Final Project Report
CMPE 471 / CMPS 497: Reinforcement Learning

Group ID: 9
Hams Gelban (2104047)
Rouaa Naim (202106763)
Group of 2

May 10, 2025

1. Abstract

This project explores reinforcement learning (RL) methods for lane-keeping in a simulated highway environment. Using the `highway-env` environment, the objective is to train an agent that maintains its lane while avoiding collisions and optimizing driving efficiency. We will implement and compare both tabular RL methods (e.g., Q-learning with discretized states) and function approximation methods (e.g., linear approximators with tile coding). The agent's performance will be critically evaluated under varying exploration strategies (such as ε-greedy and softmax), learning rate schedules, and reward function designs.

We will assess each method's effectiveness in handling the environment's continuous state space and analyze the limitations and trade-offs between simplicity and generalization. Key metrics will include average cumulative reward, stability in lane, and frequency of collisions. The final report will detail these evaluations and provide visualizations of learning performance over time. This project aims to provide practical insights into the design of efficient RL agents for autonomous driving scenarios.

2. Literature Review

Lane keeping is a classical control problem often formulated as a Markov Decision Process (MDP), where an agent selects actions to maintain a desired trajectory. Reinforcement learning (RL) methods have been widely applied to such problems due to their ability to learn directly from interaction without a prior model of the environment Kaelbling et al. [1996].

Traditional Q-learning Watkins and Dayan [1992] has been successful in discrete environments; however, it struggles with continuous state spaces like driving scenarios. To address this, discretization techniques have been used to approximate continuous variables, but they can lead to coarse approximations or an explosion in the state space dimensionality Sutton and Barto [2018]. As an alternative, function approximation methods such as linear value

functions and tile coding provide a scalable solution. Tile coding, introduced by Sutton, enables generalization across continuous inputs by mapping states into overlapping binary feature vectors Sutton and Barto [2018].

Exploration strategies also play a crucial role in learning performance. The ϵ-greedy strategy, where actions are randomly selected with probability ϵ and the best-known action is selected otherwise, remains a standard approach Sutton and Barto [2018]. Another popular alternative is softmax (Boltzmann) exploration, where actions are chosen probabilistically based on their estimated values Kaelbling et al. [1996].

The `highway-env` environment Leurent [2020] provides a controlled highway driving simulation with continuous state spaces and discrete action spaces, making it an ideal platform to test both tabular and function approximation methods. Prior work such as Al-Sallab et al. Al-Sallab et al. [2016] applied deep RL approaches for lane keeping, but simpler linear methods with tile coding have also shown success in low-dimensional driving tasks, offering faster convergence and better interpretability when compared to deep networks for relatively simple environments.

In this project, we implement and critically compare tabular Q-learning with state discretization against linear function approximation with tile coding. We also explore the impacts of different exploration strategies and learning rate schedules, aiming to achieve stable and efficient lane-keeping behavior in the `highway-env` environment.

3. System Model and Problem Definition

3.1 Environment Description

We use the `highway-env` environment, which simulates a multi-lane highway driving scenario. The ego vehicle (the agent) is placed among multiple traffic participants and must maintain safe and efficient driving behavior. For our experiments, we specifically used the Kinematics observation configuration, which provides a continuous vector that includes the speed, lateral position, heading, and the positions and velocities of the ego vehicle, rather than the alternative representation of the occupancy grid. We selected the Kinematics configuration because it offers a lower-dimensional, more interpretable feature set, making it better suited for function approximation methods like tile coding and allowing a fairer comparison against tabular methods. The environment offers a discrete action space and a continuous state space, making it suitable for evaluating both tabular and function approximation reinforcement learning methods.

The default observation space is a continuous vector including the ego vehicle's speed, lateral position, heading, and the positions and velocities of nearby vehicles. The action space consists of five discrete actions: `IDLE`, `LEFT`, `RIGHT`, `FASTER`, and `SLOWER`, which correspond to lane changes and speed adjustments.

3.2 Agent Objective

The agent's goal is to:

- Maintain lane center as long as possible.

- Avoid collisions with other vehicles.
- Drive efficiently by keeping a target speed.

To guide this behavior, a shaped reward function is used, with components that:

- Penalize collisions and off-road behavior.
- Encourage staying in the right lane.
- Reward forward movement at optimal speed.

3.3 Problem Definition

Formally, we define the RL problem as a Markov Decision Process (S, A, P, R, γ) where:

- S is the continuous state space provided by the environment (e.g., positions, velocities, lane index).
- A is the discrete action space consisting of five driving actions.
- P represents the unknown transition dynamics of the environment.
- $R(s, a, s')$ is the reward function shaped to favor lane stability and safety.
- $\gamma \in [0, 1]$ is the discount factor.

3.4 Constraints and Challenges

The main challenges in this setting are:

- **Continuous state space:** Requires either state discretization or function approximation.
- **Sparse rewards:** Collisions are rare, and rewards may not provide dense feedback without shaping.
- **Dynamic interactions:** Multiple agents interact on the road, increasing complexity.

3.5 Evaluation Metrics

To evaluate the agent's performance, we use:

- **Average cumulative reward** per episode.
- **Collision frequency** across evaluation episodes.
- **Lane deviation** and time spent centered in lane.

These metrics will allow us to compare the effectiveness of tabular vs. approximate methods under different learning and exploration setups.

4. Methodology

4.1 Tabular Methods

We implemented and evaluated three tabular reinforcement learning algorithms: Q-learning, Double Q-learning, and Prioritized Sweeping. All methods shared a common structure in terms of state representation, reward shaping, and dynamic hyperparameter tuning.

State Discretization The `highway-env` environment provides a continuous observation space with five ego-centric kinematic features: presence, position (x), lateral lane position (y), forward velocity (vx), and lateral velocity (vy). To enable tabular learning, we discretized each of these features using `np.digitize()` over predefined bin intervals. This transformation produced a compact but expressive finite state representation: $(p\_bin, x\_bin, y\_bin, vx\_bin, vy\_bin)$.

Reward Shaping To guide learning toward safer and more efficient driving behavior, a shaped reward function was designed to:

- Encourage higher forward velocity, scaled to the environment's maximum speed.
- Penalize collisions with a large negative reward.
- Provide mild positive rewards for maintaining the center lane (assumed ideal).
- Discourage unnecessary lane changes with a small penalty.
- Encourage overtaking when feasible.

Dynamic Hyperparameter Tuning Both the exploration rate (ϵ) and learning rate (α) were dynamically decayed to ensure early-stage exploration and later-stage convergence:

- ϵ was initialized at 1.0 and decayed exponentially to 0.05 with a decay rate of 0.995.
- α was initialized at 0.1 and decayed to 0.01 with a decay rate of 0.99.

4.1.1 Q-Learning

Q-learning is an off-policy, model-free algorithm that updates action-value estimates using the Bellman optimality equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right) \tag{1}$$

Action selection followed an ϵ-greedy policy, gradually favoring exploitation as ϵ decayed.

4.1.2 Double Q-Learning

Double Q-learning mitigates overestimation bias by maintaining two independent Q-tables, Q_1 and Q_2. At each step, one table is updated using the action derived from the other:

$$Q_1(s,a) \leftarrow Q_1(s,a) + \alpha \left(r + \gamma Q_2(s', \arg\max_{a'} Q_1(s',a')) - Q_1(s,a) \right) \tag{2}$$

Action selection is based on the sum of the estimates:

$$\arg\max_a \left(Q_1(s,a) + Q_2(s,a) \right)$$

4.1.3 Prioritized Sweeping

Prioritized Sweeping combines model-based planning with efficient updates. After each environment step, the transition model $(s,a) \rightarrow (r,s')$ is recorded, and a priority metric is computed:

$$P = \left| r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right| \tag{3}$$

Transitions exceeding a threshold θ are added to a priority queue. A fixed number of planning updates are performed on the highest-priority transitions, focusing learning where it matters most.

4.2 Function Approximation

To overcome the curse of dimensionality and handle continuous state features more naturally, we implemented function approximation techniques using tile coding and linear semi-gradient updates. This allowed generalization across unseen states while retaining interpretable updates.

Feature Representation We defined a five-dimensional, normalized feature vector derived from the ego vehicle's kinematics and nearby traffic:

- Ego position (x) and velocity (vx)
- Lane index (normalized to $[0,1]$)
- Relative distance to the nearest front vehicle in the same lane
- Relative velocity with respect to that vehicle

These features were passed through a multi-dimensional tile coder with 16 overlapping tilings. The resulting feature vector was a sparse binary activation of tiles, enabling high-resolution generalization without explicit discretization of the entire space.

Custom feature extraction To improve the efficiency and relevance of the state representation, we designed a custom feature extraction function. This function condensed the raw high-dimensional observation array into a compact set of five key features: normalized ego vehicle position, velocity, lane index, the minimum distance to the front vehicle in the same lane, and the relative velocity to that vehicle. This extended state representation allowed the agent to focus on the most critical factors for decision-making, reduced the input dimensionality, and improved the efficiency of the tile coding process.

Exploration and Learning Rate Both agents used an ϵ-greedy policy for action selection, with exploration decaying from 1.0 to a minimum of 0.15. The learning rate α was scaled inversely with the number of tilings to ensure stable and balanced updates across overlapping tiles.

Reward Shaping The same reward function used in tabular methods was applied here to ensure consistency in learning objectives across different representations. This includes speed incentives, collision penalties, and lane-discipline shaping.

Function approximation enabled the agent to learn effective driving policies in a richer, continuous environment where tabular methods would struggle due to state explosion or data inefficiency.

4.2.1 Expected SARSA with Tile Coding

Expected SARSA is an on-policy reinforcement learning algorithm that updates its action-value estimates using the expected Q-value over all possible actions, weighted by the current ϵ-soft policy:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \sum_{a'} \pi(a'|s') Q(s',a') - Q(s,a) \right) \tag{4}$$

The function approximation implementation used tile-coded feature vectors and linear weight vectors for each action. Semi-gradient updates were applied only on the active tiles, ensuring scalable and efficient learning. Expected SARSA yielded smoother and more stable policies, particularly in noisy or ambiguous traffic conditions.

4.2.2 Q-Learning with Function Approximation

Q-learning with function approximation is an off-policy algorithm that applies the Bellman optimality equation with a maximum operator:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right) \tag{5}$$

This method used the same tile-coded feature representation as Expected SARSA, with independent weight vectors for each action. Updates were performed using the semi-gradient method to adjust only the weights corresponding to active features. While Q-learning exhibited more aggressive value propagation, it was more sensitive to overestimation in uncertain environments compared to its on-policy counterpart.

5. Results and Discussion

5.1 Tabular Learning

We evaluated three tabular reinforcement learning methods: Q-learning, Double Q-learning, and Prioritized Sweeping. All models used a unified setup involving state discretization, a custom reward function, and dynamic hyperparameter tuning. The agent's state was encoded as a discretized tuple (x_bin, y_bin, vx_bin), and the reward function incentivized efficient, collision-free driving with a preference for staying in the rightmost lane. The agents were evaluated over 1000 episodes, tracking both episodic rewards and episode lengths.

5.1.1 Q-Learning

Q-learning displayed consistent improvement over time, achieving the highest final reward among the three methods (75.21). Its learning curve showed moderate early-stage variance, but the integration of epsilon and alpha decay helped stabilize policy updates in later episodes. As training progressed, Q-learning exhibited strong lane-following behavior and prolonged episode durations, often reaching the environment's time limit.

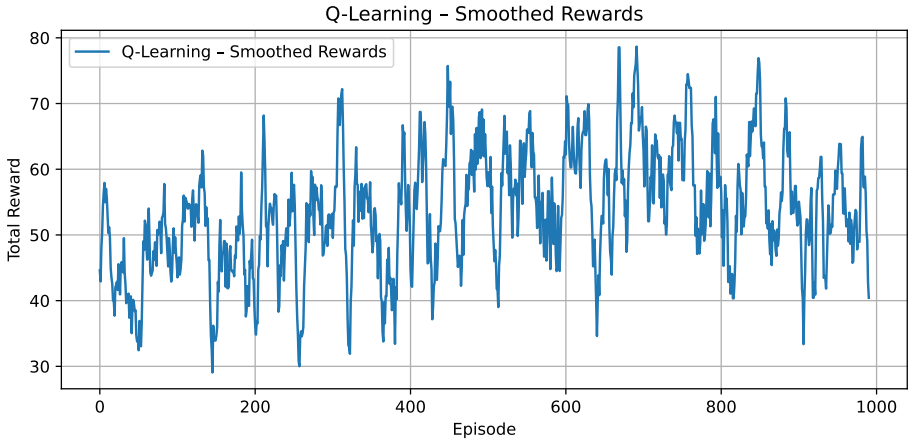

Figure 1: Q-Learning Smoothed Rewards

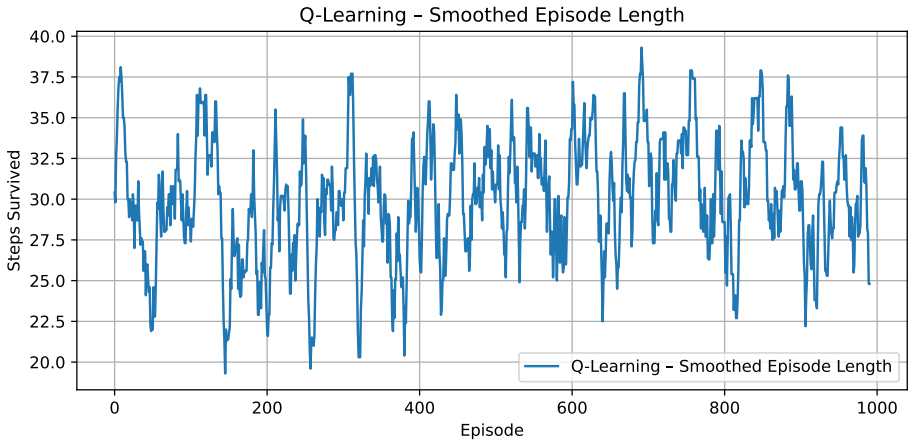

Figure 2: Q-Learning Episode Length

5.1.2 Double Q-Learning

Double Q-learning achieved the highest average reward (60.56), indicating strong early performance and rapid initial convergence. However, its final reward dropped significantly to 15.56, suggesting a possible collapse in policy stability. This could be due to the interplay between decaying learning rates and reduced exploration, which may have inhibited sufficient policy refinement in later episodes.

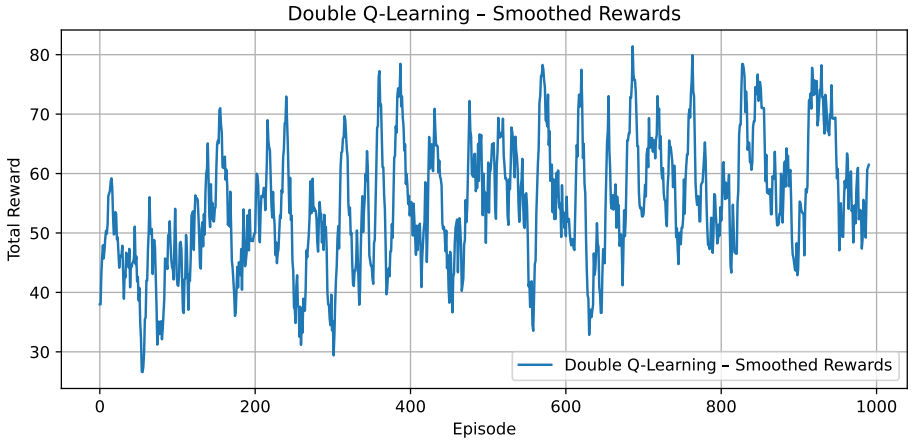

Figure 3: Double Q-Learning Smoothed Rewards

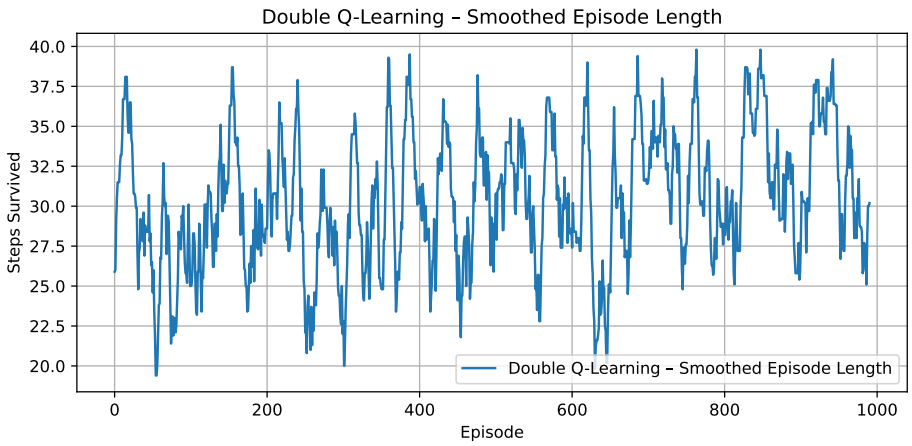

Figure 4: Double Q-Learning Episode Length

5.1.3 Prioritized Sweeping

Prioritized Sweeping offered a strong balance between performance and stability. While it did not achieve the highest peak reward, it maintained consistent learning with a relatively low standard deviation (13.89). The agent demonstrated efficient planning and early convergence, achieving a final reward of 50.95 and sustaining long survival durations per episode.

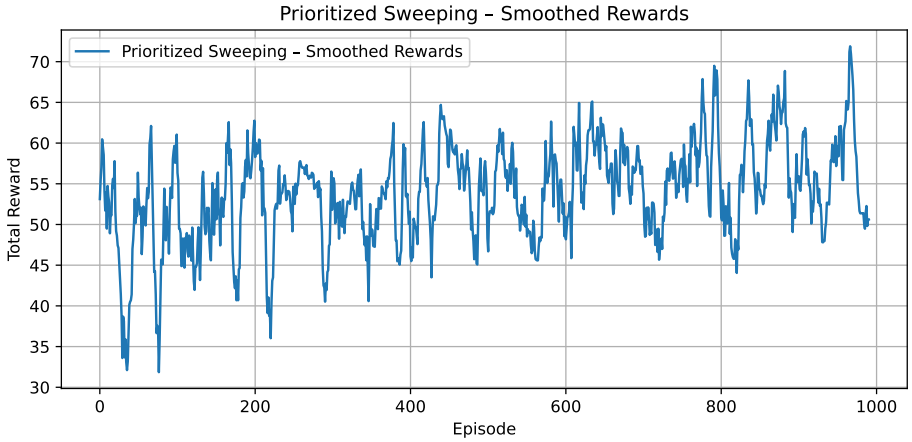

Figure 5: Prioritized Sweeping Smoothed Rewards

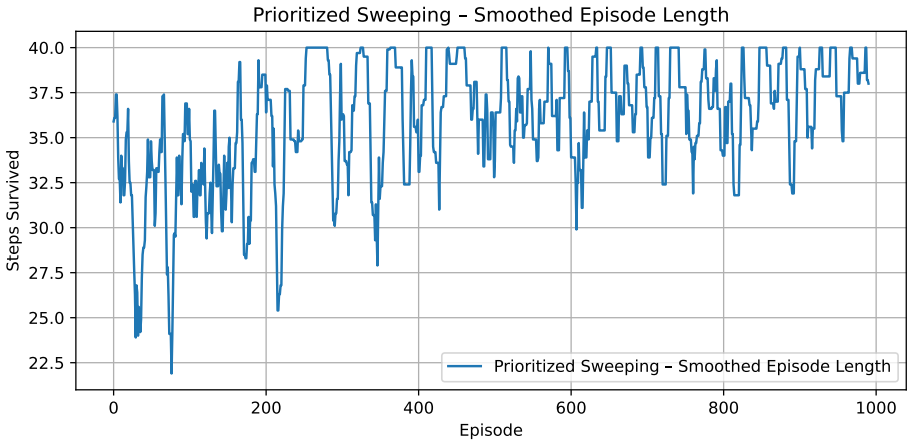

Figure 6: Prioritized Sweeping Episode Length

5.1.4 Comparison of Tabular Methods

Each of the three algorithms exhibited distinct learning characteristics and trade-offs in terms of stability, convergence, and final policy quality.

Q-learning achieved the highest final reward of 75.21, suggesting strong long-term learning when combined with decaying hyperparameters. However, it also showed significant variance (standard deviation of 27.64), indicating instability during the early and middle phases of training.

Double Q-learning produced the highest average reward overall (60.56), reflecting efficient early learning and initial convergence. Despite this, its final reward dropped sharply to 15.56, revealing sensitivity to the decaying learning rate and potential issues with maintaining stable policies over longer training periods.

Prioritized Sweeping emerged as the most consistent method. It maintained a high average reward (56.26) and the lowest standard deviation (13.89), indicating smooth and stable learning. Although its final reward (50.95) did not match Q-learning's peak, it balanced learning speed with robustness, making it a reliable choice in scenarios where stability is prioritized.

In summary, Q-learning excels in long-term policy refinement, Double Q-learning in fast early learning, and Prioritized Sweeping in learning stability and planning efficiency.

Method	**Average Reward**	**Standard Deviation**	**Final Reward**
Q-Learning	51.97	27.64	**75.21**
Double Q-Learning	**60.56**	28.49	15.56
Prioritized Sweeping	56.26	**13.89**	50.95

Table 1: Performance metrics computed across 1000 episodes.

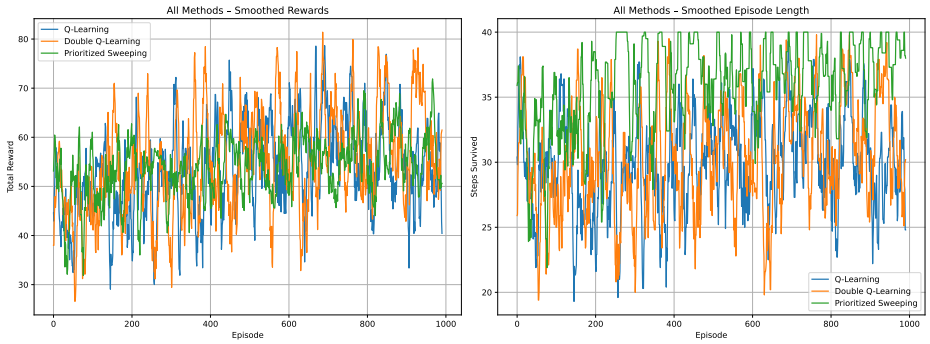

Figure 7: Comparison of Tabular Methods — Rewards and Episode Lengths

5.2 Function Approximation

We evaluated two reinforcement learning methods with function approximation: Expected SARSA with tile coding and Q-learning with tile coding. Both models used a unified setup with continuous state representation, a custom reward function, and epsilon decay for exploration control. The agent's state was encoded using tile coding over continuous features, enabling generalization across similar states while preserving fine-grained distinctions. The reward function incentivizes efficient, collision-free driving with a preference to stay in the rightmost lane. The agents were evaluated over 1000 episodes, tracking both episodic rewards and episode lengths, with particular focus on convergence speed and stability.

5.2.1 Expected SARSA

As shown in figure 8 we evaluated the performance of the Expected SARSA agent with tile coding over 1000 training episodes. The agent showed rapid initial learning, with a steep increase in episodic rewards during the first few hundred episodes as it learned to navigate the highway environment efficiently. Because the use of an epsilon-soft policy, Expected SARSA was able to balance exploration and exploitation effectively, adapting to diverse traffic scenarios and avoiding local optima. We observed some oscillations in the reward curve, especially during the middle phase of training, which can be attributed to the stochastic nature of the expected value updates and ongoing policy adjustments as epsilon decayed. Overall, the agent consistently improved both average reward and episode length, achieving high performance by the end of training. The results demonstrate that Expected SARSA, combined with tile coding and engineered features, can learn stable and near-optimal driving policies in a continuous, dynamic environment.

Figure 8: Expected SARSA agent Rewards

Q-learning We evaluated the Q-learning agent with tile coding over 1000 training episodes shown in figure 9. The agent showed steady improvement in episodic rewards, with a gradual rise during the early episodes as it learned to optimize its policy. Q-learning's off-policy updates allowed it to focus on maximizing long-term rewards, leading to stable value estimates and consistent policy refinement over time. While the initial learning phase progressed more cautiously, the agent displayed robust improvement as training continued, with fewer fluctuations in performance and sustained increases in average reward and episode length. By the end of training, the Q-learning agent demonstrated effective handling of the environment, consistently achieving efficient and collision-free driving behaviors.

Figure 9: Q-learning agent Rewards

5.2.2 Comparison of Function Approximation

We compared the performance of the Expected SARSA and Q-learning agents using three key metrics: average reward, standard deviation, and final reward after 1000 episodes. Both agents achieved strong performance, but with notable differences. Expected SARSA reached an average reward of 279.97 with a higher standard deviation of 53.53, reflecting

its more exploratory and occasionally variable behavior during training. In contrast, Q-learning achieved a slightly higher average reward of 283.16 and a lower standard deviation of 38.32, indicating more stable and consistent learning. Notably, Q-learning reached a final reward of 295.00, outperforming Expected SARSA's final reward of 276.18, which suggests that Q-learning's off-policy updates allowed it to converge to a stronger final policy. Overall, both methods were effective, but Q-learning showed an advantage in stability and final performance.

Method	Avg Reward	Std Dev	Final Reward
Expected SARSA	279.97	53.53	276.18
Q-learning	283.16	38.32	295.00

Table 2: Comparison of Expected SARSA and Q-learning performance over 1000 episodes.

5.3 Comparison and Insights

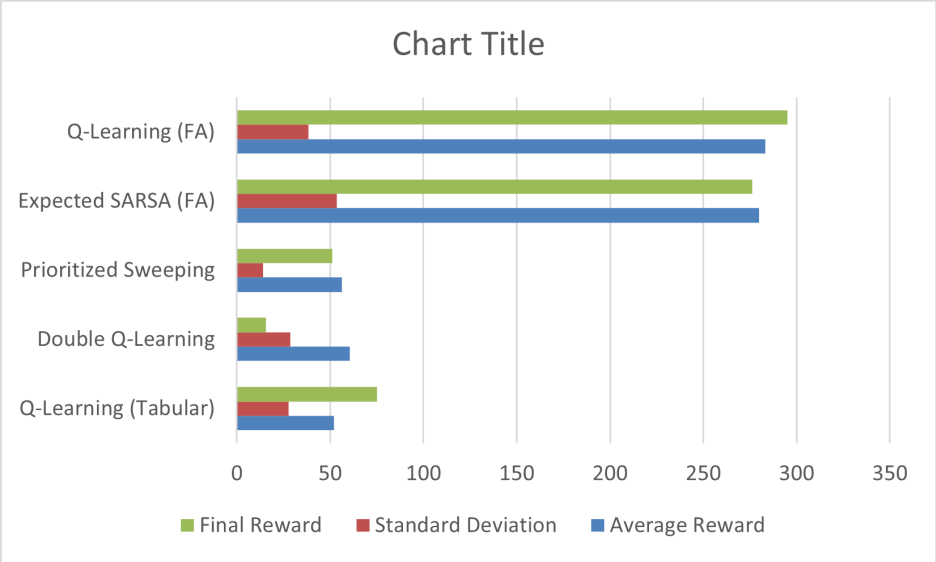

Figure 10: Performance Comparison of Tabular and Function Approximation RL Methods

The graph 10 illustrate comparison of the three tabular reinforcement learning methods - Q-learning, Double Q-learning, and Prioritized Sweeping — against function approximation approaches using tile coding with expected SARSA and Q-learning. The tabular agents operated on a discretized state space, while the function approximation agents leveraged continuous features and generalization through tile coding.

According to the tabular results in Table 1, Double Q-learning achieved the highest average reward (60.56) but showed a sharp drop in final reward (15.56), indicating challenges in policy stability over time. Prioritized Sweeping exhibited the lowest standard deviation (13.89), reflecting stable but moderately performing learning. Q-learning reached the highest

final reward (75.21) among tabular methods, suggesting strong late-stage convergence despite more variable performance overall.

In contrast, the function approximation agents consistently achieved significantly higher rewards across all metrics, as shown in the function approximation table 2. This highlights the advantage of tile coding in handling continuous state spaces and improving generalization. Overall, while tabular methods performed reasonably well in discretized settings, function approximation methods proved more scalable and robust in complex, high-dimensional environments.

6. Conclusions and Discussion

- **Summary of findings:** Our study evaluated both tabular and function approximation reinforcement learning methods in the context of autonomous driving in the highway environment. The tabular agents achieved moderate performance, showing reasonable learning in discretized state spaces but struggling to scale effectively. In contrast, the function approximation agents achieved higher average rewards, greater stability, and better final performance, demonstrating the benefits of using tile coding for generalization over continuous states. Importantly, we found that we could not achieve these high levels of performance without reshaping the reward function to explicitly incentivize efficient, collision-free driving and lane preferences, which greatly accelerated learning and improved policy quality across all methods.

- **Which approach worked best and why:** The function approximation approaches, particularly Q-learning with tile coding, worked best in our experiments. This is because tile coding enabled the agents to generalize across similar states, reducing the need for exhaustive state-action pair storage and allowing efficient learning in a continuous, high-dimensional environment.

- **Limitations of our methods:** One limitation of the tabular methods was their reliance on discretization, which can oversimplify the environment and lead to suboptimal policies. For the function approximation agents, while tile coding improved generalization, it still required manual feature engineering and parameter tuning. Additionally, none of the methods included adaptive exploration strategies beyond basic epsilon decay, which may have limited their efficiency.

- **Suggestions for future improvements:** Future work could explore more advanced state abstraction methods, such as automatic feature selection or learned embeddings, to reduce manual engineering. Extending the agents with deep reinforcement learning approaches, such as Deep Q-Networks (DQN), could enable them to scale to even more complex environments without explicit feature design.

Overall, our findings highlight the importance of combining well-chosen algorithms, effective function approximation, and carefully designed reward structures to achieve robust and scalable reinforcement learning solutions in complex environments.

References

Ahmad Al-Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. End-to-end deep reinforcement learning for lane keeping assist. In *Machine Learning for Intelligent Transportation Systems Workshop (MLITS) at NeurIPS*, 2016.

Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

Edouard Leurent. *Safe and Efficient Reinforcement Learning for Behavioural Planning in Autonomous Driving*. PhD thesis, Université de Lille, 2020.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.

C. J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3–4):279–292, 1992.