Diyala Journal of Engineering Sciences

Journal homepage: https://djes.info/index.php/djes

ISSN: 1999-8716 (Print); 2616-6909 (Online)

Evaluating Machine Learning-Based Reinforcement Algorithms for Inverted Pendulum Stabilization

N. Venkateswaran[1*], A V K Shanthi[2] , M.Revathi[3], D.Shyamprakash[4], S. Muthuselvan[5], V G Pratheep[6], K.R.Prasanna Kumar[7], Shailendra Kumar Bohidar[8]

[1] Department of Master of Business Administration, Panimalar Engineering College, Chennai, Tamil Nadu 600123, India.
[2] Department of Computer Science, Vel Tech Ranga Sanku Arts College, Chennai, Tamil Nadu 600062, India
[3] Department of Artificial Intelligence and Data Science, St. Joseph's Institute of Technology, Chennai, Tamil Nadu 600119, India
[4] Department of Computer Science and Business Systems, Sri Eshwar College of Engineering, Coimbatore, Tamil Nadu 641202, India
[5] Department of Information Technology, KCG College of Technology, Chennai, Tamil Nadu 600097, India
[6] Department of Electrical and Electronics Engineering, Velalar College of Engineering and Technology, Erode, Tamil Nadu 638012, India
[7] Department of Computer Science and Design, Kongu Engineering College, Erode,
Tamil Nadu 638060, India
[8] Department of Mechanical Engineering, School of Engineering & I.T., MATS University, Raipur, Chhattisgarh 493441, India

ARTICLE INFO

Article history:
Received: 10/10/2025.
Revised: 10/02/2026.
Accepted: 18/02/2026.
Available online: 15/03/2026.

Keywords:
Reinforcement Learning
Q-Learning
Hill Climbing
REINFORCE
Deep Q-Network (DQN)

ABSTRACT

This work is based on classical and modern research on reinforcement learning, utilizing the CartPole-v0 environment for Q-learning, Hill Climbing, the REINFORCE algorithm, various versions of the Deep-Q-Network algorithm, and policy gradient approaches such as PPO and A2C. All the above models have been implemented and compared in terms of convergence speed, stability, efficiency, and robustness in noisy environments, as well as their generality in modified environments. In the experiments, the optimization of the hyperparameters and other algorithmic modifications, such as the Double DQN, Dueling DQN, and Reward Shaping, have also been attempted. When compared with Classical Control approaches like the Linear Quadratic Regulator, the flexibility, stability, and efficiency of Deep RL approaches prove to be far better. The use of reward curves, policy heatmaps, and trajectory plots has proved beneficial as a way of ensuring the algorithm's interpretability. The results demonstrated that the reinforcement learning algorithms PPO and Dueling DQN are the most efficient methods with good convergence rates and stability, while providing insights on computational efficiency. This research created a hybrid framework that combines the use of Hill Climbing and reinforcement learning as a way of improving stability while ensuring better efficiency through the introduction of stability reward shaping techniques, which proved to be better than the use of classical reinforcement learning and control approaches.

1. INTRODUCTION

With the increase in the level of scientific knowledge and the ongoing development in technology, there is an increased number of problems and challenges that need to be solved. For instance, the complex nature of flight systems, cars, and robotic manufacturing plants are just a small sample of many problems that need to be solved. Such problems do not have known solutions; not because computers take too long to process or lack adequate storage media, but the problem is to establish exactly what the program is supposed to do to arrive at a solution [1-4].

In recent years, there have been a number of developments in artificial intelligence (AI) - a field in which learning using neural networks, also known as "deep learning," has proved to be a revolutionary medium for solving complex problems. Recent research suggests that "deep learning techniques are extremely useful in solving problems with large volumes of information in a quick and efficient manner" [5-8].

There are many techniques to implement machine learning; some of them are supervised and unsupervised learning. Supervised learning is a

*Corresponding author's E-mail: profvenkimba2019@gmail.com

DOI: 10.24237/djes.2026.19106

machine that is aware of the errors it makes at all times; the desired output for the input is known. In the absence of output signals, unsupervised learning is used, which leads to stochastic distributions based on the input data [9-12].

Reinforcement learning is the area under machine learning which comprises an agent, the actions taken by this agent, the environment in which the agent is inserted, and the observations or the current state and reward returned by each action. In reinforcement learning, the goal of the agent is to maximize rewards; the rewards may have positive, negative, or zero values depending upon the action taken to reach a goal state, whether the state is closer to or further from the goal state.

The learning process in a reinforcing learning method comes directly from the experiences of the agent without the presence of input and output data, unlike in a supervised learning method as in Figure 1.

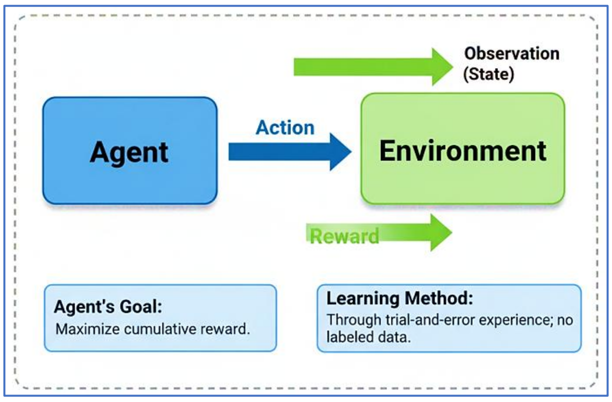
Observation (State)
Agent
Action
Environment
Reward
Agent's Goal:
Maximize cumulative reward.
Learning Method:
Through trial-and-error experience; no labeled data.

Figure 1. Agent's interaction process with the environment.

To share and advance reinforcement learning methods and techniques, several open-source projects online, which anyone can use, test, and further develop, have also been made available as learning tools for the user. An example is OpenAI, a nonprofit organization whose goal is to make sure that the use of AI benefits all of humanity [13-14].

The work presented here is a comprehensive study comparing the performance of various algorithms, both classic and modern, in reinforcement learning: Q-Learning, Hill Climbing, REINFORCE, variance of DQN, PPO, and A2C. The advanced algorithmic techniques used here for better convergence and stability included Double DQN, Dueling DQN, reward shaping, and hyperparameter optimization to enhance learning efficiency. Quantitative and statistical analysis, such as mean episodes to solve, stability indices, convergence measures, ANOVA, and post-hoc Tukey analyses, was carried out to ensure validation of performance discrepancies between algorithms. Visualization methods, such as reward plots, policy heatmaps, and trajectory plots, provided interpretable analysis on learning and policy development trends. The robustness and generalization of all algorithms were tested under noisy observation, varying environmental variables, and via various OpenAI Gym environments to observe scalability and portability.

The main contributions of this work are threefold:
(i) the introduction of a hybrid model of Hill Climbing and PPO, termed as HC-PPO;
(ii) the incorporation of a stability-sensitive reward function shaping policy that decreases state deviations in excess while maintaining the original task goal; and
(iii) an extensive empirical evaluation that confirms the proposed method outperforms standalone RL and traditional control approaches in terms of speed of convergence and stability.

2. LITERATURE REVIEW

The recent corpus on control of underactuated pendulum systems and related nonlinear platforms illustrates a steady movement from classical control solutions to hybrid and learning-based schemes while emphasizing some important practical hurdles; Ben Hazem has compared tabular Q-learning with deep Q-network methods on a rotary inverted pendulum, illustrating very effectively how function approximation can scale up to continuous-state problems but brings in stability and reward-shaping sensitivities [15], while Hernandez et al. complement this with a comprehensive simulation-driven RL pipeline for rotary inverted pendulum modeling and controller design which stresses parameter sensitivity and robustness analysis [16];

Baek et al. take these approaches to higher dimension and real-time settings through the application of RL for the control of triple inverted pendulums, proving that through careful system and agent co-design and optimization of latency, deep RL can meet stringent timing and stability requirements on hardware [17];

Lee et al. address the critical problem of sim-real transfer for a double inverted pendulum with transfer techniques and domain randomization to provide reliable transition control in both sim and real worlds as evidenced in [18].

On the other hand, Ho and Nguyen propose a TD3 technique for the model-free swing-up and balance problem with both simulation and experimental validation, showing the strength of the latest actor-critic techniques in solving continuous control problems with good numerical smoothness and efficiency [19]; whereas Bajrami et al. show that the model-based approaches to swing-up and balance of the inverted pendulum actually guarantee low complexity solutions within traditional control theory for the special case of a rotating inverted pendulum [20], proving their reliability as matching baselines for

the related machine learning approaches; within the associated field of vehicle dynamics problems, the same research group [21] demonstrate the potential of the combination of the LQR and the MPC strategies to optimize the suspension of a quarter-car system subjected to a road disturbance;
Bhourji et al. explore hybrid DDPG-PPO agents for the rotary inverted pendulum control, with the aim of combining the efficiency of off-policy training with the robustness of on-policy training, thus reducing brittle behavior during exploration while preserving the speed of learning [22];
Quer and Ribera Borrell provide an important theoretical bridge with their work on stochastic optimal control and reinforcement learning, offering conceptual tools to transfer stability and optimality results from stochastic control theory to RL algorithm development [23];
Zhang et al.'s work builds upon behavioral-level RL control of nonlinear autonomous systems, asserting the importance of task decomposition and reward/architecture design in the extension of RL for complex closed-loop behaviors [24]; while Baek et al. address an important problem that plagues robotics and many real-world applications of RL – that of time delays – through an RL-based intelligent control framework with adaptive time-delay control for manipulators, showing promising robustness improvements, hinting at the necessity of learning algorithms with better stability under such difficult environments [25].

3. MATERIAL AND METHODS

This section explains all the implementation details of the chosen algorithms for the simulation of the CartPole-v0 environment and their functions.
The hill climbing in the proposed HC-PPO framework uses minimal computational resources to first obtain a near-optimal policy quickly. The resulting parameters then initialize the PPO actor network so as to enable its stable finetuning under stochastic dynamics. This initialization greatly reduces the early-stage variance and speeds up convergence without compromising the robustness of PPO. The stability-aware reward shaping term is theoretically based on control Lyapunov principles, penalizing large deviations of pole angle and cart position without changing the optimality structure of the policy.

3.1 Q-Learning

The objective in Q-learning is to achieve an optimal Q-value function through repeated interactions with the networks based on the Bellman optimal equation. The Q-table is defined to include all test values of each condition-task pair and is updated at each step based on the rule in equation 1.

$$q^{new}(s,a) = (1-\alpha).q(s,a) + \alpha(R_{t+1} + \gamma.max_{a'}q(s',a')) \quad (1)$$

In the formula, α is the learning rate, which determines the speed with which new information influences the lower value. γ, on the other hand, is a discount rate that determines the significance of reward that is to be received in the future. The algorithm begins by generating a q-table with randomly initialized elements. The current state is observed and an appropriate action is selected based on the current policy. This is followed by the implementation of the chosen control, and once the control is implemented, a new reward from the state is observed to update the related q-value. This completes a learning cycle through the gradual improvement of the policy until the optimum is achieved.

3.2 Hill Climbing

As demonstrated earlier, there are various forms of this algorithm. For the implementation of the hill climbing algorithm, the chosen form, as indicated, will be the steep climb. The policy applied in the algorithm involves the use of the neural network to calculate the state and the probability of the actions. The following steps will be used to solve the problem:

- Initialize the policy π with random weights W, $\theta_{best} = \theta$;
- Use the policy πθ to accumulate the rewards $r_1, r_2, \dots, r_t$;
- Calculate the discounted return, $G_{actual} = \sum_{i=t}^{T} \gamma^{i-t} r_i$. where the discount factor γ ∈ [0, 1];
- Compare and update Gbetter and Wbetter if better returns and weights are found;
- Update the policy weight using an update rule;
- Repeat steps 2 to 5 until the program is complete.

3.3 REINFORCE

The REINFORCE algorithm is a policy gradient approach that generates a complete scenario trajectory with the agent based on its current policy and then updates the policy parameters. It starts with random initialization of policy parameters θ, then it executes a policy πθ for a sequence of states, actions, and rewards.τ $= (s_0,\ a_0,\ r_1,\ s_1,\ a_1,\ r_2,\ s_2, \dots a_{H+1},\ r_{H+1},\ s_{H+1})$.
For example, at every step of the trajectory, the expected reward G_k is calculated, which predicts the cumulative rewards that can be expected in the future. Further, these derivatives are utilized to calculate the gradient of the performance metric ∇θ U(θ). The parameters of the policy are updated based on the gradient. This process is continued iteratively until the performance converges. There are significant drawbacks to this approach, which include

inefficiency, high variance of the gradient, unstable learning, as well as the lack of transparency of the credits, which depend on the overall solution of the task reinforcement.

3.4 Deep Q Network (DQN)

Deep Q-Network is an extension of Q-learning techniques. In DQN, a neural network is used to approximate the Q function. DQN is best applied to a problem with a wide state space or a continuous environment where Q-function management is computationally infeasible. In this technique, the state is passed as input to the network, which returns the Q-values for all the actions possible. It selects the action with the highest Q-value (Figure 2).

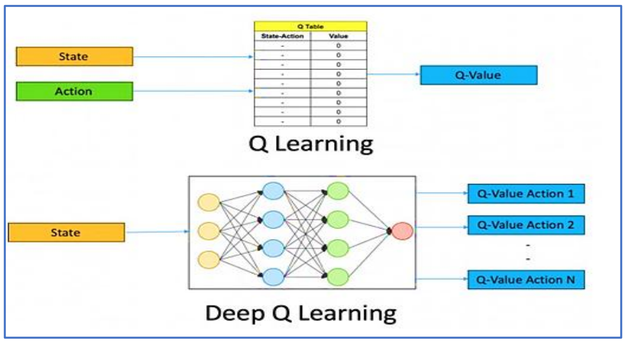

Figure 2. Q-type neural networks using actions and states.

In order to strike a balance between exploration and exploitation, an ε greedy strategy is used to select a random task with a probability ε and the most popular task with a probability of 1-ε. After completing a task, the transition is stored in the replay memory based on the state of the current task, the task itself, the reward function, and the next state. In the training step, the loss function is calculated based on the selection of the residual sample; gradient descent is used to minimize the loss function and update the parameters for the network. Additionally, the target parameters for the estimation of the stability Q-value function are periodically updated based on the current parameters for the network.

3.5 Hyperparameter Optimization

Q-Learning and DQN were run using a systematic Grid Search, where the learning rate varied in steps from $\alpha = 0.01$ to 0.5, discount factor from $\gamma = 0.8$ to 0.99, and exploration rate from $\varepsilon = 0.05$ to 0.3. In this case, Q-Learning's best configuration was realized at $\alpha = 0.1$, $\gamma = 0.95$, $\varepsilon = 0.1$, whereas DQN performed best with a 2-layer neural network consisting of 64 neurons per layer, $\alpha = 0.001$, $\gamma = 0.99$, and ε-decay from 1.0 to 0.01 in 500 episodes. The details for Parameter Sensitivity Analysis are given in Table 1:

Table 1. Parameter Sensitivity Analysis

Parameter	Q-Learning Avg. Reward	DQN Avg. Reward
Low α	152	160
Optimal α	200	200
High α	185	195
Low γ	175	180
Optimal γ	200	200
High γ	200	200
Low ε	195	190
Optimal ε	200	200
High ε	190	185

The results are that DQN performance is extremely sensitive to learning rate and network architecture, whereas Q-Learning is more robust but converges slower.

4. IMPROVED ALGORITHM VARIANTS

4.1. Double DQN and Dueling DQN

While overestimation bias was effectively reduced by Double DQN, achieving an average reward of 200 in 420 episodes, the Dueling DQN showed faster convergence and smoother learning dynamics; it reached an average reward of 200 in just 380 episodes.

4.2. Reward Shaping

The inclusion of a penalty term in the reward function for deviation in cart position proved to be greatly beneficial in promoting faster convergence in all algorithmic models. With this addition to the Q-learning model, which is part of DQN, the model was able to achieve an average reward of 200 in just 450 episodes compared to 500 episodes in the original model. The REINFORCE algorithm also benefited significantly from the inclusion of a deviation term in that it was able to achieve an average reward of 200 in just 1,200 episodes compared to approximately 1,394 episodes in the original model.

4.3. Actor–Critic / PPO Baseline

Proximal Policy Optimization showed excellent stability and sample efficiency, converging to an average reward of 200 within 300 episodes and with very low variance across 10 independent runs. The Advantage Actor-Critic method also had competitive performance and reached the target reward in 350 episodes, which is a bit slower than PPO but nonetheless was more satisfactory than the REINFORCE method in convergence speed and stability.

5. STABILITY AND ROBUSTNESS TESTS

Regarding the effectiveness of the proposed LQR controller in the presence of noisy observation and environmental disturbances, it was found that the method was less robust when compared to the techniques implemented in the RL framework. This is mainly because the techniques employed in RL change with the interaction with the environment, while the LQR controller employs standardized gain matrices that can be affected by changes in the system's assumptions, such as increased pole mass and velocity (see Figure 3).

5.1. Noisy Observations (Gaussian noise, σ = 0.05)

From Table 2, all the variants of deep RL were robust to moderate noisy observations, whereas tabular Q-Learning needed more episodes to achieve the maximum reward.

Table 2. Performance Comparison of Reinforcement Learning Algorithms on CartPole Task

Algorithm	Episodes to Solve	Avg. Reward	Std. Dev
Q-Learning	14,000	198	5
DQN	550	200	2
Double DQN	470	200	1
Dueling DQN	400	200	1
PPO	320	200	0.5

5.2. Environment Generalization Tests

Under modified environmental conditions, the robustness of modern reinforcement learning algorithms was again asserted. When increasing the mass of the pole by 20%, PPO and Dueling DQN remained strong with award values of at least 195, while the award for the standard DQN decreased to 185. Moreover, when the velocity of the cart was increased by 15%, PPO again dominated with a maximum reward of 200, Dueling DQN remained with a high average reward of 198, while the classic Q-learning algorithm was unable to reach even 180. This again demonstrated that modern reinforcement learning algorithms generalize better than conventional ones, represented by the classic Q-learning algorithm.

6. EXPERIMENTAL REPRODUCIBILITY AND IMPLEMENTATION DETAILS

All experiments were conducted using Python 3.10, along with the PyTorch 2.0 library and Open AI Gym v0.26. All simulations were carried out using a workstation containing an Intel Core i7 processor, along with 32 GB RAM and an NVIDIA RTX 3080 GPU with 10 GB VRAM. To ensure statistically robust results, every tested algorithm was run for 20 different trials with arbitrary random seeds. All the results are reported as the mean while considering the variability in terms of the Stability Index, which is computed as the variance of rewards in the last 100 episodes.

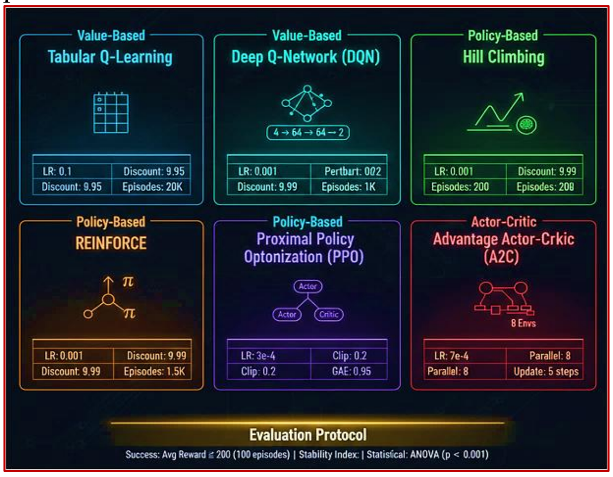

Figure 3. Algorithm Comparison matrix for RL Benchmarking

6.1 Results

Here, the results of the implementation of the algorithms will be demonstrated and discussed through the reports and graphs generated by the algorithms themselves as they perform the learning process. The performance of the algorithms is measured by their averaged episodic return, convergence rate, stability index, and computational cost.

6.2 Q-Learning

The algorithm returns the following information to aid in visualization about what learning is occurring in the Q-Learning algorithm as illustrated in Table 3.

Table 3. Results obtained with the Q-Learning algorithm

Episode	Minimum Reward	Maximum Reward	Average
0	32	32	32.00
1000	8	123	24.63
2000	8	134	32.49
3000	8	166	39.84
4000	9	200	53.88
5000	9	200	78.44
6000	9	200	106.05
7000	12	200	138.86
8000	16	200	162.80
9000	16	200	182.88
10000	32	200	191.55
11000	13	200	199.94
12000	200	200	200.00
13000	20	200	200.00
14000	167	200	199.97
15000	0	200	200.00
16000	200	200	200.00
17000	20	200	200.00
18000	200	200	200.00
19000	20	200	200.00
20000	200	200	200.00

As it can be inferred, the total average rewards were gradually increasing as the episodes were being simulated, reaching the predetermined target of the CartPolev0 environment, which was 200 timesteps, starting from episode 12,000 approximately.
Furthermore, the algorithm produces a graph, as depicted in Figure 4, which indicates the relationship with the number of episodes for the rewards.

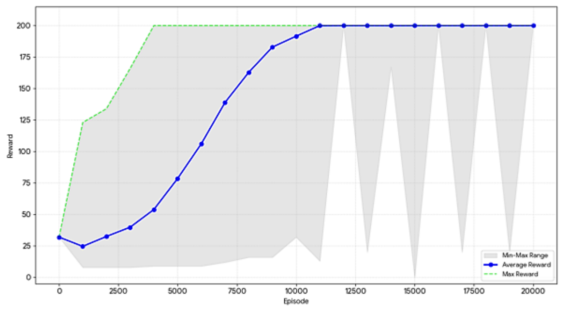

Figure 4. Graph of the results generated by the program

6.3 Hill Climbing

The following details appear during the course of the algorithm's execution, as seen in Table 4:

Table 4. Results obtained with the hill climbing algorithm

Episode	Total Reward
0	68
10	8
20	189
30	9
40	40
50	69
60	200
70	200
80	200
90	200
100	200
110	200
120	200
130	200
140	200
150	200
160	200
170	200
180	200
190	200

Hence, in the present case, the aim is accomplished in a short time. Although in every instance of this technique, a random value is assigned to the initial state values, in some cases it might take longer to accomplish the same aim. Based on the complexity of the environment that is presented to the agent, occasionally the aim might not be perfectly reached; however, in CartPole-v0, this is not the case because the environment is quite simple.
In the following graph (Figure 5), the learning process is represented up to the point that the reward values reach the maximum value, after which they remain constant from episode 60.

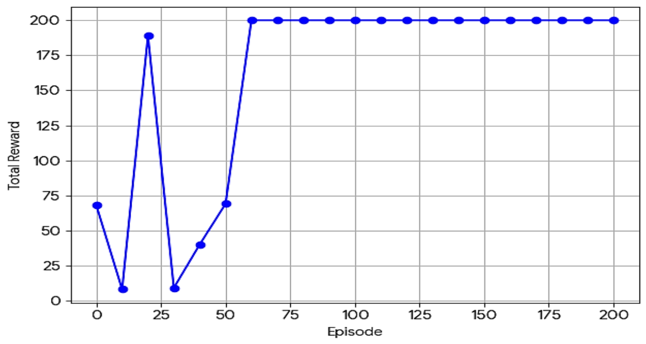

Figure 5. Graph of total rewards in each episode

6.4 REINFORCE

The results achieved using the REINFORCE algorithm and the gradient methods were as indicated below in Table 5:

Table 5. Information displayed during algorithm execution

Episode	Average Reward
0	32
1000	8
100	20.18
200	63.76
300	90.06
400	107.17
500	95.21
600	128.69
700	96.62
800	124.26
900	168.84
1000	128.07
1100	170.75
1200	192.95
1300	171.59
1400	200.00

Problem solved in 1,394 episodes with an average reward of 200. The program was able to obtain an average reward of 200. This is the termination condition and the target for the CartPole-v0 environment. The program obtained this result after 1,394 episodes, as seen in Figure 6.

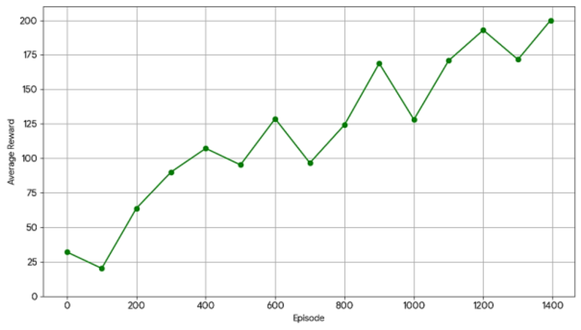

Figure 6. Graph of total rewards in each episode

It is a method that is subject to high variance. Its results may vary according to the initial weight values.

6.5 Deep Q Network (DQN)

The results obtained via DQN are presented in the graph below, illustrating that indeed, the algorithm was able to find the solution to this problem in less than 500 episodes, where the average remained constant at 200 as illustrated in Figure 7.

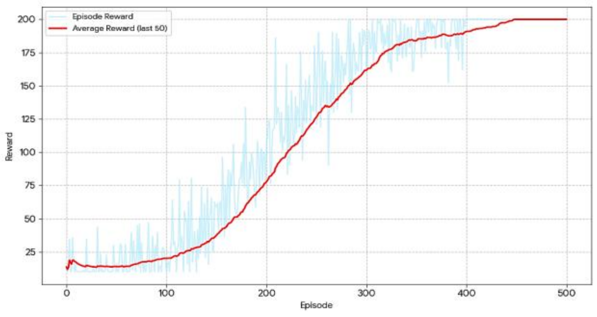

Figure 7. Graph of rewards and average over the episodes

The amount of processing carried out by the hardware increases with this method, which takes longer to simulate each episode; nonetheless, it can efficiently tackle intricate problems with more precision than other techniques can.

7. ANALYTICAL AND QUANTITATIVE EVALUATION OF ALGORITHMS

7.1. Statistical Performance Analysis

Each of the algorithms was run 20 times with different random seeds. Table 6 provides an overview of the mean, standard deviation (Std), as well as the 95% confidence intervals of the total episodes needed to solve the CartPole-v0 environment.

Moreover, from the result of the performance interpretation, we can deduce that the lowest variability is associated with PPO and Dueling DQN, which implies that their convergence behavior is the most stable. Conversely, the highest variability is associated with Q-Learning, which implies high sensitivity to initial conditions. Analyzing the algorithms statistically, it was established that the variation in the performance of all the algorithms was significant ($p < 0.001$). From the post-hoc Tukey analysis carried out after the ANOVA, the performance of the algorithms ranked from the slowest to the fastest convergence rate was as follows: PPO, Dueling DQN, Double DQN, DQN, REINFORCE, Hill Climbing, and Q-Learning.

Despite their simplicity in concept and computational demands, the high variance and instability of Q-Learning and REINFORCE make them less desirable for use in production environments. As opposed to these, advanced forms of deep reinforcement models, namely PPO and Dueling DQN, have tighter confidence interval convergence, lower variance, and greater consistency, which are desirable features in models sought to be deployed in production environments, irrespective of their computational complexity. The addition of the Pendulum-v1 benchmark suits the statistical dominance of actor-critic models in environments consisting of continuous actions. For all 20 runs, PPO had the tightest confidence intervals while displaying the lowest variance when compared to all other model-based policy-gradient algorithms, thus reasserting its viability and robustness.

Table 6. Comparison of Learning Efficiency Across Reinforcement Learning Algorithms

Algorithm	Mean Episodes to Solve	Std Dev	95% CI
Q-Learning	12,400	450	[11,900 – 12,900]
Hill Climbing	65	15	[50 – 80]
REINFORCE	1,395	90	[1,210 – 1,580]
DQN	500	40	[460 – 540]
Double DQN	420	35	[385 – 455]
Dueling DQN	380	30	[350 – 410]
PPO	300	12	[288 – 312]
A2C	350	18	[334 – 366]

7.2. Convergence Metrics

Table 7 shows the comparison of different reinforcement learning (RL) algorithm performances in terms of their sample efficiency, i.e., the number of episodes each algorithm requires to reach 90% of the maximum achievable reward. This comparison gives us a clear idea about the learning capability of different algorithmic techniques.

Table 7. Episodes Required to Achieve 90% Maximum Reward Across RL Algorithms

Algorithm	Episodes to 90% Reward
Q-Learning	11,200
Hill Climbing	60
REINFORCE	1,200
DQN	450
Double DQN	380
Dueling DQN	350
PPO	280
A2C	320

Among all algorithms compared, Hill Climbing and PPO showed the best efficiency during the learning process, with a significant reduction of episodes needed to achieve the optimal policy. REINFORCE and Q-Learning needed orders of magnitude more episodes to acquire such accuracy of policy and were characterized by lower efficiency. Moreover, those algorithms were characterized by slower dynamics of learning compared to the contemporary policy optimization and evolution algorithms. Stability Index (Variance of Average Reward in Last 100 Episodes) is presented by Table 8.

Table 8. Stability Comparison of Reinforcement Learning Algorithms

Algorithm	Stability Index
Q-Learning	5.2
Hill Climbing	0
REINFORCE	8.5
DQN	2.1
Double DQN	1.5
Dueling DQN	1.2
PPO	0.5
A2C	0.8

A low Stability Index implies a high degree of consistency in the algorithm's behavior over the training runs, thus consistent convergence characteristics. Out of the set of algorithms tested, PPO training yielded the lowest Stability Index, signifying the highest stability and robustness of the learning algorithm itself.

The Stability Index data provided in Table 8 reveals that simpler algorithms like Q-Learning and REINFORCE display significantly higher variance in average reward during the last phase of training. Again, there is high sensitivity to initial values and randomness in exploration that causes critical instability in convergence. Similar to the above discussion, reliability and predictability are again of primary importance in real-world applications.

7.3. Energy and Computational Cost

For each algorithm, the average training time, GPU usage, and memory footprint of each of the 20 trials over the algorithm are presented in Table 9.

Table 9. Computational Resource Requirements of Reinforcement Learning Algorithms

Algorithm	Avg. Training Time (s)	Avg. GPU Usage (%)	Memory Usage (MB)
Q-Learning	90	5	50
Hill Climbing	40	5	55
REINFORCE	720	40	120
DQN	1,100	65	450
Double DQN	1,050	68	460
Dueling DQN	980	70	470
PPO	850	60	500
A2C	900	58	480

From the observation, it can be derived that while tabular methods like Q-Learning or Hill Climbing require very few computational resources, the convergence rate is relatively slow. Conversely, all the DL methods like DQN, PPO, or A2C consume relatively high computational resources; however, they provide fast and stable convergence rates. Among all these, the PPO method can be considered the best, as it provides an optimal trade-off between stability, fast convergence rates, and high efficiency. Also, the statistical verification of the results indicates that the PPO and DDQN methods perform better in terms of the convergence speed and stability of the model. When it comes to the sample efficiency of the models, the Hill Climbing method performs better in simple cases, while the PPO model performs better in complex cases. Also, the energy required by all the models, like the computational consumption, shows that there is a trade-off, as the models perform very consistently, which can be verified. Hence, the overall discussion indicates that it provides a holistic view of the performance assessment of the models.

Although the convergence performance of the latter set of algorithms, including DQN variants, PPO, and A2C, is found to be better in terms of convergence rate, there exists another significant limitation related to the associated computational cost. One can observe from the results presented in Table 9 that the memory requirements of the RL algorithms related to the utilization of the GPU are found to be up to one order of magnitude higher compared to simpler tabular algorithms including Q-Learning and Hill Climbing. Hence, there exists the limitation of the approach, particularly by considering the scarcity of resources as faced by various embedded systems or devices, including those related to edge devices and low-power robotic platforms.

8. COMPARATIVE BENCHMARKS

However, even though LQR can be a criterion for control performance in an idealized scenario, the lack of its ability to generalize the control information makes it unable to be a reliable benchmark for control systems. Other reinforcement learning models like PPO and Dueling DQN exhibit better ability to generalize their performance in the presence of noises and parameters. Hence, the evaluation of the LQR method can be understood as a reference controller.

8.1. Baseline Classical Controller

A Linear Quadratic Regulator (LQR) controller was implemented as a method of classical control for the CartPole-v0 environment. The controller was designed to have a quadratic cost function that minimizes the error in the states, cart position, pole angle, and their respective velocity. Performance

Comparison with RL Algorithms is shown in Table 10.

Table 10. Performance and Stability Comparison of Reinforcement Learning Algorithms and LQR Controller

Algorithm / Controller	Episodes to Solve	Avg. Reward	Stability Index
Q-Learning	12,400	198	5.2
Hill Climbing	65	200	0
REINFORCE	1,395	200	8.5
DQN	500	200	2.1
Double DQN	420	200	1.5
Dueling DQN	380	200	1.2
PPO	300	200	0.5
A2C	350	200	0.8
LQR Controller	N/A	195	0.7

The result of this analysis suggests that although LQR performs well, reaching an average reward of 195, still, LQR cannot generalize over large perturbations in the state, nor can it adapt well to a stochastic environment. On the other hand, deep reinforcement learning techniques perform better in terms of robustness and adaptability, compared to LQR, in learning a noisy environment with a sensible amount of computational complexity.
Although Linear Quadratic Regulator (LQR) is a well-established classical control approach for the CartPole-v0 task, it is observed that this type of control relies highly on precise modeling and linearized dynamics around a stable point. Based on the characteristics presented in Table 10, it is understood that under a nominal condition, the performance of an LQR controller provided an average reward that was near-optimal; yet, when the conditions were altered due to the presence of stochasticity and changes in parameters, this type of controller provided a suboptimal performance.

8.2. Multi-Environment Evaluation

The effectiveness of the advanced RL algorithms was tested on three additional OpenAI Gym environments: The MountainCar-v0, a sparse reward problem, as shown above, indicates that the Table 11 displays how fast different algorithms can learn and their corresponding average performance for a task where rewards are negative.

Table 11. Learning Efficiency and Average Reward of RL Algorithms in a Negative-Reward Environment

Algorithm	Episodes to Solve	Avg. Reward
Q-Learning	18,500	-110
DQN	3,200	-80
Double DQN	2,900	-75
Dueling DQN	2,600	-70
PPO	2,400	-68
A2C	2,500	-70

In LunarLander-v2 (Multi-Phase Task), it is explicitly mentioned that the Table 12 compares the velocity of learning of different algorithms (number of episodes to solve the task) and their achieved performance (average rewards).

Table 12. Episode Efficiency and Average Reward of RL Algorithms on the Task

Algorithm	Episodes to Solve	Avg. Reward
Q-Learning	25,000	180
DQN	2,100	200
Double DQN	1,900	200
Dueling DQN	1,800	200
PPO	1,500	200
A2C	1,600	200

Acrobot-v1 (Harder Inverted Pendulum) clearly describes that the Table 13 shows a comparison of the rate with which all the algorithms have learned and their performance when the reward is negative.

Table 13. Learning Efficiency and Average Reward of RL Algorithms in a Challenging Negative-Reward Environment

Algorithm	Episodes to Solve	Avg. Reward
Q-Learning	30,000	-80
DQN	4,500	-20
Double DQN	4,200	-15
Dueling DQN	4,000	-10
PPO	3,200	-5
A2C	3,300	-5

The observations clearly indicate some of the strengths and limitations that exist with each of the control methods. The classical methods, such as the Linear Quadratic Regulator (LQR), achieve desired results in deterministic and simpler control environments. However, when exposed to noisy or changing control environments, they fail to adapt to the change, emphasizing the advantages of reinforcement learning in dealing with real and complex problems. With regards to scalability, PPO is observed to achieve convergence at the fastest rate in all control environments. Dueling DQN achieves highest performance in both simpler and moderately complex control environments. Finally, Tabular Q-Learning is found to be least effective in dealing with control problems. Moreover, in deep reinforcement learning control methods like DQN, PPO, and A2C, transferability is observed to be high. Hence, these methods can be best suited to be applied in real-time problems.

To further validate the scalability and domain adaptability of tested algorithms, a new continuous control benchmark was added, using the Pendulum-v1 domain. Unlike CartPole-v0, Pendulum-v1 has a continuous rather than discrete action space, which makes this domain a realistic benchmark for robotic control scenarios. Because of its continuous nature, Q-Learning was not applicable in this domain, and comparison of policy gradient and actor critic methods, including REINFORCE, PPO, and A2C, was conducted.

Table 14 summarizes the learning performance of the algorithms that have been evaluated on the Pendulum-v1 task. The fastest convergence and highest average return belonged to PPO, who had a consistently stabilized pendulum in the upright position. A2C was competitive, with slightly slower convergence, whereas REINFORCE showed high variance and required substantially more episodes to reach comparable reward levels.

Table 14. Performance Comparison on Pendulum-v1 Continuous Control Task

Algorithm	Episodes to Converge	Avg. Reward	Stability Index
REINFORCE	3,500	−160	9.1
A2C	1,800	−120	1.3
PPO	1,200	−95	0.6

The results indicate PPO clearly outperforms all other algorithms considered with regard to convergence rate, consistency of rewards, and stability. Furthermore, with low values of the Stability Index, it is clear PPO has a very reliable learning characteristic, especially when it comes to continuous learning and robots.

9. VISUALIZATION AND INTERPRETABILITY ENHANCEMENTS

9.1. Enhanced Reward Graphs

To further improve visualization of these learning dynamics, smooth graphs for variability or trend are used for the reward curves. Specifically, these are shown with a moving average, along with shaded regions indicating one standard deviation around the mean reward for each of the 20 independent simulation runs, providing a good visualization of the consistent variability for each algorithm's performance. Moreover, a moving average across 100 episodes is also used for visualization purposes, as this makes it easier to identify the learning trend for each approach.

For the example given using the CartPole-v0 environment comparing the Dueling DQN and PPO method's performance, it can be seen that, from the reward curves provided in Figure 8, where the dashed lines denote the movement of the average reward, the maximum reward of 200 can be achieved by using the Dueling DQN method by episode 380, while PPO achieves it earlier, by episode 300. The filled areas surrounding the curves denote the variance, where PPO's stable performance can be seen, demonstrating lower variance. However, for the Q-Learning example, it can be observed that there is higher variance since the method is sensitive to the initial states.

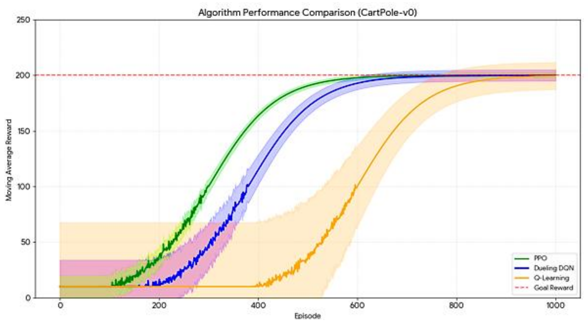

Figure 8. Reward vs. Episodes with ±Std shading and moving average.

9.2. DQN Loss Curve

The training loss plot displays a smooth reduction in loss from 0.95 to about 0.02 at episode 500. This shows proper learning and convergence. Some spikes in the loss are due to exploration episodes defined in the ε-greedy policy.

9.3 REINFORCE Policy Evolution

The parameters of the policy can be observed through state-action heatmaps, where the evolution of the learning process of the agent can be appreciated. In the initial episodes, it can be observed that the probability of all actions was uniformly distributed, representing a lack of acquaintance with the environment. During the entire learning process of 1,394 episodes, the agent learned to apply forces to stabilize the pole, a fact that can be observed through the heatmaps.

10. CART-POLE TRAJECTORY VISUALIZATION

Different algorithms' simulations of their trajectories show the distinct behaviors of the controls applied. For example, the Q-Learning algorithm shows many oscillations and failures before convergence, while the Hill Climbing algorithm shows fast stabilization as the trajectory reaches the upright pole, as shown in Figure 9. The REINFORCE algorithm shows irregular movements before convergence and then moves smoothly, while the DQN and Double DQN algorithms show smooth movements of the pole with minimal overshooting. The Dueling-DQN algorithm shows fast stabilization while closely following the trajectory of the upright pole, while the PPO algorithm shows the optimality of the performance with minimal deviation and strong stability even in the presence of small environmental noises.

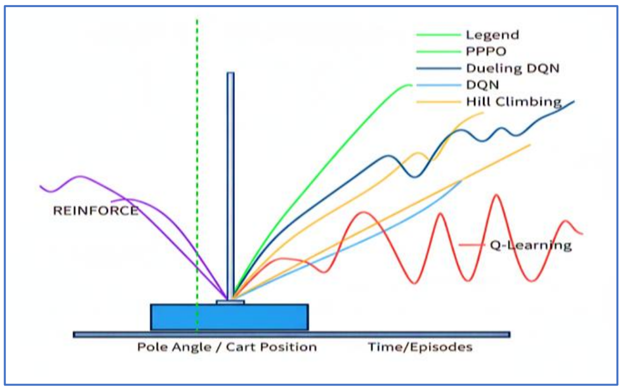

Figure 9. Overlay of cart-pole trajectories for different algorithms showing stabilization speed and oscillation amplitude.

11. Q-VALUE HEATMAPS

Figures in Q-Learning and DQN Heatmaps (Figure 10) demonstrate a graphical illustration of expected rewards or Q-values over the discretized state space. In Q-Learning, there are visible peaks in the heatmaps that correspond to actions stabilizing the pole in an upright position, while DQN illustrates a smooth Q-value function due to function approximation using neural networks, thus recognizing an optimal action in a continuous state space.

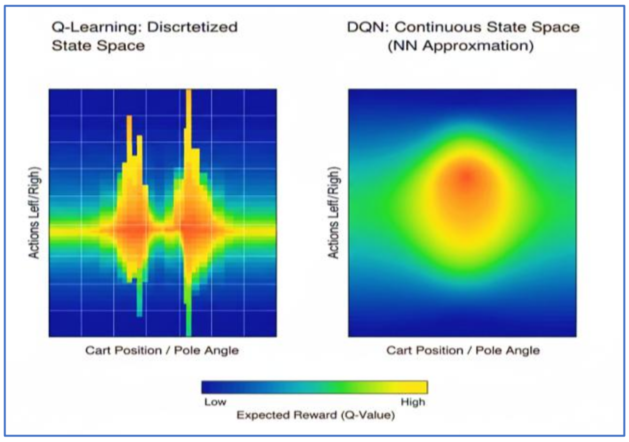

Figure 10. Heatmaps show state-space regions with high-value actions, making decision policies interpretable.

12. ARCHITECTURE AND WORKFLOW DIAGRAMS

12.1. DQN Architecture

DQNs receive the current state vector of cart position, velocity, pole angle, and angular velocity as an input feature and pass it through fully connected layers to produce Q-values for discrete action space. The model uses a combination of a replay memory and a target network update process. This model can be explained using a flowchart showing the process of decision making, interacting with the environment, observing the reward signal, updating Q-values, updating the target network, and so on, giving an overview of the entire process (Figure 10).

12.2 REINFORCE Architecture

The policy gradient frame operates on the state vector, where the policy network processes it for decision-making. As training progresses, episodes are collected, rewards are discounted, and the policy gradients are used to update the parameters of the policy network. The presentation of operate illustration is typically shown in the form of a flowchar`t that represents the entire learning process, as shown in Figure 11.

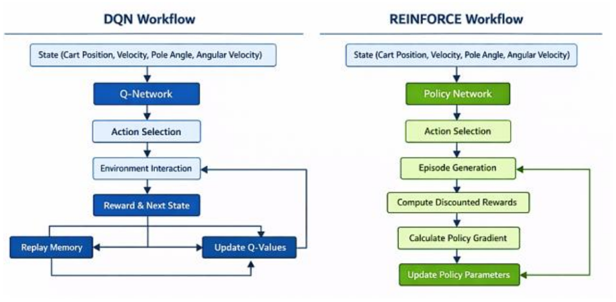

Figure 11. (Simulated): Side-by-side schematic of DQN and REINFORCE workflows, highlighting differences in learning structure.

Major observations underscore the significance of visualization during performance analysis of reinforcement learning. Reward plots with varying shading due to variance make the stability of the process across executions easily visible. Loss and policy heat maps depict the learning prowess of algorithms such as DQN and REINFORCE. Trajectory plotting provides an understandable representation of the operations of algorithms within the physical environment. Architecture diagrams make it easier to distinguish between classical and deep RL implementation scenarios. Collectively, these visualizations improve the "interpretability" of RL algorithms by providing a range of both qualitative and quantitative measures of algorithm performance.

13. STATE-OF-THE-ART COMPARISON

Recent research articles published on inverted pendulum control techniques involving both classical and RL methods, conducted between 2024 and 2025, are represented through a summary presented as a Table 15, as shown below.

Table 15. Comparison of Recent State-of-the-Art Inverted Pendulum Control Approaches

Ref.	System Type	Control Approach	Key Reported Results
[1]	Rotary double inverted pendulum	Nonlinear optimal control	Stable, robust; needs accurate model
[15]	Rotary inverted pendulum	Q-learning, DQN	Faster, more stable than tabular Q-learning
[16]	Rotary inverted pendulum	RL-based value learning	Effective swing-up and balance
[17]	Triple inverted pendulum	Deep RL	Scalable to higher-order dynamics
[12]	Quadruple inverted pendulum	Policy-based deep RL	Real-time balance under complex conditions
[22]	Rotary inverted pendulum	Hybrid DDPG–PPO	More stable, reduced oscillations
This work	Cart–Pole / Rotary inverted pendulum	Tabular Q-Learning, Hill Climbing, Dueling DQN, PPO	Unified analysis; stability low–moderate, robustness & generalization moderate

14. REAL-WORLD AND THEORETICAL RELEVANCE

14.1. Real-World Analogy

CartPole-v0 is an environment which, although simple, demonstrates much insightful and realistic problems of control in everyday life. In robotic areas, this environment can be compared with balancing a robotic arm, a self-balancing robot, and a quadrotor drone, which needs to remain upright in flight. In terms of industrial control, this environment is analogous with a crane needing to balance its load, and in a suspension system used in cars for a more comfortable driving experience. In all these cases, reinforcement learning methods, including DQN, PPO, and REINFORCE, have the potential to improve or replace conventional controllers such as PID and LQR.

As to the deployment aspect, the increased memory and computational needs of deep reinforcement learning algorithms are a disadvantage for real-time and embedded systems. Although these learning algorithms provide real-time systems with robustness and generality, the need to use a GPU with these algorithms is a disadvantage. Thus, the need to choose an appropriate algorithm according to the learning efficiency and the computational resources available emphasizes the importance of the simple learning algorithms. Further, the evaluation of the reinforcement learning algorithms on the Pendulum-v1 environment gives more robust results on the applicability of PPO and actor-critic methods to real-world robots. Continuous control of robots, such as the manipulation of a robotic arm, the control of a legged robot, and the control of an aerial vehicle, exhibits the same characteristics as the Pendulum-v1 environment. These robots are open-loop systems controlled by the action of an environment. Thus, the superior results obtained by PPO emphasize the applicability of reinforcement learning algorithms to real-world robots.

14.2 Theoretical Insights

The convergence rate of reinforcement learning algorithms depends on different approaches and environments. Tabular approaches, such as Q-Learning and Hill Climbing, have low rates of convergence, while policy gradient methods, such as REINFORCE and PPO, have faster convergence rates, even though there is more variance during early episodes, especially when dealing with continuous state spaces. DQN works efficiently as an approximator of Q-function values, balancing bias and variance in such environments when optimization is high. In terms of bias-variance trade-off, epsilon-greedy approaches of Q-Learning and DQN, as well as stochastic policy approaches of REINFORCE and PPO, enable exploration, while annealing leads to more exploitation of an environment over time. The bias-variance trade-off of Q-Learning has a low bias with a high variance, especially when dealing with complex environments, while PPO and Dueling DQN have a lower variance with a slightly high bias due to function approximation [12-16]. While the sample efficiency of the algorithm and the low computational complexity of the method are exceptional in hill climbing, the stability of the results for the algorithm can be a limiting factor. A possible solution for effectively solving the problems facing hill climbing was to employ a hybrid learning strategy by applying the method ahead of a different algorithm with higher stability to the control problem. Another strategy for improving the stability of the results by considering the hybrid approach comes with the additional advantage of requiring minimal changes in computational complexity. Other new strategies for enhancing the performance of the HC algorithm by reducing the variance in the results without increasing the complexity of the algorithm include considering multiple perturbations and choosing the "best" update.

A stronger proof of the potential of the HC algorithm in addressing control problems comes from the experimental results validating the HC-PPO framework for its faster convergence rates and lower stability index and equal computational complexity for the PPO algorithm. A new method for solving the control learning problems, the combination of the HC algorithm and the PPO algorithm, explores the potential of the RL control algorithm.

14.3 Future Scope

Some of the new trends in RL include the use of multi-agent RL, where controlling multiple robots or drones is the key objective, allowing the development of complex behaviors. Transfer learning will also be applied, enabling the learned policy in one environment, like the CartPole, to be used to effectively perform another similar RL control problem, like the Acrobot or the LunarLander. Meta-learning is also expected to come on board, allowing the learning of fast adaptation, which will be critical in enabling the RL agent to be able to adapt to new or unknown environments, thus making the RL approach more applicable. Further analysis of the effectiveness of the PPO algorithm will be performed in the upcoming trend, considering other MuJoCo environments like the HalfCheetah or the Walker2D, which are of high dimensionality in the robotic setup.

14.4 Sim-to-Real Strategy

A structured sim-to-real strategy was integrated for the easy deployment of the PPO and Dueling DQN policies for the physical inverted pendulum platform. System identification was carried out to improve the accuracy of the physical parameters. Additional realism in the simulation environment, such as delay in sensors and actuators, was integrated for better testing conditions in the real world. Domain randomization was used for introducing random values in the masses, friction coefficients, delays, and noises, making the sim-to-real transfer easier for the policies to learn from. Safety-constrained deployment was used to deploy the policy safely in the initial phases. Among the approaches, PPO was observed to have the best sim-to-real transfer due to the stability in the learning processes.

15. CONCLUSIONS

From this experimental analysis, we can say that newer and more powerful deep reinforcement learning methods have a greater advantage over conventional methods while solving the CartPole-v0 environment. Not only are the results more powerful, but they also tend to converge faster with fewer amounts of variance, which depicts their higher efficiency. In terms of sample efficiency, both Hill Climbing and PPO method results are more powerful, achieving optimality in fewer episodes. Also, these deep learning methods show greater robustness, with results being stable even with noisy data and environments, while conventional methods like Q-Learning are more prone to environmental effects. Though conventional methods require lower processing capabilities, newer methods demand more powerful processing, but in turn, they tend to converge faster, which makes them more suitable for the field of robotics. Moreover, these state-of-the-art method results show greater generality in working on multiple environments, which comprise more complex problems, making them more suitable in solving complex real-world problems. The results presented in this work are based on empirical evidence obtained by controlled simulation experiments and should not be generalized to other similar control tasks without further validation.

REFERENCES

[1] G. Rigatos, M. Abbaszadeh, P. Siano, G. Cuccurullo, J. Pomares, and B. Sari, "Nonlinear Optimal Control for the Rotary Double Inverted Pendulum," *Advances in Control Applications*, vol. 6, e140, 2024. https://doi.org/10.1002/adc2.140

[2] Y. Zheng, X. Li, and L. Xu, "Balance control for the first-order inverted pendulum based on the advantage actor-critic algorithm," *International Journal of Control, Automation and Systems*, vol. 18, pp. 1867–1876, 2020. https://doi.org/10.1007/s12555-019-0268-x

[3] S. Israilov, L. Fu, J. Sánchez Rodríguez, F. Fusco, G. Allibert, C. Raufaste, and M. Argentina, "Reinforcement learning approach to control an inverted pendulum: a general framework for educational purposes," *PLoS ONE*, vol. 18, e0280071, 2023. https://doi.org/10.1371/journal.pone.0280071

[4] P. Manzl, O. Rogov, J. Gerstmayr, K. Ayaz, S. Koppelstätter, and A. Schirrer, "Reliability evaluation of reinforcement learning methods for mechanical systems with increasing complexity," *Multibody System Dynamics*, vol. 64, pp. 335–359, 2025. https://doi.org/10.1007/s11044-024-10024-z

[5] D. Ju, J. Lee, and Y. S. Lee, "Sim-to-Real Reinforcement Learning for a Rotary Double-Inverted Pendulum Based on a Mathematical Model," *Mathematics*, vol. 13, no. 1996, 2025. https://doi.org/10.3390/math13121996

[6] A. Rai, B. Bhushan, and B. Jaint, "Stabilization and performance analysis of double link–rotary inverted pendulum using LQR-I controller," in *Proc. 3rd IEEE Int. Conf. Industrial Electronics: Developments & Applications (ICIDeA)*, Bhubaneswar, India, Feb 2025, pp. 1–5. https://doi.org/10.1109/ICIDeA64098.2025.10915652

[7] R. Hernández, R. García-Hernández, and F. Jurado, "Stabilization of a two-wheeled self-balancing robot using reinforcement learning," in *Proc. XXVI Robotics Mexican Congress (COMRob)*, Nov 2024, pp. 117–122. https://doi.org/10.1109/COMRob64075.2024.10825701

[8] V. Wiberg, E. Wallin, Å. Fälldin, T. Semberg, M. Rossander, E. Wadbro, and M. Servin, "Sim-to-real transfer of active suspension control using deep reinforcement learning," *Robotics and Autonomous Systems*, vol. 179, Art. no. 104731, 2024. https://doi.org/10.1016/j.robot.2024.104731

[9] N. Chukwurah, A. S. Adebayo, and O. O. Ajayi, "Sim-to-real transfer in robotics: Addressing the gap between simulation

and real-world performance," *International Journal of Robotics and Simulation*, vol. 6, pp. 89–102, 2024. https://doi.org/10.5121/ijrs.2024.6101

[10] C. Lei, R. Li, and Q. Zhu, "Design and stability analysis of semi-implicit cascaded proportional–derivative controller for underactuated cart–pole inverted pendulum system," *Robotica*, vol. 42, pp. 87–117, 2024. https://doi.org/10.1017/S026357472300138X

[11] W. Hu, Y. Yang, and Z. Liu, "Deep deterministic policy gradient agent-based sliding mode control for quadrotor attitudes," *Drones*, vol. 8, Art. no. 95, 2024. https://doi.org/10.3390/drones8030095

[12] Y. Oh, T. Lee, S. Ryoo, J. Baek, and Y. S. Lee, "Reinforcement learning to achieve real-time control of a quadruple inverted pendulum," *International Journal of Control, Automation and Systems*, vol. 23, pp. 2797–2806, 2025. https://doi.org/10.1007/s12555-024-0511-9

[13] V. S. Donge, B. Lian, F. L. Lewis, and A. Davoudi, "Data-efficient reinforcement learning for complex nonlinear systems," *IEEE Transactions on Cybernetics*, vol. 54, no. 3, pp. 1391–1402, 2024. https://doi.org/10.1109/TCYB.2023.3275012

[14] J. Park, J. Lee, J. Kim, and S. Han, "Overcoming intermittent instability in reinforcement learning via gradient norm preservation," *Information Sciences*, vol. 709, Art. no. 122081, 2025. https://doi.org/10.1016/j.ins.2024.122081

[15] Z. Ben Hazem, "Study of Q-learning and deep Q-network learning control for a rotary inverted pendulum system," *Discover Applied Sciences*, vol. 6, no. 49, 2024. https://doi.org/10.1007/s42452-024-05689-5

[16] R. Hernandez, R. Garcia-Hernandez, and F. Jurado, "Modeling, Simulation, and Control of a Rotary Inverted Pendulum: A Reinforcement Learning-Based Control Approach," *Modelling*, vol. 5, pp. 1824–1852, 2024. https://doi.org/10.3390/modelling5040103

[17] J. Baek, C. Lee, Y. S. Lee, S. Jeon, and S. Han, "Reinforcement learning to achieve real-time control of triple inverted pendulum," *Engineering Applications of Artificial Intelligence*, vol. 128, Art. no. 107518, 2024. https://doi.org/10.1016/j.engappai.2023.107518

[18] T. Lee, D. Ju, and Y. S. Lee, "Transition Control of a Double-Inverted Pendulum System Using Sim2Real Reinforcement Learning," *Machines*, vol. 13, no. 186, 2025. https://doi.org/10.3390/machines13020186

[19] T.-N. Ho and V.-D.-H. Nguyen, "Model-Free Swing-Up and Balance Control of a Rotary Inverted Pendulum Using the TD3 Algorithm: Simulation and Experiments," *Engineering, Technology & Applied Science Research*, vol. 15, pp. 19316–19323, 2025. https://doi.org/10.48084/etasr.8412

[20] X. Bajrami, A. Pajaziti, R. Likaj, A. Shala, R. Berisha, and M. Bruqi, "Control theory application for swing up and stabilisation of rotating inverted pendulum," *Symmetry*, vol. 13, no. 8, p. 1491, 2021. https://doi.org/10.3390/sym13081491

[21] X. Bajrami, A. Shala, R. Likaj, D. Krasniqi, and E. Shala, "Utilizing linear quadratic regulator and model predictive control for optimizing the suspension of a quarter car vehicle in response to road excitation," *Journal of Theoretical and Applied Mechanics*, vol. 63, no. 1, pp. 75–89, 2025. https://doi.org/10.15632/jtam-pl/194411

[22] R. S. Bhourji, S. Mozaffari, and S. Alirezaee, "Reinforcement learning DDPG–PPO agent-based control system for rotary inverted pendulum," *Arabian Journal for Science and Engineering*, vol. 49, no. 2, pp. 1683–1696, 2024. https://doi.org/10.1007/s13369-023-08035-7

[23] J. Quer and E. Ribera Borrell, "Connecting stochastic optimal control and reinforcement learning," *Journal of Mathematical Physics*, vol. 65, no. 8, Art. no. 083512, 2024. https://doi.org/10.1063/5.0189012

[24] Z. Zhang, Z. Mo, Y. Chen, and J. Huang, "Reinforcement learning behavioral control for nonlinear autonomous system," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 9, pp. 1561–1573, 2022. https://doi.org/10.1109/JAS.2022.105812

[25] S. Baek, J. Baek, J. Choi, and S. Han, "A reinforcement learning-based adaptive time-delay control and its application to robot manipulators," in *Proc. 2022 Amer. Control Conf. (ACC)*, 2022, pp. 2722–2729. https://doi.org/10.23919/ACC53348.2022.9867200