Model-Free vs. Model-Based Reinforcement Learning in a Stochastic GridWorld

Felipe Campoverde

Virginia Tech

fcampoverdeg@vt.edu

CS 4824 — Machine Learning

Final Project Report

Fall 2025

Abstract

We investigate how reinforcement learning (RL) agents learn in a stochastic 11x11 GridWorld environment using three methods: Q-Learning, SARSA, and the model-based Dyna-Q algorithm. The study examines sample efficiency, robustness to stochastic transitions, planning depth sensitivity, and seed-dependent variability. Experiments include baseline learning curves, planning sweeps over the number of simulated updates K, wind-induced stochasticity, and a layout-shift robustness test. Results show that Dyna-Q significantly accelerates convergence through planning, while model-free methods demonstrate greater stability under environmental noise. Despite differences in learning speed, all algorithms achieve similar final greedy-policy returns. This work highlights the trade-offs between model-free stability and model-based speed, and provides a reproducible benchmark for analyzing classical RL algorithms under stochasticity.

1 Introduction

GridWorld environments provide an interpretable and computationally lightweight testbed for analyzing the behavior of reinforcement learning algorithms. In stochastic settings—where transitions can randomly deviate due to wind or noise—learning dynamics differ substantially across algorithms. Understanding these differences is important for diagnosing stability issues, assessing sample efficiency, and motivating more advanced model-based approaches.

This project focuses on comparing model-free temporal-difference methods (Q-Learning and SARSA) with the model-based Dyna-Q algorithm in a stochastic GridWorld. Prior studies typically analyze these methods qualitatively, but few perform a systematic comparison including: (1) planning-step sweeps, (2) seed sensitivity, (3) robustness to environmental noise, and (4) layout-shift adaptation.

Contributions. This work provides:

- A unified implementation of Q-Learning, SARSA, and Dyna-Q with standardized logging and evaluation.
- A planning-depth sweep ($K = \{0, 5, 10, 20, 50\}$) revealing how simulated updates affect convergence.
- A robustness analysis across wind disturbances and layout shifts.
- A seed-sensitivity study showing how randomness affects returns, steps, and Dyna-Q's internal model.

All code, plots, and configurations are publicly available in the project's GitHub repository: `https://github.com/fcampoverdeg/reinforcement_learning`.

2 Background and Related Work

Reinforcement learning formalizes sequential decision-making as a Markov Decision Process (MDP) consisting of states S, actions A, transitions P, and rewards R. Temporal-difference (TD) learning methods update value estimates using bootstrapped predictions.

2.1 Q-Learning

Q-Learning [1] is an off-policy TD control method. Its update rule is:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right].$$

By learning from the greedy target $\max_{a'} Q(s',a')$ regardless of the behavior policy, Q-Learning can learn an optimal policy while following an ϵ-greedy exploratory policy.

2.2 SARSA

SARSA [2] is an on-policy method that evaluates and improves the same behavior policy. Its update rule is:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma Q(s',a') - Q(s,a) \right],$$

where (s,a,r,s',a') is the transition experienced under the current policy. This tends to produce more conservative policies in stochastic environments, because exploratory actions are directly reflected in the updates.

2.3 Dyna-Q

Dyna-Q [3] combines direct learning and planning. After each real transition (s,a,r,s'), a learned model $\hat{P}, \hat{R}$ is updated and K simulated updates are performed from stored transitions.

Conceptually, Dyna-Q augments real samples with model-generated samples, thereby increasing the number of effective updates per environment interaction. Prior work shows Dyna-Q improves sample efficiency [4] but may be sensitive to model errors, especially under high stochasticity.

3 Methods

3.1 Environment

The experiments use a custom 11x11 stochastic GridWorld featuring:

- Walls (gray) that are impassable.
- Pits with terminal reward -1.
- An absorbing goal state with reward $+1$.
- A small step cost of -0.01 on non-terminal moves.
- Wind with probability 0.1, randomly pushing the agent off its intended path.

The environment's stochasticity influences exploration trajectories, learned models in Dyna-Q, and the consistency of episode lengths. This setup is designed to expose differences between on-policy, off-policy, and model-based methods under uncertainty.

3.2 Training Configuration

All algorithms use:

- ϵ-greedy exploration with exponential decay over episodes.
- Learning rate α and discount factor γ defined in a shared configuration structure (`TrainConfig`).
- Logging of episode returns, steps, and Q-table snapshots every fixed number of episodes.

Random seeds control:

- Transition outcomes (wind events and other stochastic transitions).
- Random actions during ϵ-greedy exploration and tie-breaking.
- The order of Dyna-Q's sampled transitions during planning.
- The trajectory and episode length indirectly via stochasticity.

3.3 Planning Sweep for Dyna-Q

For Dyna-Q, the number of planning steps K is varied to study the impact of model-based updates:

$$K \in \{0, 5, 10, 20, 50\}.$$

When $K = 0$, Dyna-Q reduces to a purely model-free algorithm with the same update as Q-Learning but still maintaining a (unused) model. For larger K, the algorithm performs more simulated updates per real step, effectively multiplying the rate at which Q-values are updated.

4 Experimental Design

4.1 Baseline Experiments

Each algorithm is trained for a fixed number of episodes (e.g., 700–1000), and the following metrics are recorded:

- Average return per episode.
- Number of steps until termination (goal or pit).
- Value heatmaps of the learned Q-table.
- Greedy policy evaluations over 30 rollouts for each snapshot.

These baselines illustrate the learning dynamics in the original GridWorld with moderate wind.

4.2 Robustness Tests

To probe robustness, two additional perturbations are introduced:

1. **Windy environment:** the probability and/or effect of wind is increased, making transitions more stochastic.
2. **Layout shift:** additional walls and pits are added to the GridWorld, altering shortest paths and introducing new traps.

For each perturbed environment, all three algorithms are retrained with the same hyperparameters (or initialized from the original policy, depending on the experiment) to measure adaptability.

4.3 Evaluation Protocol

Policies are evaluated over 30 episodes at selected checkpoints. This sample size provides a reasonable compromise: it reduces the variance of performance estimates while remaining computationally cheap. Although stochasticity and seed variance still introduce noise, the mean return and mean steps converge sufficiently to allow clear comparison across algorithms and planning depths.

5 Results

This section presents learning curves, robustness experiments, and seed-stability results for Q-Learning, SARSA, and Dyna-Q in the stochastic GridWorld and its variants.

5.1 Per-Algorithm Learning Dynamics

Q-Learning. Figure 1 shows the episodic return for Q-Learning, along with a rolling average. Despite significant noise early in training (due to off-policy updates under stochastic transitions), the agent gradually converges to a high-return policy. Episode length (Figure 2) decreases over time, indicating that the learned policy reaches the goal more quickly and avoids pits.

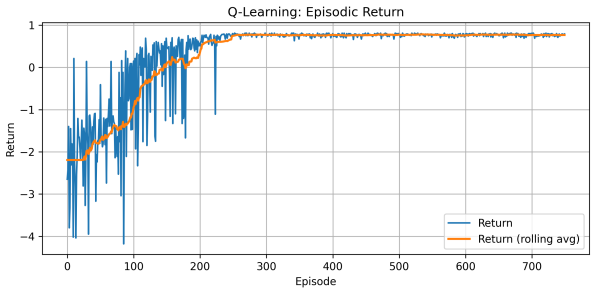

Figure 1: Q-Learning: episodic return in the baseline stochastic GridWorld (with rolling average).

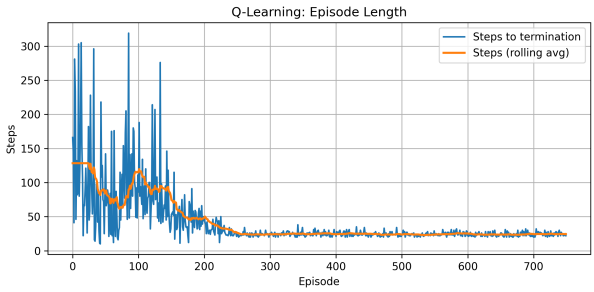

Figure 2: Q-Learning: episode length (steps to termination) in the baseline stochastic GridWorld.

SARSA. SARSA exhibits smoother learning dynamics (Figure 3), with smaller spikes in return and a more gradual reduction in episode length (Figure 4). This is consistent with on-policy learning, where exploratory actions are explicitly accounted for in the update, leading to more conservative, risk-aware policies in the presence of wind.

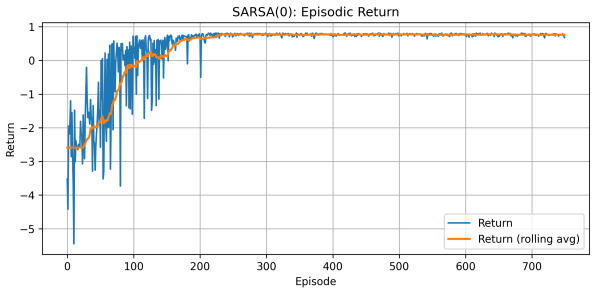

Figure 3: SARSA: episodic return in the baseline stochastic GridWorld.

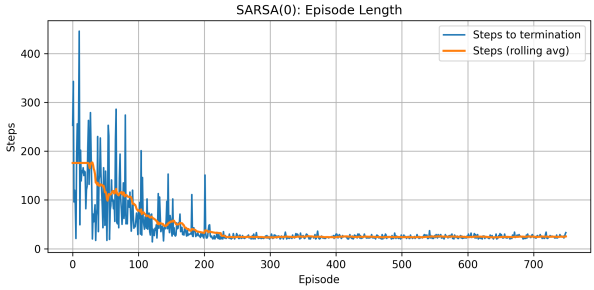

Figure 4: SARSA: episode length (steps to termination) in the baseline stochastic GridWorld.

Dyna-Q. Dyna-Q learns much faster initially (Figure 5), due to additional planning updates per real transition. Episode lengths also drop quickly (Figure 6). However, transient instability is visible: when the learned model does not fully capture the stochastic dynamics, simulated updates can temporarily push Q-values in suboptimal directions.

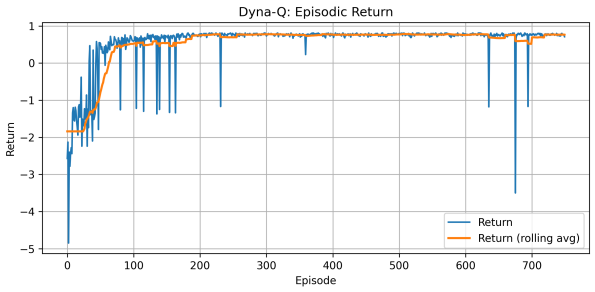

Figure 5: Dyna-Q: episodic return in the baseline stochastic GridWorld.

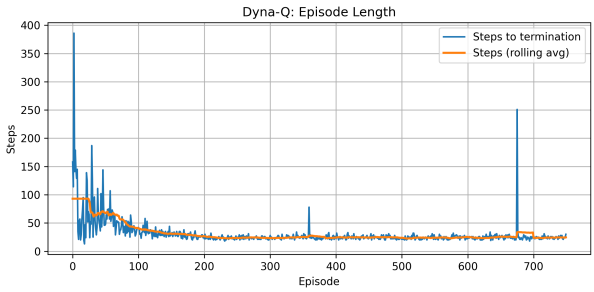

Figure 6: Dyna-Q: episode length (steps to termination) in the baseline stochastic GridWorld.

5.2 Planning Step Analysis

For Dyna-Q, we sweep the number of planning steps $K \in \{0, 5, 10, 20, 50\}$. Figure 7 shows that small to moderate values of K sharply accelerate convergence, whereas very large K yield diminishing returns and occasional instability. This supports the intuition that there is a practical range of planning depth where simulated updates are helpful without over-amplifying model errors.

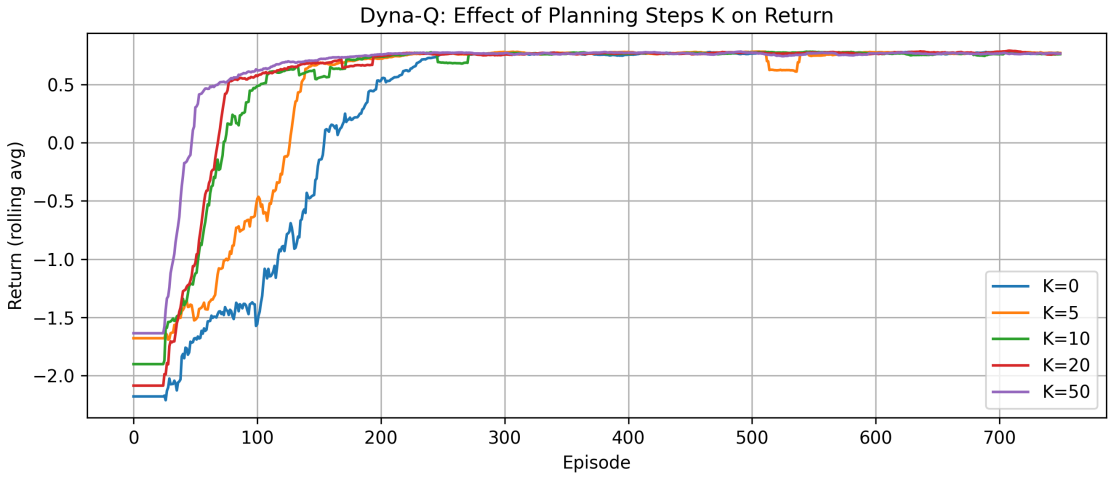

Figure 7: Dyna-Q planning sweep: effect of planning steps K on episodic return.

5.3 Robustness to Wind

To test robustness to increased transition noise, we train all algorithms on a "windy" version of the GridWorld, where the wind probability is higher. Figure 8 shows the return curves, and Figure 9 shows the episode lengths.

SARSA is the most stable in this setting, maintaining relatively smooth learning curves and avoiding catastrophic failures. Q-Learning exhibits larger variance, overestimating risky paths that are frequently disrupted by wind. Dyna-Q still converges quickly, but is more sensitive to model mismatch, as planning uses a model that cannot fully capture the heightened randomness.

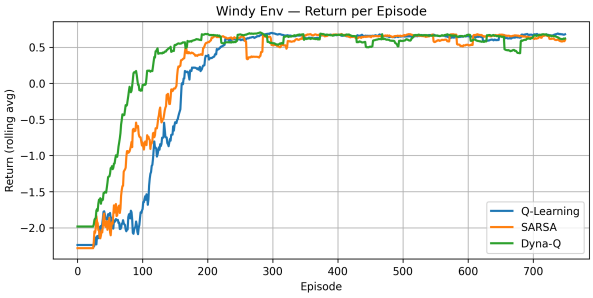

Figure 8: Windy environment: return per episode (rolling average) for all algorithms.

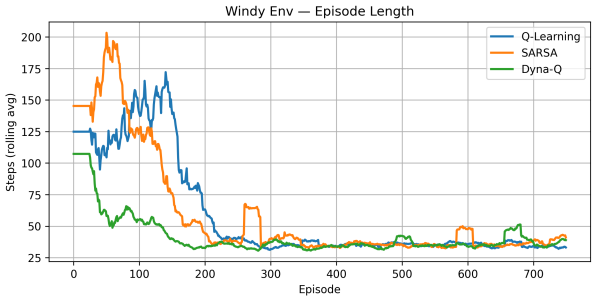

Figure 9: Windy environment: episode length (rolling average) for all algorithms.

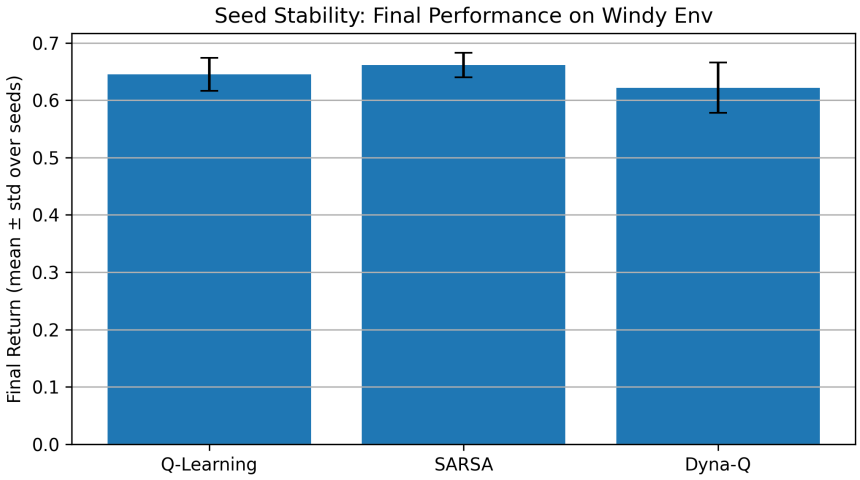

Figure 10: Seed stability on the windy environment: final return mean ± standard deviation across seeds for each algorithm.

5.4 Robustness to Layout Shift

We next modify the GridWorld layout by adding extra walls and pits, tightening corridors and introducing new local traps while preserving reachability of the goal. Zero-shot evaluation of baseline-trained policies reveals an immediate drop in performance, as expected, but all algorithms can adapt when retrained on the new layout.

Figure 11 and Figure 12 show learning curves when training directly on the layout-shifted environment. Dyna-Q again converges faster, while Q-Learning and SARSA adapt more slowly but steadily.

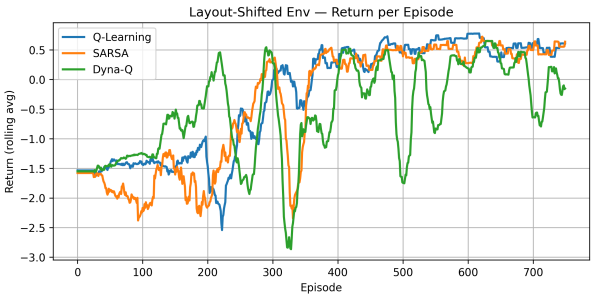

Figure 11: Layout-shifted environment: return per episode (rolling average) for all algorithms.

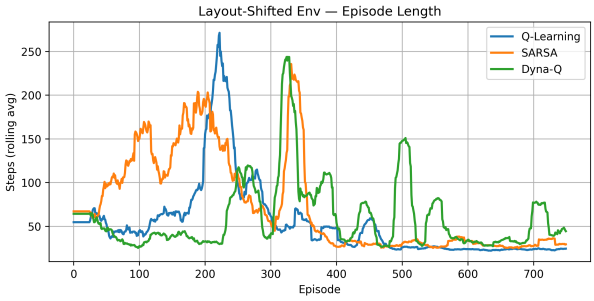

Figure 12: Layout-shifted environment: episode length (rolling average) for all algorithms.

5.5 Seed Stability under Layout Shift

Finally, we examine seed stability on the layout-shifted environment by training each algorithm across multiple random seeds and computing the mean and standard deviation of final returns over the last training episodes. Figure 13 summarizes the results.

Dyna-Q achieves the highest mean final return but also shows slightly higher variability across seeds, consistent with its sensitivity to early model errors. SARSA yields the most stable (lowest-

variance) performance, while Q-Learning lies in between, reflecting its off-policy nature.

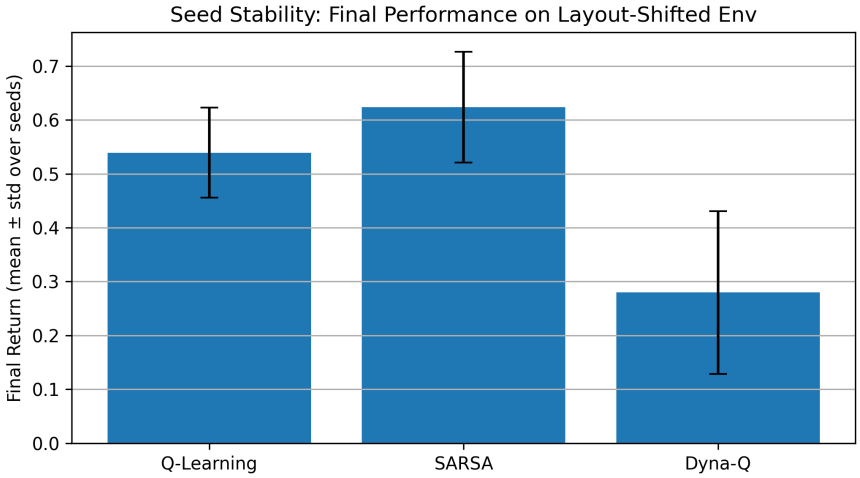

Figure 13: Seed stability on the layout-shifted environment: final return mean ± standard deviation across seeds for each algorithm.

6 Discussion

The results highlight fundamental trade-offs in RL under stochasticity:

- **Model-based gain:** Planning dramatically accelerates early learning by providing additional updates from a learned model.
- **Model-free robustness:** SARSA's on-policy nature provides more stability in noisy or risky environments, at the cost of slower convergence.
- **Model imperfection:** Dyna-Q's learned model can amplify early transition errors under high stochasticity, particularly when K is large.
- **Policy similarity:** Despite different convergence profiles, final greedy policies converge to similar quality in this tabular GridWorld.

Limitations of this work include the use of a single tabular environment, fixed hyperparameters, and a simple last-visit model for Dyna-Q. Additionally, the analysis does not consider function approximation or continuous state spaces, where model errors and instability can be more severe.

7 Conclusion and Future Work

This project provides a reproducible comparison of three classical RL algorithms under stochastic conditions. Planning improves sample efficiency but introduces sensitivity to model inaccuracies. Model-free algorithms remain more stable but require more episodes to converge.

Future work includes:

- Implementing prioritized sweeping and comparing it to vanilla Dyna-Q.
- Exploring Dyna-Q++ and uncertainty-aware planning mechanisms.
- Scaling to larger or continuous environments where function approximation is required.
- Adding real-time visualization tools for inspecting trajectories, value functions, and model errors during training.

These extensions would further clarify when and how model-based methods should be deployed in practice.

Acknowledgments

This project was completed for CS 4824: Machine Learning at Virginia Tech. I thank the course staff and classmates for feedback throughout the project.

References

[1] Christopher JCH Watkins. Learning from delayed rewards. 1989.

[2] Gavin A Rummery and Mahesan Niranjan. On-line q-learning using connectionist systems. Technical report, University of Cambridge, 1994.

[3] Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine Learning Proceedings 1990*, pages 216–224. Elsevier, 1990.

[4] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.