DRLIC: Deep Reinforcement Learning for Irrigation Control

Xianzhong Ding
University of California, Merced
xding5@ucmerced.edu

Wan Du
University of California, Merced
wdu3@ucmerced.edu

ABSTRACT

Agricultural irrigation is a major consumer of freshwater. Current irrigation systems used in the field are not efficient, since they are mainly based on soil moisture sensors' measurement and growers' experience, but not future soil moisture loss. It is hard to predict soil moisture loss, as it depends on a variety of factors, such as soil texture, weather and plants' characteristics. To improve irrigation efficiency, this paper presents *DRLIC*, a deep reinforcement learning (DRL)-based irrigation system. *DRLIC* uses a neural network (DRL control agent) to learn an optimal control policy that takes both current soil moisture measurement and future soil moisture loss into account. We define an irrigation reward function that facilitates the control agent to learn from past experience. Sometimes, our DRL control agent may output an unsafe action (e.g., irrigating too much water or too little). To prevent any possible damage to plants' health, we adopt a safe mechanism that leverages a soil moisture predictor to estimate each action's performance. If it is unsafe, we will perform a relatively-conservative action instead. Finally, we develop a real-world irrigation system that is composed of sprinklers, sensing and control nodes, and a wireless network. We deploy *DRLIC* in our testbed composed of six almond trees. Through a 15-day in-field experiment, we find that *DRLIC* can save up to 9.52% of water over a widely-used irrigation scheme.

1 INTRODUCTION

Agriculture is a major consumer of ground and surface water in the United States, accounting for approximately 80% of the Nation's consumptive water use and over 90% in many Western states[1]. California's 2019 almond acreage is estimated at 1,530,000 acres, and almond irrigation is estimated to consume roughly 195.26 billion gallons per year [4, 5]. With a historic drought afflicting the Western states, it is imperative to improve the irrigation efficiency for saving our limited freshwater reserve. This work is focused on the irrigation efficiency of almond orchards.

The primary goal of agricultural irrigation is to guarantee the trees' health and maximize production. To do so, the trees' soil moisture should be maintained with a range between the Field Capacity (FC) level and the Management Allowable Depletion (MAD) level. If the soil moisture is lower than the MAD level, the almond trees will turn brown or even die. If the soil moisture is higher than the FC level, excess water in the soil will reduce the movement of oxygen, impacting the ability of the tree to take in water and nutrients. Both FC and MAD levels can be determined by the type of plants and soil. For a specific orchard, we need to know the soil type. We can then find the FC and MAD levels for a specific soil type by referring to a manual [6].

[1]Irrigation Water Use: https://www.ers.usda.gov/

To maintain the soil moisture between the MAD and FC range, the sprinklers need to be opened every day or several days, depending on the soil moisture change. Due to the high evaporation loss in California, daily irrigation is recommended by the Almond Board of California [6] and used in some existing irrigation systems [7, 8]. Current micro-sprinkler irrigation systems normally irrigate plants at night, since irrigating during the day causes higher evaporative water loss (14-19%) [9]. Therefore, the irrigation scheduling problem is to decide the irrigation water volume for each sprinkler to guarantee that the soil moisture will be still within the MAD and FC range at next irrigation time. The decision is based on the current soil moisture level and the predicted soil moisture loss of next day. The latter is determined by soil type, local weather, and plants' properties (e.g., the root's length and the number of leaves). The irrigation's goal is to irrigate the trees with a proper amount of water, so that the soil moisture will be still above the MAD level at the next irrigation time.

Optimal irrigation control strategies should model the soil moisture loss that will be experienced before the next irrigation time. If we have such a soil moisture prediction model, conventional Model Predictive Control (MPC) methods can be used to decide the optimal amount of water to irrigate. However, the performance of these methods relies highly on the accuracy of the soil moisture prediction model [10, 11]. It is hard to obtain an accurate model for an almond orchard, because the soil moisture is affected by many factors, including soil type, topography and surrounding environment (e.g., ambient temperature, humidity, and solar radiation intensity), and internal transpiration from plants [12]. In addition, customized soil moisture models are required for different orchards, limiting the scalability of MPC-based methods. Due to the above two limitations, MPC-based methods have not been used in orchards.

The irrigation systems currently used in orchards are ET-based or sensor-based control methods. Evapotranspiration (ET) is an estimate of moisture lost from soil, subject to weather factors such as wind, temperature, humidity, and solar irradiance. All these weather factors are being measured by weather stations. Local ET value is also publicly available [13] and updated every hour. Based on the ET values since the last irrigation time, ET-based irrigation controllers start the sprinklers to compensate for the soil moisture loss. However, they do not consider the soil moisture loss of next day before the next irrigation time. If the soil moisture loss in the last day does not equal the soil moisture loss that will happen in the next day, ET-based irrigation may under-irrigate or over-irrigate. In addition, a safe margin of water [14] is normally added, making ET-based methods over-irrigate in most cases [7].

With accurate soil moisture sensors, irrigation controllers can react directly to the soil moisture level [7]. The commonly-used controllers are "rule-based", in which a certain amount of water will be supplied once soil moisture deficiency is detected. However, parameters for the time and the amount to irrigate are generally

978-1-6654-9624-7/22/$31.00 ©2022 IEEE
DOI 10.1109/IPSN54338.2022.00011

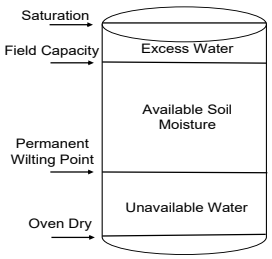


Figure 1: The various levels of the soil water content [1].

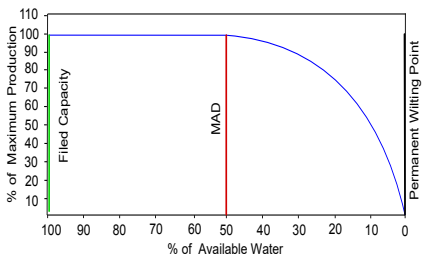


Figure 2: How plant production (growth) is affected by soil water content [2].

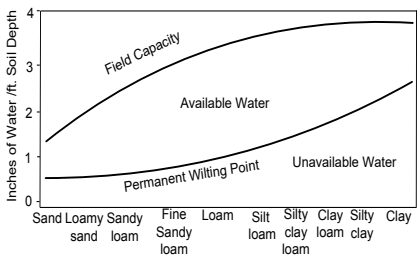


Figure 3: Relationship between available water capacity and soil texture [3].

tuned by growers by their experience. Without predicting how much water will be lost, sensor-based irrigation normally does not systematically take into account future weather information, such as rain and wind in next day.

To solve the limitations of the above existing irrigation schemes, we develop *DRLIC*, a practical Deep Reinforcement Learning (DRL)-based irrigation system, which automatically learns an optimal irrigation control policy by exploring different control actions. In *DRLIC*, a control agent observes the *state* of the environment, and chooses an *action* based on a control policy. After applying the action, the environment transits to a next state and the agent receives a *reward* to its *action*. The goal of learning is to maximize the expected cumulative discounted reward. *DRLIC*'s control agent uses a neural network to learn its control policy. The neural network maps "raw" observations to the irrigation decision for the next day. The state includes the weather information (e.g., ET and Precipitation) of today and next day.

To minimize the irrigation water consumption while not impacting the trees' health, we design a reward function that considers three specific situations. If the soil moisture result is higher than the FC level or lower than the MAD level, we will give the control agent a negative reward. If the soil moisture result is within the MAD and FC range, we will give the control agent a positive reward inversely proportional to the water consumption.

Ideally, *DRLIC*'s control agent should be trained in a real orchard of almond trees. However, due to the long irrigation interval (one day in our case), the control agent can only explore 365 control actions per year. It will take 384 years to train a converged control agent. Therefore, to speed up the training process, we train our control agent in a customized soil-water simulator. The simulator is calibrated by the 2-month soil moisture data of six almond trees and can generate sufficient training data for *DRLIC* using 10-year weather data.

Working as an irrigation controller in the field, the control agent may meet some states that it has not seen during training, especially for the control agent trained in a simulated environment. In this situation, the control agent may make a poor decision that violates plants' health, i.e., making the soil moisture level lower than the MAD level or higher than the FC level. To handle the gap between the simulated environment and the real orchard, we design a safe irrigation mechanism. If *DRLIC*'s control agent outputs an unwise action, instead of executing that action, we use the ET-based method to generate another action. We use the soil moisture model of our soil-water simulator to verify whether an action is safe or not.

To evaluate the performance of *DRLIC*, we build an irrigation testbed with micro-sprinklers currently used in almond orchards. Six almond trees are planted in two raise-beds. Each tree has a sensing and control node, composed of an independently-controllable micro-sprinkler and a soil moisture measurement set (two sensors deployed at different depths in the soil). Each node can send its sensing data to our server via IEEE 802.15.4 wireless transmission, and receive irrigation commands from the server.

We have deployed our testbed in the field and collected soil moisture data from six sensing and control nodes for more than three months. We use 2-month data to train our soil moisture simulator and 0.5-month data to validate its accuracy. After training *DRLIC*'s control agent, we have deployed the controller in our testbed for 15 days. Experiment results demonstrate that *DRLIC* can reduce the water usage by 9.52% over the ET-based control method, without damaging the almond tree health.

We summarize the main contributions of this paper as follows:

- We design *DRLIC*, a DRL-based irrigation method for agricultural water usage saving.
- A set of techniques have been proposed to transform *DRLIC* into a practical irrigation system, including our customized design of DRL states and reward for optimal irrigation, a validated soil moisture simulator for fast DRL training, and a safe irrigation module.
- We build an irrigation testbed with customized sensing and actuation nodes, and six almond trees.
- Extensive experiments in our testbed show the effectiveness of *DRLIC*.

2 IRRIGATION PROBLEM

Soil Water Content Parameters. Soil is a plant's water reservoir. Water can fill up to 35% of the space in soil. Soil water content is the amount of water in the soil, which is often measured as a percentage of water by volume (%) or by inches of water per foot of root (in/ft). Soil moisture sensors are used to measure the soil water content (%) at one location in the soil. For a tree with a root of several feet, multiple soil moisture sensors may be deployed in different depths along with the root. The root is divided into a certain number of pieces. A soil moisture sensor is deployed at the middle point of each piece. The soil water content of the tree can be calculated as $V = \sum_{j=1}^{M} \varphi_j * d_j$, where M is the number of moisture

sensors installed at different depths (M is 2 in our experiments); φ_j is the reading measured by the jth soil moisture sensor; and d_j is the depth that the jth moisture sensor covers. If such a set of soil moisture sensors are used to measure the soil water content of a region, they will be deployed under a typical tree that has similar soil water content with most of the trees in the region.

A healthy plant's root must be within a sufficient supply of water. Figure 1 shows two critical levels of soil water content for plants' health [1]. 1) If the soil water content is below the Permanent Wilting Point (PWP), plants cannot suck necessary moisture out of the soil. Keeping soil below the PWP level for an extended period of time will cause plants to wilt or eventually die. 2) If the soil water content of a tree is above Field Capacity (FC), the soil has an over-abundance of water, which will cause water waste and rotting of the root over time (impacting the trees' health). Therefore, *the goal of irrigation systems is to maintain soil water content between the PWP level and the FC level.*

For fruit trees like almond, production is the major goal of irrigation. To maximize the production, we need to maintain the soil water content above the Management Allowable Depletion (MAD) level, instead of the PWP level. Figure 2 depicts the relationship between soil water content and plant production for almond trees [2]. The curve and the MAD level may be different for different fruits. From Figure 2, we can see that the MAD level for almond trees is the median value (50%) between the FC level and the PWP level. Therefore, *almond trees can achieve their maximum production, as long as we maintain the soil water content above the MAD level.*

How to Determine these Parameters in an Orchard? The soil water content range between the FC level and the PWP level is the Available Water holding Capacity (AWC) of the soil. As shown in Figure 3, different soil types have different AWCs [3]. The soil's AWC may be affected by its texture, presence and abundance of rock fragments, and its depth and layers. The soil's AWC increases as it becomes finer-textured from sands to loam [3], and the soil's AWC decreases as it contains more clay from loam to clay [3].

The AWC of a tree, V_{awc}, can be calculated as $V_{awc} = \sigma_{awc} * D_{foot}$, where σ_{awc} is the soil's AWC and D_{foot} is the tree's root depth in the unit of feet. The AWC for different soil types, σ_{awc}, can be found in [3].

The PWP level for a soil type, V_{pwp}, can also be calculated as $V_{pwp} = \varphi_{pwp} * D_{inch}$, where φ_{pwp} is the soil moisture content at the wilting point of that soil type and D_{inch} are the root depth of the plant in the unit of inches. φ_{pwp} for a specific soil type can be found in [3].

Based on the above two parameter (V_{awc} and V_{pwp}), we can also obtain the FC level as $V_{fc} = V_{awc} + V_{pwp}$, and the MAD level as $V_{mad} = \alpha * V_{awc} + V_{pwp}$, where α is set to 50% for almond trees.

How to Use these Parameters for Irrigation? The goal of irrigation is to maintain the soil water content of plants between the FC level and the MAD level. To correctly set an irrigation system, we need to know the soil' AWC in the orchard and the PWP level (V_{awc} and V_{pwp}). We can determine these two parameters based on the above method, as long as we know the soil type. If the orchard is large, the soil type varies in space and these two parameters change too. We need to adapt the setting of these two parameters in the irrigation system accordingly.

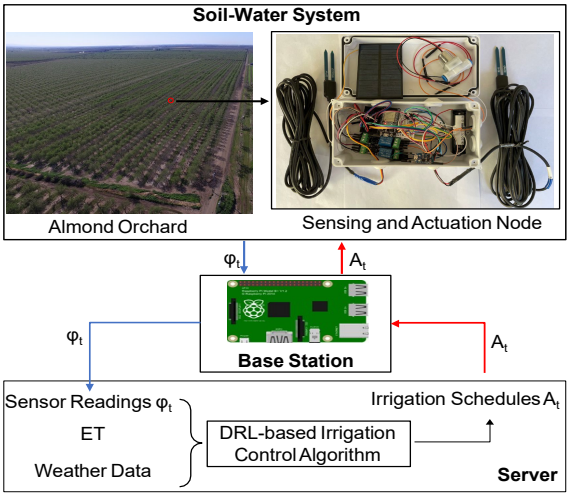


Figure 4: *DRLIC* System Architecture.

How Many Valves to Control in an Orchard? Ideally, the sprinkler for each tree should be individually controlled, since the ET of each tree in an orchard varies from 0.12 to 0.20 inches [15]. Moreover, the soil type also varies spatially in an orchard [6], e.g., there are 10 soil type differences with soil clay loam accounting for from 45.6% to 54.7% and 0 to 8 percent slopes in a 60-acre orchard of California [2]. However, there are around 75-125 almond trees in one acre, it is costly to deploy a soil moisture sensor under each tree. Thus, an orchard is normally divided into several irrigation regions based on the similarity of soil texture. A valve is used in each irrigation region to control all the sprinklers. The irrigation problem of a large orchard is to control a number of valves. This paper is focused on irrigation scheduling, but not field partitioning. A simple way to partition an orchard into several irrigation regions is to survey the soil samples across the orchard using an auger. Growers normally conduct the survey for other purposes too, such as planning the density of trees and fertilizing the trees.

3 *DRLIC* SYSTEM DESIGN

In this section, we first give an overview of *DRLIC*. We model the irrigation problem as a Markov decision process. We design a DRL-based irrigation scheme and a safe irrigation module.

3.1 Overview

Figure 4 shows the system architecture of *DRLIC*, which is composed of two key components, i.e., a wireless network of sensing and actuation sprinkler nodes, and a DRL-based control algorithm.

For an almond orchard, we install the sensing and actuation node for each irrigation region. One sensing and actuation node is equipped with a set of soil moisture sensors that are deployed at different depths in the soil. Sensing data is transmitted to the base station via an IEEE 802.15.4 network. The *Base Station* collects the data from *DRLIC* nodes and sends them to a local server using Wi-Fi. These sensing data collected from all *DRLIC* nodes creates a "snapshot" of the soil moisture readings φ_t across the entire orchard.

[2]Soil Map: https://casoilresource.lawr.ucdavis.edu/gmap/

On the server, the DRL-based irrigation control agent makes irrigation decisions based on the soil moisture sensors' readings, ET and weather data from local weather stations. It provides the optimal irrigation schedule for all *DRLIC* nodes. The objective of *DRLIC* is to minimize the total irrigation water consumption while meeting the requirement of almond health. The server will send the generated irrigation schedules A_t to all *DRLIC* nodes. By receiving a command, a node may open its sprinkler by a latching solenoid with two relays. The implementation details of the nodes will be introduced in Section 4.

3.2 MDP and DRL for Irrigation

We adopt the daily irrigation scheme, i.e., the irrigation starts at 11 PM every day. Each time, the controller decides how long to open each sprinkler to guarantee that the soil water content will be still within the MAD and FC range tomorrow night. The future soil water content is determined by the current soil water content, the irrigated water volume, the trees' water absorption, and soil water loss (caused by runoff, percolation and ET). Such a sequential decision-making problem can be formulated as a Markov Decision Process (MDP), modeled as <*S, A, T, R*>, where

- S is a finite set of states, which includes sensed moisture level from orchard and weather data from local station.
- A is a finite set of irrigation actions for all control valves.
- T is the state transition function defined as $T : S \times A \rightarrow S$. The soil water content at next time step is determined by current soil water content and the irrigation action.
- R is the reward function defined as $S \times A \rightarrow \mathrm{R}$, which qualifies the performance of a control action.

Based on the above MDP-based irrigation problem formulation, we will find an optimal control policy $\pi(s)^* : S \rightarrow A$, which maximizes the accumulative reward R. We cannot apply conventional tools (e.g., dynamic programming) to search for the optimal control policy, because the state transition function is hard to analytically characterize. In this paper, we consider an RL-based approach to generating irrigation control algorithms. Unlike previous approaches that use pre-defined rules in heuristic algorithms, our approach will learn an irrigation policy from observations.

DRL is a data-driven learning method. It has been widely applied in many control applications [16–20]. DRL learns an optimal control policy through interacting with the environment. At each time step t, the control agent selects an action $A_t = a$, given the current state $S_t = s$, based on its policy π_θ.

$$a \sim \pi_\theta(a|s) = \mathbb{P}(A_t|S_t = s; \theta) \tag{1}$$

In DRL, the control policy is approximated by a neural network parameterized by θ [21]. When the control agent takes the action a, a state transition $S_{t+1} = s'$ occurs based the system dynamics f_θ (Equation 2), and the control agent receives a reward $R_{t+1} = r$.

$$s' \sim f_\theta(s, a) = \mathbb{P}(S_{t+1}|S_t = s, A_t = a) \tag{2}$$

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_{\pi_\theta}[r] \tag{3}$$

Due to the Markov property, both reward and state transition depend only on the previous state. DRL then finds a policy π_θ that maximizes the expected reward (Equation 3).

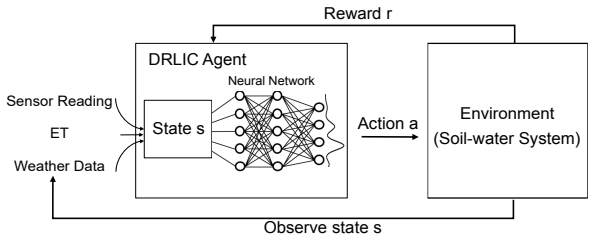


Figure 5: Deep Reinforcement Learning in *DRLIC*.

Why do we use DRL for irrigation control?

- DRL learns an optimal irrigation control policy directly from data, without using any pre-programmed control rules or explicit assumptions about the soil-water environment.
- DRL allows us to use domain knowledge to train an irrigation control agent (a neural network) without labeled data.
- The generalization ability of the neural network enables the control agent to better handle the dynamically-varying weather and ET data.

3.3 Deep Reinforcement Learning in *DRLIC*

Figure 5 summarizes the DRL architecture of *DRLIC*. The irrigation control policy (*DRLIC* Agent) is derived from training a neural network. The agent takes a set of information as input, including current soil water content, today's weather data (e.g., ET and precipitation), and the predicted weather data for tomorrow. Based on the input, the agent outputs the best action, i.e., the amount of water to irrigate. Until the next day at 11 PM, the resulting soil water content is observed and passed back to the agent to calculate a reward. The agent uses the reward to update the parameters of the neural network for better irrigation control performance. Next, we introduce the design of each DRLIC component.

3.3.1 State in DRLIC. The state in our irrigation MDP model contains the information of three parts. (a) Sensed state, which is the soil water content measured by *DRLIC* nodes. (b) Weather-related state, which includes the current and predicted state variables from weather station. (c) Time-related state, which is about date information.

Sensed State. The soil water content of each irrigation region, calculated by Equation 6 using sensor reading φ from *DRLIC* node.

Weather-related State. It is a vector containing the weather information of current day and next day: ET (inch), Precipitation (inch), maximum, average, minimum Temperature (°F), maximum, average, minimum Humidity (%), average Solar Radiation (Ly/day), average Wind Speed (mph), Predicted ET by Equation 16 (inch), and forecasted Precipitation (inch) from local weather station.

Time-related State. Date including the month. The soil moisture may vary in different months.

3.3.2 Action in DRLIC. Based on the current state outlined above, our irrigation scheduling is to find the best amount of water to irrigate (inch), which can maintain plant health (or maximize production) with minimum water consumption. The action is a vector that contains the water amount to irrigate for each irrigation region in an orchard. When the agent outputs an action, we will convert the amount of irrigation water to the open time duration (td) td_i

for ith micro-sprinkler. It is calculated as $td_i = a_i/I$, where I is the irrigation rate. We set I to 0.018 inch/min according to the specifications of the micro-sprinklers used in our testbed.

3.3.3 Reward in DRLIC. We define the reward function to express our objective of achieving good plant health with minimum water consumption. Both plant health and water consumption should be incorporated in the reward function. As we know from Section 2, to achieve the maximum production of almond trees, we need to maintain the soil water content between the MAD level and FC level. We use the soil water content deviation from these two levels as a proxy for plant health.

To minimize water consumption while not affecting the plant health, we consider three situations in the design of the reward, as shown in Equation 5. First, when the soil water content (V_i) for ith irrigation region is higher than the FC (V_{fc}) level, the irrigated water is more than the plants' need. In this case, the plants' health is affected by over-irrigated water, and water consumption is too high. Second, when V_i is between V_{fc} and V_{mad}, the plants are in good health. In this case, we strive to maintain the V_i close to V_{mad} to save water, so we give a reward inversely proportional to the water consumption. Third, when V_i is lower than V_{mad}, the plants are under water stress. The plants' health is significantly impacted, proportional to the distance between V_i and V_{mad}.

By considering the above three situations, our reward function is defined as follows:

$$R = -\sum_{i=1}^{N} R_i \tag{4}$$

$$R_i = \begin{cases} \lambda_1 * (V_i - V_{fc}) + \mu_1 * a_i, & V_i > V_{fc} \\ \mu_2 * a_i, & V_{fc} > V_i > V_{mad} \\ \lambda_3 * (V_{mad} - V_i) + \mu_3 * a_i, & V_i < V_{mad} \end{cases} \tag{5}$$

$$V = \textstyle\sum_{j=1}^{M} \varphi_j * d_j \tag{6}$$

$$V_{mad} = \alpha * V_{awc} + V_{pwp} \tag{7}$$

$$V_{fc} = V_{awc} + V_{pwp} \tag{8}$$

$$V_{pwp} = \varphi_{pwp} * D_{inch} \tag{9}$$

$$V_{awc} = \sigma_{awc} * D_{foot} \tag{10}$$

where N is the number of irrigation regions in one orchard. a is the amount of water from the RL agent. σ_{awc} and φ_{pwp} are set by referring to the manual of California Almond Board [6] based on our specific soil type in our testbed. Equations 6, 7, 8, 9 and 10 have been introduced in Section 2.

In our current implementation, the parameters of our reward function are set to the values shown in Table 1, based on the specifications of our testbed. The parameters in Equation 5 (i.e., λ_1, μ_1, μ_2, λ_3 and μ_3) are set to the best values that provide the best rewards during training. Their values are set by grid search, which will be introduced in detail in Section 5. The values of these parameters in Table 1 confirm with our design goal of the reward function. First, when V_i is larger than V_{fc}, we give penalties due to both plants' health and water consumption ($\lambda_1 = 3$, but $\mu_1 = 8$).

Table 1: Parameter Setting in Reward.

Parameter	Value	Parameter	Value
λ_1	3	α	50 (%)
μ_1	8	D_{inch}, D_{foot}	23.62 inches, 1.97 (feet)
μ_2	3	d	11.81 (inches)
λ_3	10	φ_{pwp}	10 (%)
μ_3	1	σ_{awc}	2.4 (in./ft.)

Second, when V_i is lower than V_{mad}, we give a higher penalty due to plants' health ($\lambda_3 = 10$, but $\mu_3 = 1$).

3.4 DRLIC Training

3.4.1 Policy Gradient Optimization. In the above DRL framework, a variety of policy gradient algorithms can be used to train the irrigation control agent. Policy gradient algorithms achieve the objective in Equation 3 by computing an estimate of the policy gradient and optimizing the objective through stochastic gradient ascent (Equation 11).

$$\theta \leftarrow \theta + \alpha \bigtriangledown_\theta \mathbb{E}_{\pi_\theta}[r] \tag{11}$$

In this work, we use proximal policy optimization (PPO) [22], which has been successfully applied in many applications such as navigation [23] and games [24]. PPO is known to be stable and robust to hyperparameters and network architectures [22].

PPO minimizes the loss function in Equation 12, which is equivalent to maximizing the Monte Carlo estimate of rewards with regularization. The advantage function $\hat{A}_t$ given by Equation 13 is used to estimate the relative benefit of taking an action from a given state.

$$L_{PPO}(\theta) = -\hat{\mathbb{E}}_t[min(w_t(\theta)\hat{A}_t, clip(w_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{12}$$

$$\hat{A}_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i} \tag{13}$$

$$w_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{14}$$

In Equation 14, $\pi_\theta(a_t|s_t)$ is the policy being updated with the loss function and $\pi_{\theta_{old}}(a_t|s_t)$ is the policy that was used to collect data with environment interaction. As the data collection policy differs from the policy being updated, it introduces a distribution shift. The ratio $w_t(\theta)$ corrects for this drift using importance sampling. The ratio of two probabilities can blow up to large numbers and destabilize training, so the ratio is clipped with ϵ.

3.4.2 Data Collection and Preprocessing. On day t, the *DRLIC* agent observes a state s (e.g., moisture level), and then chooses an action (water amount). After applying the action, the soil-water environment's state transits to s_{t+1} next day and the agent receives a reward r. After that, a data pair (s_t, a_t, r_t, s_{t+1}) can be collected. We conduct data normalization by subtracting the mean of the states/action and dividing by the standard deviation. We use 10-year weather data (2010-2020) to generate the data pairs in our dataset, which will be used to train our *DRLIC* agent.

3.4.3 Training Process. Ideally, *DRLIC*'s control agent should be trained in an orchard of almond trees. A well-trained DRL agent needs 384 years to converge due to the long control interval of irrigation systems. It is impossible to train *DRLIC* agent in an orchard. A feasible solution is to refer to a high-fidelity simulator.

Algorithm 1: *DRLIC* Training Algorithm

Input: State s, Action a, Reward r, an initialized policy, π_θ;
Output: A trained irrigation control agent ;

for *i=0,..., # Episodes* **do**
  State ← Soil-water environment
  $\theta_{old} \leftarrow \theta$;
  for *t = 0, ..., Steps* **do**
    $\hat{a}_t = \pi_\theta(s_t)$;
    $s_{t+1}, r_{t+1} = env.step(\hat{a}_t)$;
  Compute $\hat{A}_t$;
  With minibatch of size M;
  $\theta \leftarrow \theta - \alpha \nabla_\theta L_{PPO}(\theta)$;

However, there are no such simulators available in the soil-water domain. Then we decide to leverage a data-driven simulator to speed up the training process. We employ the soil water content predictor introduced in Section 3.5 as our soil-water simulator. The simulator allows *DRLIC* to "experience" the weather of 10 years in several minutes.

The training procedure of *DRLIC* is outlined in Algorithm 1. We train *DRLIC* using 1000 episodes and length of an episode as 30 days. For each episode, we can collect 30 training data pair (s_t, a_t, r_t, s_{t+1}) under different weather data and leverage Equation 12 to optimize the objective in Equation 3 through stochastic gradient ascent. The training ends once Algorithm 1 converges: at the end of each episode the total reward obtained is compared with the previous total. If the current episode reward does not change by ± 3%, we consider the policy has converged. If the policy does not converge, the training will continue up to a maximum of 100 training iterations (# episodes = 100). After the training, we will deploy the trained *DRLIC* agent into the real almond orchard.

When we are given a new environment (e.g., a new orchard), we first need to collect the real-world irrigation data of new environment by existing controller (e.g., ET-based control) to build a soil water content predictor to describe the water balance in the root zone soil. Then we leverage the soil water content predictor to speed up the training process, after that, we deploy the well-trained *DRLIC* agent for this new orchard.

3.5 Safe Mechanism for Irrigation

We design a safe mechanism that integrates the RL and ET controller in a coupled close-loop. Figure 6 illustrates the workflow of safe mechanism, with the following key elements. (i) Different from the pure RL framework, we introduce a safety moisture condition detector to evaluate whether the RL algorithm outputs a safe action. (ii) If so, the action goes to the RL agent, who will be in charge of irrigation control. (iii) Otherwise, we will use an ET-based controller to generate an action for that control cycle. (iv) *DRLIC* will the RL agent for the future control cycles. We now introduce the soil water content predictor and safety condition detector.

Soil Water Content Predictor. To enable early detection of an unsafe action, we design a soil water content predictor to predict the moisture trend after taking an action. Then we design a safe condition detector to detect almond health penalty $p(t)$. The idea is to detect whether the damage metric for an almond tree is higher

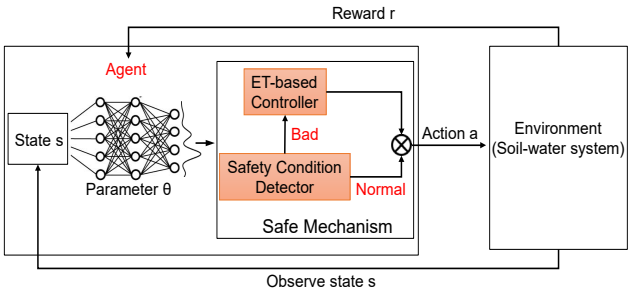


Figure 6: Reinforcement Learning with Safe Mechanism.

Table 2: Coefficients of Predictor for Each Tree.

	c1	c2	c3	b	R^2	NRMSE
Tree1	0.973	0.288	-0.103	0.003	0.982	0.062
Tree2	0.937	0.325	-0.121	0.013	0.985	0.071

than a threshold. If so, the detector will command *DRLIC* to switch from RL to ET-based controller.

We design a soil water content predictor to describe the water balance in the root zone soil. The variations of water storage in the soil are caused by both inflows (irrigation and precipitation) and outflows (evapotranspiration). This leads to the following mathematical expression:

$$V_{i,t+1} = c_1 * V_{i,t} + c_2 * (A_{i,t} + P_t) + c_3 * E_t + b \quad (15)$$

$$E_t = \Gamma_c * RA * TD^{(1/2)} * (T_t + 17.8°C) \quad (16)$$

where $V_{i,t+1}$ denotes the predicted moisture level in the root zone for ith irrigation region after taking the action from RL, E_t and P_t are the plants' ET and the measured rainfall. In time period t, and $A_{i,t}$ is the irrigation amount for ith irrigation region. c_1, c_2, and c_3 are coefficients. It is assumed in this work that runoff and water percolation are proportional to soil moisture level [25–27] in Equation 15. All the coefficients can be determined by means of system identification techniques [28]. All variables are normally expressed in inches.

The weather data can be get from local weather station. For ET, we adopt the simple calculation model established in [29]. As shown in Equation 16, where Γ_c is a crop-specific parameter. RA stands for extraterrestrial radiation, which is in the same unit as E_t. TD denotes the annual average daily temperature difference, which can be derived from local meteorological data, and T_t is the average outdoor temperature during the tth time period.

Safety Condition Detector. We employ the difference between predicted moisture level and lower bound as a detector to estimate the almond tree damage. As explained in Section 2, MAD is the lower bound. Then we use $\sum_{i=1}^{N}(V_{mad} - V_{i,t+1})$ as a safety condition detector, $V_{i,t+1}$ denotes the predicted moisture level from ith irrigation region for t timestep. V_{mad} is the water content lower bound. *DRLIC* will evoke ET-based controller once safety condition detector detects the dangerous irrigation action.

Parameter Learning of our Soil Water Content Predictor. We leverage the designed testbed to collect the irrigation amount of almond trees for 2 months. The ET value for each day is collected from a local weather station [13] and the moisture level for each

Figure 7: Testbed and Microsprinkler Irrigation System.

tree is collected by the designed *DRLIC* node. Then the linear least square method was applied to estimate the coefficients. R^2 is used to explain the strength of the relationship between the moisture level and related factors. Normalized root-mean-square error (NRMSE) is used as a goodness-of-fit measure for predictors. The results are shown in Table 2, we can see that the R^2 is close to 1 indicating that the irrigation, ET and precipitation have a strong relationship with soil water content for the tree. The NRMSE is less than 0.1 which means that the predictor can achieve accurate prediction for soil water content.

4 TESTBED AND HARDWARE

4.1 Testbed and Microsprinkler Description

Figure 7 shows our micro-sprinkler irrigation testbed. The micro-sprinkler irrigation system is installed and designed to be identical in hardware, micro-sprinkler coverage, etc. This irrigation system measures 290 cm x 160 cm, with micro-sprinklers arranged in a 3x2 grid, each 97cm from the next. The micro-sprinklers chosen were 1/4 ', 360 ° pattern by Rainbird, which are currently considered state-of-the-art in micro-sprinkler technology. Six all-in-one young almond trees were planted into the testbed (three for each). The average height is 2 meters. The soil with 2.7 m ^{3} volume is collected from a local orchard that is a typical loam soil and the plant-available water-holding capacity is 2.4 inches of water per foot.

4.2 *DRLIC* Node Development.

The designed DRLIC node in Figure 4 consists of four main parts: sensors, actuator, power supply and transmission module.

Sensors: It consists of several moisture sensors for different depths. The moisture sensors vary in their sensitivity and their volume of soil measured. Each moisture sensor for 12-inch depth provides accurate quantitative soil moisture assessment following the Almond Board Irrigation Improvement Continuum [6]. We

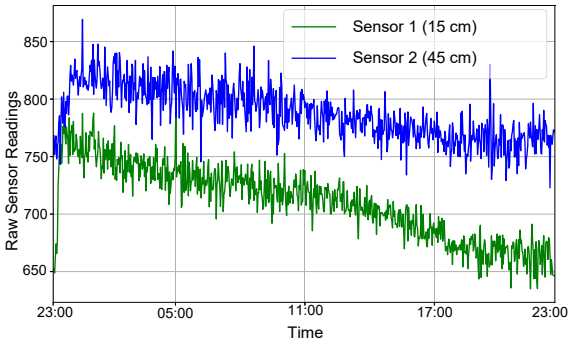
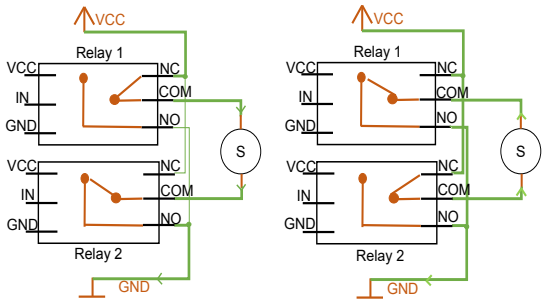


Figure 8: Daily Soil Moisture Readings.



(a) Positive Current Pulse. (b) Negative Current Pulse.

Figure 9: On and off Circuit Diagram for Latching Solenoid.

assign 2 moisture sensors for each *DRLIC* node since the depth of root zone of the almonds in our testbed is 24 inches.

A key feature of the *DRLIC* node is the ability to measure the volumetric water content in the surrounding soil. We opted to purchase research-quality Decagon EC-5 sensors [3], with a reported accuracy of ±3%. Raw sensor readings collected over a period of one day with a high sampling frequency can be seen in Figure 8. The sensors report the dielectric constant of the soil, which is an electrical property highly dependent on the volumetric water content (VWC).

$$\varphi(m^3/m^3) = 9.92 * 10^{-4} * raw\_reading - 0.45 \quad (17)$$

A linear calibration Function 17 above provided by the sensor manufacturer is used to convert the raw readings to VWC. The range of φ is between 0% and 100%. φ of saturated soils is generally 40% to 60% depending on the soil type.

Actuator: It consists of a latching solenoid with two relays. A standard solenoid requires constant power to allow water to flow, making it a poor choice for a battery-powered system. The nine-volt performance all-purpose alkaline batteries from Amazon can only continue to power the standard 12V DC solenoid for 8 hours. To extend *DRLIC* node lifetime, we chose to use a latching solenoid for micro-sprinkler actuation, requiring only a 25ms pulse of positive (to open) or negative (to close) voltage. The h-bridge is usually used to produce a bi-directional current to control the latching

[3]Decagon devices. http://www.decagon.com/products/soils/

solenoid [30]. However, it needs a special design to meet different voltages requirements for the ESP32 and latching solenoid.

In order to control the latching solenoid, we design a circuit diagram using two relays to operate with a very little connection overhead. A relay is an electrically operated switch. Figure 9 shows the turn-on and off circuit diagram for latching solenoid. When both the relays are off, there is no current going through the solenoid (S). Initially, both the relays are in a normally closed (NC) position. To turn the solenoid on, Relay 1 is switched from NC to normally open(NO) for 25ms, providing the positive current pulse through the solenoid. The current path shown in Figure 9(a) is: VCC -> NC_1 -> COM_1 -> S -> COM_2 -> NO_2 -> GND. To turn the solenoid off, Relay 2 is switched from NC to NO for 25ms, de-latching the solenoid to the closed position. The current path shown in Figure 9(b) is: VCC -> NC_2 -> COM_2 -> S -> COM_1 -> NO_1 -> GND. To prevent over-irrigation in the event of a power failure, we have the power supply module to continuously provide the power.

Power Supply: Power supply consisted of a 5v, 1.2W solar panel for energy-harvesting and a 18650 Lithium Li-ion battery with a capacity of 3.7V 3000 MAH for energy storage. The TP4056 lithium battery charger module comes with circuit protection and prevents battery over-voltage and reverse polarity connection. All sensors (1 ESP32, 2 moisture sensors, 2 relays and 1 latching solenoid) are powered with this power supply module. It can provide continuous power to prevent over-irrigation in the event of a power failure for the actuator module.

Transmission Module: Transmission includes uplink and downlink. In the uplink path, the moisture sensor readings from the field are sampled by the ESP32, a low-cost, low-power system on a chip (SoC) series with Wi-Fi capability. The readings are then sent from ESP32 to the base station as input for the optimal control. In downlink path, the control command calculated by the DRL agent will be routed to all ESP32 to turn on or off the solenoids.

5 IMPLEMENTATION

In this section, we illustrate in detail the implementation of *DRLIC* and tuning hyper-parameters.

***DRLIC* Implementation Details** We implement *DRLIC* in python using widely available open-source frameworks, including Pandas, Scikit-learn and Numpy. The control scheme - *DRLIC* is implemented using the scalable reinforcement learning framework, RLlib [31]. RLlib supports TensorFlow, TensorFlow Eager, and PyTorch. RLlib provides multi-ways for us to customize the training process of the target environment modeling, neural network modeling, action set building and distribution, and optimal policy learning. The 10-year weather data (2010-2020) are collected for *DRLIC*, with 9 years used for training and the remaining 1 year used for testing. In our implementation of *DRLIC*, we use the Adam optimizer for gradient-based optimization with a learning rate of 0.01. The discount factor is 0.99. The neural network model is 2 hidden layers with 256 neurons for each. The local server for training and running *DRLIC* is a 64 bit quad-core Intel Core i5-7400 CPU at 3.00 GHz that runs Ubuntu 18.04.

Training Details and Tuning Hyper-parameters. The performance of *DRLIC* agent is sensitive to the hyperparameter values chosen. Unfortunately, there is no simple approach that allows *DRLIC* agent to understand whether a specific value for a given parameter would improve total reward. To address this issue and further increase *DRLIC* 's performance, we leverage a tuning approach to optimize the *DRLIC* 's hyperparameters, such as λ, μ associated with rewards and penalties, and the learning rates. In particular, we employ grid search which allows us to specify the range of values to be considered for each hyper-parameter. The grid search process constructs and evaluates our model using every combination of the hyper-parameters. Finally, we employ cross-validation to evaluate each learned model.

6 EVALUATION

In this section, we evaluate the performance of *DRLIC* in the field. We evaluate *DRLIC* system for 15 days in the real world.

6.1 Experiment Setting

6.1.1 Baseline Strategy: We compare *DRLIC* to two state-of-the-art irrigation control schemes introduced in Section 7.

ET-Based Irrigation Control [6]. To implement an ET-based controller, we query a local weather station for the previous day's ET loss. To compensate for the loss, we use the sprinkler's irrigation rate provided by its dataset to calculate how long the system should be activated for irrigation.

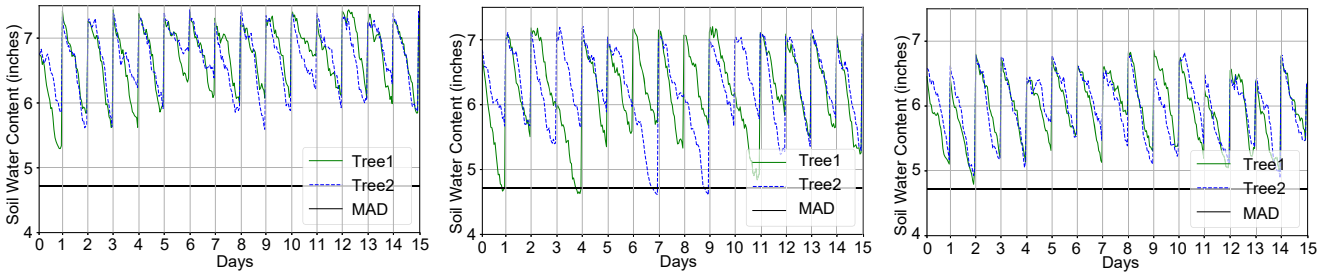
Sensor-based Irrigation Control [7]. The sensor-based controller has two thresholds, the lower and upper soil water content levels. The first is set at 4.96 inches, 10% higher than MAD to avoid the under irrigation occurring prior to the wetting front arriving at the sensor depth. The latter is set to 6.97 inches, 5% below FC to allow for some rainfall storage. We carefully set these two thresholds based on the soil environment of our testbed.

6.1.2 Performance Metrics. We evaluate the performance of *DRLIC* and two baseline systems in terms of two performance metrics.

Quality of Service. Although the irrigation system has no control over solar exposure and soil nutrients, it has direct control over the moisture levels in the soil. For this reason, our primary metric for irrigation quality is the system's ability to maintain soil moisture above this MAD threshold at all times at all of our measured locations. By doing so, we are guaranteeing that the plant has sufficient moisture to be healthy and no production loss. In this paper, we call this the quality of service of the irrigation system.

Water Consumption. As each sprinkler uses a water supply and we directly control the times at which each micro-sprinkler is active, we can monitor the amount of water consumed by these three systems at all times to determine the efficiency of each system. Thus another metric is the water consumption, which we would like to minimize subject to the quality of service constraints.

6.1.3 Experiments in our Testbed. We validate the *DRLIC* system with baselines in real-world deployment in terms of plant health and water consumption for 15 days. In the case study, we have six almond trees in our testbed as shown in Figure 7. *DRLIC*, sensor-based control and ET-based control are used to irrigate the upper, middle and lower two trees separately since there is no runoff between trees in our testbed. To allow three irrigation systems to operate independently, Every micro-sprinkler is controlled by



(a) ET-based Method (b) Sensor-based Method (c) *DRLIC*

Figure 10: Daily Soil Water Content of Different Irrigation Methods (15 Days).

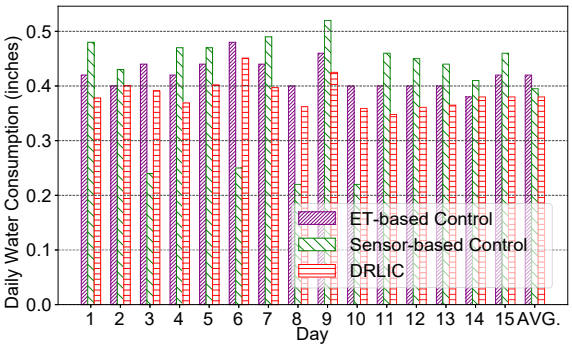


Figure 11: Daily Water Consumption.

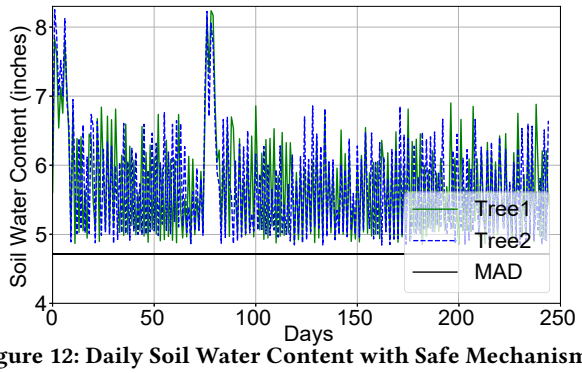


Figure 12: Daily Soil Water Content with Safe Mechanism.

a *DRLIC* node. In this way, the only difference among the three systems is the schedules sent to the nodes.

6.2 Experiment Results

6.2.1 Quality of Service. Irrigation systems are installed to maintain almond health with no production loss. Figure 10(a), 10(b), and 10(c) shows the daily soil water content in the field for ET-based control, Sensor-based control and *DRLIC*. The black horizontal line shows the MAD level. If soil water content is below this line, tree health will be impacted. We can see that *DRLIC* and ET system can maintain the soil water content above MAD threshold during the 15 days deployment and thus meet the requirement of almond health. However, the trees irrigated by Sensor-based method are in an under-irrigation period of 18 hours for four days (day 1, 4, 7 and 9) since the soil water content of Sensor-based method is lower than the MAD. The reason is that the moisture level of previous day is close but not reached to MAD, so the sensor-based method will not irrigate even though the moisture level is in an under-irrigation trend. *DRLIC* system can irrigate what the trees need based on the learned model about the water changes in the soil and maintain the soil water content close to MAD level.

All three underlying irrigation systems begin with enough water content on the first day. We see that the soil water contents of the two trees in ET control system are much above the FC threshold. In our deployment of *DRLIC* against the ET control strategy in Figure 10(a), we see that soil water content for these two trees is different and much higher than the MAD level. This emphasizes the limitations of ET and the core of our work. The irrigated regions don't receive moisture the same way, and most of the time, the ET-based controller irrigates more water than the plant needs.

6.2.2 Water Consumption. When a decision must be made to switch to a new almond irrigation control system, a primary concern is the efficiency of the proposed system. The system's ability to return its investment based on increased efficiency will often dictate the acceptance of the technology. In addition, the environmental benefits of reduced freshwater consumption are clear and help promote system adoption.

In our experimental setup, the water source provided by each micro-sprinkler is pressure-regulated to the industry standard, 30 psi. Each micro-sprinkler head distributing water uses a clearly-defined amount of water per unit time, as described in the almond irrigation manual [6]. By tracking exactly when each micro-sprinkler is actuated by the system, we can determine very accurately how much water has been consumed.

Figure 11 shows the daily irrigation amount of two trees actuated by ET-based control, sensor-based control and *DRLIC* in a 15 days' deployment experiment. From this figure, we can see that *DRLIC* can save an average 9.52% and 3.79% of the water compared with ET-based and Sensor-based control during 15 days deployment experiment. ET-based control is a centralized control method to irrigate all almond trees without considering their specific need. Sensor-based control is water-efficient by monitoring the moisture and irrigating when the moisture level is lower than the MAD level. However, the thresholds are site-specific and not optimal. *DRLIC* can learn optimal irrigation control by interacting with the local weather and soil water dynamic environment.

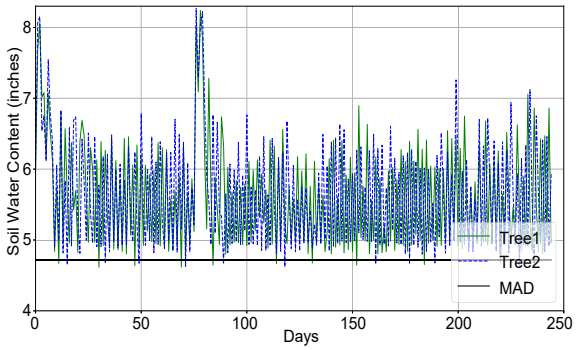


Figure 13: Daily Soil Water Content (w/o Safe Mechanism).

6.3 Effect of our Safe Irrigation Mechanism.

In the 15 days' deployment, we find that there are two days (Day 2 and 14 in Figure 10(c)) *DRLIC* triggers the ET-control method. This can also be validated from Figure 11, we can find the water consumption of ET method and *DRLIC* on days 2 and 14 are the same. We check the weather data to understand the reason and find that the wind speeds of days 2 and 14 are 7.2 and 11.9 mph receptively which is much higher than the average 2.8 mph of the other 13 days.

We now run *DRLIC* with and without safe mechanism for a whole growing season in simulation, labeled as Robust-RL and RL-only, respectively. Figure 12 and 13 show the daily soil water content of Robust-RL and RL-only for a same growing season 2020, respectively. From the almond's perspective, Robust-RL maintains health with 0 days below the MAD level. The RL-only irrigation method has 21 days below the MAD level. The reason is that the RL models trained from past weather data "misbehave" on the test weather data. while it may be possible to train on changing weather to obtain a robust policy, no offline training can ever cover all possible weather changes. The RL agent with safe mechanism from *DRLIC*, however, is robust to weather changes because the safety condition detector will detect the dangerous actions from RL agent and the ET system will take control.

6.4 Effect of proposed Reward.

In this section, we discuss the simulation results of *DRLIC* with different rewards for a whole growing season (March 1st to October 31st, 246 days).

In order to minimize the water consumption while not affecting the plant health, we consider three situations in the reward. 1) The soil water content (V_i) is higher than the FC (V_{fc}) level. 2) V_i is between V_{fc} and V_{mad}. 3) V_i is lower than V_{mad}. Only in the second situation, the plants are in good health. To evaluate our reward function, we compare it with a simple reward ($DRLIC_{MAD}$) that only maintains V_i above V_{mad}. It is commonly used in the sensor-based method [7]. The reward is defined as: $R = -\sum_{i=1}^{N} \lambda_3 * (V_{mad} - V_i) + \mu_3 * a_i, V_i < V_{mad}$. This function gives more penalty to plant health when V_i lower than V_{mad} since plants' health is significantly impacted. All the parameters are the same in Section 3.3.3.

Figure 17 shows the water consumption of *DRLIC* with our proposed reward (*DRLIC*) and the simple reward (*DRLIC* _MAD). *DRLIC* can save 2.04% more water than *DRLIC* _MAD, as the latter does not consider the case when V_i is higher than V_{mad}. *DRLIC* considers two more situations by giving different penalties to plants' health and water consumption. The first case is over-irrigation. The water consumption is too high. Therefore, the penalty for water consumption is higher than plant health. In the second case, the plants are in good health. *DRLIC* strives to maintain the V_i close to V_{mad} to save more water.

Table 3: Micro-sprinkler Node Manufacture Cost.

Component	Price	Component	Price
Moisture Sensor x 2	$250	ESP32	$6.5
18650 Li-ion battery	$3	Solar Panel	$4.3
Latching Solenoid	$4	Switch Relay x 2	$5
Waterproof Enclosure	$12	Maintenance Fee	$10
		Total	$294.8

6.5 *DRLIC* Policy Convergence.

Figure 14 shows the RL training process and the policy converges around the 500th training iteration. We define the length of an episode as 30 days. We randomly vary the soil water content for each tree between the FC (7.08 inches) and MAD (4.72 inches) at the beginning of each episode. By doing so, the policy is exposed to different soil water content conditions and learns to avoid water depletion than the MAD level during training. At the beginning of the experiment, the RL policy receives a larger negative reward as it does not know a valid sequence of actions that maximize the reward. The policy converges at the 500th training iteration. The whole training (i.e. 1000 training iterations) takes $\sim$ 4 hours using a 64-bit quad-core Intel Core i5-7400 CPU at 3.00 GHz.

6.6 Energy Consumption of Sensor Nodes

From a wireless sensor network standpoint, the ability of a system to operate for a long period of time without user intervention is fundamental. *DRLIC* nodes are no different, especially if they are meant to be put on the ground. For this reason, our hardware and software were designed to consume as little energy as possible. *DRLIC* nodes were fitted with a latching solenoid, allowing the flow of water to be turned on or off with a short pulse of power, rather than a constant supply. For additional energy savings, the radio in each node is duty-cycled, activating for only a 10 second period every 1 minute. We need this high data frequency, the reason is that the base station can send an off command to *DRLIC* with a minute granularity. In our devices, the four peripherals that consume significant energy are the two moisture sensors, solenoid, two relays and radio. To meet this energy, we design an energy harvesting mechanism by leveraging one 5/6 V 1.2 W solar panel.

Figure 15 shows the energy consumption for different sensors. Each moisture sensor sample requires 10 mA of power for 10 ms, and each flip of the latching solenoid requires 380 mA of power for 30ms. The ESP32 radio requires 180 mA of power for 50ms when in transmitting mode. The relay requires 250 mA for 20 ms for switching on or off. In our system, to ensure we don't cut power too early, we add a safety band of 50% on the timing on both of these devices, triggering for 15 ms and 45 ms for the sensor and solenoid, respectively. Overall, the solar-harvest mechanism can meet the daily requirement of all the sensors in *DRLIC* node.

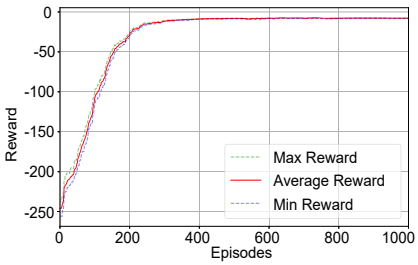


Figure 14: Reinforcement Learning Policy Convergence.

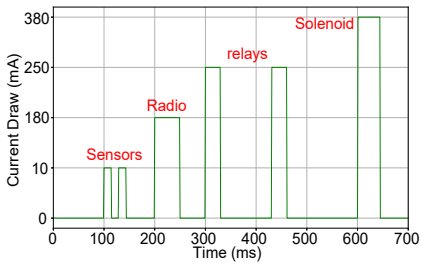


Figure 15: Energy Profile for Different Kinds of Sensors.

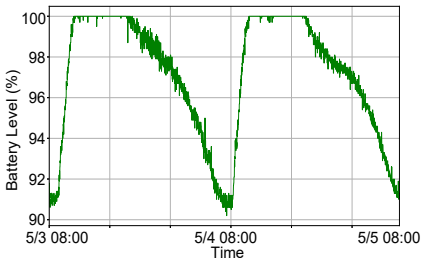


Figure 16: Battery Charging and Discharging Cycle.

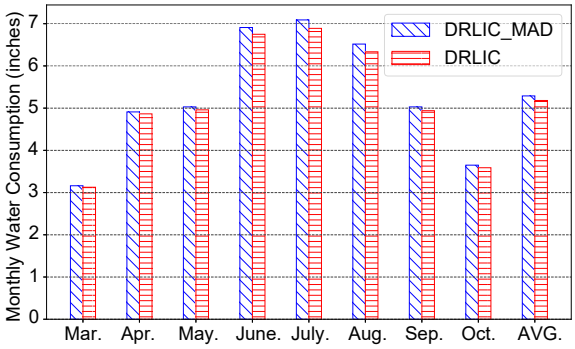


Figure 17: Water Consumption for *DRLIC* with Different Reward

Figure 16 shows the two days' energy charging and discharging process. After a night discharge, the 18650 battery level is increasing at 9:15 am on May 3rd. It usually takes 2 hours to fully charge the battery (9:15 - 11:35 am). The battery level will keep 100% from 11:35 am to 18:45 pm, the energy harvested from solar can meet the energy requirement of all sensors in *DRLIC* node. The battery will discharge from 100% at 18:45 pm of May 3rd to the 90.7% at 8:45 am of May 4th. Then the whole energy charging and discharging process repeat. The lowest battery level is an average of 90%. In the 2 week's deployment, we find that even on a cloudy day, the battery can also be charged and will take one more hour to be fully charged.

6.7 Return on Investment

A primary concern to purchasing or upgrading an irrigation control system is the return on investment, i.e., how long does it take to save enough money from water consumption to cover the cost of the new irrigation system. To calculate the return on investment of *DRLIC*, we take into account the initial investment cost of the *DRLIC* system and the money saved from the less water consumption provided by our increased irrigation efficiency.

We first calculate the cost to develop a single *DRLIC* node. All the components of a *DRLIC* node can be found in a consumer electronics store and a home improvement store. Table 3 lists the cost of all components. In total, a *DRLIC* sensing and actuation node costs $294.8. A large portion of the budget is the cost of two soil moisture sensors. We use two expensive soil moisture sensors that provide accurate measurement and a long lifetime.

The factors that mostly influence the payback of our system are water price and water volume saved by *DRLIC*. Water price varies considerably in different irrigation district and over time. This study assumed 100% groundwater usage and availability. Each tree costs $11.3 for irrigation water per month. Based on our experiment results, *DRLIC* can save 9.52% of water expense per month, corresponding to $1.08. Normally, almond orchards have 100 trees per acre. As a result, *DRLIC* can save $108 per month. Take a 60-acre almond orchard with 10 irrigation regions as an example. Each irrigation region is six acres. *DRLIC* can save $648 in each irrigation region per month.

In each irrigation region, we need to deploy one *DRLIC* node, which costs $294.8. The other irrigation components will use the existing infrastructure, such as the pipelines and micro-sprinklers under each tree. The cost of upgrading the existing irrigation system with our irrigation control system is $294.8 for one irrigation region in an orchard. Every month, our system can save $648. Therefore, it only needs half a month for our irrigation system to return the investment.

7 RELATED WORK

ET-Based Irrigation Control. As the weather is a primary water source or sinks in an irrigated space, systems have been developed to use weather as input for control. The simplest of these systems use standard fixed-schedule irrigation, but allow a precipitation sensor to override control to save water during rain. The more complicated systems, now the industry standard, use evapo-transpiration (ET), an estimate of the amount of water lost to evaporation and plant transpiration to do efficient water-loss replacement [32]. Some providers boast an average 30% reduction in water consumption, but as with all industry irrigation systems, ET-based systems are limited by centralized control, and can not provide site-specific irrigation, reducing potential system efficiency and quality of control.

Sensor-based Irrigation Control. With the introduction of more accurate and efficient soil moisture sensors, work has been done to create irrigation controllers that react directly to moisture levels in the soil [7]. Moisture sensors buried in the root zone of trees accurately measure the moisture level in the soil and transmit this data to the controller. The controller then adjusts the pre-programmed watering schedule as needed. There are two types of soil moisture sensor-based systems: 1) Suspended cycle irrigation systems. Suspended cycle irrigation systems use traditional timed

controllers and automated watering schedules, with start times and duration. The difference is that the system will stop the next scheduled irrigation cycle when there is enough moisture in the soil. 2) Water on-demand irrigation requires no programming of irrigation duration (only start times to water). This type maintains two soil moisture thresholds. The lower one to initiate watering, and the upper one to terminate watering [7]. However, without a model of the way water is lost, these thresholds are usually set based on experience and are not optimal.

Model-based Irrigation Control. In [30], a mechanistic PDE model of moisture movement within irrigated space is built. Using this model, an optimal watering schedule can be found to maintain a proper moisture level. However, the PDE model is not updated over time and future weather prediction is not taken into account. To tackle these two limitations, the same authors further improve the control system in [14]. The PDE model is eschewed in favor of an adaptive approach that involves models trained from sensor data. Long-term and short-term models are developed to describe the relationship of runoff between sprinklers in the movement of water through the soil.

As indicated by the authors [14], their system is designed for turf irrigation, and it is unlikely to provide benefit in shrubbery or tree irrigation. First, the turf soil moisture is affected by water runoff on soil surface and the overlapping coverage of sprinklers. The models in [14] are focused on capturing the relationship of runoff between sprinklers. For tree irrigation, however, there is little runoff due to the tree space. The soil moisture model for tree irrigation needs to consider the soil-water relationship under different depths. Second, as shown in [33], the decay of volumetric water content derived from the long-term model of [14] was shown to be much quicker than the real-world scenarios. It is bound to irrigate lightly and frequently, which has been found to be inefficient [34].

DRL-based Control. DRL has been applied in many applications, such as network planning [16], cellular data analytics [17], sensor energy management [35], mobile app prediction [36, 37] and building energy optimization [20, 38]. However, this paper tackles some unique challenges for irrigation control. First, we define an irrigation reward function that considers three cases for tree irrigation, as introduced in Section 3.3.3. Second, to prevent any possible damage to plants' health, we adopt a safe mechanism that replaces some unwise actions generated by DRL agent. Third, due to the data inefficiency, we leverage a data-driven simulator to speed up the training process.

8 CONCLUSIONS

We present *DRLIC*, a DRL-based irrigation system that generates optimal irrigation control commands according to current soil water content, current weather data and forecasted weather information. A set of techniques have been developed, including our customized design of DRL states and reward for optimal irrigation, a validated soil moisture simulator for fast DRL training, and a safe irrigation module. We design *DRLIC* irrigation node and build a testbed of six almond trees. Extensive experiments in real-world and simulation show the efficiency of *DRLIC* system.

9 ACKNOWLEDGMENTS

We would like to thank our anonymous shepherd and reviewers for their constructive comments. We also thank Danny Royer for helping us set up the testbed. This research is partially supported by the National Science Foundation under grants #CCF- 2008837, a 2020 Seed Fund Award from CITRIS and the Banatao Institute at the University of California, and a 2022 Faculty Research Award through the Academic Senate Faculty Research Program at the University of California, Merced.

REFERENCES

[1] Field capacity. https://nrcca.cals.cornell.edu/soil/CA2/CA0212.1-3.php.
[2] R Troy Peters, Kefyalew G Desta, and Leigh Nelson. Practical use of soil moisture sensors and their data for irrigation scheduling. 2013.
[3] Soil quality indicators. https://www.nrcs.usda.gov/Internet/FSE_DOCUMENTS/nrcs142p2_053288.pdf.
[4] Julian Fulton, Michael Norton, and Fraser Shilling. Water-indexed benefits and impacts of california almonds. *Ecological indicators*, 96:711–717, 2019.
[5] 2018 Almond Board of California. Water footprint for almonds. https://almonds.com/sites/default/files/2020-05/Water_footprint_plus_almonds.pdf.
[6] Almond Board of California. Almond irrigation improvement continuum. https://www.almonds.com/sites/default/files/2020-02/Almond-Irrigation-Improvement-Continuum.pdf.
[7] GL Grabow, IE Ghali, RL Huffman, et al. Water application efficiency and adequacy of et-based and soil moisture–based irrigation controllers for turfgrass irrigation. *Journal of irrigation and drainage engineering*, 2013.
[8] Xianzhong Ding and Wan Du. Smart irrigation control using deep reinforcement learning. In *ACM/IEEE IPSN*, 2022.
[9] Yenny Fernanda Urrego-Pereira, Antonio Martínez-Cob, and Jose Cavero. Relevance of sprinkler irrigation time and water losses on maize yield. *Agronomy Journal*, 2013.
[10] Dilini Delgoda, Hector Malano, Syed K Saleem, and Malka N Halgamuge. Irrigation control based on model predictive control (mpc): Formulation of theory and validation using weather forecast data and aquacrop model. *Environmental Modelling & Software*, 2016.
[11] Camilo Lozoya, Carlos Mendoza, Leonardo Mejía, Jesús Quintana, Gilberto Mendoza, Manuel Bustillos, Octavio Arras, and Luis Solís. Model predictive control for closed-loop irrigation. *IFAC Proceedings Volumes*, 2014.
[12] Bruno Silva Ursulino, Suzana Maria Gico Lima Montenegro, Artur Paiva Coutinho, et al. Modelling soil water dynamics from soil hydraulic parameters estimated by an alternative method in a tropical experimental basin. *Water*, 2019.
[13] California department of water resources. https://www.cimis.water.ca.gov/.
[14] Daniel A Winkler, Miguel Á Carreira-Perpiñán, and Alberto E Cerpa. Plug-and-play irrigation control at scale. In *ACM/IEEE IPSN*, 2018.
[15] Haoyu Niu, Dong Wang, and YangQuan Chen. Estimating actual crop evapotranspiration using deep stochastic configuration networks model and uav-based crop coefficients in a pomegranate orchard. In *Autonomous Air and Ground Sensing Systems for Agricultural Optimization and Phenotyping V*. International Society for Optics and Photonics, 2020.
[16] Hang Zhu, Varun Gupta, Satyajeet Singh Ahuja, Yuandong Tian, Ying Zhang, and Xin Jin. Network planning with deep reinforcement learning. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021.
[17] Zhihao Shen, Wan Du, Xi Zhao, and Jianhua Zou. Dmm: fast map matching for cellular data. In *ACM MobiCom*, 2020.
[18] Miaomiao Liu, Xianzhong Ding, and Wan Du. Continuous, real-time object detection on mobiledevices without offloading. In *IEEE ICDCS*, 2020.
[19] Devanshu Kumar, Xianzhong Ding, Wan Du, and Alberto Cerpa. Building sensor fault detection and diagnostic system. In *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pages 357–360, 2021.
[20] Xianzhong Ding, Wan Du, and Alberto Cerpa. Octopus: Deep reinforcement learning for holistic smart building control. In *ACM BuildSys*, 2019.
[21] 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.
[22] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
[23] Artem Molchanov, Tao Chen, Wolfgang Hönig, James A Preiss, Nora Ayanian, and Gaurav S Sukhatme. Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors. In *IEEE IROS*, 2019.
[24] Christopher Berner, Greg Brockman, Brooke Chan, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[25] Su Ki Ooi, Iven Mareels, Nicola Cooley, Greg Dunn, and Gavin Thoms. A systems engineering approach to viticulture on-farm irrigation. *IFAC Proceedings Volumes*, 2008.

[26] Guotao Cui and Jianting Zhu. Infiltration model based on traveling characteristics of wetting front. *Soil Science Society of America Journal*, 82(1):45–55, 2018.

[27] Guotao Cui and Jianting Zhu. Prediction of unsaturated flow and water backfill during infiltration in layered soils. *Journal of Hydrology*, 557:509–521, 2018.

[28] Dilini Delgoda, Syed K Saleem, Hector Malano, and Malka N Halgamuge. Root zone soil moisture prediction models based on system identification: Formulation of the theory and validation using field and aquacrop data. *Agricultural Water Management*, 2016.

[29] George H Hargreaves and Zohrab A Samani. Reference crop evapotranspiration from temperature. *Applied engineering in agriculture*, 1(2):96–99, 1985.

[30] Daniel A Winkler, Robert Wang, Francois Blanchette, et al. Magic: Model-based actuation for ground irrigation control. In *ACM/IEEE IPSN*, 2016.

[31] Eric Liang, Richard Liaw, Robert Nishihara, et al. Rllib: Abstractions for distributed reinforcement learning. In *ICML*. PMLR, 2018.

[32] Richard G Allen, Luis S Pereira, Dirk Raes, Martin Smith, et al. Crop evapotranspiration-guidelines for computing crop water requirements-fao irrigation and drainage paper 56. *Fao, Rome*, 1998.

[33] Akshay Murthy, Curtis Green, Radu Stoleru, Suman Bhunia, Charles Swanson, and Theodora Chaspari. Machine learning-based irrigation control optimization. In *ACM BuildSys*, 2019.

[34] Light and frequent irrigation. https://www.usga.org/course-care/water-resource-center/our-experts-explain--water/is-it-better-to-irrigate-light-and-frequent-or-deep-and-infreque.html.

[35] Francesco Fraternali, Bharathan Balaji, Dhiman Sengupta, Dezhi Hong, and Rajesh K Gupta. Ember: energy management of batteryless event detection sensors with deep reinforcement learning. In *ACM SenSys*, 2020.

[36] Zhihao Shen, Kang Yang, Zhao Xi, Jianhua Zou, and Wan Du. Deepapp: a deep reinforcement learning framework for mobile application usage prediction. *IEEE Transactions on Mobile Computing*, 2021.

[37] Kang Yang, Xi Zhao, Jianhua Zou, and Wan Du. Atpp: A mobile app prediction system based on deep marked temporal point processes. In *IEEE DCOSS*, 2021.

[38] Xianzhong Ding, Wan Du, and Alberto E Cerpa. Mb2c: Model-based deep reinforcement learning for multi-zone building control. In *Proceedings of the 7th ACM international conference on systems for energy-efficient buildings, cities, and transportation*, 2020.