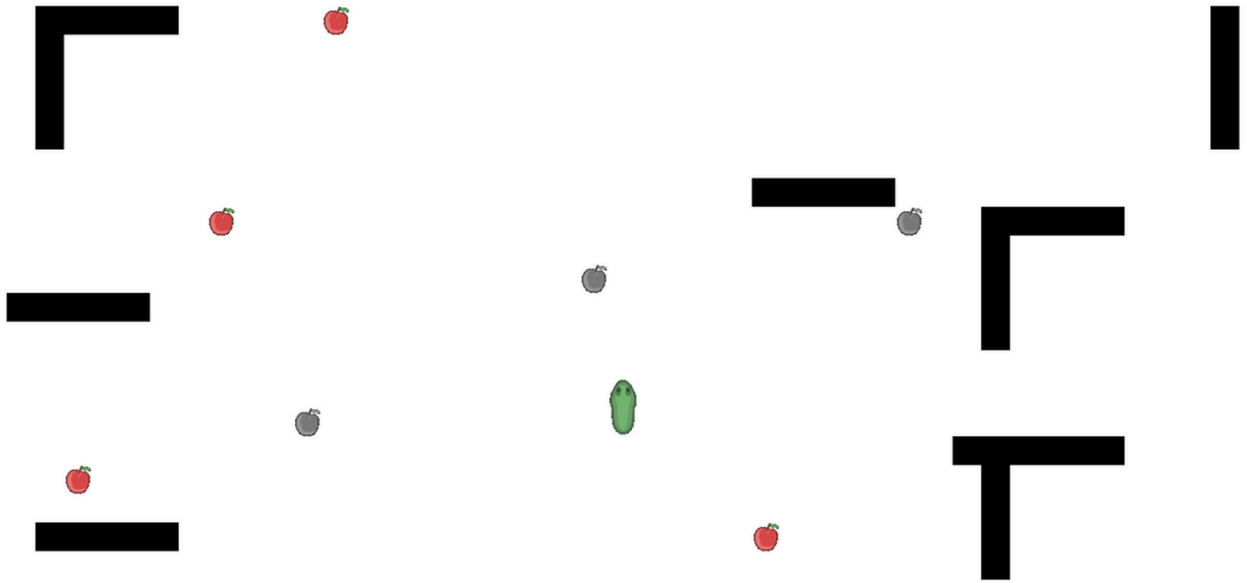
universidade de aveiro

MINI PROJECT 2

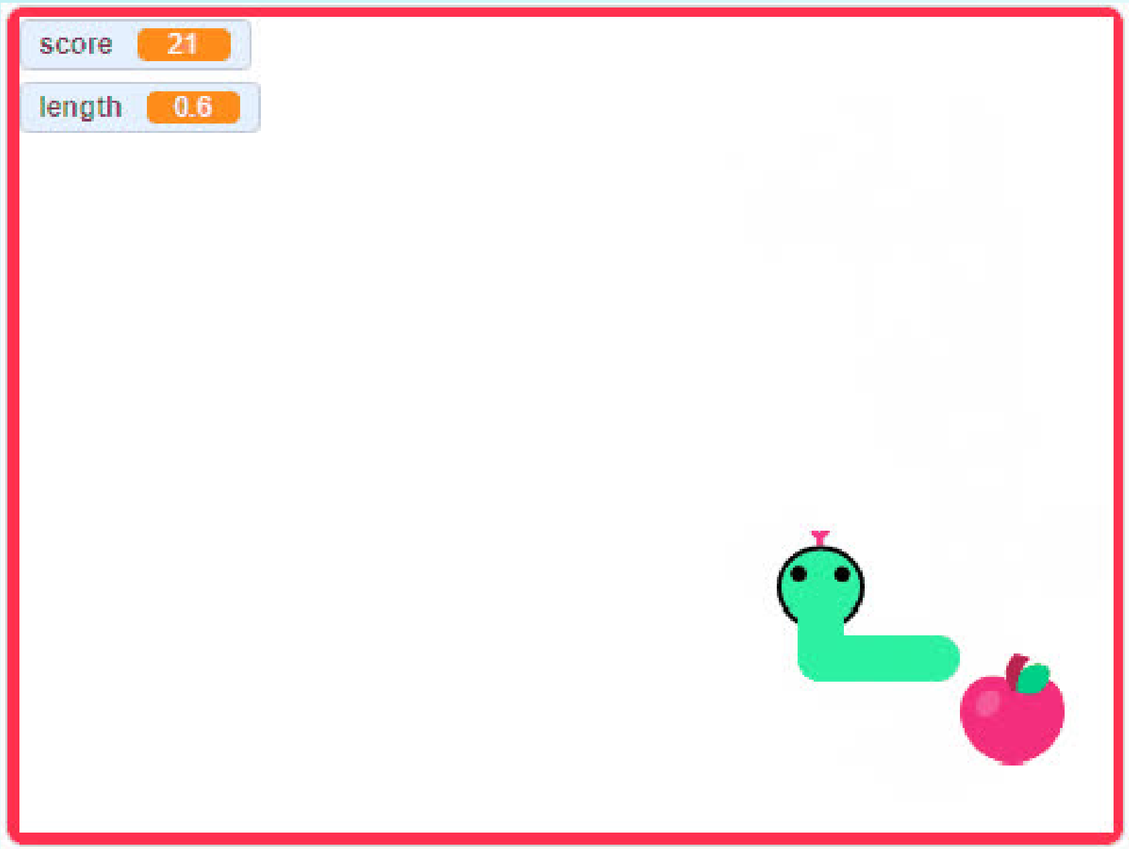


Daniel Emídio (108986)
David Palricas (108780)
Márcio Tavares (108096)

Professor
Luís Seabra Lopes
(lsl@ua.pt)

INTRODUCTION

Objective:



Tools:

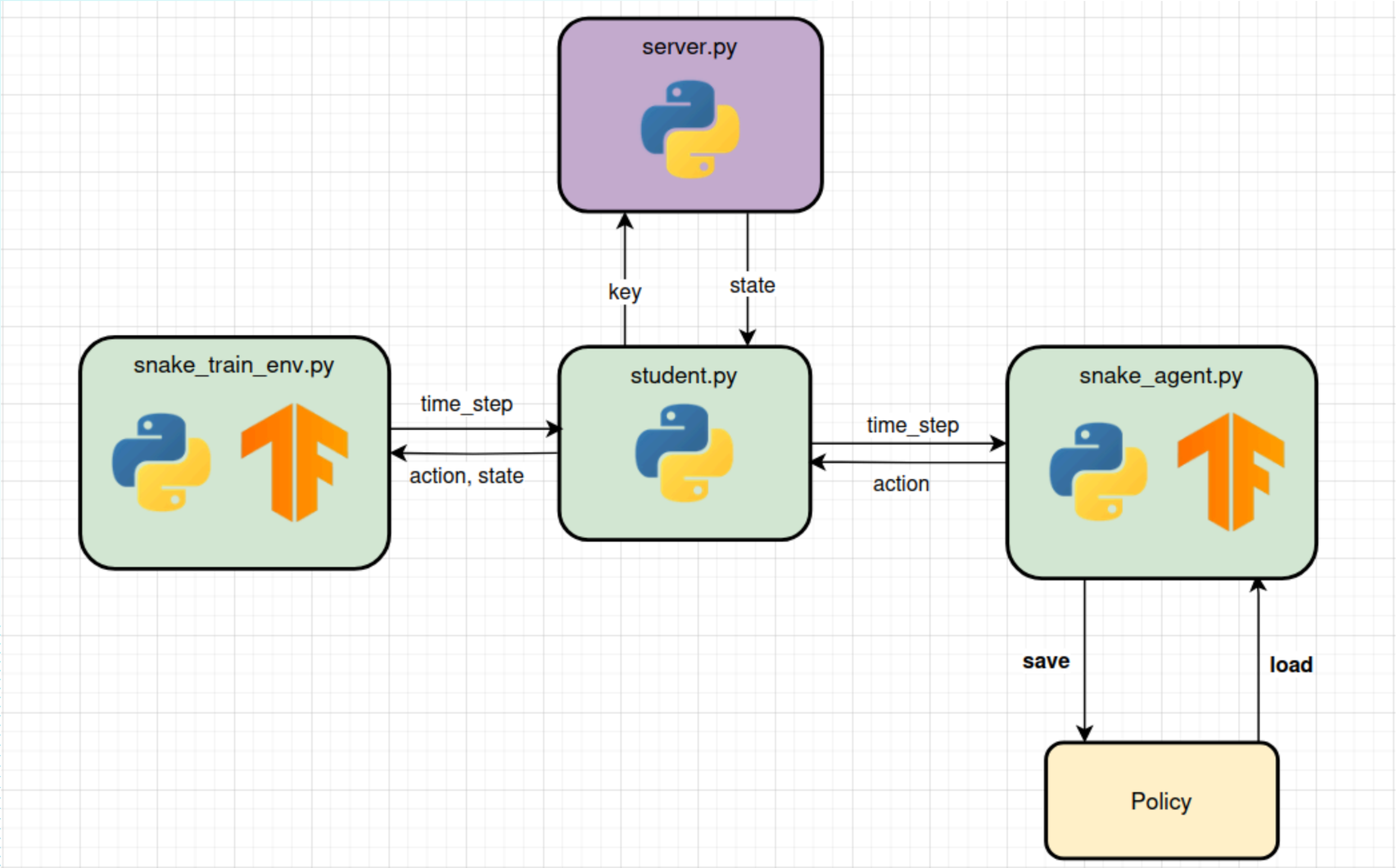
python™

Reinforcement Learning with T-Agents

OVERVIEW

- **Arquitecture**
- **Environment**
  - State representation;
  - Reward function;
  - Step handling.
- **Agent**
  - Algorithm selection;
  - Network architecture;
  - Episode handling.
- **Analysis**
  - Performance and results;
  - Learning progression;
  - Agent behavior.
- **Implementation**
  - Implementation Challenges;
  - Optimization Attempts;
  - Future Optimization Directions.
- **Demo**

ARQUITECTURE

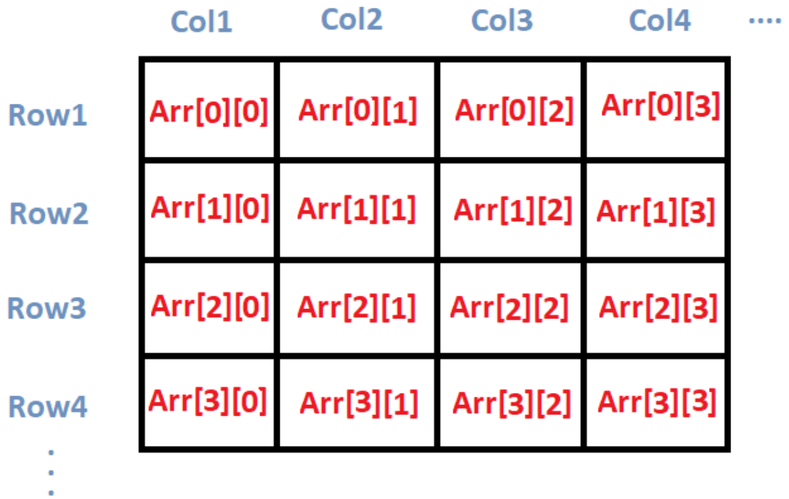


ENVIRONMENT

State representation

Map:

0. Empty
1. Wall
2. Food
3. S_Food
4. Body




```
'map': array_spec.BoundedArraySpec(
    shape=map_shape,
    dtype=np.int32,
    minimum=0,
    maximum=4,
    name='map'),
'traverse': array_spec.BoundedArraySpec(
    shape=(),
    dtype=np.int32,
    minimum=0,
    maximum=1,
    name='traverse'),
'range': array_spec.ArraySpec(
    shape=(),
    dtype=np.int32,
    name='range'),
'direction': array_spec.BoundedArraySpec(
    shape=(),
    dtype=np.int32,
    minimum=0,
    maximum=3,
    name='direction'),
'timeout': array_spec.BoundedArraySpec(
    shape=(),
    dtype=np.int32,
    minimum=0,
    maximum=timeout,
    name='timeout'),
'score': array_spec.ArraySpec(
    shape=(),
    dtype=np.int32,
    name='score')
```

ENVIRONMENT

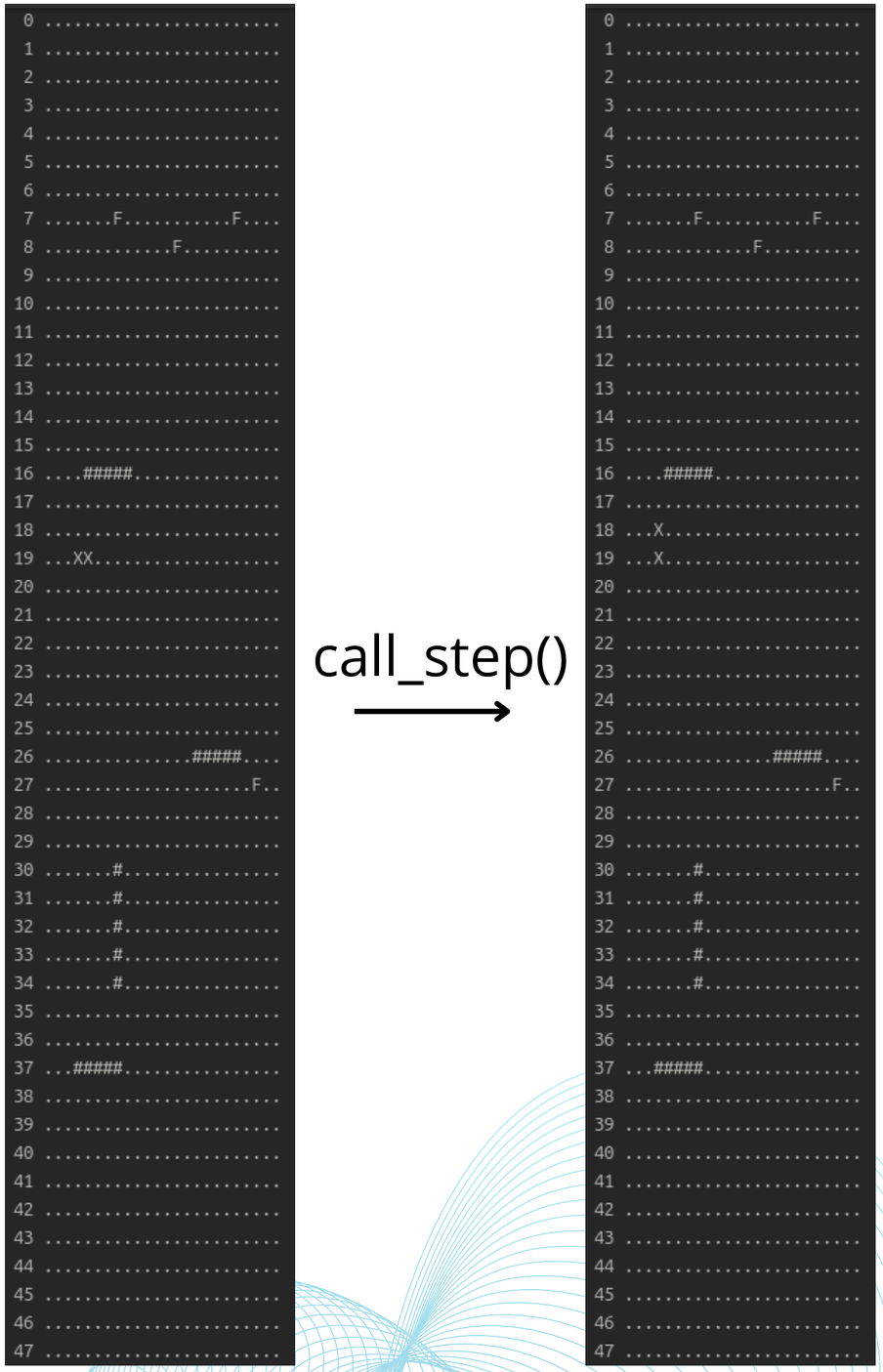
Reward function

```python
def _calculate_reward(self):
        """cálculo de reward."""
        reward = 0.0

        # 1. Penalidade fim de jogo
        game_over_penalty = self._check_game_over_penalty()
        if game_over_penalty != 0:
            return game_over_penalty

        # 2. Reward por comer comida
        food_reward = self._calculate_food_reward()
        reward += food_reward

        # 3. Reward/penalty baseado na distância
        distance_reward = self._calculate_distance_reward()
        reward += distance_reward

        # 4. Penalty por movimentos perigosos
        danger_penalty = self._calculate_immediate_danger_penalty()
        reward += danger_penalty

        # 5. Reward de sobrevivência
        survival_reward = 0.2  # Mais incentivo para ficar vivo
        reward += survival_reward

        # 6. Penalty de eficiência
        efficiency_penalty = self._calculate_efficiency_penalty()
        reward += efficiency_penalty

        # Normalizar reward
        reward = max(-100, min(reward, 100))
        return float(reward)
```

ENVIRONMENT

Step handling

1. Gameover verification
2. Update scalar observations
3. Update *map* acording to sight
4. Update snake position
5. Calculate reward



AGENT

Type

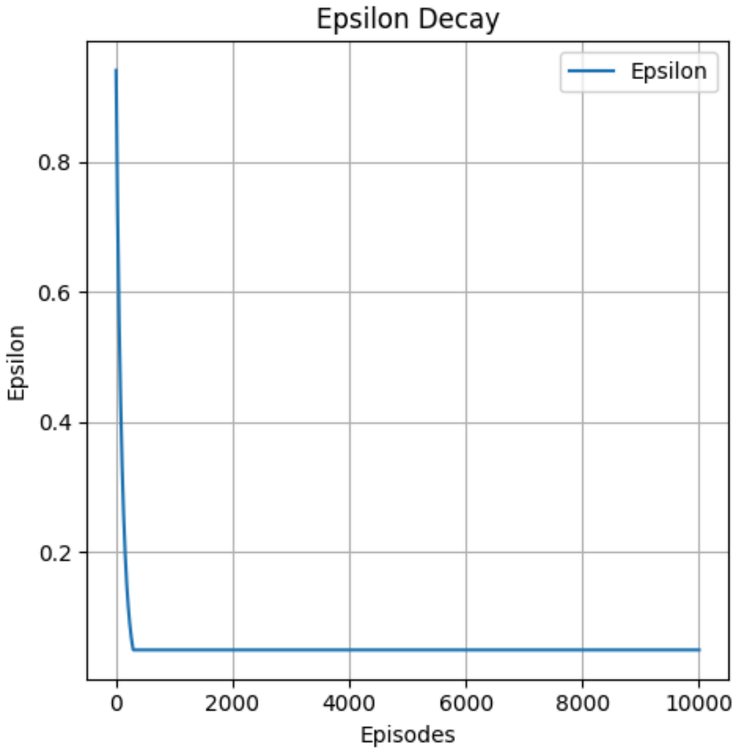
Learning Strategy

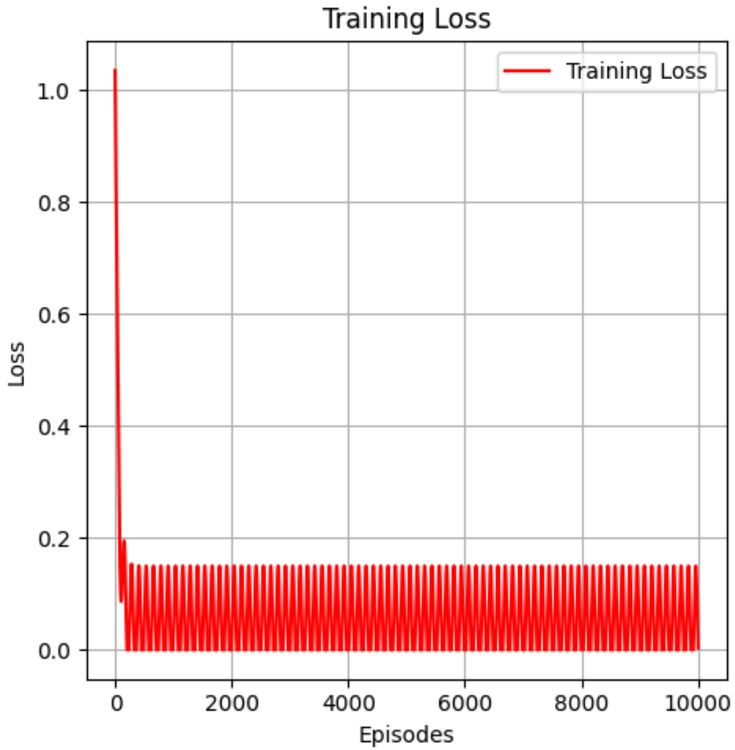
- Epsilon-Greedy
- Replay Buffer
- Q Network

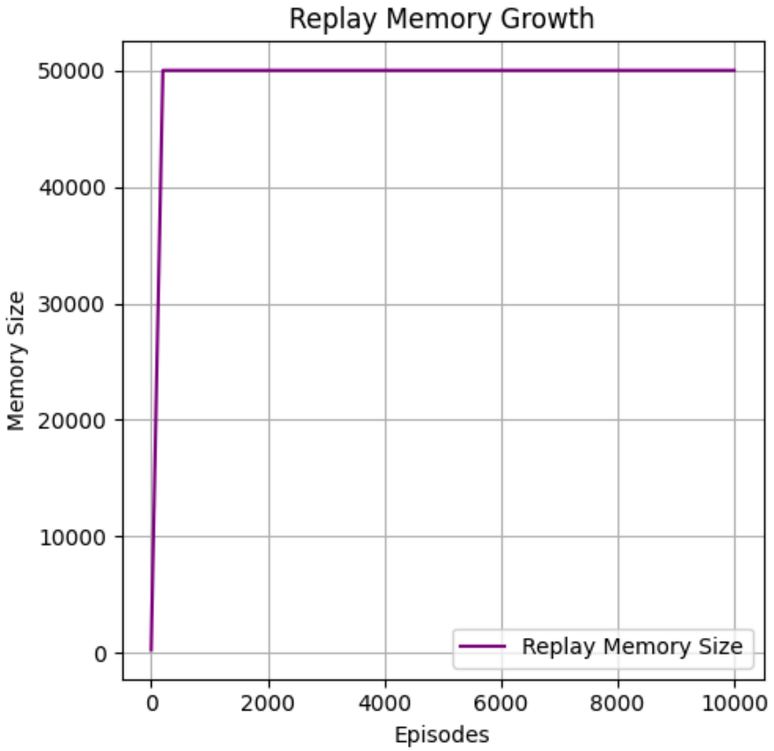
```python
class DQNAgent(SnakeBaseAgent):
    """Deep Q-Network agent for Snake game with dynamic observation handling."""

    def __init__(self, env, learning_rate=0.0005, epsilon=0.95, epsilon_decay=0.9995,
                 epsilon_min=0.05, memory_size=50000, batch_size=64, target_update_freq=500):
        """Initialize the DQN agent with improved hyperparameters."""
        super().__init__(env)

        # Initialize all required attributes
        self.learning_rate = learning_rate
        self.epsilon = epsilon
        self.epsilon_decay = epsilon_decay
        self.epsilon_min = epsilon_min
        self.memory_size = memory_size
        self.batch_size = batch_size
        self.target_update_freq = target_update_freq

        # Experience replay buffer
        self.memory = deque(maxlen=memory_size)

        # Networks - will be initialized on first state
        self.q_network = None
        self.target_network = None
        self.optimizer = None

        # Current network input shape (will be set dynamically)
        self.current_map_shape = None

        # Training variables
        self.warmup_steps = 1000  # Steps before training starts
        self.step_count = 0  # Initialize step counter

        # Initialize tracking attributes that might be expected by base class
        self.episode_count = 0
        self.scores = []
        self.training_losses = []

        # Set action size - assuming 4 actions for snake (up, down, left, right)
        self.action_size = 4
```

ANALYSIS

Performance and results

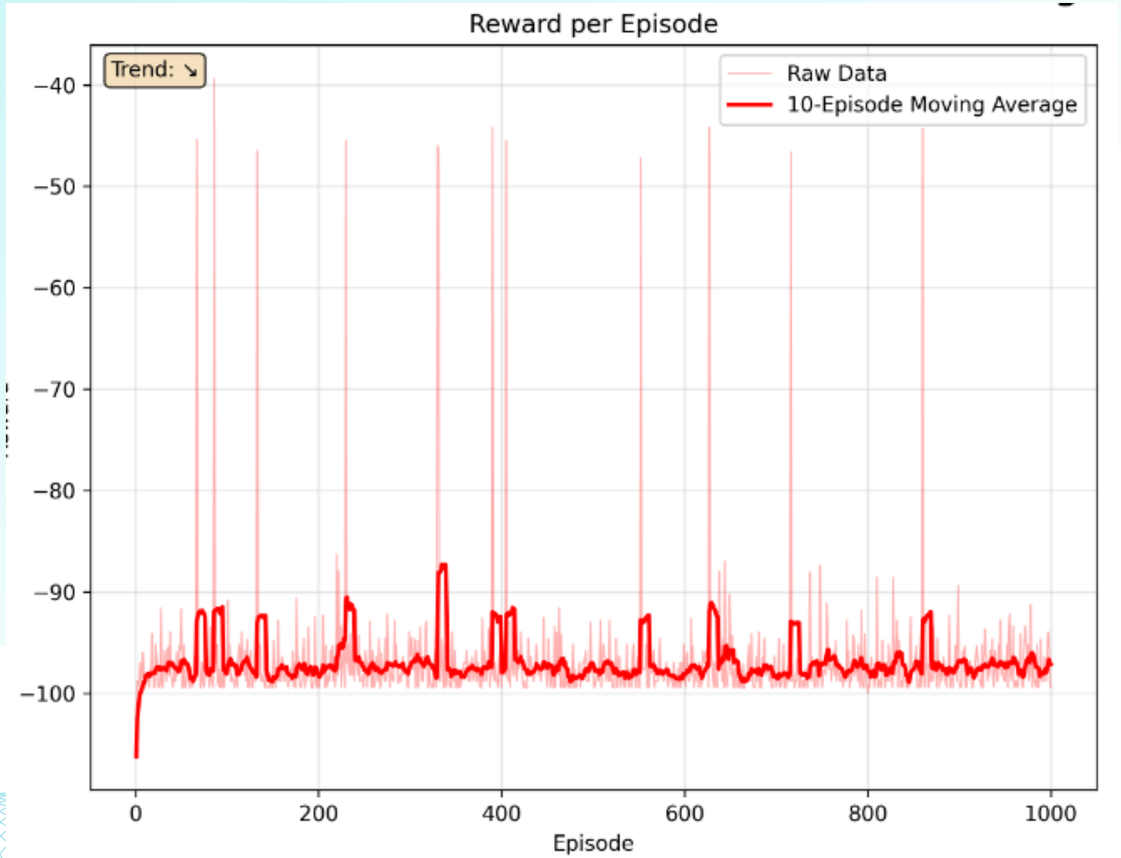


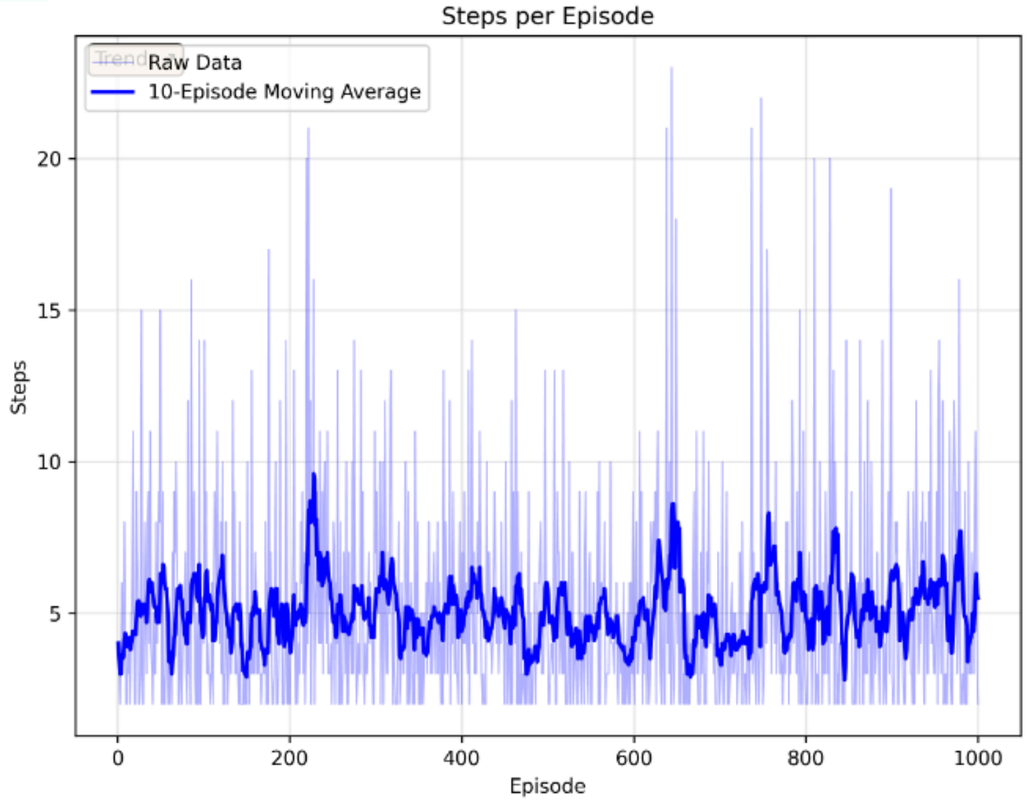




ANALYSIS

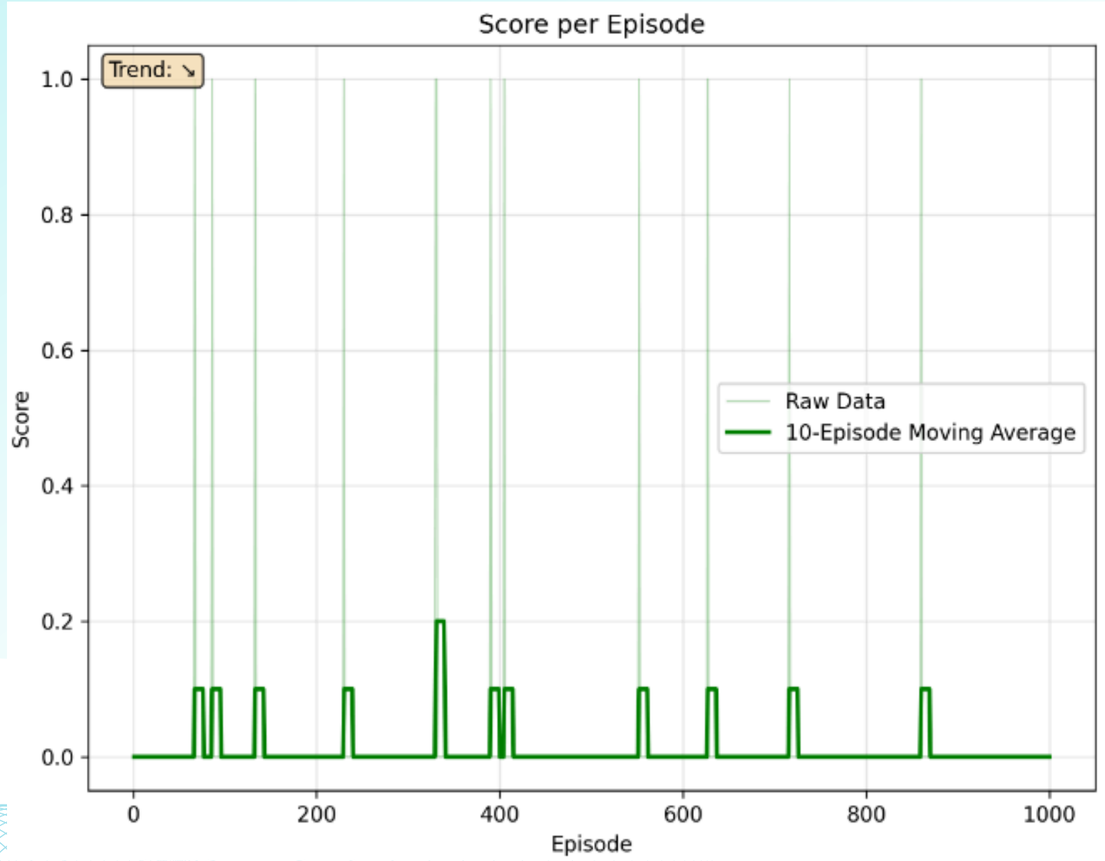
Learning progresion

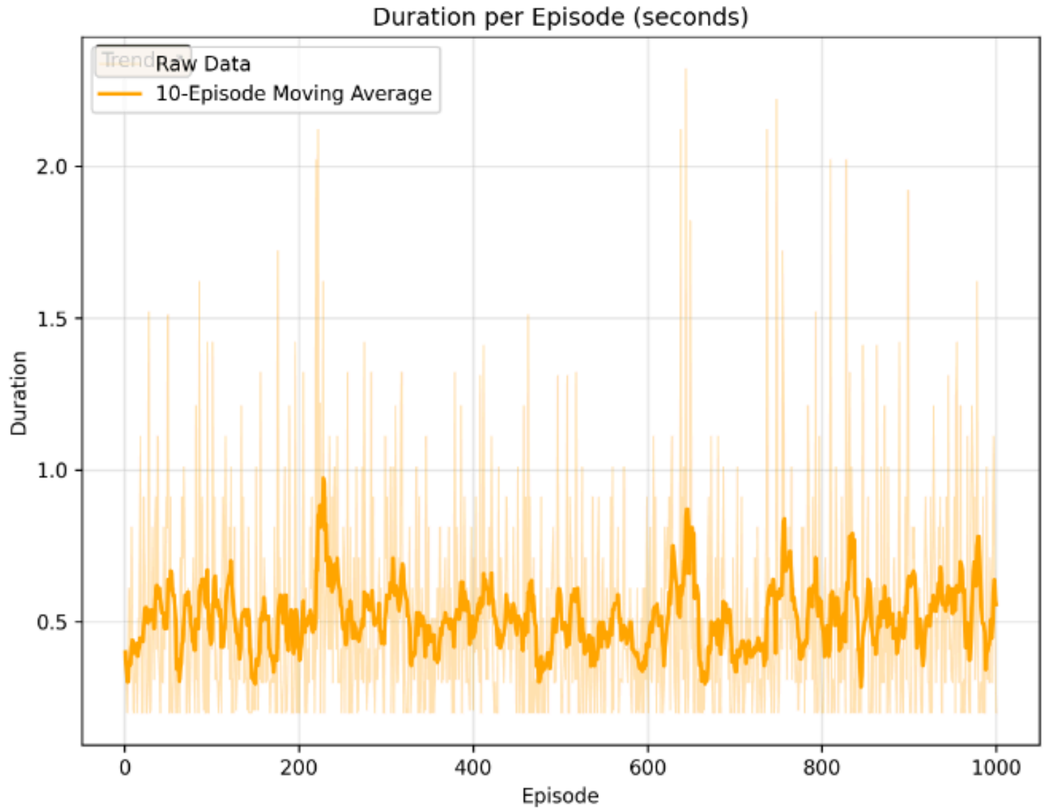




ANALYSIS

Learning progresion





IMPLEMENTATION

Challenges

Student/Server synchronization:

asyncio

```python
class SnakeGame:
    """Class to manage the Snake game state and communication."""

    def __init__(self, logger=None):
        self.first_state = None
        self.message_queue = asyncio.Queue()
        self.listener_task = None
        self.first_state_received = asyncio.Event()
        self.logger = logger or logging.getLogger('SnakeGame')
```

Logging:

```
- SnakeGame - INFO - Logging initialized. Log file: logs/snake_training_20250528_120323.log
- SnakeGame - INFO - Starting training mode...
- SnakeGame - INFO - Server: localhost:8000, Agent: daniel, Episode limit: 100
- SnakeGame - INFO - Pre-trained policy will be loaded: policy/dqn_agent_final_200
- SnakeGame - INFO - Environment initialized
- SnakeGame - INFO - DQN Agent initialized
- SnakeGame - INFO - Pre-trained policy loaded from policy/dqn_agent_final_200
- SnakeGame - INFO - Starting from episode 200, new limit: 300
- SnakeGame - INFO - Starting Episode 201
- SnakeGame - INFO - Episode 201 started successfully
- SnakeGame - INFO - Step 200, Training Loss: 0.0294
- SnakeGame - INFO - Step 400, Training Loss: 0.0240
- SnakeGame - INFO - Episode 201 - Game Over! Final score: 1
- SnakeGame - INFO - Episode 201 completed - Reward: -25.35, Steps: 453, Score: 1, Duration: 45.93s
- SnakeGame - INFO - Starting Episode 202
- SnakeGame - INFO - Episode 202 started successfully
- SnakeGame - INFO - Episode 202 - Game Over! Final score: 1
- SnakeGame - INFO - Episode 202 completed - Reward: -5.60, Steps: 132, Score: 1, Duration: 13.46s
- SnakeGame - INFO - Starting Episode 203
- SnakeGame - INFO - Episode 203 started successfully
- SnakeGame - INFO - Step 600, Training Loss: 0.0117
- SnakeGame - INFO - Step 800, Training Loss: 0.0400
- SnakeGame - INFO - Episode 203 - Game Over! Final score: 2
- SnakeGame - INFO - Episode 203 completed - Reward: -3.20, Steps: 218, Score: 2, Duration: 22.04s
- SnakeGame - INFO - Starting Episode 204
- SnakeGame - INFO - Episode 204 started successfully
- SnakeGame - INFO - Episode 204 - Game Over! Final score: 0
```

IMPLEMENTATION

Challenges

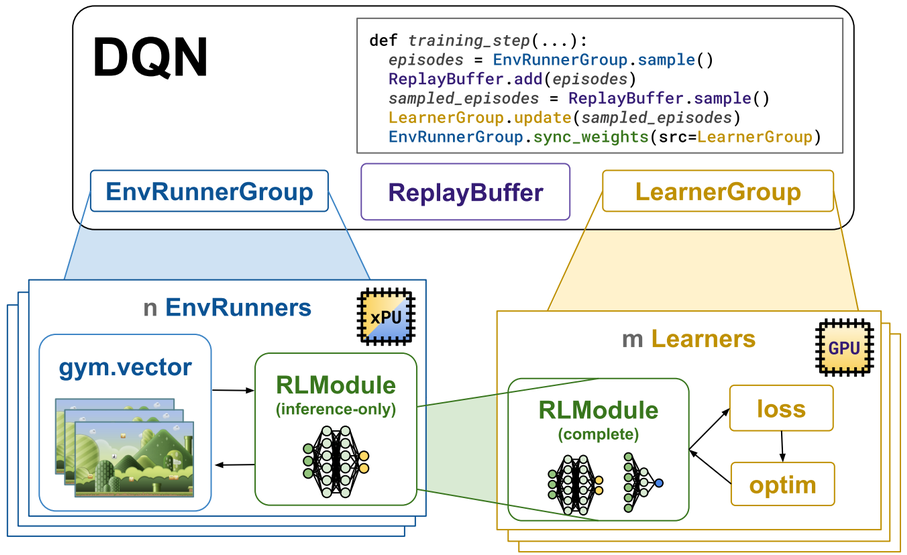
Episode duration:

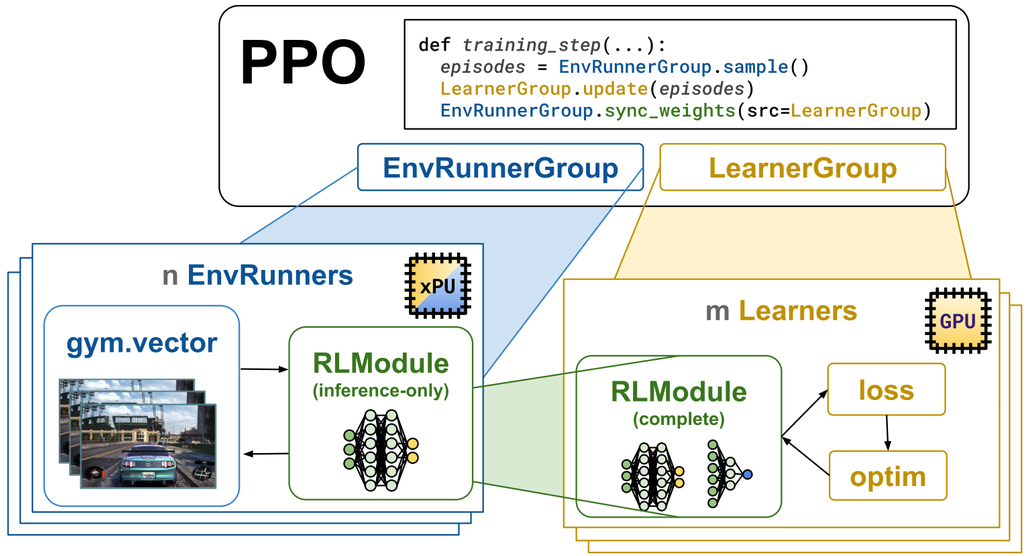
```
7 - SnakeGame - INFO - Episode 1986 started successfully
5 - SnakeGame - INFO - Episode 1986 - Game Over! Final score: 0
5 - SnakeGame - INFO - Episode 1986 completed - Reward: -47.35, Steps: 549, Score: 0, Duration: 55.43s
6 - SnakeGame - INFO - Episode 1987 started successfully
4 - SnakeGame - INFO - Episode 1987 - Game Over! Final score: 0
5 - SnakeGame - INFO - Episode 1987 completed - Reward: -9.50, Steps: 39, Score: 0, Duration: 3.94s
2 - SnakeGame - INFO - Episode 1988 started successfully
9 - SnakeGame - INFO - Episode 1988 - Game Over! Final score: 3
9 - SnakeGame - INFO - Episode 1988 completed - Reward: -10.10, Steps: 658, Score: 3, Duration: 66.41s
2 - SnakeGame - INFO - Episode 1989 started successfully
3 - SnakeGame - INFO - Episode 1989 - Game Over! Final score: 0
3 - SnakeGame - INFO - Episode 1989 completed - Reward: -11.25, Steps: 22, Score: 0, Duration: 2.22s
3 - SnakeGame - INFO - Episode 1990 started successfully
4 - SnakeGame - INFO - Episode 1990 - Game Over! Final score: 0
4 - SnakeGame - INFO - Episode 1990 completed - Reward: -10.75, Steps: 31, Score: 0, Duration: 3.13s
8 - SnakeGame - INFO - Episode 1991 started successfully
7 - SnakeGame - INFO - Episode 1991 - Game Over! Final score: 0
7 - SnakeGame - INFO - Episode 1991 completed - Reward: -21.60, Steps: 256, Score: 0, Duration: 25.87s
```

IMPLEMENTATION

Optimization Attempts

- Reformulated observation_spec;
- Improve reward function;
- Tested agent using PPO.





IMPLEMENTATION

Future work

Implement parallelized training

Try other types of agent alghoritms

	DQN	DDPG	C51	TD3	PPO	REINFORCE
Discrete	✓		✓		✓	✓
Continuous		✓		✓	✓	

THANKS FOR LISTENING

Do you have any question?