Comparative Analysis of Quantum Reinforcement Learning in Autonomous Navigation

Khaled Ammoura
American University of Beirut
Maroun Semaan Faculty of Engineering
Beirut, Lebanon
kaa74@mail.aub.edu

Mohamad El Labban
American University of Beirut
Maroun Semaan Faculty of Engineering
Beirut, Lebanon
mhe56@mail.aub.edu

Mohamad Faour
American University of Beirut
Maroun Semaan Faculty of Engineering
Beirut, Lebanon
msf20@mail.aub.edu

***Abstract*—Autonomous navigation systems often face significant challenges in dynamic and uncertain environments. Systems that particularly rely on Reinforcement Learning (RL) methods can't handle such environments since RL approaches struggle with high uncertainty and convergence speed when handling complex parameters. Thus, Quantum Reinforcement Learning (QRL) emerged as a promising solution. This approach combines the principles of quantum computing and reinforcement learning, where it leverages techniques such as superposition and entanglement to enhance RL systems. This paper presents a comparative analysis of QRL applied to autonomous navigation tasks, where we focus on the advantages present and highlight the limitations. We examine various QRL techniques, such as VQCs, and measure them through metrics to assess key trends and highlight constraints. Our comparative analysis aims to provide readers with insight into QRL applications in autonomous navigation and the future trajectory of these systems.**

***Index Terms*—Quantum Reinforcement Learning, Autonomous Navigation, Variational Quantum Circuits, Comparative Analysis, NISQ, CARLA, Frozen Lake**

I. Introduction

As we observe in our world, autonomous navigation is becoming increasingly integrated with our devices with each passing day. From self-driving cars to robots capable of navigating rough environments, the algorithms driving these devices and machines continuously face the hard task of choosing the best decision to navigate complex, dynamic, and unexpected environments where the cost of a wrong decision may be severe.

Classical Reinforcement Learning (RL) methods, which learn decision policies through a feedback-reward loop while interacting with the environment, showed promise, particularly in such domains [1]. RL works by adapting its strategy, also called policy, according to the observable environment states. Based on the feedback, it modifies the policy in order to maximize the expected current and future rewards return.

Although this field seems promising, the approach it takes comes with its own shortcomings. As the complexity of real-world scenarios increases, classical RL begins to show serious limitations: long training times, difficulty scaling to high dimensional state spaces, and vulnerability to uncertainty remain pressing concerns [1]

In response to these challenges, the emerging field of Quantum Reinforcement Learning (QRL) offers a new perspective. QRL utilizes quantum computing principles—specifically superpositions and entanglement—to accelerate and improve RL's policy optimization and exploration phases. Theoretically, QRL may provide great algorithmic enhancements. By allowing simultaneous computations of states and environment configurations. This capability promises to make the learning more efficient and also provides potential for solving otherwise time-consuming problems in a timely manner [2]. Despite the promising potential, QRL in autonomous navigation is still in its early stages. Many different QRL strategies have been tested on simple benchmarks or in simulations that do not represent the real world or its dynamics fairly [4]. Moreover, the constraints that are imposed and assumed by the current-generation(NISQ devices) hardware devices limit the scale and ability of quantum computations [3].

II. Literature Review

A. RL in Autonomous Navigation

Early reinforcement learning work applied to navigation tasks made use of approaches such as Q-learning (only popular today due to the simplicity and effectiveness) and SARSA in grid-based environments, providing agents the capability to learn simple policies to maximize rewards in an unknown environment [4]. Although these basic reinforcement learning methods have strong decision-making abilities, their perception of the environment is weak [5]. Recent advances in machine learning, mainly deep learning, have enabled the development of algorithms such as Deep Q-Networks and Proximal Policy Optimization that enable agents to process complex inputs like images and rich state spaces [6]. Applications range from simplified driving simulators to robotic pathfinding tasks [5].

Despite these advances, RL and DRL have challenges of scalability, efficiency, and uncertainty. Such vast action spaces limit scalability in the case of real driving [7], while the generally slow convergence speed of the approach makes these models expensive computationally and time-consuming for deployment. Noisy observations and unpredictable obstacles in real traffic scenarios increase uncertainty and are the cause of policy failure under dynamic conditions [8]. Researchers have, therefore, proposed methods such as model-based RL, which integrates the dynamics of the environment into learning

to improve sample efficiency, and hierarchical RL, which decomposes tasks into manageable sub-tasks to handle complex state-action spaces [9].

B. From Quantum Computing to Reinforcement Learning

Quantum computing can offer a fundamentally new computing paradigm that may help circumvent challenges with RL. Mainly, the quantum properties, such as superposition and entanglement, allow multiple-state representations and their correlations, simultaneously, which may contribute to the optimization of policies faster or even more effective exploratory strategies [10], [11]. Although noisy hardware and decoherence in NISQ devices are major challenges with current quantum hardware, there is active research in hybrid quantum-classical methods to achieve incremental quantum advantages without fully error-corrected systems [11], [12]. For example, even the most constrained quantum devices available today can use quantum operations within RL frameworks in strategic ways that may offer improved performance compared to traditional methods [12].

C. Quantum Reinforcement Learning and Variational Quantum Circuits

Quantum Reinforcement Learning (QRL), as it stands today, arises from using the concept of superposition and entanglement in the RL pipeline. This shifts away from relying only on classical computation approaches and towards more quantum operations that yield visible improvements in terms of convergence speed and resource utilization. Currently, we see their implementation in confined, relatively simplistic environments like FrozenLake and MiniGrid. These environments allow for the observation of the potential of hybrid quantum-classical approaches [15]. demonstrated its success in partially observable environments, while [16] proved its usefulness in simple navigation tasks.

While these results sound promising, they are still far from being relevant to the real-world scale, limited by the hardware constraints of the current quantum hardware devices. However, experiments revealed that using VQCs in near-term quantum devices can provide performances comparable to classical methods with a significant reduction in the parameter space [16], [17]. One promising potential is the advantage provided by variational circuits relative to classical approaches in continuous control tasks, hinting at the potential for scaling to real-world applications [18].

D. Deep Reinforcement Learning in-context of Collision-Free Navigation

In the realm of Collision-Free Navigation (CNF) [17] systems the classical state-of-the-art approach is to formulate it as Partially Observable Markov Decision process (POMDP) [18] and solve it through DRL. The complexity of each environment may differ from a simple lane to end-to-end driving in urban scenarios. Knowing that, each environment may follow an independent architecture and algorithms to achieve their ultimate goal.

Many studies have been conducted to optimize and standardize a state-of-the-art baseline to be considered as the optimal solution for such problems. The first model to ever exist was Asynchronous Advantage Actor-Critic (A3C) [19], which was introduced by Volodymyr Mnih et al. on 2016 (even though the research emerged in 2015, the publication was in 2016). It is the first asynchronous actor-critic algorithm that uses multiple agents running in parallel to improve training efficiency, which leads to a faster training due to parallel processing and improves exploration and stability compared to synchronous methods that will be discussed later in this section. However, this algorithm faces some limitations due to the possibility of getting a stale gradient [20] due to delays between experience collection and policy updates. Moreover, it requires higher computational resource requirements due to parallelization. The CFN analog of A3C is the Globally Asynchronous Advantage Actor-Critic CADRL (GA3C-CADRL) [21], which is anextension of the first DRL-based CFN Model, Collision Avoidance with Deep Reinforcement Learning (CADRL) [22], which uses DRL for multi-agent collision avoidance in 2D environments. GA3C-CADRL uses the A3C algorithm for parallelized training to improve efficiency and stability, in which it focuses on dynamic multi-agent navigation scenarios. However, it has the same limitation as that of A3C. To address these limitations, in 2016, researchers created a synchronous variant of A3C, known as Advantage Actor-Critic (A2C) [23]. This approach performs synchronized updates between all agents to attain stable and consistent gradient estimates, which is particularly beneficial in tasks requiring precise control, such as CNF. However, this approach introduces new limitations where it appears to be slower compared to asynchronous methods and have less exploration efficiency [24]. Similar to A3C, A2C has a CFN analog which is Advantage Actor-Critic CADRL with Path Planning (A2C-CADRL-p). This algorithm is inspired by GA3C-CADRL, which combines symbolic path planning with the A2C algorithm for CFN in autonomous driving. It offers a balanced approach by combining advantages of each of A2C algorithm and symbolic path planning, in which it accounts for the stability of A2C algorithm with the deterministic nature of symbolic path planning for CFN. By delegating speed control to the DRL agent and steering to the path planner, A2C-CADRL-p simplifies the learning process and improves interpretability. However, it is bounded by the path planner's efficiency, since it has a huge overhead due to the combination path planning with DRL. Due to the synchronous nature of A2C, this approach results in a longer training period compared to that of GA3C-CADRL, since the model's scalability may suffer in large-scale multi-agent environments [25]. These limitations highlight the need for more adaptive, scalable, and efficient methods in CFN, particularly for complex and dynamic multi-agent environments.

In response, researchers have explored many advanced deep reinforcement learning algorithms such as Soft Actor-Critic (SAC) [26] and Proximal Policy Optimization (PPO) [27], which address issues of stability, exploration, and sample

efficiency. SAC is an off-policy actor-critic that maximizes both the cumulative reward (expected return) and the entropy of the policy by introducing entropy regularization to encourage exploration. In the context of CFN, SAC has been applied extensively on many complex environments where the autonomous agent had to perform dynamical adjustments to account for the chaotic entropy of the system due to the unpredictability of colliding with obstacles. Compared to A2C-CADRL-p, SAC has a noticeable advantage on it due in terms of exploration and stability, this is because of the huge drawbacks of higher computational complexity due to the need to manage multiple Q-networks. Concerning PPO, it prevents the policy from applying large, destabilizing changes. To ensure that, it uses clipped objective function [27] that ensures that the new policy doesn't deviate from the old policy, which leads to the avoidance of destabilization. Since CFN requires a balance between exploration and exploitation while preserving the stability performance of the system, PPO became a popular choice. Compared to A2C, PPO diminishes the catastrophic policy failures during training as it offers a more stable policy update. Therefore, PPO-based CFN models can achieve higher success rates and lower collision rates in both simulated and real-world environments.

In this paper, we are going to introduce in the section of CARLA [28], a classical DRL baseline known as, Navigation using Advantage Actor-Critic (NavA2C), which serves as the counterpartto a quantum-supported DRL model which is Navigation using Quantum Deep Reinforcement Learning (Nav-Q).

III. Methodology

A. Analytical Scope

Our approach is a comparative analysis that involves Quantum Reinforcement Learning (QRL) applications in autonomous navigation environments. Consequently, our objective is to study the current QRL applications by focusing on two benchmark problems: the CARLA simulator and Frozen Lake. We aim to provide insights into the state of the art of QRL in such environments by highlighting the advantages it offers over classical RL and the limitations that arise under today's hardware.

B. Literature Selection Criteria

For our comparative analysis, we will proceed with our selection process as follows:

Inclusion Criteria:

- The study must implement or analyze QRL.
- The work must evaluate performance on navigation-related tasks, explicitly involving Frozen Lake or the CARLA environment.
- The paper must present at least one quantitative performance metric.

Exclusion Criteria:

- Studies focusing solely on theoretical frameworks.
- Work that mentions QRL or related techniques but provides no evaluation on navigation tasks.

Using these criteria, we selected a set of studies that aim to represent a cross-section of the current state of QRL in autonomous navigation.

C. Chosen Benchmarks: Frozen Lake and CARLA

We selected Frozen Lake and CARLA as focal points for this comparative analysis due to their distinct characteristics and complementary roles in RL research:

Frozen Lake:

This is an OpenAI Gym environment that is a frozen grid-based navigation puzzle. The agent is rewarded for traversing from a start position to a goal position without falling through any holes. Additionally, the state transitions can be partially stochastic, as the agent moves across a slippery surface. This introduces uncertainty that tests the agent's adaptability. Frozen Lake's simplicity and controllability make it an ideal starting point for evaluating QRL approaches. Moreover, the uncertainty of the slippery grid is a great way to study how QRL reacts to uncertainty. Studies conducted on this environment can help demonstrate whether integrating quantum phenomena yields advantages in convergence, efficiency, or policy stability over classical RL methods.

CARLA Simulator:

CARLA is an open-source autonomous driving simulator. It offers a rich, dynamic environment with complex sensory inputs, variable traffic conditions, and unpredictable events that better approximate real-world complexity. QRL applications in such settings, like CARLA, are relatively new and uncommon. These applications are important for exploring whether the advantages observed in simpler benchmarks translate to more realistic domains. Performance in CARLA underlines the scalability of QRL, its adaptability, and its readiness for real-world navigation challenges.

This involves exploring both Frozen Lake and CARLA-oriented studies on a spectrum ranging from highly controlled, proof-of-concept experiments to more challenging simulations. This dual focus will enable us to investigate how the identified patterns persist, weaken, or evolve when the tasks become progressively more complex and/or realistic.

IV. Forzen Lake Comparative Analysis

A. Introduction

The testing ground for reinforcement learning algorithms we chose is Frozen Lake, which is essentially a very simple but stochastic grid-based maze. Classical RL methods, such as tabular Q-learning and DQNs, have proven successful at solving this environment. Their policies allow an agent to make consistent, correct decisions toward achieving the goal state. Meanwhile, potential emerging alternatives include Quantum Reinforcement Learning (QRL) approaches. QRL plans to minimize its complexity in parameters by using VQC and parameterized quantum models while theoretically matching or surpassing the convergence speed of classical RL without the present limitations of the hardware. Here in [1-6], the references considered approach to one extent or another the

scenario of Frozen Lake using NISQ-compatibility quantum circuits along with classical training loops.

B. Convergence and Sample Complexity

TABLE I
CONVERGENCE AND SAMPLE COMPLEXITY IN THE FROZEN LAKE ENVIRONMENT

Method	Iterations / Time Steps to Converge	Sample Complexity
Q-Learning	**80 iterations**	Low
DQN	290 iterations	Moderate
VQ-DQN	190 iterations	Low
HQC-PPO	10,330 time steps	Moderate
PQC Policies	Comparable to DQN	Low

Key Findings:

- **Q-Learning** converges most rapidly (80 iterations) under deterministic conditions, but exhibits limited robustness in stochastic environments [30].
- **DQN** manages stochastic transitions more effectively at the cost of higher sample complexity (290 iterations) [30].
- **VQ-DQN** achieves faster convergence (190 iterations) than DQN, benefiting from quantum-enhanced exploration [30].
- **HQC-PPO** attains strong performance in stochastic settings, though requiring 10,330 time steps, indicating moderate sample complexity [31].
- **PQC Policies** perform comparably to DQN, leveraging quantum-enhanced expressiveness with low sample complexity [32].

C. Success Rates and Robustness

All references point towards the fact that once trained, the QRL agents learn near-optimal and stable policies. According to Drăgan et al. [34], QRL agents learning in stochastic environments give stable maximum rewards and are able to keep up with uncertainty like classical PPOs. Chen et al. [33] as well as Lokes et al. [38] also confirm stable performance among various Frozen Lake formulations. Although these documents do not always measure success rates compared to classical RL, the thinking is that QRL has proven itself at par with classical methods for high-success-probability-high-policy-stability achievements.

D. Computational Efficiency and Parameter Complexity

The most turfed under QRL advantage is parameter efficiency. The parameter efficiency of Chen et al. [33] shows that QRL will save parameters substantially above tabular Q-learning (56.25) and fewer than would be needed by a classical deep network. Drăgan et al. [34] show even more dramatically by numbers that QRL uses orders of magnitude fewer parameters than classical neural networks. Also, Sun et al. [35] and Lokes et al. [38] emphasize QRL models that are structurally scalable with the problem linearly in parameters while classical methods scale quadratically (DQN) and cubically (tabular Q-learning). Moll & Kunczik [36] enlighten the present day; indeed, QRL is parameter-efficient now. But classical simulation of quantum circuits requires time and resources which are heavy right now. Meyer et al. [37] further reduce the number of optimization steps required per iteration for improved scalability and effective parameter usage for grander instances by incorporating warm-start initialization.

E. Limitations and Practical Considerations

Quantum Reinforcement Learning (QRL) appears gorgeous to lay an advantage but theoretical regarding real use with current Noisy Intermediate Scale Quantum (NISQ) devices. All the references talk about limitations: quantum simulators run slowly, while true hardware is noisy and depending on the limited number of qubits and the gate fidelity measures where Chen et al. [33], Moll & Kunczik [36], and Meyer et al. [37] acknowledge these overheads due to hardware and runtime, thus negating parameter saving and potential speed-ups.

However, it could be different in the future. Techniques such as amplitude encoding, as defined by Chen et al. [33] and Lokes et al. [38], would enable the exploitation of exponentially compressing parameters. As the number of qubits increases and the noise lessens, the theoretical merits of QRL—fewer parameters, potentially faster convergence, and stable policies—could gain significance in actual practice.

F. Conclusion

In a nutshell across all six references [33]–[38], QRL application to Frozen Lake testifies that QRL could at least perform as well as classical RL algorithms. In some situations, QRL outperformed DQN by comparatively faster convergence [34], [36], or could find more efficient circuit architectures [35]. Scaling linearly in parameter complexity is the strongest selling point, demonstrated consistently [33]–[35], [38], while warm-start techniques push scalability even further [37]. As of now, there is no large-scale, error-corrected quantum hardware, so all realized gains are largely theoretical and bound to simulation. Yet, the combined evidence says with strong conviction that as quantum hardware continues maturing, so would QRL as being one competent, parameter-efficient, and possibly faster alternative to classical RL even in the most canonical scenarios—like Frozen Lake.

V. CARLA COMPARATIVE ANALYSIS

In this section, we will conduct a comparative analysis between the classical DRL algorithm for CNF, NavA2C, which is "inspired by A2C-CADRL-p, combining a symbolic path planning with a modified A2C agent, such that the DRL agent is only responsible for the speed action" [39], with its quantum analog, Nav-Q, which is "inspired by NavA2C (Section 3.3), which comprises two major components: an Anytime Weighted Hybrid A* Path Planner, determining the steering angle for the car, and DRL components—a hybrid actor-critic model responsible for deter mining speed actions (accelerate, maintain, decelerate)" [39].

A. Architecture and Design

In this section we are going to explore different aspects of the architecture of each of the two algorithms.

TABLE II
COMPARATIVE OVERVIEW OF METHODS IN THE FROZEN LAKE ENVIRONMENT

Method (Reference)	Convergence Speed vs. Classical RL	Success & Robustness in Stochastic Frozen Lake	Parameter Complexity & Efficiency	Practical Limitations & Notes
VQ-DQN (Chen et al. [33])	Comparable to classical RL; no explicit claim of faster convergence vs. DQN, but stable convergence achieved	Stable policies, low variance in rewards, matches classical RL in policy stability	About 56% fewer parameters than tabular Q-learning; fewer than classical DQN	NISQ device constraints limit real-time advantage; relies on classical simulation
Hybrid QRL (PQC-PPO) (Drăgan et al. [34])	Converges faster than classical PPO-like methods; fewer episodes required with well-chosen circuit design	Achieves near-optimal rewards, stable under stochastic transitions	Significantly fewer parameters (e.g., 41 vs. 1245) than classical models	Complex to implement; quantum advantage depends on improved hardware
DQAS-Based QRL (Sun et al. [35])	10–30% faster convergence by automatically optimizing circuit architectures	Maintains stable performance, handles uncertainties similarly to classical RL	DQAS finds more parameter-efficient circuits, improving efficiency	Results are theoretical/simulated; requires better hardware and noise reduction for real-world gains
VQC-Based QRL vs. Q-Learning and DQN (Moll & Kunczik [36])	Faster convergence than DQN but slower than simple tabular Q-learning	Good policy stability once trained, performance matches classical RL eventually	Linear parameter scaling vs. classical quadratic/cubic scaling	Large simulation overhead; current quantum devices not practical for advantage yet
WS-VarQPI (Meyer et al. [37])	Warm-start reduces training steps per iteration, improving effective convergence rate	Achieves optimal policy for larger grids (8x8); stable performance	Parameter reuse (warm-start) enhances efficiency; few qubits and shallow circuits	Still reliant on classical simulation; real advantage awaits better hardware
VQ-DQN With Basis Encoding (Lokes et al. [38])	Under certain reward configs, converges in fewer episodes than classical methods	Strong stability across reward-structure variations	Linear parameter complexity, outperforms Q-learning (cubic) and DQN (quadratic)	Classical simulation overhead is large; amplitude encoding suggested for future compression

1) Path Planner: For determining the steering angle α_t, both NavA2C (Fig. 1) and Nav-Q (Fig. 2) utilize the Anytime Weighted Hybrid A* path planner. This approach allows the DRL agent to focus solely on speed control by offloading the steering task to a classical planner.

2) Feature Encoder: In both models, the feature encoder E_{s_i} consists of CNN layers that reduce the dimensions of the car intention I (bird's-eye view semantic segmentation). This encoding technique captures the important features of the environment and planned path, providing a basis for decision-making.

3) LSTM: Both models use an LSTM to capture the temporal dependencies. The LSTM's hidden state h_t compresses information from past and current inputs, helping the agent handle noisy or incomplete observations.

4) Actor π_θ: The actor network in both models is a dense (fully connected) neural network that produces a 3-dimensional vector representing the probabilities of speed actions {accelerate, maintain speed, decelerate}. Following this approach, it leads to any differences in performance being attributed to the critic component of the model.

5) Critic V_ϕ: In NavA2C, the critic is similar to that of the actor. It consists of a classical dense neural network that estimates the value function $V(s)$. Due to the determinism of this approach, the agent may suffer from limited exploration and struggles to escape local optima. On the other hand, in Nav-Q, the critic introduces a quantum sector to the model by using Parameterized Quantum Circuit (PQC), where the

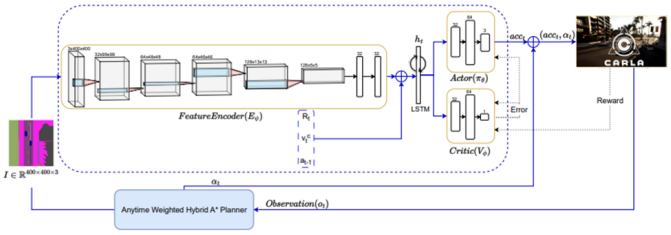

Fig. 1. NavA2C architecture

PQC leverages the quantum advantages like superposition and entanglement to examine a wider range of potential outcomes. During training, the quantum-enhanced critic improves the evaluation of actions, potentially leading to faster convergence and better exploration. However, due to the limitations of the quantum hardware, only the classical actor is used to make the deployment feasible.

B. Training Efficiency and Stability

In NavA2C, due to the synchronous gradient updates, where multiple trajectories are collected and the policy is updated based on the aggregated gradients, stability is ensured during the training phase, but it leads to longer convergence time. Moreover, due to the excessive gradient calculations across multiple trajectories before each update, the training time took longer. On the contrary, Nav-Q achieved faster convergence though better evaluation of value functions since it utilizes

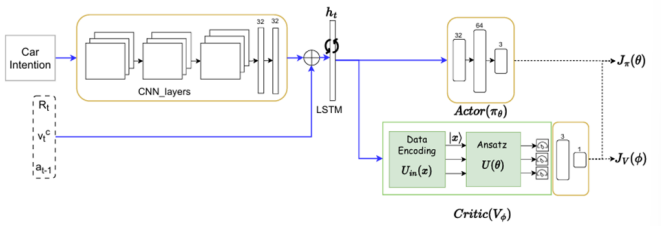

Fig. 2. Quantum-supported Deep Reinforcement Learning agent in Nav-Q. In contrast to the complete architecture of NavA2C illustrated in Figure 1, this depiction focuses solely on the DRL agent. The primary distinction from the classical model lies in the DRL agent's critic, which is replaced by a hybrid quantum-classical algorithm. Its design is essential for training and evaluating the loss function. [39]

the quantum properties using quantum circuits. Upon the usage of quantum enhancements, learning became more stable, especially in such a complex dynamic urban environment.

C. Exploration vs. Exploitation

In RL, there is always a trade-off between exploration (trying to new actions to discover better strategies) and exploitation (leveraging know strategies to maximize rewards), so achieving an effective balance is crucial for optimal performance. NavA2C and Nav-Q differ significantly in the way they handle this balance due to their respective architectures. NavA2C's critic tends to prioritize exploitation of the learned policy, in other words, once the critic finds a path that leads to a reasonable reward, it prefers to continue exploiting this path rather than exploring for better ones. The classical critic of NavA2C evaluates the policy deterministically, which ensures consistent evaluations, but it may get stuck in local optima. In dynamic or complex environment, this deterministic approach fails to identify better paths or strategies that require broader exploration, due to the unpredictable obstacles that the agent faces, the dynamical changes of optimal paths, and the diverse strategies the environment requires to succeed. Nav-Q's critic can handle a broader range of possibilities during training, which enhances its ability to explore new and potentially better strategies. Due to its quantum properties, it prevents the agent from getting stuck in a local optima since it provides richer and more diverse evaluations. By evaluating multiple paths concurrently, Nav-Q can explore better trajectories and adjust its policy accordingly. Moreover, CARLA consists of dynamic environments, uncertain scenarios and complex tasks, which makes Nav-Q a good approach in such systems.

D. Computational Complexity

When comparing both these approaches, the computational complexity should be considered. This factor would affect the feasibility of implementing these algorithms on real-world autonomous applications. Since NavA2C adapts in both actor and critic to the use of a classical neural network, it has relatively low complexity requirements, which make it suitable for real-time applications in autonomous applications. Due to its classical architecture, NavA2C can be deployed on standard hardware (CPUs/GPUs) without the need for specialized resources. Due to the adaptation of PQC in the implementation of Nav-Q, a huge complexity overhead was introduced during training. Simulating quantum circuits on classical computers requires substantial computational resource allocation as the parameters of the circuit increase (e.g., depth, number of qubits, etc.). Without access to an efficient quantum simulator or actual quantum hardware, Nav-Q would be impractical for real-time training due to the long processing times and high computational demands. During testing, Nav-Q uses only the classical actor for decision-making, which makes its deployment feasible on classical hardware.

E. Performance Metrics

In [39], they assessed the performance of Nav-Q with its classical baseline (NavA2C) through three categories of metrics:

a) Trainability:: It describes how well an RL model can learn to accomplish a specific task in a certain desired manner within its environment. Therefore, analyzing the Return vs. Episodes curve is considered as the standard approach to measure such a metric, which depicts the expected return smoothed over multiple episodes. Moreover, the mean and standard deviation of the Area Under the Curve (AUC) are calculated to compare the convergence rate and learning stability of different models.

b) Driving Policy:: The driving policy is evaluated using three metrics:

1) Overall Safety Index (SI): Number of cases where crashes and near-misses remain below 20%.
2) Crash and Near-Miss Rates (%): Measures the safety of the learned policy.
3) Time to Goal (TTG): Average time taken to reach the goal.

c) Learnability:: It refers to the ability of quantum and classical models to learn complex, diverse functions beyond the immediate problem at hand. To learn more about the way this metric was measured refer to [39].

F. Results

1) Trainability: In [39], they performed different configurations to choose the optimal architecture for Nav-Q and compare it with the classical baseline NavA2C, they ran various configurations of the PQC for the critic, varying the number of qubits and layers (fig. 3). They tested first a critic with a PQC using 4 qubits and 6 qubits and number of layers ranging from 1 to 3 (Table 1)(fig. 3), they concluded that 4 qubit circuit is more suitable, refer to [39] for more details. Afterwards, they compared the same 4 qubits circuits but with different number of layers. In conclusion, they found out the critic with a PQC using 4 qubits and 2 layers serves as the best model. Lastly, they compared the best configuration of Nav-Q with its classical counterpart NavA2C. In particular, they trained NavA2C for 10,000 episodes and compared it with Nav-Q. Results are shown in Figure 5.

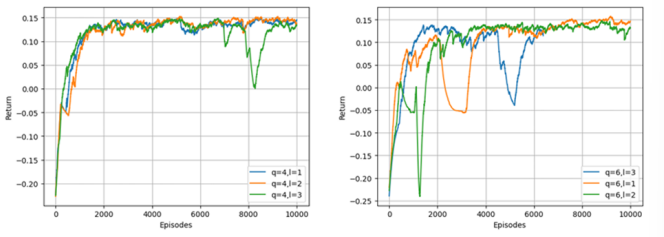

Fig. 3. Return vs. Episodes smoothed curves of Nav-Q with a 4-qubit (left) and 6-qubit (right) parametrized quantum circuit and number of layers ranging from 1 to 3. [39]

Method	#qubits	#layers	#parameters (d)	Training Time (days)
Nav-Q	4	1	29	6.8
		2	53	10.87
		3	77	14.64
	6	1	31	8.14
		2	55	11.74
		3	79	18.46

TABLE III
TRAINING TIMES AND NUMBER OF PARAMETERS FOR DIFFERENT CRITIC ARCHITECTURES OF NAV-Q

2) Driving Policy: Nav-Q achieves a slightly faster mean time-to-goal (TTG) compared to NavA2C. While NavA2C exhibits marginally lower crash and near-miss rates, both models achieve similar safety indices (SI). Nav-Q reveals a higher convergence rate in the first 4,000 episodes, meaning that the introduced quantum-enhanced critic allows the DRL agent to be faster in training than NavA2C compared to NavA2C. However, when training for long enough, both models achieve comparable driving policy, with similar average rewards. (fig .5-6) For measuring convergence rate and stability, six trainings with different weight initializations were performed. The results show:

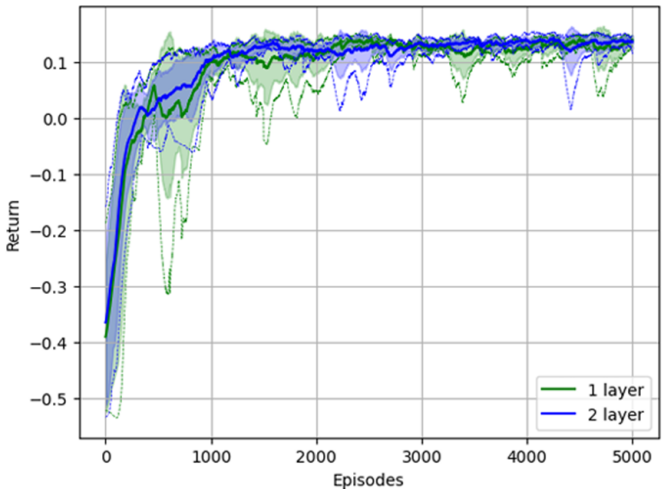

Fig. 4. Comparison of Return vs. Episodes graph of two different 4-qubit Nav-Q architectures with 1 and 2 layers respectively, executed for six different parameters initializations.

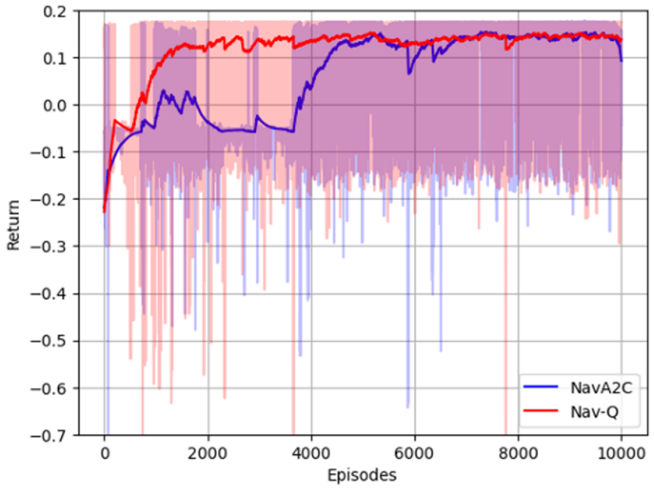

Fig. 5. Comparison of Return vs. Episodes graph of NavA2C and Nav-Q with 4 qubits and 2 layers. [39]

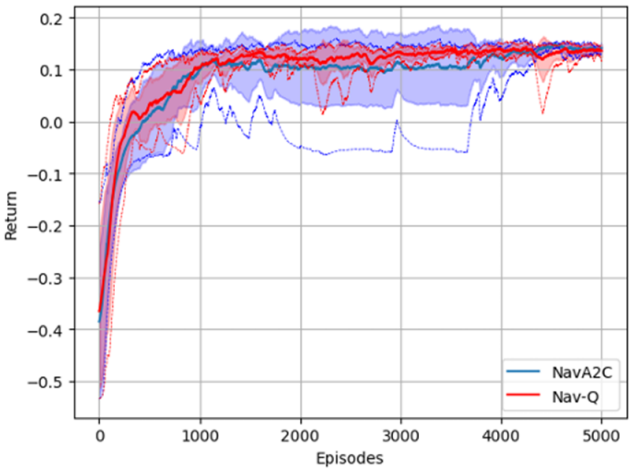

Fig. 6. Comparison of the smoothed Reward vs. Episodes curves for Nav-Q with 4 qubits and 2 layers, and NavA2C averaged over multiple runs. The red-colored plots in the graph correspond to Nav-Q while the blue colored plots refer to NavA2C. The thick solid line represents the mean of 6 runs, each with a different weight initialization of the overall architecture. The colored shaded area represents the standard deviation and the thin dotted line represents the maximum and minimum return values. [39]

- Convergence: Nav-Q achieves a higher average AUC (2440.6) compared to NavA2C (2362.95), indicating better overall performance improvement.
- Stability: Nav-Q exhibits a lower standard deviation (758.78) than NavA2C (902.73), suggesting greater stability across different initializations.

Overall, these findings emphasize that even though both models attain similar performance for long-term predictions, Nav-Q exhibits faster convergence speeds and greater stability during training.

3) Driving Policy: After training both Nav-Q and NavA2C models for 10,000 episodes, their driving policies were evaluated with the following results (Table 2)(fig. 9). Table 2: Comparison of driving policy of NavA2C and Nav-Q with 4

Method	#qubits	#layers	#parameters (d)	ED ($d_{\gamma,n}$)	Normalized ED $\widetilde{d}_{\gamma,n} = \frac{d_{\gamma,n}}{d}$
Nav-Q	4	1	29	8.14	0.28
	4	2	53	10.79	0.20
	4	3	77	13.30	0.17
	6	1	31	6.76	0.22
	6	2	55	9.11	0.17
	6	3	79	12.97	0.16
NavA2C	NA	NA	2305	56.65	0.02

Fig. 7. TABLE V Comparison of all the configurations tested for Nav-Q and NavA2C in terms of the number of parameters and the effective dimension [39].

qubits and 2 layers after10,000 episodes of training.

Method	#para-meters	Time to goal (s)	crashes (%)	near-misses (%)	Safety Index (SI)	Mean Return	Training Time (days)
Nav-Q	53	**11.74**	13.23	23.62	3	0.10	10.87
NavA2C	2305	11.77	**12.47**	**22.39**	3	**0.12**	**2.7**

TABLE IV
COMPARISON OF DRIVING POLICY OF NAVA2C AND NAV-Q WITH 4 QUBITS AND 2 LAYERS AFTER 10,000 EPISODES OF TRAINING.

TTG: Nav-Q achieves a slightly faster mean time-to-goal compared to NAvA2C, which concludes that the quantum enhanced model reaches the destination more quickly.

Collisions (Crashes and Near-Misses): NavA2C performs slightly better in terms of lower crash and near-miss rates compared to Nav-Q. However, the Scenario Index (SI), which measures the number of scenarios where crashes and near-misses remain below 20

Driving Policy Similarity: Despite differences in convergence rate and stability, both models learn similar driving policies, with no significant difference in test-phase behavior. Training Time: Nav-Q requires a notably longer training time compared to NavA2C due to the complexity of simulating the quantum circuit during each forward pass and performing standard backpropagation. This suggests that the current optimization strategy for quantum-supported architectures may need custom improvements to enhance efficiency.

4) Learnability: The learnability of the models were evaluated with the following results (Table 3): Normalized Effective Dimension (ED): The normalized effective dimension represents the proportion of active parameters relative to the total number of parameters in the model. Results in Table 3 show that Nav-Q has a higher normalized ED compared to NavA2C. This concludes that the quantum critic in Nav-Q utilizes a larger portion of the model space, demonstrating a greater capacity to model complex functions than the classical critic in NavA2C. Insights from ED: A higher Ed suggests that Nav-Q can potentially learn more intricate patterns, offering an advantage in tasks requiring the modeling of complex relationships.

G. Conclusion

Between NavA2C and Nav-Q, we find that Nav-Q outperforms all aspects of the comparison for CFN in CARLA due to having a quantum enhanced critic improving exploration, training stability, and convergence speed. Nav-Q surpasses NavA2C in dynamic and uncertain environments with lower collision rates, higher success rates, and reduced time-to-goal. NavA2C provide efficient computation and can be deployed in real-time structured environments, but its deterministic nature limits exploration. Therefore, for tasks demanding adaptability in complex and dynamic scenarios, Nav-Q is the better choice, provided the additional computational resources during training are available.

VI. Conclusion and Future Work

This paper explored the potential of Quantum Reinforcement Learning (QRL) in the field of autnomous navigation, starting from the basic simplistic environments, such as the *FrozenLake* environment, to more advanced and realistic environments such as *CARLASimulator*. The analysis revealed that QRL techniques may provide effective learning with fewer resources and increase flexibility when addressing situations with uncertainty, all of which are likely to be important in the context of many real applications. However, these benefits are challenged by the current capabilities of current-era NISQ hardware and the overhead of simulation, which have to be solved for QRL to be maximally useful.

For future research, the potential to develop QRL for multi-agent systems is an interesting topic that can be built upon this work. Employing the quantum property of entanglement would allow enhancing the communication between several QRL-based agents so that agents can devise decision-making strategies that are not classical and thus learn from others' experiences without having to go through the same experience. Further studies should explore how agents can manage to exchange information using entangled states and modify their strategies to account for the strategies of other agents that are constantly changing. This line of research can use quantum recursive algorithms, quantum communications, and various optimization techniques so that more advanced navigation challenges can be addressed in more complex systems.

References

[1] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," arXiv preprint arXiv:1904.12901, 2019.

[2] S. Y. C. Chen, "An Introduction to Quantum Reinforcement Learning (QRL)," arXiv preprint arXiv:2409.05846, 2024.

[3] J. Kim, "Quantum Reinforcement Learning: Concepts, Models, and Applications," in Intelligence of Things: Technologies and Applications, Springer, 2024.

[4] F. S. P. da Silva Neves, 'Reinforcement learning for robotic navigation in obstacle scattered environments," 2021.

[5] Y. Zhang, H. Wang, and J. Liu, 'Motion planning for mobile robots—Focusing on deep reinforcement learning: A systematic review," 2021.

[6] B. Balaji, S. Mallya, S. Genc, and S. Gupta, 'Deepracer: Autonomous racing platform for experimentation with sim2real reinforcement learning," 2020.

[7] C. R. Thompson, R. R. Talla, and J. C. S. Gummadi, 'Reinforcement learning techniques for autonomous robotics," 2019.

[8] A. Mohan, A. Zhang, and M. Lindauer, 'Structure in reinforcement learning techniques," 2023.

[9] W. Hu, H. Wang, and M. He, 'Uncertainty-aware hierarchical reinforcement learning for long-horizon tasks," 2023.

[10] M. Radic, 'Quantum-enhanced machine learning in the NISQ era," 2019.

[11] A. Sinha, 'Machine-learning driven transformations and analysis of quantum circuits for NISQ-era hardware," 2023.

[12] T. Kalamarakis, 'Hybrid quantum-classical algorithms and applications in scheduling problems," 2024.

[13] K. Kimura, A. Yoshikawa, and T. Sato, "Quantum reinforcement learning in partially observable environments," Proceedings of the International Conference on Quantum Computing, 2021.

[14] X. Chen, Y. Liu, and Z. Zhang, "Hybrid quantum-classical reinforcement learning for simple navigation tasks," IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 7, pp. 2121–2133, Jul. 2019.

[15] R. Lokes, P. Ahuja, and S. Gupta, "Variational quantum circuits for hybrid reinforcement learning: A case for near-term devices," Quantum Information Processing, vol. 21, no. 3, p. 91, Mar. 2022.

[16] Y. Lan, "Variational quantum circuits for continuous control tasks," Journal of Quantum Information Science, vol. 13, no. 2, pp. 127–141, 2021.

[17] Autonomous Robots Lab, "Collision-Free Navigation," Autonomous Robots Lab. 2024.
https://www.autonomousrobotslab.com/collision-free-navigation.html

[18] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and Acting in Partially Observable Stochastic Domains". 1998.
https://people.csail.mit.edu/lpk/papers/aij98-pomdp.pdf

[19] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning".
https://arxiv.org/abs/1602.01783

[20] J. Stanley and A. Jannesari, "Addressing Stale Gradients in Scalable Federated Deep Reinforcement Learning,".
https://doi.org/10.1145/3624062.3624170

[21] J. Everett, M. Shen, and J. P. How, "Motion Planning Among Dynamic, Decision-Making Agents with Deep Reinforcement Learning".
https://doi.org/10.48550/arXiv.1805.01956

[22] J. Long, T. Fan, and C. Liu, "Deep-Learned Collision Avoidance Policy for Distributed Multi-Agent Navigation".
https://doi.org/10.48550/arXiv.1609.06838

[23] S. Huang, B. Liang, and Y. Liu, "A2C is a Special Case of PPO".
https://arxiv.org/abs/2205.09123

[24] H.-C. Jang, Y.-C. Huang, and H.-A. Chiu, "A Study on the Effectiveness of A2C and A3C Reinforcement Learning in Parking Space Search in Urban Areas Problem".
https://ieeexplore.ieee.org/document/9289269

[25] M. Grzelczak and P. Duch, "Deep Reinforcement Learning Algorithms for Path Planning Domain in Grid-Like Environment," Nov. 2021.
https://doi.org/10.3390/app112311335

[26] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". 2018.
https://arxiv.org/abs/1801.01290

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms". 2017.
https://arxiv.org/abs/1707.06347

[28] CARLA Simulator.

[29] https://carla.org/

[30] M. Moll and L. Kunczik, "Comparing Quantum Hybrid Reinforcement Learning to Classical Methods," Universität der Bundeswehr München, 2023.

[31] T.-A. Drăgan, M. Monnet, C. B. Mendl, and J. M. Lorenz, "Quantum Reinforcement Learning for Solving a Stochastic Frozen Lake Environment and the Impact of Quantum Architecture Choices," Fraunhofer IKS, 2023.

[32] A. Jerbi, A. B. Latorre, S. Lloyd, and M. Cerezo, "Parameterized Quantum Policies for Reinforcement Learning," in NeurIPS 2021.

[33] S. Y.-C. Chen, C.-H. H. Yang, J. Qi, P.-Y. Chen, X. Ma, and H.-S. Goan, "Variational Quantum Circuits for Deep Reinforcement Learning," IEEE Access, vol. 8, pp. 141007–141024, 2020.

[34] T.-A. Drăgan, M. Monnet, C. B. Mendl, and J. M. Lorenz, "Quantum Reinforcement Learning for Solving a Stochastic Frozen Lake Environment and the Impact of Quantum Architecture Choices," arXiv:2212.07932, 2022.

[35] Y. Sun, Y. Ma, and V. Tresp, "Differentiable Quantum Architecture Search for Quantum Reinforcement Learning," arXiv:2309.10392, 2023.

[36] M. Moll and L. Kunczik, "Comparing quantum hybrid reinforcement learning to classical methods," Human-Intelligent Systems Integration, vol. 3, pp. 15–23, 2021.

[37] N. Meyer, J. Murauer, A. Popov, C. Ufrecht, A. Plinge, C. Mutschler, and D. D. Scherer, "Warm-Start Variational Quantum Policy Iteration," arXiv:2404.10546, 2024.

[38] S. Lokes, C. S. J. Mahenthar, S. P. Kumaran, P. Sathyaprakash, and V. Jayakumar, "Implementation of Quantum Deep Reinforcement Learning Using Variational Quantum Circuits," 2022 International Conference on Trends in Quantum Computing and Emerging Business Technologies (TQCEBT), pp. 1-4, 2022.

[39] A. Sinha, A. Macaluso, and M. Klusch, "Nav-Q: Quantum Deep Reinforcement Learning for Collision-Free Navigation of Self-Driving Cars," 2023.
https://arxiv.org/abs/2311.12875

[40] F. Pusse and M. Klusch, "Hybrid Online POMDP Planning and Deep Reinforcement Learning for Safer Self-Driving Cars," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1013–1020, 2019.
https://doi.org/10.1109/IVS.2019.8814125