Farm-gym: A modular reinforcement learning platform for stochastic agronomic games

Odalric-Ambrym Maillard, Timothée Mathieu, Debabrota Basu

Farm-gym: A modular reinforcement learning platform for stochastic agronomic games

Odalric-Ambrym Maillard, Timothée Mathieu, Debabrota Basu

Équipe Scool, Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189- CRIStAL, F-59000 Lille, France
odalric.maillard@inria.fr

Abstract

We introduce `Farm-gym`, an open-source farming environment written in Python, that models sequential decision-making in farms using Reinforcement Learning (RL). `Farm-gym` conceptualizes a farm as a dynamical system with many interacting entities. Leveraging a modular design, it enables us to instantiate from very simple to highly complicated environments. Contrasting many available gym environments, `Farm-gym` features intrinsically stochastic games, using stochastic growth models and weather data. Further, it enables to create farm games in a modular way, activating or not the entities (e.g. weeds, pests, pollinators), and yielding non-trivial coupled dynamics. Finally, every game can be customized with .yaml files for rewards, feasible actions, and initial/end-game conditions. We illustrate some interesting features on simple farms. We also showcase the challenges posed by `Farm-gym` to the deep RL algorithms, in order to stimulate studies in the RL community.

1 Introduction: motivation and challenges

Managing a farm during a growing season requires the farmer to take a sequence of decisions in a field to reach her objective. Farmers take these decisions under uncertainty (e.g. weather) while incorporating observations and various interactions between entities of the agronomic ecosystem (plant, weeds, pests, etc.) on-the-go. Developing novel management strategies is a time consuming and expensive process that often requires trial and error. Recently, an increasing attention has been drawn to improve farm management with data-driven methods, such as machine learning.

Reinforcement Learning (RL) (Sutton and Barto 2018) is a popular paradigm of machine learning that models the problem of *decision making* under uncertainty. Hence, while interacting with a dynamical system, called environment. Development of RL algorithms for different applications often commence with developing simulators or gamified platforms that mimic the challenges of the real-world complex problems in computationally controllable manner. For example, Atari games (Bellemare et al. 2013) provided a stimulating benchmark to develop RL algorithms in the context of computer games. Mujoco (Todorov, Erez, and Tassa 2012) is proposed as a physics-driven engine for fostering RL algorithms for continuous control and robotics. OpenAI (Brockman et al. 2016) proposed a set of lightweight RL environments with a standardized Application Programming Interface (API), called OpenAI `gym`. Presently, `gym` API became a reference in the RL community to create standardized RL environments in order to compare performances of RL algorithms. Following `gym`, several platforms are developed to foster reproducible frameworks and improved algorithms. For example, the highway environment (Leurent 2018) aims at safe planning in the context of traffic with multiple vehicles and unknown dynamics, and NLPgym (Ramamurthy, Sifa, and Bauckhage 2020) focuses of Natural Language Processing challenges. Often these `gym` frameworks have deterministic environments which are unknown and reveal themselves through noisy observations.

Challenges. In contrast, growing plants in a farm is inherently *stochastic* as it involves external uncontrolled entities (e.g. weather, insects, weeds) and interaction between different entities (e.g. plants and soil). These interactions and impacts of the entities demonstrate a coupled dynamics, which is often hard to model. Since existing deep RL algorithms can handle stochasticity only in small environments, farming provides an interesting challenge to RL community.

Another key challenge is the ability to *observe* or *measure a specific entity*. In contrast to traditional gym environments, where everything can be measured or observed, each observation in agronomy incurs a cost that must be balanced with the need to act on the field (e.g. watering, harvesting). Hence, agronomic decision making invoke an observation-action trade-off, which is missing in RL literature. Hence, we develop `Farm-gym` to foster both the RL research, and interaction between agronomy and RL communities.

Contributions. We develop `Farm-gym`[1] with the aim to create a modular environment that mimics the dynamics of an agricultural farm but still is simple enough to control and diagnose. `Farm-gym` takes a gamification approach in contrast to the high-precision simulators whose goal is to accurately model an individual entity. Each farm is constructed with multiple entities in a *modular* way, which can be seen as a game. We can choose which dynamics of the environment to simulate, and also to combine these dynamics to study specific coupling phenomena. Hence,

[1] Code is available at: https://github.com/farm-gym/farm-gym

Farm-gym proposes many variants of the farming problem, which can serve to model particular challenges, such as non-stationarity, coupling, long-term planning, active observation, etc. Specially, the Farm-gym platform is designed to host a number of environments built by combining different modules, cost-constraint and scores, such that each environment can be considered as a separate game of various and progressive difficulty. This provides even a non-expert an opportunity to play with and study parts of the complex farming problem with simple games.

For RL researchers, the agronomy-inspired Farm-gym offers a bouquet of stimulating challenges to study and experiment with, such as intrinsically *stochastic* dynamics, interaction and *coupled* dynamics, the *observation-action trade-off*, and user-defined *score function*[2]. We experimentally demonstrate the impact of the coupled dynamics (Section 3), and the challenges encountered by classical deep RL algorithms (Section 4). Experimental results indicate that designing learning strategies capable of simultaneously solving many Farm-gym games will help pave the way towards real-life applicable RL.

While we provide a few initial games and entity modules, the platform allows easy addition of new games and modules (e.g. fungus, slugs). Hence, Farm-gym also offers the possibility to study games involving more real-like complex and refined interactions in future.

Related works. (Garcia 1999) first proposed an RL agent interacting with a crop simulator to learn to maximize wheat yield while restricting nitrogen fertilization. Recently, (Sun et al. 2017; Chen et al. 2021; Yang et al. 2020) use different RL algorithms to maximize the yield while learning the best irrigation schedules. (Trépos et al. 2014) learns a technical itinerary that considers the date of sowing and the potential weather conditions. *Unfortunately, none of these works provides an open source and standardized RL environment.*

In this context, a few gym environments are proposed that model various decision-making problems while trying to optimize the yield of a given crop. For example, CropGym (Overweg, Berghuijs, and Athanasiadis 2021) focuses on fertilization strategies. (Kemanian et al. 2022) introduces CyclesGym that simulates multi-year crop strategies (e.g. crop-rotation strategies). But often the models considered here are deterministic like traditional gym environments. Recently, (Gautron et al. 2022) introduces gym-DSSAT, an environment with maize fertilization and irrigation problems. It is backed by the DSSAT crop models, allowing accurate simulation of a wide range of real-world growing conditions. However, apart from the weather, all other processes are deterministic.

The existing works consider specific settings of interest and focus on optimizing simple strategies (e.g. date of sowing or fertilization) for maximizing yield output with accurate models of plants. They do not consider either interactions between different entities (e.g. weeds, insects and plants), or complete stochasticity as seen in real-world. In contrast, Farm-gym provides a holistic platform to study different entities and interactions in a farm under stochasticity and user-defined objectives.

[2]The score function is defined as the reward of the user minus the cost due to observation and intervention.

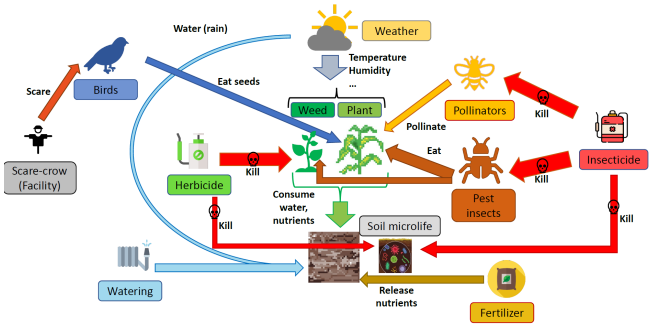

Figure 1: Interactions between Farm-gym entities.

2 Farm-Gym: An RL platform for solving gamified agronomy challenges

Modeling an agronomic system such as a farm, is a challenging task for several reasons. As mentioned before, a farm is an intrinsically *stochastic* system, e.g. subject to the variation of weather or pest attacks, with *interaction* of many entities at spatial (plant, crop, etc.) and temporal scales (day/night, week, month, year). This creates a *coupled dynamics* (predator-prey dynamics). Additionally, *observations* are *costly* and should be *actively* acquired in order to optimize a *risk-sensitive* and *personalized* objective functions. Here, we elaborate the components of Farm-gym.

Interacting entities. A Farm-gym environment consists of a set of *fields*. Each field has a pre-defined size and contains various *entities*. Entities are the physical components of the agrosystem, e.g weather, soil, plants, pollinators, weeds. Entities also include fertilizers, pesticides, and herbicides that slowly diffuses in the soil, and *facilities*, which affect the farm's dynamics, e.g. a scarecrow deterring birds. *Each entity is a modular block, coming with its own state-variables, actions (observations, interventions), and dynamics*. The dynamics may depend on state-variables of other entities allowing coupled effects. The states and actions are of mixed, i.e. both continuous or discrete. Figure 1 illustrates the different entities currently implemented in Farm-gym together with the flow of interactions. Depending on each entity, different interventions that Farm-gym allows fall into the following categories:

- Plant: sowing, harvesting, removing a plant,
- Weeds: removing weeds,
- Soil: watering the soil,
- Fertilizer: fertilizing,
- Cides: scattering herbicide, scattering insecticide,
- Facility: adding a scarecrow.

Hence, we conceptualize a farm as a big dynamical system, with states given by the concatenation of all state-variables of all entities. Due to the high number of observation, a subset of the observations can be selected, either

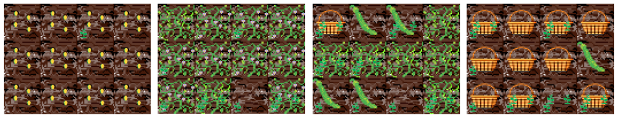

Figure 2: Single field, 3×4 bean farm at 4 different days.

to guide the learner or to model partially observed phenomena. Since a *learner* only accesses a subset of all internal variables, a `Farm-gym` problem is a Partially Observable Markov Decision Process (POMDP). This is similar in spirit to the real problems faced by farmers with unobserved factors. Unlike classical RL environments that are static POMDPs, here the system evolves even when no action is taken. A learner, while not completely knowing the dynamics, must find when to use each action depending on the task. While for classical MDP and POMDPs models, the states or observations are given for free to the learner, `Farm-gym` promotes the case of *actively-observed MDPs*.

Formally, the *actions* are *interventions* defined with entities, and *observations* that request the current value of a specific state variable. The observations can incur some cost depending on the score function. Finally, a *reward* can be computed when a game stops. Terminal (or end-game) conditions are defined as Boolean functions computed on the state variables. Typical end-game conditions include reaching a certain day of the year, or when all plants are dead or harvested. But more complex conditions can be defined. The full user-guide is available on the github page[3].

Modular environment. Modular design of `Farm-gym` allows customizing agronomy games with co-interacting entities. For example, Figure 2 demonstrates evolution of a single Plant (bean) and Weeds.

In `Farm-gym`, each instantiated farm generates configurable (YAML) files to specify the subset of allowed actions, the cost of each action/observation, the rewards, and the initial conditions. For instance, beyond the classical yield, `Farm-gym` allows to include the soil health index, a biodiversity index in the reward, etc.

Interestingly, `Farm-gym` also allows generating games with several entities of the same type, such as different plants for a poly-culture agrosystem (e.g. carrots and beans), different types of fertilizers, etc. This is to facilitate modelling poly-culture or crop-rotation games. Specifically, each entity comes with parameters, that can be instantiated in a YAML file for various instances. Currently, `Farm-gym` has three plants (corn, bean, tomato), three type of soils (clay, loam, sand), two cides (insecticide, pesticide), and two fertilizers (slow, fast diffusion). Each of them are different instances of the same entity with different parameters. We are also working on incorporating more instances.

Highly stochastic environment. In `Farm-gym`, the weather can be loaded from historical data to which stochastic noise is added.

Further, as many entities depend on some "favorable conditions" to transit from one state to another, we find it convenient to introduce a generic function in order to model this principle for different entities. Formally, we introduce an exponential general linear function

$$\mathcal{E}(\beta, y, \mathcal{Y}) = \exp\left(-\beta_0 - \sum_{j \in \mathcal{J}} \beta_j d(y_j, \mathcal{Y}_j)\right),$$

where $\beta = (\beta_0, (\beta_j)_j)$ is a vector of weight parameters, $y = (y_j)_j$ is a vector of real-values, $\mathcal{Y} = (\mathcal{Y}_j)_j$ a set of intervals in $\mathbb{R}$, and $d(y, \mathcal{Y})$ denotes the distance from y to the set $\mathcal{Y}$. Each entity transition (e.g. from seed to sprout, etc.) is modeled using a probability distribution that follows an aforementioned exponential model. This allows building an intrinsically stochastic environment. It should be noted that these plant models are inspired by characteristics of real plants. However, the goal of `Farm-gym` is not to reproduce accurate high-precision dynamics of actual plants from an agronomy perspective. Rather, the goal is to offer a gamified and approximate version to create flexible models and algorithms that can be combined and extended.

Challenges. Using stochastic processes in the dynamics of each entity (e.g. stochastic growth models) challenges the learning algorithms. Indeed, today's algorithms (e.g. GO-explore (Ecoffet et al. 2019)) designed to explore and solve complex environments such as Montezuma (Bellemare et al. 2013) are based on function approximation using deep learning and are not designed to handle a stochastic dynamic. In a stochastic environment, one may not obtain the same trajectory twice when applying the same policy. As a result, algorithms struggle to explore and successfully solve a far-gym environment, as demonstrated in Section 4. On the other hand, strategies inspired from multi-armed bandit (Lattimore and Szepesvári 2019) are designed to handle noise, however they struggle with large state-action space dynamics. Hence, we expect `Farm-gym` to foster interesting crossbreeding between deep RL community focusing on scalable algorithms and deterministic environments, and RL theory community focusing on model-based RL and multi-armed bandits with stochasticity.

Large and structured action space. Each entity comes with its own set of actions, which can be parametrized. For instance, the action watering from entity soil requires the amount of water to be specified. Further `Farm-gym` supports *asking for a list of observations and interventions each day*. This blows up the action space to A^k, when allowing k many actions a day, in comparison to daily action space A. Hence, a farm game can feature a large number of actions. These actions are mixed, i.e. they take both continuous and discrete values. On the other hand, the action space is very structured, and some actions (e.g. removing weeds) directly affect only part of the system. But a learner has to learn this structure to optimize the actions. We believe this fosters research on studying reinforcement learning in the context of a large and structured action space.

Observation-action trade-off. In `Farm-gym`, the agent must perform a specific action (observation-step) with an associated non-negative cost in order to retrieve the relevant information, e.g. the amount of water in the soil, the average number of insects in the field, etc. before intervening. The

[3]https://github.com/farm-gym/farm-gym/blob/main/FarmGym.pdf

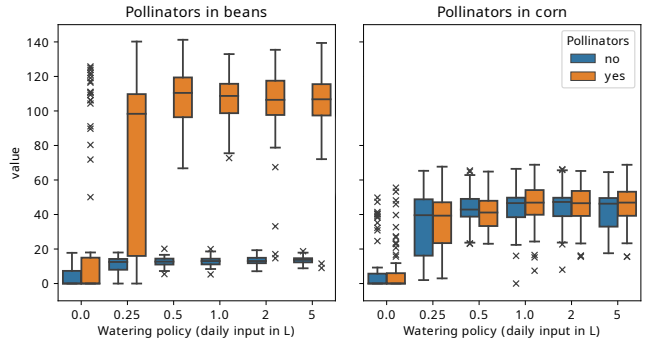

Figure 3: Effect of watering with or without pollinators. All plots are built using 100 runs. Reward is the yield.

agent must learn to identify when these additional observations are crucial to improve the management of the field, i.e. it must learn when to act for observing. To some extent this requires the agent to evaluate the state of its knowledge about the system and update its values by observing as a farmer would. This creates a non-trivial problem of *actively-observed MDPs*, which is hardly studied in RL community.

As we have summarised the functioning and challenges posed by `Farm-gym`, we experimentally instantiate them in the following sections.

3 Illustrating coupled dynamics in a farm

In this section, we illustrate basic results on a few farms featuring various entities: Soil, Plant, Pollinators, Weeds, Pests. They highlight the non-trivial dynamics that emerges, when multiple entities interact in `Farm-gym`. For simplicity of exposure, we use simple hand-crafted policies, all farms are 1×1, and the agent has access to all observations.

In Figure 3, for clay-type Soil, we illustrate the effect of having pollinators in the field. Different plants react differently to the pollinators. For instance, corn does not depend much on insect pollination, while beans depend a lot on it.

In Figure 4, we illustrate another coupling phenomenon between weeds and pests. While weeds compete with the plant in terms of nutrients and water, some can also protect the main plant from pests. In that case, removing the weeds may not be the best option. We consider the learner spreads a tiny amount of herbicide at different frequencies. At low frequency, the effect is to slow-down the developments of weeds. In contrast, while used at high frequency, herbicide kills the weeds, causing the pests to attack the plant only.

4 Challenges for training a deep RL agent

To investigate the challenges present when learning on farm-gym environments, we train a popular deep RL algorithm, PPO (Proximal Policy Optimization) (Schulman et al. 2017), on a basic farm. The farm consists of a 1x1 plot of land, the crop are beans, and the weather is from the north of France. We include bugs and weeds. The farmer can use water, herbicide, pesticide, and fertilizer, or do nothing. There are 9 possible actions with discrete values, and 17 observations which are, for simplification, in a box space. Further details are given in Appendix B. All the observations are free and

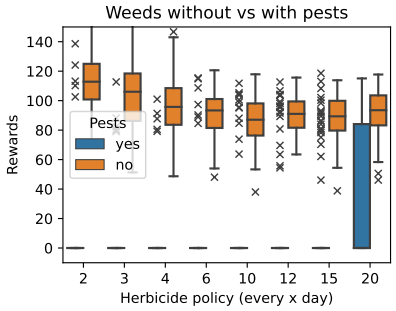

Figure 4: Effect of spreading low herbicide on weeds, with or without pest insects. Reward is the yield.

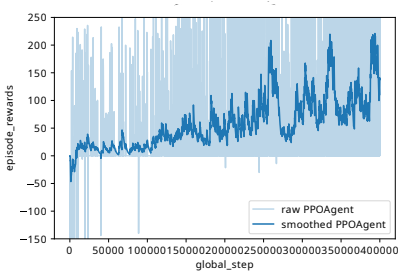

Figure 5: Training of one PPO agent on a basic farm. Reward is the yield penalized by low health index (details in App B).

the algorithm takes one action at each time step. In Figure 5, we present the cumulative reward observed while the training the PPO agent. We observe that the raw plot (represented in light blue in Figure 5) is highly variable. Thus, we use smoothing, similar to Tensorboard's[4] smoothing, in order to visualize the increase of the cumulative reward. This variability of the reward is a big challenge for PPO, and it is linked to the stochastic nature of the environment. The results indicate that PPO might not be the optimal choice for this environment as it does not take into account the time-dependent dimension of the problem. However, it gives a benchmark and shows that a classical RL algorithm manage to learn something on this environment.

On the other hand, we design an *expert agent* tailored to the same farm. This agent uses expert knowledge, such as beans should be planted after April, and there should be a small amount of pesticide and herbicide used before planting to clean the ground. The mean episodic reward, computed on 128 evaluation episodes and 15 seeds, is 433 ± 50 for the expert agent against 75 ± 79 for PPO. This shows that classical deep RL agents are still far from expert agent on this task, and poses a challenge to be solved in future.

Acknowledgments

We acknowledge H.D Carvajal and J. Teigny for their support and discussions. O-A. Maillard acknowledges the support of the Métropole Européenne de Lille (MEL), ANR, Inria, Université de Lille, I-SITE ULNE through the AI chair Apprenf number R-PILOTE-19-004-APPRENF. T. Mathieu acknowledges the support of the A.Ex Inria SR4SG.

[4] https://github.com/tensorflow/tensorflow

References

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Chen, M.; Cui, Y.; Wang, X.; Xie, H.; Liu, F.; Luo, T.; Zheng, S.; and Luo, Y. 2021. A reinforcement learning approach to irrigation decision-making for rice using weather forecasts. *Agricultural Water Management*, 250: 106838.

Ecoffet, A.; Huizinga, J.; Lehman, J.; Stanley, K. O.; and Clune, J. 2019. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*.

Garcia, F. 1999. Use of reinforcement learning and simulation to optimize wheat crop technical management. In *Proceedings of the International Congress on Modelling and Simulation (MODSIM'99) Hamilton, New-Zealand*, 801–806.

Gautron, R.; Padrón, E. J.; Preux, P.; Bigot, J.; Maillard, O.-A.; and Emukpere, D. 2022. gym-DSSAT: a crop model turned into a Reinforcement Learning environment. Research Report RR-9460, Inria Lille.

Kemanian, A. R.; Shi, Y.; White, C. M.; Montes, F.; Stöckle, C. O.; Huggins, D. R.; Cangiano, M. L.; Stefani-Faé, G.; and Nydegger Rozum, R. K. 2022. The Cycles Agroecosystem Model: Fundamentals, Testing, and Applications. *SSRN Electronic Journal*.

Lattimore, T.; and Szepesvári, C. 2019. *Bandit Algorithms*. Cambridge University Press.

Leurent, E. 2018. An Environment for Autonomous Driving Decision-Making. https://github.com/eleurent/highway-env.

Overweg, H.; Berghuijs, H. N.; and Athanasiadis, I. N. 2021. CropGym: a reinforcement learning environment for crop management. *arXiv preprint arXiv:2104.04326*.

Ramamurthy, R.; Sifa, R.; and Bauckhage, C. 2020. NLP-Gym – A toolkit for evaluating RL agents on Natural Language Processing Tasks. arXiv:2011.08272.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sun, L.; Yang, Y.; Hu, J.; Porter, D.; Marek, T.; and Hillyer, C. 2017. Reinforcement learning control for water-efficient agricultural irrigation. In *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*, 1334–1341. IEEE.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033. IEEE.

Trépos, R.; Lemarié, S.; Raynal, H.; Morison, M.; Couture, S.; and Garcia, F. 2014. Apprentissage par renforcement pour l'optimisation de la conduite de culture du colza. In *14e Journées Francophones Planification, Décision, Apprentissage pour la conduite de système-JFPDA*, 1–13.

Yang, Y.; Hu, J.; Porter, D.; Marek, T.; Heflin, K.; and Kong, H. 2020. Deep reinforcement learning-based irrigation scheduling. *Transactions of the ASABE*, 63(3): 549–556.

A Additional experiments on farm dynamics

Like many agronomy platforms, `Farm-gym` models the weather and soil conditions. To illustrate the effect of soil on the plant cycle, we explore a simple deterministic policy that waters the same amount each day and harvests when the plant reaches a ripe stage. In this experiment, the weather is fixed to an (artificial) dry weather. Thus, the water input only comes from watering. `Farm-gym` enables us to easily customize the weather with .csv files.

Figure 6 illustrates a clear difference between the type of soils. For example, with less that 2L of water input in the sand soil, the plant does not grow, while it grows in clay.

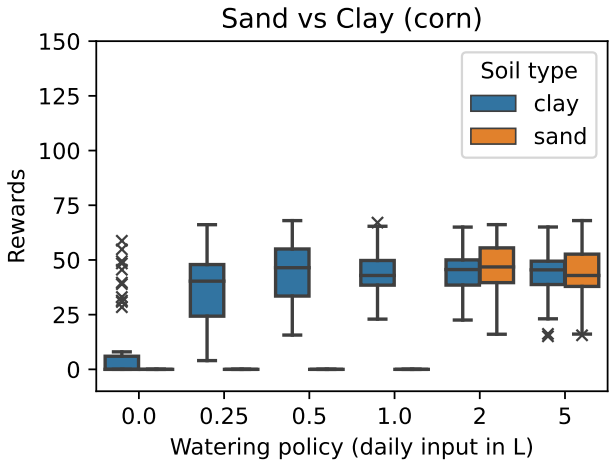

Figure 6: Effect of constant watering in different soils.

B Additional Experiments on learning with an RL-agent

The farm considered in Section 4 consists of an 1x1 farm, with beans in a loam ground in the north of France that has rainy and moderately cold weather. There are pests, weeds and pollinators. The farmer have 9 possible actions each day: doing nothing, watering 1L or 5L of water, sowing some seeds, harvesting, scatter some pesticides, scatter some herbicide, scatter some fertilizer, removing weeds by hand. There are 17 observations given freely each day: day (from 1 to 365), temperature (mean, max and min in °C), rain amount (categorical), sun-exposure (from 1 to 5), consecutive dry days, stage of growth of the plant (categorical), number of fruits (int), size of the plant (in cm), soil wet surface (m2.day-1), fertilizer amount (kg), pollinators occurrence (proportion) , weeds grow (int), weeds flowers (int), weight of fruits (in g), micro-life health index (in percent).

An episode begins with nothing in the soil, and it ends once either the plant has grown and has been harvested, or the plant is dead, or a year ends. The reward consists of the amount of fruit harvested (in g, given only once at the end of the episode), and a negative reward of -2 given each day for which the micro-life index went below 10%.

In Figure 7, we present the number of times each action is used during the evaluation of a trained PPO agent over 128 episodes. It shows that the agent can learn a sparse set of frequent actions to take. It also learns actions related to the micro-life index maintenance, such as not frequently using herbicides and pesticides, in addition to the ones leading to higher yields, such as watering and scattering fertilizers. This result shows an opportunity to learn lessons about farming using an off-the shelf RL agent and `Farm-gym`.

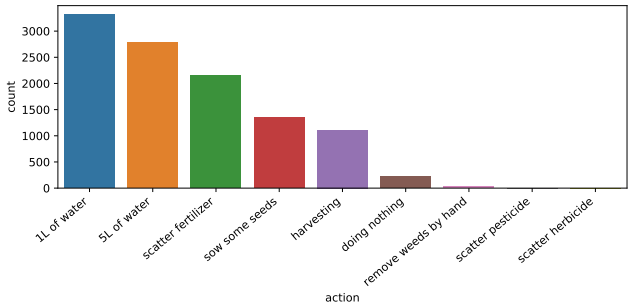

Figure 7: Actions of a given trained PPO algorithm taken during 128 evaluation episodes

In Figure 8, we present the cumulative reward observed while the training the DQN agent. Similarly to PPO, the raw plot is highly variable, hence we display its smooth version. As for PPO, the results indicate that DQN might not be the optimal choice for this environment.

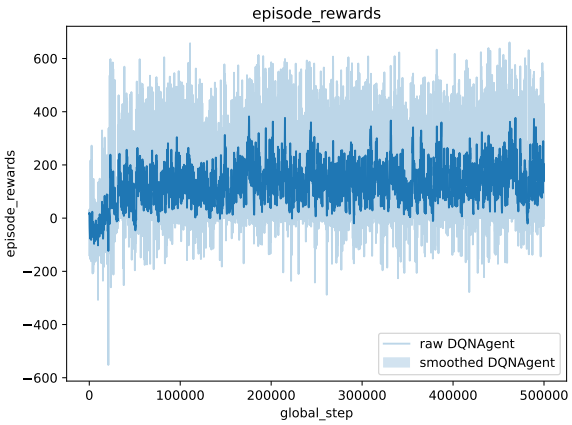

Figure 8: Training of one DQN agent on a basic farm.