



Smart_IrriGO



Intelligent Irrigation System



Ming Zhe Yeoh, Siew Mooi Lim, Shu Han Wong
Faculty of Computing and Information Technology,
TAR UMT, Malaysia

Overview & Objectives

Big Picture:

1. Address inefficiencies in agricultural irrigation systems.
2. Reduce water usage while maximizing crop yield.

Main Objectives:

1. Adapt irrigation decisions to real-time environmental changes
2. Balance short-term crop health with long-term yield optimization
3. Demonstrate gains over fixed-schedule methods



1.0 Problem Definition

Problem Framing and Topic Description



Tackles

Tackles inefficiencies in traditional irrigation systems.



Controls

Controls irrigation based on environmental conditions (e.g., soil moisture, temperature)



Improve

Aims to improve resource efficiency and overall productivity



Design

Design an Reinforcement Learning (RL) agent to decide irrigation timing and quantity. Furthermore, use dynamic inputs to maximize crop yield and minimize water consumption

Reinforcement Learning Objectives

1. Develop a RL agent that learns and adapts irrigation in real-time
2. Balance short term and long-term goals
 - a. **Short-term goal:**
 - i. Maintain soil moisture within the optimal range for current growth
 - ii. Respond to daily weather
 - iii. Conserve water
 - iv. Minimize daily evapotranspiration loss
 - v. Get positive reward for good decision-making
 - b. **Long-term goal:**
 - i. Accumulate consistent moisture control across critical growth stages
 - ii. Maintain high yield potential
 - iii. Stay within budgeted water usage
 - iv. Get a large final season bonus based on yield potential at the end
3. Compare performance against conventional irrigation methods to demonstrate improved water efficiency and crop production

Expected Outcome

1. Higher cumulative rewards compared to fixed schedule methods
2. Demonstrated reduction in water usage
3. Increased corn yield

Challenges



**Handling Sparse
Rewards (yield
measured at
season's end)**

**Balancing
Exploration
vs.
Exploitation**

**Managing
Long-Term
Decision-Making
Effects**

Evaluation Metrics

| | |
|--|--|
| Total Rewards ($J(\pi)$) | Sum of rewards over a growing season, reflecting yield and water efficiency. |
| Water Usage Efficiency | Water used per unit yield: $W \cup E=W/Y$. |

Complexity of the Environment

- **Environment Type:**
 - **A simulated cornfield environment**
- **Complexity:**
 - **Dynamic and partially observable**
 - **With continuous state variables (e.g., soil moisture levels) and continuous action spaces (e.g., irrigation amounts)**
- **Computation Resource:**
 - **Training may require moderate resource if using deep RL algorithms like DQN or PPO**

Theoretical Background

Reinforcement Learning Agent Components

| | |
|-------------------------|---|
| Policy | Defines the agent's strategy for selecting irrigation actions based on observed states. |
| Value Function | Estimates the expected long-term reward for a given state or action, guiding the agent's decisions. |
| Environment Interaction | The agent observes the current state (e.g., soil moisture), takes an action (e.g., irrigate), receives a reward (e.g., based on crop health), and transitions to a new state. |

Theoretical Background

Importance in Decision Making:

- Each decision (whether to irrigate today, how much water to use, or when to wait) has effect on the harvest after a few months
- RL agent experimenting, figuring out how today's actions tie into tomorrow's outcomes, and adjusting along the way
- Traditional rule-based systems irrigates every three days, regardless of the current environmental condition. This may cause the soil to be soaked or dehydrated.

Theoretical Background

Importance in Decision Making:

| Method | Adaptability | Water Efficiency | Performance |
|----------------|--------------|------------------|-------------|
| Fixed Schedule | No | Low | Suboptimal |
| RL-based | Yes | High | Optimized |

Comparison of Algorithms

| | Model-Free | Model-Based |
|-------------|--|--|
| Concept | Learn directly from trial-and-error interactions without modeling the environment. | Build an internal model of the environment (e.g., predicting crop response to irrigation) and use it for planning. |
| Example | Q-Learning, DQN, PPO | Dyna-Q |
| Application | Complex, unknown dynamics environments | Predictable environment |

Comparison of Algorithms

| Algorithms | Type | Pros | Cons |
|-----------------------|-------------|--------------------------------------|--|
| Q-Learning | Model-Free | Simple, well-understood | Struggles with continuous state spaces |
| DQN | Model-Free | Scales well with deep learning | Requires significant computational power |
| PPO | Model-Free | Stable, efficient for large problems | Requires hyperparameter tuning |
| Model-Based RL | Model-Based | Efficient with small datasets | Needs an accurate environment model |

2.0 Environment and Agent Definition

Defining the Action and State Space

| | |
|-------------------------|--|
| Environment | A simulated cornfield that replicates real-world agricultural conditions and allows the RL agent to control irrigation |
| State Space | Continuous variables capturing the environment and crop status |
| Action Space | Options for irrigation decision |
| Reward Functions | Aim to maximize the corn yields while reducing unnecessary irrigation and wasting water |

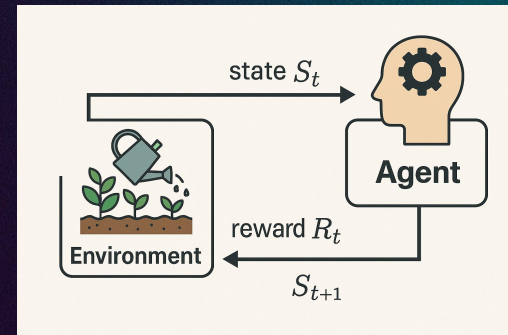
Agent and Learning Objectives

- **Agent:** An RL decision-maker that learns an irrigation policy
- **Learning Objective:**
 - Maximise cumulative reward over a growing season
 - Adapt irrigation to maintain crop health while conserving water, balancing immediate needs with long-term goals

Theoretical Background

Theoretical Background:

- **Markov Decision Process (MDP):**
 - **States:** Environmental and crop status
 - **Actions:** Irrigation choices
 - **Rewards:** Based on yield and water efficiency
 - **Transitions:** Impact of irrigation on future states
 - **Discount Factor (γ):** Balances immediate vs. future rewards
- **Agent's Use of MDP:**
 - Learning a policy that maximizes expected cumulative rewards by exploring state-action pairs and refining its strategy over time



3.0 Environment Creation

Defining the Environment

Environment:

- Custom OpenAI-Gym environment SimpleCornIrrigationEnv → 120-day corn-growth simulation
- Stochastic weather generator per region (Arid / Temperate / Tropical): daily temperature, rainfall, next-day rain chance
- Dynamic soil-moisture balance: irrigation + effective rainfall – evapotranspiration (ET) updates each step
- Five physiological growth stages (Seedling → Maturity) with stage-specific optimal-moisture targets & reward weights
- Difficulty modes (Easy / Normal / Hard) scale rainfall frequency and variance to stress-test policies

State, action and rewards definition

Continuous Variables:

| Variable | Encoding in State | Purpose / Note |
|---------------------------|---|---|
| Soil Moisture | Integer 0 – 100 % | Core hydrological signal, drives moisture-error reward and soil-water balance updates. |
| Temperature | Current temperature (°C) | Determines daily evapotranspiration, influencing water-balance and conservation rewards. |
| Recent Rainfall | Rainfall amount on the current day (mm) | Adds exogenous water input to soil, used to credit or penalize irrigation decisions for conservation. |
| Next Day Rain Probability | Probability of rain on the next day (%) | Lets the agent anticipate likely rainfall and postpone irrigation when precipitation is expected. |

State, action and rewards definition

Discrete Variables:

| Variable | Encoding in State | Purpose / Note |
|------------------|---|---|
| Day-in-Season | Integer 0 – 119 (120 = terminal) | Supplies temporal context for stage transitions and season termination. |
| Growth Stage | Index 0 – 4 → Seedling (0), Jointing (1), Staminate (2), Filling (3), Maturity (4) | Drives stage-specific optimal-moisture targets and reward scaling. |
| Region Type | One-hot vector (Arid, Temperate, Tropical) or single index outside the core state (fixed per episode) | Determines baseline weather pattern and irrigation dose map. Because it never changes mid-episode, we typically store it once in env rather than in every state. |
| Difficulty Level | One-hot (Easy, Normal, Hard) or single index outside the core state | Scales rainfall frequency/variance; also fixed at reset(), so it need not be included in every observation unless you train a single policy shared by all difficulties. |

State, action and rewards definition

Action Space

Irrigation actions are represented categorically with four discrete levels:

| Action Index | Description | Irrigation Amount (mm) - Arid | Irrigation Amount (mm) - Temperate | Irrigation Amount (mm) - Tropical |
|--------------|-------------------|----------------------------------|---------------------------------------|--------------------------------------|
| 0 | No Irrigation | 0 | 0 | 0 |
| 1 | Light Irrigation | 8 | 5 | 3 |
| 2 | Medium Irrigation | 16 | 10 | 6 |
| 3 | Heavy Irrigation | 24 | 15 | 12 |

Reward Function of Moisture Management

Calculate reward for maintaining soil moisture close to optimal value

| Conditions | Reward Value | Scaling Factor |
|--------------------------------|--------------|--|
| Absolute moisture error < 5% | +10 | Multiplied by stage_importance (1.0–1.5) |
| Absolute moisture error 5–10% | +5 | Multiplied by stage_importance (1.0–1.5) |
| Absolute moisture error 10–15% | 0 | Multiplied by stage_importance (1.0–1.5) |
| Absolute moisture error 15–20% | -5 | Multiplied by stage_importance (1.0–1.5) |
| Absolute moisture error > 20% | -10 | Multiplied by stage_importance (1.0–1.5) |

Reward Function of Water Conservation

Calculate reward for avoiding irrigation when rainfall is sufficient

| Conditions | Reward Value | Scaling Factor |
|--|--------------|--|
| Rainfall $> 1.2 \times$ daily ET, no irrigation | +7 | Multiplied by stage_importance (1.0–1.5) |
| Rainfall $> 1.2 \times$ daily ET, irrigation applied | -7 | Multiplied by stage_importance (1.0–1.5) |
| Rainfall $>$ daily ET, no irrigation | +5 | Multiplied by stage_importance (1.0–1.5) |
| Rainfall $>$ daily ET, irrigation applied | -5 | Multiplied by stage_importance (1.0–1.5) |
| Rainfall $> 0.7 \times$ daily ET, minimal irrigation ($\leq 5\text{mm}$) | +3 | Multiplied by stage_importance (1.0–1.5) |
| Rainfall $> 0.7 \times$ daily ET, over-irrigation ($> 5\text{mm}$) | -2 | Multiplied by stage_importance (1.0–1.5) |
| Rainfall $\leq 0.7 \times$ daily ET (normal irrigation needed) | 0 | Multiplied by stage_importance (1.0–1.5) |

Reward Function of Yield Impact

Calculate reward based on change in yield potential due to moisture management

| Conditions | Reward Value | Scaling Factor |
|-----------------------------------|--|---|
| Moisture error < 7% (excellent) | $\text{yield_potential} \pm 0.01 \times \text{stage_importance}$ | $\text{Reward} = 10 \times (\text{new_yield} - \text{old_yield})$ |
| Moisture error 7–15% (acceptable) | $\text{yield_potential} \pm 0.003$ | $\text{Reward} = 10 \times (\text{new_yield} - \text{old_yield})$ |
| Moisture error > 15% (poor) | $\text{yield_potential} \pm 0.02 \times (\text{moisture_error} - 15) / 10 \times \text{stage_importance}$ | $\text{Reward} = 10 \times (\text{new_yield} - \text{old_yield})$ |

Reward Function of Base Bonus

Calculate constant bonus added to total reward

| Conditions | Reward Value | Scaling Factor |
|----------------|--------------|----------------|
| Always applied | +2.0 | None |

Reward Function of Final Yield Bonus

Calculate bonus added at season end based on final yield potential

| Conditions | Reward Value | Scaling Factor |
|--|---|----------------|
| Applied when the season ends (done = True) | $(\text{yield_potential} - 1.0) \times 50$ | None |

Total Reward Function

$$R = (\text{Moisture Reward} + \text{Conservation Reward}) \times \text{Stage_Importance} + \text{Yield_Reward} + 2.0$$

- Moisture management: ± 10 scaled by stage-importance (1.0–1.5)
- Water conservation: +7 to -7 depending on rainfall vs ET and irrigation choice
- Yield impact: $\Delta \text{Yield} \times 10$ (yield potential bounded 0.1–2.0)
- Base bonus +2 each step; final season bonus $50 \times (\text{Yield} - 1.0)$

Reset & Step Mechanisms

Reset Method

- Clears episode state randomises **current step** within the first quarter of the season to diversify starting conditions.
- Sets **current day = 0**, **yield potential = 1.0**, **total water used = 0**.
- Assigns region-specific starting soil-moisture range:
 - *Arid 30-50 %* • *Temperate 40-60 %* • *Tropical 50-70 %*
- Empties logs (water use, rainfall, soil-moisture, yield, reward).
- Returns initial observation vector via `_get_observation()`.

Step Method

1. **Reward** computed by reward function
2. **Advance time**
 - a. `current_day += 1`
 - b. `current_step = (current_step + 1) % season_length`
3. **Check termination:**
 - a. `done = current_day ≥ season_length (120 days)`.
4. **Generate next observation** with updated weather & soil state.
5. **If done**
 - `Add final bonus = 50 × (yield_potential - 1.0)`.
 - `Return info dict: {final_yield, total_water_used, water_efficiency}`.
6. **Return tuple:** `(observation, reward, done, False, info)`.

Support for Episodic Reinforcement Learning

Episodic RL Support:

- Designed for episodes that span multiple steps (up to 100)
- Facilitates continuous learning through trial and error by resetting after each episode, allowing the RL agent to improve over time.

4.0 Reinforcement Learning Algorithm Selection

RL Algorithms & Model Selection

Chosen Algorithm:

- **DQN (Deep Q-Networks)**
- **Proximal Policy Optimization (PPO)**
- **Advantage Actor-Critic (A2C)**
- **Dyna-Q**

Model-free vs Model-based Decision

Model-based Decision is better in this project.

- The agent learns direct the interaction with the simulated environment.
- This is suitable given the complexity of agricultural conditions (e.g., varying weather and soil types)

Discrete vs. Continuous Action Space

Discrete action space will be used in this project

- This aligns with model-free capabilities
- It simplifies the learning process

Theoretical Background

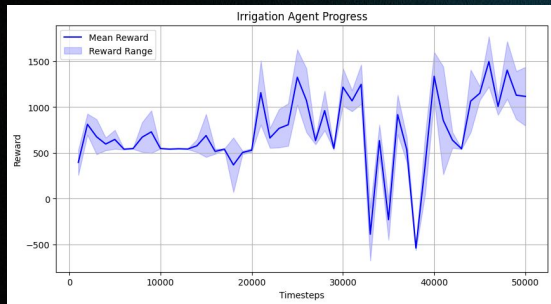
Model-Free vs. Model-Based Agents:

| Model-Free Agents | Model-Based Agents |
|---|--|
| Do not need explicit model of the environment | Planning and simulation are made possible by learning of a model to forecast future states or rewards. |
| Agents learn directly from experience, updating policies or value functions based on observed rewards and state transitions | Agents execute look-ahead planning and may produce synthetic data from the model |
| Appropriate for issues with high unpredictability, like agricultural systems with erratic weather | Limited in situations where the environment is highly stochastic or uncertain |

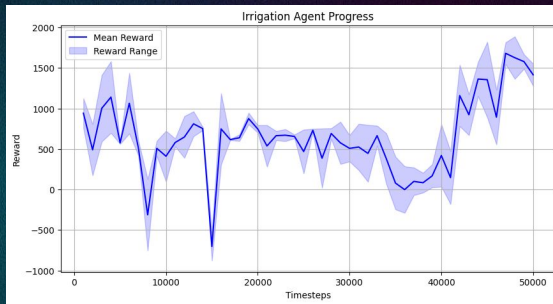
5.0 Reinforcement Learning Algorithm Exploration

Deep Q-Network (DQN)

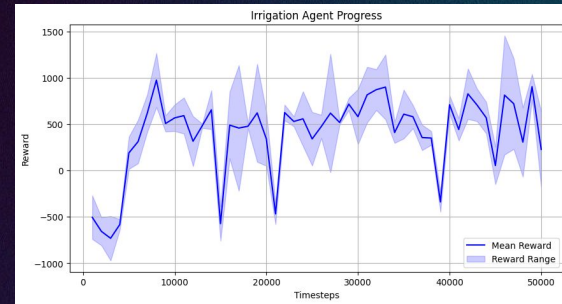
ϵ -greedy exploration



Final Exploration of 0.05







Final Exploration of 0.01



Final Exploration of 0.001

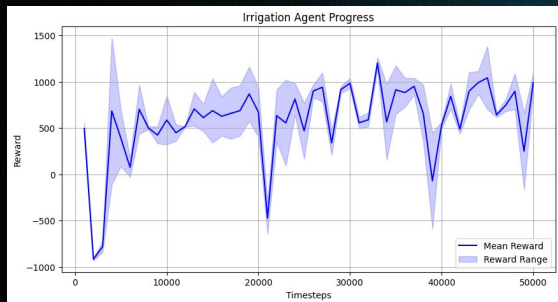
Deep Q-Network (DQN)

ϵ -greedy exploration

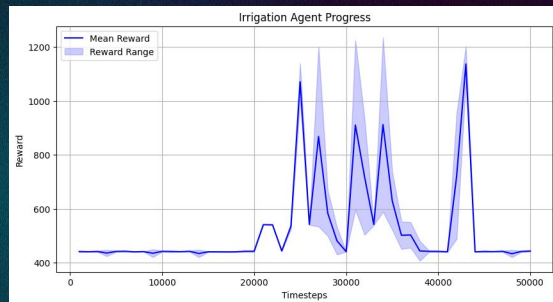
|  Exploration Rate |  Final Reward |  Variation |  Total Steps |
|--|--|---|---|
| 0.05 | 1.12e+03 | ± 319.84 | 50000 |
| 0.01 (Ideal rate!) | 1.42e+03 | ± 136.74 | 50000 |
| 0.001 | 229.42 | ± 408.98 | 50000 |

Deep Q-Network (DQN)

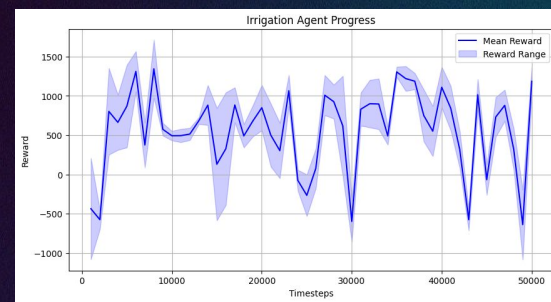
Performance Across different Learning Rate



Learning Rate of 0.01







Learning Rate of 0.0001



Learning Rate of 0.0005

Deep Q-Network (DQN)

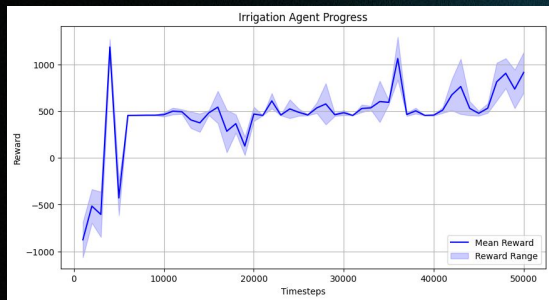
Performance Across different Learning Rate

|  Learning Rate |  Final Reward |  Variation |  Total Steps |
|---|--|--|---|
| 0.01 | 987.76 | ± 319.84 | 50000 |
| 0.0001 | 443.6 | ± 2.26 | 50000 |
| 0.0005 | 1.19e+03 | ± 178.14 | 50000 |

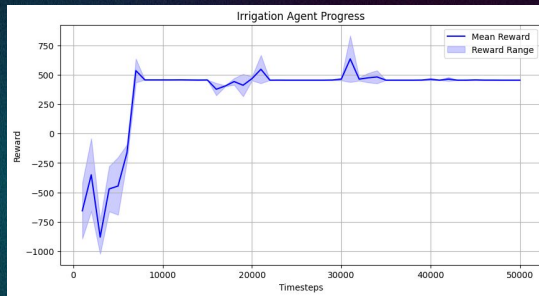
0.0001 yielded the most consistent outcomes but **0.0005** with variance-reduction strategies might produce better trade-offs for maximising reward.

Deep Q-Network (DQN)

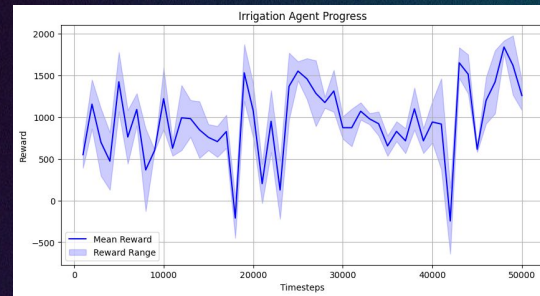
Different randomness test on controlled environment



Easy




Normal



Hard

Deep Q-Network (DQN)

Performance Across different Difficulty

|  Difficulty |  Final Reward |  Variation |  Total Steps |
|--|--|--|---|
| Easy | 913.07 | ± 215.63 | 50000 |
| Normal | 453.80 | ± 0.00 | 50000 |
| Hard | 1.26e+03 | ± 177.27 | 50000 |

Deep Q-Network (DQN)

Strengths:

- The agent shows the best performance in the hard environment, demonstrating its ability to function well in challenging and unpredictable situations.
- It implies that DQN may find high-reward policies even in high-variance, uncertain situations if the proper hyperparameters are used and exploration is done consistently ($\epsilon=0.01$).
- The agent was extremely deterministic under stable settings, as indicated by the zero variation in the typical setting. This could be a sign of dependable policy convergence.

Deep Q-Network (DQN)

Weaknesses:

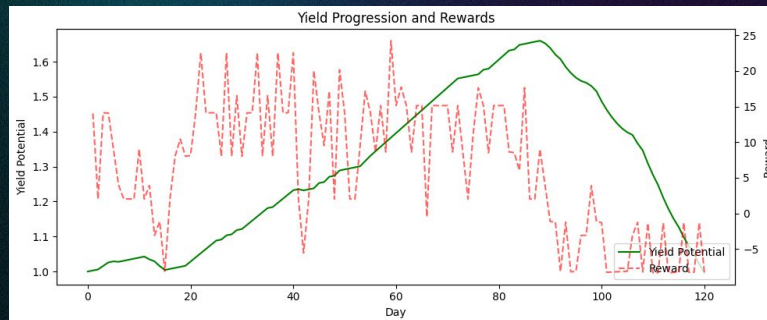
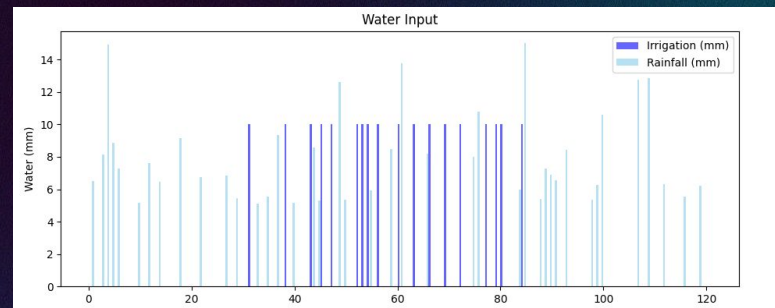
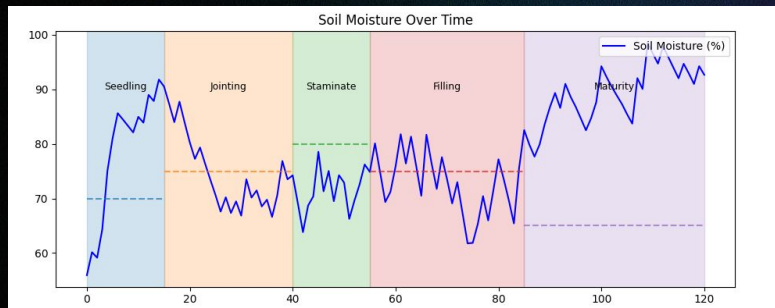
- **Easy environment** - produce a lower payout with a high volatility. This suggests that when the environment is too deterministic, DQN may overfit to spurious patterns, resulting in unstable policies.
- **Normal environment** - obtain the lowest reward. It could indicate a local minimum trap.
- **The core weakness:** inherent reliance on stationarity

Deep Q-Network (DQN)

Areas of improvements:

1. **Adaptive Exploration Strategies** - adjusts the exploration rate based on the agent's uncertainty.
2. **Reward Scaling and Normalization** - using return-based scaling as a normalization method
3. **Recurrent Architectures for Temporal Dependencies** - LSTM layers were incorporated into the Deep Recurrent Q-Network (DRQN)
4. **Policy Gradient Methods in Non-Stationary Environments**

Proximal Policy Optimization (PPO)



Proximal Policy Optimization (PPO)

Season Summary(Region Type = **Temperate**, Difficulty = **Normal**)

- Final Yield: 1.00
- Total Water Used: 180.0 mm
- Total Rainfall: 326.7 mm
- Water Efficiency: 0.553

Proximal Policy Optimization (PPO)

Impact of hyperparameters on performance

| Parameter | Higher value | Lower value |
|-----------------------|--|--|
| Learning rate | <ul style="list-style-type: none">• Allow model to learn faster initially and escape local minima more easily• May cause instability and divergence in complex environments | <ul style="list-style-type: none">• Lead to stable training and reduce the risk of divergence• May cause very slow learning and might get stuck in local optima |
| Clip range | <ul style="list-style-type: none">• Leads to faster learning• Training process might be unstable | <ul style="list-style-type: none">• More stable• Slower convergence and underfitting |
| n step returns | <ul style="list-style-type: none">• Bring more accurate gradient estimation and stable updates• Become slower and consume more memory | <ul style="list-style-type: none">• Lead to faster update and consume less memory• May produce noisier gradient and less stable learning |

Proximal Policy Optimization (PPO)

Impact of hyperparameters on performance

| Parameter | Higher value | Lower value |
|------------|--|---|
| Batch size | <ul style="list-style-type: none">Leads to a more stable learning process and smoother updates | <ul style="list-style-type: none">Cause more noisy updates |
| Gamma | <ul style="list-style-type: none">Cause the agent to be focuses on long-term strategy | <ul style="list-style-type: none">Cause the agent to focus on immediate rewards |
| gae_lambda | <ul style="list-style-type: none">Brings a low bias but high variance result | <ul style="list-style-type: none">Brings a low variance but high bias result |

Proximal Policy Optimization (PPO)

Strengths:

- This algorithm has a stable training process and clipped policy updates that will make it highly effective for learning robust irrigation strategies in environments like SimpleCornIrrigationEnv, where consistent adaptation to noisy sensor data and fluctuating soil moisture is essential.

Proximal Policy Optimization (PPO)

Weaknesses:

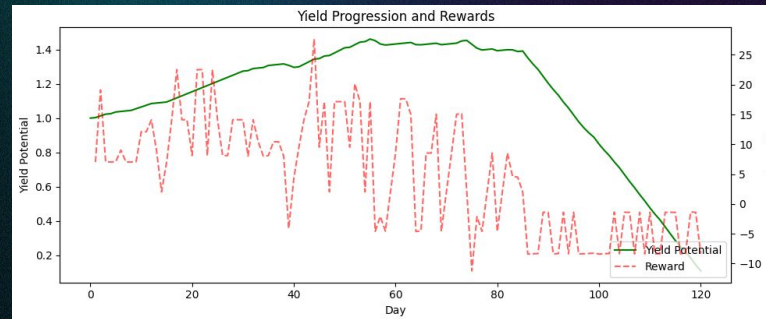
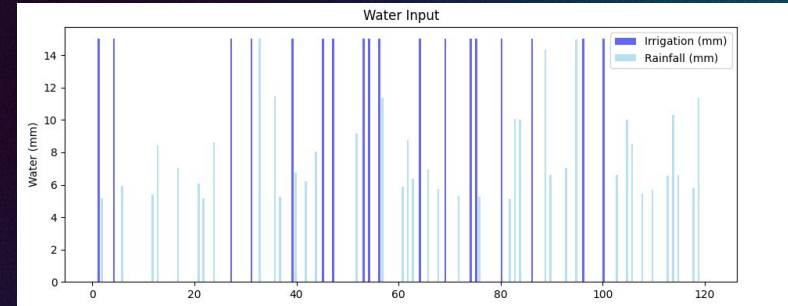
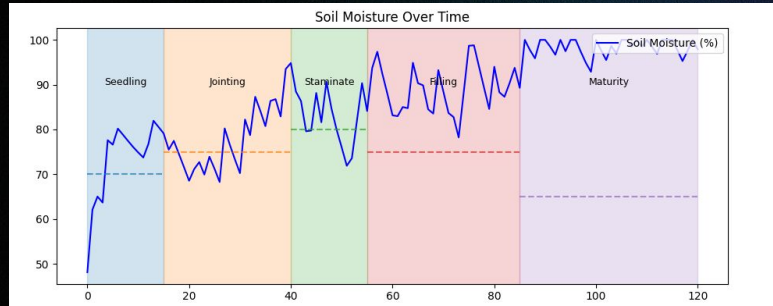
- This algorithm has a on-policy nature and the reliance on fresh data, which can lead to sample inefficiency, which may slow adaptation in environments with stochastic weather and sparse rewards unless balanced with frequent evaluations and careful tuning.

Proximal Policy Optimization (PPO)

Areas of improvements:

1. **Experience Replay for On-Policy Methods** - Integrating experience replay.
2. **Dynamic Advantage Normalization** - Normalizing the advantage estimates dynamically within PPO to stabilize training and mitigate large variance in policy updates
3. **Curriculum Learning for Environment Complexity** - Training models on increasingly complex tasks to improve learning efficiency
4. **Entropy Coefficient Scheduling** - Dynamically adjusting the entropy coefficient during training
5. **Temporal Feature Extraction via Recurrent Layers** - Incorporating LSTM layers within PPO architectures allows the agent to handle partial observability

Actor-Critic (A2C)



Actor-Critic (A2C)

Season Summary (Region Type = **Temperate**, Difficulty = **Normal**)

- Final Yield: 0.11
- Total Water Used: 270.0 mm
- Total Rainfall: 314.2 mm
- Water Efficiency: 0.041

Actor-Critic (A2C)

Impact of hyperparameters on performance

| Parameter | Higher value | Lower value |
|----------------------------|--|--|
| Learning rate | <ul style="list-style-type: none">• Accelerate initial learning• May lead to unstable training | <ul style="list-style-type: none">• Provide finer parameter tuning• Slower training |
| Discount factor | <ul style="list-style-type: none">• Makes the model more focused on long-term rewards• Suitable for tasks that require long-term planning | <ul style="list-style-type: none">• Makes the model more focused on immediate rewards• Suitable for tasks with clear immediate feedback |
| Entropy Coefficient | <ul style="list-style-type: none">• Encourages the model to explore more possible actions | <ul style="list-style-type: none">• Encourages the model to utilize the best known moves |
| n step returns | <ul style="list-style-type: none">• Reduce estimation bias but may increase variance | <ul style="list-style-type: none">• Reduce variance but may increase bias |

Actor-Critic (A2C)

Strengths:

- This algorithm is able to handle continuous observations and its stable training make it well-suited for learning irrigation policies in SimpleCornIrrigationEnv, where precise state tracking (e.g., soil moisture, weather) is critical.

Actor-Critic (A2C)

Weaknesses:

- The sample inefficiency and exploration challenges may limit A2C's ability to quickly adapt to the environment's stochastic weather and sparse rewards, potentially requiring extensive training or careful hyperparameter tuning.

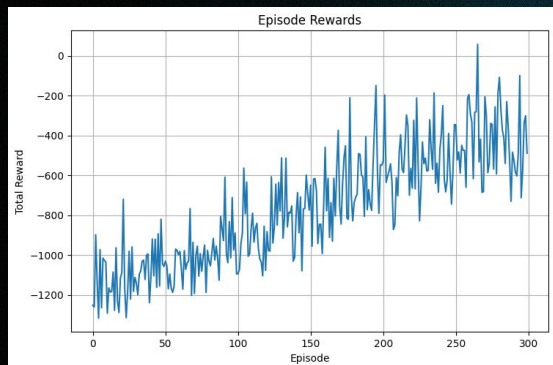
Actor-Critic (A2C)

Areas of improvement:

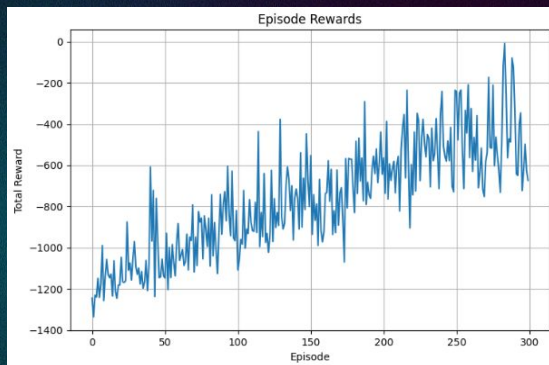
1. **Sample efficiency** - The model required significant training time and episodes to reach its performance level. Introducing experience replay mechanisms could help reuse valuable past experiences to improve learning efficiency.
2. **Exploration strategy refinement** - The current entropy coefficient of 0.01 provides only moderate exploration. Implementing adaptive entropy coefficients that adjust based on learning progress could improve exploration-exploitation balance.
3. **Network architecture enhancements** - Adding recurrent layers (LSTM/GRU) would help the agent better capture temporal dependencies in soil moisture and weather patterns, which is crucial for irrigation decisions.
4. **Value function estimation accuracy** - The relatively high standard deviation in rewards (66.39) suggests potential inconsistency in value estimation. Implementing Generalized Advantage Estimation (GAE) could reduce variance while maintaining acceptable bias.
5. **Handling partial observability** - The irrigation environment is inherently partially observable with uncertain weather forecasts. Incorporating state augmentation or belief state modeling could improve performance.
6. **Multi-task learning** - The current A2C focuses solely on irrigation optimization, but a multi-objective approach that explicitly balances water conservation and crop yield might achieve better overall performance.

Dyna-Q

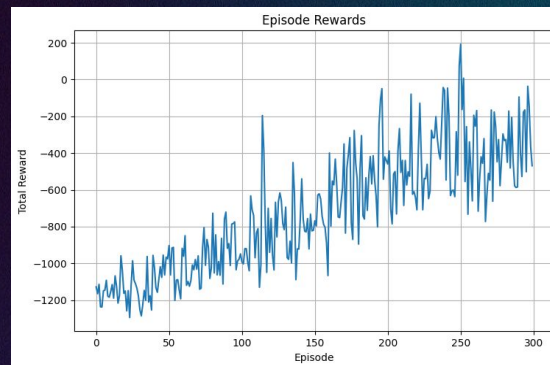
ϵ -greedy exploration



Final Exploration of 0.05







Final Exploration of 0.01



Final Exploration of 0.001

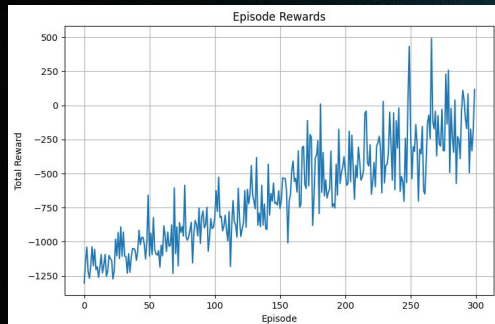
Dyna-Q

ϵ -greedy exploration

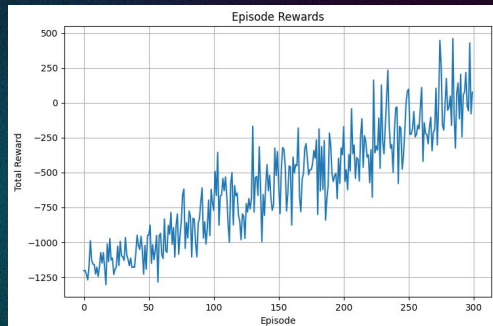
|  Exploration Rate |  Final Reward |  Variation |  Total Episodes |
|--|--|---|--|
| 0.05 (Best Rate!) | -182.17 | ± 180.39 | 300 |
| 0.01 | -345.57 | ± 227.08 | 300 |
| 0.001 | -238.67 | ± 274.30 | 300 |

Dyna-Q

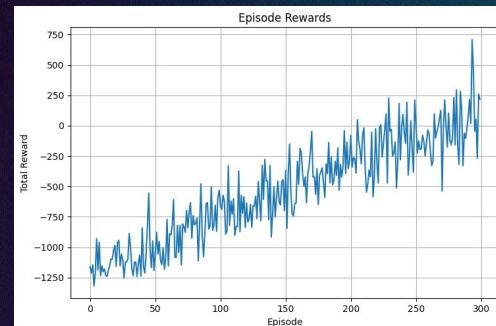
Performance Across different Learning Rate



Learning Rate of 0.1



Learning Rate of 0.05



Learning Rate of 0.01

Dyna-Q

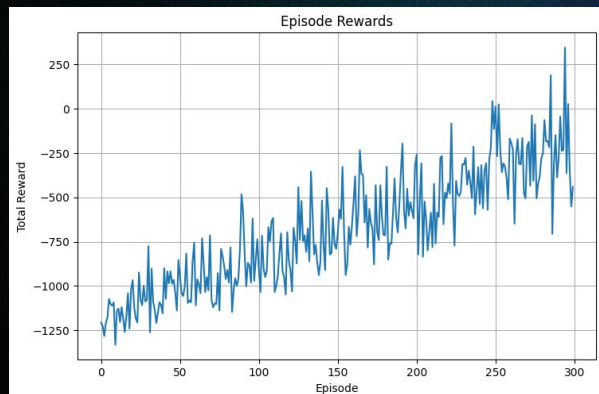
Performance Across different Learning Rate

|  Learning Rate |  Final Reward |  Variation |  Total Episodes |
|---|--|--|--|
| 0.1 | -280.32 | ± 325.19 | 300 |
| 0.05 | -80.55 | ± 421.39 | 300 |
| 0.01 | -56.34 | ± 544.99 | 300 |

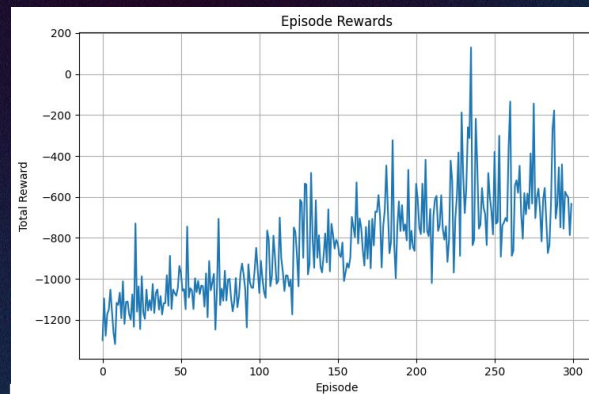
0.01 led to the highest cumulative reward but with a high variation. While a rate of 0.05 also yielded strong performance with a lower variation compare to learning rate 0.01. The learning rate of 0.1 has a low variance.

Dyna-Q

Performance Across different Discount Factor











Discount Factor of 0.95



Discount Factor of 0.99

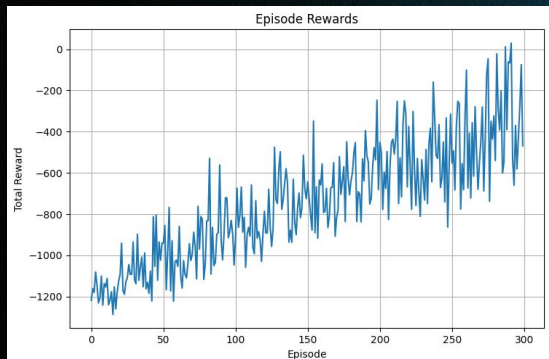
Dyna-Q

Performance Across different Discount Factor

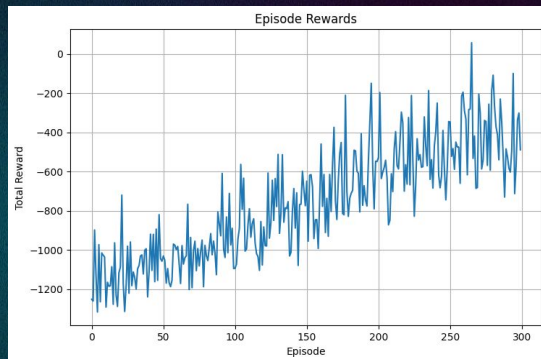
|  Discount Factor |  Final Reward |  Variation |  Total Episodes |
|--|--|--|--|
| 0.90 | -37.86 | ± 306.07 | 300 |
| 0.99 | -165.64 | ± 192.37 | 300 |
|  Discount Factor |  Final Reward |  Variation |  Total Episodes |

Dyna-Q

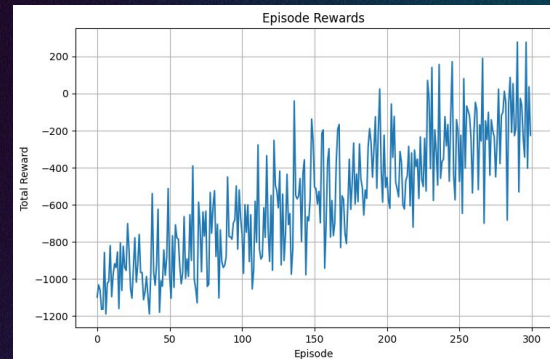
Different randomness test on controlled environment



Easy







Normal



Hard

Dyna-Q

Performance Across different Difficulty

|  Difficulty |  Final Reward |  Variation |  Total Episodes |
|--|--|--|--|
| Easy | -20.58 | ± 262.63 | 300 |
| Normal | -182.17 | ± 180.39 | 300 |
| Hard | -298.15 | ± 540.68 | 300 |

Dyna-Q

Strengths:

- **Learning Efficiency (Potentially):**
 - The episode rewards generally show an upward trend over the 300 episodes across all difficulties level. This suggests that the Dyna-Q model is capable of learning and improving its performance within a reasonable number of interactions.
- **Adaptability to Different Difficulty Levels:**
 - The model demonstrates the ability to learn in environments with varying levels of complexity. While the final rewards differ significantly, the learning curves in all three scenarios indicate that the model is adapting its policy to the specific challenges of each environment.
- **Model-Based Reinforcement Learning Advantage:**
 - The nature of model-based has allows it to perform simulated experiences in addition to real-world interactions.

Dyna-Q

Weaknesses:

- **Sensitivity to Environment Complexity:**
 - Based the previous records, the final rewards achieved by the Dyna-Q agent decrease significantly as the environment difficulty increases. This indicates that the model struggles more to find optimal policies in more complex environments.
- **Increased Reward Variation in Harder Environments:**
 - The variation also increases when the difficulty level increases. This suggests that the learning process becomes less stable and as the environment becomes more challenging.
- **Computational Cost of Planning:**
 - Planning adds a computational overhead. The results don't directly show this, but it's a consideration for model-based methods.

Dyna-Q

Areas of improvements:

1. Enhanced Exploration Strategies

- Apply curiosity-driven exploration that use a learned feature embedding and rewards the agent for transitions that lead to high prediction error in the next state

2. Improved Model Learning

- Agents can learn a compressed, low-dimensional representation of the environment using a Variational Autoencoder (VAE)

3. More Efficient Planning

- Uses trajectory sampling and optimization to improve decision-making

4. Addressing Dynamic Environments

- Apply Model-Agnostic Meta-Learning (MAML), that can be incorporated into model-based reinforcement learning to allow fast adaptation and re-learning of the environment dynamics.

6.0 Reinforcement Learning Algorithm Comparison

Evaluation Results for Each Model

A2C Evaluation Results

| Metric | Value |
|-----------------------|--------------------|
| Mean Reward | 830.31 \pm 66.39 |
| Min Reward | 689.63 |
| Max Reward | 948.57 |
| Mean Episode Length | 120.00 |
| Mean Yield | 0.97 |
| Mean Water Usage | 201.00 mm |
| Mean Water Efficiency | 0.0049 yield/mm |

PPO Evaluation Results

| Metric | Value |
|-----------------------|--------------------|
| Mean Reward | 720.88 \pm 65.43 |
| Min Reward | 604.44 |
| Max Reward | 815.15 |
| Mean Episode Length | 120.00 |
| Mean Yield | 0.76 |
| Mean Water Usage | 201.00 mm |
| Mean Water Efficiency | 0.0038 yield/mm |

DQN Evaluation Results

| Metric | Value |
|-----------------------|----------------------|
| Mean Reward | 1074.98 \pm 235.05 |
| Min Reward | 598.80 |
| Max Reward | 1248.12 |
| Mean Episode Length | 120.00 |
| Mean Yield | 1.45 |
| Mean Water Usage | 195.00 mm |
| Mean Water Efficiency | 0.0074 yield/mm |

Dyna-Q Evaluation Results

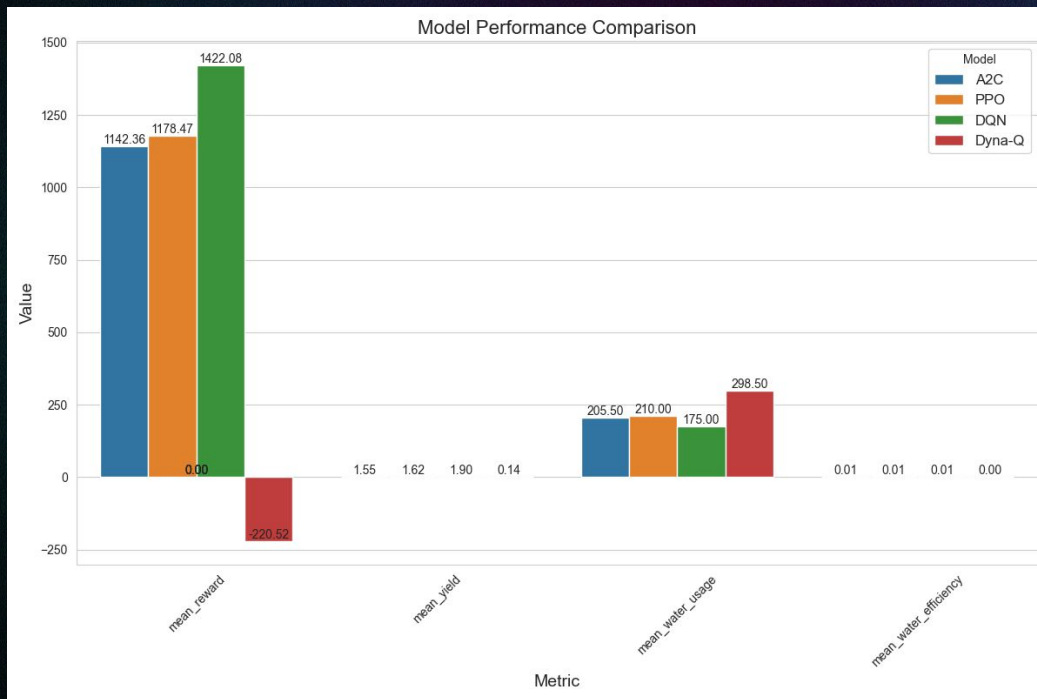
| Metric | Value |
|-----------------------|---------------------|
| Mean Reward | -40.29 \pm 263.31 |
| Min Reward | -330.60 |
| Max Reward | 471.06 |
| Mean Episode Length | 120.00 |
| Mean Yield | 0.21 |
| Mean Water Usage | 298.50 mm |
| Mean Water Efficiency | 0.0007 yield/mm |

Model Comparison

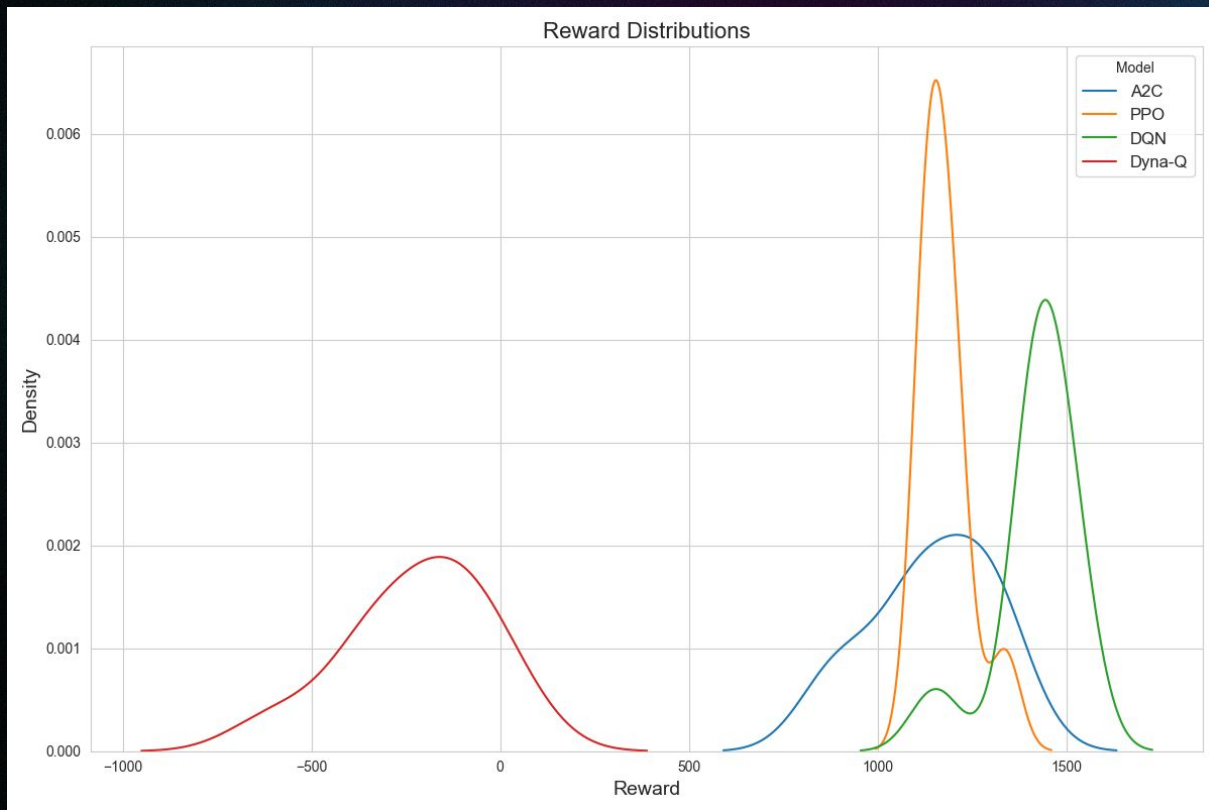
Model Comparison

| Metric | A2C | PPO | DQN | Dyna-Q |
|-----------------------------|--------|--------|---------|--------|
| Mean Reward | 830.31 | 720.88 | 1074.98 | -40.29 |
| Reward Std | 66.39 | 65.43 | 235.05 | 263.31 |
| Mean Episode Length | 120.00 | 120.00 | 120.00 | 120.00 |
| Mean Yield | 0.97 | 0.76 | 1.45 | 0.21 |
| Mean Water Usage (mm) | 201.00 | 201.00 | 195.00 | 298.50 |
| Water Efficiency (yield/mm) | 0.0049 | 0.0038 | 0.0074 | 0.0007 |

Model Comparison



Reward Distribution



Short-list model

Model Scores

| Model | Score | Rank |
|--------|--------|------|
| DQN | 1.0000 | 1 |
| PPO | 0.7883 | 2 |
| A2C | 0.7714 | 3 |
| Dyna-Q | 0.0000 | 4 |

Model Recommended

Recommendation model: DQN

Score: 1.0000

This recommendation is based on the following weighted assessment:

- mean_reward (Weight: 0.40)
- mean_water_efficiency (Weight: 0.30)
- mean_yield (Weight: 0.20)
- mean_water_usage (Weight: 0.10)

Thank you