Machine Learning Project

Reinforcement Learning using Q-Learning on FrozenLake

Guided By:

Prof. Harald Stein

Project Carried Out By:

Akshay Kumar	- Coding
Rishav Badhan	- Documentation
Arnav Singh Bharadwaj	- Presentation

Table of Contents

1. Problem Statement

Reinforcement Learning (RL) is a branch of machine learning in which an agent learns to make decisions by interacting with an environment and receiving feedback in the form of rewards. Unlike supervised learning, RL does not rely on labeled datasets; instead, the agent must discover optimal behavior through trial and error.

The objective of this project is to implement and analyze a tabular Q-learning algorithm on the FrozenLake environment provided by the Gymnasium framework. The focus is not only on achieving a successful solution but also on understanding how learning evolves over time through visual feedback.

To achieve this goal, a complete end-to-end system was developed consisting of a backend training engine and a frontend visualization interface. This system enables real-time observation of environment interactions, reward progression, and Q-table evolution, making reinforcement learning concepts more intuitive and interpretable.

2. Environment Description

The FrozenLake-v1 environment is a discrete grid-world problem commonly used to introduce reinforcement learning concepts. The environment consists of a 4×4 grid representing a frozen surface where the agent must navigate from a starting position to a goal state.

Each cell in the grid corresponds to a discrete state. The agent can perform four actions: move up, move down, move left, or move right. Some grid cells represent holes, which are terminal states that end the episode unsuccessfully. Reaching the goal state ends the episode successfully.

The reward structure of FrozenLake is sparse: the agent receives a reward of +1 only when it reaches the goal, and 0 for all other transitions. This sparse feedback makes learning challenging and highlights the importance of effective exploration strategies.

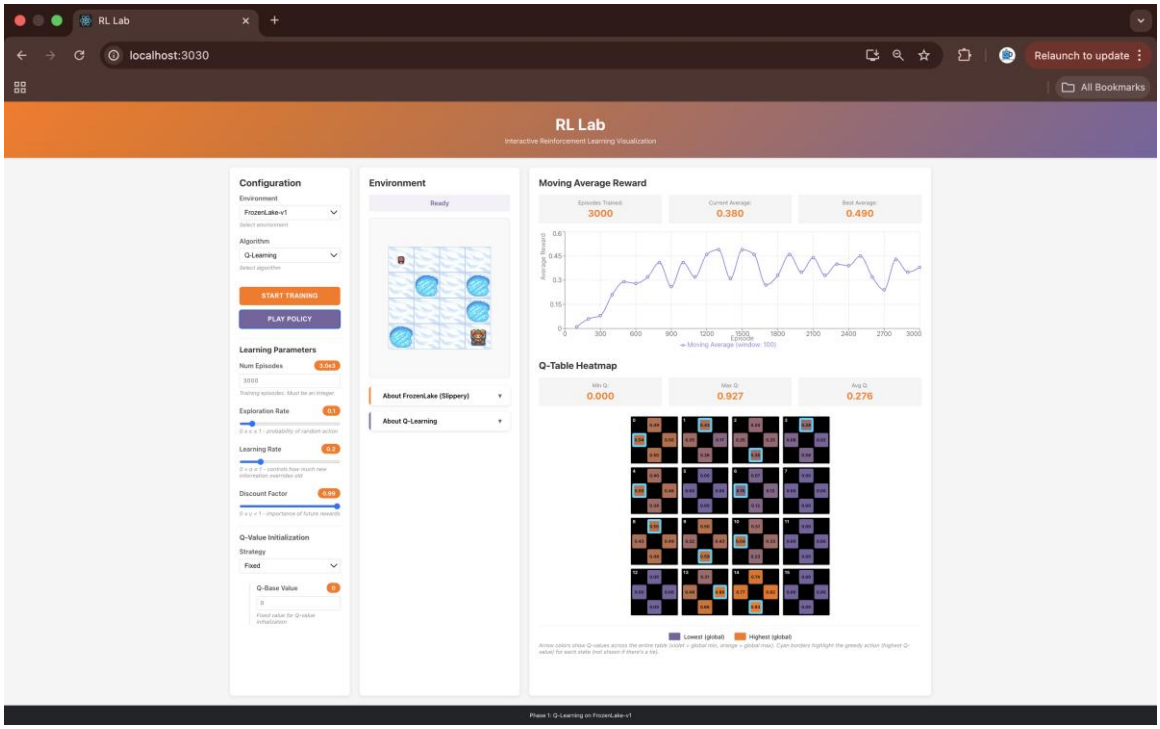

Figure 1: FrozenLake 4×4 grid environment showing the start state, frozen tiles, holes, and the goal state.

Two variants of the environment are used in this project:
- Deterministic (NoSlip): Actions always result in the intended movement.
- Stochastic (Slippery): Actions may result in unintended movements due to slipping.

The deterministic variant allows clear observation of convergence, while the stochastic variant demonstrates the impact of uncertainty on learning performance.

3. Methodology

3.1 Environment Analysis

What: The FrozenLake environment was analyzed to understand its state space, action space, transition dynamics, and reward structure.

Why: A thorough understanding of the environment is required to design an appropriate reinforcement learning solution and to correctly interpret learning behavior.

How: The grid layout, terminal states, possible transitions, and reward signals were examined. States were mapped to integer indices to support tabular learning.

Result: A complete mapping between grid positions and discrete states was established, enabling effective application of tabular Q-learning.

3.2 Q-Table Initialization

What: Initialization of the Q-table containing action-value estimates for all state-action pairs.

Why: Initial Q-values influence early agent behavior and exploration dynamics.

How: Two initialization strategies were supported: zero initialization and random initialization within a specified range. These strategies were selectable through the frontend interface.

Result: Random initialization encouraged early exploration, while zero initialization provided stable and predictable convergence behavior.

3.3 Hyperparameter Selection

What: Selection of learning rate (α), discount factor (γ), exploration rate (ε), and number of training episodes.

Why: These hyperparameters directly affect learning speed, stability, and the balance between exploration and exploitation.

How: Hyperparameters were exposed through the user interface and adjusted experimentally across multiple training runs.

Result: Effective learning was observed for α values between 0.1 and 0.5, γ values close to 1, and ε values around 0.1.

3.4 Action Selection Strategy

What: An ε-greedy action selection strategy was implemented.

Why: Purely greedy policies may converge prematurely to suboptimal solutions.

How: With probability ε, the agent selects a random action; otherwise, it selects the action with the highest Q-value.

Result: The agent was able to explore sufficiently while still exploiting learned knowledge.

3.5 Q-Learning Update Rule

What: Q-values were updated using the Bellman optimality equation.

Why: The update rule allows the agent to incorporate new experience into its value estimates.

How: After each action, the Q-value was updated according to:
$Q(s,a) \leftarrow Q(s,a) + \alpha [r + \gamma \max Q(s',a') - Q(s,a)]$

Result: Reward information propagated backward through the state space, enabling convergence toward optimal values.

3.6 Training Loop Design

What: An episode-based training loop was designed.

Why: Episodes allow the agent to experience complete trajectories from start to termination.

How: For each episode, the agent repeatedly selected actions, observed transitions, updated Q-values, and terminated upon reaching a goal or hole.

Result: Progressive improvement of policy quality over episodes was observed.

3.7 Performance Evaluation

What: Evaluation of learning progress during training.

Why: Single episodic rewards are noisy and do not clearly reflect learning trends.

How: A moving average of episodic rewards was computed and visualized in real time.

Result: Convergence trends and stability of learning were clearly visible.

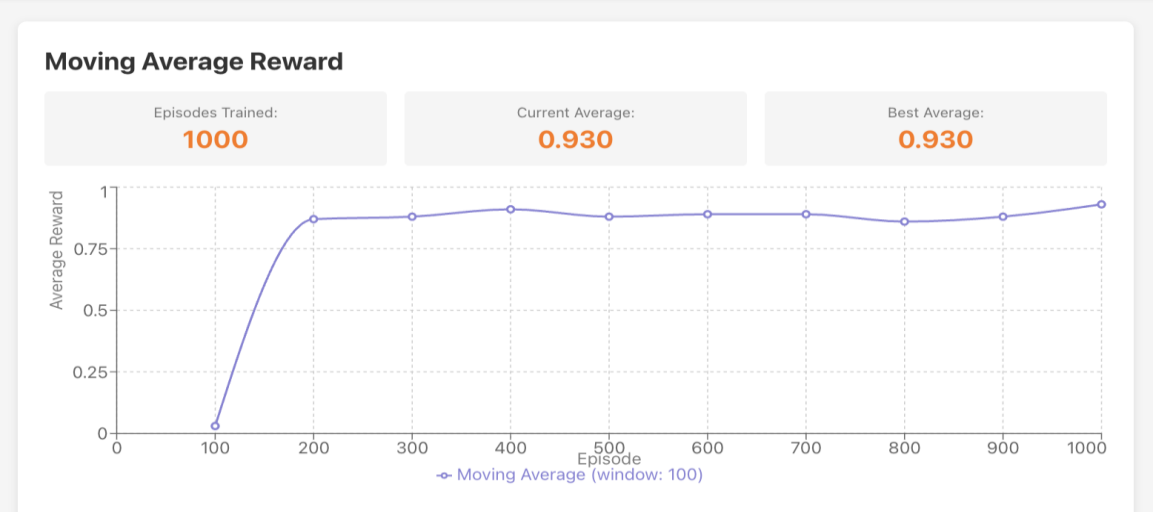

Figure 2: Moving average episodic reward during training in the deterministic FrozenLake environment.

4. Results and Analysis

In the deterministic environment, the agent consistently learned an optimal policy that successfully reached the goal state. The reward curve demonstrated rapid convergence after sufficient exploration.

In the stochastic environment, learning improved success rates but did not guarantee success in every episode due to random transitions. Analysis of the Q-table revealed lower confidence values compared to the deterministic case.

Visualization of the Q-table provided insight into how value estimates propagate from the goal state back to earlier states.

Figure 3: Learned Q-table visualization after training, with arrows indicating the optimal action for each state.

5. Discussion and Conclusion

This project demonstrates a complete reinforcement learning pipeline using Q-learning. The combination of algorithm implementation and real-time visualization provides strong educational value.

Limitations include the scalability of tabular methods to large or continuous state spaces. Future work includes extending the framework to Deep Q-Networks (DQN) and other advanced reinforcement learning algorithms

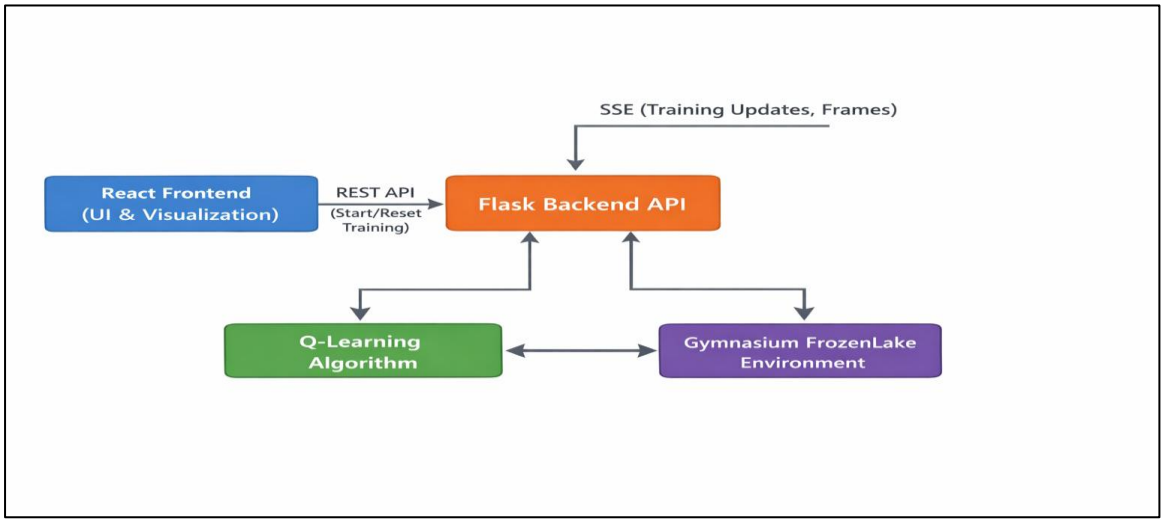

Figure 4: System architecture of the RL Lab application showing frontend visualization, backend control, Q-learning module, and environment interaction.

Overall, the project successfully bridges theoretical reinforcement learning concepts with practical implementation.

6. Code Availability

All source code, including backend services, frontend visualization, and Docker configuration files, is included in the submitted project repository