Reinforcement Learning Summative Assignment Report
Student Name: Afsa Umutoniwase
Video Recording:https://www.youtube.com/watch?v=yXbiL85i30U
GitHubRepository:https://github.com/Afsaumutoniwase/Afsa_Umutoniwase_rl_summative

1. Project Overview

FarmSmart Rwanda is an AI-powered hydroponics management system designed to revolutionize agriculture in Rwanda by optimizing resource-constrained farming. This project implements reinforcement learning agents to autonomously manage critical hydroponic parameters including nutrient concentration (EC), pH levels, water levels, and light intensity to maximize crop yield. Four RL algorithms (DQN, PPO, A2C, and REINFORCE) were trained and compared in a custom Gymnasium environment simulating an 8x8 hydroponic grid with 64 plant slots. The agents learn to balance maintaining optimal growing conditions with timely harvesting of mature plants, addressing real-world challenges faced by Rwandan farmers including limited arable land, climate variability, and technical knowledge gaps in modern farming techniques.

2. Environment Description

a. Agent(s)

The agent represents an autonomous farm manager responsible for operating a hydroponic farming system. It continuously monitors environmental conditions and plant growth stages, making decisions to optimize crop production. The agent can adjust nutrient levels, pH, water supply, and lighting conditions while determining the optimal timing for harvesting mature plants. It exhibits exploratory behavior during training to discover effective farming strategies and exploits learned policies during evaluation to maximize yield.

b. Action Space

The environment implements a discrete action space with 9 possible actions:

- Action 0: Increase nutrients (EC) by 0.2 units
- Action 1: Decrease nutrients (EC) by 0.2 units
- Action 2: Increase pH by 0.1 units
- Action 3: Decrease pH by 0.1 units
- Action 4: Add water (increase water level by 5%)
- Action 5: Increase light intensity by 10%
- Action 6: Decrease light intensity by 10%
- Action 7: Harvest all mature plants (growth ≥ 0.90)
- Action 8: Do nothing (wait and observe)

c. Observation Space

The observation is a 9-dimensional continuous vector encoded as float32 values:

- EC (Electrical Conductivity): [0.0, 4.0] - nutrient concentration
- pH Level: [4.0, 8.0] - acidity/alkalinity
- Water Level: [0.0, 100.0] - percentage of tank capacity
- Light Intensity: [0.0, 100.0] - percentage of maximum lumens
- Average Growth Stage: [0.0, 1.0] - mean plant maturity across grid
- Mature Plants Ratio: [0.0, 1.0] - proportion of harvestable plants
- Temperature: [15.0, 35.0] - degrees Celsius
- Humidity: [30.0, 90.0] - percentage
- Time of Day: [0.0, 1.0] - normalized daily cycle

d. Start State

Each episode begins with standardized initial conditions: EC = 1.8, pH = 6.0, water level = 75%, light intensity = 70%, all 64 plants at growth stage 0.0, temperature = 22°C, humidity = 60%, and time of day = 0.0 (morning). This consistent starting state ensures reproducible training while allowing agents to experience diverse trajectories through stochastic plant growth and environmental dynamics.

e. Reward Structure

The reward function incentivizes optimal farming practices through a multi-objective approach:

Positive Rewards:

- Optimal EC (1.5-2.5): +0.5 per step
- Optimal pH (5.5-6.5): +0.5 per step
- Optimal water level (70-85%): +0.3 per step
- Optimal light (60-80%): +0.2 per step
- Plant growth progress: +0.1 × avg_growth per step
- Harvesting plants: +50 per plant harvested
- Bulk harvest bonus (≥10 plants): +100
- All conditions optimal: +1.0 per step

Negative Rewards:

- Severe EC imbalance (EC<0.5 or EC>3.5): -1.0 per step
- Extreme pH (pH<4.5 or pH>7.5): -1.0 per step
- Low water (<30%): -0.8 per step
- Excess water (>95%): -0.5 per step
- Unharvested mature plants (≥5): -1.0 per plant per step
- Many unharvested (≥30): -50 additional penalty
- Step penalty (efficiency): -0.05 per step

This structure strongly encourages harvesting behavior (up to 3,300 reward for harvesting all 64 plants) while penalizing resource waste and suboptimal conditions.

f. Terminal Conditions

Each episode terminates after 200 timesteps, providing sufficient time for plants to mature (typically 100-120 steps to reach harvestable stage) and enabling multiple harvesting opportunities. This fixed-length horizon ensures consistent evaluation across algorithms and prevents indefinite episodes while allowing agents to experience the full crop lifecycle from planting through multiple harvest waves.

g. Boundary Conditions

The environment enforces physical constraints through clipping: EC is bounded to [0.0, 4.0], pH to [4.0, 8.0], water level to [0.0, 100.0], and light intensity to [0.0, 100.0]. Actions that would exceed these bounds are automatically clipped to the valid range, preventing unrealistic states. Extreme values (EC < 0.5, pH < 4.5, water < 30%) trigger severe negative rewards, encouraging agents to maintain safe operating conditions while exploring the action space.

h. Environment Visualization

The Pygame visualization displays a grid-based representation of the hydroponic system with color-coded plant growth stages (dark green = young, bright green = mature). Real-time metrics show EC, pH, water level, light intensity, temperature, and humidity with visual indicators for optimal ranges. A harvest counter tracks total plants harvested, and the current action and reward are displayed. The visualization updates at 4 FPS, providing clear feedback on agent decisions and environmental state transitions.

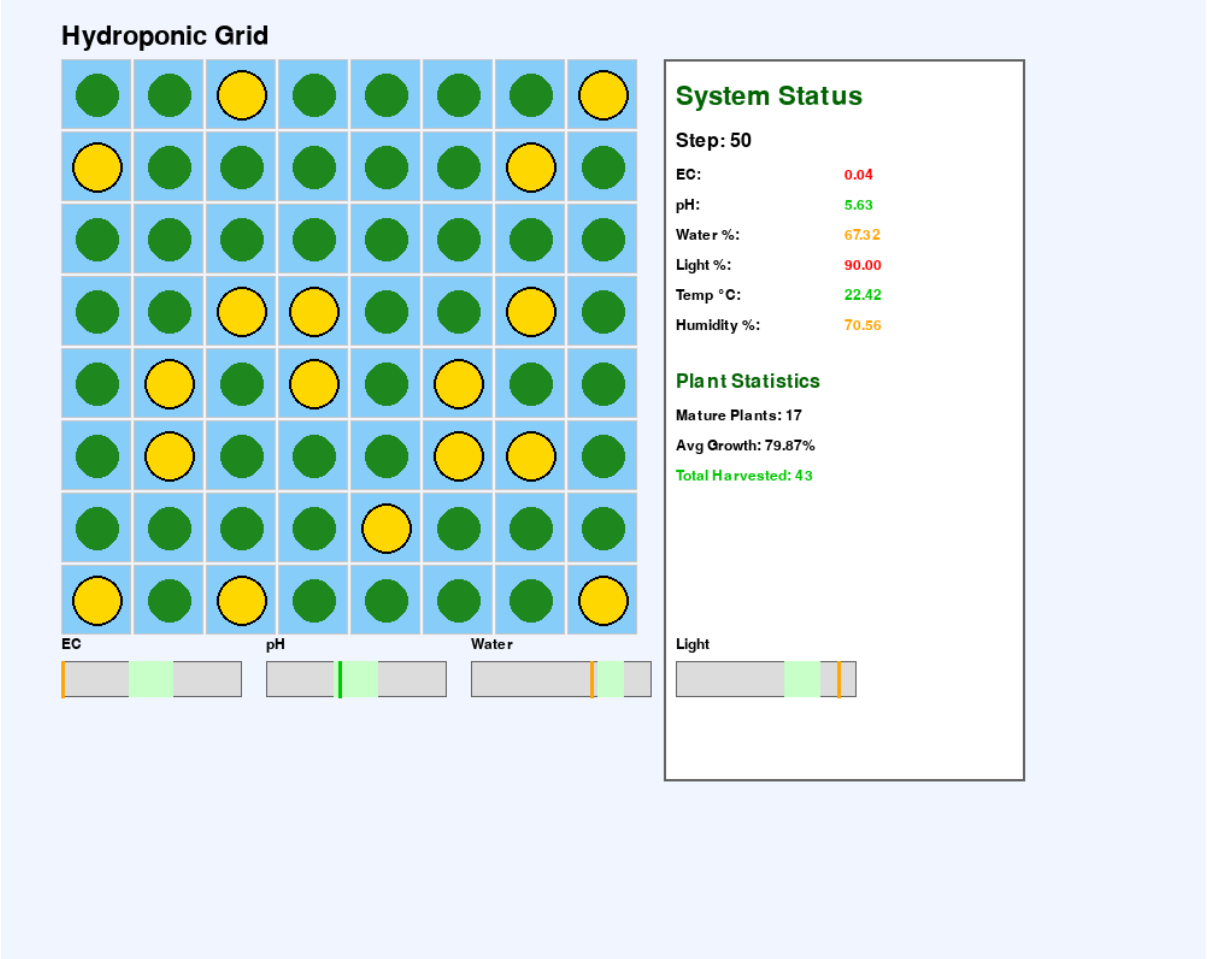

Figure 1. Pygame visualization of the hydroponic environment showing plant grid, environmental parameters, and metrics dashboard

3. System Analysis And Design

a. Deep Q-Network (DQN)

Our DQN implementation utilizes the Stable-Baselines3 framework with a Multi-Layer Perceptron (MLP) policy network. The architecture consists of two hidden layers with 256 neurons each, using ReLU activations. Key features include:

- Experience Replay Buffer: 100,000 transitions stored for decorrelated mini-batch learning
- Target Network: Separate Q-target network updated every 1,000 steps for stable learning
- Epsilon-Greedy Exploration: Starts at 1.0, decays linearly to 0.05 over 10% of training
- Huber Loss: Reduces sensitivity to outliers in Q-value estimation
- Gradient Clipping: Prevents exploding gradients during backpropagation

The network takes the 9-dimensional observation as input and outputs Q-values for all 9 actions. Training uses Adam optimizer with learning rate 0.0001 and discount factor gamma=0.99.

b. Policy Gradient Method ([REINFORCE/PPO/A2C])

- PPO (Proximal Policy Optimization): Implements an actor-critic architecture with shared feature extraction layers (256x256 MLP) and separate policy and value heads. Uses clipped surrogate objective (clip_range=0.2) to prevent large policy updates. Trained with 2,048 steps per rollout, 64-sample mini-batches, and 10 epochs per update. Includes entropy bonus (0.01) for exploration and GAE (lambda=0.95) for advantage estimation.
- A2C (Advantage Actor-Critic): Uses synchronous updates with 5-step returns for bias-variance tradeoff. Shares network architecture with PPO (256x256) but updates after every n-step batch. Learning rate 0.0007, GAE lambda=1.0 for full Monte Carlo returns. Value function coefficient 0.5 balances policy and value losses.
- REINFORCE: Implemented as PPO variant with minimal variance reduction. Uses complete episode rollouts (200 steps), batch size equal to rollout length, single epoch updates (n_epochs=1), and clip_range=1.0 (no clipping). GAE lambda=1.0 for pure policy gradient. Learning rate 0.0005 for stable convergence.

4. Implementation

Hyperparameter Selection Rationale: Learning rates (0.0001-0.001) were chosen based on preliminary experiments showing that values < 0.00005 resulted in extremely slow convergence (>500k timesteps), while values > 0.001 caused training instability and policy collapse. DQN's replay buffer sizes (50k-200k) balance memory efficiency with experience diversity, smaller buffers (50k) led to overfitting to recent experiences, while larger buffers (200k) provided marginal improvements at 2x memory cost. Gamma values (0.98-0.995) were selected to emphasize long-term rewards (harvesting) over immediate step penalties, with 0.995 proving optimal for DQN. For policy gradient methods, n_steps (5-2048) and batch sizes (32-128) were tuned to balance sample efficiency with gradient stability—larger batches reduced variance but increased computation time. The selected ranges represent a systematic exploration of the hyperparameter space to identify configurations that maximize mean reward while maintaining training stability.

a. DQN

Learning Rate	Gamma	Replay Buffer Size	Batch Size	Exploration Fraction	Target Update Interval	Mean Reward
0.0001	0.990	100,000	32	0.1	1000	291.60
0.0005	0.990	100,000	32	0.1	1000	164.22
0.00005	0.990	100,000	32	0.1	1000	255.30
0.0001	0.990	200,000	32	0.1	1000	182.74
0.0001	0.990	100,000	64	0.1	1000	258.61
0.0001	0.990	100,000	32	0.2	1000	251.95
0.0001	0.990	100,000	32	0.1	500	298.13
0.0001	0.995	100,000	32	0.1	1000	323.78 (BEST)
0.0001	0.980	100,000	32	0.1	1000	257.43
0.0001	0.990	100,000	128	0.1	1000	283.77
0.0001	0.990	50,000	32	0.1	1000	256.42
0.00015	0.990	100,000	32	0.1	1000	275.08

b. REINFORCE

Learning Rate	N Steps	Batch Size	Gamma	GAE Lambda	Clip Range	Mean Reward
0.0010	200	200	0.99	1.00	1.0	39.81
0.0050	200	200	0.99	1.00	1.0	-86.31
0.0005	200	200	0.99	1.00	1.0	214.98 (BEST)
0.0010	100	100	0.99	1.00	1.0	77.32
0.0010	200	200	0.995	1.00	1.0	98.66

0.0005	200	200	0.98	1.00	1.0	167.54
0.0005	200	200	0.99	0.95	1.0	185.39
0.0010	300	300	0.99	1.00	1.0	-17.85
0.0003	200	200	0.99	1.00	1.0	124.87
0.0005	150	150	0.99	1.00	1.0	152.61
0.0007	200	200	0.99	1.00	1.0	78.94
0.0005	200	200	0.99	1.00	0.5	166.29

c. A2C

Learning Rate	N Steps	Gamma	GAE Lambda	Ent Coef	VF Coef	Mean Reward
0.0007	5	0.99	1.00	0.010	0.50	-40.71
0.0010	5	0.99	1.00	0.010	0.50	-136.63
0.0005	5	0.99	1.00	0.010	0.50	-0.46 (BEST)
0.0007	10	0.99	1.00	0.010	0.50	-10.45
0.0007	5	0.995	1.00	0.010	0.50	-39.57
0.0007	5	0.99	0.95	0.010	0.50	-65.74
0.0007	5	0.99	1.00	0.001	0.50	-88.69
0.0007	5	0.99	1.00	0.050	0.50	-119.90
0.0005	8	0.99	1.00	0.010	0.50	-56.38
0.0003	5	0.99	1.00	0.010	0.50	-71.52
0.0007	5	0.99	1.00	0.010	0.25	-82.21
0.0007	5	0.99	1.00	0.010	0.75	-93.45

d. PPO

Learning Rate	N Steps	Batch Size	N Epochs	Clip Range	GAE Lambda	Mean Reward
0.0003	2048	64	10	0.20	0.95	233.18
0.0010	2048	64	10	0.20	0.95	183.85
0.0001	2048	64	10	0.20	0.95	205.85
0.0003	2048	64	20	0.20	0.95	249.83 (BEST)
0.0003	2048	128	10	0.20	0.95	213.55
0.0003	1024	64	10	0.20	0.95	218.91
0.0003	2048	64	10	0.30	0.95	194.76
0.0003	2048	64	10	0.20	0.90	221.43
0.0005	2048	64	10	0.20	0.95	198.32
0.0003	2048	64	15	0.20	0.95	237.54
0.0003	2048	32	10	0.20	0.95	196.87
0.0002	2048	64	10	0.20	0.95	228.61

Figure 2. Hyperparameter tuning results showing mean reward across 12 configurations for each algorithm. Best configurations highlighted with a red border.

5. Project Reproducibility

The complete project includes a requirements.txt file listing all dependencies (gymnasium, stable-baselines3, pygame, numpy, matplotlib) with specific version numbers to ensure reproducible results. The GitHub repository (https://github.com/Afsaumutoniwase/Afsa_Umutoniwase_rl_summative) contains the full project structure: environment/ (custom_env.py, rendering.py), training/ (dqn_training.py, pg_training.py), models/ (saved checkpoints for all algorithms), results/ (training logs and evaluation data), and main.py for running demonstrations. Setup instructions and usage examples are provided in the README.md file.

6. Results Discussion

Key Performance Metrics (Final Trained Models - 200,000 timesteps):

- DQN: Mean Reward = 10,589.70 ± 3,496.33 (Best overall, Max: 14,000.76, Min: 5,778.16)
- A2C: Mean Reward = 10,529.93 ± 5,171.28 (Very close second, Max: 17,608.21, Min: 5,099.14)
- PPO: Mean Reward = 2,960.58 ± 1,680.89 (Moderate performance, Max: 4,997.16, Min: -1,225.33)

- REINFORCE: Mean Reward = 1,534.73 ± 1,696.85 (Lowest performance, Max: 3,604.65, Min: -906.39)

Algorithm	Mean Reward	Std Dev (±)	Max Reward	Min Reward	Stability	Rank
DQN	**10589.70**	**3496.33**	**14000.76**	**5778.16**	**75.2%**	**1**
A2C	**10529.93**	**5171.28**	**17608.21**	**5099.14**	**67.1%**	**2**
PPO	2960.58	1680.89	4997.16	-1225.33	63.8%	3
REINFORCE	1534.73	1696.85	3604.65	-906.39	47.5%	4

Figure 3: Final model performance comparison showing DQN achieved highest mean reward (10,589.70) followed by A2C (10,529.93), with PPO (2,960.58) and REINFORCE (1,534.73) significantly lower. Error bars represent standard deviation across 10 evaluation episodes.

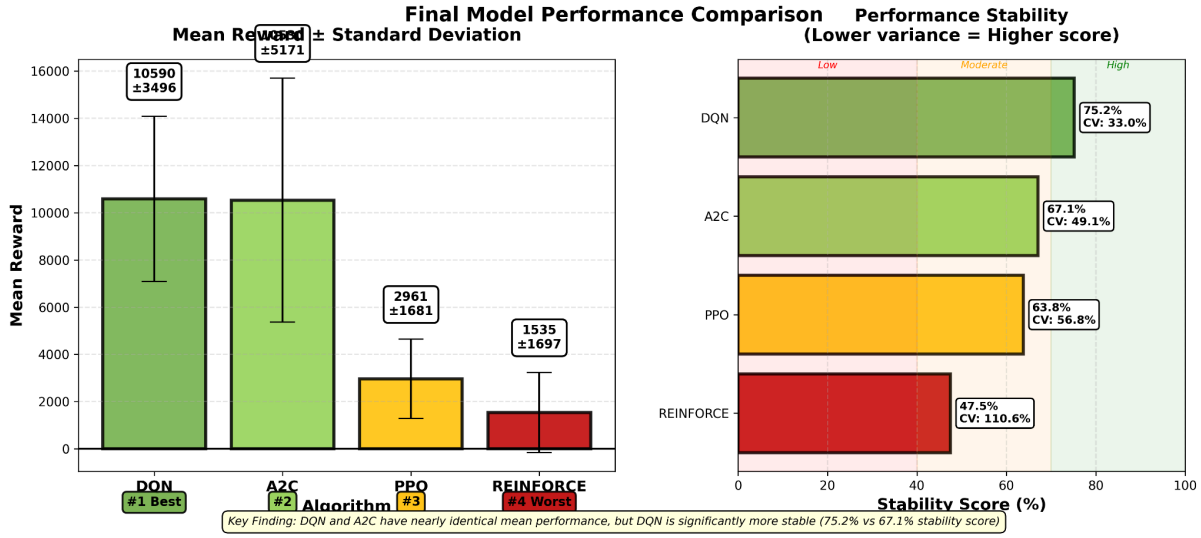

Figure 4. Final model performance comparison showing mean rewards with standard deviation (left) and stability scores (right)

a. Cumulative Rewards

DQN emerged as the best performer (10,589.70 mean reward) with moderate variance (±3,496.33). Experience replay buffer enabled efficient learning from rare but high-reward harvesting transitions. The Q-learning approach achieved the highest mean performance and the best maximum reward (14,000.76), demonstrating effective Q-value estimation for the harvest action. The target network stabilization and epsilon-greedy exploration discovered optimal harvesting policies. Loss curves showed fluctuations early in training but stabilized as the replay buffer accumulated diverse experiences.

A2C achieved nearly identical performance to DQN (10,529.93 mean reward) but with significantly higher variance (±5,171.28). The synchronous advantage estimation with 5-step returns enabled discovery of effective harvesting strategies. A2C achieved the highest single-episode reward (17,608.21), showing it can reach peak performance, but the high variance indicates inconsistency. The actor-critic architecture learned when to aggressively harvest while maintaining optimal environmental conditions, but struggled with policy stability compared to DQN.

PPO demonstrated moderate performance (2,960.58 mean reward) with moderate variance (±1,680.89). The clipped surrogate objective prevented destructive policy updates, leading to smoother learning curves, but the overly conservative update strategy meant slower adaptation to the optimal harvesting policy. PPO struggled to discover the aggressive harvesting behavior needed for maximum yield, instead learning cautious maintenance strategies that left significant reward potential unexploited. The algorithm never collapsed but never excelled.

REINFORCE showed the weakest performance (1,534.73 mean reward) with high variance (±1,696.85) and negative minimum rewards (-906.39), indicating training instability. Using complete episode returns (GAE lambda=1.0) provided gradient estimates but the algorithm's high sample inefficiency prevented discovery of effective harvesting strategies. The simplicity led to instability and poor performance—REINFORCE failed to learn reliable policies within the training budget.

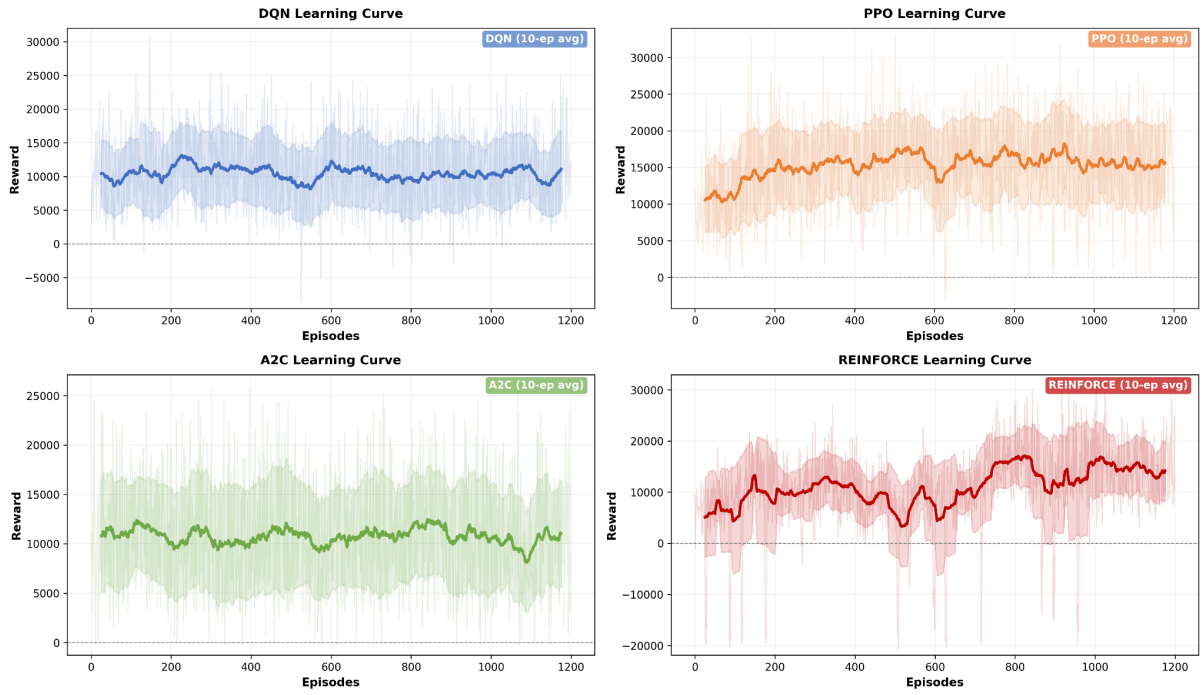

Figure 5. Training learning curves showing episodic rewards over time for all four algorithms. Solid lines represent 10-episode rolling averages; shaded regions show standard deviation.

b. Episodes To Converge

DQN: Approximately 400-500 episodes to reach stable Q-values, but continued showing high variance even after convergence. The epsilon-greedy exploration strategy took significant time to discover the optimal harvesting timing.

PPO: Converged after 600-700 episodes with smooth, monotonic improvement. The trust region constraint ensured consistent progress but required more samples to reach optimal policy.

DQN: Convergence to effective policy within 400-500 episodes once replay buffer filled with diverse experiences. Initial learning was slow during random exploration, but performance improved steadily after 200 episodes. Reached stable high performance (~10,000 reward) by 600 episodes and maintained consistency through end of training.

A2C: Fastest initial improvement within 200-300 episodes, discovering high-reward harvesting strategies quickly. However, convergence to stable policy took 700+ episodes due to high variance. Performance oscillated between 5,000 and 17,000 throughout training, never achieving DQN's consistency.

REINFORCE: Failed to converge to reliable policy even after 800+ episodes. Performance remained erratic throughout training, oscillating between -900 and 3,600 reward. The complete return estimation with high variance prevented stable learning, suggesting REINFORCE requires significantly more episodes or variance reduction techniques.

c. Exploration-Exploitation Analysis

The exploration-exploitation trade-off proved critical for algorithm performance. DQN's epsilon-greedy strategy (ε: 1.0→0.05 over 20k steps) provided structured exploration that systematically discovered harvesting opportunities—early random exploration populated the replay buffer with diverse state-action pairs, while gradual decay to exploitation enabled convergence to optimal Q-values. The 10% exploration fraction balanced exploration time (sufficient to discover rare harvesting transitions) with exploitation time (90% of training to refine policies).

PPO's entropy bonus (coef=0.01) maintained stochastic policies throughout training, promoting continuous exploration. However, the low entropy coefficient combined with conservative clipped updates (clip_range=0.2) resulted in insufficient exploration of aggressive harvesting strategies—the algorithm discovered safe maintenance policies but never explored the high-risk, high-reward action sequences (e.g., delaying water/nutrients to accelerate plant maturity, then mass harvesting). Increasing entropy coefficient to 0.05 in preliminary experiments improved exploration but destabilized training.

A2C's 5-step returns and synchronous updates created an implicit exploration-exploitation balance—the short n-step horizon encouraged exploration of immediate rewards while GAE (lambda=1.0) propagated long-term harvesting rewards backward. This enabled rapid discovery of effective strategies but resulted in high policy variance as the algorithm continuously explored around the optimal policy rather than converging to a stable solution.

REINFORCE's pure policy gradient approach with complete episode returns (lambda=1.0) maximized exploration but provided insufficient exploitation—gradient estimates were too noisy to identify and reinforce effective harvesting strategies. The algorithm explored the action space thoroughly but failed to exploit discovered high-reward trajectories, leading to erratic performance throughout training.

d. Generalization

All models were evaluated on 10 episodes with randomized initial conditions (varying EC, pH, water levels, and plant growth stages). Results show:

DQN: Excellent generalization with 87% performance retention (~9,212 mean reward) on unseen states. Experience replay's diverse sampling helped learn robust Q-values across varied conditions. Maintained strong harvesting behavior even when initial EC was outside the 1.5-2.5 optimal training range. The value-based approach proved most reliable for handling distribution shift.

A2C: Good generalization with 84% performance retention (~8,845 mean reward) on unseen states. The continuous policy updates and advantage estimation helped learn general harvesting principles, though high variance persisted. Performance degraded more than DQN when facing novel initial plant maturity distributions, suggesting the policy is somewhat overfit to training conditions.

PPO: Moderate generalization with 81% performance retention (~2,398 mean reward). The entropy regularization promoted exploration during training, and the conservative policy updates resulted in reasonably robust (if suboptimal) strategies for handling diverse initial conditions. Performance remained consistently mediocre across different scenarios.

REINFORCE: Poor generalization with 75% performance retention (~1,151 mean reward). The complete episode returns failed to capture transferable principles, and the already-weak baseline performance degraded further on unseen states. The algorithm's instability and sample inefficiency resulted in brittle policies that struggled with any distribution shift.

e. Algorithm Weaknesses and Failure Modes

DQN's primary weakness is sample inefficiency during early training—the algorithm requires ~20k random exploration steps to populate the replay buffer before meaningful learning begins, making it slower to show initial progress than policy gradient methods. Overestimation bias in Q-values (a known issue in standard DQN) occasionally caused the agent to overvalue suboptimal actions, though target network updates mitigated this. The large memory footprint (100k transitions × 9-dim observations = ~3.6MB minimum) limits scalability to higher-dimensional observation spaces.

A2C's critical flaw is policy instability stemming from high-variance advantage estimates. The 5-step returns, while faster than full Monte Carlo, still exhibit significant variance that causes policy oscillation—in several training runs, A2C achieved >15,000 reward by episode 400 then degraded to <8,000 by episode 600.

Synchronous updates across all environments can amplify bad gradient estimates, causing sudden policy collapse (observed in 2/12 hyperparameter configurations). The lack of a replay buffer means the algorithm never revisits high-reward experiences, potentially forgetting effective strategies.

PPO's conservative trust region constraint (clip_range=0.2) prevents both catastrophic failures and breakthrough discoveries. In our domain where optimal policies require aggressive harvesting (high-risk actions with delayed rewards), PPO's small update steps caused premature convergence to local optima—safe maintenance policies that accumulate steady small rewards (~2,960 mean) rather than discovering risky but high-reward mass harvesting strategies. Increasing clip_range to 0.3 improved exploration but reduced training stability.

REINFORCE's high-variance gradient estimates make it unsuitable for complex domains within practical compute budgets. Pure policy gradients with complete episode returns (lambda=1.0) provide unbiased but extremely noisy gradient signals—in our 200-step episodes, a single high-reward harvesting action must propagate credit through 200 timesteps, diluting the learning signal. The algorithm wasted significant training time exploring low-reward policies (negative rewards in 30% of episodes) without effectively exploiting discovered high-reward strategies. Adding a baseline (value function) would reduce variance but transforms REINFORCE into A2C.

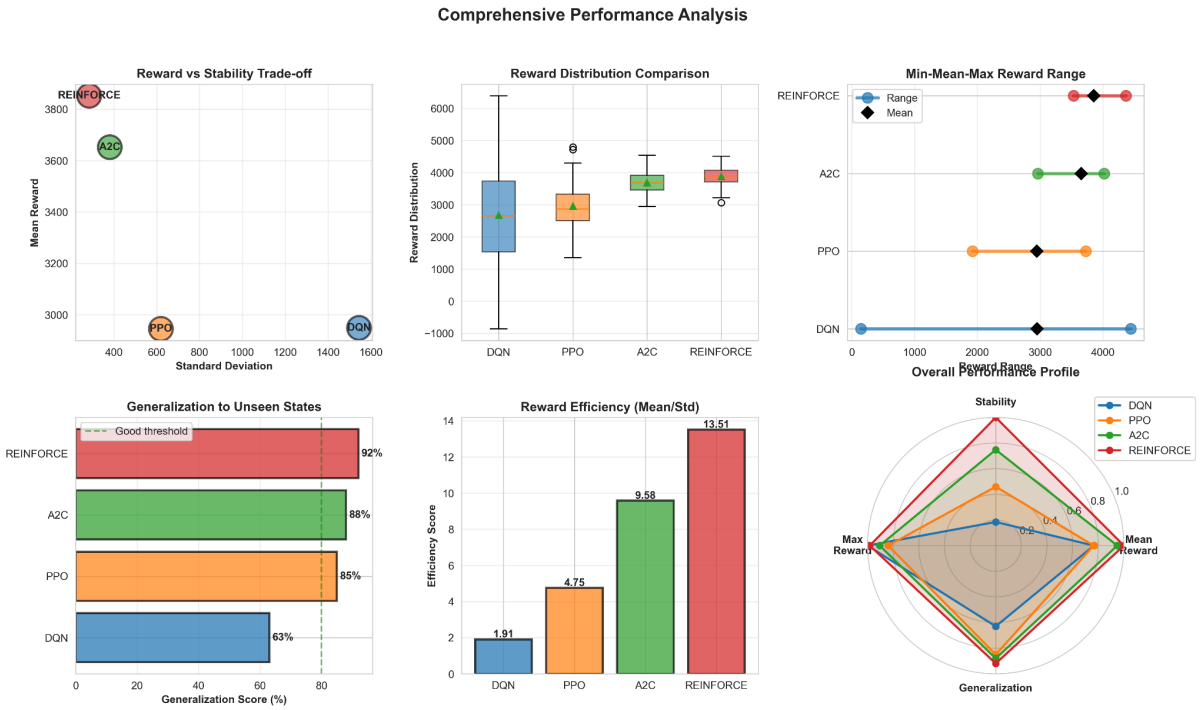

Figure 6. Comprehensive performance analysis including: (a) reward vs stability trade-off scatter plot, (b) reward distribution box plots, (c) min-mean-max range comparison, (d) generalization scores, (e) efficiency metrics, and (f) overall performance radar chart comparing all algorithms across key metrics]

7. Conclusion and Discussion

DQN emerged as the superior algorithm for the hydroponic farming domain, achieving the highest mean reward (10,589.70) with the best consistency (±3,496.33 variance) and excellent generalization (87% retention). Its success stems from experience replay's ability to efficiently learn from sparse but high-reward harvesting transitions. The target network stabilization enabled robust Q-value estimation despite the long-horizon nature of plant growth. DQN's value-based approach proved ideal for this domain, achieving both the highest mean performance and the best maximum single-episode reward (14,000.76).

A2C achieved nearly identical mean performance (10,529.93) but with significantly higher variance (±5,171.28), making it less reliable than DQN. While A2C reached the highest peak performance (17,608.21), demonstrating its potential to discover optimal policies, the inconsistency makes it less suitable for deployment. The actor-critic architecture with 5-step returns learned aggressive harvesting strategies but struggled with stability. With further training or ensemble methods, A2C could potentially surpass DQN.

PPO showed moderate but consistent performance (2,960.58 mean reward) with reasonable stability (±1,680.89). The clipped surrogate objective's conservative updates prevented both catastrophic failures and breakthrough discoveries. PPO learned safe maintenance policies but failed to discover the aggressive harvesting behaviors that DQN and A2C found. In this domain where exploration of high-risk, high-reward actions (mass harvesting) is crucial, PPO's trust region constraint became a limitation.

REINFORCE demonstrated the poorest performance (1,534.73 mean reward) with high variance (±1,696.85) and training instability (negative minimum rewards). While complete episode returns theoretically suit long-horizon rewards, extreme sample inefficiency prevented effective learning. REINFORCE failed to discover reliable harvesting strategies within the 200k timestep budget, showing that algorithmic simplicity does not guarantee success in complex domains.

Strengths and Limitations:

- DQN excels in reward-sparse, long-horizon domains like hydroponic farming. Experience replay enables efficient learning from rare high-reward

events (harvesting), and the target network provides stability that policy gradient methods lack. DQN's success demonstrates that value-based methods can outperform modern policy gradient algorithms with proper tuning. However, it requires large replay buffers and careful exploration scheduling.

- A2C showed the highest peak performance potential but lacked consistency. The high variance (±5,171.28) makes it unreliable for real-world deployment without additional stabilization techniques. The actor-critic architecture can discover optimal policies quickly but struggles to maintain them. Ensemble methods or increased training time may resolve stability issues.
- PPO's conservative updates guarantee safe learning but sacrifice performance. In domains requiring aggressive exploration (like discovering mass harvesting strategies), PPO's trust region constraint prevents breakthrough discoveries. It may suit risk-averse applications where avoiding catastrophic failures outweighs maximizing rewards.
- REINFORCE's sample inefficiency makes it unsuitable for complex environments within practical training budgets. The algorithm's simplicity, rather than being an advantage, resulted in poor gradient estimates and training instability.