Robust Model-based Reinforcement Learning for Autonomous Greenhouse Control

Wanpeng Zhang ZHANGWP19@MAILS.TSINGHUA.EDU.CN
Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen, China

Xiaoyan Cao HOLMESCAOXY@GMAIL.COM
Department of Informatics, Xiamen University, Xiamen, China

Yao Yao Y-YAO19@MAILS.TSINGHUA.EDU.CN
Zhicheng An AZC19@MAILS.TSINGHUA.EDU.CN
Xi Xiao XIAOX@SZ.TSINGHUA.EDU.CN
Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen, China

Dijun Luo DIJUNLUO@TENCENT.COM
Tencent AI Lab, Shenzhen, China

Editors: Vineeth N Balasubramanian and Ivor Tsang

Abstract

Due to the high efficiency and less weather dependency, autonomous greenhouses provide an ideal solution to meet the increasing demand for fresh food. However, managers are faced with some challenges in finding appropriate control strategies for crop growth, since the decision space of the greenhouse control problem is an astronomical number. Therefore, an intelligent closed-loop control framework is highly desired to generate an automatic control policy. As a powerful tool for optimal control, reinforcement learning (RL) algorithms can surpass human beings' decision-making and can also be seamlessly integrated into the closed-loop control framework. However, in complex real-world scenarios such as agricultural automation control, where the interaction with the environment is time-consuming and expensive, the application of RL algorithms encounters two main challenges, i.e., sample efficiency and safety. Although model-based RL methods can greatly mitigate the efficiency problem of greenhouse control, the safety problem has not got too much attention. In this paper, we present a model-based robust RL framework for autonomous greenhouse control to meet the sample efficiency and safety challenges. Specifically, our framework introduces an ensemble of environment models to work as a simulator and assist in policy optimization, thereby addressing the low sample efficiency problem. As for the safety concern, we propose a sample dropout module to focus more on worst-case samples, which can help improve the adaptability of the greenhouse planting policy in extreme cases. Experimental results demonstrate that our approach can learn a more effective greenhouse planting policy with better robustness than existing methods.

Keywords: Reinforcement Learning; Agricultural Automation

1. Introduction

The traditional agricultural production mode is highly dependent on the weather, which has been unable to meet the increasing demand for fresh and healthy food from global population growth. Around the world, some countries have gradually regarded the greenhouse

industry as the main force in agricultural production Rijswick (2018). Modern high-tech greenhouses are equipped with standard sensors and actuators (such as heating, lighting, CO_2 dosing, irrigation, etc.) to empower precision agriculture. To improve crop yield and quality, managers regularly regulate a suitable environment for crop growth by overseeing the greenhouse climate and crop growth state. In addition to increasing the harvest, the corresponding energy consumption is another key consideration since natural resources are facing the challenge of exhaustion George et al. (2018). Although the automatic greenhouse is an ideal solution to deal with the food crisis, skilled managers capable of autonomous greenhouse control are scarce Sparks (2018). Furthermore, even a seasoned manager is not able to monitor and manage too many greenhouses simultaneously.

To provide a favorable climate for crop growth in a modern high-tech greenhouse, growers need to manually determine control strategies, such as lighting and irrigation, according to their planting experience. Then the strategy is fed into the process computer to take effect in the greenhouse. Sensors will continuously measure the climate and crop growth state, and feedback the measured data to the grower for analysis and decision-making. The grower needs to balance production and resource consumption during a 3-5 months period Hemming et al. (2020), which implies a tremendous decision-making space. The complexity of decision-making has led to growers only giving coarse-grained control strategies, which do not make full use of the rich greenhouse states information.

With breakthroughs in AI, new technologies have been introduced into agriculture to help improve production, such as automatic pruning fruit trees You et al. (2020), intelligent spraying systems Kim et al. (2020), cooperative control Li et al. (2021). RL has shown the potential to outperform humans in decision making via enabling both deep decision search at the macro-level and fine-grained control at the micro-level, making it well suited for automated control scenarios. By formulating autonomous greenhouse control as a Markov decision process (MDP) problem, there has been some research application of RL algorithm in an agricultural scene Parameswaran and Sivaprasath (2016); Wang et al. (2020) .

However, RL still faces two major challenges in practical application. First is the sample efficiency problem. Although RL have been achieved excellent results in some fields, they often rely on huge training samples, which is not practical in the real world. Since there is no trial-and-error cost in a virtual simulation environment, algorithms are free to perform exploratory learning and thus learn strategies that can circumvent wrong decisions. However, in the real world, where much trial-and-error implies huge costs (e.g., machine damage, crop death), it becomes a major challenge to learn decisions that can circumvent errors within limited training samples. Second, the robustness of RL is also a key challenge in real-world application scenarios. Once the environment is perturbed during the training phrase, the algorithm performance may be affected and degrade significantly. Furthermore, in the deployment phase, the inconsistency between the deployment environment and the training environment can also affect the performance of the trained strategy.

In this paper, we investigate how RL can be better applied to autonomous greenhouse control. To address the sample efficiency challenge, we introduce the model ensemble approach. In this approach, samples from the real environment are not directly handed over to the RL algorithm for policy optimization but are used to model the environment. Then the model is used to simulate the environment to add more simulated samples and accelerate the learning efficiency. To further enhance the safety of the automated planting policy, we

add a sample dropout module to the RL algorithm. The module enable our algorithm to selectively discard a portion of samples with excessive reward, to focus more on worst-case samples, improving the adaptability of the planting policy in extreme cases, and solve the safety challenge of RL in a real-world deployment.

2. Related Work

2.1. AI for Agriculture

Agriculture is an area of extreme importance, which faces several challenges from sowing to harvest Bannerjee et al. (2018). With the rapid development of AI in recent years Huang and Rust (2018), more and more AI technologies are being applied in agricultural automation Jha et al. (2019).

During crop growing, crop disease is a matter of grave concern to a farmer. Significant expertise and experience are required to detect an ailing plant and take the necessary steps for recovery. Ghaffari et al. develop an electronic nose, and intelligent systems approach to detect diseases in tomato crops Ghaffari et al. (2010). Issues on soil and irrigation management are also very vital in agriculture. Improper irrigation and soil management lead to crop loss and degraded quality. Manek and Singh compared several neural network architectures to predict rainfall using four atmospheric inputs Manek and Singh (2016). Dahikar and Rode proposed a neural model to predict different crop yields using atmospheric inputs and fertilizer consumption Dahikar and Rode (2014).

Most agricultural automation works like the ones mentioned above are challenging to integrate into a holistic system. A macro-level centralized planning system for controlling entire farms remains under study. RL algorithms are often used to learn control policies in complex environments with the potential to outperform humans Schulman et al. (2015), as long as they have comprehensive observation information about the entire environment. An automated irrigation system is developed RL technique to control greenhouse remotely to minimize human involvement Parameswaran and Sivaprasath (2016). RL algorithms are also used to optimize the autonomous greenhouse climate control to resource consumption Wang et al. (2020).

Also, some works on agriculture with robotics as well, where mobile robots are internally powered by some AI algorithms to facilitate their jobs. In our opinion, this direction of research is promising in the sense that, conceivably, autonomous robots can significantly boost the management of large-scale crop farms as human labor becomes increasingly expensive and conventional machines are not intelligent enough. A few works down the line are Hagras et al. (2002) and Zhou et al. (2014) where they discuss online learning of robots and visual navigation of mobile robots.

2.2. Challenges of RL in Agricultural Applications

In recent years, RL has achieved remarkable results in a wide range of areas, including simulated control problems Levine et al. (2016), outperforming human performances on Go and games Mnih et al. (2015); Silver et al. (2016). However, applying reinforcement learning to agricultural applications requires addressing the two major challenges of sample efficiency and robustness.

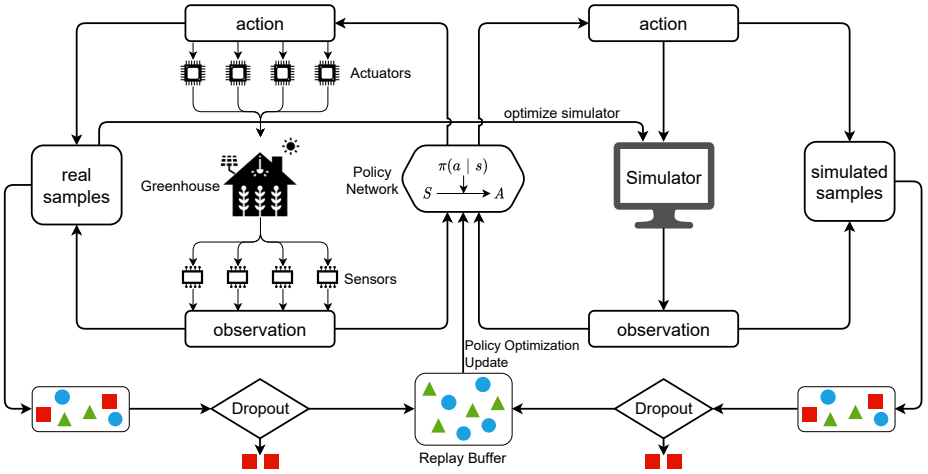


Figure 1: A modular framework for greenhouse automation. It mainly consists of a greenhouse environment, a crop growth simulator and the planting algorithm. The core part is the policy module, which receives observations to decide the next action. The RL algorithm uses samples to optimize policies continuously. These three parts alternate in a cycle that eventually leads to an automated planting policy.

Model-based methods have shown excellent abilities to reduce the sample complexity Deisenroth et al. (2013). Previous works Levine et al. (2016); Chua et al. (2018); Janner et al. (2019) have empirically shown significant sample efficiency improvements. However, model accuracy is a major barrier for policy quality, and it is challenging to build an accurate model in high-dimensional tasks Abbeel et al. (2006). Thus the policy learned on inaccurate models typically leads to performance degradation due to cumulative model error Sutton (1996); Asadi et al. (2019). While improving sample efficiency, the control policy is affected by the discrepancy between the simulator and the real environment. For this problem, previous works (e.g. PETS Chua et al. (2018), ME-TRPO Kurutach et al. (2018), SLBO Luo et al. (2018), MB-MPO Clavera et al. (2018)) use ensembles of bootstrapped probabilistic transition models to properly incorporate two kinds of uncertainty into the transition model. Concretely, individual probabilistic models capture aleatoric uncertainty or the noise due to the inherent stochasticity. The bootstrapping ensemble procedure can capture epistemic uncertainty or uncertainty in the model parameters aroused from insufficient training data. Empirical works Levine et al. (2016); Janner et al. (2019); Yao et al. (2021) have demonstrated that the ensemble of probabilistic models is an efficient way to handle both of two uncertainties, allowing for a competitive model-based learning algorithm.

Moreover, the robustness issue of RL is one of the challenges hindering its application to complex tasks. Though current state-of-the-art model-based methods have achieved outstanding performance, the derived policies are often only adequate for the environment in which they are trained, and when deployed to perturbed real environments, the policy performance tends to degrade dramatically or even behave dangerously. Robust control Zhou and Doyle (1998) is a branch of control theory focused on finding optimal policy

under worst-case situations. Policy learned through robust control has better generalization performance. Robust adversarial RL methods, such as the RARL Pinto et al. (2017) and NR-MDP Tessler et al. (2019) methods, learn robust decision policy by adversarially adding minimax objects, while EPOpt method introduces conditional value at risk (CVaR) Tamar et al. (2015); Rajeswaran et al. (2016), by optimizing the CVaR object, a robust policy with better performance is obtained. We follow these ideas and design a sample dropout module that improves safety in tomato greenhouse automation.

3. Notations and Preliminaries

We consider a Markov decision process (MDP), defined by $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$ where $\mathcal{S} \in \mathbb{R}^{d_s}$ is the state space, $\mathcal{A} \in \mathbb{R}^{d_a}$ is the action space, $r(s,a) : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function, $\gamma \in [0,1]$ is the discount factor, and $\mathcal{P}(s'|s,a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0,1]$ is the probability distribution of the next state given current state s and action a, or use the form $s' = \mathcal{P}(s,a) : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ denotes the state transition function when the environment is deterministic.

Let $V^{\pi_\theta,\mathcal{P}}(s)$ denotes the expected return or expectation of cumulative rewards starting from initial state s, i.e., the expected sum of discounted rewards following policy $\pi_\theta(a|s)$ and state transition function $\mathcal{P}(s,a)$:

$$V^{\pi_\theta,\mathcal{P}}(s) = \mathop{\mathbb{E}}_{\{a_0,s_1,\ldots\}\sim\pi_\theta,\mathcal{P}}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t) \mid s_0 = s\right] \tag{1}$$

For simplicity of symbol, let $V^{\pi_\theta,\mathcal{P}}$ denotes the expected return over random initial states: $V^{\pi_\theta,\mathcal{P}} = \mathop{\mathbb{E}}_{s_0\in\mathcal{S}}\left[V^{\pi_\theta,\mathcal{P}}(s_0)\right]$. The goal of RL is to maximize the expected return by finding the optimal decision policy

$$\pi_\theta^* = \arg\max_{\pi_\theta} V^{\pi_\theta,\mathcal{P}} \tag{2}$$

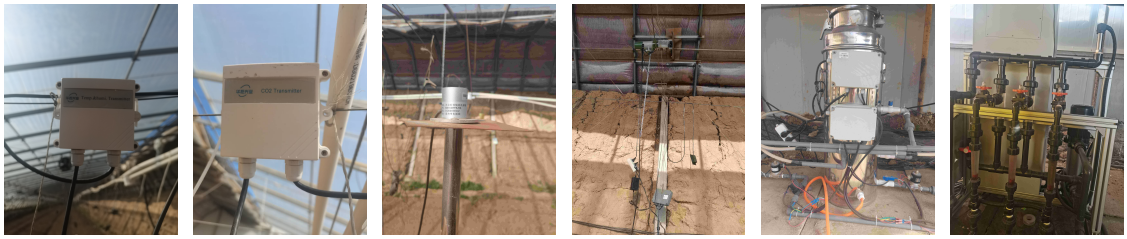
In model-based RL, an approximated transition model $\mathcal{M}(s,a)$ is learned by interacting with the environment, the policy $\pi(a|s)$ is then optimized using the model-free method with samples from the environment and data generated by the model. We use the parametric notation $\mathcal{M}_\phi, \phi \in \Phi$ to specifically denote the model trained by a neural network, where Φ is the parameter space of models.

4. Proposed Approach

4.1. System overview

We propose the framework shown in the Fig. 1, which mainly consists of three components that can be executed asynchronously:

1. The collection of real samples through a real agricultural greenhouse environment and a large number of IoT hardware devices.
2. The generation of samples through rapid simulation in a tomato growth simulator.
3. Based on the observation samples, the RL algorithm is used to continually train and optimize the policy, which makes decisions based on current information and performs the next action to obtain new observations, leading to the next cycle.



(a) (b) (c) (d) (e) (f)

Figure 2: Equipments in our greenhouse. (a)temperature sensor. (b)CO_2 sensor. (c)PAR sensor. (d)ventilation controller. (e)CO_2 producer. (f)fertigation controller.

Specifically, we use multiple sensors to obtain observations, such as temperature, humidity, CO_2 concentration, etc. Here are the details of how the main sensors work:

- Temperature, humidity and CO_2 sensor: multiple measuring boxes are hung in the greenhouse and their readings are averaged (sometimes with different weighting factors for different locations) to get the parameters of temperature, humidity, and CO_2 as the feed for heating, humidity, ventilation, and CO_2 supply.
- PAR sensor: the greenhouse keeps track of the supply of PAR parameter by measuring the Photosynthetically Active Radiation. In the simulated greenhouse, a PAR sensor is modeled to describe the PAR-level just above the crop.

The observation samples are then transmitted to the decision-making policy module, which can generate corresponding actions. These actions are executed in the greenhouse through multiple actuators. Here are the details of how the main actuators work:

- Ventilation controller: when the greenhouse temperature exceeds the ventilation setpoint, the vents will be opened to a certain extent. The vents' opening angle is expressed as a percentage, and it is determined by the deviation between the current greenhouse temperature and the temperature-setpoint.
- CO_2 producer: when the CO_2 concentration drops below the setpoint, the CO_2 producer will supply CO_2 through a piping network.
- Fertigation controller: The crops will be automatically irrigated by drip irrigation according to the related setpoints, including irrigation time and watering quantity.

All the observation and action samples are stored in the data cache to optimize both the simulator and policy.

4.2. Algorithm

Although RL is a powerful tool for policy learning, it faces a significant challenge in training efficiency, often relying on a large number of interactions with the environment. Based on the data collected, we developed a tomato growth simulator close to the real environment.

Algorithm 1: Robust Model-based RL for Autonomous Greenhouse Control

Function `Dropout(`$\mathcal{B}$, p`)`**:**

Calculate $r_p(\mathcal{B})$: the p-percentile of batch $\mathcal{B}$

for $x \in \mathcal{B}$ **do**

if $r(x) \leq r_p(\mathcal{B})$ **then**

Fill x into $\mathcal{B}_{\text{drop}}$

end

end

return $\mathcal{B}_{\text{drop}}$

Function `Main`**:**

Initialize hyperparameters, policy π_θ, environment replay buffer $\mathcal{D}_{\text{real}}$, simulator replay buffer $\mathcal{D}_{\text{sim}}$

for N_{epoch} *iterations* **do**

Take an action in the greenhouse environment using policy π_θ; add samples to $\mathcal{D}_{\text{real}}$

Mask $\mathcal{D}_{\text{real}}$ into $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_{\text{N}}\}$

for N_{train} *iterations* **do**

Load the pre-trained model and train on $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_{\text{N}}\}$

Get the trained model ensemble collection $\mathcal{M} = \{\mathcal{M}_{\phi_1}, \mathcal{M}_{\phi_2}, \ldots, \mathcal{M}_{\phi_N}\}$ with clip restriction

for $t = 1, 2, \ldots, T$ **do**

Select a model $\mathcal{M}_t$ from $\mathcal{M}$ with probability $\Pr\{\mathcal{M}_t = \mathcal{M}_{\phi_i} \mid i \sim P_{\mathcal{M}}, i \in \{1, 2, \ldots, N\}\}$

Perform rollouts on model $\mathcal{M}_t$ with policy π_θ and get samples $x = (s_{t+1}, s_t, a_t)$

Fill these samples into batch $\mathcal{B}^{\pi_\theta, \mathcal{M}}$

end

$\mathcal{B}_p^{\pi_\theta, \mathcal{M}}$ = `Dropout(`$\mathcal{B}^{\pi_\theta, \mathcal{M}}$, p`)`; Fill the data of $\mathcal{B}_p^{\pi, \mathcal{M}^\alpha}$ into $\mathcal{D}_{\text{sim}}$

end

$\mathcal{B}_p^{\pi_\theta}$ = `Dropout(`$\mathcal{D}_{\text{real}}$, p`)`; $\mathcal{D}_{\text{real}} = \mathcal{B}_p^{\pi_\theta}$

Optimize policy π_θ on $\mathcal{D}_{\text{sim}}$ and $\mathcal{D}_{\text{real}}$: $\theta \leftarrow \theta - \lambda \nabla_\theta V_p^{\pi_\theta, \mathcal{M}}(\mathcal{D}_{\text{sim}}; \mathcal{D}_{\text{real}})$

end

return

The simulator allows us to rapidly perform action interactions and generate simulation samples that simulate a tomato crop's entire growth cycle within seconds. These samples are then provided to RL algorithms for policy optimization.

Specifically, we firstly pre-train a model based on the cumulative greenhouse crop data, i.e., $\mathcal{D}_{\text{real}}$, to express the relationship between crop growth and the parameters of greenhouse. The initialization of model parameters is based on expert knowledge, which can effectively alleviate the sample consumption in the early stage of model learning. Besides, we add restrictions to the simulation states based on prior knowledge in agriculture to avoid unreasonable anomalies (e.g., excessive temperature and CO_2 concentration) due to model generalization, enabling the simulator to simulate more realistic and complex situations and improve the robustness of the learned planting policy, thus improving the adaptability of the policies to a variety of complex scenarios. And with the idea of Dyna-style RL (Sutton

(1991)), we continually transmit the collected real samples to the simulator for further optimization of the tomato growth model in the policy optimization cycle, which enables the simulator to provide simulation samples for the framework more efficiently and accurately. We also use the model ensemble approach in order to capture the uncertainty in the real greenhouse environment.

We first build the binary masks $h_{i,j}$ from the Bernoulli distribution with parameter $p \in (0, 1]$, and we perform the bootstrap mask

$$H_j = \{h_{i,j} \sim \text{ Bernoulli } (p) \mid i \in \{1, \ldots, N\}\} \tag{3}$$

on each sample data $d_j \in \mathcal{D}_{\text{real}}$, then we can generate N subsets: $\mathcal{D}_i = \{h_{i,j} \bigodot d_j \mid d_j \in \mathcal{D}_{\text{real}}\}$ where $\bigodot$ indicates that d_j is retained in the set $\mathcal{D}_i$ if $h_{i,j} \neq 0$, otherwise not. And then, based on the pre-trained simulator, we use the new data from the subsets to fine-tune the model. We learn a collection of fine-tuned simulator models $\mathcal{M} \doteq \{\mathcal{M}_{\phi_1}, \mathcal{M}_{\phi_2}, \ldots, \mathcal{M}_{\phi_N}\}$. We use parametric notation $\mathcal{M}_\phi, \phi \in \Phi$ to specifically denote the model trained by neural network, where Φ is the parameter space of models. Each member of the collection $\mathcal{M}$ is a probabilistic neural network whose outputs $\mu_{\phi_i}, \sigma_{\phi_i}$ parametrize a Guassian distribution:

$$s' = \mathcal{M}_{\phi_i}(s, a) \sim \mathcal{N}(\mu_{\phi_i}(s, a), \sigma_{\phi_i}(s, a)) \tag{4}$$

The corresponding loss of the simulator models is

$$\mathcal{L}_{\mathcal{M}_{\phi_i}} = \sum_{t=1}^{N} [\mu_{\phi_i}(s_t, a_t) - s_{t+1}]^\top \sigma_{\phi_i}^{-1}(s_t, a_t)[\mu_{\phi_i}(s_t, a_t) - s_{t+1}] + \log |\sigma_{\phi_i}(s_t, a_t)| \tag{5}$$

In our method, the simulator is learned to replace the real environment $\mathcal{P}$. The policy $\pi_\theta(a|s)$ is then optimized using the samples from both the environment and simulator. Optimizing the expected return in a general way as RL methods allows us to learn a policy that performs best in expectation over the simulator. However, best expectation doesn't mean that the result policies can perform well in each individual model, since there could be high variability in performance for different models. This instability typically leads to risky decisions when facing poorly-informed states at deployment.

Inspired by previous works Tamar et al. (2015); Rajeswaran et al. (2016) which optimize conditional value at risk (CVaR) to explicitly seek a robust policy, we add a sample dropout module to the RL algorithm, which selectively discards a portion of samples with excessive reward, to focus more on worst-case samples, improving the adaptability of the planting policy in extreme situations, and solve the safety challenges of RL in real-world deployment, aiming to further enhance the safety of the automated planting policy.

More specifically, to generate a prediction from our model collection, we first select a model $\mathcal{M}_t$ with probability

$$\Pr\{\mathcal{M}_t = \mathcal{M}_{\phi_i} \mid i \sim P_{\mathcal{M}}, i \in \{1, 2, \ldots, N\}\} \tag{6}$$

at each time step t, where $P_{\mathcal{M}}$ is a probability distribution (defaults to a random distribution). We then use the model $\mathcal{M}_t$ to interact with the policy π_θ and perform a simulated rollout using the selected model, i.e., $s_{t+1} \sim \mathcal{M}_t(s_t, a_t), a_{t+1} \sim \pi_\theta(a|s_t)$. In this way, we can quickly generate a large number of crop growth simulation samples, and our simulator

can be improved as we receive real samples from the greenhouse during the RL cycle. Then we fill these rollout samples $x = (s_{t+1}, s_t, a_t)$ into a batch and retain a p-percentile dropout subset with more pessimistic rewards. We use $\mathcal{B}_p^{\pi_\theta,\mathcal{M}}$ to denote the p-percentile dropout rollout batch:

$$\mathcal{B}_p^{\pi_\theta,\mathcal{M}} = \left\{x | x \in \mathcal{B}^{\pi_\theta,\mathcal{M}}, r(x) \leq r_p(\mathcal{B}^{\pi_\theta,\mathcal{M}})\right\} \tag{7}$$

where

$$\mathcal{B}^{\pi_\theta,\mathcal{M}} = \left\{x | x \triangleq (s_{t+1}, s_t, a_t) \sim \pi_\theta, \mathcal{M}\right\} \tag{8}$$

is the sample batch collected by performing policy π_θ on the model ensemble $\mathcal{M}$ and $r_p(\mathcal{B}^{\pi_\theta,\mathcal{M}})$ is p-percentile of reward values in batch $\mathcal{B}^{\pi_\theta,\mathcal{M}}$. The expected return of dropout batch rollouts is denoted by $V_p^{\pi_\theta,\mathcal{M}}$:

$$V_p^{\pi_\theta,\mathcal{M}} = \mathbb{E}\left[\sum\nolimits_{\{s_0,a_0,\ldots\}\sim\mathcal{B}_p^{\pi_\theta,\mathcal{M}}}\left[\gamma^t r(s_t, a_t)\right]\right] \tag{9}$$

Then, we can perform policy gradient update by

$$\boldsymbol{\theta}' = \boldsymbol{\theta} - \lambda\nabla_{\boldsymbol{\theta}} V_p^{\pi_\theta,\mathcal{M}} \tag{10}$$

Where λ is the learning rate. The overall pseudo code is shown as Algorithm 1.

5. Experiments

With the sensors and actuators deployed in our tomato greenhouse, we collect 22 kinds of observation variables, constituting a 275-dimensional observation space $\mathcal{S}$, and 6 kinds of control variables, constituting a 52-dimensional action space $\mathcal{A}$. Also, we use the $Netprofit$ (USD/m^2) as the target reward for training. $Netprofit = Gains - Costs$, where $Gains$ are obtained through yields and price, and $Costs$ include resource consumption (electricity, heat, CO_2, and water) and crop maintenance costs. The details of observation space and action space are shown in the Table 1. Since it's difficult to have a ground truth simulator in a complex environment like the greenhouse. However, due to the long experiment period in real world, we have to consider using the simulator as the test environment (different from our pre-trained models). We use a commercial simulator designed by human experts, which simulates the real greenhouse environment as much as possible, but it can only be used for testing due to low computational efficiency. Since the simulator is a black-box model independent of the training environment, it's reasonable and fair for testing.

5.1. Analysis of Performance

We train two versions of our method on the greenhouse simulator, one with sample dropout ($p = 0.8$, the choice of parameter p will be analyzed in section 5.3) and one without sample dropout ($p = 1$). Additionally, we adopt the soft actor-critic (SAC) algorithm, a widely used model-free RL algorithm, as a baseline for comparison. With $horizon = 120$, these algorithms are trained with a 120-day $Netprofit$ as an optimization target.

As shown in the left of Fig. 3, the algorithm with dropout converges better and has less variance than the one without dropout. This is mainly because the sample dropout module will discard a portion of samples with excessive feedback values, avoiding the local

Table 1: Observation Space (Above) and Action Sapce (Below)

Name	Min	Max	Dim
temperature setpoint	13	32	24
CO_2 setpoint	400	1000	24
light-on time	0	24	1
light-off time	0	24	1
irrigation start time	0	24	1
irrigation stop time	0	24	1
outside solar radiation	0	2000	24
outside temperature	-30	50	24
outside humidity	0	100	24
wind speed	0	25	24
virtual sky temperature	-20	20	24
greenhouse air temperature	-30	100	24
greenhouse air humidity	0	100	24
greenhouse air CO_2 concentration	400	1000	24
light intensity just above crop	0	2000	24
cumulative amount of irrigation per day	0	10	1
cumulative amount of drain per day	0	10	1
leaf area index	0	10	1
current number of growing fruits	0	1000	1
cumulative harvest in terms of fruit fresh weight	0	100	1
cumulative harvest in terms of fruit dry weight	0	100	1
planting days	0	365	1
temperature setpoint	13	32	24
CO_2 setpoint	400	1000	24
light on time	0	24	1
light off time	0	24	1
irrigation start time	0	24	1
irrigation stop time	0	24	1

optimum. Meanwhile, the agent pays more attention to the worst-case states so that its variance is smaller. As for the SAC algorithm, it performs worse than our algorithm, which is caused by the low sample efficiency of the model-free method, making it difficult to learn enough information with limited samples.

Further, we evaluate the planting policies learned by the different algorithms on the tomato simulator, as shown in the right of Fig. 3. We observe that: (1) the policies have similar performance in the early stage when the crop is not growing; (2) when the crop starts to harvest, our algorithm outperforms both the SAC algorithm and skilled labor.

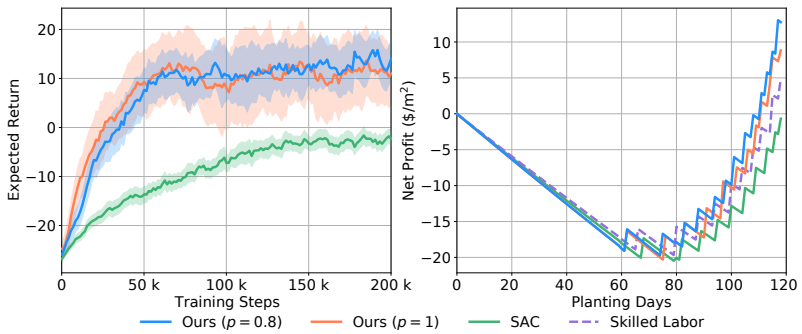


Figure 3: (1) The left plot shows the training curves for our method with dropout ($p = 0.8$), our method without dropout ($p = 1$) and the SAC baseline (the original version). Each with 5 seeds for training. Solid curves indicate the mean of all trials with different seeds. Shaded regions correspond to standard deviation among trials; (2) The right plot shows the evaluation curves of the policies trained by all the algorithms. The purple dashed line is the reference profit for skilled labor.

5.2. Analysis of Robustness and Safety

In order to verify the robustness improvement from the sample dropout module in our algorithm, we design a set of anti-disturbance experiments by perturbing the temperature (°C) in the interval $[25, 31]$ and the CO_2 concentration (ppm) in the interval $[400, 1000]$. We test the algorithm with and without dropout separately (shown in Fig. 4). In the heat map, each pixel represents the expected reward of the algorithm after training the same number of steps in each perturbed environment. Moreover, the closer the color to red (hotter) means a higher reward, and vice versa. Obviously, the algorithm without dropout can only achieve a normally expected reward in a small area closer to the standard area. In contrast, the algorithm with dropout can maintain a higher expected reward in the more disturbed area, demonstrating that the sample dropout module can improve the robustness of the algorithm.

To further analyze the benefits of dropout, we set outside solar radiation (Iglob), greenhouse air temperature (AirT), and greenhouse air humidity (AirRH) as anomalous parameters, which are the critical factors to crop growth. According to these anomalous parameters, we test our algorithms in the simulator. Additionally, we use the fresh weight and retention rate of crops as indicators to evaluate the algorithm performance. The setting of anomalous parameters and the experimental results are shown in Table 2.

Based on the results in Table 2, we find that the algorithm with dropout has a higher fresh weight and retention rate under anomalous conditions, which shows higher safety, thus more promising for real-world applications.

5.3. Analysis of Hyperparameter

In this section, we investigate the sensitivity of our algorithm to the hyperparameter p (details in Equation 7). We vary the parameter p from 1.0 to 0.6, representing the rate of discarded samples. Further downward adjustment of the parameter p is no longer worth

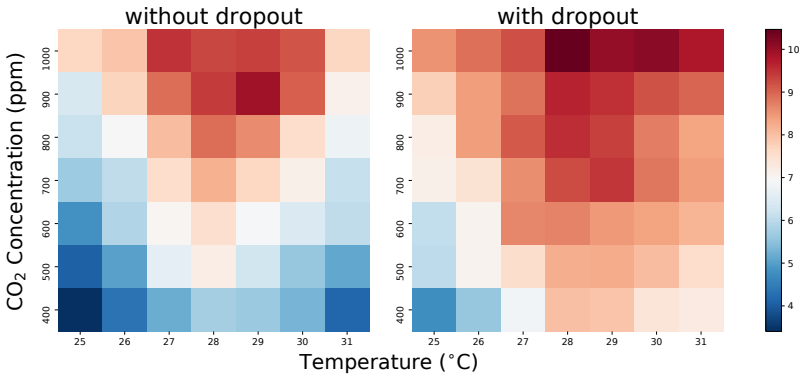


Figure 4: The robustness performance is depicted as heat maps for various environment settings. The algorithms without ($p = 1.0$) and with ($p = 0.8$) dropout are conducted separately. Each pixel in the heat map represents the average reward in one specific experiment. The closer the color to red (hotter) means the higher reward, the better the algorithm performance in that environment, and vice versa. Each experiment stops after 500,000 steps.

Table 2: Exception Test

Parameter	**Fresh weight**		**Retention rate**	
	$p = 1$	$p = 0.8$	$p = 1$	$p = 0.8$
AirT $\in (35, 40)$	38.51	**45.24**	80.23%	**85.35%**
AirT $\in (-2, 10)$	30.31	**38.49**	63.14%	**72.62%**
AirRH $= 90$	32.69	**39.30**	68.10%	**74.16%**
Iglob $= 0$	36.76	**43.71**	76.59%	**82.47%**

investigating, which is difficult for the algorithm to obtain enough information. This conclusion is confirmed in the following experiments.

Firstly, We test the algorithm with dropout with different p values ($p = 1.0$ for no dropout) in the standard environment for multiple sets of experiments. The results are shown in the left of Fig. 5. We observe that when $p \in \{0.8, 0.9, 1.0\}$, the corresponding $Netprofit$ are close, implying that the algorithms have similar performance under these parameters. When $p < 0.8$, it can be seen that the performance of the algorithm decreases significantly.

Next, we test the algorithm under different p values with different disturbed environments. Then we take the mean value of these results as the final result. A larger mean value means better robustness of the algorithm. Specifically, we set up four different disturbed environments, controlling the temperature $\in \{26, 30\}$ and the CO_2 concentration $\in \{500, 900\}$. The results are shown in the right of Fig. 5. We observe that the optimal parameter value is 0.8. Moreover, a similarly significant decrease in the robustness of the algorithm starts when $p < 0.8$.

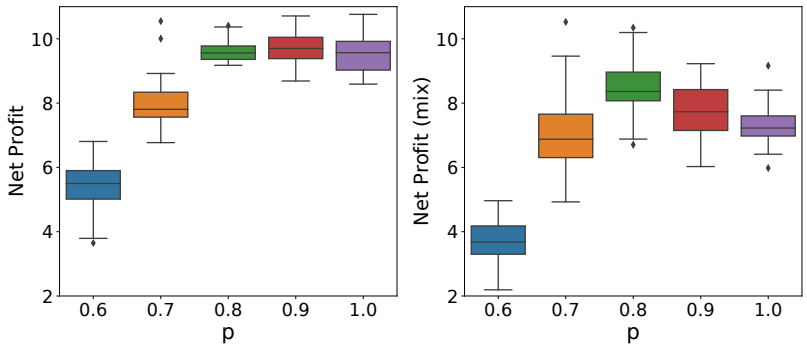


Figure 5: The effect of adjusting parameter p: The left box plot shows the $Netprofit$ that the algorithms can achieve with different parameters in the standard environment; The right box plot shows the average values of $Netprofit$ that the algorithm can achieve with different parameters in the four disturbed environments (Temperature $\in \{26, 30\}$; CO_2 concentration $\in \{500, 900\}$). Each experiment stops after 500,000 steps. The x-axis represents the parameter p of dropout, and the y-axis represents the corresponding net profits.

6. Conclusions and Future Work

In this work, we propose a robust model-based RL framework to alleviate the sample inefficiency and safety concern in greenhouse automation. To be specific, our framework utilizes sensors and actuators deployed in the greenhouse to collect observation and action samples to learn an ensemble of environment models and optimize the policy in a Dyna-style manner. Experimental results demonstrate the effectiveness and superiority of our framework in terms of robustness and efficiency, contribute to better crop growth with a higher safety guarantee.

Our future work will incorporate more prior knowledge of agriculture to improve the simulator. We will also deploy the planting policies trained by the real greenhouses algorithm to evaluate the framework through long-term real-world experiments further. Besides, we plan to use offline RL methods to improve sample utilization, reduce training costs, and use meta RL methods to transfer learning different crop species to improve the algorithm's generalization performance.

Acknowledgments

This work was done when Wanpeng Zhang, Xiaoyan Cao, Yao Yao and Zhicheng An was intern at Tencent AI Lab. This work was also supported in part by the National Natural Science Foundation of China (61972219), the Research and Development Program of Shenzhen (JCYJ20190813174403598, SGDX20190918101201696), the National Key Research and Development Program of China (2018YFB1800601), and the Overseas Research Cooperation Fund of Tsinghua Shenzhen International Graduate School (HW2021013).

References

Pieter Abbeel, Morgan Quigley, and Andrew Y Ng. Using inaccurate models in reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 1–8, 2006.

Kavosh Asadi, Dipendra Misra, Seungchan Kim, and Michel L Littman. Combating the compounding-error problem with a multi-step model. *arXiv preprint arXiv:1905.13320*, 2019.

Gouravmoy Bannerjee, Uditendu Sarkar, Swarup Das, and Indrajit Ghosh. Artificial intelligence in agriculture: A literature survey. *International Journal of Scientific Research in Computer Science Applications and Management Studies*, 7(3):1–6, 2018.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. *Advances in Neural Information Processing Systems*, 2018-Decem(NeurIPS):4754–4765, 2018. ISSN 10495258.

Ignasi Clavera, Jonas Rothfuss, John Schulman, Yasuhiro Fujita, Tamim Asfour, and Pieter Abbeel. Model-based reinforcement learning via meta-policy optimization. In *Conference on Robot Learning*, pages 617–629. PMLR, 2018.

Snehal S Dahikar and Sandeep V Rode. Agricultural crop yield prediction using artificial neural network approach. *International journal of innovative research in electrical, electronics, instrumentation and control engineering*, 2(1):683–686, 2014.

Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. *A survey on policy search for robotics*. now publishers, 2013.

Gerard George, Simon JD Schillebeeckx, and Teng Lit Liak. The management of natural resources: An overview and research agenda. *Managing Natural Resources*, 2018.

Reza Ghaffari, Fu Zhang, Daciana Iliescu, Evor Hines, Mark Leeson, Richard Napier, and John Clarkson. Early detection of diseases in tomato crops: An electronic nose and intelligent systems approach. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2010.

Hani Hagras, Martin Colley, Victor Callaghan, and Malcolm Carr-West. Online learning and adaptation of autonomous mobile robots for sustainable agriculture. *Autonomous Robots*, 13(1):37–52, 2002.

Silke Hemming, Feije de Zwart, Anne Elings, Anna Petropoulou, and Isabella Righini. Cherry tomato production in intelligent greenhouses—sensors and ai for control of climate, irrigation, crop yield, and quality. *Sensors*, 20(22):6430, 2020.

Ming-Hui Huang and Roland T Rust. Artificial intelligence in service. *Journal of Service Research*, 21(2):155–172, 2018.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, pages 12519–12530, 2019.

Kirtan Jha, Aalap Doshi, Poojan Patel, and Manan Shah. A comprehensive review on automation in agriculture using artificial intelligence. *Artificial Intelligence in Agriculture*, 2:1–12, 2019.

Jeongeun Kim, Jeahwi Seol, Sukwoo Lee, Se-Woon Hong, and Hyoung Il Son. An intelligent spraying system with deep learning-based semantic segmentation of fruit trees in orchards. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3923–3929. IEEE, 2020.

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*, 2018.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Wenhao Li, Xiangfeng Wang, Bo Jin, Dijun Luo, and Hongyuan Zha. Structured cooperative reinforcement learning with time-varying composite action space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. *arXiv preprint arXiv:1807.03858*, 2018.

Aishwarya Himanshu Manek and Parikshit Kishor Singh. Comparative study of neural network architectures for rainfall prediction. In *2016 IEEE Technological Innovations in ICT for Agriculture and Rural Development (TIAR)*, pages 171–174. IEEE, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

G Parameswaran and K Sivaprasath. Arduino based smart drip irrigation system using internet of things. *International Journal of Engineering Science and Computing*, 6(5): 5518–5521, 2016.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. *arXiv preprint arXiv:1703.02702*, 2017.

Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

Cindy Rijswick. World vegetable map 2018: More than just a local affair. Website, 2018. https://research.rabobank.com/far/en/sectors/regional-food-agri/world_vegetable_map_2018.html.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Brian Sparks. What is the current state of labor in the greenhouse industry?, 2018. https://www.greenhousegrower.com/management/what-is-the-current-state-of-labor-in-the-greenhouse-industry.

Leonid Kuvayev Rich Sutton. Model-based reinforcement learning with an approximate, learned model. In *Proceedings of the ninth Yale workshop on adaptive and learning systems*, pages 101–105, 1996.

Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2(4):160–163, 1991. ISSN 0163-5719. doi: 10.1145/122344.122377.

Aviv Tamar, Yonatan Glassner, and Shie Mannor. Optimizing the CVaR via sampling. *Proceedings of the National Conference on Artificial Intelligence*, 4:2993–2999, 2015.

Chen Tessler, Yonathan Efroni, and Shie Mannor. Action robust reinforcement learning and applications in continuous control. *arXiv preprint arXiv:1901.09184*, 2019.

Lu Wang, Xiaofeng He, and Dijun Luo. Deep reinforcement learning for greenhouse climate control. In *2020 IEEE International Conference on Knowledge Graph (ICKG)*, pages 474–480. IEEE, 2020.

Yao Yao, Li Xiao, Zhicheng An, Wanpeng Zhang, and Dijun Luo. Sample efficient reinforcement learning via model-ensemble exploration and exploitation. *arXiv preprint arXiv:2107.01825*, 2021.

Alexander You, Fouad Sukkar, Robert Fitch, Manoj Karkee, and Joseph R Davidson. An efficient planning and control framework for pruning fruit trees. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3930–3936. IEEE, 2020.

Jun Zhou, Qin Chen, Quan Liang, et al. Vision navigation of agricultural mobile robot based on reinforcement learning. *Nongye Jixie Xuebao= Transactions of the Chinese Society for Agricultural Machinery*, 45(2):53–58, 2014.

Kemin Zhou and John Comstock Doyle. *Essentials of robust control*, volume 104. Prentice hall Upper Saddle River, NJ, 1998.