AFRICAN LEADERSHIP UNIVERSITY
[BSE]
[ML TECHNIQUES II]
[SUMMATIVE ASSIGNMENT]

Reinforcement Learning Summative Assignment Report

Student Name: Josiane Ishimwe

Video Recording:
https://docs.google.com/document/d/1Fli-sQ-m-vozIKXEKDls4BHIMEFLYi7O8SJGh8Rogio/edit?usp=sharing
GitHub Repository: https://github.com/Joh-Ishimwe/josiane_ishimwe_rl_summative

1. Project Overview

This project develops Agrika, a smart tractor monitoring system designed for smallholder farmers in Rwanda who use four-wheel tractors. The system addresses unexpected equipment breakdowns, poor maintenance practices, and limited access to technical support by implementing a reinforcement learning (RL) environment to optimise tractor fleet management. Using a custom Gym environment, four RL algorithms (DQN, REINFORCE, PPO, A2C) are trained to minimise downtime and maintenance costs while maximising productivity. The approach simulates a 7-day week, balancing tractor operations, maintenance, and rest under varying weather and demand conditions.

2. Environment Description

2.1 Agent(s)

The agent represents a fleet manager controlling three Four-wheel tractors. Each tractor has attributes: hours used, condition (0-100), and days since last maintenance. The agent's goal is to optimise tractor usage to maximise productivity while minimising breakdowns and costs, constrained by weather (Rainy/Dry) and demand (Low/Moderate/High). And one limitation of the agent is its potential lack of adaptability to sudden, unmodeled environmental changes, such as extreme weather events or unexpected equipment failures.

2.2 Action Space

The action space is discrete with 27 actions (3^3 combinations), representing decisions for three tractors: operate (work in the field), rest (park in the rest zone), or maintain (undergo maintenance). Each action is a tuple of three integers, one per tractor, ensuring comprehensive control over the fleet.

2.3 State Space

The state space is a 13-dimensional vector capturing:

- Tractor states (3 tractors × 3 attributes: hours used, condition, days since maintenance) = 9 dimensions
- Weather (rainy/dry) for today and tomorrow = 2 dimensions
- Current day in the 7-day season = 1 dimension
- Working demand (high/moderate/low) = 1 dimension

Observations are encoded as a NumPy array of floats, providing the agent with partial but critical environmental information.

2.4 Reward Structure

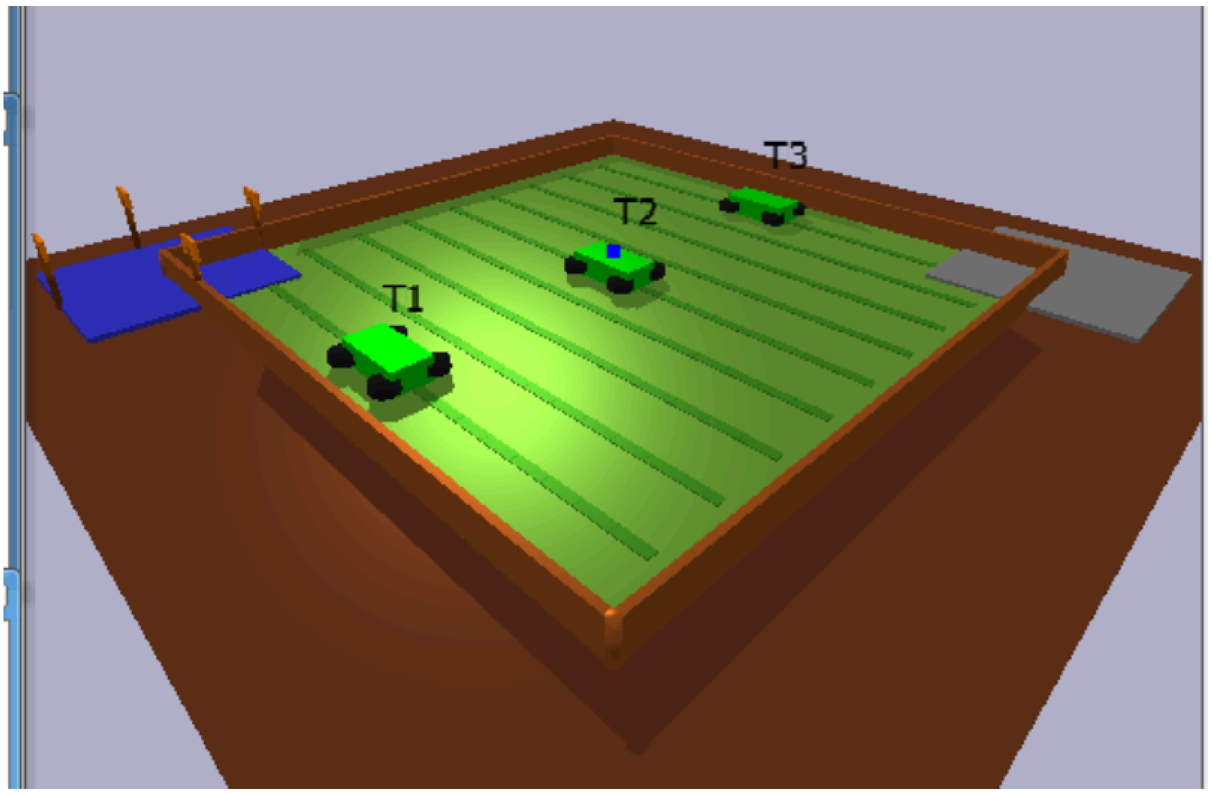
The reward function balances productivity, maintenance costs, and breakdowns:

- **Productivity Reward**: Positive reward proportional to field work completed (higher in good weather, lower in rainy conditions).
- **Maintenance Cost Penalty**: Negative reward for maintenance actions (cost = -0.5 per maintenance).
- **Breakdown Penalty**: Negative reward if a tractor breaks down, based on probability (base 0.005, modified by hours used, condition, and days since maintenance).
- **Reward Formula**:
  {Reward} = {daily_productivity} - {daily_maintenance_cost} - {daily_breakdown_penalty}] The agent maximises cumulative rewards over a 7-day episode.

2.5 Environment Visualization

The environment is visualized using PyBullet, showing a 3D farm with tractors moving in straight-line field work. Visual elements include:

- **Tractors**: Labeled T1, T2, T3, colored by condition (green=healthy, yellow=warning, red=critical).
- **Weather**: Sun particles (dry) or raindrops (rainy).
- **Field**: Green for worked areas, brown for unworked, with progress markers.
- **Zones**: Grey rest park, blue maintenance zone, brown fence boundaries.



3. Implemented Methods

3.1 Deep Q-Network (DQN)

The DQN uses a neural network with three fully connected layers (input: 13, hidden: 128, output: 27) to approximate Q-values. Key features include:

- **Experience Replay**: Buffer size of 10,000 to store state-action-reward transitions.
- **Target Network**: Updated every 100 steps to stabilize training.
- **Epsilon-Greedy Exploration**: Epsilon decays from 1.0 to 0.01 over 10,000 steps.

3.2 Policy Gradient Method ([REINFORCE/PPO/A2C])

- **REINFORCE**: Uses a policy network (256 neurons, ReLU, dropout 0.2) to output action probabilities. Incorporates entropy regularization (coefficient 0.01) for exploration and a curriculum with increasing weather changes and breakdown multipliers.
- **PPO**: Employs a clipped objective function with a policy network (deep_large architecture) and value network. Uses vectorized environments (n_envs=4) for parallel training.

- **A2C**: Combines policy and value networks (both 256 neurons), leveraging advantage estimation to reduce variance. Trained with optimized hyperparameters.

5. Hyperparameter Optimization

5.1 DQN Hyperparameters

Hyperparameter	Optimal Value	Summary
Learning Rate	0.0005	Balanced convergence speed and stability; higher values caused divergence.
Gamma (Discount Factor)	0.99	Prioritized long-term rewards, suitable for 7-day episodes.
Replay Buffer Size	10 000	Sufficient for storing diverse transitions, avoiding memory issues.
Batch Size	64	Improved gradient updates; larger sizes slowed training.
Exploration Strategy	Epsilon-Greedy	Epsilon decay from 1.0 to 0.01 ensured exploration without overfitting.

5.2 Tables for PPO

Hyperparameter	Optimal Value	Summary

Learning Rate	0.0003	Balanced policy updates; higher rates caused instability.
Gamma (Discount Factor)	0.99	Effective for long-term reward optimization in short episodes.
Clip Range	0.2	Clipped objective stabilized training, avoiding large policy updates.
Batch size	128	Efficient for vectorized environments, improving sample efficiency.

5.3 A2C Hyperparameters

Hyperparameter	Optimal Value	Summary
Learning Rate	0.0003	Ensured stable policy and value updates; higher rates increased variance.
Gamma (Discount Factor)	0.99	Consistent with other methods, prioritized long-term rewards.
N Step	2048	Long rollouts improved advantage estimation, enhancing policy updates.

Batch Size	128	Balanced computation and learning stability in vectorized environments.

5.4 Reinforcement Hyperparameters

Hyperparameter	Optimal Value	**Summary** How did the hyperparameters affected the overall performance of your agent. Mention what worked well and what didn't.
Learning Rate	0.0002	Low rate prevented policy oscillation, ensuring stable learning.
Gamma (Discount Factor)	0.99	Encouraged long-term planning for maintenance scheduling.
Entropy Coefficient	0.1	Promoted exploration, preventing premature convergence to suboptimal actions.
Episodes per stage	500	Curriculum learning with 4 stages improved robustness to

		weather/breakdowns.

5.3 Metrics Analysis

Cumulative Reward PPO and A2C outperformed DQN and REINFORCE, achieving higher mean rewards (e.g., PPO: ~240, A2C: ~230) due to stable policy updates. REINFORCE showed higher variance, while DQN converged slower.
Training Stability DQN loss decreased steadily but fluctuated due to exploration. PPO and A2C

maintained stable policy entropy (~2.0–2.7), indicating balanced exploration. REINFORCE entropy increased, reflecting curriculum learning.
Episodes to Convergence PPO and A2C converged within ~1,000 episodes, DQN required ~1,500, and REINFORCE took ~2,000 due to curriculum stages. Convergence was measured as stable mean rewards over 30 evaluation episodes.
Generalization All models were tested on unseen weather and demand patterns. PPO generalized best, maintaining high rewards (~220). A2C followed (~210), while DQN and REINFORCE struggled with high breakdown scenarios, dropping to ~150 and ~170, respectively.

6. Conclusion and Discussion

PPO performed best in the Agrika environment due to its clipped objective, balancing exploration and stability, making it ideal for the dynamic tractor management task. A2C was competitive but slightly less robust in unseen scenarios. DQN struggled with the high-dimensional action space, and REINFORCE's variance limited its reliability. Strengths of PPO/A2C include sample efficiency and adaptability, while DQN's exploration was less effective, and REINFORCE required careful curriculum design. Future improvements could include multi-agent RL for cooperative tractor scheduling and real-world data integration to enhance model robustness.