Reinforcement Learning Visualizer - Technical Report

1. Project Overview

The **Reinforcement Learning Visualizer** is an interactive web application built to demonstrate and visualize core concepts of Reinforcement Learning (RL). It provides a hands-on learning experience, allowing users to experiment with different RL algorithms, environments, and parameters to better understand how reinforcement learning works in practice.

2. Implemented Environments

2.1 GridWorld (5×5 Grid)

- **Type**: Discrete state space
- **Actions**: 4 (Up, Right, Down, Left)
- **Features**:
    - Goal state located at the bottom-right corner
    - Obstacles placed at positions [6, 12]
    - Deterministic transitions (fixed outcomes)
    - Complete transition model available
- **Rewards**: Users can configure the goal reward, step penalty, and obstacle penalty.

2.2 Frozen Lake (4×4 Grid)

- **Type**: Discrete state space
- **Actions**: 4 (Up, Right, Down, Left)
- **Features**:
    - Slippery transitions (33% chance of slipping)
    - Holes placed at positions [5, 7, 11, 12]
    - Goal at the bottom-right corner
    - Complete transition model available
- **Rewards**: Users can adjust the goal reward, hole penalty, and step penalty.

2.3 CartPole

- **Type**: Continuous state space (discretized into 100 states)
- **Actions**: 2 (Left, Right)
- **Features**:

    - Simulates continuous physics (pole balancing)
    - State discretization based on position (10 bins) and angle (10 bins)
    - No transition model required
- **Rewards**: Customizable step reward and fail penalty.

2.4 Mountain Car

- **Type**: Continuous state space (discretized into 100 states)
- **Actions**: 3 (Left, Nothing, Right)
- **Features**:
    - Simulates continuous physics, including gravity and friction
    - State discretization based on position (10 bins) and velocity (10 bins)
    - Goal state at position ≥ 0.5
    - No transition model required
- **Rewards**: Users can adjust goal reward and step penalty.

2.5 Breakout Environment

- **Type**: Discrete state space (200 states)
- **Actions**: 3 (Left, Stay, Right)
- **Features**:
    - A simplified version of the classic Breakout game
    - Real-time physics for paddle, ball, and bricks
    - Real-time collision detection
    - No transition model required
- **Rewards**: Configurable rewards for brick destruction, miss penalties, and win conditions.

3. Implemented Algorithms

3.1 Dynamic Programming Methods

Policy Evaluation

- **Category**: DP (Requires transition model)
- **Description**: Evaluates the value function of a given policy.
- **Use Case**: GridWorld, Frozen Lake

Policy Iteration

- **Category**: DP (Requires transition model)
- **Description**: Alternates between policy evaluation and improvement.
- **Use Case**: GridWorld, Frozen Lake

Value Iteration

- **Category**: DP (Requires transition model)
- **Description**: Computes the optimal value function directly.
- **Use Case**: GridWorld, Frozen Lake

3.2 Model-Free Methods

Monte Carlo

- **Category**: Model-Free (No transition model needed)
- **Description**: Learns from complete episodes.
- **Use Case**: All environments

TD(0)

- **Category**: Temporal Difference
- **Description**: One-step temporal difference learning.
- **Use Case**: All environments

n-step TD

- **Category**: Temporal Difference
- **Description**: Multi-step temporal difference.
- **Use Case**: All environments

3.3 Control Algorithms

SARSA

- **Category**: On-Policy TD Control
- **Description**: On-policy action-value learning.
- **Use Case**: All environments

Q-Learning

- **Category**: Off-Policy TD Control
- **Description**: Off-policy optimal action-value learning.
- **Use Case**: All environments

4. Parameter Adjustment Capabilities

4.1 Algorithm Parameters

Parameter	Range	Description	Applicable Algorithms
γ (Gamma)	0-1	Discount factor for future rewards	All algorithms
α (Alpha)	0.01-1	Learning rate for updates	Q-Learning, SARSA, TD methods
ε (Epsilon)	0-1	Learning rate for updates	Q-Learning, SARSA, Monte Carlo
n-step	1-10	Look-ahead steps	n-step TD
Iterations	1-500	DP algorithm iterations	Policy/Value Iteration

4.2 Training Parameters

- **Episodes**: 10-1000 training episodes
- **Batch Size**: Automatic batch processing for smooth UI updates
- **Progress Tracking**: Real-time progress bar and episode count

4.3 Environment Reward Customization

Each environment has customizable reward settings:

- **GridWorld**: Goal reward, step penalty, obstacle penalty
- **Frozen Lake**: Goal reward, hole penalty, step penalty
- **CartPole**: Step reward, fail penalty, max steps
- **Mountain Car**: Goal reward, step penalty, max steps
- **Breakout**: Brick reward, miss penalty, step penalty, win reward

5. Visualization Techniques

5.1 Environment Visualization

Grid-based Environments (GridWorld, Frozen Lake)

- **Heat Map**: State values color-coded using a blue gradient.
- **Policy Arrows**: Indicate optimal action directions.
- **Agent Position**: Orange circle representing the agent's position.
- **Special States**:
  - Green for the goal state.
  - Red for obstacles/holes.
- **Interactive Cells**: Click to inspect state values.

Continuous Environments (CartPole, Mountain Car)

- **Physics Simulation**: Real-time motion visualization.
- **Parameter Displays**: Displays real-time angle, position, and velocity.
- **Force Indicators**: Displays velocity vectors and applied forces.
- **Boundary Markers**: Failure boundaries shown on the environment.

Breakout Environment

- **Game Elements**: Bricks, paddle, and ball with 3D effects.

- **Score Display**: Real-time score and brick count.
- **Velocity Vector**: Ball direction indicator.
- **Status Overlay**: "PLAYING" indicator during inference.

5.2 Training Visualization

Value Function Display

- **Grid Layout**: Displays state values in matrix format.
- **Color Coding**: Blue intensity represents state values.
- **Tooltips**: Hover to see exact values.
- **Cell Selection**: Click to highlight specific states.

Metrics Charts

- **Episode Returns**: Green line chart showing reward progression.
- **Convergence (ΔV)**: Red line chart showing DP algorithm convergence.
- **Statistics Display**:
    - Last return value
    - Average return
    - Episode count
    - Total steps

5.3 Control Panel Visualization

Status Indicators

- **Badge System**: Color-coded status (Training, Trained, Running, Paused).
- **Parameter Display**: Displays current values for γ, α, ε.
- **State Counter**: Shows the current environment state.

Progress Tracking

- **Progress Bar**: Displays training completion percentage.
- **Episode Counter**: Shows the current/total episodes.
- **Real-time Updates**: Live updates of parameter changes.

5.4 Interactive Features

- **Real-time Parameter Adjustment**: Sliders provide instant feedback.
- **Environment Switching**: Seamless transitions between environments.
- **Algorithm Comparison**: Compare algorithms side-by-side.
- **Reward Customization**: Dynamically adjust reward settings.
- **Training Control**: Includes start, pause, resume, and reset functions.

6. Key Features

1. **Modular Design**: Separate environments, algorithms, and UI components.
2. **Real-time Updates**: Canvas animations and live parameter feedback.
3. **Error Handling**: Graceful handling of NaN/Infinity values.
4. **Responsive Design**: Adapts to different screen sizes.
5. **Performance Optimization**: Batch training and efficient rendering.