Comparative Analysis of Value-Based and Policy-Gradient Reinforcement Learning Agents on the CartPole Environment

Devansh Lodha
Computer Science and Engineering
IIT Gandhinagar
Gandhinagar, India
devansh.lodha@iitgn.ac.in
Roll No: 23110091

Laksh Jain
Artificial Intelligence
IIT Gandhinagar
Gandhinagar, India
laksh.jain@iitgn.ac.in
Roll No: 23110185

Surriya Gokul
Computer Science and Engineering
IIT Gandhinagar
Gandhinagar, India
surriya.gokul@iitgn.ac.in
Roll No: 23110324

***Abstract*—Reinforcement Learning (RL) provides a robust framework for solving sequential decision-making problems. In this work, we present a comparative analysis of six distinct RL algorithms applied to the classic control problem of CartPole-v1. We categorize these algorithms into two families: Value-Based methods (DQN, Double DQN, SARSA) and Policy-Based methods (REINFORCE, REINFORCE with Baseline, Actor-Critic). We investigate the trade-offs between on-policy and off-policy learning, as well as the impact of variance reduction techniques. Our empirical results demonstrate that while simple policy gradient methods suffer from high variance, the addition of a baseline and the Actor-Critic architecture significantly improves stability. Similarly, within value-based methods, we analyze the effectiveness of Double Q-learning in mitigating overestimation bias compared to standard DQN. The Actor-Critic agent demonstrated the most consistent convergence, balancing the sample efficiency of value methods with the stability of policy optimization.**

Code Repository: **https://github.com/Jain-Laksh/Multiagent-Systems-Project**

***Index Terms*—Reinforcement Learning, DQN, SARSA, Policy Gradient, Actor-Critic, CartPole**

I. Introduction

Reinforcement Learning (RL) is a paradigm of machine learning where an agent learns to make sequences of decisions by interacting with an environment to maximize a cumulative reward signal [1]. Unlike supervised learning, where the agent is provided with correct labels, an RL agent must learn through trial and error, balancing the exploration of new strategies with the exploitation of known rewards.

This project focuses on the classic control problem, *CartPole-v1* [2], a standard benchmark in the Gymnasium environment. The objective is to balance a pole attached to a moving cart by applying forces to the left or right. Despite its apparent simplicity, CartPole serves as an excellent testbed for analyzing the fundamental properties of RL algorithms, particularly convergence speed and stability.

We implement and compare two major families of algorithms:

1) **Value-Based Methods:** These methods estimate the value of being in a state (or taking an action) and derive a policy from these estimates. We examine the Deep Q-Network (DQN) [3], its extension Double DQN [4], and the on-policy SARSA algorithm.
2) **Policy-Based Methods:** These methods directly parameterize and optimize the policy without consulting a value function for action selection. We examine the REINFORCE algorithm [5], REINFORCE with Baseline, and the hybrid Actor-Critic architecture.

Our analysis highlights the "Four Pillars of RL": Optimization, Delayed Consequences, Exploration, and Generalization. We specifically investigate the stability of learning curves and the effectiveness of variance reduction techniques in policy gradients versus the bias introduced by bootstrapping in value-based methods.

II. Theoretical Background

The problem is formalized as a Markov Decision Process (MDP), defined by the tuple (S, A, P, R, γ), where S is the state space, A is the action space, $P(s'|s, a)$ is the transition probability, $R(s, a)$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor.

A. Value-Based Learning

Value-based methods aim to learn the Action-Value function $Q^\pi(s, a)$, which represents the expected discounted return starting from state s, taking action a, and following policy π:

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right] \quad (1)$$

1) Q-Learning and DQN: Q-Learning is an off-policy algorithm that seeks to satisfy the Bellman Optimality Equation. It approximates the optimal value function $Q^*(s, a)$ utilizing a target based on the maximum possible future value:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \delta_t \quad (2)$$

where $\delta_t = R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t)$ is the TD error. Deep Q-Networks (DQN) stabilize this process using Experience Replay to break temporal correlations and a fixed Target Network to prevent the moving target problem [3].

2) Double DQN: Standard DQN is known to overestimate action values because the maximization step ($\max_{a'}$) in the target tends to select overestimated values. Double DQN [4] decouples selection from evaluation:

$$Y_t^{DDQN} = R_{t+1} + \gamma Q_{target}(S_{t+1}, \arg\max_a Q_{online}(S_{t+1}, a)) \tag{3}$$

3) SARSA: SARSA (State-Action-Reward-State-Action) is the on-policy counterpart to Q-learning. Instead of assuming the optimal next action, it updates based on the actual next action A_{t+1} chosen by the current behavior policy (e.g., ϵ-greedy):

$$\delta_t = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \tag{4}$$

This makes SARSA "safer" during training as it accounts for the agent's exploration.

B. Policy-Based Learning

Policy methods parameterize the policy $\pi_\theta(a|s)$ directly and optimize parameters θ via gradient ascent on the objective $J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[G_0]$.

1) Policy Gradient Theorem: The gradient of the objective can be estimated without a model of the environment dynamics [1]:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} [\nabla_\theta \log \pi_\theta(a_t|s_t) G_t] \tag{5}$$

2) REINFORCE and Baselines: The REINFORCE algorithm uses the Monte Carlo return G_t as an unbiased estimate of the value. However, this estimator has high variance. To reduce variance without introducing bias, a state-dependent baseline $b(s)$ is subtracted from the return:

$$\nabla_\theta J(\theta) \approx \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)(G_t - b(s_t)) \tag{6}$$

Commonly, the value function $V(s)$ serves as the baseline.

3) Actor-Critic: Actor-Critic methods combine the advantages of both approaches. The "Critic" learns a value function $V_w(s)$ to approximate the return (bootstrapping), reducing variance at the cost of some bias. The "Actor" updates the policy using the TD error from the Critic:

$$\delta_t = R_{t+1} + \gamma V_w(S_{t+1}) - V_w(S_t) \tag{7}$$

This δ_t serves as an estimate of the Advantage function $A(s, a)$.

III. Methodology

In this section, we outline the neural network architectures and specific algorithmic implementations used for both value-based and policy-based agents.

A. Network Architectures

To ensure a fair comparison while acknowledging the differing representational needs of each algorithm, we utilized Multi-Layer Perceptrons (MLPs) for all function approximators.

1) Value-Based Networks: For DQN, Double DQN, and SARSA, the Q-network consists of an MLP with two hidden layers.

- **Input Layer:** 4 units (State dimension).
- **Hidden Layers:** Two layers of 64 neurons each, using ReLU activation.
- **Output Layer:** 2 units (Action dimension), representing $Q(s, \text{left})$ and $Q(s, \text{right})$.

The relatively smaller network size (64 units) was found to be sufficient for the Q-function to converge without overfitting.

2) Policy-Based Networks: For REINFORCE and Actor-Critic, we observed that the policy requires a more complex representation to map states directly to action probabilities effectively.

- **Actor Network:** Input dimension 4, two hidden layers of 128 neurons (ReLU), and a Softmax output layer for action probabilities.
- **Critic Network (Baseline):** Input dimension 4, two hidden layers of 128 neurons (ReLU), and a linear output for state-value estimation $V(s)$.

We found that reducing the hidden size to 64 for policy methods resulted in poor convergence, indicating a need for higher capacity to capture the policy landscape.

B. Algorithmic Enhancements

To stabilize training, we incorporated standard deep RL techniques:

- **Experience Replay (DQN/DDQN):** We utilized a replay buffer of capacity 10,000. Transitions (s, a, r, s', d) are sampled uniformly (batch size 64) to break temporal correlations in the training data.
- **Soft Target Updates:** Instead of hard updates, the target network parameters θ^- track the online network θ using a Polyak averaging coefficient $\tau = 0.01$: $\theta^- \leftarrow \tau\theta + (1-\tau)\theta^-$.
- **Optimizer:** We employed the Adam optimizer for all networks.

IV. Experimental Setup

All experiments were conducted on the `CartPole-v1` environment from the Gymnasium library. The goal is to keep the pole upright for as long as possible, with a maximum episode duration of 500 steps. The episode terminates if the pole angle exceeds $\pm 12^\circ$ or the cart position exceeds ± 2.4 units.

A. Hyperparameter Configuration

Table I summarizes the final hyperparameters selected after tuning. A distinct difference in configuration was necessary for the two families of algorithms.

TABLE I
HYPERPARAMETER SETTINGS

Parameter	Value-Based	Policy-Based
Learning Rate (LR)	1×10^{-4}	Actor: 3×10^{-4} Critic: 1×10^{-3}
Hidden Units	64×64	128×128
Gamma (γ)	0.99	0.99
Batch Size	64	N/A (Monte Carlo)
Memory Size	10,000	N/A
Soft Update (τ)	0.01	N/A
Exploration (ϵ)	$1.0 \rightarrow 0.001$	Stochastic Policy

B. Stability and Catastrophic Forgetting

A critical observation during our experimentation was the sensitivity of policy-based methods to hyperparameter selection compared to value-based methods.

While DQN and SARSA showed robust convergence with standard hyperparameters, REINFORCE and Actor-Critic exhibited signs of *catastrophic forgetting*—a phenomenon where the agent's performance dramatically collapses after a period of high rewards. Specifically, with higher learning rates or smaller network capacities (64 units), policy agents would solve the environment (reaching 500 reward) around episode 200-300, only to degrade to near-random performance by episode 400.

To mitigate this, we:

1) Increased the network capacity to 128 hidden units.
2) Tuned the learning rates, specifically using a lower rate for the Actor (3×10^{-4}) compared to the Critic (1×10^{-3}) to ensure policy updates did not outpace value estimation accuracy.
3) Limited the training horizon for policy methods to the point of convergence to prevent late-stage divergence.

V. RESULTS AND DISCUSSION

In this section, we analyze the training performance of the agents. We focus on three key metrics: convergence speed (episodes to reach maximum reward), stability (variance in rolling average), and peak performance.

A. Value-Based Agent Analysis

Figure 4 presents the learning curves for the three value-based agents. We utilize the rolling average of rewards (window size 100) to filter out high-frequency noise and visualize the underlying trend.

1) **DQN (Fig. 1):** The standard DQN agent successfully learns to solve the task, reaching the maximum reward of 500. However, the learning curve exhibits noticeable fluctuations between episodes 200 and 600. This instability is characteristic of the maximization bias inherent in Q-learning, where the `max` operator overestimates action values, leading to "optimistic" but potentially suboptimal updates.
2) **Double DQN (Fig. 2):** The effectiveness of decoupling action selection from evaluation is evident. The learning curve is significantly smoother than that of DQN. By reducing overestimation bias, the agent adopts a more stable policy update trajectory, reaching consistent solution-level performance with fewer dips in performance.
3) **SARSA (Fig. 3):** As an on-policy algorithm, SARSA learns the value of the current exploratory policy (ϵ-greedy). Its convergence is robust but slightly slower to reach the theoretical maximum compared to Double DQN. This is expected behavior; SARSA pays a "safety cost" during training by accounting for the possibility of random exploration steps.

B. Policy-Based Agent Analysis

Figure 5 compares the policy-gradient approaches directly.

The impact of variance reduction is the dominant factor here:

- **REINFORCE:** The vanilla Monte Carlo approach (Red curve) is highly unstable. Without a baseline, the high variance in returns (G_t) causes the gradient updates to be noisy, requiring over 1000 episodes to show significant learning.
- **Actor-Critic:** The Actor-Critic agent (Blue curve) is the clear top performer. By using a Critic to bootstrap value estimates, it performs low-variance updates at every time step. It solves the environment faster than both REINFORCE and the value-based methods.

C. Overall Conclusion

While Double DQN offers the most stable performance among value-based methods, the **Actor-Critic** architecture proved to be the most efficient overall for CartPole-v1. It balances the stability of policy optimization with the sample efficiency of bootstrapping.

VI. CONCLUSION

In this work, we conducted a comprehensive comparative analysis of reinforcement learning algorithms on the CartPole-v1 control problem. We implemented and evaluated three value-based methods (DQN, Double DQN, SARSA) and three policy-based methods (REINFORCE, REINFORCE with Baseline, Actor-Critic).

Our findings align with fundamental RL theory:

- **Variance Reduction is Key:** In policy gradients, the transition from pure Monte Carlo (REINFORCE) to bootstrapping (Actor-Critic) is essential for practical convergence speeds.
- **Bias-Variance Trade-off:** While Actor-Critic introduces bias via bootstrapping, the reduction in variance allows for significantly faster learning than unbiased Monte Carlo methods.
- **Addressing Overestimation:** Double DQN successfully mitigated the stability issues of standard DQN, providing the most robust performance among value-based candidates.

For the specific task of CartPole, the **Actor-Critic** architecture proved to be the most effective, balancing the stability of

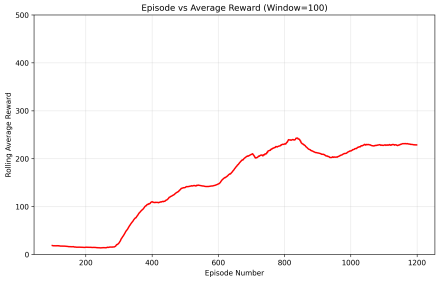

Fig. 1. DQN Learning Curve

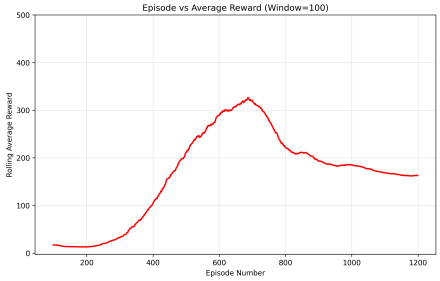

Fig. 2. Double DQN Learning Curve

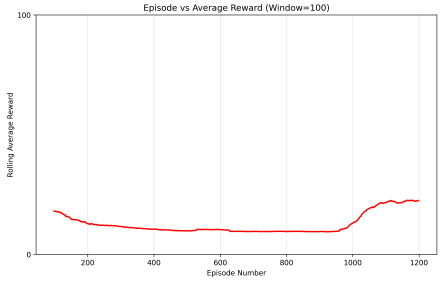

Fig. 3. SARSA Learning Curve

Fig. 4. Comparative analysis of Value-Based methods. Double DQN (Fig. 2) exhibits the smoothest ascent to the optimal reward, mitigating the instability seen in standard DQN (Fig. 1). SARSA (Fig. 3) converges but with a more conservative trajectory.

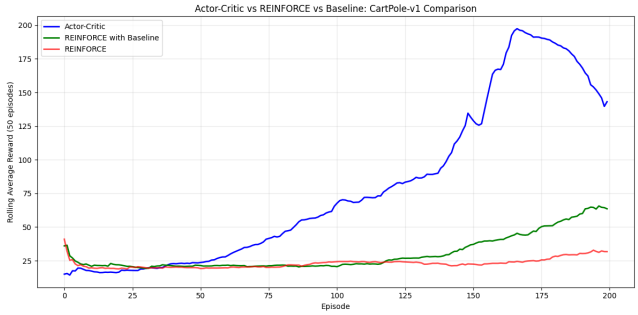

Fig. 5. Rolling Average Reward for Policy-Based Agents. The Actor-Critic architecture (Blue) demonstrates superior sample efficiency and stability compared to REINFORCE (Red) and REINFORCE with Baseline (Green).

policy search with the efficiency of value estimation. Future work could extend this analysis to continuous action spaces using algorithms like PPO or DDPG, where policy-based methods traditionally excel.

VII. Author Contributions

The work presented in this paper was divided among the authors as follows:

- **Laksh Jain:** Responsible for the **Value-Based** Reinforcement Learning implementations. This included the development of the DQN, Double DQN, and SARSA agents, as well as the tuning of the experience replay buffer and target network mechanisms on the CartPole environment.
- **Surriya Gokul:** Responsible for the **Policy-Based** Reinforcement Learning implementations. This involved developing the REINFORCE and Actor-Critic agents, investigating the impact of baselines on variance reduction, and tuning the differing network architectures required for policy gradients.
- **Devansh Lodha:** Responsible for compilation of this report. This involved structuring the comparative analysis and authoring the technical documentation in LaTeX.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE transactions on systems, man, and cybernetics*, no. 5, pp. 834–846, 1983.

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

[4] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[5] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.