Transfer Learning in Reinforcement Learning for Agricultural Decision Support in Polyculture Systems using FarmGym

Kai Haith

Abstract

This study investigates the performance of Deep Q-Network (DQN) and Proximal Policy Optimization (PPO) agents in decision support for agricultural settings, ranging from simple monoculture environments to a more complex polyculture system. Specifically, the research evaluates the performance of these reinforcement learning (RL) algorithms in managing farms with single crop types (beans and corn) and a polyculture farm (beans, corn, and tomatoes). A critical aspect of this study is examining the efficacy of transfer learning, where agents pretrained in monoculture settings are deployed in different monoculture environments or a polyculture environment. By comparing baseline performances and assessing the impact of transfer learning, this research aims to shed light on the efficiency of DQN and PPO agents in different agricultural scenarios, contributing to the development of more intelligent and sustainable farming practices through advanced AI methodologies.

1 Introduction

1.1 Context and Motivation

Agriculture is experiencing an evolution as smart farming techniques become critical for improving resource efficiency, crop yield, and environmental impacts. Integrating Reinforcement learning with agricultural systems can provide solutions to managing the complexities of irrigation, fertilization, planting, and pest control. Polyculture farming, growing multiple crops on the same land at the same time, is an even more complex system than the monoculture farming that dominates agriculture today. However, its potential to negate the negative environmental impacts and resource deficiency issues seen in traditional monoculture cropping makes it a promising frontier in agriculture. RL could be a critical factor in optimizing these systems due to its ability to learn based on interactions with an environment. Training an RL agent from scratch in a complex environment like polyculture farming can be computationally inefficient. By applying Transfer learning, this time-consuming process could be improved by

leveraging training from simpler monoculture environments to related polyculture environments. This project aims to investigate the effectiveness of transfer learning in RL agents initially trained in simpler monoculture environments and then deployed in a more complex polyculture environment, using Farm-Gym as a gamified testbed.

1.2 Research Focus and Objectives

The primary objective of this research is to compare the performance of DQN and PPO agents in different farming environments. The study first establishes baseline performances of these agents in monoculture environments and polyculture environments then assesses their adaptability through transfer learning. Transfer learning, in this context, involves pretraining the agents in a simple monoculture environment and then deploying them in either a different monoculture environment or a more complex polyculture farm. This approach aims to evaluate whether prior learning in a less complex environment can enhance the agent's performance and learning efficiency in a more challenging setting.

1.3 Significance of the Study

Understanding the performance of RL algorithms in agriculture is crucial for developing intelligent systems capable of handling the variability and complexity inherent in farming. This research holds significance not only for its potential to improve agricultural productivity but also for demonstrating an application of transfer learning beyond games and robotics.

2 Background / Related Work

2.1 Background

Reinforcement Learning has been used extensively in a variety of domains, with prominent examples being fields such as gaming and robotics. In these fields as well as in agriculture, the goal is to create agents that can learn from interactions with an environment to make optimal decisions. In agriculture, RL is currently being used on many monoculture farms to optimize irrigation schedules, harvesting times, and planting strategies like crop rotation, planting layout, and crop selection. For a biological system and a natural environment, RL is ideal because it can take into account uncertainties like weather or pest control. More complexity is added when RL agents are applied to polyculture farms which have to take into consideration different watering and fertilizer needs as well as different harvesting times and pest control strategies. Transfer learning, which aims to transfer an agent with knowledge gained in one environment to a different environment, could be ideal for keeping reinforcement learning efficient in a more complex polyculture farm.

Polyculture farming is not a new practice but is only now being revisited on a large scale as a possible alternative to monoculture farming. Monocul-

ture farming has dominated our modern agricultural landscape because it is extremely efficient and scalable to farm a single crop. Any farming strategy (irrigation, fertilization, pesticide, weeding, etc.) can be applied to an entire field which leads to more efficient specialized tools and less labor. However, as monoculture farming is being found to lead to environmental degradation, crop vulnerability, and lower yields, polyculture farming is gaining attention for its benefits to pest control, disease reduction, and soil health. While there has been some demonstrated use of RL in monoculture environments, there is far less in polyculture environments. This is likely because monoculture environments can be treated uniformly with a single watering strategy, weeding strategy, etc. Polyculture environments necessitate more complex strategies that take into account the differing biological needs of each crop as well as the relationship dynamics between various crops. For example, corn, which grows to be quite tall, might provide too much shade if grown next to tomatoes or lettuce.

While using RL in agriculture is a relatively new strategy, there is a substantial amount of work that has been done in robotics and gaming that can be used as a starting point for training these agents in an agricultural setting. For instance, gaming deals with large quantities of uncertainty much like agricultural agents will need to. Experiments have been done in the gaming and robotics fields that apply transfer learning to leverage an agent's skill at one task ex. One v. one games applied to 3 v. 4 games. To take advantage of this large amount of research, this study uses 'Farm-Gym' which is a gamified simulation of a farm. This environment allows exploration into the potential application of transfer learning between monoculture and polyculture farms, while slightly simplifying the testing environment to fit the project scope.

2.2 Related Work

The concept of transfer learning in reinforcement learning (RL) has been extensively researched, particularly in the context of games and task complexity. The study "Transfer Learning via Inter-Task Mappings for Temporal Difference Learning" [10] explores transfer learning in game tasks, demonstrating its effectiveness in improving learning efficiency across different tasks. Similarly, "Learning Invariant Feature Spaces to Transfer Skills with Reinforcement Learning" [3] highlights the transfer of knowledge in robotics tasks, underscoring the utility of transfer learning in bridging the gap between simple and complex task environments. These studies lay the groundwork for applying transfer learning from simpler monoculture environments to the more intricate dynamics of polyculture farming.

In the realm of agriculture, reinforcement learning has been gaining traction as a tool for intelligent decision-making. The paper "Simulating Polyculture Farming to Tune Automation Policies for Plant Diversity and Precision Irrigation" [1] is a pioneering work that applies RL to polyculture farming. It introduces AlphaGardenSim for modeling small-scale polyculture tasks, providing a foundation for RL-based decision-making in agricultural settings, especially concerning factors like light, temperature, water, fertilizer, and planting.

"The role of big data in the agri-food system" [11] demonstrates the efficacy of RL agents in monoculture environments, particularly in optimizing planting, irrigation, and harvesting. This research suggests a potential pathway for employing RL in the more complex domain of polyculture farming.

"Artificial intelligence algorithms and models for sustainable agriculture production" [2] delves into multi-agent systems in agriculture, emphasizing the need for sophisticated modeling and decision-making in polyculture farming. This work highlights the diverse interactions within polyculture systems and the necessity for advanced AI solutions to manage them. Moreover, the review "Does plant diversity benefit agroecosystems? A synthetic review"[5] provides an in-depth examination of polyculture farming, discussing its benefits and complexities. This paper is instrumental in understanding the challenges involved in modeling polyculture environments and underscores the need for innovative solutions to enhance polyculture farming methods.

Finally, "Deep Learning in Agriculture" [4] reviews deep learning methods applied to smart farming. While this paper emphasizes the potential of AI in agricultural tasks, it stops short of addressing the application of transfer learning specifically to agricultural decision-making. Nonetheless, it serves as a valuable reference point for understanding the broader context of AI in agriculture.

3 Technical Background

3.1 Transfer Learning

In transfer learning, the agent leverages knowledge from a source task to improve learning efficiency or performance in a target task. This can involve transferring various aspects of the learning process, such as value functions, policies, or features. The transferred knowledge is typically represented by a set of parameters obtained from the source task, which are then used to initialize or guide learning in the target task. For instance, in "Transfer Learning via Inter-Task Mappings for Temporal Difference Learning", the focus is on transferring knowledge through mappings between tasks. If F and G represent the feature representations of the source and target tasks, respectively, a mapping function M:F$\rightarrow$G can be learned [10]. This mapping is used to transfer the value function or policy from the source to the target task. In "Learning Invariant Feature Spaces to Transfer Skills with Reinforcement Learning", the approach is to learn feature representations that are invariant across different tasks. This is achieved by optimizing a feature extractor such that the learned features are effective across various tasks [3].

In the context of RL, transfer learning can significantly reduce the sample complexity and training time, especially in complex environments. By transferring knowledge from simpler or related tasks, the agent can start with a better initial policy or value function estimate for the new task, leading to faster convergence. In this study, transfer learning is applied to transition RL agents (both DQN and PPO) from monoculture to polyculture farming environments.

The learned policies or value functions from the monoculture tasks (source) are used to initialize or guide the learning in the polyculture task (target), with the expectation of improved learning efficiency and performance.

3.2 Proximal Policy Optimization

Stable Baselines 3 (SB3) is a set of reliable implementations of reinforcement learning (RL) algorithms in Python, using PyTorch as the underlying deep learning framework [9].

Algorithm 1 PPO-Clip

Input: initial policy parameters θ_0, initial value function parameters ϕ_0
for $k = 0, 1, 2, ...$ **do**
  Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
  Compute rewards-to-go $\hat{R}_t$.
  Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
  Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \quad g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

  typically via stochastic gradient ascent with Adam.
  Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

  typically via some gradient descent algorithm.
end for

Figure 1: PPO-Clip algorithm pseudocode [8]

Proximal Policy Optimization (PPO) is a prominent policy gradient method known for its effectiveness and stability across a wide range of environments. PPO, introduced by OpenAI, is designed to address the complexities of policy optimization in RL[8]. PPO is a type of policy gradient algorithm that operates by directly optimizing the policy itself, as opposed to optimizing a value function from which the policy is derived. PPO modifies the policy gradient objective to minimize large updates, thus ensuring stable learning. This is achieved through the PPO-Clip objective. PPO employs an actor-critic approach, where the 'actor' updates the policy based on the calculated gradients, and the 'critic' estimates the value function. Clipping stabilizes the policy by using ϵ to dictate how far away the new policy can go from the old while still profiting the objective.

PPO's Probability Ratio measures the change in the policy

$$r_t(\theta) = \frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{\text{old}}}\left(a_t \mid s_t\right)} \tag{1}$$

where π_θ is the policy under the current parameters θ, a_t is the action at time t, and s_t is the state at time $t.\theta_{\text{old}}$ are the policy parameters before the update.

The objective function is given by:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right] \tag{2}$$

Here, $\hat{A}_t$ is an estimator of the advantage function at time t, and ϵ is a hyper-parameter (typically small, e.g., 0.1 or 0.2). The clip function ensures that the ratio $r_t(\theta)$ remains within the range $[1-\epsilon, 1+\epsilon]$, preventing large policy updates.

The advantage $\hat{A}_t$ can be computed using Generalized Advantage Estimation (GAE). GAE, introduced by Schulman et al., is a technique for estimating the advantage function that balances bias and variance.

$$\hat{A}_t^{GAE(\gamma,\lambda)} = \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}^V \tag{3}$$

Here, $\delta_t^V = r_t + \gamma V\left(s_{t+1}\right) - V\left(s_t\right)$ is the TD error at time t, γ is the discount factor, and λ is the GAE parameter that balances bias and variance.

When the advantage is negative its contribution to the objective is reduced to

$$L\left(s, a, \theta_k, \theta\right) = \max\left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)}, (1-\epsilon)\right) A^{\pi_{\theta_k}}(s, a). \tag{4}$$

the objective will increase if the action becomes less likely: $\pi_\theta(a \mid s)$ decreases. Max in this term puts a limit to how much the objective can increase. Once $\pi_\theta(a \mid s) < (1-\epsilon)\pi_{\theta_k}(a \mid s)$, it is capped at $(1-\epsilon)A^{\pi_{\theta_k}}(s, a)$. [8].

3.3 Deep Q-Learning

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Figure 2: DQN algorithm pseudocode [7]

The version of DQN that stablebaselines3 uses is based on the paper "Playing Atari with Deep Reinforcement Learning" [9][7]. Deep Q-Network (DQN), is an extension of the classic Q-learning algorithm with deep neural networks. DQN has been a fundamental algorithm in advancing the field of deep reinforcement learning, particularly after its success in mastering various Atari games.

The primary idea behind DQN is to use a deep neural network to approximate the Q-function. The Q-function $Q(s, a)$ represents the expected cumulative reward of taking action a in state s and following the optimal policy thereafter. In traditional Q-learning, this function is stored in a table, but in problems with large state spaces, such as video games or robotics, this becomes impractical. DQN solves this by using a neural network to estimate the Q-values.

The fundamental update rule in Q-learning is given by the Bellman equation:

$$Q^{\text{new}}\ (s_t, a_t) \leftarrow Q\left(s_t, a_t\right) + \alpha \left[r_{t+1} + \gamma \max_a Q\left(s_{t+1}, a\right) - \ Q\left(s_t, a_t\right)\right] \tag{5}$$

where s_t and a_t are the current state and action, r_{t+1} is the reward received after taking action a_t, γ is the discount factor, and α is the learning rate. $Q(s, a)$ is approximated by a neural network $Q(s, a; \theta)$, where θ are the parameters (weights) of the network. The network takes a state s as input and outputs Q-values for all possible actions.

The loss function for DQN, which is used to update the network weights, is defined as the mean squared error between the predicted Q-values and the target Q-values:

$$L(\theta) = \mathbb{E}\left[(y - Q(s, a; \theta))^2\right] \tag{6}$$

where $y = r + \gamma \max_{a'} Q(s', a'; \theta^-)$is the target Q-value, s' is the next state, and $\theta^-$are the weights of a separate target network.

DQN uses an experience replay mechanism and a target network to stabilize training. Experiences (s, a, r, s') are stored in a replay buffer, and the neural network is trained on a mini-batch of randomly sampled experiences from this buffer. The target network, with parameters θ^-, is a copy of the main network, but its weights are updated less frequently. This helps to stabilize the learning process by providing consistent targets for a while[7].

4 Experiment Design

4.1 Environment

This study was completed using FarmGym, a gym platform that provides the ability to create diverse agronomy games. The platform allows users to create games inspired from realistic challenges in agronomy and agroecology in a modular way. Each game is created by specifying a farm layout and implementing a subset of modules that define entities such as weather, soil, plant, weeds, pests, pollinators, etc.. Scoring functions, constraint and stopping conditions, and actions can all be customized for each game[6].

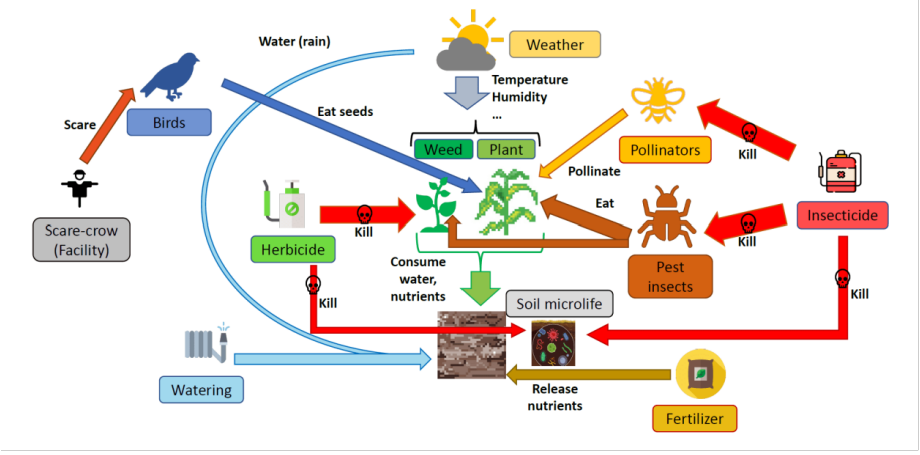

Figure 3: Entities dynamics map for FarmGym environment [6]

4.1.1 Scoring

Each farm has a specified cost function which defines the cost of each particular action/observation, and a reward/final reward function which calculates a reward using the state variables of the entities of each field. The cost and reward functions are called at each timestep (day), for each executed action and each field, and their result is added to form the daily reward. The final reward

function is only called at the end of the episode when the stopping conditions are triggered according to the user-defined rules of the game [6]. Farm-gym provides a basic-score module, that enables the user to specify the cost of each action and observation as well as set the entities to be considered for rewards using a yaml configuration file. The intervention (action) costs as well as the observation costs are defined in a yaml file, where a fixed value is assigned to any action that is applied or observation that is made. For all of the farms in this study, an observation cost of 0 was defined for all observation parameters. Watering and sowing seeds had a cost of 2 due to real-world resource costs, harvesting had a cost of 0 to encourage the agents to harvest the plants over doing nothing [6]. The daily rewards are 0, except when a plant moves from one development stage to another, in which case it is given the value 1. This is a very sparse reward signal. The final rewards are a subset of basic agronomic values, with weights provided in the configuration file. For the farms in this study, the Final harvest weight is a large positive reward signal while the Final resources added weight (water) is a smaller negative reward signal. This is meant to represent the cost of irrigation in real agricultural situations and push the agents to minimize irrigation where possible. The final reward is computed according to $r = \sum_j w_j r_j$ where $j \in \{$ bio, resource, soil, yield, plant $\}$. The $(w_j)_j$ are the weights specified in the yaml file. The r_j is computed based on the entities. - r_{resource} counts the total amount of water used during the interaction, - r_{yield} measures the total yield in kg of the harvest of all harvested plants [6]

4.1.2 Actions

For each action taken, the field and the plant for it to be taken on is specified in the yaml configuration file. The observation actions for all the farms included information about the weather such as temperature, humidity, sun, and rain, information about the soil such as the moisture content, the available nutrients, and the micro life, and information about the plants such as its size, stage, flowers, fruit weight, etc. The intervention actions for all of the farms in this study were limited to watering a discrete amount from the array [1L, 2L, 3L, 4L, 5L], sowing seeds every 10 cm, harvesting the plant, or micro-harvesting the plant. The farmer entity was limited to 1 action per field per day and 2 observations per field per day.

4.1.3 Monoculture Farm

The simple monoculture farm that the transfer learning agents were originally trained on was specified to have 3 plots all with loamy soil, and weather from Lille, France. All of the plots were growing beans.

Figure 4: Snapshot of experimental monoculture farm [6]

4.1.4 Different Monoculture Farm

The second monoculture farm that the transfer learning agents were tested in was specified to have 3 plots all with clay soil, and weather from Montpellier, France. All of the plots were growing corn.

Figure 5: Snapshot of experimental different monoculture farm [6]

4.1.5 Polyculture Farm

The polyculture farm that the agents were tested in was specified to have 3 plots all with loamy soil, and weather from Lille, France. Each plot was growing one of beans, corn, or tomatoes.

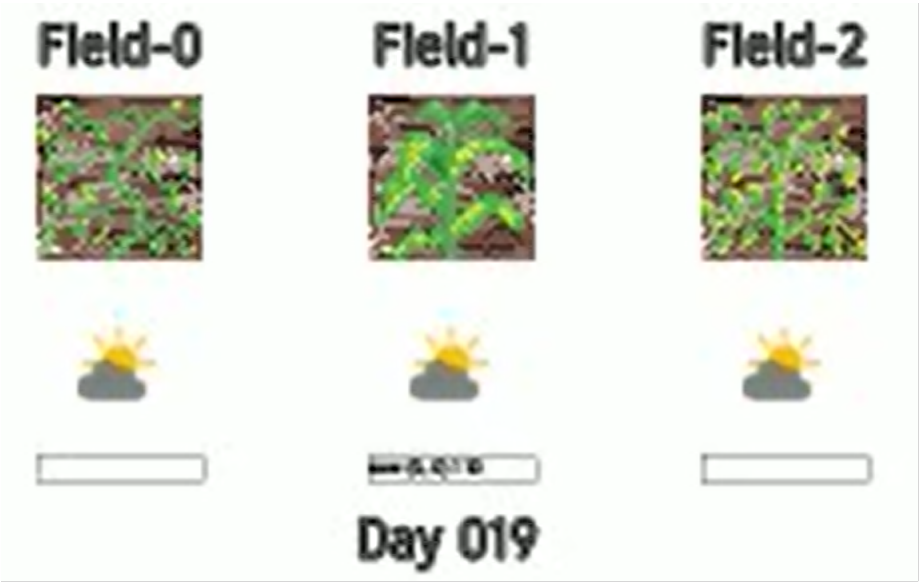

Figure 6: snapshot of experimental polyculture farm [6]

4.2 Algorithm Intialization

4.2.1 StableBaselines3 PPO

Parameters were set to the below defaults, parameter tuning was attempted using optuna and gpyopt, however, performance wasn't improved and computational cost made this optimization method infeasible.

"classstable$_b aselines3.ppo.PPO(policy = MlPolicy, env, learning_r ate = 0.0003, n_s teps = 2048, batch_s ize = 64, n_e pochs = 10, gamma = 0.99, gae_l ambda = 0.95, clip_r ange = 0.2, clip_r ange_v f = None, normalize_a dvantage = True, ent_c oef = 0.0, v f_c oef = 0.5, max_g rad_n orm = 0.5, use_s de = False, sde_s ample_f req = -1, rollout_b uf fer_c lass = None, rollout_b uf fer_k wargs = None, target_k l = None, stats_w indow_s ize = 100, tensorboard_l og = None, policy_k wargs = None, verbose = 0, seed = None, device =' auto',_i nit_s etup_m odel = True)$"[9]

4.2.2 StableBaselines3 DQN

Parameters were set to the below defaults, parameter tuning was attempted using optuna and gpyopt, however, performance wasn't improved and computational cost made this optimization method infeasible.

"classstable$_b aselines3.dqn.DQN(policy = MlPolicy, env, learning_r ate = 0.0001, buf fer_s ize = 1000000, learning_s tarts = 50000, batch_s ize = 32, tau = 1.0, gamma = 0.99, train_f req = 4, gradient_s teps = 1, replay_b uf fer_c lass = None, replay_b uf fer_k wargs = None, optimize_m emory_u sage = False, target_u pdate_i nterval = 10000, exploration_f raction = 0.1, exploration_i nitial_e ps = 1.0, exploration_f inal_e ps =

$0.05, max_grad_norm = 10, stats_window_size = 100, tensorboard_log = None, policy_kwargs = None, verbose = 0, seed = None, device =' auto',_i nit_setup_model = True)$" [9]

4.3 Training Procedure

Each agent was trained on the simple farm for a total of 100000 timesteps, where each timestep is a single day. The agents were then tested on the simple farm over 10 episodes. This procedure was repeated for 3 seeds and the training and testing rewards were averaged for each agent. A random trained instance of each agent was saved. These agents were then tested with no additional training on the different simple farm using the same procedure as before. These agents were then trained with an additional 100,000 timesteps on the new environment and evaluated again. New agents were created for the different farm environments trained for 100000 timesteps and evaluated as above. The saved agents from the simple farm were also tested with no additional training on the polyculture farm using the same procedure as before. These agents were then trained with an additional 100,000 timesteps on the new environment and evaluated again. New agents were created for the polyculture farm environment and trained for 100000 timesteps and evaluated as above.

5 Results

This study investigated the performance of reinforcement learning agents using Deep Q-Networks (DQN) and Proximal Policy Optimization (PPO) across three types of agricultural environments: simple monoculture, different monoculture, and polyculture farms. Additionally, the efficacy of transfer learning was assessed by transferring trained agents from a monoculture to a different monoculture and then a polyculture environment, with and without further tuning.

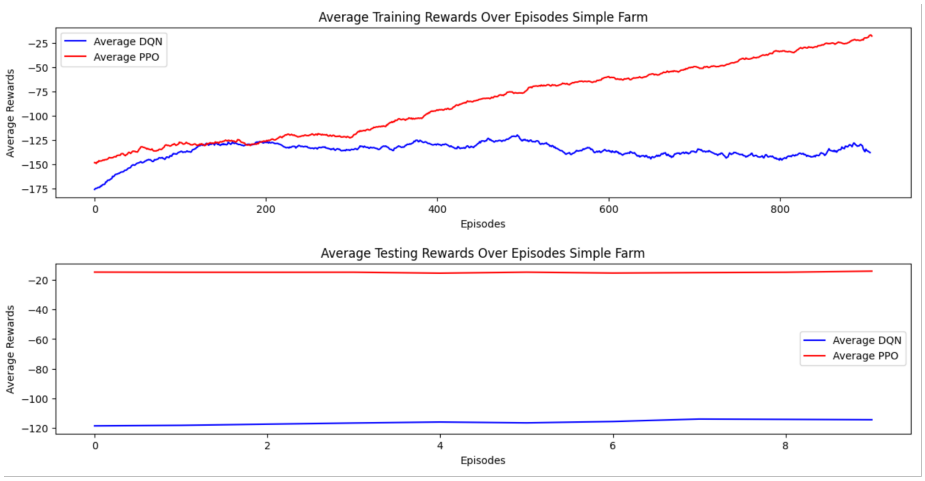

Figure 7: a) Average training rewards for a monoculture farm from DQN and PPO agents over 3 seeds with 100000 timesteps each. b)Average testing rewards for a monoculture farm from DQN and PPO agents over 3 seeds and 10 episodes.

In the simple farm environment, neither agent converged to an optimal policy. However, the average training rewards indicated a superior performance of PPO over DQN, with a higher and more stable convergence indicating a better fit for environments with straightforward dynamics. During testing, PPO consistently outperformed DQN, suggesting that PPO's policy optimization is more effective in simple, less variable environments.

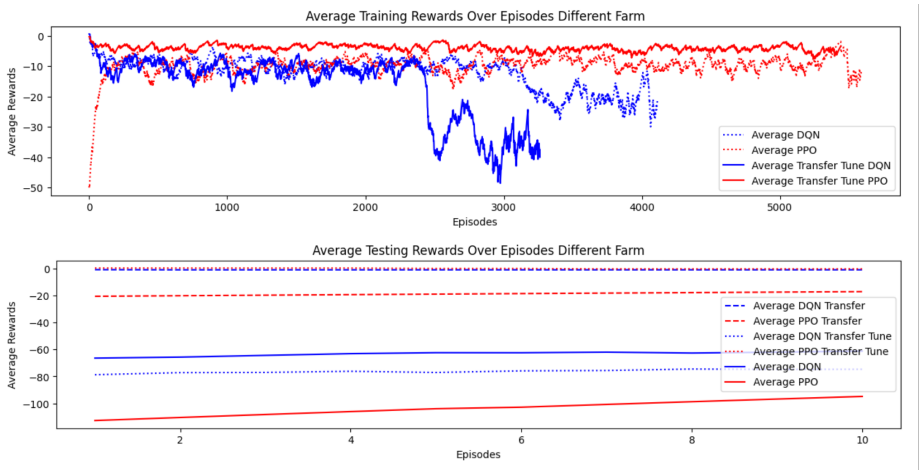

Figure 8: a) Average training rewards for a different monoculture farm from transferred DQN and PPO agents and new DQN and PPO agents over 3 seeds with 100000 timesteps each. b)Average testing rewards for a different monoculture farm from transferred DQN and PPO agents, transferred and tuned DQN and PPO agents, and new DQN and PPO agents over 3 seeds and 10 episodes.

In the different monoculture environment, which represents a simple agricultural setting with slightly different dynamics than the original farm, in training, both the baseline and the tuned transfer PPO agents outperformed the DQN agents. The DQN agents failed after approximately 60000 training timesteps. When the PPO agent trained in the monoculture environment was transferred to the different environment (referred to as 'transferred agents'), there was a noticeable improvement in the initial rewards, demonstrating the benefits of transfer learning. Moreover, when the transferred agent was fine-tuned in the different environment ('tuned transfer agents'), it outperformed the baseline PPO agent who learned from scratch, showcasing that even a small amount of additional learning on top of the transferred knowledge could boost performance.

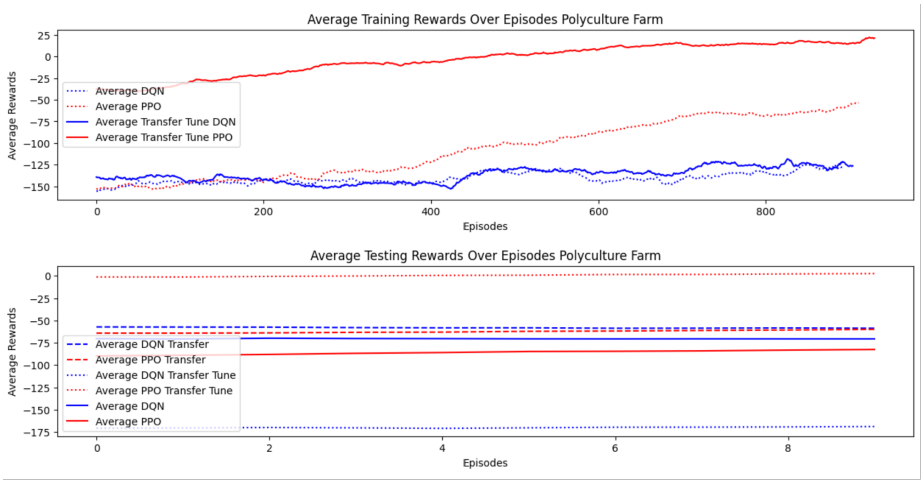

Figure 9: a) Average training rewards for a polyculture farm from transferred DQN and PPO agents and new DQN and PPO agents over 3 seeds with 100000 timesteps each. b)Average testing rewards for a polyculture farm from transferred DQN and PPO agents, transferred and tuned DQN and PPO agents, and new DQN and PPO agents over 3 seeds and 10 episodes.

In the polyculture farm environment, which represents a complex agricultural setting, both the baseline DQN and PPO agents started with lower training rewards due to the increased complexity. However, when agents trained in the monoculture environment were transferred to the polyculture environment (referred to as 'transferred agents'), there was a noticeable improvement in the initial rewards for both DQN and PPO, demonstrating the benefits of transfer learning. Moreover, when these transferred agents were fine-tuned in the polyculture environment ('tuned transfer agents'), they outperformed the baseline agents. The tuned transfer agents exhibited a faster adaptation and higher overall reward in the polyculture environment compared to both the baseline and the non-tuned transferred agents. This reveals the potential of transfer learning not just to kickstart learning in a new environment but also to achieve better performance with less data, which is crucial in real-world scenarios where data collection can be costly and time-consuming.

6 Conclusion

6.1 Discussion/Limitations

The exploration of reinforcement learning (RL) in agricultural decision-making systems, specifically using Deep Q-Networks (DQN) and Proximal Policy Optimization (PPO), has yielded promising results, particularly when combined with transfer learning strategies. However, several limitations warrant discus-

sion, particularly regarding the impact of sparse rewards and observation spaces on model performance.

Sparse Rewards: A significant challenge encountered during training was the sparse nature of rewards within the agricultural environments. Sparse rewards can lead to slower learning rates and may require a larger number of episodes for the agents to adequately converge to optimal policies. This is because the agents receive less frequent feedback on their actions, which is crucial for adjusting their strategies toward achieving higher yields. The sparse rewards may have contributed to the low performance of the DQN agents.

Observation Spaces: The complexity of the observation spaces is another factor that likely impacted model performance. Reinforcement learning agents are designed to process a larger and more complex set of inputs to make informed decisions. By adding in extra entities such as pests, weeds, and fertilizers or otherwise densifying the state space, the agent's performance might show improvement. It's possible that minimizing the observation cost somehow demotivated the agents from making thorough observations.

Convergence to Appropriate Rewards: The agents' convergence to appropriate reward levels is an ongoing challenge. While transfer learning has shown to provide a head start in learning within new environments, ensuring that the agents converge to truly optimal policies requires further refinement. The agents must not only achieve a high level of performance but also sustain it consistently. Additional work involving the fine-tuning of hyperparameters, the refinement of reward structures, or the addition of more complex entities and actions could be necessary to guide the agents toward faster and more stable convergence.

While the use of RL and transfer learning in agricultural systems is a burgeoning field with substantial potential, the issues of sparse rewards, complex observation spaces, and the need for improved convergence highlight the necessity for continued research and development in this domain.

6.2 Further Work

The current study has laid a framework for applying transfer learning in reinforcement learning (RL) to agricultural decision support systems, using environments of varying complexities. Building on these findings, there are several options for future work, which could be crucial for advancing the field and realizing the practical applications of transfer learning in RL in agriculture.

Incorporation of Additional Entities: Future iterations of this research should consider expanding the simulation environment to include additional entities that are representative of a real-world farm. This could include entities such as weeds, pollinators, pests, fertilizers, pesticides, hedges, etc. Including a wider range of entities would not only increase the realism of the simulation but also challenge the RL agents to develop more sophisticated policies that account for the interactions between these entities that are provided by FarmGym. This complexity mirrors the multifaceted nature of real-world agricultural ecosystems and economic models.

Scaling to Larger Farms: Another critical area of exploration is the application of transfer learning to larger-scale farms. The current study focused on smaller, more controlled environments. However, the efficiency of scaling RL decision support in agriculture could be greatly improved by transfer learning. Scaling up involves not just larger physical spaces but also more complex management decisions, such as the allocation of resources across vast areas and the strategic planning of planting and harvesting cycles.

Testing with Advanced RL Algorithms: While DQN and PPO provided significant insights into the RL capabilities in agriculture, other advanced algorithms could offer improvements. For instance, Double Deep Q-Networks (DDQN) could be evaluated for their ability to reduce the overestimation of action values and potentially provide more stable and accurate learning outcomes. Similarly, algorithms that utilize hindsight experience replay or those that can handle continuous action spaces, such as Deep Deterministic Policy Gradient (DDPG), could be tested.

Algorithmic Innovations for Sparse Rewards: Given the challenges presented by sparse rewards, future work should also look into algorithmic innovations specifically designed to tackle this issue. Techniques such as reward shaping, curriculum learning, and hierarchical reinforcement learning could offer new ways to accelerate learning in environments where feedback is infrequent. These methods could enable agents to learn from intermediate goals or leverage prior knowledge to form more complex behaviors.

In summary, the potential of transfer learning in RL in agriculture is vast, and the next steps should focus on broadening the scope of the environments and testing a wider range of algorithms.

References

[1] Y Avigal et al. Simulating polyculture farming to tune automation policies for plant diversity and precision irrigation. pages 238–245, 2020.

[2] Bruno Basso, John Antle, Davide Cammarano, and Kaylee Errecaborde. Artificial intelligence algorithms and models for sustainable agriculture production. *Agricultural Systems*, 188:103026, 2021.

[3] Abhishek Gupta, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. *arXiv preprint arXiv:1703.02949*, 2017.

[4] Andreas Kamilaris and Francesc X Prenafeta-Boldú. Deep learning in agriculture: A survey. *Computers and Electronics in Agriculture*, 147:70–90, 2018.

[5] Deborah K Letourneau, Inge Armbrecht, Beatriz Salgado Rivera, Jorge M Lerma, Eric J Carmona, Martha C Daza, Sara Escobar, Victoria Galindo, Carmen Gutierrez, Sandra Duque Lopez, et al. Does plant diversity benefit

agroecosystems? a synthetic review. *Ecological Applications*, 21(1):9–21, 2011.

[6] Odalric-Ambrym Maillard. Farm-gym: A modular reinforcement learning platform for agronomy games. 2022.

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[8] OpenAI. Proximal policy optimization algorithm. https://spinningup.openai.com/en/latest/algorithms/ppo.htmlpseudocode, 2019. Accessed: yyyy-mm-dd.

[9] Stable Baselines3 Contributors. Stable baselines3. https://github.com/DLR-RM/stable-baselines3, 2023. Accessed: yyyy-mm-dd.

[10] Matthew E Taylor, Peter Stone, et al. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(1):2125–2167, 2007.

[11] Alfons Weersink, Evan Fraser, David Pannell, Elizabeth Duncan, and Sarah Rotz. The role of big data in the agri-food system. *The Electronic Journal of Information Systems in Developing Countries*, 85(5):e12056, 2019.