environments are built on top of the simulation software, as for example, Sinergym [130], Gym-EPlus [131], or EnerGym [132], which model energy consumption and control in buildings, simulate energy management in electrical vehicles (EVs) [133, 134], and simulate EV charging stations [135]. Various environments are described and analyzed in [130].

RL algorithms typically operate in large-dimensional action and state spaces, which presents a major challenge for quantum and classical computing implementations. Past studies have endeavored to address this concern by incorporating diverse encoding and dimensionality reduction strategies within the context of OpenAI Gym challenges. These strategies encompass techniques like data re-upload [144], amplitude encoding [145], and the transformation of pixel inputs into quantum data [19]. Building upon these foundations, we extend our investigation to more expansive environments, thereby assessing the scalability of Quantum Reinforcement Learning within real-world scenarios.

In this article, we show that using quantum circuits with the minimum possible number of qubits can not only accurately represent the required information, but also achieve superior performance metrics compared to

circuits that require more qubits. This finding is particularly relevant as qubits are a limited resource subject to noise in quantum computers and time consuming in simulators. Furthermore, circuits with more qubits are more susceptible to falling into Barren Plateaus, emphasizing the importance of optimizing quantum circuits for Reinforcement Learning. Additionally, across all conducted experiments, it consistently emerges that a quantum model surpasses the classical model.

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

The manuscript is organized as follows: Sections 7.2.2 provide an overview of Artificial Neural Networks, Quantum Neural Networks, and Reinforcement Learning, respectively. Section 7.2.3 explains the proposed methods used to build QRL agents. Section 7.2.4 describes the experimental setup and analysis of the results. Finally, the discussion and conclusions of the proposal and their results in QRL energy efficiency scenarios are presented in Sections 7.2.4 and 7.2.4, respectively.

## 7.2.2 Background

For article self-completeness, this section provides a brief introduction to Artificial Neural Networks (ANNs), Quantum Neural Networks (QNNs), and Reinforcement Learning (RL), which are the main topics covered in this work. Firstly, Section 7.2.2, describes the fundamentals of Artificial Neural Networks. This includes an overview of the basic structure and functioning of ANNs. Then, Section 7.2.2, introduces Quantum Neural Networks, and finally Section 7.2.2 presents the basics of Deep Reinforcement Learning (DRL) to learn optimal policies for sequential decision-making problems.

## Artificial Neural Networks

Artificial Neural Networks (ANNs) [137] are computational models inspired by the structure and functioning of the human brain. They are used for a wide variety of tasks, including classification, regression, and pattern recognition. ANNs can be viewed mathematically as a directed graph, where each node represents an artificial neuron, and each edge is assigned with a weight that determines the strength of the connection between two nodes. Based on the connection pattern or architecture, ANNs can be classified into two broad categories: feed-forward networks and recurrent networks. In feed-forward networks, the graphs have no loops and the information flows in only one direction, from input to output. Recurrent networks, on the other hand, have loops because of feedback connections, which allow them to handle time-series data and sequential processing [146].

In the most common family of feed-forward networks, called MultiLayer Perceptron (MLP) whose graphical representation is shown in Fig. 7.17, neurons are organized into layers that have unidirectional connections between them. Typically, there are three or more layers: an input layer where data are presented to the network through an input buffer, an output layer with a buffer that holds the output response to a given input, and one or more intermediate or "hidden" layers [147].

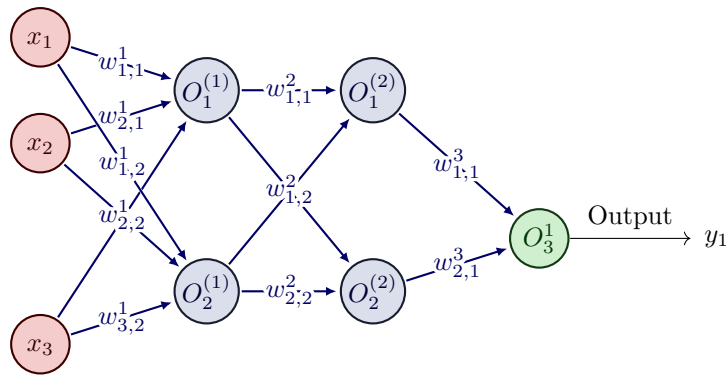


Figure 7.17: **Example of a MultiLayer Perceptron** with 3 inputs, 2 layers with 2 neurons each, and 1 output.

## Quantum Neural Networks

Quantum Machine Learning (QML) models constitute one of the most promising applications of near-term quantum computers [19, 20]. One of the main tools in QML are Quantum Neural Networks (QNNs), which can be implemented as Variational Quantum Circuits (VQCs). VQCs are a class of hybrid algorithm based on a quantum circuit with trainable parameters optimized using classical algorithms. They are able to approximate any continuous function like a classical neural network [21, 19], therefore, they can perform optimization, approximation and classification tasks. The trainable parameters are optimized using gradient-based or gradient-free methods. Fig. 7.18 shows the general schema of a VQC. This hybrid methodology encompasses the following steps [28]:

1. **Pre-processing (CPU)**: In this step we start from a classical input data  $x$ , and targets at preparing the classical data before its encoding into a quantum state. The step encompasses classical preprocessing techniques such as normalizations, changes of scale, etc., with a classical neural network (Fig. 7.21).
2. **Quantum Embedding (QPU)**: The objective of the encoding layer, also known as quantum embedding, is to map the classical data to quantum states  $|x\rangle$  by means of an encoding circuit using parameterized quantum gates. There are different encoding strategies depending on the problem nature, such as the basis encoding, amplitude encoding, or tensor product/angle encoding, among others. In this manuscript, we use the tensor product encoding technique [29] and probability encoding, as an efficient and novel encoding technique that uses a reduced number of qubits.
3. **Variational Layer (QPU)**: The quantum state  $|x\rangle$  is the input of the variational layer (ansatz), which implements the internal behaviour of Quantum neural Networks and it is composed of entanglement and rotation gates parameterized with learnable parameters  $\theta$ . A classical optimization algorithm will optimize these parameters, aiming at minimizing a given cost function. The choice of the encoding strategy, as well as the construction of this layer, are crucial to obtain optimal results. In this work we use the same composition of rotation and entanglement gates in all experiments, and it is inspired by previous works [56, 94, 139].

4. **Measurement Process (QPU/CPU):** In this final step, the quantum state provided by the ansatz is measured and decoded to obtain the desired output. The critical point in quantum measurement is to find an optimal way to associate outputs of the observations with target classes. The selection of the observables used to read out information from the quantum model is crucial to achieve a good performance. In this work, we calculate as output the expectation of the  $\sigma_z$  observable. The value of this observable is in the range  $[-1, +1]$ , where the bound  $+1$  means that ket  $|0\rangle$  is always returned, and  $-1$  means that ket  $|1\rangle$  is always returned as output. In addition, we use for the second proposal the state-vector, since the number of qubits is less than the number of outputs.
5. **Post-processing (CPU).** This step gathers the outputs returned by the QPU and make required transformation to the data before returning the outcomes to the user and to the cost function during the learning process.
6. **Learning (CPU).** This last step computes the cost function optimizes the ansatz parameters  $\theta$  with a classic optimization algorithm (Adam, SGD, etc.). We remark that gradient-free approaches such as COBYLA or SPSA can also be used to approximate the update direction of the parameters.

The selection of the coding strategy and the observables are essential for the construction of efficient and accurate models. Building expressive but simple circuits plays a fundamental role to deal with noise and Barren Plateaus phenomenon, which refers to gradients that vanish exponentially, similar to what happens in classical neural networks, but which are becoming even more present in VQC, and increasing exponentially with the number of qubits, layers, and entanglement gates. This effect is specially relevant when the number of input features or variables is large. In this work, we propose a compact circuit design as a potential solution to mitigate the risk of falling into a Barren Plateau, as well as to reduce the model complexity.

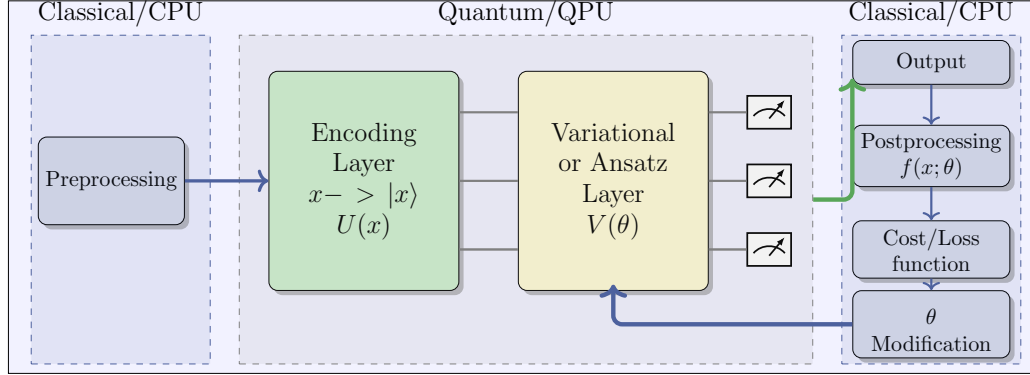


Figure 7.18: **General VQC Schema.** The dashed gray line encompasses the steps executed in a Quantum Processing Unit (QPU) and the dashed blue line shows the steps executed in a CPU.

## Deep Reinforcement Learning

Reinforcement learning [36] is a type of machine learning inspired by behavioral psychology. A decision-making entity, named the agent, will modify its behavior considering the rewards and penalties it receives from its interaction with an unknown environment. RL provides the fundamental basis to describe how intelligent agents learn autonomously to take actions in unknown environments in order to maximize the notion of cumulative reward. DRL unifies classic RL algorithms together with neural networks, and its general schema is shown in Fig. 7.19: When an Agent interacts with an environment, the state of the latter is unknown to the former except for the environment observation it can perceive. At a given time instant  $t$ , the environment observation is written as  $s_t$ . Then, the agent selects an action  $a_t$  from an available action set, and performs it over the environment. After that, the environment evolves to the next state and returns the new state observation  $s_{t+1}$  and a reward  $r_t$  to the agent. The reward informs the agent how good the choice of the conditioned action has been; and it is used to enhance its performance in the next interaction.

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

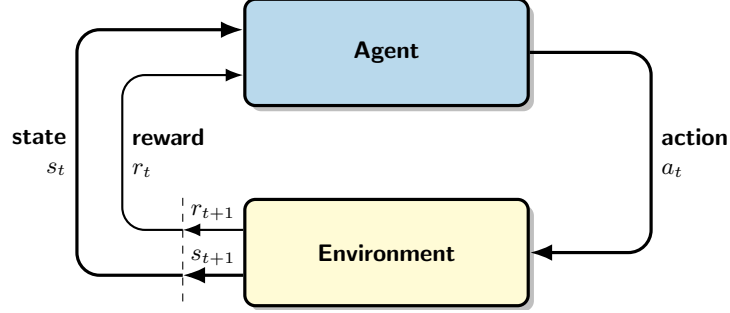


Figure 7.19: **General Reinforcement Learning diagram** [37]. The agent at time  $t$  perceived the state  $s_t$  and considering this state selects an action  $a_t$ . The environment evolves to a state  $s_{t+1}$  returning to this agent this state and a reward  $r_{t+1}$ .

at time  $t$ , and  $r(s_t, a_t, s_{t+1})$  is the reward function for executing action  $a_t$  at state  $s_t$  and then performing a transition from  $s_t$  to  $s_{t+1}$ .

The objective of the agent is to maximize its total reward over a sequence of agent–environment interactions  $\tau$  starting at time  $t_0$ , called the *return* and defined in Equation (7.20), where  $\gamma$  is a hyperparameter to establish how important recent or older rewards are for the learning, known as *discount factor*. In order to maximize  $R(\tau)$ , it is necessary that the agent learns the best action  $a$  to be performed at a given state  $s$ , i.e., the *policy*  $\pi(a|s)$ , this function models the agent’s behavior in the environment, and gives the probability of taking action  $a$  in a given state  $s$ . In RL we consider two important functions: the value of a state–action pair  $Q(s, a)$  in Equation (7.21) (the expected return obtained in the trajectory starting from state  $s$  and action  $a$ ), and the value of a state  $V(s)$  in Equation (7.22) (the expected return obtained in the trajectory starting from state  $s$ ). A third relevant concept is the advantage of the state–action pair  $Adv(s, a)$  (Equation (7.23)), which returns the advantage of choosing action  $a$  in state  $s$  with respect to the other actions available in the action set for the same state  $s$ .

$$R(\tau) = \sum_{t=t_0}^{\infty} r_t \gamma^{t-t_0} \quad (7.20)$$

$$Q(s, a) = \mathbb{E}_{\tau \sim \pi} (R(\tau) | s_t = s, a_t = a) = \sum_{s'} p(s' | s, a) (r(s, a, s') + \gamma \sum_{a'} \pi(a' | s') Q(s', a')) \quad (7.21)$$

$$V(s) = \mathbb{E}_{\tau \sim \pi} (R(\tau) | s_t = s) = \sum_i R(\tau_i) \pi(a_i | s) \quad (7.22)$$

$$Adv(s, a) = Q(s, a) - V(s) \quad (7.23)$$

Deep Reinforcement Learning attempts to use a (deep) artificial neural network to learn the optimal policy. In the literature, two main families of algorithms have been highlighted in the last few years: Deep Q-Networks (DQN) [38] and policy gradient [39]. DQN approximates the function  $Q(s, a)$  by training an artificial neural network, while policy gradient directly learns an optimal policy.

In this paper, we have utilized the DQN algorithm, its training is inspired by the classic Q-learning method, and aims to minimize the loss function in Equation (7.24), where  $\hat{Q}(s, a)$  represents the  $a$ -th output value of the neural network given input  $s$ . This algorithm employs two separate networks to facilitate the learning process. The first is the DQN-Deep Neural Network, which takes the current state  $s$  as input and generates an approximate action-value function (Q-value) for each possible action. The second network is called the target network, which is a copy of the original network and is updated with a delayed frequency to reduce correlations with the target. Additionally, the algorithm employs experience replay, a biologically inspired mechanism that randomizes data, eliminating correlations in the observation sequence and smoothing over changes in the data distribution. This approach helps stabilize the learning process, which has several causes: the correlations present in the sequence of observations, the fact that small updates to  $Q$  may significantly change the policy and therefore change the data distribution, and the correlations between the action-values ( $Q$ ) and the target values  $r + \gamma \max_{a'} Q(s', a')$  [38].

$$MSE = \sum_t \left( r(s_t, a_t, s_{t+1}) + \gamma \max_{a_{t+1}} [\hat{Q}(s_{t+1}, a_{t+1})] - \hat{Q}(s_t, a_t) \right)^2 \quad (7.24)$$

### 7.2.3 Methodology

In this manuscript we adopt a hybrid quantum-classical schema for the agent with a set of parameters  $\theta = (\theta_1, \dots, \theta_k)$  that are learned through interaction with a classical (non-quantum) environment, as depicted in Fig. 7.20. At the beginning of an episode, the quantum-classical hybrid agent receives

the environment state  $s_t$  and uses this information to select an action  $a_t$  based on its policy  $\pi_\theta(a|s)$ . Subsequently, the agent applies this action to the environment perceiving a reward  $r_t$  and the next environment state  $s_{t+1}$ . The policy  $\pi_\theta$  is given by a VQC that runs on a Quantum Processing Unit (QPU). The optimization of this policy is achieved by updating the parameters  $\theta$ , which is performed by a classical learning algorithm running on a Central Processing Unit (CPU), with the aim of minimizing a cost function [94, 139].

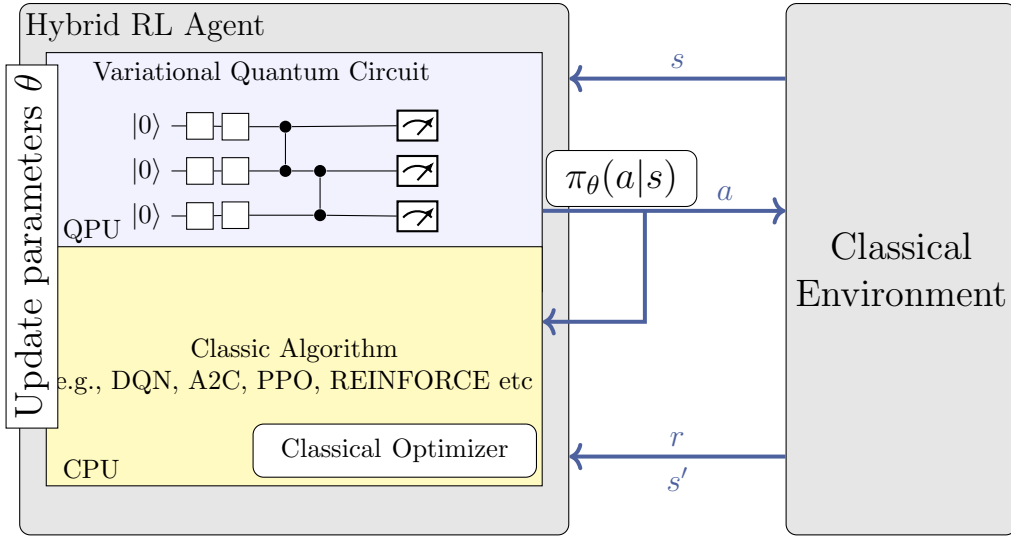


Figure 7.20: **QRL schema**. Example of hybrid agent and classical/quantum interaction to optimize the trainable parameters of the policy using a classical learning algorithm.

In this study, we propose three different QRL architectures for analysis and comparison against classical and compacted models. The architectures are illustrated in Fig. 7.21, Fig. 7.22 and Fig. 7.23, and the aim is to evaluate their performance relative to classical models, while demonstrating the potential of compacted models requiring fewer qubits than other hybrid architectures. The primary hypothesis is that compacted models offer superior performance due to their ability to map classical input data accurately without loss of information and their reduced susceptibility to Barren Plateaus problems.

Through comprehensive experiments and analyses, we aim to demonstrate the efficacy and trade-offs of these architectures and provide valuable insights into their practical application in real-world settings. Specifically, we present three hybrid quantum-classical architectures.

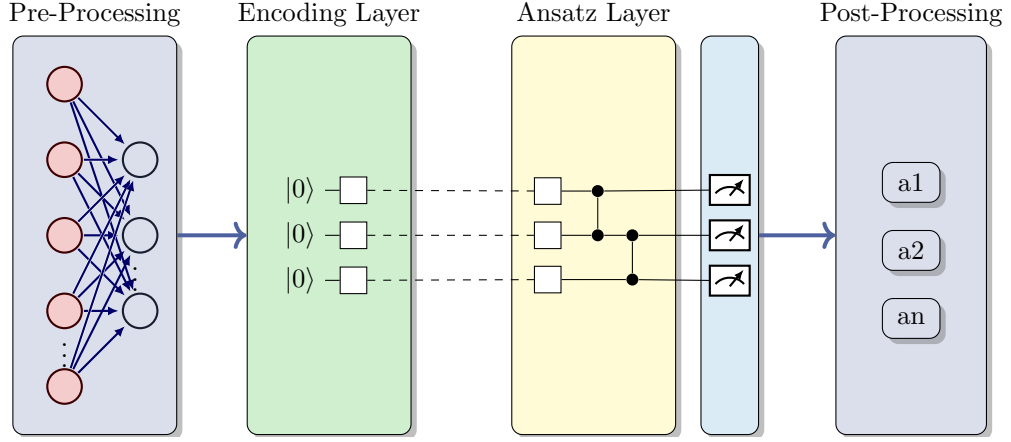


Figure 7.21: **First proposed architecture.** Hybrid classical/Quantum Neural Network. The pre-processing stage utilizes a linear layer, for dimensionality reduction purposes, that is dimensioned with as many input neurons as the observation space dimension and as many output neurons as the action space dimension. This ensures that the expected output dimension is consistent with the dimension used in the quantum circuit. The output of the pre-processing stage is then fed into the encoding circuit, followed by the variational layers. The final output is obtained by measuring each qubit in the computational z-basis and post-processed using a Softmax function.

The first model utilizes a hybrid architecture for the agent’s policy, using a linear layer that serves to reduce the dimensionality of the state space. The number of input neurons of the linear layer corresponds to the dimension of the state space, while the number of output neurons corresponds to the dimension of the action space. As a result, the quantum circuit that follows this layer contains a number of qubits equal to the dimension of the action space, as depicted in Fig. 7.21. Tensor product encoding is employed to convert the observation  $s_t$  into a quantum state  $|s_t\rangle$ . In this work, we calculate as output the expectation of the  $\sigma_z$  observable, i.e., the operation  $\langle\phi|\sigma_z|\phi\rangle$ , for each qubit. The value of this observable is in the range  $[-1, +1]$ , where the bounds  $+1$  means that ket  $|0\rangle$  is always returned, and  $-1$  means that ket  $|1\rangle$  is always returned as output.

The second model builds upon the first one by incorporating the data-reuploading and scaling technique. As discussed in section 7.2.2, an important aspect of designing VQCs is how classical data can be encoded. While

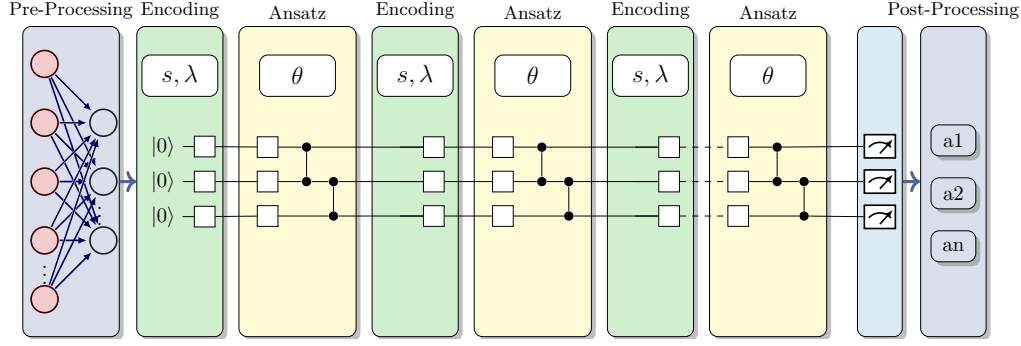


Figure 7.22: **Second proposed architecture.** Hybrid classical/Quantum Neural Network. This architecture incorporates a linear layer to reduce the dimensionality of the input data, along with data-reuploading and scaling techniques. The encoding circuit of this architecture is composed of several unitary gates that are interleaved with the ansatz circuits. The  $\lambda$ -input scaling parameters and the  $\theta$ -weights are the learnable parameters for the encoding and ansatz circuits, respectively. It is worth noting that the number of encoding and ansatz circuits is determined by the number of layers defined in the model, which is three layers in the example shown in this figure.

several encoding methods have been proposed in the literature [29], recent studies on the expressiveness of quantum circuits suggest the advantage of repeated encodings also referred to as data-reuploading [106, 107], in a similar way that a classic ResNet architecture works [137]. The use of this technique, in conjunction with trainable weights on inputs, can lead to the creation of richer models that have the potential to approximate the objective function more accurately. Quantum models have a natural ability to learn periodic functions in data, then, using repeated gate encoding can increase the frequency spectrum of the model, allowing it to represent a wider range of data. However, it is important to ensure that the data is scaled appropriately to fit within the period of the function being learned, thereby enabling the model to learn and represent the data effectively. Furthermore, classical neural networks take several times the same input when processing the data within hidden layers, therefore and considering the non-cloning theorem, the use of data-reuploading makes sense [106, 107, 57]. This technique enhances the model's ability to represent classical data accurately and approximate the objective function effectively. It involves multiple re-uploadings of the encoding layer, where each re-uploading utilizes trainable parameters to

improve the data encoding process. This model is structured with interleaved encoding layers that consist of  $R_x$  gates, and variational layers that consist of  $R_y$ ,  $R_z$ , and  $C_z$  gates. The layout of this model is depicted in Fig. 7.22. Each layer is comprised of one encoding circuit and one ansatz circuit, repeated for the number of layers specified.

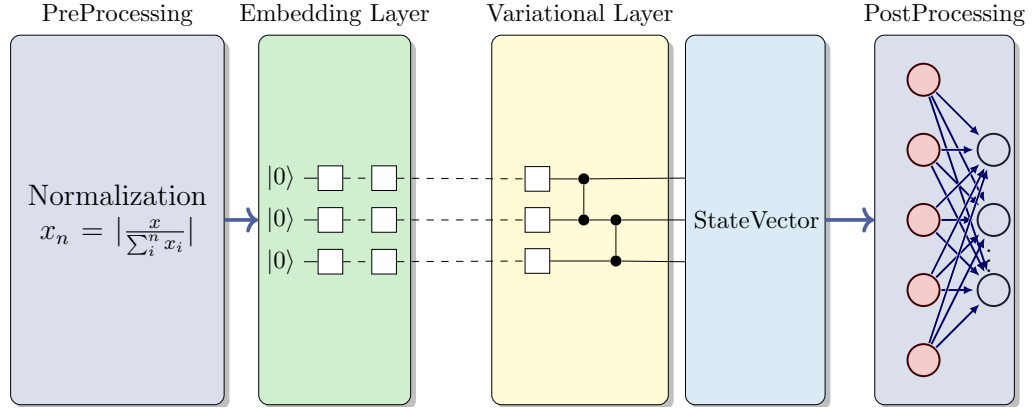


Figure 7.23: **Third proposed architecture.** This architecture involves a pre-processing step in which the classical data is normalized, a necessary requirement for the subsequent encoding strategy. Then, the encoding circuit illustrated in Fig. 7.24 is applied, which employs only  $\log_2 N$  qubits, where  $N$  is the dimension of the observation space, in order to obtain the corresponding quantum state that will feed the ansatz circuit. Finally, the resulting state-vector of this quantum system is subject to post-processing using a classical linear layer, which transforms the dimension obtained with  $2^n$  where  $n$  is the number of qubits, into the action space dimension.

Finally, the third model uses a Variational Quantum Circuit with  $\log_2 N$  qubits to encode  $N$  features without loss of information, in contrast to a linear layer. This encoding technique is known as Amplitud encoding, which is achieved applying  $cRy$  and  $Ry$  rotations over the qubits. To be more specific, a combination of gates is employed to map  $|0\dots 0\rangle \rightarrow |x\rangle = \sum_i a_i |i\rangle$  where  $|i\rangle$  is the  $i$ th entry of the computational basis. The data used in this approach must be normalized, such that  $\sum_i |a_i|^2 = 1$ . Fig. 7.24 depicts an example of the circuit encoding for a system with two qubits. As an example, four features can be represented as follows:

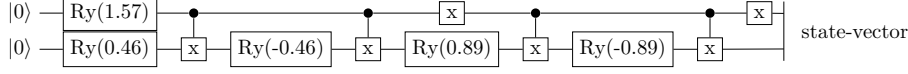


Figure 7.24: **Encoding circuit for the data n=0.2, 0.3, 0.4, 0.1).** Using the decomposition of cRy gates.

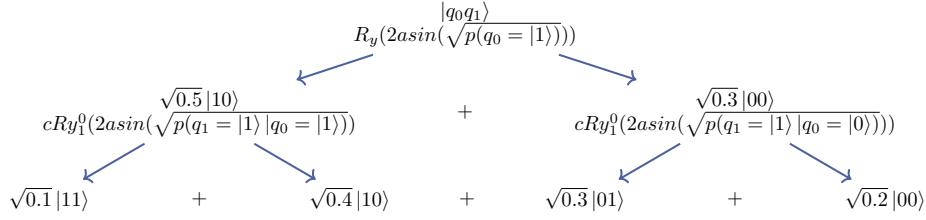


Figure 7.25: Example of amplitude embedding and probability amplitudes decomposition for the data  $n = (0.2, 0.3, 0.4, 0.1)$ .

$$|00\rangle \rightarrow a_{00} |00\rangle + a_{01} |01\rangle + a_{10} |10\rangle + a_{11} |11\rangle \quad (7.25)$$

With two qubits previously initialized to  $|0\rangle$  and applying a single qubit rotation gate  $U(\theta)$  defined in Equation (7.26) on the first qubit, we obtain:

$$U(\theta) = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & -\cos \theta \end{pmatrix} \quad (7.26)$$

$$\begin{aligned} & U(\theta_1) |0\rangle \otimes |0\rangle \\ &= \sqrt{1 - (|a_{10}|^2 + |a_{11}|^2)} |00\rangle + \sqrt{|a_{10}|^2 + |a_{11}|^2} |10\rangle \\ &= \sqrt{|a_{00}|^2 + |a_{01}|^2} |00\rangle + \sqrt{|a_{10}|^2 + |a_{11}|^2} |10\rangle \end{aligned}$$

where  $\theta_1$  is given by:

$$\arcsin(\sqrt{|a_{10}|^2 + |a_{11}|^2}) \quad (7.27)$$

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

$$\cos(\theta_2) = \frac{a_{00}}{\sqrt{|a_{00}|^2 + |a_{01}|^2}}, \sin(\theta_2) = \frac{a_{01}}{\sqrt{|a_{00}|^2 + |a_{01}|^2}} \quad (7.28)$$

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

$$\cos(\theta_3) = \frac{a_{10}}{\sqrt{|a_{10}|^2 + |a_{11}|^2}}, \sin(\theta_3) = \frac{a_{11}}{\sqrt{|a_{10}|^2 + |a_{11}|^2}} \quad (7.29)$$

Therefore, the complete operation will be:

$$\begin{aligned} & \sqrt{|a_{00}|^2 + |a_{01}|^2} |0\rangle \otimes U(\theta_2) |0\rangle + \\ & \sqrt{|a_{10}|^2 + |a_{11}|^2} |1\rangle \otimes U(\theta_3) |0\rangle \\ & = a_{00} |00\rangle + a_{01} |01\rangle + a_{10} |10\rangle + a_{11} |11\rangle \end{aligned} \quad (7.30)$$

---

**Algorithm 1** Amplitude Encoding Algorithm

---

**Input:**  $NQubits, x$   $\triangleright x$  is the classicData  $x$ ,  $NQubits$  is the number of qubits  
**Output:**  $StateVector$

**Require:**  $x = |\frac{x}{\sum_i^n x_i}|$   $\triangleright$  Normalization of ClassicData

- 1:  $ConditionList \leftarrow \{\epsilon, s | s \in (1|0)|(1|0)\{2\} |(1|0)\{3\} \dots |(1|0)\{NQubits - 1\}\}$   $\triangleright$  for 2 qubits:  $['', '1', '11', '10', '0', '01', '00']$
- 2: **for**  $basis$  in  $[0 \text{ to } len(x)]$  **do**
- 3:    $basisStates \leftarrow binary(basis, NQubits)$   $\triangleright$  Building the basis States for ClassicData
- 4: **end for**
- 5: **for**  $conditioning$  in  $ConditionList$  **do**
- 6:    $currentq \leftarrow len(conditioning)$
- 7:    $sum \leftarrow 0$
- 8:    $probl \leftarrow 0$
- 9:   **for**  $state$  in  $basisStates$  **do**
- 10:     **if**  $state \text{ startsWith}(conditioning)$  **then**
- 11:        $p \leftarrow x[state]$   $\triangleright$  the value of  $x$  normalized in state ( $basisState$ )
- 12:        $sum \leftarrow sum + p$
- 13:       **if**  $state[currentq] = '1'$  **then**
- 14:          $probl \leftarrow probl + p$
- 15:       **end if**
- 16:     **end if**
- 17:   **end for**
- 18:    $probl \leftarrow probl / sum$   $\triangleright$  After iterating over the basis states for the current conditioning, we compute the angles.
- 19:    $circuit \leftarrow \epsilon$
- 20:   **if**  $conditioning = \epsilon$  **then**  $\triangleright$  Ry over first angle and qubit 0
- 21:      $circuit \leftarrow circuit + Ry(2asin(\sqrt{probl}), qubit = 0)$
- 22:   **else**
- 23:     **for**  $j$  in  $conditioning$  **do**
- 24:       **if**  $conditioning[j] = '0'$  **then**
- 25:          $circuit \leftarrow circuit + PauliX(qubit = j)$
- 26:       **end if**
- 27:     **end for**
- 28:      $controls \leftarrow [0, 1, \dots, currentq]$   $\triangleright$  cRy gates implementation for the rest of angles
- 29:      $circuit \leftarrow circuit + Ry(asin(\sqrt{probl}), qubit = currentq)$
- 30:      $circuit \leftarrow circuit + ControlledPauliX(controlQubits = controls, targetQubit = currentq)$
- 31:      $circuit \leftarrow circuit + Ry(-asin(\sqrt{probl}), qubit = currentq)$
- 32:      $circuit \leftarrow circuit + ControlledPauliX(controlQubits = controls, targetQubit = currentq)$
- 33:     **for**  $j$  in  $conditioning$  **do**
- 34:       **if**  $conditioning[j] = '0'$  **then**
- 35:          $circuit \leftarrow circuit + PauliX(qubit = j)$
- 36:       **end if**
- 37:     **end for**
- 38:   **end if**
- 39: **end for**

---

The diagram in Fig. 7.25 depicts an example of the encoding process for 2 qubits and 4 features. The general algorithm used for amplitude encoding is outlined in Algorithm 1.

Regarding the ansatz layer, it is worth noting that all models employ a combination of rotation and entanglement gates, which have demonstrated favorable outcomes in various algorithms [56]. The primary aim is to construct circuits with a high level of expressivity, enabling a close approximation of the objective function. This is achieved by employing rotation and entanglement gates, as well as increasing the number of layers in the circuit. However, there is a point of diminishing returns, where increasing these components beyond a certain limit can actually have a negative impact on the convergence of the algorithm.

We remark that the selection of encoding and read-out strategies significantly impact the convergence of the algorithm. In our first proposed approach, we employed PauliZ gates on all qubits for measurement, aligning the number of qubits with the number of outputs. In contrast, for the second approach, where the number of outputs exceeds the number of qubits, we opted for a different strategy. We leveraged the state-vector in the measurement process and employed a linear layer to ensure the desired output dimensionality. Specifically, we employed  $2^N$  neurons for  $N$  qubits, and matched the number of output neurons to the dimensionality of the action space.

## 7.2.4 Experimentation

In the literature we can find different energy-efficiency and management scenarios. Simulation tools such as Energy Plus [128] or Modelica [129] play a fundamental role in the training and testing of models before deploying them in real energy systems. RL environments are built on top of this simulation software, for example, Sinergym [130], Gym-EPlus [131], or EnerGym [132] to model energy consumption and control in buildings.

In this section, we will test Quantum Reinforcement Learning algorithms in various energy efficiency scenarios. Our goal is to compare the results of all proposed models, including the classical MLP, and verify whether QRL and quantum technologies are appropriate for energy efficiency in environments with a large number of variables. Through this experiments, we aim to demonstrate that efficient encoding strategies and hybrid techniques can successfully handle high dimensionality, enabling us to achieve better results than their classical counterparts. We evaluate the influence of different

design choices on learning performance through numerical simulations in three classical benchmarking environments related to energy efficiency. These environments are available in *Sinergym*, an EnergyPlus-supported simulator that contains reference problems for energy efficiency and HVAC control in buildings and facilities.

Section 7.2.4 addresses a use case of classic control of HVAC energy-saving in buildings; Section 7.2.4 solves a datacenter environment; and, finally, Section 7.2.4 addresses a warehouse environment with a small office.

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

## Use Case 1: 5-Zones-Building

**Problem Statement** The target building in this study is the environment *Eplus-demo-v1* and focuses on the scenario *5ZoneAutoDx* [130]. This building is based in Pittsburgh, PA, USA, consists of a single-floor rectangular of 100 ft with five zones, four exterior and one interior, regularly occupied by office workers; oriented 30 degrees east of north (see Fig. 7.26). The overall building height is 10 feet; there are windows on all four facades, single and double panel constructions with 3mm and 6mm glass and argon or 6mm or 13mm air gap; the window-to-wall ratio is approximately 0.29. The south and north facades have glass doors. The walls are wooden shingles over plywood, R11 insulation, and Gypboard; The south wall and door have overhangs. The roof is a gravel built-up roof with R-3 mineral board insulation and plywood sheathing and finally, the floor area is 463.6 m<sup>2</sup> (5000 ft<sup>2</sup>).

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

- **State space:** The state space contains 20 features; 16 are described in Table 7.11 and 4 are reserved for the environment.

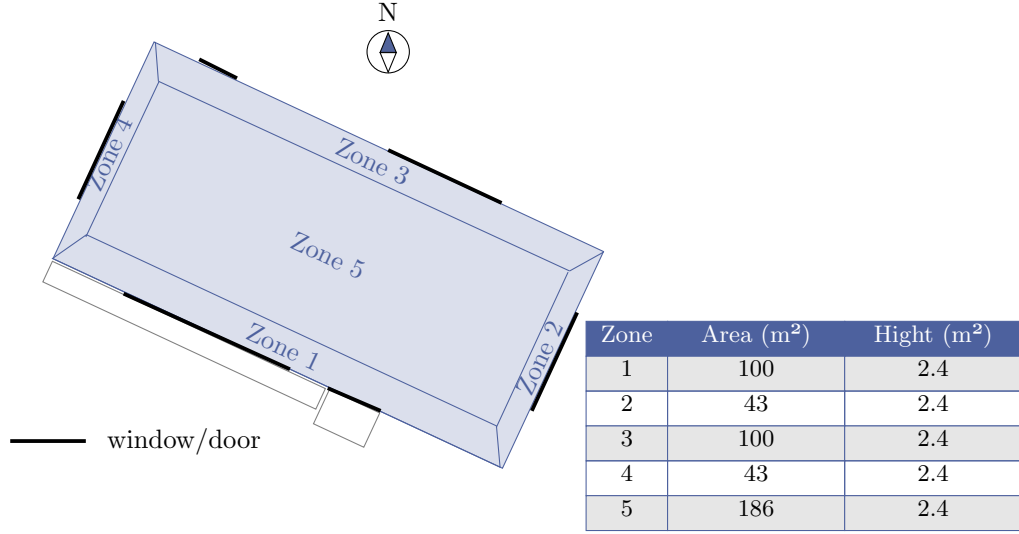


Figure 7.26: **Zone building plan.**

- **Action space:** The action space contains a set of 10 discrete actions described in Table 7.12. The bounds for heating and cooling setpoint temperature are  $[15, 22.5]$  and  $[21, 30]$  respectively.
- **Reward function:** Reward is always negative. This means that the perfect reward would be 0, indicating perfect power consumption and temperature comfort. In addition, there are two temperature comfort ranges (winter and summer) and a weight of energy and comfort. Therefore, the total reward is calculated as the sum of reward of energy and comfort multiplied by their respective relevance weights, given by  $W_{energy}$ , as a hyperparameter. The reward function is the same for all three use cases but can be customized and added to the environment.

$$\begin{aligned}
 reward = & W_{energy} \times reward_{energy} \\
 & + (1.0 - W_{energy}) \times reward_{comfort}
 \end{aligned} \tag{7.31}$$

In this experimentation we design a classic MultiLayer Perceptron (MLP) neural network agent with 20 inputs (environment state dimension) and 10 outputs (environment action dimension), training the agent in the *Eplus-demo-v1* environment with the Deep Q-Network algorithm (DQN) [38]. After that, we also train three different quantum agents designed with the methodologies

Table 7.11: Observation Variables for Use case 1, comprising 16 variables, with an additional 4 empty variables reserved for cases where they are necessary based on the specific requirements of the problem

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

Table 7.12: Action variables

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

described in Section 7.2.3, and configure each model as described in the following section. Finally, the settings for the environment and DQN algorithm remain the same for the classic baseline and quantum agents, to make a fair performance comparison.

**Experimental Settings** In this section we evaluate the three proposed architectures for quantum agents, comparing them to the classical feedforward MultiLayer Perceptron (MLP). The first implementation for the agent’s policy

is a hybrid architecture with one linear layer and one Variational Quantum Circuit 7.21; the second proposal using the same architecture adding data-reuploading and scaling methods 7.22 and finally the third approach using encoding strategies achieves to use a compacted and simplest model but very expressive representing all the information included in the features of the environment, showed in Fig. 7.23.

In the present study, we employed the DQN RL method to train our agents. To ensure the validity and robustness of our results, we conducted 10 independent runs, each with distinct initial random seeds. The DQN algorithm utilizes two distinct networks to facilitate the learning process. The first network is the DQN-Deep Neural Network, which takes the current state  $s$  as input and generates an approximate action-value function (Q-value) for each possible action. The second network is known as the target network, and it is parameterized by a copy of the original network. This target network is initialized with the same weights as the DQN network and is updated with a delayed frequency, thereby reducing correlations with the target. Additionally, uses a biologically inspired mechanism termed experience replay that randomizes over the data, thereby removing correlations in the observation sequence and smoothing over changes in the data distribution. This approach helps stabilize the learning process, which has several causes: the correlations present in the sequence of observations, the fact that small updates to  $Q$  may significantly change the policy and therefore change the data distribution, and the correlations between the action-values ( $Q$ ) and the target values  $r + \gamma \max_{a'} Q(s', a')$  [38].

Table 7.13: Structure of DQN classical/quantum networks used in use case 1

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

The configuration details for all models are summarized in Table 7.13. In the first solution, the quantum agent utilizes a total of 10 qubits, with each qubit corresponding to a possible action selection. Initially, the environment state, consisting of 20 values, is inputted to a linear layer with 20 input neurons employed to compute a linear combination, enabling dimensionality reduction and facilitating subsequent processing with 10 qubits. This is followed by a ReLU activation function. Subsequently, the input information is scaled to the range of  $[-\pi/2, \pi/2]$  prior to being fed into the quantum embedding process. The encoded information from the input layer is then passed through five consecutive quantum variational layers, as described in Section 3.1. Finally, the expectation value of the  $\sigma_z$  operator is separately measured for each qubit, yielding the Q-value associated with each action. In this particular case, the Q-networks employed in the algorithm consist of a total of 300 parameters. This parameter count is calculated as follows:  $20 \times 10 = 200$  (pre-processing layer) +  $5$  (number of layers)  $\times 2$  (learning parameters used in Ry and Rz)  $\times 10$  (number of qubits). The second solution, which is a variant of the previous approach, incorporates data-reuploading and scaling methods. In this case, the Q-networks consist of a total of 350 parameters:  $20 \times 10 = 200$  (pre-processing layer) +  $5$  (number of layers)  $\times 1$  (learning parameter used in Rx for encode the data)  $\times 10$  (number of qubits) +  $5$  (number of layers)  $\times 2$  (learning parameters used in Ry, Rz for ansatz circuit)  $\times 10$ . In the third solution, the architecture consists of encoding and ansatz circuits composed of 5 qubits, where the number of qubits is determined by  $\log_2 N$ , with N representing the number of features in the observation space, in this particular environment, N is equal to 20. The total number of parameters in the third solution's architecture is 225:  $15$  (number of layers)  $\times 3$  (learning parameters used in Rx, Ry and Rz for ansatz circuit)  $\times 5$  (number of qubits). Finally, the classical network consists of an input layer with 20 neurons, a hidden layer of 100 neurons, and an output layer with 10 neurons. The total number of parameters in this case is 3110:  $20 \times 100$  (weights) +  $100$  (bias) +  $10 \times 100$  (weights) +  $10$  (bias).

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

**Results** In this section we present the results from the four models. Table 7.14 provides a comprehensive summary of the average, best, and worst total accumulated reward obtained by the MLP and the VQC models, along with the corresponding computational time required for each experiment, measured in seconds. The evaluation of the results is based on the total accumulated reward obtained by the agent in each episode, which encompasses the summation of the reward function described in Equation (7.31) over all steps of an episode. The average accuracy is defined as the mean total accumulated reward calculated across the 10 different runs conducted.

Table 7.14: Results obtained by MLP and VQC agents in the use case 1. Column 1: MLP agent (classic) and the different VQC agents proposed (quantum); Column 2: Average total reward after training; Column 3: Best total reward after training; Column 4: Worst total reward after training; Column 5: Computational time in seconds; Column 6: Total reward (test with deterministic policy)

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

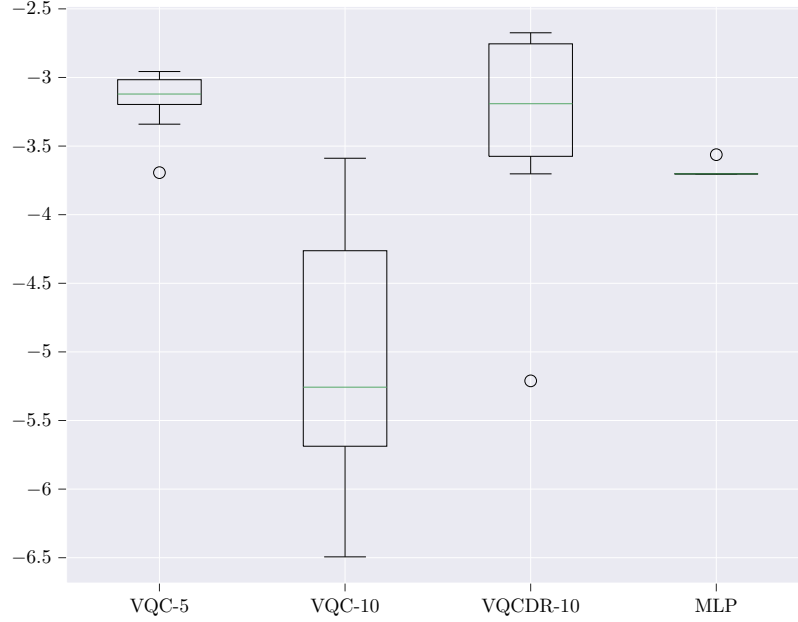


Figure 7.27: Boxplots illustrating the distribution of average total reward obtained from four different approaches in Use case 1, arranged from left to right as follows: a VQC with Amplitude encoding (5 qubits), VQC with linear layer pre-processing, VQC with data-reuploading and scaling methods, and MLP.

One significant observation derived from the outcomes presented in Table 7.14 is consistently, at least one quantum agent outperforms the MLP model across all evaluated categories. However, it is noteworthy that the computational time for the VQC models was comparatively higher due to the utilization of simulators instead of real quantum computers. Despite this trade-off in computation time, the quantum agents demonstrated superior performance across all evaluated metrics, solidifying their effectiveness and potential in this domain.

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

Based on the boxplot analysis presented in Fig. 7.27, several conclusions



Figure 7.28: Learning curves obtained from four distinct methods in Use Case 1. The approaches are: VQC with Amplitude Encoding using 5 qubits, VQC with a linear layer at pre-processing, a variant of the latter approach which incorporates data-reuploading and scaling methods, and the use of MLP.

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

outcomes in each experimental run.

Finally, from Fig. 7.28, it is noticeable that the model utilizing amplitude encoding displays the best learning curve. This is indicated by a decreasing standard deviation of total rewards across episodes in the final iterations, suggesting a more stable and consistent learning process. Following closely is the data-reuploading and scaling methods, which also demonstrate a relatively favorable learning curve, albeit with a slightly higher standard deviation. These findings highlight the effectiveness of amplitude encoding and the potential benefits of incorporating data-reuploading and scaling techniques in improving the learning performance of quantum agents.

## Use Case 2: Datacenter

**Problem Statement** The target building in this study is the *Eplus-datacenter-mixed-discrete-v1* environment, which is one of the environments available in the *Sinergym* simulator [130]. Specifically, the study focuses on the scenario *2ZoneDataCenterHVAC*.

This building model is based in NYC, USA. It comprises two zones: the West zone, with a dimension of 232.26 m<sup>2</sup>, and the East zone, with a dimension of 259.08 m<sup>2</sup>. The building structure consists of mass walls, a regular slab floor, and a regular roof without windows. The IT equipment in the data center is air-cooled. Each zone within the building is served by its own airloopHVAC system. The air loops are equipped with a DX cooling coil and a chilled water cooling coil, which are responsible for the cooling process in the respective zones (see Fig. 7.29).

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

The state space is presented in Table: 7.15, while the action space remains consistent with Use case 1 as described in Section 7.2.4. The reward function is defined by equation (7.31).

In this particular use case, we developed a classic MultiLayer Perceptron (MLP) neural network agent with 29 input nodes representing the environment state, and 10 output nodes. The agent is trained in the *Eplus-datacenter-mixed-discrete-v1* environment. Additionally, we train three quantum agents using the methodologies outlined in Section 7.2.3. It is important to note that

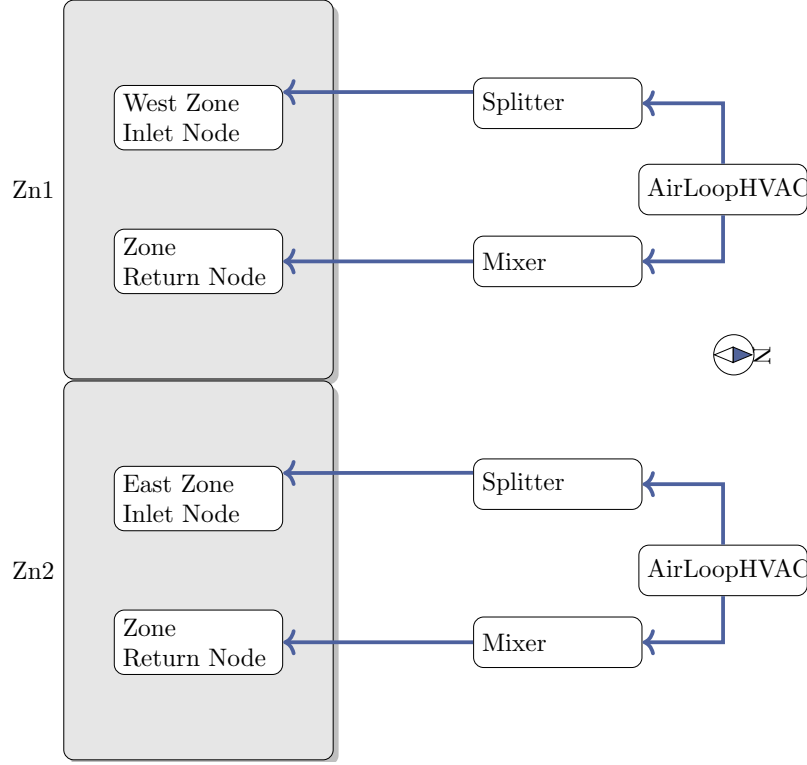


Figure 7.29: The figure illustrates the HVAC components used in the Zone Datacenter. The components include Air Economizer, Direct Evaporative Cooler, Indirect Evaporative Coolers, Single Speed DX Cooling Coil, Chilled Water Coil, and VAV with No Reheat Air Terminal Unit. These components play a crucial role in the data center’s cooling system, providing different cooling mechanisms and control options to maintain optimal temperature conditions.

the environment settings and DQN algorithm parameters are kept consistent across all agents to ensure a fair performance comparison.

**Experimental Settings** For this experiment, we employ four models with identical architectures as those used in the previous use case, as described in Section 7.2.3. The first model is a hybrid model, which incorporates a linear layer for dimensionality reduction. In this model, the Q-networks comprise a total of 390 parameters:  $29 \times 10 = 290$  (pre-processing layer) + 5 (number of layers)  $\times$  2 (learning parameters used in  $R_y$  and  $R_z$ )  $\times$  10 (number of

Table 7.15: Observation Variables for Use case 2, comprising 25 variables, with an additional 4 empty variables included to accommodate specific usage scenarios that require them

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

qubits). The second solution, a variant of the previous model, incorporates data-reuploading and scaling techniques. In this model, the Q-networks are composed of 440 parameters:  $29 \times 10 = 290$  (pre-processing layer) + 5 (number of layers)  $\times$  1 (learning parameter used in Rx for encode the data)  $\times$  10 (number of qubits) + 5 (number of layers)  $\times$  2 (learning parameters used in Ry, Rz for ansatz circuit)  $\times$  10. The third solution utilizes an architecture composed of encoding and ansatz circuits, with 5 qubits corresponding to the logarithm base 2 of the number of features in the observation space ( $N=29$  in this environment). The number of parameters in this model is 250: 25 (number of layers)  $\times$  2 (learning parameters used in Rx, Ry for ansatz circuit)  $\times$  5. Finally, the classical network consists of an input layer with 29 neurons, a hidden layer with 100 neurons, and an output layer with 10 neurons. Consequently, the total number of parameters in this classical network is 4010:  $29 \times 100$  (weights) + 100 (bias) +  $10 \times 100$  (weights) + 10

(bias). The configuration details for these models are presented in Table 7.16.

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

Table 7.16: Structure of DQN classical/quantum networks used in Use case 2

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

**Results** The performance results are presented in Table 7.17, which provides an overview of the average, best, and worst total accumulated reward achieved by the three quantum agents and the MLP. Additionally, the table includes the computation time of each experiment, measured in seconds. The average accuracy refers to the average total accumulated reward obtained from 10 independent runs, where different but fixed seeds were used to ensure reproducibility.

One notable observation from the results presented in Table 7.17 is the consistent performance of at least one quantum agent compared to the MLP model across all evaluated categories. Despite the higher computation time associated with VQC due to the use of simulators instead of real quantum computers, the quantum agents demonstrate superior performance. In particular, the VQC model with 5 qubits, utilizing efficient encoding with amplitude encoding, excels for execution time, mirroring the findings from the previous use case. This model showcases the ability to maintain expressiveness and effectively approximate the objective function even with a smaller number of qubits ( $\log_2 N$  for  $N$  features). As a result, it proves to be a highly competitive and promising option for the given task.

Continuing with the analysis and examining the boxplot in Fig. 7.30, we can observe that the VQC agent with 5 qubits exhibits a relatively moderate interquartile range, indicating a reasonable range of variability in its performance. Although the range is not extremely narrow, the absence of outliers and the proximity of the median to the center suggest a consistent and stable performance. This suggests that this agent achieved the best result in robustness. Following closely is the VQC agent with 10 qubits, although it shows two outliers. The MLP model performs relatively well, with a moderate interquartile range and no outliers. However, the median of the MLP model is located above the box in the boxplot, indicating a slightly higher performance compared to the center of the interquartile range. Lastly, the VQC agent with 10 qubits and the data-reuploading and scaling technique displays a wider dispersion of data points, indicating a larger range of variability in its performance.

Finally, to conclude the analysis of results and regarding learning, in the Fig. 7.31 we can observe a good evolution of learning with very similar curves for all the agents. Notably, there is a decrease in variance in the later episodes, indicating increased stability and convergence in learning. This suggests that the chosen methodologies and architectures successfully capture the environment’s patterns and dynamics. Overall, the learning

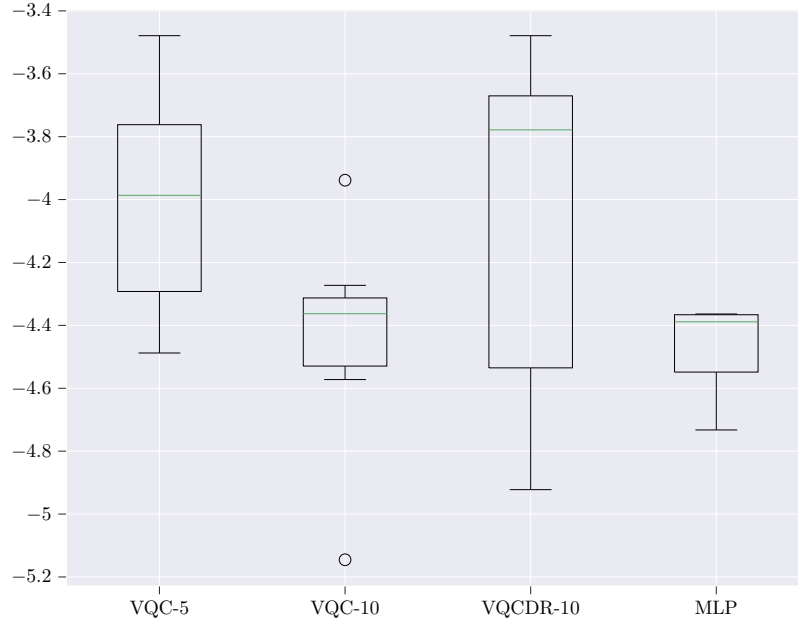


Figure 7.30: Boxplots illustrating the distribution of average total reward obtained from four different approaches in Use case 2, arranged from left to right as follows: a VQC with Amplitude encoding (5 qubits), VQC with linear layer pre-processing, VQC with data-reuploading and scaling, and MLP.

curves exhibit reliable and consistent learning performance, highlighting the agents' capability to adapt to the task.

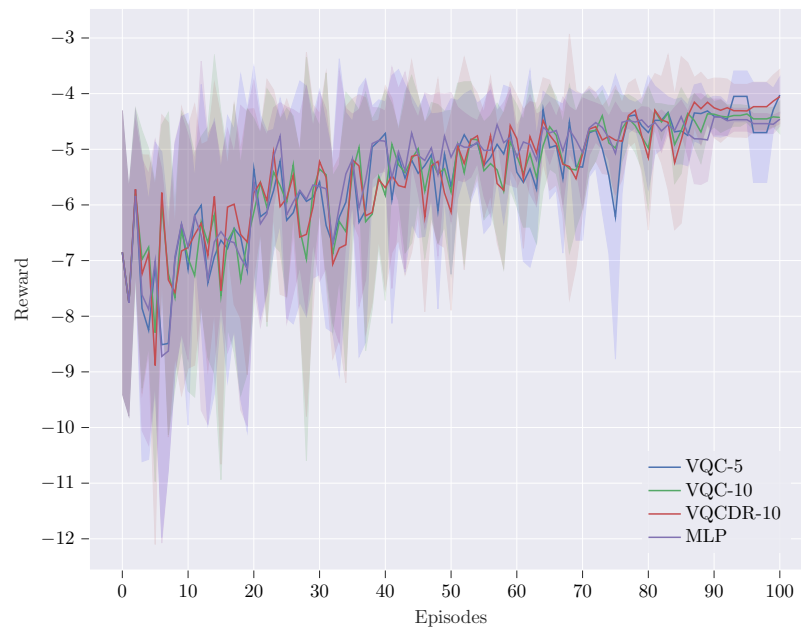


Figure 7.31: Learning curves obtained from four distinct methods in Use case 2. The approaches are: VQC with Amplitude Encoding using 5 qubits, VQC with a linear layer at pre-processing, a variant of the latter approach which incorporates data-reuploading and scaling methods, and the use of MLP.

Table 7.17: Results obtained by MLP and VQC agents in the Use case 2. Column 1: MLP agent (classic) and the different VQC agents proposed (quantum); Column 2: Average total reward after training; Column 3: Best total reward after training; Column 4: Worst total reward after training; Column 5: Computational time in seconds; Column 6: Total reward (test with deterministic policy)

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

### Use Case 3: Warehouse

**Problem Statement** The focus of this use case is the *Eplus-warehouse-mixed-discrete-v1* environment, which is one of the environments available in the *Sinergym* simulator [130]. Specifically, this study concentrates on the *ASHRAE9012016-Warehouse-Denver* scenario.

The building in this study is located in Denver, USA, and has a total area of 49,495 square feet (330 ft x 150 ft). It consists of three zones: bulk storage, fine storage and office. The office area, which is the only section with windows, is enclosed on two sides and at the top by the fine storage zone. The exterior walls are constructed with a metal surface, wall insulation and gypsum board. The roof is also made of metal surface with roof insulation. Additionally, there are skylights present over the bulk and fine storage areas (see Fig. 7.32).

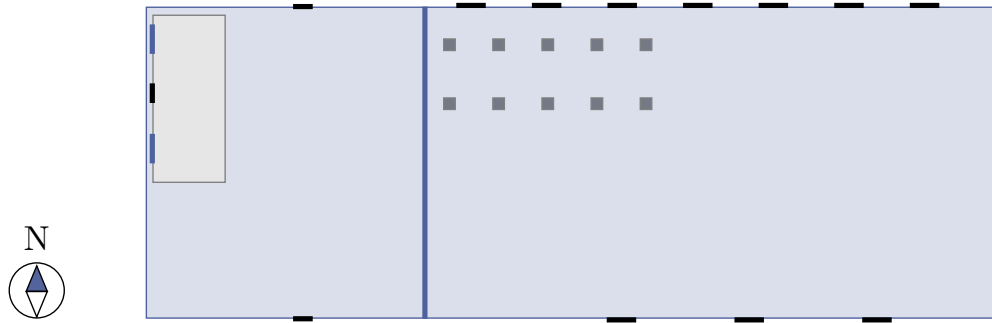


Figure 7.32: **Warehouse plan.** Illustrating bulk storage, fine storage, and an adjacent office area within the fine storage section.

The state space is represented in Table: 7.18, which provides an overview of the variables included in the state space for this use case. The action space is the same as the previous use cases 7.2.4 and the reward function is defined in Equation (7.31).

**Experimental Settings** For this experiment, we utilize four models with an identical architecture, as detailed in Section 7.2.3. The first model, referred to as the VQC-10 qubits model, incorporates a linear layer for dimensionality reduction. The Q-networks within this model are composed of a total of

Table 7.18: Observation Variables for Use case 3. Comprising 19 variables, with an additional 4 empty variables included to accommodate specific usage scenarios that require them

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

330 parameters:  $23 \times 10 = 230$  (pre-processing layer) + 5 (number of layers)  $\times 2$  (learning parameters used in Ry and Rz)  $\times 10$  (number of qubits). And the second solution, a variant of the previous model, incorporates data-reuploading and scaling techniques to the solution. This variant consists of a total of 380 parameters:  $23 \times 10 = 230$  (pre-processing layer) + 5 (number of layers)  $\times 1$  (learning parameter used in Rx for encode the data)  $\times 10$  (number of qubits) + 5 (number of layers)  $\times 2$  (learning parameters used in Ry, Rz for ansatz circuit)  $\times 10$ . For the third solution, the architecture consists of encoding and ansatz circuits with 5 qubits, where the number of qubits is determined by  $\log_2 N$  with N being the number of features in the observation space, which is 23 in this environment. The total number of parameters in this model is 100:  $10$  (number of layers)  $\times 2$  (learning parameters used in Ry, Rz for ansatz circuit)  $\times 5$ . Finally, the classical network used in this experiment comprises an input layer with 23 neurons, a hidden layer with 150 neurons and an output layer with 10 neurons. Consequently, the total number of parameters in this classical network is 5110:  $23 \times 150$  (weights) + 150 (bias) +  $10 \times 150$  (weights) + 10 (bias). The detailed configuration of these models can be found in Table: 7.19.

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

Table 7.19: Structure of DQN classical/quantum networks used in use case 3

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

**Results** The performance results for the third use case are summarized in Table 7.20. This table presents the average, best and worst accumulated reward achieved by the three quantum agents and the MLP model. The computation time for each experiment is also included, measured in seconds. Furthermore, the average accuracy, defined as the average total accumulated reward over 10 different runs with fixed seeds for reproducibility, is reported.

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

Furthermore, the robustness analysis presented in Fig. 7.33 demonstrates that the agent utilizing amplitude encoding exhibits the highest level of robustness among all the models. This is evident from the absence of outliers and the positioning of its median in the middle of the box, indicating consistent and stable performance. The MLP model also performs relatively well in terms of robustness, with a narrower interquartile range. However, it does have two outliers, suggesting some variability in its performance. Finally, the VQC model with 10 qubits and VQC with data-reuploading and trainable input weights show a similar level of robustness. The former, although without outliers and with the median very close to the center, exhibits a larger interquartile range, while the latter has a narrower interquartile range but a higher median and an outlier.

To conclude the analysis, Fig. 7.34 demonstrates that all the models exhibit good learning curves, with a decrease in variance in the later epochs. It is worth noting that the MLP model shows a relatively poorer learning curve compared to the other models.

Table 7.20: Results obtained by MLP and VQC agents in the Use case 3. Column 1: MLP agent (classic) and the different VQC agents proposed (quantum); Column 2: Average total reward after training; Column 3: Best total reward after training; Column 4: Worst total reward after training; Column 5: Computational time in seconds; Column 6: Total reward (test with deterministic policy)

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

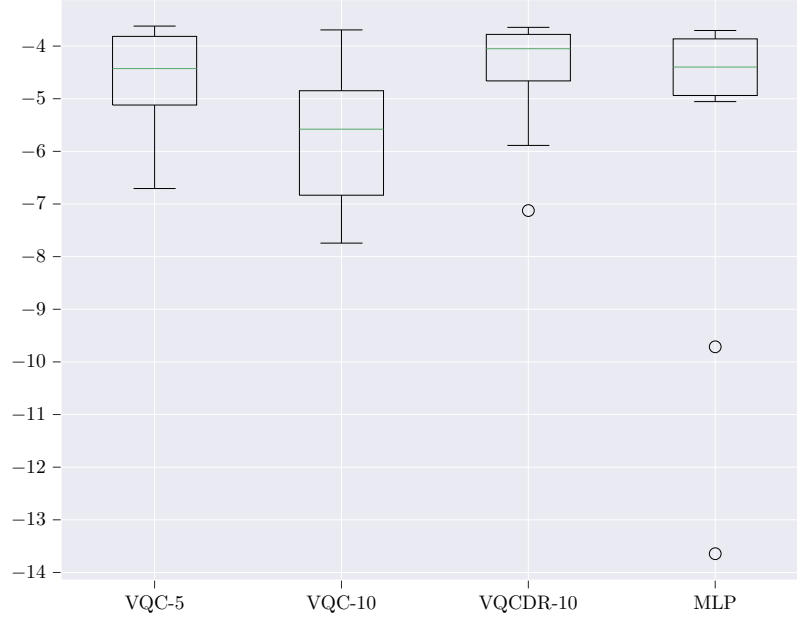


Figure 7.33: Boxplots illustrating the distribution of average total reward obtained from four different approaches in Use case 3, arranged from left to right as follows: a VQC with Amplitude encoding (5 qubits), VQC with linear layer pre-processing, VQC with data-reuploading and scaling methods, and MLP.

## Discussion

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

An overarching insight derived from the analysis of the results across the three use cases is the feasibility and effectiveness of employing Quantum Reinforcement Learning approaches in decision-making processes. These quantum approaches demonstrate the ability to achieve superior performance compared



Figure 7.34: Learning curves obtained from four distinct methods in Use case 3. The approaches are: VQC with Amplitude Encoding using 5 qubits, VQC with a linear layer at pre-processing, a variant of the latter approach which incorporates data-reuploading and scaling methods, and the use of MLP.

to their classical counterparts, while also exhibiting reduced complexity (requiring fewer parameters during the training). Additionally, the Quantum Reinforcement Learning approaches investigated in this study offer various strategies for addressing the challenge of dimensionality reduction, achieving excellent results, particularly through the utilization of compact circuits with the minimum number of required qubits. This highlights the versatility and efficiency of quantum algorithms in effectively handling high-dimensional observation spaces, enabling the agents to make informed decisions while minimizing computational complexity. The combination of superior performance and reduced dimensionality further reinforces the potential of Quantum Reinforcement Learning as a powerful tool for tackling complex decision-making problems. During the training process, all the agents demonstrated stability with lower standard deviation in terms of the total reward. However, consistently across all use cases, the VQC agent utilizing amplitude encoding emerged as the top-performing agent. This agent not only achieved the

highest performance in terms of total reward but also showcased superior robustness compared to other agents. Its ability to consistently outperform other models in various use cases highlights the effectiveness and versatility of Quantum Reinforcement Learning with amplitude encoding.

On the other hand, it is important to acknowledge that the computation time of Quantum Reinforcement Learning approaches is typically longer compared to their classical counterparts. This is primarily due to the utilization of quantum simulators instead of real quantum computers. Although quantum models offer the advantage of reduced model complexity, the simulation of quantum systems introduces additional computational overhead. Furthermore, it is worth noting that the proposed quantum models were not evaluated on contemporary Noisy Intermediate-Scale Quantum (NISQ) hardware. Accessing and conducting large-scale experiments on current quantum devices presents challenges, and the inherent noise in these systems can impact the performance and reliability of the models. These limitations underscore the need for ongoing advancements in quantum hardware and improved accessibility to facilitate real-world testing and validation of Quantum Reinforcement Learning algorithms.

## Conclusions and Future work

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

The selection of an appropriate encoding strategy plays a crucial role in the convergence of the Quantum Reinforcement Learning algorithm. In our study, we employed data-reuploading and scaling techniques [106, 107] and amplitude encoding. These strategies proved to be effective, yielding promising results across the three quantum architectures. Notably, the agent utilizing amplitude encoding consistently outperformed the other architectures, exhibiting superior performance across all the use cases studied in this work.

Addressing the scaling behavior of Quantum Reinforcement Learning

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

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

Secondly, we highlight the potential application of Quantum Reinforcement Learning in energy efficiency and management scenarios, which has been previously demonstrated in our earlier work [139]. By leveraging the power of quantum algorithms and combining them with Reinforcement Learning techniques, we showcase the capabilities of quantum agents in addressing complex decision-making problems in the field of energy efficiency.

And finally, we have addressed critical technical challenges, such as Barren Plateaus [97], potentially leading to prolonged convergence times and, in some cases, impeding optimization altogether. We have successfully mitigated the Barren Plateau issue through the application of higher or adaptive learning rate techniques, allowing us to overcome the stagnation commonly associated with flat regions of the optimization landscape. Additionally, the strategic selection of quantum circuits with optimal qubit counts played a pivotal role in alleviating Barren Plateau-related convergence issues. Our investigation revealed that by carefully designing circuits with the appropriate number of qubits, we were able to circumvent the most severe instances of plateau-induced optimization difficulties. These solutions have not only improved convergence but have also laid the foundation for more stable and efficient Quantum Neural Network training. However, despite these solutions, further research is necessary to comprehensively address the Barren Plateau challenge in Quantum Neural Networks. Exploring alternative parameter initialization methods and novel optimization strategies remains an open area of investigation.

It is important to note that the inherent instability of the DQN algorithm,

which in these use cases has been solved using adaptive learning rates and/or a higher number of layers, highlights the need for alternative neural network models that offer greater stability. Brain-inspired neural network models, for example, hold promise as they bridge the fields of neuroscience, artificial intelligence, and Quantum Computing. Research efforts should be directed towards exploring these models and investigating their potential in Quantum Deep Reinforcement Learning scenarios.

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

Lastly, it is our strong belief that the progress made in the areas of Deep Reinforcement Learning (DRL), Quantum Neural Networks (QNN), and future developments in Quantum Computing hardware will play a pivotal role in advancing the field of Quantum Machine Learning and Quantum Reinforcement Learning. The synergy between DRL and QNN holds great promise for addressing real-world challenges that are beyond the capabilities of classical approaches. By harnessing the power of Quantum Computing and combining it with the sophisticated learning capabilities of DRL, we can expect significant advancements in various domains, including optimization, pattern recognition, and decision-making. Quantum Machine Learning and Quantum Reinforcement Learning have the potential to unlock new insights, uncover hidden patterns, and provide innovative solutions in areas such as drug discovery, financial modeling, logistics optimization, and more.

In conclusion, future research in Quantum Machine Learning and Quantum Reinforcement Learning should focus on addressing the challenges posed by Barren Plateaus, exploring alternative neural network models, optimizing simulation techniques, and leveraging advancements in quantum hardware.

By pursuing these avenues, we can anticipate significant advancements in the fields of Quantum Machine Learning and Quantum Reinforcement Learning, with broad implications for various real-world applications.

## 7.3 Brain-inspired agents for Quantum Reinforcement Learning

|                                                                                                                                                                                                                                                     |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Reference</b>                                                                                                                                                                                                                                    |
| E. Andrés, M. P. Cuéllar, and G. Navarro.<br>Brain-inspired agents for Quantum Reinforcement Learning. <i>Mathematics</i> , vol. 12, no. 8, 2024. <a href="https://www.mdpi.com/2227-7390/12/8/1230">https://www.mdpi.com/2227-7390/12/8/1230</a> . |
| <b>Status</b>                                                                                                                                                                                                                                       |
| Published                                                                                                                                                                                                                                           |
| <b>Impact factor</b>                                                                                                                                                                                                                                |
| IF: 3.4<br>H-Index: 68                                                                                                                                                                                                                              |
| <b>Category</b>                                                                                                                                                                                                                                     |
| Engineering (Q2); Computer Science (Q2)<br>SJR: 0.475                                                                                                                                                                                               |
| <b>DOI</b>                                                                                                                                                                                                                                          |
| <a href="https://doi.org/10.3390/math12081230">doi:10.3390/math12081230</a>                                                                                                                                                                         |
| <b>Journal-Editorial</b>                                                                                                                                                                                                                            |
| Mathematics-MDPI                                                                                                                                                                                                                                    |
| <b>Logo</b>                                                                                                                                                                                                                                         |
|  <b><i>mathematics</i></b><br>an Open Access Journal by MDPI                                                                                                     |

## ***Brain-inspired agents for Quantum Reinforcement Learning***

**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 conducted a comparative analysis of our model against existing quantum models, including Quantum Neural Networks (QNN), QLSTM, QSNN and their classical counterparts, elucidating its unique contributions. Through empirical results, we demonstrated the effectiveness of utilizing quantum models inspired by the brain, which outperform classical approaches and other quantum models in optimizing energy use case. Specifically, in terms of average, best and worst total reward, test reward, robustness and learning curve.

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

### **7.3.1 Introduction**

Recent progress in artificial intelligence (AI) has spurred the emergence of brain-inspired AI, an interdisciplinary field blending principles from neuro-

science, psychology, and computer science to enhance the development of more robust systems [9]. 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 [10].

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 [10, 108, 109, 110, 111, 112, 113]. 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 [114, 115, 116].

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 [148, 149]. Another significant area of research involves the emulation of theory of mind capabilities, achieved through Spiking Neural Networks for actor Networks and Artificial Neural Networks for critic networks [150]. Attentional Reinforcement Learning techniques have also found applications in various domains [151, 152, 153, 154, 155, 156].

Quantum machine learning has arisen as an intriguing focal point within the scientific community. Notably, Variational Quantum Circuits (VQC) also known as Quantum Neural Networks (QNN) have exhibited success across various domains, including unsupervised learning [25] and supervised learning [22, 23, 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 [139, 157].

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 [158]. Several examples of Quantum brain-inspired works exists, such as the study of Li et al [159] 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 [160]. Moreover, our study is motivated by the widely accepted memory model proposed by Atkinson and Shiffrin [2], which delineates the Short-Term Store (STS) for transient memory storage and the Long-Term Store (LTS) for maintaining memories over extended periods [161, 162].

Supporting this perspective, 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, our approach ensure the preservation of both temporal and spatial information throughout the learning process. 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 [73]. Catastrophic forgetting, characterized by the

loss of previously learned knowledge when new information is introduced, presents a significant challenge in real-time systems with small batch sizes, often limited to one [117]. In essence, our innovative approach combines the strengths of QSNN for novel learning with QLSTM’s capacity to preserve and build upon existing knowledge. Furthermore, the integration of QSNN and QLSTM not only aligns with the biological architecture of the human brain but also provides a comprehensive solution to the challenge of catastrophic forgetting in continual learning scenarios. It ensures a seamless transition from short-term to long-term memory, preserving both temporal and spatial information throughout the learning process and enabling the system to adapt to new knowledge without erasing previously acquired insight.

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., [53, 54, 55]) and quantum models (e.g., [139, 157]).

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.

### 7.3.2 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 7.3.2, 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.

Subsequently, in Section 7.3.2, we explore the architecture and operation of Long Short-Term Memory networks followed by the Section 7.3.2, which introduces Reinforcement Learning principles and their integration with (deep) Artificial Neural Networks.

Lastly, in Section 7.3.2, we introduce Quantum Neural Networks, elucidating their key concepts and principles.

## 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 2.5 [72].

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 led 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 below. Additionally, SNNs possess 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 [73, 74].

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 [75].

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  $\theta$ , 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 2.4.

Louis Lapicque [76] 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 2.4b).

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 [73].

$$I_{in}(t) = \frac{U(t)}{R} \quad (7.32)$$

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

$$\tau \frac{dU(t)}{dt} = -U(t) + I_{in}(t)R \quad (7.33)$$

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

$$\tau \frac{U(t + \Delta t) - U(t)}{\Delta t} = -U(t) + I_{in}(t)R \quad (7.34)$$

Extracting the membrane potential in the subsequent step:

$$U(t + \Delta t) = \left(1 - \frac{\Delta t}{\tau}\right) U(t) + \frac{\Delta t}{\tau} I_{in}(t)R \quad (7.35)$$

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

$$U(t + \Delta t) = \left(1 - \frac{\Delta t}{\tau}\right) U(t) \quad (7.36)$$

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  $\beta = 1 - \Delta t/\tau$ .

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

$$\beta = 1 - \frac{1}{\tau} \Rightarrow U(t + 1) = \beta U(t) + (1 - \beta)I_{in}(t + 1) \quad (7.37)$$

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

$$U(t) = I_{in}(t)R + [U_0 - I_{in}(t)R]e^{-t/\tau} \quad (7.38)$$

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  $t = 0$

$$U(t) = U_0 e^{-t/\tau} \quad (7.39)$$

If we compute Eq.(7.39) at discrete intervals of  $t, (t + \Delta t), (t + 2\Delta t) \dots$ , then we can determine the ratio of the membrane potential between two consecutive steps using:

$$\beta = \frac{U_0 e^{-(t+\Delta t)/\tau}}{U_0 e^{-t/\tau}} = \frac{U_0 e^{-(t+2\Delta t)/\tau}}{U_0 e^{-(t+\Delta t)/\tau}} = \dots \Rightarrow \beta = e^{-\Delta t/\tau} \quad (7.40)$$

This equation for  $\beta$  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):

$$WX(t) = I_{in}(t) \quad (7.41)$$

$X(t)$  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:

$$U(t+1) = \beta U(t) + WX(t+1) \quad (7.42)$$

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

$$U(t) = \underbrace{\beta U(t-1)}_{\text{decay}} + \underbrace{WX(t)}_{\text{input}} - \underbrace{S_{out}(t-1)\theta}_{\text{reset}} \quad (7.43)$$

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  $\theta$  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:

$$S_{out}(t) = \begin{cases} 1, & \text{if } U(t) > \theta \\ 0, & \text{otherwise} \end{cases} \quad (7.44)$$

Various techniques exist for training SNN's [73], 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:

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial S} \underbrace{\frac{\partial S}{\partial U}}_{\{0, \infty\}} \frac{\partial U}{\partial I} \frac{\partial I}{\partial W} \quad (7.45)$$

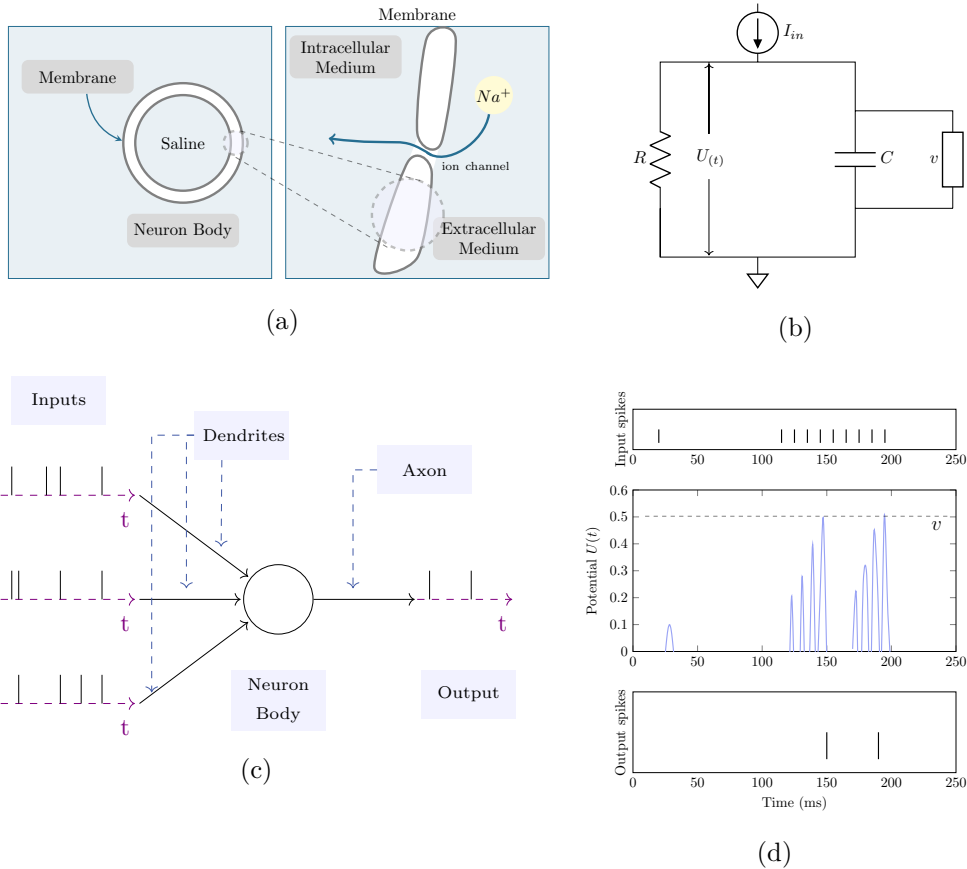


Figure 7.35: The Leaky Integrate-and-Fire Neuron Model [73] involves an insulating lipid bilayer membrane that separates the interior and exterior environments. Gated ion channels enable the diffusion of charge carriers, such as  $Na^+$ , through the membrane.7.35a. This neuron function is often modeled using an RC circuit. When the membrane potential exceeds the threshold, a spike is generated 7.35b. Input spikes are transmitted to the neuron body via dendritic branches. Accumulation of sufficient excitation triggers spike emission at the output 7.35c. A simulation depicting the membrane potential  $U(t)$  reaching a threshold of  $\theta = 0.5V$ , leading to the generation of output spikes 7.35d

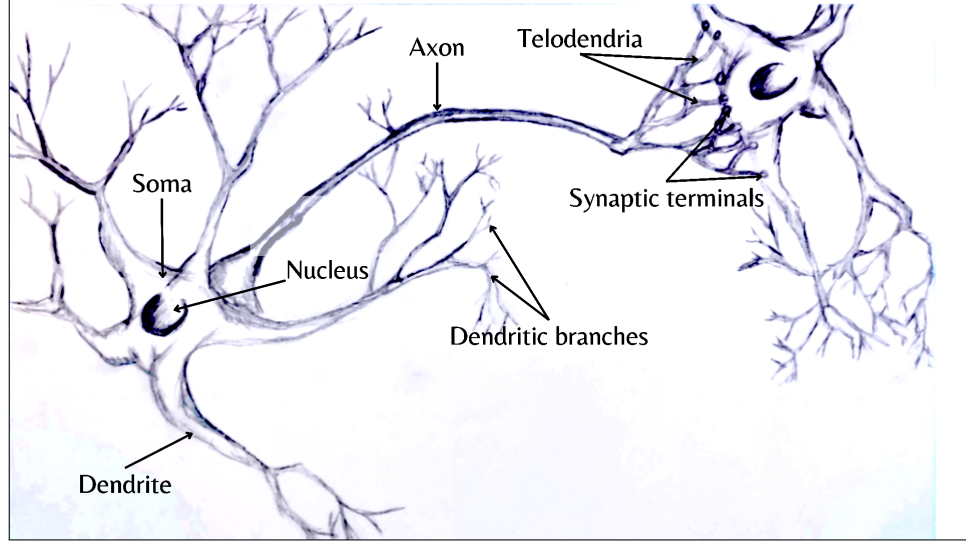


Figure 7.36: The standard structure of a neuron includes a cell body, also known as soma, housing the nucleus and other organelles; dendrites, which are slender, branched extensions that receive synaptic input from neighboring neurons; an axon; and synaptic terminals.

Nevertheless, the derivative of the Heaviside step function from (2.13) 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 [73].

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 descent method. The Figure 7.37 shown the general pipeline for this model for a classification task and the detailed architecture is defined in section 7.3.3.

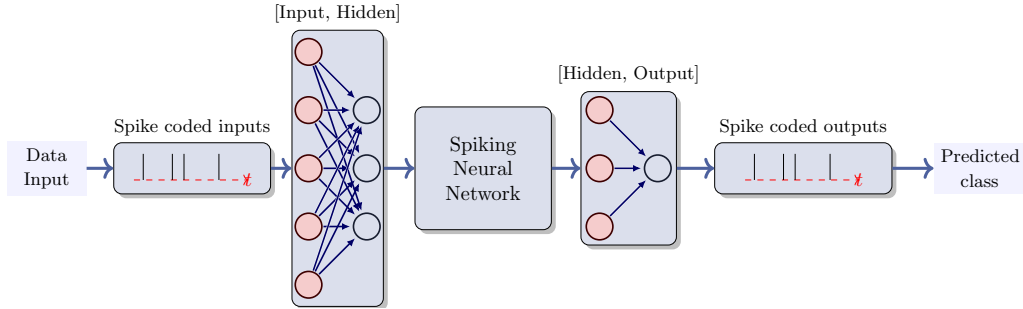


Figure 7.37: **SNN pipeline.** The input data for an SNN may undergo transformation into a firing rate or alternative encodings to produce spikes. Subsequently, the model is trained to accurately predict the correct class by employing encoding strategies, such as the highest firing rate or firing first, 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 [77] or in the decoding and understanding of muscle activity and kinematics from electroencephalography signals [78], utilizing HyperDimensional Computing (HDC) and SNN for the MNIST classification problem [77]. 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 [79].

## 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 [82]. 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 [84, 85].

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 [86]. They consist of three internal gates, known as input, forget and output gates, detailed in Figure 2.7.

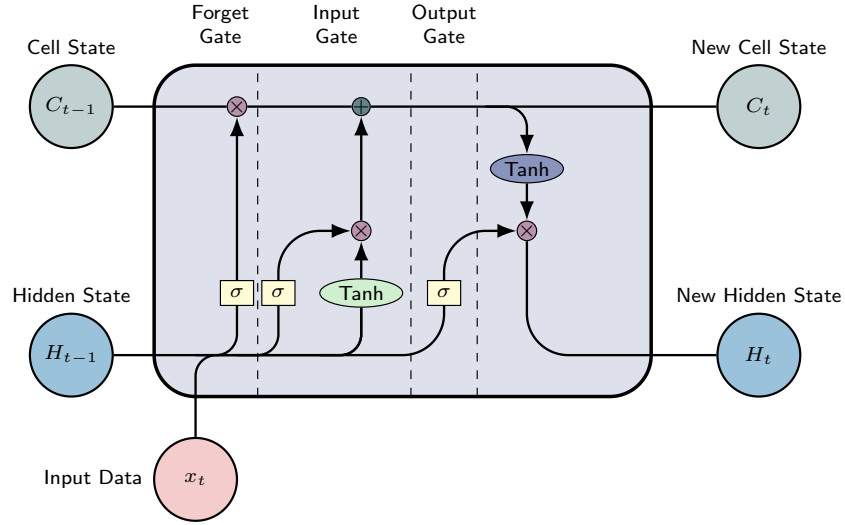


Figure 7.38: **LSTM Cell Architecture:** Featuring three essential gates (forget, input and output gates). The  $\tanh$  and  $\sigma$  blocks symbolize the hyperbolic tangent and sigmoid activation functions, correspondingly.  $x_t$  represents the input at time  $t$ ,  $h_t$  denotes the hidden state, 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 [163]:

- **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.

- **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 7.43 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:

$$f_t = \sigma(w_f \cdot v_t + b_f) \quad (7.46)$$

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:

$$\begin{aligned} i_t &= \sigma(W_i \cdot v_t + b_i) \\ \tilde{C} &= \tanh(W_c \cdot v_t + b_c) \end{aligned} \quad (7.47)$$

- 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  $\widetilde{C}_t$ :

$$C_t = f_t \cdot c_{t-1} + i_t \cdot \widetilde{C}_t \quad (7.48)$$

- 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.

$$o_t = \sigma(W_o \cdot v_t + b_o) \quad (7.49)$$

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.

$$h_t = \tanh(C_t) \cdot o_t \quad (7.50)$$

## Deep Reinforcement Learning

Reinforcement Learning (RL) [36] 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 unknown 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 illustrated in Figure 7.39: 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.

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

$$P(s'|s, a) = P[s_{t+1} = s' | s_t = s, a_t = a] \quad (7.51)$$

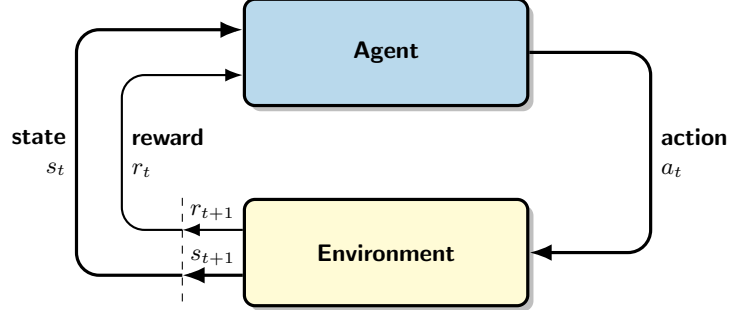


Figure 7.39: **General Reinforcement Learning diagram** [37]. At time  $t$ , the agent perceives state  $s_t$  and, based on this state, selects an action  $a_t$ . The environment then transitions to a new state  $s_{t+1}$  and provides the agent with this new state along with a reward  $r_{t+1}$ .

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  $\tau$ , beginning at time  $t_0$ . This cumulative reward, referred to as the return and defined in Equation (2.7), is influenced by the hyperparameter  $\gamma$ , which determines the relative significance of recent versus past rewards in the learning process.  $\gamma$  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 characterizes 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 (2.8), 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 (2.9), 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 (2.10)), which quantifies the benefit of selecting action  $a$  in state  $s$  compared to the other available actions in the same state  $s$ .

$$R(\tau) = \sum_{t=t_0}^{\infty} r_t \gamma^{t-t_0} \quad (7.52)$$

$$Q(s, a) = \mathbb{E}_{\tau \sim \pi} (R(\tau) | s_t = s, a_t = a) = \sum_{s'} p(s' | s, a) (r(s, a, s') + \gamma \sum_{a'} \pi(a' | s') Q(s', a')) \quad (7.53)$$

$$V(s) = \mathbb{E}_{\tau \sim \pi} (R(\tau) | s_t = s) = \sum_i R(\tau_i) \pi(a_i | s) \quad (7.54)$$

$$Adv(s, a) = Q(s, a) - V(s) \quad (7.55)$$

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) [38] and policy gradient [39]. 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 (7.57). 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: [38, 39]. In this manuscript, we employ the Advantage Actor-Critic (A2C) training algorithm [39], which is characterized by its designed loss function outlined in Equation (7.56).

$$LogLoss = - \sum_t Adv(s_t, a_t) \log \pi(a_t | s_t) \quad (7.56)$$

$$MSE = \sum_t \left( r(s_t, a_t, s_{t+1}) + \gamma \max_{a_{t+1}} [\hat{Q}(s_{t+1}, a_{t+1})] - \hat{Q}(s_t, a_t) \right)^2 \quad (7.57)$$

## Quantum Neural Networks

Quantum Neural Networks (QNNs) play a pivotal role in Quantum Machine Learning (QML) [19, 20], 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 [21, 19], enabling tasks such as optimization, approximation, and classification.

Furthermore, it is noteworthy that Quantum Neural Networks can be efficiently simulated in coherent Ising machines (CIMs) [164]. Additionally, there are other noteworthy advancements in the realm of Quantum Neural Networks, as evidenced by studies such as those conducted by Hou et al. [165], Zhao et al. [166], Zhou et al. [167], Ding et al. [168], Tian et al. [169] and a comprehensive review by Jeswal et al. [170].

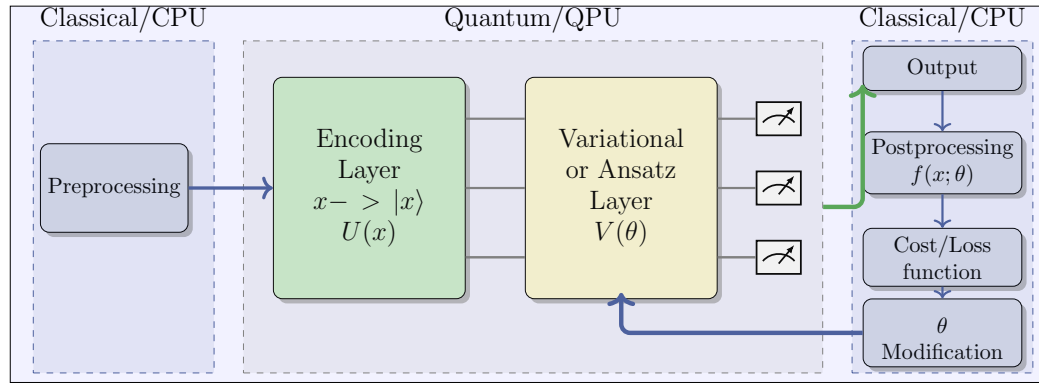


Figure 7.40: **General VQC Schema.** The dashed gray line delineates the process carried out within a Quantum Processing Unit (QPU), while the dashed blue line illustrates the processes executed in a CPU.

Figure 2.2 depicts the overall structure of a VQC. This hybrid approach involves the following stages [28]:

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 [29].

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.
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 COBYLA, are also capable of estimating parameter updates.

### 7.3.3 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 [73]. Catastrophic forgetting, which occurs when new information causes the network to lose previously learned knowledge [117], 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 [171], using ensembles of networks [172], implementing pseudo-replay techniques [173], and penalizing weights that exhibit excessively rapid changes [174]. 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 4.7 illustrates this architecture.

The subsequent model is a Quantum Spiking Neural Network (QSNN), implemented using the Python package `snnTorch` [175]. The architecture of the QSNN is showed in Figure 7.44. 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 7.42 and the entire structure in Figure 7.43. 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.

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 7.44 and 7.43 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

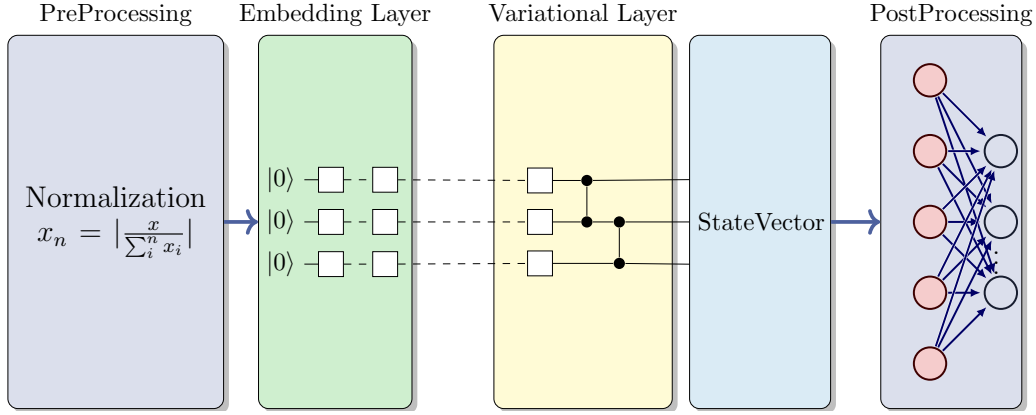


Figure 7.41: **QNN architecture.** This architecture involves several steps. First, classical data is normalized as a preprocessing step, which is necessary for the subsequent encoding strategy. Then, the amplitud encoding algorithm is applied, using only  $\log_2 N$  qubits, where  $N$  corresponds 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.

illustrated in Figure 7.45

### 7.3.4 Experimentation

In the literature, various scenarios related to energy efficiency and management are explored. Simulation tools like Energy Plus [128] or Modelica [129] 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 [131], Sinergym [130] or EnerGym [132] 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

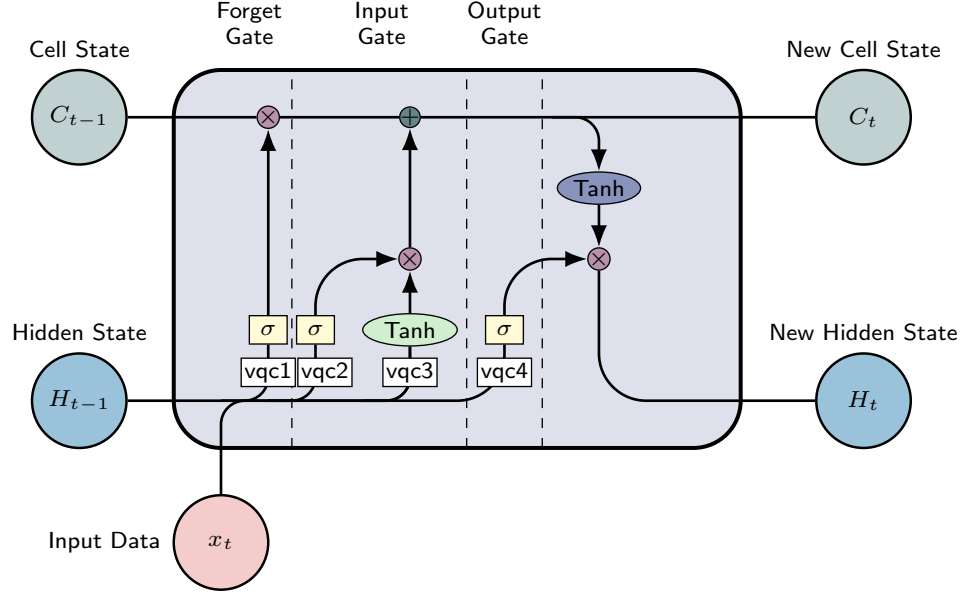


Figure 7.42: **QLSTM Cell**. Each VQC box follows the structure outlined in Figure 2.2. The  $\sigma$  and  $\tanh$  blocks denote the sigmoid and the hyperbolic tangent activation functions, respectively.  $x_t$  represents the input at time  $t$ ,  $h_t$  denotes the hidden state and  $c_t$  signifies the cell state. Symbols  $\otimes$  and  $\oplus$  signify element-wise multiplication and addition, respectively.

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.

## Problem Statement

The focus of this study is the building environment named *Eplus-5zone-hot-discrete-v1*, specifically targeting the scenario *5ZoneAutoDx* [130]. Situated in Arizona, USA characterized by subtropical weather and desert heat, the

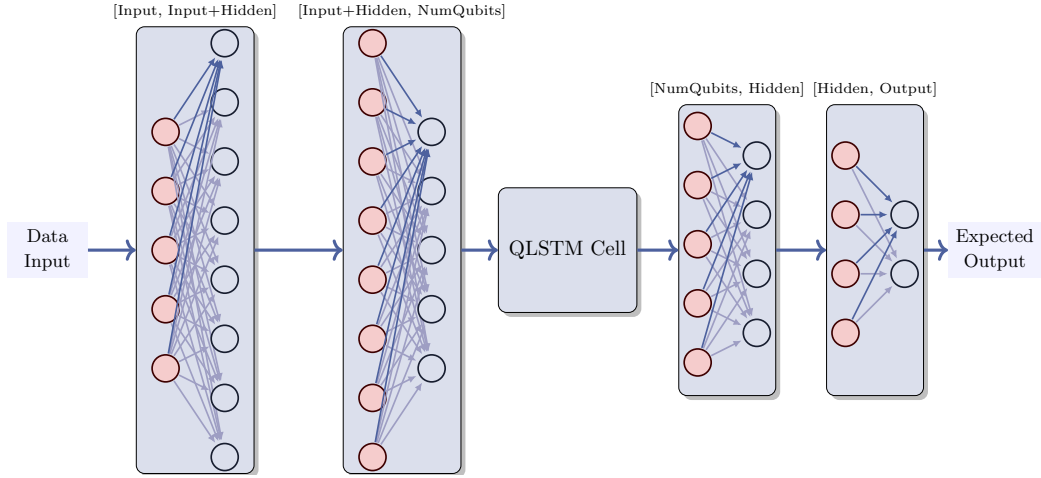


Figure 7.43: **QLSTM Architecture.** The input data passes through an initial classical layer, which receives the inputs data and produces the concatenated size formed by the input dimension and hidden size. This output then passes through a second classical layer, which outputs the same size as the number of qubits expected by the Quantum Qlstm cell, whose architecture is detailed in Figure 2.2. Subsequently, this output is received by another classical layer that transform it into output of the hidden size. Finally, this output is further transformed into the expected output.

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 7.46). 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  $463.6 \text{ m}^2$  ( $5000 \text{ feet}^2$ ).

A state  $s$  within this environment encapsulates a series of past observations

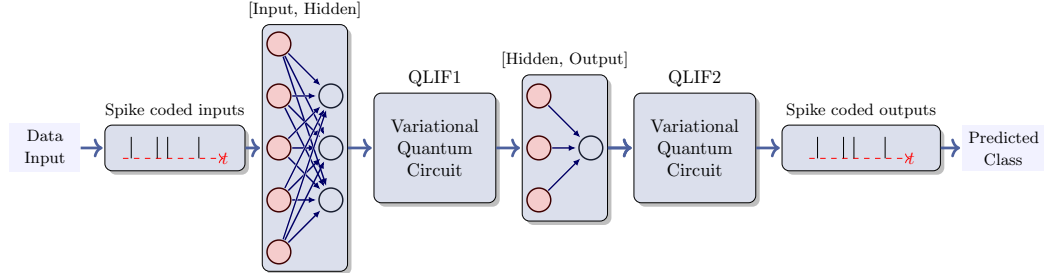


Figure 7.44: **QSNN Architecture.** This architecture involves an encoding step where classical data is translated into spikes. The network comprises two LIFs orchestrated by VQC and is trained using gradient descent. Various encoding strategies can be utilized, including the utilization of the highest firing rate or firing first, among others.

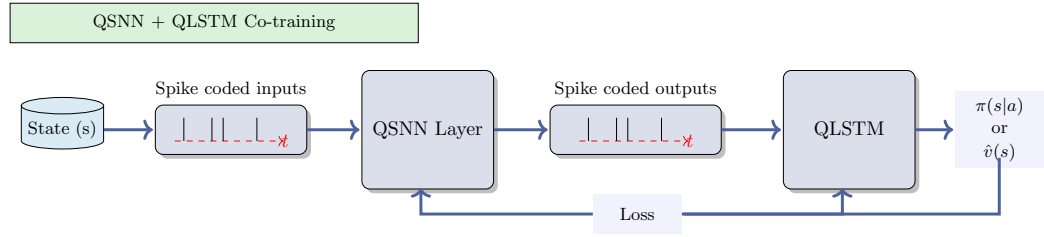


Figure 7.45: **Training Process of QSNN-QLSTM.** An instance of QSNN (see Figure 7.44) is combined with a QLSTM instance (refer to Figure 7.43) for joint training. The loss computed from the output of QLSTM is utilized for the concurrent update of QSNN and QLSTM.

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:

- **State space:** The state space encompasses 17 attributes; with 14 detailed in Table 7.22, 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 7.23. The temperature bounds for heating and cooling are  $[12, 23.5]$  and  $[21.5, 40]$  respectively.

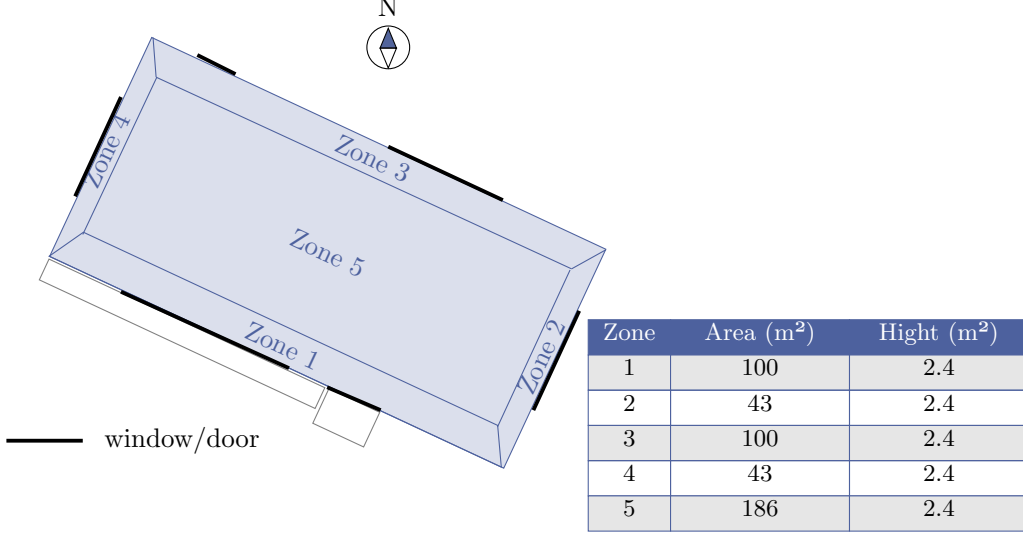


Figure 7.46: **Zone building plan.**

- **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

$$r_t = -\omega\lambda_P P_t - (1 - \omega)\lambda_T (|T_t - T_{up}| + |T_t - T_{low}|) \quad (7.58)$$

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);  $\omega$  represents the weight assigned to power consumption (and consequently,  $1 - \omega$ , the comfort weight), and  $\lambda_P$  and  $\lambda_T$  are scaling constants for consumption and comfort, respectively [130].

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

Table 7.22: 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 <sup>2</sup>  |
| Site-Direct-Solar-Radiation-Rate-per-Area                       | W/m <sup>2</sup>  |
| 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 7.23: 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                         |

1 output for the critic. This agent underwent training within the *Eplus-5zone-hot-discrete-v1* environment using the Advantage Actor-Critic (A2C) algorithm [39]. Additionally, we trained a classic Spiking Neural Network (SNN) and a classic long short-term memory (LSTM) both described in Section 7.3.3. Subsequently, four different quantum agents were trained using the methodologies described in Section 7.3.3, 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.

## 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 7.24 and 7.25 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)  $\times$  3 (rotations)  $\times$  5 (qubits) + 32  $\times$  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  $\times$  450 (weights) + 450 (bias) + 450  $\times$  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 7.3.3, 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:  $17 \times 15$  (weights) + 15 (bias) + 15 (layers)  $\times$  3 (learning parameters used in the three rotation gates for the ansatz circuit)  $\times$  5 (qubits)  $\times$  2 (number of QLIF cells) +  $15 \times 10$  (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:  $17 \times 15$  (weights) + 15 (bias) +  $15 \times 10$  (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 7.3.3, involves an actor component consisting of 1.481 parameters. This model includes a Linear layer that transforms an input size of 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):  $17 \times 42$  (weights) + 42 (bias) +  $42 \times 5$  (weights) + 5 (bias). It then incorporates the parameters for the corresponding Variational Quantum Circuits (VQCs):  $4 \times [5$  (layer)  $\times$  5 (qubits)  $\times$  1 (learning parameter employed in the Rotation  $\times$  gate for the ansatz circuit)]. Following this, another Linear layer transforms the number of qubits into the hidden size:  $5 \times 25$  (weights) + 25 (bias). The final layer is a Linear transformation from hidden size to output dimension:  $25 \times 10$  (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).  $4 \times (17 + 25 + 1) \times 25 + 4 \times$  hidden (for the first LSTM layer) +  $[4 \times (25 + 25 + 1) \times 25 + 4 \times$  hidden (for the rest of LSTM layers)]  $\times 4 + 25 \times 10 + 10$  (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 7.3.3, 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:  $[(17 \times 15 + 15) + (5 \text{ (layers)} \times 3 \text{ (learning parameters utilized in the three rotation gates within the ansatz circuit)} \times 5 \text{ (qubits)} \times 2 \text{ (number of QLIF cells)}) + (15 \times 10 + 10)] + [(15 \times 140 \text{ (concat size)} + 140) + (140 \times 5 + 5) + (5 \text{ layers} \times 4 \text{ (VQCs)} \times 1 \text{ (rotation gate)} \times 5 \text{ qubits}) + (5 \times 125 + 125 \text{ (for a Linear layer)}) + (125 \times 10 + 10 \text{ (for the output layer)})]$ . 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 7.24: 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                                              |
|------------------------|---------------------------------------------------|---------------------------------------------------------|---------------------------------------------------|
| <b>Optimizer</b>       | Adam(lr=1e-4)                                     | Adam(lr=1e-3)                                           | Adam(lr=1e-3)                                     |
| <b>Batch Size</b>      | 32                                                | 16                                                      | 32                                                |
| <b>BetaEntropy</b>     | 0.01                                              | 0.01                                                    | 0.01                                              |
| <b>Discount Factor</b> | 0.98                                              | 0.98                                                    | 0.98                                              |
| <b>Steps</b>           | -                                                 | 15                                                      | -                                                 |
| <b>Hidden</b>          | -                                                 | 15                                                      | 25                                                |
| <b>Layers</b>          | Actor: [Linear[17, 450], ReLU<br>Linear[450, 10]] | Actor: [Linear[17, 15]<br>Lif1, Lif2<br>Linear[15, 10]] | Actor:[LSTM(17, 25, layers=5)<br>Linear[25, 10]]  |
|                        | Critic: [Linear[17, 450], ReLU<br>Linear[450, 1]] | Critic: [Linear[17, 15]<br>Lif1, Lif2<br>Linear[15, 1]] | Critic: [LSTM(17, 25, layers=5)<br>Linear[25, 1]] |

Table 7.25: 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                                                             |
|--------------------------|---------------------------------------------------------|-------------------------------------------------------|----------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------|
| <b>Optimizer</b>         | Adam(lr=1e-2)                                           | Adam(lr=1e-3)                                         | Adam(lr=1e-3)                                                                                            | [(Adam(QSNN.parameters,lr= 1e-2),<br>Adam(QLSTM.parameters, lr=1e-2))] |
| <b>Batch Size</b>        | 128                                                     | 16                                                    | 128                                                                                                      | 128                                                                    |
| <b>Pre-processing</b>    | Normalization                                           | Normalization                                         | -                                                                                                        | Normalization                                                          |
| <b>Post-processing</b>   | -                                                       | -                                                     | -                                                                                                        | -                                                                      |
| <b>BetaEntropy</b>       | 0.01                                                    | 0.01                                                  | 0.01                                                                                                     | 0.01                                                                   |
| <b>Discount Factor</b>   | 0.98                                                    | 0.98                                                  | 0.98                                                                                                     | 0.98                                                                   |
| <b>Steps</b>             | -                                                       | 15                                                    | 15                                                                                                       | 15                                                                     |
| <b>Hidden</b>            | -                                                       | 15                                                    | 25                                                                                                       | 15 (QSNN), 125 (QLSTM)                                                 |
| <b>Layers</b>            | Actor: [[5 QNN]<br>ReLU<br>Linear[2 <sup>N</sup> , 10]] | Actor: [ Linear[17, 15]<br>15 QSNN<br>Linear[15, 10]] | Actor: [ Linear[17, 42]<br>Linear[42, 5]<br>4 VQCs<br>Linear[25, 5]<br>Linear[5, 25]<br>Linear[25, 10]]  | 5 QSNN<br>5 QLSTM                                                      |
|                          | Critic: [[5 QNN<br>ReLU<br>Linear[2 <sup>N</sup> , 1]]  | Critic: [ Linear[17, 15]<br>15 QSNN<br>Linear[15, 1]] | Critic: [ Linear[17, 42]<br>Linear[42, 5]<br>4 VQC'S<br>Linear[25, 5]<br>Linear[5, 25]<br>Linear[25, 1]] |                                                                        |
| <b>Qubits</b>            | 5                                                       | 5                                                     | 5                                                                                                        | 5                                                                      |
| <b>Encoding Strategy</b> | Amplitud Encoding                                       | Amplitud Encoding                                     | Angle Encoding                                                                                           | Amplitud Encoding                                                      |

## Results

The results of this use case are outlined in Table 7.26, 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 consistently 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 7.47), 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 7.47, 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 7.48, 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 7.26: 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    |
| QNN       | -11.12  |            | -9.89  |            | -12.39 |            | -13.32 |        | 467.4    |
| QLSTM     | -10.72  |            | -10.14 |            | -11.14 |            | -12.19 |        | 335.42   |
| QNN-QLSTM | -9.40   |            | -8.26  |            | -12.73 |            | -11.83 |        | 962.5    |

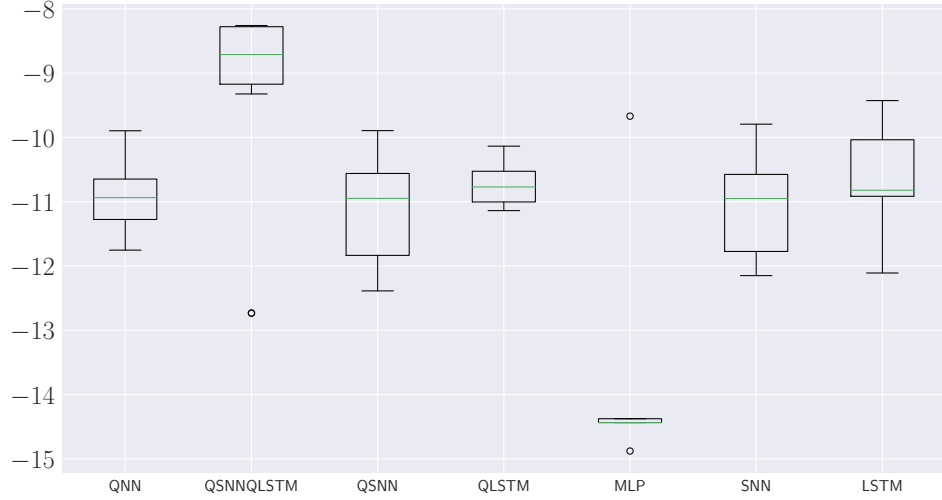


Figure 7.47: Boxplots that depict the distribution of average total reward achieved by quantum (QNN, QSNN, QLSTM and QSNN-QLSTM) and classical models (MLP, SNN and LSTM).

### 7.3.5 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, the 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 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. Furthermore, the cost of computational time is notable, resulting in longer experiments and the finding of optimal hyperparameters. These constraints emphasize

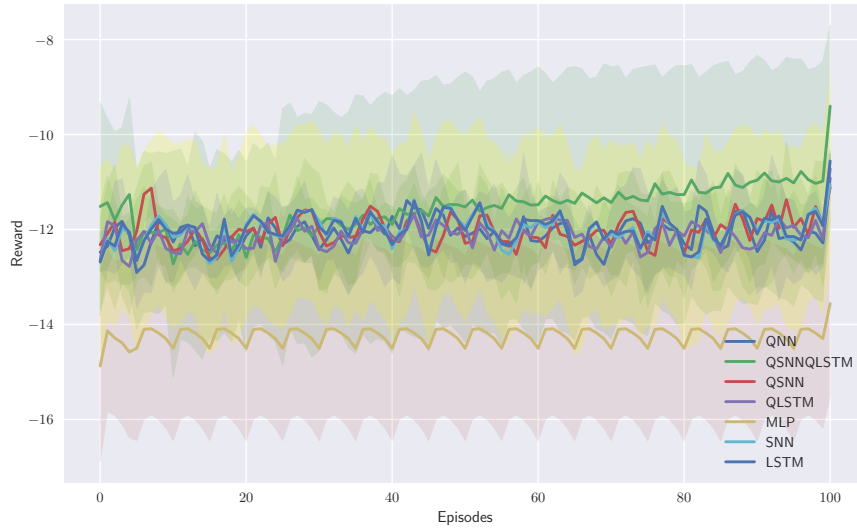


Figure 7.48: Learning curves obtained from seven models. Three classical models: MLP, SNN and LSTM. Four quantum models: QNN, QSNN, QLSTM and new brain-inspired model QSNN-QLSTM.

the necessity for continual advancements in both quantum hardware and simulators to enable decreased computation times.

### 7.3.6 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 [11]. 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) [158].

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 [121]. Nevertheless, this capability does not always occur [122], and exploring the distinctions is of significance to both AI and neuroscience [123].

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 [10].

Finally, in the advancement of Quantum Neural Networks, the development of new training algorithms that harness the principles of Quantum Computing, alongside new brain-inspired models such as the one presented here, will be crucial to enhance convergence and reduce execution times. However, the most significant impact on the progress of Quantum Neural Networks will heavily depend on hardware development, as it accelerates experiments and facilitates drawing conclusions

## Chapter 8

# Brain-inspired for enhanced pattern recognition: Unifying Quantum Spiking Neural Networks and QLSTM

This chapter introduces the current work, which is presently under review by the journal Quantum Machine intelligence(Springer Nature). The study explores the application of a variation of the novel brain-inspired model introduced in Section 7.3 to additional use cases, such as anomaly detection, aiming to gain insights into its generalization, adaptability, and learning process. In this context, we seek to validate our hypothesis, which suggests that this model can learn, memorize, and filter out noisy and infrequent events while strengthening information with stronger space-time correlations, thereby aiding in the mitigation of BPs. Through this chapter, our goal is to elucidate the purpose and anticipated outcomes of the study.

|                                                                                                                                                                   |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Reference</b>                                                                                                                                                  |
| E. Andrés, M. P. Cuéllar, and G. Navarro.<br>Brain-inspired for enhanced pattern recognition: Unifying Quantum Spiking Neural Networks and QLSTM.                 |
| <b>Status</b>                                                                                                                                                     |
| Peer-Review                                                                                                                                                       |
| <b>Impact factor</b>                                                                                                                                              |
| IF: 4.8<br>H-Index: 18                                                                                                                                            |
| <b>Category</b>                                                                                                                                                   |
| Computational Theory and Mathematics (Q1); Software (Q1); Theoretical Computer Science (Q1); Artificial Intelligence (Q2), Applied Mathematics (Q1)<br>SJR: 1.025 |
| <b>DOI</b>                                                                                                                                                        |
|                                                                                                                                                                   |
| <b>Journal-Editorial</b>                                                                                                                                          |
| Quantum Machine Intelligence-SpringerNature                                                                                                                       |
| <b>Logo</b>                                                                                                                                                       |
|                                                                                |

## *Brain-inspired for enhanced pattern recognition*

**abstract:**Recent advances in the fields of deep learning and Quantum Computing have paved the way for innovative developments in artificial intelligence. In this manuscript, we leverage these cutting-edge technologies to introduce a novel model that emulates the intricate functioning of the human brain, designed specifically for the detection of anomalies such as fraud in credit card transactions. Leveraging the synergies of Quantum Spiking Neural Networks (QSNN) and Quantum Long Short-Term Memory (QLSTM) architectures, our approach is developed in two distinct stages, closely mirroring the information processing mechanisms found in the brain's sensory and memory systems. In the initial stage, similar to the brain's hypothalamus, we extract low-level information from the data, emulating sensory data processing patterns. In the subsequent stage, resembling the hippocampus, we process this information at a higher level, capturing and memorizing correlated patterns. We will compare this model with other quantum models such as Quantum Neural Networks among others and their corresponding classical models.

**keywords**Anomaly Detection, Quantum Neural Networks, Quantum Spiking Neural Quantum Long Short-Term Memory, Brain-inspired models

### **8.0.1 Introduction**

In recent years, the field of artificial intelligence has witnessed remarkable advancements. A notable outcome of this progress is the emergence of brain-inspired artificial intelligence, a field that amalgamates insights from neuroscience, psychology, and computer science to develop more efficient and powerful AI systems [9]. Additionally, quantum machine learning has emerged as a compelling area of interest within the scientific community. Notably, Variational Quantum Circuits (VQC) also known as Quantum Neural Networks (QNN) have demonstrated significant success across diverse domains, including supervised learning [22, 23, 24] and unsupervised learning [25]. While research on Quantum Reinforcement Learning (QRL) is still in its early stages, recent studies have demonstrated that QNNs can outperform classical models in Reinforcement learning (RL) scenarios, achieving better cumulative rewards

with fewer parameters to be learned [139].

In classical machine learning, challenges persist when working with imbalanced datasets, limited data availability, or low data quality. Our aim is to investigate whether quantum models can yield improved results under such conditions. Moreover, we seek to explore the intriguing question of whether Quantum Brain-Inspired Models are capable of detecting patterns that elude classical and Quantum Neural Networks (QNN) models, potentially leading to enhanced learning outcomes and improved false positive and false negative rates.

Drawing inspiration from a theory of PreFrontal Cortex and hippocampus function, the hippocampus plays a role in the formation and retrieval of specific memories. Meanwhile, the PreFrontal Cortex accumulates features of related memories, shaping the 'context' of interconnected experiences. This context might encompass details like a list in which a set of words appeared, a common location for multiple events, or a shared set of ongoing task rules governing memory-related decisions. Consequently, when cued to a particular context, the PreFrontal Cortex influences the retrieval of memories pertinent to a context within the hippocampus and other brain regions [160].

And finally, motivating this study we consider the widely accepted memory model of Atkinson and Shiffrin [2], including Short-Term Store (STS) consisting of the memory in storage for a short amount of time, and Long-Term Store (LTS) refers to the memory maintained for long periods of time [161, 162].

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. Additionally, In the realm of traditional neural networks, the temporal constants of neurons and synapses typically span time scales ranging from 1 to 100 milliseconds. However, challenges arise when dealing with problems that necessitate long-term associations exceeding the slowest neuron or synaptic time constant. Such challenges are prevalent in natural language processing and Reinforcement Learning, crucial for understanding human behavior and decision-making. This imposes a substantial burden on the learning process, where vanishing gradients impede the convergence of

neural networks [73].

To address these challenges, Long Short-Term Memory networks (LSTMs) and later, Gated Recurrent Units (GRUs) [176], introduced slow dynamics designed to overcome memory and vanishing gradient issues in traditional Recurrent Neural Networks (RNNs). For Spiking Neural Networks, complementing fast neural dynamics with a variety of slower dynamics becomes essential to effectively tackle long-range association problems [73].

Motivated by this, the exploration of the union between Quantum Spiking Neural Networks (QSNN) and Quantum Long Short-Term Memory (QLSTM) becomes compelling. The quantum nature of these architectures holds the promise of overcoming temporal limitations, offering diverse dynamics adaptable to a broad range of time scales. This innovative combination not only has the potential to address persistent challenges in long-term association learning but may also unlock new possibilities in real-time dynamic information processing, providing a significant boost to the efficiency and adaptability of neural networks across various applications.

In this manuscript, we will conduct a comparative study in the context of anomaly detection, specifically targeting fraud detection in bank account transactions. We will evaluate three different architectures, including Spiking Neural Networks (SNN) and Artificial Neural Networks (ANN), as well as quantum architectures such as Quantum Spiking Neural Networks (QSNN) and Quantum Neural Networks (QNN). Additionally, we will explore the synergy between these quantum architectures and Quantum Long Short-Term Memory (QLSTM). Our research will focus on determining which of these architectures exhibits optimal performance in a scenario characterized by highly imbalanced data and limited data availability.

## 8.0.2 Background

For article self-completeness, this section provides a concise introduction to Spiking Neural Networks (SNNs), Long Short-Term Memory, and Quantum Neural Networks (QNNs), which constitute the primary topics covered in this work.

Firstly, in Section 8.0.2, we delve into the fundamentals of Spiking Neural Networks. This section offers an overview of the basic structure and operation of SNNs, alongside a discussion of Hebbian Theory.

Subsequently, in Section 8.0.2, we explore the architecture and operation of Long Short-Term Memory networks.

Lastly, in Section 8.0.2, we introduce Quantum Neural Networks, elucidating their key concepts and principles.

## Spiking Neural Networks

SNNs are inspired by the brain's communication scheme between neurons, similar to the encoding and maintenance of working memory or short-term memory in the PreFrontal Cortex, as well as 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 8.2 [72].

Although, DNNs are historically brain-inspired, there are fundamental differences in their structure, neural computations, and learning rule compared to the brain. One of the most important differences is the way that information propagates between their units. It is this observation that leads to the realm of Spiking Neural Networks (SNNs). In the brain the communication between neurons is done by broadcasting trains of potentials or spike trains to downstream neurons. These individual spikes are sparse in time, so each spike has high information content. Thus, information in SNNs is conveyed by spike timing, including latencies, and spike rates. In a biological neuron, a spike is generated when the running sum of changes in the membrane potential, which can result from pre-synaptic stimulation, crosses a threshold. The rate of spike generation and the temporal pattern of spike trains carry information about external stimuli and ongoing calculations.

ANNs communicate using continuous-valued activations and, for that reason, SNNs are more efficient because, as explained below, spike events are sparse in time. Additionally, SNNs also have the advantage of being intrinsically sensitive to the temporal characteristics of information transmission that occurs in the biological neural systems. The precise timing of every spike

is highly reliable for several areas of the brain, suggesting an important role in neural coding (in sensing information-processing areas and neural-motor areas) [73, 74]. SNNs have applications in many areas of pattern recognition such as visual processing, speech recognition and medical diagnosis. Deep SNNs are great candidates to investigate neural computation and different coding strategies in the brain. Training deep Spiking Neural Networks is in its early phases and there is an important open question about their training such as let on-line learning while avoiding catastrophic forgetting [75].

Spiking neurons operate on a weighted sum of inputs. Rather than passing the result through a sigmoid or ReLU non-linearity, the weighted sum contributes to the membrane potential  $U(t)$  of the neuron. If the neuron is sufficiently excited by this weighted sum, and the membrane potential reaches a threshold  $\theta$ , then the neuron will emit a spike to its subsequent connections. But most neuronal inputs are spikes of very short bursts of electrical activity. It is quite unlikely for all input spikes to arrive at the neuron body in unison. This indicates the presence of temporal dynamics that ‘sustain’ the membrane potential over time. Figure 8.1.

Louis Lapicque [76] observed that a spiking neuron can be roughly compared to a low-pass filter circuit, consisting of a resistor (R) and a capacitor (C). This concept is known as the leaky integrate-and-fire neuron model. Even today, this idea remains valid. Physiologically, the neuron’s capacitance is due to the insulating lipid bilayer forming its membrane, while the resistance comes from gated ion channels that regulate the flow of charged particles across the membrane (see Figure 8.1b).

The behavior of this passive membrane can be described using an RC circuit, following Ohm’s Law. It states that the membrane’s potential, measured between the input and output of the neuron, is proportional to the current passing through the conductor.

$$I_{in}(t) = \frac{U(t)}{R} \quad (8.1)$$

The dynamics of the passive membrane modelled using an RC circuit can be represented as:

$$\tau \frac{dU(t)}{dt} = -U(t) + I_{in}(t)R \quad (8.2)$$

where  $\tau = RC$  is the time constant of the circuit. Following the Euler method,

$dU(t)/dt$  without  $\Delta t \rightarrow 0$ :

$$\tau \frac{U(t + \Delta t) - U(t)}{\Delta t} = -U(t) + I_{in}(t)R \quad (8.3)$$

Isolating the membrane potential at the next step:

$$U(t + \Delta t) = \left(1 - \frac{\Delta t}{\tau}\right) U(t) + \frac{\Delta t}{\tau} I_{in}(t)R \quad (8.4)$$

To single out the leaky membrane potential dynamics, assume there is no input current  $I_{in} = 0$ :

$$U(t + \Delta t) = \left(1 - \frac{\Delta t}{\tau}\right) U(t) \quad (8.5)$$

$\beta = U(t + \Delta t)/U(t)$  would be the decay rate of the membrane potential, also known as the inverse time constant. From the previous equation, this implies that  $\beta = 1 - \Delta t/\tau$ .

Let us assume that  $t$  is discretized into sequential time-steps, such that  $\Delta t = 1$ . To further reduce the number of hyperparameters assume  $R = 1 \Omega$ . Then

$$\beta = 1 - \frac{1}{\tau} \Rightarrow U(t + 1) = \beta U(t) + (1 - \beta)I_{in}(t + 1) \quad (8.6)$$

For a constant current input, this can be solved as

$$U(t) = I_{in}(t)R + [U_0 - I_{in}(t)R]e^{-t/\tau} \quad (8.7)$$

which shows how exponential relaxation of  $U(t)$  to a steady state value follows current injection, where  $U_0$  is the initial membrane potential at  $t = 0$

$$U(t) = U_0 e^{-t/\tau} \quad (8.8)$$

Assuming Eq. (8.8) is computed at discrete steps of  $t, (t + \Delta t), (t + 2\Delta t) \dots$ , then the ratio of membrane potential across two subsequent steps can be calculated using:

$$\beta = \frac{U_0 e^{-(t+\Delta t)/\tau}}{U_0 e^{-t/\tau}} = \frac{U_0 e^{-(t+2\Delta t)/\tau}}{U_0 e^{-(t+\Delta t)/\tau}} = \dots \Rightarrow \beta = e^{-\Delta t/\tau} \quad (8.9)$$

This equation for  $\beta$  is more precise than  $\beta = (1 - \Delta t/\tau)$ , as the latter is only precise for  $\Delta t \ll \tau$ .

A second non-physiological assumption is made, where the effect of  $(1 - \beta)$  is absorbed by a learnable weight  $W$  (in deep learning, the weighting factor of an input is typically a learnable parameter):

$$WX(t) = I_{in}(t) \quad (8.10)$$

$X(t)$  is an input voltage, spike, or unweighted current, and is scaled by the synaptic conductance  $W$  to generate a current injection to the neuron. This leads to the following result:

$$U(t+1) = \beta U(t) + WX(t+1) \quad (8.11)$$

where the effects of  $W$  and  $\beta$  are decoupled, thus favouring simplicity over biological precision. Finally, a reset function is appended, which activates every time an output spike is triggered:

$$U(t) = \underbrace{\beta U(t-1)}_{\text{decay}} + \underbrace{WX(t)}_{\text{input}} - \underbrace{S_{out}(t-1)\theta}_{\text{reset}} \quad (8.12)$$

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

A spike is generated if the membrane potential exceeds the threshold:

$$S_{out}(t) = \begin{cases} 1, & \text{if } U(t) > \theta \\ 0, & \text{otherwise} \end{cases} \quad (8.13)$$

There are several methods to training SNN's [73], the more commonly adopted approach is the backpropagation using spikes i.e BackPropagation Through Time (BPTT). Working backwards from the final output of the network, the gradient flows from the loss to all descents. The goal is to train the network using the gradient of the loss with respect to the weights, such that the weights are updated to minimize the loss. The backpropagation algorithm achieves this using the chain rule:

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial S} \underbrace{\frac{\partial S}{\partial U}}_{\{0, \infty\}} \frac{\partial U}{\partial I} \frac{\partial I}{\partial W} \quad (8.14)$$

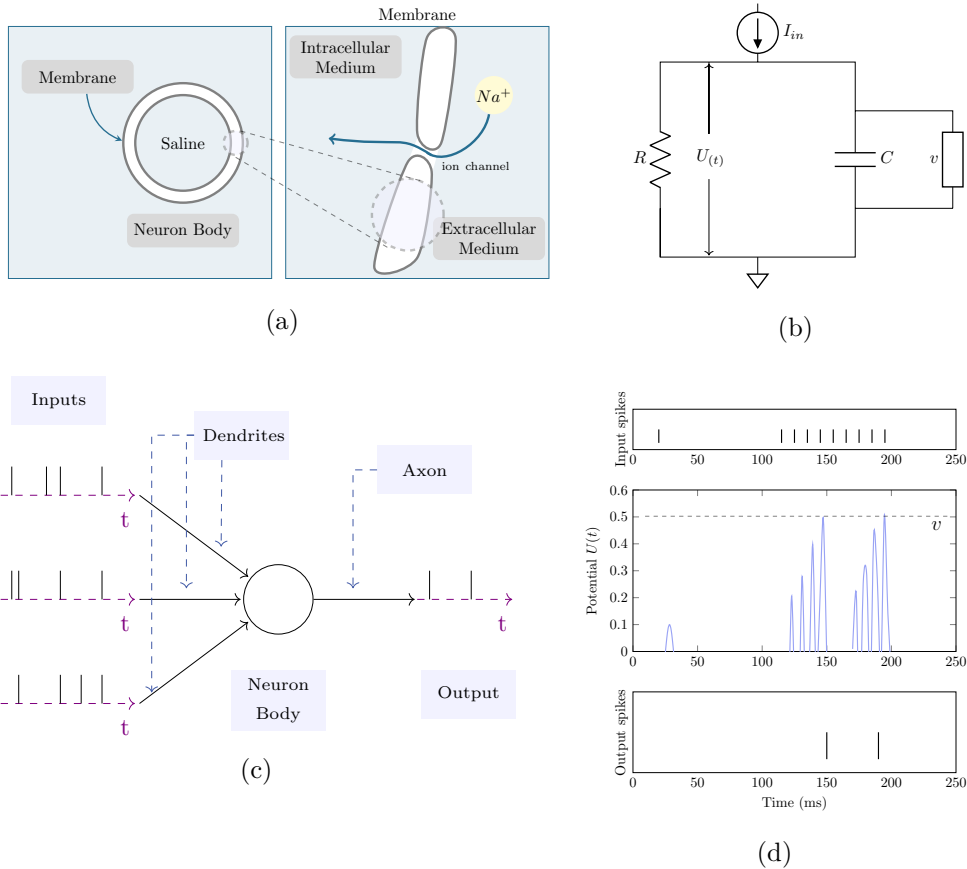


Figure 8.1: The Leaky Integrate-and-Fire Neuron Model [73] involves an insulating lipid bilayer membrane that separates the interior and exterior environments. Gated ion channels enable the diffusion of charge carriers, such as  $Na^+$ , through the membrane.8.1a. This neuron function is often modeled using an RC circuit. When the membrane potential exceeds the threshold, a spike is generated 8.1b. Input spikes are transmitted to the neuron body via dendritic branches. Accumulation of sufficient excitation triggers spike emission at the output 8.1c. A simulation depicting the membrane potential  $U(t)$  reaching a threshold of  $\theta = 0.5V$ , leading to the generation of output spikes 8.1d

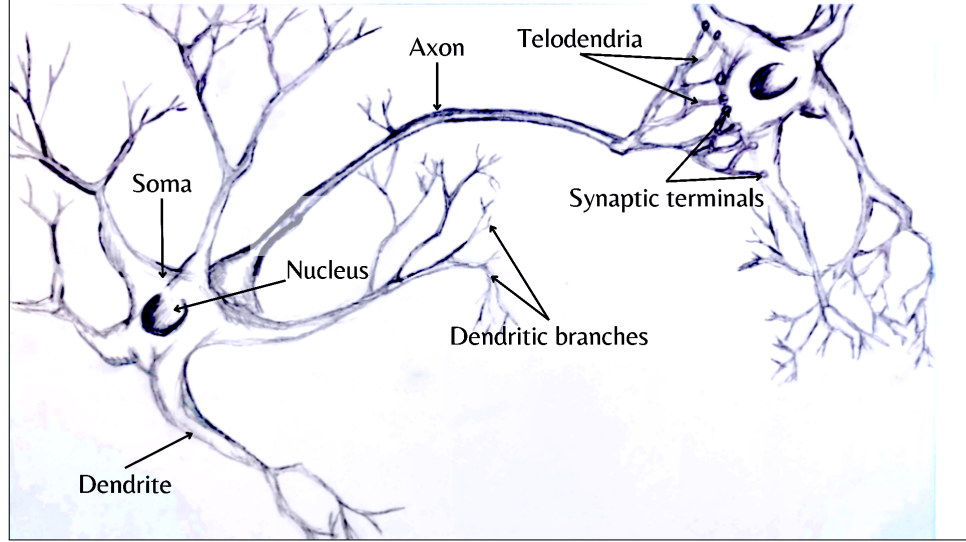


Figure 8.2: Typical morphology of a neuron. Consisting of a cell body, or soma, which contains the nucleus and other organelles, dendrites, which are fine, branched cell processes that receive synaptic input from other neurons, one axon and synaptic terminals.

However, the derivative of the Heaviside step function from (8.13) is the Dirac Delta function, which evaluates to 0 everywhere, except at the threshold  $U_{\text{thr}} = \theta$ , where it tends to infinity. This means the gradient will almost always be nulled to zero (or saturated if  $U$  sits precisely at the threshold), and no learning can take place. This is known as the dead neuron problem. The most common way to address the dead neuron problem is to keep the Heaviside function as it is during the forward pass, but during the backward, substitute the heaviside operator with a continuous function,  $\tilde{S}$  the derivative of the continuous function is used as a substitute  $\partial S / \partial U \leftarrow \partial \tilde{S} / \partial U$ , and is known as the surrogate gradient. In this manuscript we use `snntorch` library which uses arctangent as a default function [73].

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 descent method. The figure 8.3 shown the general pipeline for this model for a classification task and the detailed architecture is defined in section 8.0.3.

Previous works have been made inspiring in brain functionality emulat-

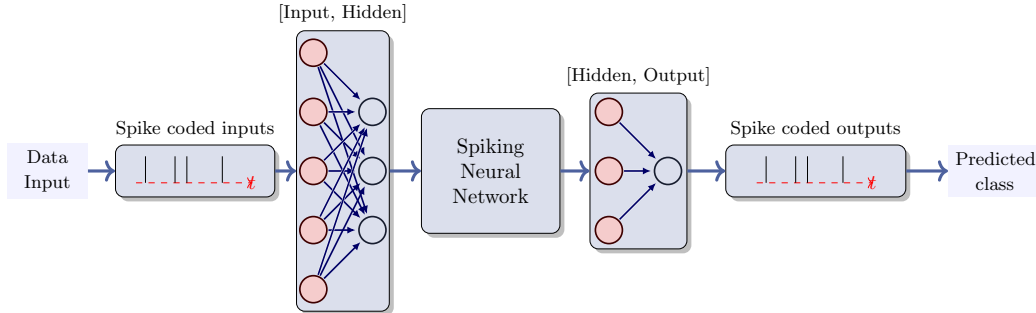


Figure 8.3: **SNN Pipeline.** Input data for an SNN can be transformed into a firing rate or other encodings to generate spikes. The network is subsequently trained to predict the correct class, employing encoding strategies such as the highest firing rate or firing first, among others

ing for clasification tasks such as MINST dataset classication using SNN and HyperDimensional Computing [77] or decoding and understanding muscle activity and kinematics from electroencephalography signals [78], using HyperDimensional Computing and Spiking (HDC) and SNN for MNIST classification problem [77] or using Reinforcement Learning for navigation in changeable and unfamiliar environments supporting neuroscientific theories that see grid cells as critical for vector-based navigation [79].

## Long Short-Term Memory

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. They are designed to overcome the RNN (Recurrent Neural Networks) problems in training because retro-propagated gradients tend to grow enormously or fade over time because the gradient depends not only on the present error but also on past errors. The accumulation of errors makes it difficult to memorize long-term dependencies. Therefore, these problems are solved by the Long Short-Term Memory neural networks (LSTM), for which it incorporates a series of steps to decide which information will be stored and which erased [82]. Additionally, the range of contextual information that standard RNNs can access in practice is quite limited. The influence of a given input on the hidden layer and, therefore, on the network output, either decays or blows up exponentially as it cycles around the network's recurrent connections. This is the vanishing gradient problem, being the second problem to overcome using

LSTM [84, 85].

The behavior of this model is required in complex problem domains like machine translation, speech recognition and time-series among others.

These networks are composed of LSTM modules which are a special type of recurrent neural network described in 1997 by Hochreiter and Schmidhuber [86]. They consist of three internal gates, known as input, forget and output gates, detailed in Figure 8.4.

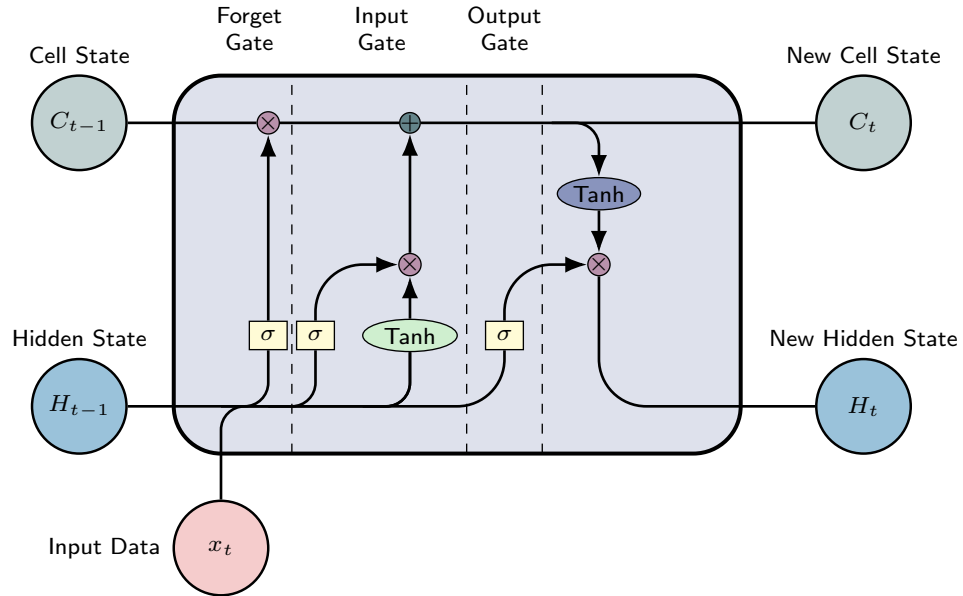


Figure 8.4: **LSTM Cell Architecture:** Featuring three essential gates (forget, input and output gates). The  $\sigma$  and  $\tanh$  blocks symbolize the sigmoid and hyperbolic tangent activation functions, respectively.  $x_t$  denotes the input at time  $t$ ,  $h_t$  represents the hidden state, 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. The output of an LSTM at a particular point in time is dependant on three things:

- Cell state: The current long-term memory of the network
- Hidden state: The output at the previous point in time

- The input data at the current time step

The internal gates mentioned above can be described as follows [163]:

- **Forget Gate:** This gate determines which data from the cell state is relevant based on the previous hidden state and the new input data. The neural network that implements this gate is build to achieve an output closer to 0 when the input data is considered irrelevant, and closer to 1 otherwise. To achieve this, we employ the sigmoid activation function. The output values from this gate are then sent upward and subjected to pointwise multiplication with the previous cell state. This pointwise multiplication implies that components of the cell state deemed irrelevant by the forget gate network will be multiplied by a number close to 0, consequently having less influence on subsequent steps.

In summary, the forget gate decides which pieces of the long-term memory should now be forgotten (have less weight) given the previous hidden state and the new input data.

- **Input gate:** Determines which new information should be incorporated into the network's long-term memory (cell state), based on the previous hidden state and new input data. The same inputs are utilize, but with the addition of a hyperbolic tangent as the activation function. This hiperbolic tangent has learned to combine the previous hidden state and new input data, resulting in a new memory update vector. Essentially, this vector contains information from the new input data within the context of the previous hidden state. This vector tells us how much to update each component of the long-term memory (cell state) of the network given the new data.

Note 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.

- **Output gate:** the objective of this gate is to decide the new hidden state by incorporating the newly updated cell state, the previous 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 final 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 8.8 shows.

Finally, we summarise the Lstm implementation steps as follows:

- The first step consists of deciding which information is to be forgotten or retained in the particular time instant, to achieve this, the sigmoid function is employed. It analyzes the previous state  $h_{t-1}$  and the current input  $x_t$ , calculating the function accordingly:

$$f_t = \sigma(W_f \cdot v_t + b_f) \quad (8.15)$$

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

- In this step, the content of the memory cell is updated by selecting new information to be stored in the cell state. The second layer, known as the input gate, comprises two components: the sigmoid function and the hyperbolic tangent (tanh). The sigmoid layer determines which values are to be updated; 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 the selected values based on their importance (ranging from -1 to 1). These two components are then combined to update the state:

$$\begin{aligned} i_t &= \sigma(W_i \cdot v_t + b_i) \\ \tilde{C} &= \tanh(W_c \cdot v_t + b_c) \end{aligned} \quad (8.16)$$

- The third step consists of updating the old cell state,  $C_{t-1}$  with the new cell state,  $C_t$ , by multiplying the old state by  $f_t$  to forget irrelevant information, and then adding the candidate  $\tilde{C}_t$ :

$$C_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{C}_t \quad (8.17)$$

- Finally, the output is computed in two steps. Firstly, a sigmoid layer is used to select the relevant portions of the cell state to be transmitted to the output.

$$o_t = \sigma(W_o \cdot v_t + b_o) \quad (8.18)$$

The cell state is then processed through the tanh layer (to normalize values within the range of -1 and 1) and multiplied by the output of the sigmoid gate.

$$h_t = \tanh(C_t) \cdot o_t \quad (8.19)$$

## Quantum Neural Networks

Quantum Neural Networks (QNNs) play a pivotal role in Quantum Machine Learning (QML) [19, 20], utilizing Variational Quantum Circuits (VQCs). These VQCs are hybrid algorithms, combining quantum circuits with trainable parameters optimized using classical techniques. They are capable of approximating continuous functions [21, 19], enabling tasks such as optimization, approximation, and classification.

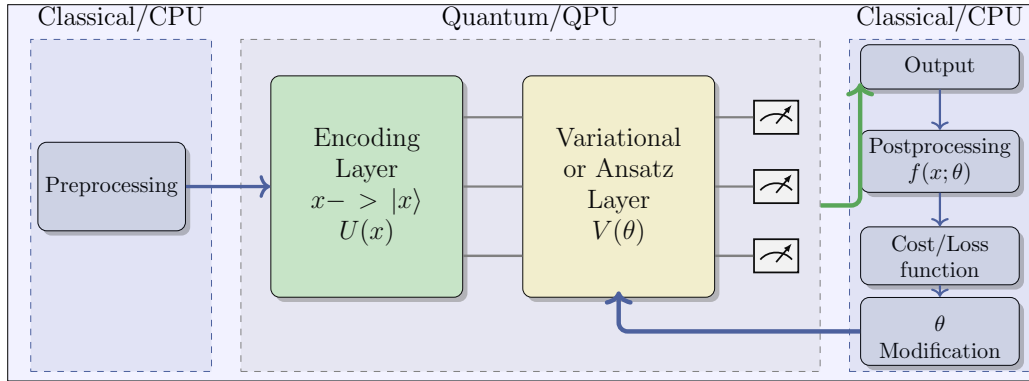


Figure 8.5: **General VQC Schema.** The dashed gray line encompasses the steps executed in a Quantum Processing Unit (QPU) and the dashed blue line shows the steps executed in a CPU.

Fig. 8.5 shows the general schema of a VQC. This hybrid methodology encompasses the following steps [28]:

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, including tensor product encoding and probability encoding [29].
3. **Variational Layer (QPU)**: This layer implements the behavior of Quantum Neural Networks using entanglement and rotation gates with learnable parameters, optimized by classical optimization algorithms.
4. **Measurement Process (QPU/CPU)**: Measuring the quantum state and decoding it to obtain the desired output. The choice of observables used for this process is crucial for performance.
5. **Post-processing (CPU)**: Transformation of QPU outputs before returning them to the user and incorporating them into the cost function during the learning process.
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 like COBYLA or SPSA can also approximate parameter updates.

### 8.0.3 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), and Quantum Long Short-Term Memory (QLSTM). Additionally, we introduce 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 [73]. Catastrophic forgetting, where new information causes the network to forget what it has previously learned [117] is especially problematic in real-time systems where the size of batch-size is 1. Several approaches to overcome catastrophic forgetting in

continual learning have been proposed, including using higher dimensional synapses [171], ensembles of networks [172], pseudo-replay [173] and penalizing weights that change excessively fast [174]. 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 8.6 illustrates this architecture.

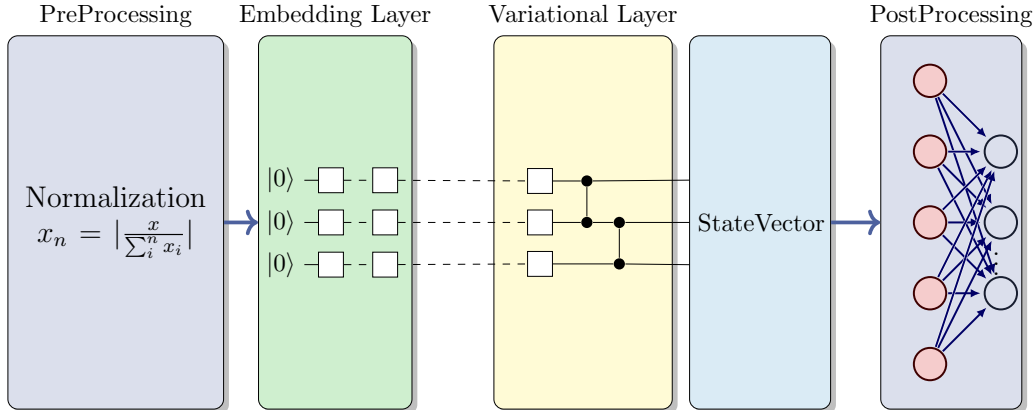


**Figure 8.6: QNN Architecture.** This architecture entails a pre-processing step where classical data undergoes normalization, a pre-requisite for the subsequent encoding strategy. Then, the amplitude encoding algorithm is applied, utilizing only  $\log_2 N$  qubits, where  $N$  corresponds to the number of independent variables. This process generates the corresponding quantum state, which is then fed into the ansatz circuit. Finally, the resulting state-vector of this quantum system is subject to 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 number of classes.

The second model is a Quantum Spiking Neural Network (QSNN), im-

plemented using the Python package `snnTorch` [175]. The architecture of the QSNN is illustrated in Figure 8.9. 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 using amplitude embedding to minimize qubit usage and an Ansatz composed of  $R_x$ ,  $R_y$ ,  $R_z$ , and  $C_z$  gates.

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 8.7 and the complete architecture in figure 8.8. 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  $R_x$  and  $C_x$  gates.

The fourth Quantum model is an innovative architecture that combines a Quantum Spiking Neural Network (QSNN) and Quantum Long Short-Term Memory (QLSTM). It undergoes co-training, updating gradients simultaneously through a multi-optimizer with distinct learning rates for each network’s parameters. The QSNN model has undergone pre-training. The training process of this model is illustrated in Figure 8.10

## 8.0.4 Experimentation

In this section, we conduct experiments to evaluate various models described in the previous section for anomaly detection, with a specific focus on fraud detection in credit card transactions. Our objective is to compare the performance of all proposed models and determine whether the novel model composed of QSNN and QLSTM achieves superior results. We performed 10 independent runs, each initiated with distinct random seeds, which were also used for dataset splitting. This approach ensures a diverse assessment across different partitions for each model.

Additionally, we utilize F1-score as the fundamental metric for the analysis and comparison of all models.

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

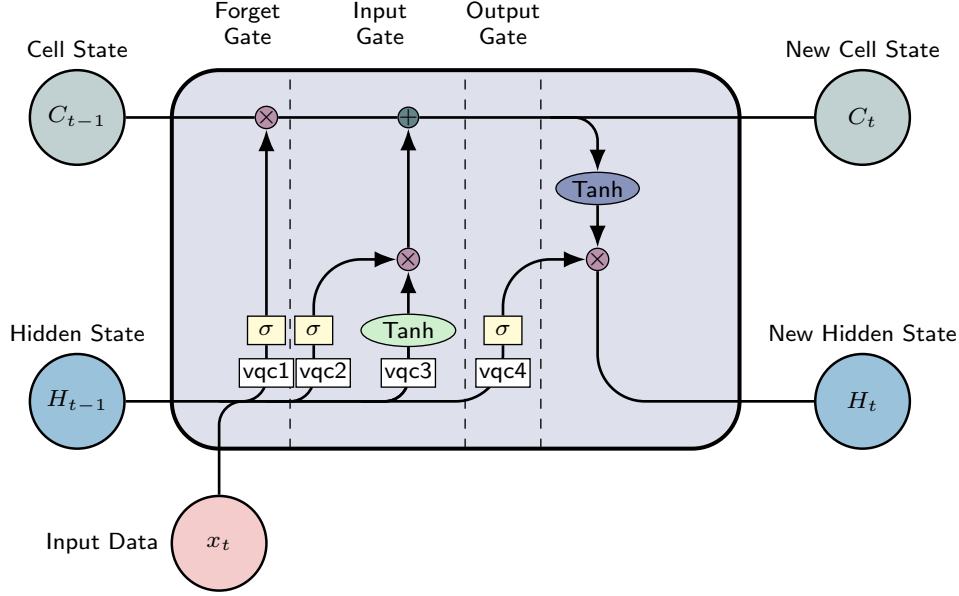


Figure 8.7: **QLSTM Cell.** Each VQC box is the form as detailed in Figure 8.5. The  $\sigma$  and  $\tanh$  blocks represent the sigmoid and the hyperbolic tangent activation function, respectively.  $x_t$  is the input at time  $t$ ,  $h_t$  is the hidden state and  $c_t$  is the cell state.  $\otimes$  and  $\oplus$  represents element-wise multiplication and addition, respectively.

## Problem Statement

For the detection of fraud in credit card transactions, we utilize a Kaggle dataset comprising 30 features, which include 28 anonymized features resulting from PCA transformation and the transaction amount. The dataset, documenting transactions made by European cardholders in September 2013, spans a two-day period and includes a total of 284,807 transactions. Among these, 492 are identified as frauds. It is noteworthy that the dataset exhibits a pronounced class imbalance, with the positive class (frauds) representing a mere 0.172% of the overall transactions.

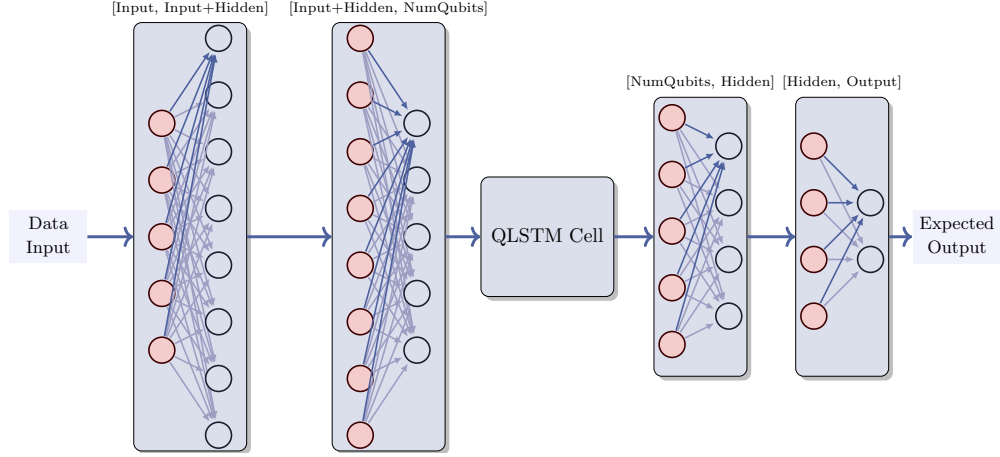


Figure 8.8: **QLSTM Architecture.** The input data passes through an initial classical layer, which receives the 30 inputs data and produces 80 outputs, corresponding to the concatenated size formed by the input dimension and hidden size. This output then passes through a second classical layer, which outputs the same size as the number of qubits expected by the Quantum Qlstm cell, whose architecture is detailed in Figure 2.2. Subsequently, this output is received by another classical layer that transform it into output of the hidden size. Finally, this output is further transformed into the number of classes, and by applying sigmoid activation function, we obtain the expected output  $y_{pred}$ .

## Experimental Settings

In this experiment, we conducted 10 independent runs, each initialized with distinct random seeds, ensuring the validity and robustness of our results. For each execution, we recorded the rates of true positives, true negatives, false positives and false negatives for f1-score calculation along with the training and validation losses, enabling a comprehensive analysis of their respective curves. Tables 8.2 and 8.3 outline the configuration of classical and quantum models, along with the corresponding hyperparameters used.

For classical models, we utilized a larger dataset consisting of 390 fraud cases paired with 5000 non-fraud cases for training, and 999 instances of non-fraud paired with 101 instances of fraud for test dataset. In the case of

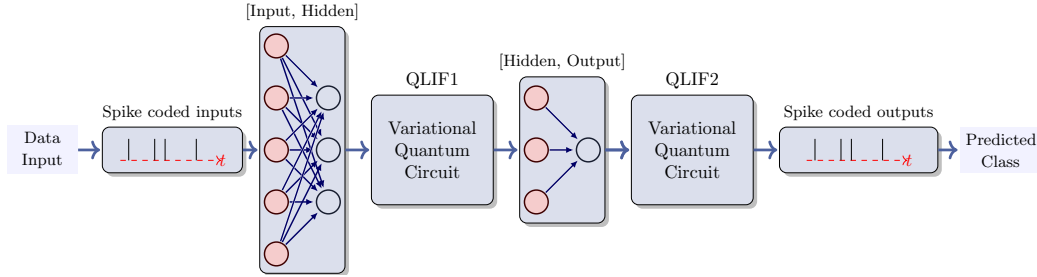


Figure 8.9: **QSNN Architecture.** This architecture involves an encoding step where classical data is translated into spikes. The network comprises two LIFs orchestrated by VQC and is trained using gradient descent to accurately predict the correct class using various encoding strategies, including the utilization of the highest firing rate or firing first, among others.

quantum models, we utilised 390 fraud cases along with 1000 non-fraud cases for training dataset, mirroring the size of classical model's test dataset. This highlights that quantum models require fewer data points to achieve favorable metrics compared to classical models, showcasing their ability to discern non-trivial patterns that classical models struggle to identify, necessitating more data for effective learning. Additionally, We allocated 20 % of training data for validation.

The first model, referred to as the QNN model, is composed of encoding and ansatz circuits with 5 qubits. The number of qubits is determined by  $\log_2 N$  with  $N$  being the number of features, and it includes one linear layer that transforms  $2^N$  into one output. This model consists of a total of 63 parameters: 2 (layers)  $\times$  3 (rotations)  $\times$  5 (qubits) + 32 (weights) + 1 (bias).

Its classical counterpart is composed of an input layer with 30 neurons, two hidden layers with 20 and 5 neurons, respectively, and an output layer with 1 neuron. Consequently, the total number of parameters in this classical network is 731:  $30 \times 20$  (weights) + 20 (bias) +  $5 \times 20$  (weights) + 5 (bias) +  $5 \times 1$  (weights) + 1 (bias).

The second model, QLSTM, as described in section 8.0.3, consists of 776 parameters. It comprises a Linear layer that transforms 80 (input size and hidden size) into 5 (number of qubits):  $80 \times 5$  (weights) + 5 (bias). Subsequently, it includes the corresponding VQCs parameters:  $4 \times [1$  (layer)  $\times 5$  (qubits)  $\times 1$  (learning parameter used in Rx for ansatz circuit)]. Another Linear layer follows, transforming the number of qubits into the hidden size:

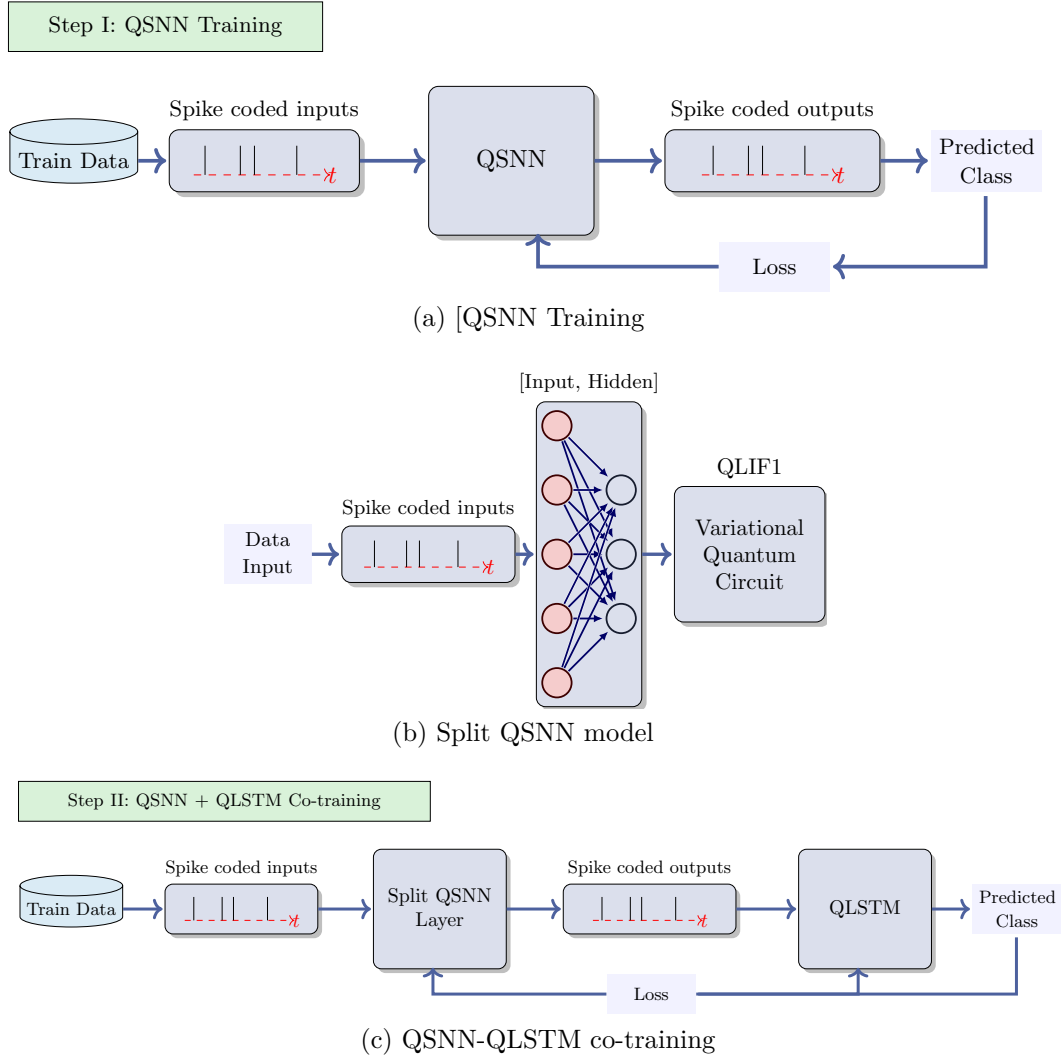


Figure 8.10: Training Process of QSNN-QLSTM. Step I involves the iterative training of QSNN. An original instance of QSNN (see Fig. 8.9) is trained independently without the influence of QLSTM, utilizing the gradient descent method. The model is then split (see Fig. 8.10b). In Step II, the split QSNN model is joined with a new instance of the QLSTM model for co-training. The loss from the output of QLSTM is used to simultaneously update both QSNN and QLSTM.

5 x 50 (weights) + 50 (bias). Finally, there is a Linear layer to transform hidden size into output dimension: 50 x 1 (weights) + 1 (bias).

Its classical counterpart is composed by a linear layer with 50 input neurons and 128 output neurons, two LSTM units, and final layer with 128 input neurons and 2 (number of classes) output neurons. Consequently, the total of parameters is 43.457: 4 x (30 + 50 + 1) x 50 + 4 x hidden (for the first LSTM layer) + 4 x (50 + 50 + 1) x 50 + 4 x hidden (for the second LSTM layer) + 128 x 50 x 1 (second hidden layer) + 128 x 1 + 1 (final output layer).

The third model, QSNN described in section 8.0.3 has 2 QLIF cells with 5 layers, 15 neurons for the hidden layer and 5 qubits and 647 parameters: 30 x 15 (weights) + 15 (bias) + 5 (layers) x 3 (learning parameters used in Rx, Ry, Rz for ansatz circuit) x 5 (qubits) x 2 (number of QLIF cells) + 15 x 2 (weights) + 2 (bias). Its classical counterpart is composed of a linear layer with 30 input neurons and 150 output neurons (hidden), two LIF neurons, and a final linear layer with 150 input neurons and two output neurons (number of classes). Consequently, the total of parameters is 4.952: 30 x 150 (weights) + 150 (bias) + 150 x 2 (weights) + 2 (bias).

The fourth model, QSNN-QLSTM, as described in Section 8.0.3, consists of 1 QLIF cell with 5 layers, 15 neurons for the hidden layer and 5 qubits. Additionally, it incorporates 1 QLSTM module with 2 layers and 25 neurons for the hidden layer. In total, the model comprises 956 parameters, calculates as follows: (30 x 15 + 15) + (5 (layers) x 3 (learning parameters used in Rx, Ry, Rz for ansatz circuit) x 5 (qubits) x 1 (number of QLIF cells)) + (40 (15 steps + 25 hidden neurons) x 5 + 5 for a Linear layer) + (2 QLSTM x 4 (VQCs) x 1 (rotation gate)) + (5 x 25 + 25 (for a Linear layer)) + (25 x 2 + 2 (for the output layer)).

## Results

The performance results for this use case are summarized in Table 8.4, which provides an overview of the average, best and worst F1 score by Quantum and Classical models.

In our results analysis, We observe the good performance of the novel model QSNN-QLSTM, which stands out as the top-performing quantum model with the highest F1 score average, alongside the LSTM model. In the worst F1-score category, once again, the QSNN-QLSTM model achieves the best result, with the LSTM model securing the second position. Finally, in

Table 8.2: Classical Models Configuration. Hyperparameters and configuration for Artificial Neural Network, Long Short-Term Memory and Spiking Neural Network respectively.

|                             | ANN                                                | LSTM                                                       | SNN                                    |
|-----------------------------|----------------------------------------------------|------------------------------------------------------------|----------------------------------------|
| <b>Optimizer</b>            | SGD lr=1e-2                                        | Adam lr=1e-3                                               | Adam lr=1e-3                           |
| <b>Scheduler</b>            | -                                                  | ReduceLROnPlateau<br>patience=30                           | -                                      |
| <b>Batch Size</b>           | 128                                                | 128                                                        | 128                                    |
| <b>Iterations</b>           | 700                                                | 350                                                        | 350                                    |
| <b>Pre-processing Steps</b> | StandardScaler                                     | StandardScaler                                             | StandardScaler                         |
| <b>Layers</b>               | [30, 20], Relu<br>[20, 5], Relu<br>[5, 2], Sigmoid | [Lstm, 30, 50, layers=2]<br>[50, 128]<br>[128, 1], Sigmoid | [30, 150]<br>Lif1 and Lif2<br>[150, 2] |

Table 8.3: Quantum Models Configuration. Hyperparameters and configuration for Quantum Neural Network, Quantum Long Short-Term Memory, Quantum Spiking Neural Network and the novel model composed by the union of last two networks.

|                              | QNN                                                 | QLSTM                                                         | QSNN                                      | QSNN-QLSTM                                     |
|------------------------------|-----------------------------------------------------|---------------------------------------------------------------|-------------------------------------------|------------------------------------------------|
| <b>Optimizer</b>             | RMSprop lr=1e-2                                     | Adam lr=1e-2                                                  | SGD lr=1e-4                               | SGD lr=1e-3                                    |
| <b>Scheduler</b>             | -                                                   | RLROnPlateau<br>patience= 30                                  | -                                         | RMSprop lr=1e-2<br>RLROnPlateau<br>patience=10 |
| <b>Batch Size</b>            | 256                                                 | 64                                                            | 16                                        | 64                                             |
| <b>Iterations</b>            | 140                                                 | 175                                                           | 700                                       | 80                                             |
| <b>Pre-processing</b>        | MinMaxScaler(0, $\pi$ )<br>Normalization<br>Sigmoid | MinMaxScaler(0, $\pi$ )<br>Normalization<br>Sigmoid           | StandardScaler<br>Normalization           | StandardScaler<br>Normalization<br>Sigmoid     |
| <b>Post-processing Steps</b> | -                                                   | -                                                             | 25                                        | 25                                             |
| <b>Layers</b>                | 2 QNN<br>Linear[2 <sup>N</sup> , 1]                 | Linear[50+30, 5]<br>1 QLSTM<br>Linear[5, 50]<br>Linear[50, 1] | Linear[30, 15]<br>5 QSNN<br>Linear[15, 2] | 5 QSNN<br>+<br>5 QLSTM                         |
| <b>Qubits</b>                | 5                                                   | 5                                                             | 5                                         | 5                                              |
| <b>Encoding Strategy</b>     | Amplitud<br>Encoding                                | Angle<br>Encoding                                             | Amplitud<br>Encoding                      | Amplitud<br>Encoding                           |

the best F1-score category, the QNN model exhibits the highest performance, followed closely by QSNN-QLSTM, LSTM and ANN, all with equal scores. Considering these tree categories, we can confirm that the QSNN-QLSTM model delivers superior overall performance, requiring fewer iterations as an additional advantage.

Continuing with the analysis and examining the boxplot in Fig 8.11, we can observe that the QNN, ANN and QSNN-QLSTM models exhibit better

robustness results than the rest.

And finally, considering the learning curves shown in Fig 8.12, we can observe a good evolution of learning with very similar curves for all the model. Notably, there is a decrease in variance in the later iterations, indicating increased stability and convergence in learning. This suggests that the chosen architectures successfully capture the dataset’s patterns.

Table 8.4: Results obtained by Classical and Quantum models. Column 1: Classical models followed by Quantum versions and the novel quantum model proposed; Column 2: Average F1 Score; Column 3: Best F1 score; Column 4: Worst F1 score.

|                   | Avg F1 score | Best F1 score | Worst F1 score |
|-------------------|--------------|---------------|----------------|
| <b>Ann</b>        | 0.95         | 0.99          | 0.91           |
| <b>Lstm</b>       | 0.97         | 0.99          | 0.93           |
| <b>Snn</b>        | 0.94         | 0.96          | 0.90           |
| <b>Qnn</b>        | 0.94         | 1.0           | 0.40           |
| <b>QLstm</b>      | 0.84         | 0.94          | 0.65           |
| <b>Qsnn</b>       | 0.89         | 0.98          | 0.74           |
| <b>Qsnn-QLstm</b> | 0.97         | 0.99          | 0.93           |

### 8.0.5 Discussion

We introduced 3 classical architectures and 4 quantum architectures, one of them incorporating a novel approach inspired by brain function to address the use case of fraud detection. A general perception derived from the analysis of the results is the effectiveness of utilizing quantum models inspired by the brain, achieving superior performance compared to classical approaches, while also exhibiting reduced complexity (requiring fewer parameters during training). On the other hand, although this model requires fewer iterations, as they have been executed in simulators, the runtime is impacted due to the computational load it carries. Accessing and conducting large-scale experiments on current quantum devices pose challenges, and the inherent noise in these systems can affect the performance and reliability of the models. These limitations underscore the need for ongoing advancements in both quantum hardware and simulators, enabling a reduction in computation times.

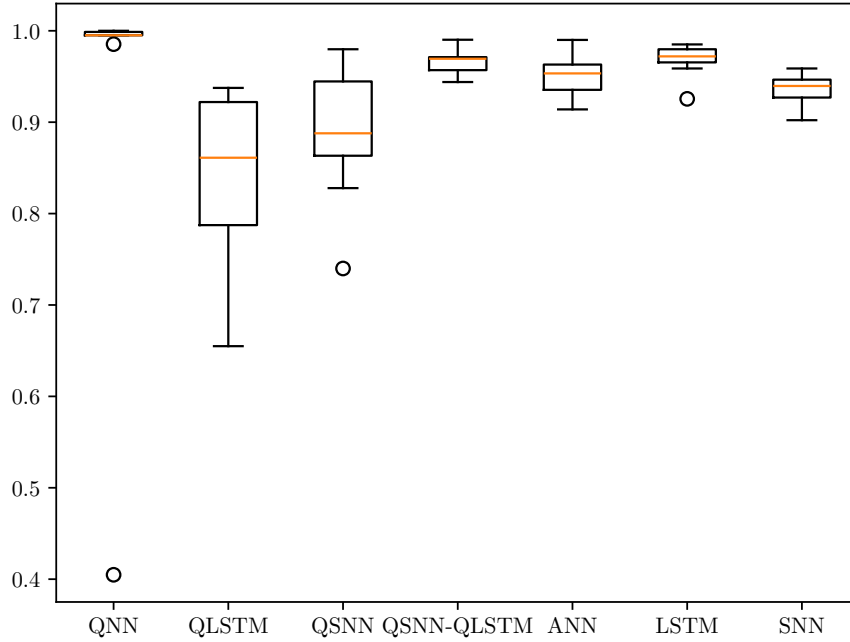


Figure 8.11: Boxplots illustrating the distribution of F1 score obtained from quantum (QNN, QLSTM, QSNN and QSNN-QLSTM) and from classical models (ANN, LSTM and SNN).

### 8.0.6 Conclusions and Future Work

While Hebbian theory has found widespread acceptance in neuroscience, it falls short of fully capturing the impact of feedback and various stimuli encountered in daily life on the learning process. This aspect has recently gained considerable attention, prompting an exploration of its incorporation into Reinforcement Learning agents using the models implemented in this study. Evaluating whether such integration yields improved results remains an intriguing avenue for future research.

Recent advancements in neuroscience-inspired learning have extended Hebbian plasticity by incorporating rewards through the exploration of the neuromodulator dopamine. In the context of Reinforcement Learning (RL), research on neo-Hebbian learning has led to several comprehensive review

Figure 8.12: Learning curves obtained from seven models. Three classical models: ANN, SNN and LSTM. Four quantum models: QNN, QSNN, QLSTM and new brain-inspired model QSNN-QLSTM.

articles, addressing diverse topics such as the consolidation of associative memories and the relationships between specific neo-Hebbian formulations and their counterparts in computational RL theory [11].

Taking this exploration further, it would be of great interest to investigate the integration of these theories into quantum agents. Unraveling the potential benefits of infusing neuroscientific insights into Quantum Reinforcement Learning could pave the way for exciting advancements in the field.

Shifting the focus to Online Learning (OL), where a single sample is provided to the learning algorithm at every time instant, there exists a critical need for novel algorithms that can adapt swiftly to changes while upholding high performance. OL, particularly in the presence of concept drift, has emerged as a topic of substantial interest in recent years, continuing to spark active debate within the research community due to its numerous open challenges [118, 119, 120]. Although this manuscript does not delve into the demonstration of OL learning, showcasing the adaptability of the model to real-time situations could be a valuable pursuit, leveraging the potential of Quantum Spiking Neural Network for real-time gradients calculation [73, 75], complemented by the use of QLSTM to mitigate catastrophic forgetting.

Furthermore, an intriguing avenue for further research lies in integrating this model into the development of agents in Reinforcement Learning (RL). Given the dynamic and evolving nature of information in RL scenarios, such integration could yield promising results and significantly contribute to the advancement of intelligent agent systems.

Abbreviations utilized in this thesis comprise:

|        |                                                  |
|--------|--------------------------------------------------|
| A2C    | Advantage Actor–Critic                           |
| ANN    | Artificial Neural Network                        |
| BP     | Barren Plateau                                   |
| BPTT   | BackPropagation Through Time                     |
| BG     | Basal Ganglia                                    |
| COBYLA | Constrained Optimization BY Linear Approximation |
| CPU    | Central Processing Unit                          |
| DNN    | Deep Neural Network                              |
| DQN    | Deep Q-Network                                   |
| DRL    | Deep Reinforcement Learning                      |
| EV     | Electrical Vehicle                               |
| GRU    | Gate Recurrent Unit                              |
| HDC    | HyperDimensional Computing                       |
| HPC    | High-Performance Computing                       |
| 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             |
| OL     | Online Learning                                  |
| PFC    | PreFrontal Cortex                                |
| PQC    | Parametrized/Parameterized Quantum Circuit       |
| 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                   |

|       |                                                    |
|-------|----------------------------------------------------|
| RL    | Reinforcement Learning                             |
| RNN   | Recurrent Neural Networks                          |
| RSA   | Rivest, Shamir and Adleman                         |
| SGD   | Stochastic Gradient Descent                        |
| SNN   | Spiking Neural Network                             |
| SPSA  | Simultaneous Perturbation Stochastic Approximation |
| S-R   | Stimulus-Response                                  |
| STS   | Short-Term Store                                   |
| vmPFC | ventromedial PreFrontal Cortex                     |
| VQA   | Variational Quantum Algorithm                      |
| VQC   | Variational Quantum Circuit                        |

# Bibliography

- [1] MohammadReza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takac. Reinforcement learning for solving the vehicle routing problem. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [2] R.C. Atkinson and R.M. Shiffrin. Human memory: A proposed system and its control processes. In Kenneth W. Spence and Janet Taylor Spence, editors, *Human Memory: A Proposed System and its Control Processes.*, volume 2 of *Psychology of Learning and Motivation*, pages 89–195. Academic Press, 1968.
- [3] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharmashan Kumar, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.
- [4] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark, 2020.
- [5] Sergey Levine, Chelsea Finn, Trevor Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *ArXiv*, abs/1504.00702, 2016.
- [6] Ahmad El Sallab, Mohammed Abdou, Etienne Perot, and Senthil Kumar Yogamani. Deep reinforcement learning framework for autonomous driving. *ArXiv*, abs/1704.02532, 2017.

- [7] Konstantinos I. Roumeliotis and Nikolaos D. Tselikas. Chatgpt and open-ai models: A preliminary review. *Future Internet*, 15(6), 2023.
- [8] Heriberto Cuayáhuatl, Steve Renals, Oliver Lemon, and Hiroshi Shimodaira. Evaluation of a hierarchical reinforcement learning spoken dialogue system. *Computer Speech & Language*, 24(2):395–429, 2010.
- [9] Lin Zhao, Lu Zhang, Zihao Wu, Yuzhong Chen, Haixing Dai, Xiaowei Yu, Zhengliang Liu, Tuo Zhang, Xintao Hu, Xi Jiang, Xiang Li, Dajiang Zhu, Dinggang Shen, and Tianming Liu. When brain-inspired ai meets agi. *Meta-Radiology*, 1(1):100005, 2023.
- [10] Chaoqiong Fan, Li Yao, Jiakai Zhang, Zonglei Zhen, and Xia Wu. Advanced reinforcement learning and its connections with brain neuroscience. *Research*, 6:0064, 2023.
- [11] Anthony Triche, Anthony S. Maida, and Ashok Kumar. Exploration in neo-hebbian reinforcement learning: Computational approaches to the exploration–exploitation balance with bio-inspired neural networks. *Neural Networks*, 151:16–33, 2022.
- [12] Lov K. Grover. Quantum mechanics helps in searching for a needle in a haystack. *Phys. Rev. Lett.*, 79:325–328, Jul 1997.
- [13] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303–332, 1999.
- [14] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6):467–488, Jun 1982.
- [15] Daoyi Dong, Chunlin Chen, Hanxiong Li, and Tzyh-Jong Tarn. Quantum reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(5):1207–1220, 2008.
- [16] Yunseok Kwak, Won Joon Yun, Soyi Jung, and Joongheon Kim. Quantum neural networks: Concepts, applications, and challenges. *2021 Twelfth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 413–416, 2021.

- [17] Dipesh Niraula, Jamalina Jamaluddin, Martha M. Matuszak, Randall K. Ten Haken, and Issam El Naqa. Quantum deep reinforcement learning for clinical decision support in oncology: application to adaptive radiotherapy. *Scientific Reports*, 11(1):23545, Dec 2021.
- [18] El Amine Cherrat, Snehal Raj, Iordanis Kerenidis, Abhishek Shekhar, Ben Wood, Jon Dee, Shouvanik Chakrabarti, Richard Chen, Dylan Herman, Shaohan Hu, Pierre Minssen, Ruslan Shaydulin, Yue Sun, Romina Yalovetzky, and Marco Pistoia. Quantum deep hedging, 2023.
- [19] Antonio Macaluso, Luca Clissa, Stefano Lodi, and Claudio Sartori. A variational algorithm for quantum neural networks. In *Computational Science – ICCS 2020*, pages 591–604. Springer International Publishing, 2020.
- [20] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001, nov 2019.
- [21] Chen Zhao and Xiao-Shan Gao. Qdnn: deep neural networks with quantum layers. *Quantum Machine Intelligence*, 3:15, 2021.
- [22] Maria Schuld, Alex Bocharov, Krysta M. Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *Phys. Rev. A*, 101:032308, 2020.
- [23] Nathan Wiebe, Ashish Kapoor, and Krysta M. Svore. Quantum nearest-neighbor algorithms for machine learning. *Quantum Information and Computation*, 15:318–358, 2015.
- [24] D. Anguita, Sandro Ridella, Fabio Riveccio, and Rodolfo Zunino. Quantum optimization for training support vector machines. *Neural networks*, 16:763–770, 2003.
- [25] Esma Aïmeur, Gilles Brassard, and Sébastien Gambs. Quantum speed-up for unsupervised learning. *Machine Learning*, 90:261–287, 2013.
- [26] Robert Sutor. *Dancing with Qubits*. Packt, 11 2019.
- [27] Sieglinde M. L. Pfaendler, Konstantin Konson, and Franziska Greinert. Advancements in quantum computing—viewpoint: Building adoption and competency in industry. *Datenbank-Spektrum*, 24(1):5–20, Mar 2024.

- [28] Peter Wittek. *Quantum Machine Learning: What Quantum Computing means to data mining*. Elsevier, 2014.
- [29] Manuela Weigold, Johanna Barzen, Frank Leymann, and Marie Salm. Expanding data encoding patterns for quantum algorithms. In *2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)*, pages 95–101, 2021.
- [30] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, 103(3), mar 2021.
- [31] Amira Abbas, David Sutter, Christa Zoufal, Aurelien Lucchi, Alessio Figalli, and Stefan Woerner. The power of quantum neural networks. *Nature Computational Science*, 1(6):403–409, Jun 2021.
- [32] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. Quantum generative adversarial networks for learning and loading random distributions. *npj Quantum Information*, 5(1):103, Nov 2019.
- [33] Javier Alcazar, Mohammad Ghazi Vakili, Can B. Kalayci, and Alejandro Perdomo-Ortiz. Enhancing combinatorial optimization with classical and quantum generative models. *Nature Communications*, 15(1):2761, Mar 2024.
- [34] Yuxuan Du, Min-Hsiu Hsieh, Tongliang Liu, and Dacheng Tao. Expressive power of parametrized quantum circuits. *Phys. Rev. Res.*, 2:033125, Jul 2020.
- [35] Mehdi Sadi, Yi He, Yanjing Li, Mahabubul Alam, Satwik Kundu, Swaroop Ghosh, Javad Bahrami, and Naghme Karimi. Special session: On the reliability of conventional and quantum neural network hardware. In *2022 IEEE 40th VLSI Test Symposium (VTS)*, pages 1–12, 2022.
- [36] Hao Dong, Zihan Ding, Shanghang Zhang, Hang Yuan, Hongming Zhang, Jingqing Zhang, Yanhua Huang, Tianyang Yu, Huaqing Zhang, and Ruitong Huang. *Deep Reinforcement Learning: Fundamentals, Research, and Applications*. Springer Nature, 2020. <http://www.deeppreinforcementlearningbook.org>.

- [37] Richard S. Sutton and Andrew G. Barto. *The Reinforcement Learning Problem*, pages 51–85. MIT Press, 1998.
- [38] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dhharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 02 2015.
- [39] Kun Shao, Dongbin Zhao, Yuanheng Zhu, and Qichao Zhang. Visual navigation with actor-critic deep reinforcement learning. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6, 2018.
- [40] Rui Nian, Jinfeng Liu, and Biao Huang. A review on reinforcement learning: Introduction and applications in industrial process control. *Computers & Chemical Engineering*, 139:106886, 2020.
- [41] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4909–4926, 2022.
- [42] Bharat Singh, Rajesh Kumar, and Vinay Pratap Singh. Reinforcement learning in robotic applications: a comprehensive survey. *Artificial Intelligence Review*, 55(2):945–990, Feb 2022.
- [43] Mohammad Noaeen, Atharva Naik, Liana Goodman, Jared Crebo, Taimoor Abrar, Zahra Shakeri Hossein Abad, Ana L.C. Bazzan, and Behrouz Far. Reinforcement learning in urban network traffic signal control: A systematic literature review. *Expert Systems with Applications*, 199:116830, 2022.
- [44] M. Mehdi Afsar, Trafford Crump, and Behrouz Far. Reinforcement learning based recommender systems: A survey. *ACM Comput. Surv.*, 55(7), dec 2022.
- [45] Ngan Le, Vidhiwar Singh Rathour, Kashu Yamazaki, Khoa Luu, and Marios Savvides. Deep reinforcement learning in computer vision: a

comprehensive survey. *Artificial Intelligence Review*, 55(4):2733–2819, Apr 2022.

- [46] S. Kevin Zhou, Hoang Ngan Le, Khoa Luu, Hien V Nguyen, and Nicholas Ayache. Deep reinforcement learning in medical imaging: A literature review. *Medical Image Analysis*, 73:102193, 2021.
- [47] Zhuofan Liao, Jingsheng Peng, Yuantao Chen, Jingyu Zhang, and Jin Wang. A fast q-learning based data storage optimization for low latency in data center networks. *IEEE Access*, 8:90630–90639, 2020.
- [48] Xavier Dutreilh, Sergey Kirgizov, Olga Melekhova, Jacques Malenfant, Nicolas Rivierre, and Isis Truck. Using reinforcement learning for autonomic resource allocation in clouds: towards a fully automated workflow. In *7th International Conference on Autonomic and Autonomous Systems (ICAS’2011)*, pages 67–74, 2011.
- [49] Shashank Swarup, Elhadi M. Shakshuki, and Ansar Yasar. Task scheduling in cloud using deep reinforcement learning. *Procedia Computer Science*, 184:42–51, 2021.
- [50] Jianxiong Wan, Jie Zhou, and Xiang Gui. Intelligent rack-level cooling management in data centers with active ventilation tiles: A deep reinforcement learning approach. *IEEE Intelligent Systems*, 36(6):42–52, 2021.
- [51] Qiong Xiao Fu, Enchang Sun, Kang Meng, Meng Li, and Yanhua Zhang. Deep q-learning for routing schemes in sdn-based data center networks. *IEEE Access*, 8:103491–103499, 2020.
- [52] Ali Habibi Khalaj and Saman K. Halgamuge. A review on efficient thermal management of air- and liquid-cooled data centers: From chip to the cooling system. *Applied Energy*, 205:1165–1188, 2017.
- [53] Naren Srivaths Raman, Adithya M. Devraj, Prabir Barooah, and Sean P. Meyn. Reinforcement learning for control of building hvac systems. In *2020 American Control Conference (ACC)*, pages 2326–2332, 2020.
- [54] Yuan Wang, Kirubakaran Velswamy, and Biao Huang. A long-short term memory recurrent neural network based reinforcement learning

controller for office heating ventilation and air conditioning systems. *Processes*, 5(3), 2017.

- [55] Qiming Fu, Zhicong Han, Jianping Chen, You Lu, Hongjie Wu, and Yunzhe Wang. Applications of reinforcement learning for building energy efficiency control: A review. *Journal of Building Engineering*, 50:104165, 2022.
- [56] Andrea Skolik, Sofiene Jerbi, and Vedran Dunjko. Quantum agents in the Gym: a variational quantum algorithm for deep Q-learning. *Quantum*, 6:720, 2022.
- [57] Sofiene Jerbi, Casper Gyurik, Simon C. Marshall, Hans J. Briegel, and Vedran Dunjko. Parametrized quantum policies for reinforcement learning, 2021.
- [58] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.
- [59] Samuel Yen-Chi Chen, Chao-Han Huck Yang, Jun Qi, Pin-Yu Chen, Xiaoli Ma, and Hsi-Sheng Goan. Variational quantum circuits for deep reinforcement learning. *IEEE Access*, 8:141007–141024, 2020.
- [60] Y. Liu, C. Xu, and S. Jin. Reinforcement learning for continuous control: A quantum normalized advantage function approach. In *2023 IEEE International Conference on Quantum Software (QSW)*, pages 83–87, Los Alamitos, CA, USA, jul 2023. IEEE Computer Society.
- [61] Alberto Acuto, Paola Barillà, Ludovico Bozzolo, Matteo Conterno, Mattia Pavese, and Antonio Policicchio. Variational quantum soft actor-critic for robotic arm control, 2022.
- [62] Dirk Heimann, Hans Hohenfeld, Felix Wiebe, and Frank Kirchner. Quantum deep reinforcement learning for robot navigation tasks, 2022.
- [63] Akash Sinha, Antonio Macaluso, and Matthias Klusch. Nav-q: Quantum deep reinforcement learning for collision-free navigation of self-driving cars, 2023.

- [64] Gyu Seon Kim, JaeHyun Chung, and Soohyun Park. Realizing stabilized landing for computation-limited reusable rockets: A quantum reinforcement learning approach, 2023.
- [65] Randall Correll, Sean J. Weinberg, Fabio Sanches, Takanori Ide, and Takafumi Suzuki. Quantum neural networks for a supply chain logistics application, 2022.
- [66] Fabio Sanches, Sean Weinberg, Takanori Ide, and Kazumitsu Kamiya. Short quantum circuits in reinforcement learning policies for the vehicle routing problem. *Physical Review A*, 105(6), June 2022.
- [67] Manoj Kumar, Upasana Dohare, Sushil Kumar, and Neeraj Kumar. Blockchain based optimized energy trading for e-mobility using quantum reinforcement learning. *IEEE Transactions on Vehicular Technology*, 72(4):5167–5180, 2023.
- [68] Won Joon Yun, Yunseok Kwak, Jae Pyoung Kim, Hyunhee Cho, Soyi Jung, Jihong Park, and Joongheon Kim. Quantum multi-agent reinforcement learning via variational quantum circuit design. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, pages 1332–1335, 2022.
- [69] Won Joon Yun, Jae Pyoung Kim, Soyi Jung, Jae-Hyun Kim, and Joongheon Kim. Quantum multi-agent actor-critic neural networks for internet-connected multi-robot coordination in smart factory management, 2023.
- [70] James Adu Ansere, Eric Gyamfi, Vishal Sharma, Hyundong Shin, Octavia A. Dobre, and Trung Q. Duong. Quantum deep reinforcement learning for dynamic resource allocation in mobile edge computing-based iot systems. *IEEE Transactions on Wireless Communications*, pages 1–1, 2023.
- [71] Junzheng Yang. *Apply Deep Reinforcement Learning with Quantum Computing on the Pricing of American Options*, pages 675–694. Internet Finance and Digital Economy, 2023.
- [72] D.O. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. Taylor & Francis, 2005.

- [73] Jason K. Eshraghian, Max Ward, Emre Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. Training spiking neural networks using lessons from deep learning, 2021.
- [74] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural Networks*, 111:47–63, 2019.
- [75] Jesus L. Lobo, Javier Del Ser, Albert Bifet, and Nikola Kasabov. Spiking neural networks and online learning: An overview and perspectives. *Neural Networks*, 121:88–100, 2020.
- [76] LM Lapicque. Recherches quantitatives sur l’excitation électrique des nerfs. *J. Physiol. Paris.*, 9:620–635, 1907.
- [77] Zhuowen Zou, Haleh Alimohamadi, Ali Zakeri, Farhad Imani, Yeseong Kim, M. Hassan Najafi, and Mohsen Imani. Memory-inspired spiking hyperdimensional network for robust online learning. *Scientific Reports*, 12(1):7641, 05 2022.
- [78] Kaushalya Kumarasinghe, Nikola Kasabov, and Denise Taylor. Brain-inspired spiking neural networks for decoding and understanding muscle activity and kinematics from electroencephalography signals during hand movements. *Scientific Reports*, 11(1):2486, 06 2021.
- [79] Andrea Banino, Caswell Barry, Benigno Uria, Charles Blundell, Timothy Lillicrap, Piotr Mirowski, Alexander Pritzel, Martin J. Chadwick, Thomas Degris, Joseph Modayil, Greg Wayne, Hubert Soyer, Fabio Viola, Brian Zhang, Ross Goroshin, Neil Rabinowitz, Razvan Pascanu, Charlie Beattie, Stig Petersen, Amir Sadik, Stephen Gaffney, Helen King, Koray Kavukcuoglu, Demis Hassabis, Raia Hadsell, and Dharshan Kumaran. Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557(7705):429–433, 05 2018.
- [80] Yanhu Chen, Cen Wang, Hongxiang Guo, Xiong Gao, and Jian Wu. Accelerating spiking neural networks using quantum algorithm with high success probability and high calculation accuracy. *Neurocomputing*, 493:435–444, 2022.

- [81] Zuyu Xu, Kang Shen, Pengnian Cai, Tao Yang, Yuanming Hu, Shixian Chen, Yunlai Zhu, Zuheng Wu, Yuehua Dai, Jun Wang, and Fei Yang. Parallel proportional fusion of spiking quantum neural network for optimizing image classification, 2024.
- [82] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5:157–166, 1994.
- [83] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.
- [84] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:855–868, 2009.
- [85] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks : the official journal of the International Neural Network Society*, 18:602–610, 2005.
- [86] Sepp Hochreiter and Jürgen Schmidhuber. Lstm can solve hard long time lag problems. *Advances in Neural Information Processing Systems*, 9:473–479, 1996.
- [87] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks, 2018.
- [88] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In Dekai Wu, Marine Carpuat, Xavier Carreras, and Eva Maria Vecchi, editors, *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October 2014. Association for Computational Linguistics.
- [89] Guo-Bing Zhou, Jianxin Wu, Chen-Lin Zhang, and Zhi-Hua Zhou. Minimal gated unit for recurrent neural networks. *International Journal of Automation and Computing*, 13(3):226–234, Jun 2016.

- [90] Rahul Dey and Fathi M. Salem. Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1597–1600, 2017.
- [91] Joel C. Heck and Fathi M. Salem. Simplified minimal gated unit variations for recurrent neural networks. In *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1593–1596, 2017.
- [92] Samuel Yen-Chi Chen, Shinjae Yoo, and Yao-Lung L. Fang. Quantum long short-term memory. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8622–8626, 2022.
- [93] Michał Siemaszko, Adam Buraczewski, Bertrand Le Saux, and Magdalena Stobińska. Rapid training of quantum recurrent neural networks. *Quantum Machine Intelligence*, 5(2):31, Jul 2023.
- [94] Yunseok Kwak, Won Joon Yun, Soyi Jung, Jong-Kook Kim, and Joongheon Kim. Introduction to quantum reinforcement learning: Theory and pennylane-based implementation. In *2021 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 416–420, 2021.
- [95] M Cerezo and Patrick J Coles. Higher order derivatives of quantum neural networks with barren plateaus. *Quantum Science and Technology*, 6(3):035006, June 2021.
- [96] Andrew Arrasmith, M. Cerezo, Piotr Czarnik, Lukasz Cincio, and Patrick J. Coles. Effect of barren plateaus on gradient-free optimization. *Quantum*, 5:558, October 2021.
- [97] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9(1):4812, Nov 2018.
- [98] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2010.

- [99] John F. Kolen and Stefan C. Kremer. *Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies*, pages 237–243. IEEE Explore, 2001.
- [100] Shai Shalev-Shwartz, Ohad Shamir, and Shaked Shammah. Failures of gradient-based deep learning, 2017.
- [101] Samson Wang, Enrico Fontana, M. Cerezo, Kunal Sharma, Akira Sone, Lukasz Cincio, and Patrick J. Coles. Noise-induced barren plateaus in variational quantum algorithms. *Nature Communications*, 12(1):6961, Nov 2021.
- [102] A. Kulshrestha and I. Safro. Beinit: Avoiding barren plateaus in variational quantum algorithms. In *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 197–203, Los Alamitos, CA, USA, sep 2022. IEEE Computer Society.
- [103] Zoë Holmes, Kunal Sharma, M. Cerezo, and Patrick J. Coles. Connecting ansatz expressibility to gradient magnitudes and barren plateaus. *PRX Quantum*, 3(1), January 2022.
- [104] Carlos Ortiz Marrero, Mária Kieferová, and Nathan Wiebe. Entanglement induced barren plateaus, 2021.
- [105] Stefan H. Sack, Raimel A. Medina, Alexios A. Michailidis, Richard Kueng, and Maksym Serbyn. Avoiding barren plateaus using classical shadows. *PRX Quantum*, 3(2), June 2022.
- [106] Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster, and José I. Latorre. Data re-uploading for a universal quantum classifier. *Quantum*, 4:226, February 2020.
- [107] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, 103(3), mar 2021.
- [108] Philippe Domenech, Sylvain Rheims, and Etienne Koechlin. Neural mechanisms resolving exploitation-exploration dilemmas in the medial prefrontal cortex. *Science*, 369(6507):eabb0184, 2020.

- [109] Alon Boaz Baram, Timothy Howard Muller, Hamed Nili, Mona Maria Garvert, and Timothy Edward John Behrens. Entorhinal and ventromedial prefrontal cortices abstract and generalize the structure of reinforcement learning problems. *Neuron*, 109(4):713–723.e7, 2021.
- [110] Rafal Bogacz and Tobias Larsen. Integration of reinforcement learning and optimal decision-making theories of the basal ganglia. *Neural computation*, 23:817–51, 04 2011.
- [111] James Houk, J. Adams, and A. Barto. A model of how the basal ganglia generate and use neural signals that predict reinforcement. *Models of Information Processing in the Basal Ganglia*, Vol. 13, 07 1995.
- [112] Daphna Joel, Yael Niv, and Eytan Ruppin. Actor–critic models of the basal ganglia: new anatomical and computational perspectives. *Neural Netw.*, 15(4-6):535–547, 2002.
- [113] Anne G E Collins and Michael J Frank. Opponent actor learning (OpAL): Modeling interactive effects of striatal dopamine on reinforcement learning and choice incentive. *Psychol. Rev.*, 121(3):337–366, 2014.
- [114] Tiago V Maia and Michael J Frank. From reinforcement learning models to psychiatric and neurological disorders. *Nat. Neurosci.*, 14(2):154–162, 02 2011.
- [115] Tiago V Maia. Reinforcement learning, conditioning, and the brain: Successes and challenges. *Cogn. Affect. Behav. Neurosci.*, 9(4):343–364, 12 2009.
- [116] John O’Doherty, Peter Dayan, Johannes Schultz, Ralf Deichmann, Karl Friston, and Raymond J. Dolan. Dissociable roles of ventral and dorsal striatum in instrumental conditioning. *Science*, 304(5669):452–454, 2004.
- [117] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In Gordon H. Bower, editor, *Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem*, volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989.

- [118] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.
- [119] Bartosz Krawczyk, Leandro L Minku, João Gama, Jerzy Stefanowski, and Michał Woźniak. Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132–156, 2017.
- [120] Viktor Losing, Barbara Hammer, and Heiko Wersing. Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing*, 275:1261–1274, 2018.
- [121] Felix Hill, Andrew Lampinen, Rosalia Schneider, Stephen Clark, Matthew Botvinick, James L. McClelland, and Adam Santoro. Environmental drivers of systematicity and generalization in a situated agent, 2020.
- [122] Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks, 2018.
- [123] Matthew Botvinick, Jane X. Wang, Will Dabney, Kevin J. Miller, and Zeb Kurth-Nelson. Deep reinforcement learning and its neuroscientific implications. *Neuron*, 107(4):603–616, 2020.
- [124] Zhe Wang and Tianzhen Hong. Reinforcement learning for building controls: The opportunities and challenges. *Applied Energy*, 269:115036, 2020.
- [125] Karl Mason and Santiago Grijalva. A review of reinforcement learning for autonomous building energy management. *Computers and Electrical Engineering*, 78:300–312, 2019.
- [126] Tianshu Wei, Yanzhi Wang, and Qi Zhu. Deep reinforcement learning for building hvac control. In *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, 2017.
- [127] Paulo Lissa, Conor Deane, Michael Schukat, Federico Seri, Marcus Keane, and Enda Barrett. Deep reinforcement learning for home energy management system control. *Energy and AI*, 3:100043, 2021.

- [128] Drury Crawley, Curtis Pedersen, Linda Lawrie, and Frederick Winkelmann. Energyplus: Energy simulation program. *Ashrae Journal*, 42:49–56, 04 2000.
- [129] Sven Erik Mattsson and Hilding Elmqvist. Modelica - an international effort to design the next generation modeling language. *IFAC Proceedings Volumes*, 30(4):151–155, 1997. 7th IFAC Symposium on Computer Aided Control Systems Design (CACSD '97), Gent, Belgium, 28-30 April.
- [130] Javier Jiménez-Raboso, Alejandro Campoy-Nieves, Antonio Manjavacas-Lucas, Juan Gómez-Romero, and Miguel Molina-Solana. Sinergym: A building simulation and control framework for training reinforcement learning agents. In *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, page 319–323, New York, NY, USA, 2021. Association for Computing Machinery.
- [131] Zhiang Zhang and Khee Poh Lam. Practical implementation and evaluation of deep reinforcement learning control for a radiant heating system. In *Proceedings of the 5th Conference on Systems for Built Environments*, BuildSys '18, page 148–157, New York, NY, USA, 2018. Association for Computing Machinery.
- [132] Paul Scharnhorst, Baptiste Schubnel, Carlos Fernández Bandera, Jaume Salom, Paolo Taddeo, Max Boegli, Tomasz Gorecki, Yves Stauffer, Antonis Peppas, and Chrysa Politi. Energym: A building model library for controller benchmarking. *Applied Sciences*, 11(8), 2021.
- [133] Renzong Lian, Jiankun Peng, Wu Yuankai, Huachun Tan, and Hailong Zhang. Rule-interposing deep reinforcement learning based energy management strategy for power-split hybrid electric vehicle. *Energy*, 197:117297, 03 2020.
- [134] Jiankun Peng, Yi Fan, Guodong Yin, and Ruhai Jiang. Collaborative optimization of energy management strategy and adaptive cruise control based on deep reinforcement learning. *IEEE Transactions on Transportation Electrification*, page 1–1, 2022. Cited by: 0.

- [135] Shuoyao Wang, Suzhi Bi, and Yingjun Angela Zhang. Reinforcement learning for real-time pricing and scheduling control in ev charging stations. *IEEE Transactions on Industrial Informatics*, 17(2):849–859, 2021.
- [136] Carlo Ciliberto, Mark Herbster, Alessandro Ialongo, Massimiliano Pontil, Andrea Rocchetto, Simone Severini, and Leonard Wossnig. Quantum machine learning: A classical perspective. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 474, 07 2017.
- [137] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [138] Robert Sutor. *Dancing with Qubits*. Packt, 11 2019.
- [139] Eva Andrés, Manuel Pegalajar Cuéllar, and Gabriel Navarro. On the use of quantum reinforcement learning in energy-efficiency scenarios. *Energies*, 15, 2022.
- [140] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
- [141] U. von Luxburg. *31st Annual Conference on Neural Information Processing Systems 2017 ; Long Beach, California, 4-9 December 2017: Editors: Ulrike Von Luxburg, Isabelle Guyon, Samy Bengio, Hanna Wallach, Rob Fergus, Roman Garnett, S.V.N. Vishwanathan*. Advances in neural information processing systems. Curran Associates, Incorporated, 2017.
- [142] Owen Lockwood and Mei Si. Reinforcement learning with quantum variational circuit. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 16(1):245–251, Oct. 2020.

- [143] A. Sannia, A. Giordano, N. Lo Gullo, C. Mastroianni, and F. Plastina. A hybrid classical-quantum approach to speed-up q-learning. *Scientific Reports*, 13(1):3913, Mar 2023.
- [144] Maniraman Periyasamy, Marc Hölle, Marco Wiedmann, Daniel D. Scherer, Axel Plinge, and Christopher Mutschler. Batch quantum reinforcement learning, 2023.
- [145] Samuel Yen-Chi Chen, Chih-Min Huang, Chia-Wei Hsing, Hsi-Sheng Goan, and Ying-Jer Kao. Variational quantum reinforcement learning via evolutionary optimization. *Machine Learning: Science and Technology*, 3(1):015025, feb 2022.
- [146] Anil K. Jain, Jianchang Mao, and K. M. Mohiuddin. Artificial neural networks: A tutorial. *Computer*, 29(3):31–44, mar 1996.
- [147] Robert E. Uhrig. Introduction to artificial neural networks. *Proceedings of IECON '95 - 21st Annual Conference on IEEE Industrial Electronics*, 1:33–37 vol.1, 1995.
- [148] Eric Chalmers, Edgar Bermudez Contreras, Brandon Robertson, Artur Luczak, and Aaron Gruber. Context-switching and adaptation: Brain-inspired mechanisms for handling environmental changes. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 3522–3529, 2016.
- [149] Federica Robertazzi, Matteo Vissani, Guido Schillaci, and Egidio Falotico. Brain-inspired meta-reinforcement learning cognitive control in conflictual inhibition decision-making task for artificial agents. *Neural Networks*, 154:283–302, 2022.
- [150] Zhuoya Zhao, Feifei Zhao, Yuxuan Zhao, Yi Zeng, and Yinqian Sun. A brain-inspired theory of mind spiking neural network improves multi-agent cooperation and competition. *Patterns*, 4(8):100775, 2023.
- [151] Ke Zhang, Xi Lin, and Meng Li. Graph attention reinforcement learning with flexible matching policies for multi-depot vehicle routing problems. *Physica A: Statistical Mechanics and its Applications*, 611:128451, 2023.

- [152] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.
- [153] Saed Rezayi, Haixing Dai, Zhengliang Liu, Zihao Wu, Akarsh Hebbar, Andrew H. Burns, Lin Zhao, Dajiang Zhu, Quanzheng Li, Wei Liu, Sheng Li, Tianming Liu, and Xiang Li. Clinicalradiobert: Knowledge-infused few shot learning for clinical notes named entity recognition. In Chunfeng Lian, Xiaohuan Cao, Islem Rekik, Xuanang Xu, and Zhiming Cui, editors, *Machine Learning in Medical Imaging*, pages 269–278, Cham, 2022. Springer Nature Switzerland.
- [154] Zhengliang Liu, Xinyu He, Lei Liu, Tianming Liu, and Xiaoming Zhai. Context matters: A strategy to pre-train language model for science education. In Ning Wang, Genaro Rebolledo-Mendez, Vania Dimitrova, Noboru Matsuda, and Olga C. Santos, editors, *Artificial Intelligence in Education. Posters and Late Breaking Results, Workshops and Tutorials, Industry and Innovation Tracks, Practitioners, Doctoral Consortium and Blue Sky*, pages 666–674, Cham, 2023. Springer Nature Switzerland.
- [155] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.
- [156] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.
- [157] Eva Andrés, M. P. Cuéllar, and G. Navarro. Efficient dimensionality reduction strategies for quantum reinforcement learning. *IEEE Access*, 11:104534–104553, 2023.

- [158] Jerome R. Busemeyer and Peter D. Bruza. *Quantum Models of Cognition and Decision*. Cambridge University Press, 2012.
- [159] Ji-An Li, Daoyi Dong, Zhengde Wei, Ying Liu, Yu Pan, Franco Nori, and Xiaochu Zhang. Quantum reinforcement learning during human decision-making. *Nature Human Behaviour*, 4(3):294–307, 03 2020.
- [160] Earl K. Miller and Jonathan D. Cohen. An integrative theory of prefrontal cortex function. *Annual Review of Neuroscience*, 24(1):167–202, 03 2001.
- [161] Per Andersen. *The hippocampus book*. Oxford university press, 2007.
- [162] David S Olton, James T Becker, and Gail E Handelmann. Hippocampus, space, and memory. *Behavioral and Brain sciences*, 2(3):313–322, 1979.
- [163] Anthony Triche, Anthony S. Maida, and Ashok Kumar. Exploration in neo-hebbian reinforcement learning: Computational approaches to the exploration–exploitation balance with bio-inspired neural networks. *Neural Networks*, 151:16–33, 2022.
- [164] Bo Lu, Lu Liu, Jun-Yang Song, Kai Wen, and Chuan Wang. Recent progress on coherent computation based on quantum squeezing. *AAPPS Bulletin*, 33(1):7, 03 2023.
- [165] Xiaokai Hou, Guanyu Zhou, Qingyu Li, Shan Jin, and Xiaoting Wang. A duplication-free quantum neural network for universal approximation. *Science China Physics, Mechanics & Astronomy*, 66(7):270362, 06 2023.
- [166] Minyuan Zhao, Yun Chen, Qin Liu, and Shengjun Wu. Quantifying direct associations between variables. *Fundamental Research*, 2023.
- [167] Zeng rong Zhou, Hang Li, and Gui-Lu Long. Variational quantum algorithm for node embedding. *Fundamental Research*, 2023.
- [168] Li Ding, Haowen Wang, Yinuo Wang, and Shumei Wang. Based on quantum topological stabilizer color code morphism neural network decoder. *Quantum Engineering*, 2022:1–8, 07 2022.
- [169] Jinkai Tian, Xiaoyu Sun, Yuxuan Du, Shanshan Zhao, Qing Liu, Kaining Zhang, Wei Yi, Wanrong Huang, Chaoyue Wang, Xingyao Wu, Min-Hsiu

- Hsieh, Tongliang Liu, Wenjing Yang, and Dacheng Tao. Recent advances for quantum neural networks in generative learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10):12321–12340, 2023.
- [170] S. K. Jeswal and S. Chakraverty. Recent developments and applications in quantum neural network: A review. *Archives of Computational Methods in Engineering*, 26(4):793–807, 09 2019.
- [171] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3987–3995. PMLR, 08 2017.
- [172] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *CoRR*, abs/1606.04671, 2016.
- [173] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *CoRR*, abs/1705.08690, 2017.
- [174] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796, 2016.
- [175] Jason K Eshraghian, Max Ward, Emre Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D Lu. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, 111(9):1016–1054, 2023.
- [176] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.