Analyzing Model-Free Reinforcement Learning Algorithms for Nitrogen Fertilizer Management in Simulated Crop Growth

Michael Vogt
Computer Science and Applied Mathematics
University of the Witwatersrand
Johannesburg, South Africa
1533057@students.wits.ac.za

Benjamin Rosman
Computer Science and Applied Mathematics
University of the Witwatersrand
Johannesburg, South Africa
benjamin.rosman1@wits.ac.za

Abstract

Establishing intelligent crop management techniques for preserving the soil, while providing next-generational food supply for an increasing population is critical. Nitrogen fertilizer is used in current farming practice as a way of encouraging crop development, however its excessive use is found to have disastrous and long-lasting effects on the environment. This can be reduced through the optimization of fertilizer application strategies. In this work, we apply some of the recent best performing reinforcement learning algorithms - the DQN, Double DQN, Dueling DDQN, and PPO - to learn novel strategies for reducing this application in a simulated crop growth setting. We provide an analysis of each agents ability and show that the Dueling DDQN agent is able to learn favourable strategies for minimizing nitrogen fertilizer application amounts, while maintaining a sufficient yield comparable to standard farming best practice.

Introduction

The global need for innovative crop management techniques for a changing climate and growing population is paramount. Agricultural sectors now need to produce more food on less arable land, while also keeping the environmental impact to a minimum. The use of nitrogen fertilizer is increasingly used as a method for encouraging crop development, producing more yield while sustaining crop quality as the earth's resources are depleted. This rapid adoption has however led to nitrogen fertilizers being excessively applied beyond the crops demand, resulting in unfortunate consequences such as soil, water and air degradation. The long term application of nitrogen fertilizer increases the acidity in the soil and may render the soil infertile, causing crops to no longer respond to its application. Beyond this, groundwater contamination may cause severe health hazards for humans and livestock alike, while nitrous oxide emissions (N_2O) are considered 300 times more harmful than carbon dioxide emissions (CO_2) in terms of their potential for global warming (Sainju, Ghimire, and Pradhan 2019).

There is a subsequent need for establishing next-generation, sustainable application techniques for reducing this impact, as this has the potential to affect humanity on a global scale. The importance of this lies in the present unknown knowledge of whether current farming best practice techniques are indeed optimal, or whether they will transfer to new scenarios at all (Binas, Luginbuehl, and Bengio 2019). Performing and evaluating new strategies in practice however, can be challenging due to the long nature of plant development and the additional resource and labour costs involved.

There has been a significant impact of reinforcement learning applied to many domains with great success, however its application to agriculture has not been adequately researched, with its so far unrecognized potential still to be explored (2019) (2021). This research utilizes model-free reinforcement learning techniques, namely the Deep Q-Network (DQN), Double Deep Q-Network (Double DQN), Dueling Double Deep Q-Network (Dueling DDQN), and Proximal Policy Optimization (PPO) algorithms to explore novel nitrogen fertilizer application approaches to this problem in a simulated crop growth setting. We show that the Dueling DDQN agent is in fact able to learn comparable strategies to farming best practice techniques by reducing the amount of fertilizer applied, thereby limiting environmental impacts while still producing a sufficient yield. The main contributions of this research provide an analysis of the ability of each of these reinforcement learning methods to efficiently manage fertilizer application amounts. A subsequent contribution is providing an initial benchmark of model-free reinforcement learning methods to be extended upon within this domain.

Background

Reinforcement learning is a method of machine learning in which an agent learns, through trial-and-error interaction with its environment, how to behave in order to maximize a numerical reward signal (Sutton and Barto 2018).

Markov Decision Processes (MDP), represented as the tuple (S, A, R, P), are a way to formalize sequential decision making processes and provide a simple mathematical framework that defines the interaction between an agent and its environment in terms of its states S, actions A, and rewards $R(s, a)$. $P(s, a, s')$ are the transition dynamics and is the probability that an agent in a state s takes an action a and progresses to state s'. A policy $\pi(a|s)$ is a stochastic rule -

mapping states to actions - used by the agent to make decisions in order to maximize its discounted cumulative future rewards (2018). A policies state and action-value functions,

$$v_\pi(s) = \mathbb{E}_\pi[\Sigma_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t = s] \quad (1)$$

$$q_\pi(s,a) = \mathbb{E}_\pi[\Sigma_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t = s, A_t = a] \quad (2)$$

assign to each state and state-action pair respectively, the expected cumulative discounted return. Here, γ, is the discount factor and instills caution in the agent, due to the uncertainty within the environment and the agents ability to gain rewards in the future. These equations describe in an absolute sense, how good a given state or state-action pair may be for the agent and obey they special self-consistency equation, known as the Bellman Equation (Sutton and Barto 2018).

It is difficult to draw an accurate taxonomy of reinforcement learning algorithms, however they broadly fall into the groups of model-based and model-free reinforcement learning. The state transition dynamics $P(s,a,s')$ and return function $R(s,a)$ are generally unknown, and in model-based reinforcement learning an agent learns a model of this unknown knowledge, through experience, and uses this for planning - which may result in improved sample efficiency. This is naturally difficult and may cause bias in the model which the agent exploits, thus potentially resulting in sub-optimal policies when being applied to real-world applications. Model-free reinforcement learning methods can be further sub-divided into value-based, policy-based, and actor-critic methods. Value-based methods approximate the state or state-action value, from which the optimal policy is then derived. This is known as an off-policy method and differs from policy-based methods, in which the policy is directly approximated and improved. Actor-critic methods are a temporal-difference approach to policy-based methods. Here the learning of the actor is still based on directly optimizing the policy, but a critic is introduced to evaluate the choice of action and provide guided learning (Sutton and Barto 2018).

Reinforcement learning in and of itself has had limited applicability, confining it to domains in which useful features may be handcrafted or domains with low-dimensional, fully observable state-spaces (Mnih et al. 2015). More recently however, reinforcement learning has been combined with non-linear function approximators such as a neural networks or decision trees, allowing them to learn complex feature representations. The DQN algorithm is one such algorithm that has shown great success in playing games such as those in the Atari 2600 domain (2015). Recently, within agriculture, the DQN algorithm has been used to determine the optimal irrigation strategy for rice based on short term weather forecasts (Chen et al. 2021). It has further shown proven success for the intelligent control of agricultural irrigation in a greenhouse plantation in Hunan (Zhou 2020).

Deep Q-Network

Q-learning combined with a multi-layered neural network forms the basis of the DQN algorithm and this is used to approximate the optimal action-values, which may be recursively defined as

$$Q^*_\pi(s,a) = \Sigma_{s',r}P(s',r|s,a)[r + \gamma max_{a'}q^*(s',a')] \quad (3)$$

Reinforcement learning is known to be unstable when combined with a non-linear function approximator, often resulting in divergence. The cause of this divergence may be due to a number of factors, such as correlations in sequential observations present in reinforcement learning. Small updates to the non-linear function approximators may cause changes in the policy, thus changing the distribution of the data. Furthermore, within the DQN algorithm correlations between the action-values and the target values, $r + \gamma max_{a'}q(s',a')$, may also cause divergence (Mnih et al. 2015). To overcome this difficulty, the DQN algorithm has been combined with an experience replay buffer in which experiences of (s,a,r,s') are stored. During training, Q-learning updates are applied to a sample of mini-batches of this experience, which are drawn uniformly at random from the replay buffer. This breaks correlations experienced by a sequence of observations, resulting in independently and identically distributed data - which assists in convergence. Another issue involved is that the target in the action-value function update is an approximation itself. If this is updated at every iteration, the result is unstable learning as we are moving towards something that is itself an estimate and non-stationary. To overcome this, a separate Q-Network is initialized with fixed parameters, which only get periodically updated (Mnih et al. 2015). The pseudo-code for the DQN algorithm can be viewed below.

Algorithm 1 Deep Q-Network with Experience Replay

```
Initialize the replay buffer $\mathcal{D}$ with capacity $N$
Initialize action-value function $Q$ with random weights $\theta$
Initialize target action-value function $Q$ with weights $\theta^- = \theta$
for each episode do
    for $t = 1, T$ do
        Select a random action $a_t$ with probability $\epsilon$
        Otherwise, select $a_t = max_a Q(s,a;\theta)$
        Execute action $a_t$ and observe reward $r_t$ and next state $s_{t+1}$
        Store transitions $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{D}$
        Sample random mini-batches of $(s_j, a_j, r_j, s_{j+1})$ from $\mathcal{D}$
        Set $y_j = \begin{cases} r_{j+1}, & \text{if } s_{j+1} \text{ is terminal} \\ r_{j+1} + \gamma \max_{a'} Q(s_{j+1}, a'; \theta'), & \text{otherwise} \end{cases}$
        Perform stochastic gradient descent step on
        $L_\theta = (y_j - Q(s_j, a_j; \theta))^2$
        Update $Q_{\theta^-} = Q_\theta$ after every $K$ steps
    end for
end for
```

Double Deep Q-Network

One lingering problem with the DQN is that it suffers from over-estimated action-values. It was not previously known whether this overestimation causes sub-optimal policies in practice, however Van Hasselt, Guez, and Silver show that this is the case. To overcome this overestimation the authors devise the Double DQN algorithm, which they show leads to more accurate action-value estimates, as well as better policies. In the standard DQN algorithm, it is indeed the max operator which leads to overoptimistic value estimates, as this tends to prefer over-estimated to under-estimated action values. This is because the same parameter values are used both to select and to evaluate the chosen action. Their devised Double DQN algorithm therefore decomposes the max operator in the target into action selection and action evaluation. Here, the selection of the action is still due to the online weights of the behaviour policy, but the parameters of the target network are used to evaluate the value of this policy. The new target update leading to better estimated action-values is

$$y^{DDQN} = r + \gamma Q(s', max_a Q(s', a, \theta), \theta') \tag{4}$$

Dueling Architectures

The combination of reinforcement learning with neural networks have generally used conventional deep learning network architectures such as convolutional neural networks or LSTMs. Although showing great success, they may not be optimally suited for reinforcement learning tasks (Wang et al. 2016). In order to find architectures more suited to model-free reinforcement learning tasks specifically, Wang et al. present a new network architecture, which they term the dueling network architecture. The dueling network is composed of a single deep model with two streams for two separate estimators - the value function $V(s;\theta)$, and advantage function $A(s,a;\theta)$. The output of these two streams are then combined to produce the action-value estimate $Q(s,a;\theta)$. The value function is a measure of how good it is to be in a particular state, while the action-value function is a measure of how valuable choosing a particular action is when in this state. The advantage function is a relative measure of the importance of each action, and relates to the value and action-value functions by

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s,a) \tag{5}$$

The main benefit of the dueling network, as claimed by Wang et al., is that it is able to generalize learning across its actions without having to change the underlying architecture of the algorithm, thus making it suitable and easy to implement with a variety of model-free reinforcement learning methods.

PPO

The Proximal Policy Optimization algorithm is a policy-based method with an actor-critic style. The motivation behind the design of this algorithm is to maintain the data efficiency and reliability in performance of the Trust Region Policy Optimization algorithm, while improving sample efficiency by only using first-order optimization methods (Schulman et al. 2017). One of the main contributions of this method is introducing a new surrogate objective function, with clipped probability ratios, that enables multiple epochs of mini-batch updates without having destructively large policy updates. This objective function has the form

$$L^{CLIP}(\theta) = \mathbb{E}_t[min(r_t(\theta)A_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)], \tag{6}$$

where $r_t(\theta)$ is the ratio of the log probability of the new policy with the log probability of the old policy. From this it can be seen that the expectation is taken over two terms, the unchanged policy-gradient objective function and a clipped policy-gradient objective function. The clipped version is truncated by $1+\epsilon$ and $1-\epsilon$, where ϵ is a chosen hyperparameter. The output of these two terms are multiplied by the advantage function estimate, after which the minimum is taken to form a lower bound on the size of the policy update, thus preventing divergence (Schulman et al. 2017).

At each time-step, a sequence of N parallel actors run the old policy within the environment collecting data for T time-steps. Next the advantage estimates are calculated and mini-batches are sampled. The surrogate loss is constructed and is optimized using gradient ascent. K epochs of optimization are performed on the same trajectory data sample.

Methodology

Reinforcement learning applied to agriculture is a newly established field of interest. Despite its infancy, this has resulted in varying applications and approaches to smart farming such as yield prediction, crop quality monitoring, and weed detection (Liakos et al. 2018). Despite these successes, the path to sustainable agriculture requires more fine-grained control of the development of the crop itself, in terms of its surrounding environmental parameters (Binas, Luginbuehl, and Bengio 2019). The two main focus areas for recent advances in crop management using reinforcement learning are finding optimal strategies for irrigation and fertilizer management. This is because of their large effect on crop growth, as well as water being a limited resource, with fertilizers having a negative effect on soil health and the environment (Sainju, Ghimire, and Pradhan 2019). With more recent focus on learning irrigation strategies, fertilizer management in crop growth has received the least attention, despite its many negative effects. This motivates our investigation into the ability of model-free reinforcement learning to learn new strategies that may acquire similar yields, while limiting the disastrous effects of nitrogen fertilizer. In order to achieve this aim the simulated crop growth environment that overweg2021cropgym developed is used, termed Crop-Gym [1].

It is a nitrogen-limited crop growth environment in which the underlying crop growth processes involved are modeled using the LINTUL-3 package, provided in the Python Crop Simulation Environment (PCSE) [2]. The parameters of the crop growth model have been set to simulate a winter wheat crop. The state space consists of multidimensional observations of the current state of the crop as output by the LINTUL-3 package, as well as weather data over the past week. The action space consists of discrete doses of nitrogen fertilizer, which may be applied by an agent in time intervals of one week. While providing a strong restriction on the limit of nitrogen supply in terms of the quantity observed in the soil, Overweg, Berghuijs, and Athanasiadis do not restrict the amount of times fertilizer is allowed to be applied. This allows for exploration of strategies outside of standard farming strategies thus far developed by experts. This is sig-

[1]https://github.com/BigDataWUR/crop-gym

[2]https://github.com/ajwdewit/pcse

nificant, as best practice knowledge may not transfer well to new scenarios caused by climate change (Binas, Luginbuehl, and Bengio 2019).

To contrast the ability of model-free reinforcement learning algorithms to efficiently learn fertilizer management strategies, two baseline agents were implemented. The first baseline is termed the standard practice agent, and mimics current farming standard best practice approaches to applying nitrogen fertilizer. This agent applies three discrete doses of nitrogen fertilizer throughout the crop growth season, which spans 8 months - starting on January 1^{st} and ending on August 1^{st}. Together with this, a reactive baseline agent was implemented which applies a high concentration of fertilizer every time the nitrogen content in the soil is depleted.

Along with these baselines, the DQN agent was implemented, as it has been successfully used within the smart-farming domain for irrigation based decision making (Zhou 2020) (Chen et al. 2021). The DQNs successor variant, the Double DQN agent was also implemented, as it has been shown to achieve more reliable strategies, with more accurate action-value estimates (Van Hasselt, Guez, and Silver 2016). To investigate the potential performance gains of dueling architectures, and due to the the overestimation caused by the DQN, the dueling network architecture was implemented along with the Double DQN algorithm. Thus the third value-based method implemented was the Dueling DDQN agent. In Wang et al. they explain that simply using the definition of the advantage function to produce the action-value estimate is not good idea. This is because given the action-value estimate $Q(s, a)$, the value and advantage functions, $V(s)$ and $A(s, a)$, cannot be uniquely recovered. They further state that poor performances are achieved when using this expression, emulating this lack of identifiability in practice. During learning, action-value estimates were produced in the more stable manner proposed, namely:

$$Q(s,a) = V(s) + A(s,a) - \frac{1}{|A|}\Sigma_{a'}A(s,a) \tag{7}$$

In order to fairly compare and investigate the effectiveness of value-based methods for fertilizer management to previously researched work, the PPO agent was implemented. Overweg, Berghuijs, and Athanasiadis implement this agent, and it is the first used for investigating the ability of reinforcement learning to find comparable fertilizer application strategies in this domain. Through the investigation of these agents and comparing them to one another along with the baselines, we provide a full analysis of their ability, as well as a more encompassing initial benchmark for model-free methods to be built upon in the future.

Experiments

In order to analyze the performance of the implemented reinforcement learning agents, this chapter outlines the specifics of the environment, the characteristics of the baseline fertilizer application strategies, the details corresponding to the implementation of the reinforcement learning agents, and the metrics used to evaluate the performance of all agents. This is described in detail to allow for reproducability.

Environment

The crop growth environment is created as a fully observable MDP, meaning the agent can make informed decisions based on all observable state variables. At each time-step during training, the agent observes state variables that are output by the LINTUL-3 winter wheat crop model. This model is calibrated using initial soil, site, and crop conditions, as well a pre-defined agromanagement strategy to be used and weather data from the past week. The output of the crop growth model importantly includes observations such as the weight of the storage organ (WSO), the total rainfall experienced (TRAIN), and the total nitrogen uptake by the crop (NUPTT). A full list of observed crop and weather variables can be viewed in the Appendix.

In common farming practice, farmers intervene in intervals of seven days (Overweg, Berghuijs, and Athanasiadis 2021). This is maintained allowing the agent to apply one of seven discrete doses every week. Included in this action space is the ability to apply no nitrogen fertilizer at all, as this is desirable due to the costs involved and the negative long-term effects of nitrogen to the soil, water and air. The action space is related to the amount of fertilizer to be applied, in kg/ha, by the equation

$$A = \{20a\,\frac{kg}{ha} \mid a \in \{0,1,2,3,4,5,6\}\} \tag{8}$$

The reward function has been modeled in such a way as to punish the agent for excessive fertilizer application amounts, while rewarding it for a large grain yield at the end of the season. It is defined to be the difference between the weight of the storage organ (w_{SO}), achieved with the agents fertilizer application strategy, and the weight of the storage organ (w^*_{SO}) achieved if no fertilizer were to be applied. From this, a negative coefficient β, multiplied by the amount of fertilizer the agent applies is subtracted. The reward function is

$$r_t = w_{SO,t} - w_{SO,t-1} - (w^*_{SO,t} - w^*_{SO,t-1}) - \beta w_{fert,t} \tag{9}$$

Here, $\beta = 10$ is set to place a strong restriction on the amount of fertilizer an agent would like to apply, thereby ensuring the learning of strategies with minimal application.

This constitutes a complete description of the environment details. No modifications to this environment created by Overweg, Berghuijs, and Athanasiadis have been made. Their implementation does however use an unpublished version of the PCSE.

Agents

The baseline agents are implemented for comparison against the reinforcement learning agents. The standard practice agent applies three doses of $60kg/ha$ nitrogen fertilizer. These are applied on the first and third week of March, with the final dose being applied on the first week of May. The reactive baseline agent applies $120kg/ha$ every time the soil nitrogen content depletes below $5kg/ha$.

Comparing to these baselines are the DQN agent, the Double DQN agent, the Dueling DDQN agent and the PPO agent. The PPO agent is implemented using the Stable

Baseline3[3] package, for which the policy and value function networks are constructed using a multi-layered perceptron (MLP) of two fully connected layers. The activation functions following the output of each layer are tanh non-linearities. The policy network is mapped to the dimension of the action space, after which a softmax function is used to assign the probabilities of selecting each available action.

Similarly, the Q-network and target network for the DQN, Double DQN, and Dueling DDQN use a basic MLP with two fully connected layers. Following the output of these layers, these agents use a rectified non-linearity.

This simple approach to the neural network design is taken to maintain consistency in order to investigate the performance of the reinforcement learning agents themselves within the fertilizer application domain. The Dueling DDQN includes two streams for the value and advantage function, as outputs for the network. This does not contradict the earlier statement, as the underlying deep model is still an MLP. These streams are not combined via an aggregating layer in the network to produce the action-value estimate Q, as is done by Wang et al., as our agent rather selects its actions according to the advantage estimate, since the value estimate is simply a scalar valued output and has no relative effect on the advantage of each action.

Many variational improvements exist in terms of more advanced underlying network architectures. These can be easily implemented to further improve the performance following this benchmark analysis.

Training Details

The reinforcement learning agents were all trained to convergence on the CropGym environment using the calibrated winter wheat crop model and 25 years worth of weather data from the Netherlands during the period of 1983 to 2016, as can be seen in Figure 1. This environment was normalized with the VecNormalize wrapper to further assist with stable learning. For this, the observations were normalized with an observation clipping of 10, while the rewards remained unchanged.

The effects of nitrogen fertilizer are often not experienced by the crop until a couple of weeks after its application (Cooper et al. 1987). This coupled with the fact that the largest growth takes place towards the end of the crop growth season, is why the discount factor for all future rewards provided by the environment, γ, is set to 1 for all agents.

For all of the agents, the dimensions of the hidden layers and tuneable hyper-parameters are found using the highly scalable Ray[Tune] optimization framework (Liaw et al. 2018) and can be viewed in the Appendix. For this, the Tree-Parzen Estimator hyper-parameter optimization algorithm was used (Bergstra et al. 2011).

Evaluation Procedure

All trained agents were evaluated on a test-bed containing the 1984, 1994, 2004, 2014, 2019 and 2020 crop growth seasons, which were naturally excluded from the training set. For each of these test-bed years, weather data was taken from $52°N, 52.5°E$. This range helps provide a full analysis of the agents ability to generalize across different decades with differering conditions. The performance of each agent was evaluated in terms of the total reward obtained, yield produced, crop nitrogen uptake, and total fertilizer application amounts. In order to further test each agents ability to generalize, we use agents trained on five random seeds, choosing the best three of each, and evaluate their performance on a fourth random seed for the mentioned evaluation metrics. This is done to maintain generalizability, while ensuring that one destructive policy of one of the seeds does not overshadow the performance of the agents.

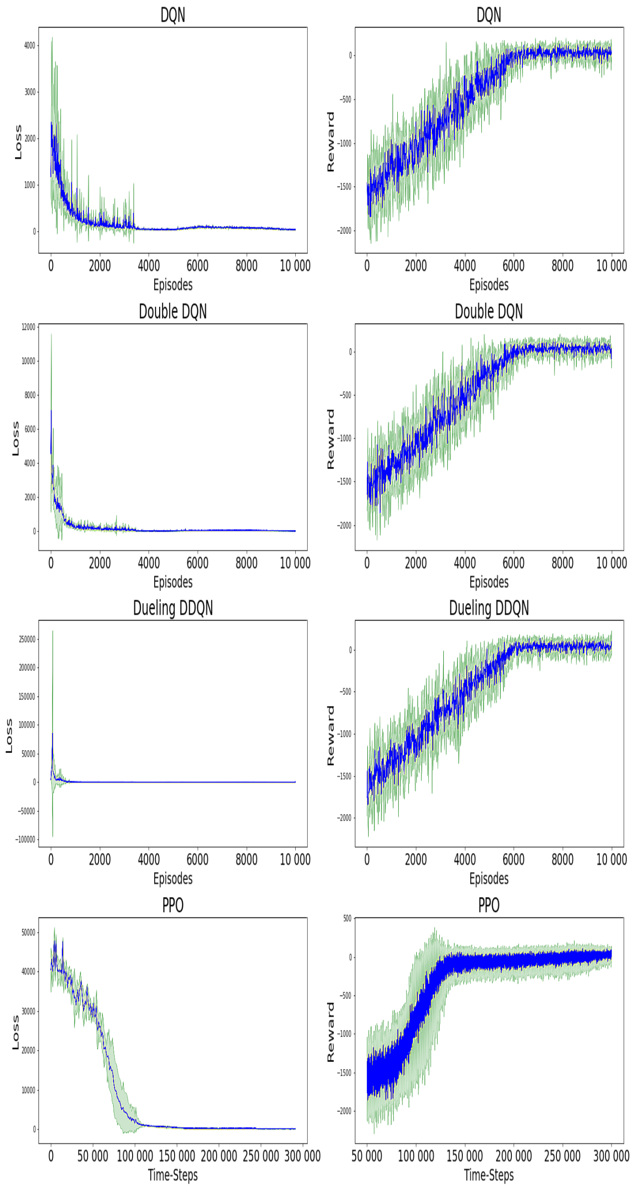

Figure 1: Loss and Reward Curves for All Implemented Agents Averaged over 5 Random Seeds.

[3]https://github.com/DLR-RM/stable-baselines3

Results

This section presents the results for the evaluation metrics discussed in the previous section for the test-bed years. In doing so, and by further discussing the findings, a full analysis of the ability of some of the best performing model-free reinforcement learning agents applied to the fertilizer crop management domain is provided.

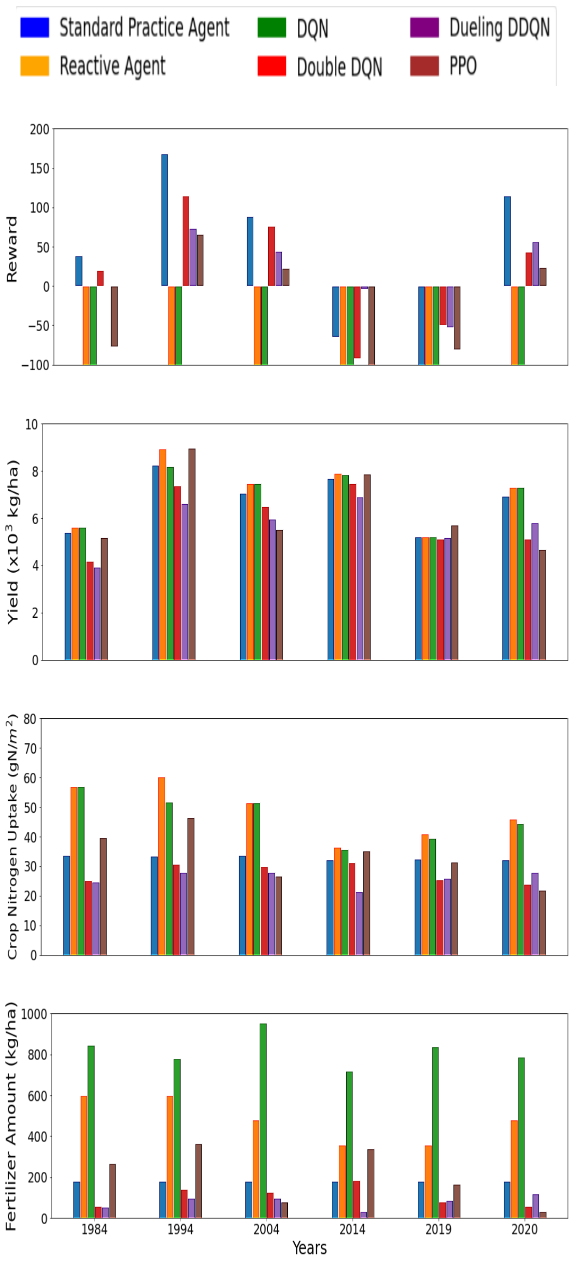

Figure 2: The Performance of each Agent in terms of Reward Obtained, Yield Produced, Crop Nitrogen Uptake, and Fertilizer Application Amount. Averaged over 3 Seeds

Year	TNSOIL	NUPTT	WSO
1984	1.95	21.1	3.44
1994	1.46	20.9	4.99
2004	1.45	21.1	4.64
2014	1.10	19.6	6.62
2019	1.30	19.8	4.87
2020	1.12	19.7	4.12

Table 1: Crop Growth Variables without any Fertilizer Application.

As previously discussed in the Introduction, the increasingly excessive use of nitrogen fertilizers in order to maintain yield is causing harm to all facets of the environment in terms of soil, water and air degradation(Sainju, Ghimire, and Pradhan 2019). There is an apparent need for finding strategies that minimize fertilizer application amounts, while still providing a sufficient yield. The reward function has been modelled in such a way as to enforce this, however the rewards obtained cannot be viewed solely as the agents ability to learn optimal strategies. Furthermore, the reward obtained, yield produced, crop nitrogen uptake, and fertilizer application amounts should be analyzed individually per year, but the evaluation of the performance of each agent should take all test-bed years into account, with the main focus on the latter three evaluation metrics. We can see this is necessary by analysing Table 1, which represents some of the important crop growth variables when no fertilizer is applied throughout the crop growth season, and comparing it to the reward plot in Figure 2. Despite having some of the smallest values for the soil nitrogen content (TNSOIL) and the nitrogen uptake by the crop (NUPTT) for the years 2014 and 2019, the yield produced (WSO) in these years are the largest and third largest from all the test-bed years. Naturally nitrogen is not the only factor inducing crop growth, and many other factors such as irrigation, rainfall, and other fertilizers present in the soil are potentially playing a role in this. This growth does however have a relative effect on the reward for those years, due to the modeling of the reward function.

From Figure 2 it can be seen that there is a strong correlation between the amount of fertilizer applied, and the subsequent uptake of nitrogen by the crop. Despite this, there appears to still be a limit as to the amount of yield produced, evident in the year 2019. This is further evidenced when analyzing the DQN agent and the reactive agent. They apply extreme amounts of fertilizer throughout all test-bed years, and subsequently they have the largest uptake of nitrogen by the crop, yet only achieve marginally better yields in the years 1984, 2004, 2014 and 2020. Due to these factors, their reward obtained is extremely low, and their application strategies are incompatible with the aims of this research, further showing the need for intelligent application strategies.

Comparing the PPO agent to the DQN and reactive agents, we can see that it successfully learns to apply less nitrogen fertilizer in the years 1984, 1994, 2014 and 2019, yet still achieves comparable yields, even surpassing the yield produced in 2019. The PPO agent can also be seen to apply the least amount of fertilizer in 2004 and 2020, and subsequently achieves the least yield in these years. When comparing it to the current baseline benchmark in farming, the standard practice agent, it can be seen that the PPO agent applies a fair amount more nitrogen in 1984, 1994 and 2004 with only slightly better yields. The PPO agent is therefore deemed to learn better policies than the DQN agent and reactive agent, however does not optimally generalize across all of the test-bed years for favourable results against the standard practice agent.

The standard practice agent applies the same fixed amount of nitrogen fertilizer for every test-bed year, thus allowing it to receive the best reward in four of the six years as the remainder of the reward is largely due to the growth incurred after this application. Despite this being a good strategy in general, for the years 2014 and 2019 it appears that little nitrogen fertilizer is needed for maximum growth. This shows the inability of this strategy to adapt to changing climate conditions. For this the Double DQN agent and the Dueling DDQN agent show favourable results.

The Double DQN agent learns an application strategy in which it applies less fertilizer than the standard practice agent in every single test-bed season. It subsequently achieves less yield in every year, however only marginally in the years 1994, 2004, 2014, and 2019, with a fair amount less fertilizer applied in 2004 and 2019. This shows the Double DQNs ability to substantially reduce fertilizer application amounts, while still achieving sufficient yield for majority of the years. When comparing the Double DQN and DQN agents, it can evidently be seen that the overestimated action-values result in sub-optimal policies when using the DQN algorithm.

The Dueling DDQN agent, although taking a cautious approach to fertilizer application, shows the best generalized application strategy across the decades represented by the test-bed years. This can be particularly seen in the 2014 and 2020 years. When looking at at the reward obtained, yield produced, and fertilizer application amount for 2014, and comparing to all other agents, it can be seen that no other agent was able to learn that this year required very little fertilizer to still achieve a sufficient yield. It applies far less fertilizer, achieves comparable yield, and subsequently has the best reward, while all other agents achieve a very negative reward due to their fertilizer application amounts. This is incredible when analyzing the 2020 crop growth season as well. For 2020, contrary to the other reinforcement learning agents, the Dueling DDQN agent has learnt that more nitrogen fertilizer is required to achieve a sufficient yield. This shows the agents ability to generalize learning across its actions for different decades, with vastly varying conditions.

The Dueling DDQN agent is concluded to learn the best generalizable fertilizer application strategies. It takes a far more cautious approach than all its competitors, however it is able to learn favourable strategies for the extreme cases of 2014 and 2020.

Related Work

Having presented the findings of our research, we compare in this section to previously researched work.

Within the domain of crop management using reinforcement learning, recent advances have focused on irrigation and fertilizer management, with more focus on irrigation management.

Within the fertilizer management domain, Garcia provides some of the earliest work, building a reinforcement learning model to achieve a satisfying crop yield, while maintaining below the prescribed limit of nitrogen found in drinking water. However, this implementation only used Q-learning to predict the three best fertilizer application dates, and is very limited in its approach and is now outdated.

After this there was limited interest into the use of reinforcement learning for fertilizer crop management. However, the combination of deep learning with reinforcement learning spurred increased interest into its applications. This allowed Overweg, Berghuijs, and Athanasiadis to provide the CropGym framework, with a focus on nitrogen fertilizer management, for the exploration of the application of reinforcement learning to sustainable agriculture. They however, only implement a PPO agent without any hyper-parameter tuning. Our research builds on their initial analysis by providing a more encompassing variety of model-free reinforcement learning algorithms applied to this domain.

Prior to our research, Overweg, Berghuijs, and Athanasiadis and Chen et al. have been the most recent successful implementations using reinforcement learning for fertilizer and irrigation management respectively. To our knowledge, there are no publications that have as of yet implemented a combination of these crop development factors.

Conclusion

The aim of this research was to provide a comprehensive analysis and an initial benchmark for the ability of some of the best performing model-free reinforcement learning algorithms to learn novel application strategies that reduce the environmental impacts of nitrogen, while sustaining a sufficient yield. This investigation has shown that agents can learn application strategies comparable to standard farming best practice approaches. The combination of the dueling network architecture with the Double DQN algorithm was shown to find the most favourable strategies, sustaining a sufficient yield across the varying soil and weather conditions experienced throughout test-bed years. The Dueling DDQN agent was successfully able to differentiate between seasons that require more fertilizer application amounts to maintain yield, and seasons that require very little. This approach was limited to only learning nitrogen fertilizer application strategies, and more research is required in order to be able to implement this in a physical system. Future work may contain the use of more detailed process-based models, further combining other crop growth factors such as irrigation, and phosphorous and potassium fertilizers, for a more realistic management of the crop.

References

Bergstra, J.; Bardenet, R.; Bengio, Y.; and Kégl, B. 2011. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems* 24.

Binas, J.; Luginbuehl, L.; and Bengio, Y. 2019. Reinforcement learning for sustainable agriculture. In *ICML 2019 Workshop Climate Change: How Can AI Help*.

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.

Cooper, P.; Gregory, P.; Keatinge, J.; and Brown, S. 1987. Effects of fertilizer, variety and location on barley production under rainfed conditions in northern syria 2. soil water dynamics and crop water use. *Field Crops Research* 16(1):67–84.

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.

Liakos, K. G.; Busato, P.; Moshou, D.; Pearson, S.; and Bochtis, D. 2018. Machine learning in agriculture: A review. *Sensors* 18(8):2674.

Liaw, R.; Liang, E.; Nishihara, R.; Moritz, P.; Gonzalez, J. E.; and Stoica, I. 2018. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature* 518(7540):529–533.

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

Sainju, U. M.; Ghimire, R.; and Pradhan, G. P. 2019. Nitrogen fertilization i: Impact on crop, soil, and environment. In *Nitrogen Fixation*. IntechOpen.

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

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

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; and Freitas, N. 2016. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, 1995–2003. PMLR.

Zhou, N. 2020. Intelligent control of agricultural irrigation based on reinforcement learning. In *Journal of Physics: Conference Series*, volume 1601, 052031. IOP Publishing.

Appendix

Crop Growth Variables

Symbol	Meaning	Unit
DVS	development stage	-
LAI	Leave area index	-
TGROWTH	Total biomass growth	g/m^2
NUPTT	Total N uptake	$g\frac{N}{m^2}$
TRAN	Crop transpiration rate	$m^3\frac{H_2O}{m^2}$
TIRRIG	Water applied by irrigation	$m^3\frac{H_2O}{m^2}$
TNSOIL	Soil nitrogen available	g/m^2
TRAIN	Total rainfall	$m^3\frac{H_2O}{m^2}$
TRANRF	Transpiration reduction factor	$\frac{mm}{day}$
TRUNOF	Soil runoff	$m^3\frac{H_2O}{m^2}$
TAGBM	Total above ground dry weight	g/m^2
TTRAN	Water removed by transpiration	$m^3\frac{H_2O}{m^2}$
WC	Soil moisture content	$m^3\frac{H_2O}{m^2}$
WLVD	Dry weight of dead leaves	g/m^2
WRT	Dry weight of roots	g/m^2
WLVG	Weight green leaves	g/m^2
WSO	Dry weight of storage organ	g/m^2
WST	Dry weight of stems	g/m^2

Weather Variables

Symbol	Meaning	Unit
TMAX	Daily maximum temperature	$^\circ C$
TMIN	Daily minimum temperature	$^\circ C$
VAP	Mean daily vapour temperature	hPA
RAIN	Precipitation	$\frac{cm}{day}$
IRRAD	Daily global radiation	$\frac{J}{m^2}/day$

Hyper-parameters

DQN

Name	Value
Optimizer	Adam
Learning Rate	0.000279324
Hidden Layer 1 Dimension	768
Hidden Layer 2 Dimension	560
Epsilon Decrement Amount	5e-06
Epsilon Start	1.0
Epsilon End	0.03
Target Update Frequency (episodes)	40
Replay Buffer Capacity	100 000
Batch Size	224
γ	1

Double DQN

Name	Value
Optimizer	Adam
Learning Rate	0.00043530
Hidden Layer 1 Dimension	848
Hidden Layer 2 Dimension	672
Epsilon Decrement Amount	5e-06
Epsilon Start	1.0
Epsilon End	0.03
Target Update Frequency (episodes)	30
Replay Buffer Capacity	100 000
Batch Size	544
γ	1

Dueling DDQN

Name	Value
Optimizer	Adam
Learning Rate	0.00097369
Hidden Layer 1 Dimension	752
Hidden Layer 2 Dimension	672
Epsilon Decrement Amount	5e-06
Epsilon Start	1.0
Epsilon End	0.03
Target Update Frequency (episodes)	50
Replay Buffer Capacity	100 000
Batch Size	384
γ	1

PPO

Name	Value
Learning Rate	0.00049782
Hidden Layer 1 Dimension	420
Hidden Layer 2 Dimension	288
N Steps	1024
Entropy Coefficient	5.2e-05
Clip Range	0.3
N Epochs	5
GAE Lambda	0.97852
Max Grad Norm	0.3
VF Coefficient	0.0888
Batch Size	512