Electronic Communications of the EASST
Volume 83 Year 2025

deRSE24 - Selected Contributions of the 4th Conference for Research Software Engineering in Germany

Edited by: Jan Bernoth, Florian Goth, Anna-Lena Lamprecht and Jan Linxweiler

Deep Reinforcement Learning in Agent-Based Model AgriPoliS to Simulate Strategic Land Market Interactions

Changxing Dong, Ruth Njiru, Franziska Appel

DOI: 10.14279/eceasst.v83.2621

Electronic Communications of the EASST (https://eceasst.org).
Published by **Berlin Universities Publishing**
(https://www.berlin-universities-publishing.de/)

Deep Reinforcement Learning in Agent-Based Model AgriPoliS to Simulate Strategic Land Market Interactions

Changxing Dong[1], Ruth Dionisia Gicuku Njiru[2] and Franziska Appel[3]

[1]dong@iamo.de, [2]njiru@iamo.de and [3]appel@iamo.de
Institut für Agarentwicklung in
Transformationsökonomien, Germany

Abstract: AgriPoliS (Agricultural Policy Simulator) is an Agent-Based Model for simulating the dynamic evolution of agricultural regions with a particular emphasis on the structural transformations influenced by economical, ecological, and societal factors. This study introduces a Reinforcement Learning (RL) framework to empower agents within AgriPoliS with strategic decision-making capabilities, specifically in the context of land market bidding. Traditional agents within AgriPoliS make decisions through Mixed Integer Programming (MIP) optimizations, inherently limited by myopic considerations of the current year's conditions. In this framework one agent within AgriPoliS is defined as the RL agent, with AgriPoliS itself serving as the learning environment. Training results with policy gradient algorithms demonstrate that the RL-enhanced agent consistently outperforms its non-learning counterparts, exhibiting strategically stable bidding behavior. To assess the robustness of the algorithm, extensive training runs are conducted with varying hyperparameters, including policy network initialization, learning rate, and exploration noise level. Our findings underscore the efficacy of RL in enhancing strategic decision-making in agricultural land markets, showcasing superior performance compared to fixed bidding strategies.

Keywords: Deep Reinforcement Learning, Policy Gradient Algorithm, Agent-Based Model, AgriPoliS, Message Queue

1 Introduction

With agricultural land being one of the most crucial components of ensuring food availability, food access and food security, the importance of land markets cannot be overlooked. Land markets with special emphasis to land rental markets in the simplest terms allow the redistribution of land assets from owners to land users who put the land to use, allow for consolidation of land plots to larger plots which allows for achievement of economies of scale and thus higher income for the land users [DSW01]. Land rental markets have a diverse effect on the structural change within the agricultural sector by affecting farm entry, farm performance, farm growth and ultimately the farm exits [KSB08, HOB13]. Owing to the importance of land rental markets, different regulations have been enacted and/or re-enacted to guide land market transactions with multiple objectives such as prevention of transfer of agricultural land to non-agricultural investors, prevention of division of agricultural land into uneconomical sizes, capping land prices

and in more recent years the agenda has shifted to climate change protection measures such as bio-energy crops, reduction of carbon gas emissions [BGM+21, LB18, OH20].

Thus farmers' participation in the agricultural land rental markets whether as land owners or land bidders is greatly tied to their individual objectives and motivation. Farmers' decisions on the land rental market are influenced by their expectation of profits to be derived from the plot of land, farms' current capacities (livestock, machinery, stables etc.), transactional costs, and the farmers' risk taking or risk aversion behaviour. The decision also considers land specific characteristics such as soil quality, climatic productivity and the spatial location of the plot [BDM22, Rah10]. There is therefore a crucial need for proper modelling of the complexity of individual behaviour and individual interactions in land rental markets.

Agent-Based Models (ABMs) have gained a lot of traction in recent years from their "bottom-up" modelling approach to capture emergent phenomena in complex systems in a way that traditional models could not. The individual entities in an ABM, referred to as "agents" are autonomous, active and heterogeneous in their decision-making process. The agents are guided by rules and heuristics in their quest to achieve goals and objectives [An12, Bon02, CH12, Eps99, RG11]. ABMs have proven quite useful in their ability to capture individual behaviour in complex systems. ABMs that embed agricultural land rental markets include AgriPoliS (Agricultural Policy Simulator) for modelling the impact of various land regulations on the structure of agriculture regions over time while accounting for interactions in the land market [HKB06, HAB19], Pampas Agent-Based Model (PM) with an embedded Land Rental Market (LARMA) model explores the dynamics of land use patterns in the Argentinean agricultural systems relying on interactions between owners and tenants in the land rental market [BPR+11] and ABMSIM simulates the structural change in dairy farms with spatial explicit land rental auction market, milk disposal markets [BW14].

Despite the usefulness of the ABMs, they mostly rely on traditional approaches where the farm behaviour is motivated by myopic profit or utility maximizing behaviour. This ignores the complexity of individual farm behaviour and the need for long term decision-making. Capital-intensive investments and specific production decisions e.g. for perennial crops have implications spanning over several years and thus should always be factored in current decision-making. Decisions are also mostly based on hard encoded rules and heuristics that should be followed by the farms while ignoring the need for individual farm goals and plans [HKB06, HBB+18]. There is therefore a need to improve modelling capabilities towards more strategic decision-making. Complementing ABMs with Deep Reinforcement Learning (DRL) which combines Reinforcement Learning (RL) with Deep Neural Networks (DNNs) has shown promise for modelling flexible and adaptive agents. Those agents make their decisions not by given rules. Instead, they interact with their environment and learn strategic behaviours and thus they could be considered as strategic agents [ZVC23, OVG+20]. DRL agents, while interacting with other agents can figures out strategies that maximize their rewards while factoring in past experiences and anticipating future experiences.

In this paper, we explore the methodological approach of integrating DRL in an ABM, AgriPoliS, where a single agent enhanced with DRL capabilities in their decision-making to facilitate strategic behaviour and interactions in the land market. The paper is structured as follows: In section 2, an overview of AgriPoliS including agents' decision making and interaction is presented. In section 3, the integration of DRL and AgriPoliS and the experimental setup is explained.

In section 4, the training results are presented and followed by a discussion and conclusion in section 5.

2 AgriPoliS

AgriPoliS (Agricultural Policy Simulator) is an ABM for simulating the development of agricultural regions, focusing on the structural changes under economical, ecological and societal factors [Bal97, HKB06]. The farms within the region are portrayed as independent farm agents interacting with each other through various markets with particular emphasis on the land market. Every farm is assumed to maximize profit for corporate farms or household income for family farms through Mixed Integer Programming (MIP) linked to farm specific factor endowments (land, labour, capital, stables, machinery, etc.), production alternatives, investment options and external policy framework. After collecting data from the Farm Accountancy Data Network (FADN) [EEM20] and Farm Structure Survey (FSS) [EE19], the farms are initialized based on well calibrated empirical data on the aggregate regional capacities and indicators of individual farms in the region.

2.1 Agents' Decision Making

In AgriPoliS, farm agents have the freedom to decide between different production activities, different investment options (e.g. new stables, machinery), whether to hire more labour or in the case of family farm whether to look for off-farm employment. The farm agents also have the freedom to decide on whether to borrow money or save money. Decisions in the current year are based on their expectation of total income that might be accrued in the next year.

A typical simulation run begins by initializing the farms and subsequent simulations spanning over a defined number of iterations where one iteration corresponds to one production year. For each iteration, the farm agent presents their bid to the land market where farm plots are allocated through competitive bidding where the highest bid wins the plot. This is followed by investment decisions and after that the farms implement their production decisions. At the end of every iteration, AgriPoliS calculates the individual agent's indicators (updates of farm endowments, production and financial status etc.) as well as aggregate indicators for the farm agents at the regional level. Based on that, the farms calculate their expected income for the upcoming year and decide whether continue or to exit farming. The process continues until the end of the simulation run.

2.2 Interaction among Agents

Agents interact through the land market which is vital in AgriPoliS as farm growth is strongly dependent on land. Farms in AgriPoliS grow predominately through renting additional land in the land rental market in an iterative auction manner. The auction market is held at the start of every iteration. Land becomes available for renting when existing rental contract are terminated or when farms decide to exit farming. Each farm determines a valuable plot of land and calculates a bid to present to the market. The bid is a function of the distance between the farmstead and the plot of land measured in terms of the transport costs, the number of adjacent plots and lastly

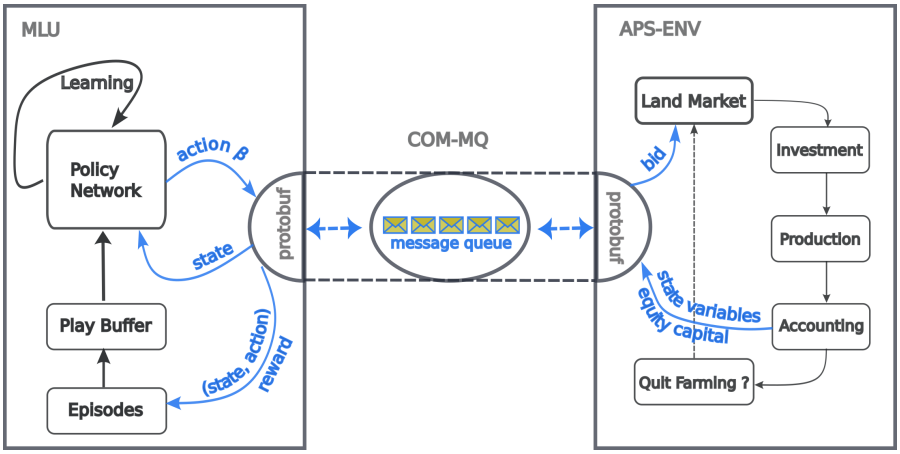


Figure 1: Reinforcement Learning framework with AgriPoliS

the agent's estimated benefit of renting an additional plot. This is considering that plots close to the farmstead and adjacent to other plots belonging to the farm are deemed more valuable. The bid is calculated according to equation 1.

$$\text{bid} = (q - \text{TC}) \cdot \beta \tag{1}$$

In this equation, q is the shadow price of the additional land (expected additional income to be generated from this plot), and TC is the transport cost. The bidding parameter β reflects the share of shadow price of additional land plot that is transferred to the land owner [Hap04]. The remaining share $1-\beta$ of the shadow price is kept by the farm. On the other hand, research of land price development [Gra18] found evidence for the need of price discrimination in regions where spatial location plays a big role. This leads to a decrease in rental prices and therefore to a reduction of the share of shadow prices that is transferred to the landowner. Based on these considerations, β is set to 0.5 in AgriPoliS. The farm with the highest bid receives the plot [KSB08]. Like with every other farm decision (investment, production, exit), the decision on how much to bid is based on the current conditions and expectations only for the next year. Farm agents' decision-making is therefore highly myopic and does not consider any strategic aspects of decision making, e.g. how their decisions in one iteration would affect future actions in subsequent iterations.

3 DRL Meets AgriPoliS

In this section, a framework to integrate DRL and AgriPoliS is presented (Figure 1). The framework is composed of three components. In the first component, the AgriPoliS is extended to also function as the environment for the Reinforcement Learning (RL) [SB18], henceforth referred to as APS-ENV. The second component is the Machine Learning Unit (MLU) which is responsible for the training of the RL agent. The third component COM-MQ is necessary to provide reliable communication between APS-ENV and MLU. COM-MQ is implemented with the message queue system ZeroMQ to transport the messages from/to MLU and APS-ENV. Some of the

messages are (de)serialized with protobuf. With COM-MQ in between, MLU and APS-ENV are developed independently. Therefore MLU and APS-ENV can be implemented in different programming languages, i.e. in Python and in C++ respectively. Principally they can be run even in different machines with known IP addresses.

3.1 The Reinforcement Learning Problem

As the first step, only one agent in AgriPoliS is enhanced with RL while the other agents adopt the standard AgriPoliS behaviour. The enhanced agent is referred here after as the DRL-agent. As we are interested in the effects of the strategic bidding behaviour of the agent, β in equation 1 is designed as the action in a continuous action space. The state space consists of the properties of the DRL-agent and the region under investigation, which include liquidity, current farm endowment (stables and machines), the distribution of remaining contract duration for rented land, prior rental rates, spatial distribution of free land plots in the region and the distribution of competing agents in the neighborhood, as shown in Table 1. The state at the end of simulation run is the terminal state. The equity capital of the agent is considered as the representative property for the agent's overall development. The equity capital of the DRL-agent is collected from the APS-ENV for every simulation iteration. As the objective is to achieve strategic bidding behaviour and at the same time have a sustainable farm growth for the DRL-agent (avoiding the risk of illiquidity), the cumulative equity capital at the end of a simulation run is considered the reward, which is defined as in equation 2 for the transition $(s,a) \rightarrow s'$, where r_i is the equity capital in the new state.

$$R(s,a,s') = \begin{cases} \sum_i r_i & \text{if } s' \text{ is terminal} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Using the state variables, the agent chooses the action (β) and accordingly calculates the bid (cf. equation 1) for the land market. Based on the success and/or failure of the bid the land endowment is updated before the agent proceeds to make investment and production decisions and obtains the results for this iteration. This continues until the end of the simulation run when the cumulative equity capital is calculated.

3.2 The Algorithm

In Figure 1, the detailed processes in the three components in the framework are illustrated. During the training, APS-ENV and MLU run independently, while interacting through sending and receiving messages at certain points in time. As the action space is continuous, the learning is approached through a deterministic policy gradient algorithm [LHP+15]. In this algorithm only one policy network is used, in which the state (Table 1) is the input and the action (β) is the output. Before the training cycles begin, the APS-ENV, the policy network and related data structures, and the COM-MQ are initialized. Pseudocode 1 illustrates the initialization process in detail while Table 2 gives a summary of the training hyperparameters. The learning is iterative with a fixed number of epochs. In each epoch, it goes through three steps: training data collection, network update and policy testing. To collect training data, the policy network

Table 1: State variables for Reinforcement Learning

Name	**Type**	**Level**	**Notes**
Terminating plots	List of integers	Farm	Distribution of plot numbers over rest contract length (1 to 5 years)
Liquidity	Real	Farm	Ability of the farm to meet short term liabilities
Farm age	Integer	Farm	Age of the farm
Investments	List of real	Farm	Remaining life of the investments
Previous rental rate	Real	Farm	The latest amount of rent paid by the agents. This is differentiated between arable land and grassland
Management coefficient	Real	Farm	This is reflecting the farms' managerial ability by manipulating the variable costs.
Free plots	Integer	Region	Number of available (remaining) free plots in the region. This is differentiated between arable land and grassland
Number of farms	Integer	Region	Number of competing farms

Pseudocode 1 Initialization function in MLU

```
1: function INITIALIZATION(N, S, Y, ENV)          ▷ ENV is an instance of APS-ENV
2:     P.init()          ▷ initialization of policy network P with arbitrary weights
3:     PlayBuffer = []                              ▷ the play buffer for training data
4:     noise.reset()                                          ▷ reset noise generator
5:     num_epochs = N                                                ▷ number of epochs
6:     num_simulations = S                          ▷ number of simulations per epochs
7:     num_years = Y          ▷ number of years for every simulation with AgriPoliS
8:     best_reward = 0                                          ▷ best episode reward
9:     MQ.init(ENV)                                  ▷ MQ is an instance of COM-MQ
10: end function
```

Table 2: Training parameters for the Reinforcement Learning

Parameter	Value	Notes
N	1000	Number of epochs
S	30	Number of simulations (episodes) per epoch
Y	10	Number of years per simulation (episode)
N_hidden	2	Number of hidden layers of policy network
S_hidden	16	Size of hidden layers of policy network
LR	10^{-4} to $5 \cdot 10^{-3}$	Learning rate for neural network
noise	0.4 to 0.01	Noise level for exploration

is used to obtain actions enhanced with noise to balance between exploration and exploitation. Note that there are also "noises" in agriculture due to random events like weather conditions and pests. They are partly implemented within AgriPoliS with different random number generators. The noise in this work is a hyperparameter in RL. It is responsible for the exploration of the policy space and is not used to simulate the randomness in agriculuture.

Since the DRL agent only get non-zero reward at terminal states, the simulations with AgriPoliS will not be stopped or interrupted and therefore they are rollout episodes with the same length, i.e. the number of simulation years. The reward is given at the end of a simulation/episode as the cumulative sum of the equity capital. The episodes are obtained by taking actions from the policy network with noises. The noise is responsible for explorating the policy space. Such episodes have different rewards. As the goal is to maximize the rewards, The state-action pairs (s,a) in the episodes with largest rewards are put in a memory called play buffer which is updated only if new episodes have larger rewards (line 7-8 in Pseudocode 3). The policy network is trained with data taken from the play buffer.

The update of the policy network exploits supervised learning algorithms where states are the inputs and actions from the play buffer are target values. Concretely, we use Mean Squared Error (MSE) as the loss function and Adaptive Moment Estimation (ADAM) as the optimizer (cf. lines 13 in Pseudocode 3). After updating the policy network, the last step of an epoch is to test the policy learned. This is a standard simulation run with AgriPoliS, obtaining the actions from the updated policy network, but without additional noises. The learned behaviour can be seen in the rewards of the testing runs of all the epochs. Interactions between MLU and APS-ENV, as the lines 6, 11, and 12 in Pseudocode 2 show, are through the COM-MQ component. Basically, the algorithm combines Monte Carlo Tree Search (MCTS) with supervised learning like in the algorithm for AlphaGo [SHM+16]. Although here the search tree is not explicit but implicitly saved in the play buffer.

3.3 Experimental Setup

As this work aims to investigate the feasibility of our approach, the investigated region only has 7 agents. They are typical farms from the region Altmark in Saxony-Anhalt, Germany. The experiments were carried out on a workstation with a Nvidia GPU (Quadro RTX 6000).

To investigate the learning behaviour, especially the robustness of the algorithm, the training

Pseudocode 2 Simulation function in MLU

```
function SIMULATION(test = False)                    ▷ Simulation with AgriPoliS
    y = 0                                            ▷ current year
    episode = []
    R = 0
    while y < num_years do
        state = MQ.get_state()
        action = P(state,w)                          ▷ action from policy network
        if not test  then
            action += noise()
        end if
        MQsend_action(action)
        equity = MQ.get_equity()
        R = R + equity
        if not test then
            episode.append((state,action))
        end if
        y += 1
    end while
    return R
end function
```

Pseudocode 3 Training process in MLU

```
initialization(N, S, Y, ENV)
e = 0                                                ▷ current epoch
while e < num_epochs do
    s = 0                                            ▷ current simulation
    while s < num_simulations do
        R = simulation()
        if R > best_reward then
            PlayBuffer.update(episode)
        end if
        best_reward = R
        s += 1
    end while
    P.update()                                       ▷ Learning with MSE loss function and ADAM
                                                     ▷ optimizer using training data from PlayBuffer
    R = simulation(test = True)                      ▷ test the learned policy
    e += 1
    output(e, R)                                     ▷ output the test run
end while
```

Table 3: Chosen hyperparameters for the training

Initialization	Learning Rate	Noise
1	$1 \cdot 10^{-4}$	0.4
2	$5 \cdot 10^{-4}$	0.3
3	$1 \cdot 10^{-3}$	0.2
4	$2 \cdot 10^{-3}$	0.1
5	$5 \cdot 10^{-3}$	0.05
		0.01

runs were carried out with systematical variations in hyperparameters. Note that the special environment APS-ENV cannot be reset during the simulation. Further, the sequential nature (land market interaction) of the AgriPoliS simulation constrains the training process, as it cannot be divided into parallel processes. This means that the training process is time-intensive. For the chosen number of epochs (1000) and number of simulations (30) per epoch, a training round takes about 14.5 hours. Therefore only three hyperparameters were chosen for the systematic investigation: learning rate, initialization of the policy network, and the noises added to action values as they greatly influence the exploration, efficiency, stability and convergence of learning in DRL. Table 3 shows the values of these parameters. The training is run by changing only one parameter while the other two parameters are kept constant, i.e. the default values in table 2.

4 Training Results

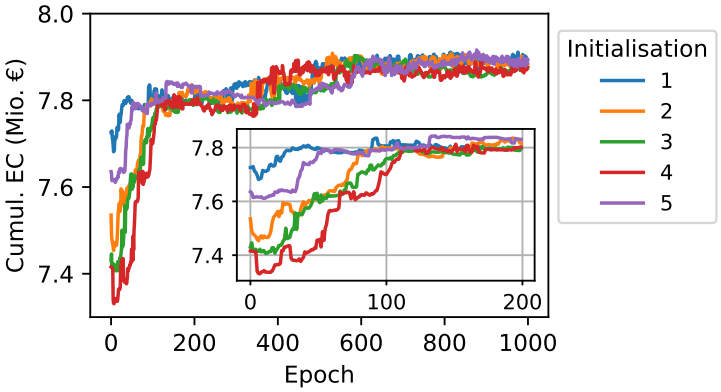
In this section, the performance of our DRL approach is demonstrated. We analyze training results with systematic variation of the hyperparameters according to Table 3. Further, we show and discuss the potential application of Deep Deterministic Policy Gradient (DDPG) algorithm.

4.1 Cumulative Rewards

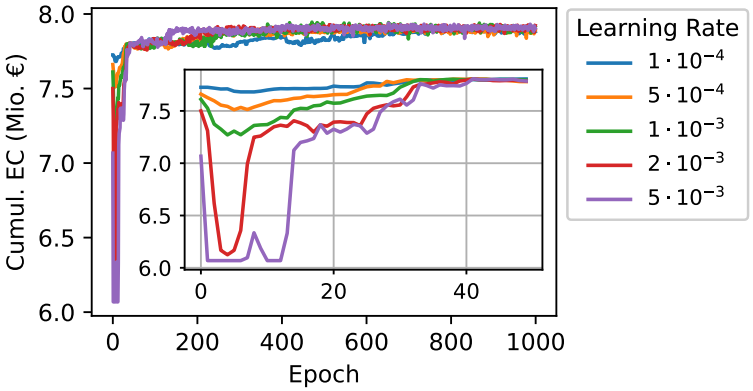
In the RL framework, although supervised learning was used, the values of the loss function can not be used as an indicator of the learning quality, since the optimal cumulative reward is unknown and the training data is dynamically updated. As cumulative reward is the object function of the optimization problem with the learning algorithm, its dependence on the number of epoch is adequate to evaluate the learning behaviour. Figure 2 shows results with variations in the hyperparameters. In general, the framework is effective and the DRL-agent can achieve a stable policy through learning. The impact of the different hyperparameters are detailed below:

Figure 2a shows the learning behaviour for the DRL agent with different initializations for the policy network. The weights of the network are initialized randomly with different seeds. The numbers 1 to 5 do not denote the values of the seeds but the sequence number of the training experiments. The inset shows the curves of the first 200 epochs. From the curve, its clear that the initialization has almost negligible effects on the learned policy, as the cumulative equity capital stabilized at the same level after about 550 epochs. The curves coincide after about 120 epochs.

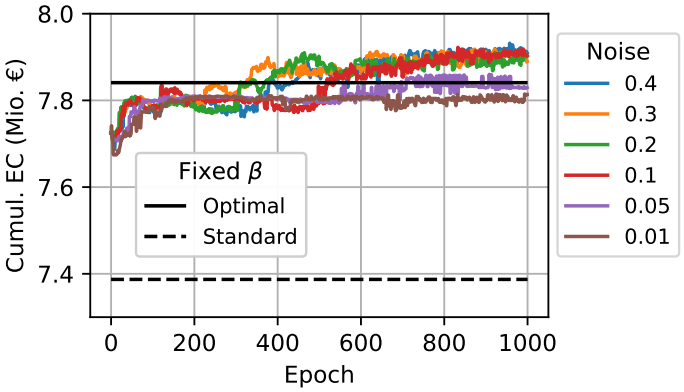
Figure 2b reflect the training curves with different learning rates (10^{-4}, $5 \cdot 10^{-4}$, 10^{-3}, $2 \cdot$



(a) Different Initializations



(b) Different Learning Rates



(c) Different Levels of Noise

Figure 2: Learning behaviour with different hyperparameters

10^{-3}, $5 \cdot 10^{-3}$). The curves coincide even earlier than with different initializations. As the inset indicates, after only 40 epochs, the agent has already learned a stable strategy. At the early learning phase, it is more stable if the learning rate is smaller. In terms of the cumulative equity capital, the values were much lower in the first epochs as compared to the other hyperparameters values (see Figure 2a and 2c). Note that the y-axes have different scales.

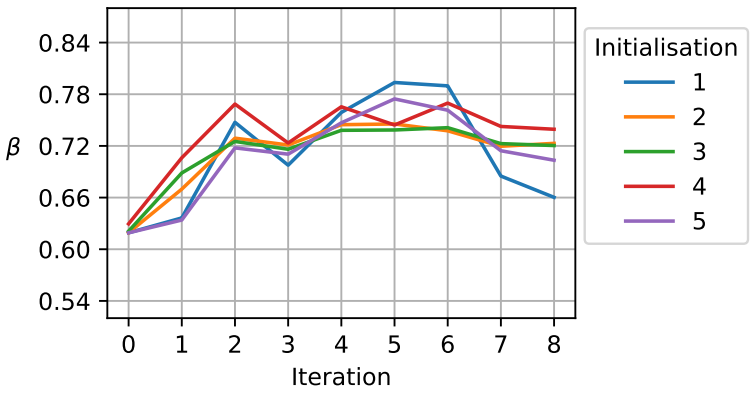
Figure 2c shows the influence of noise level on the learning behaviour. Noise is important for the agent as it helps them to discover new actions that lead to better rewards. The value of the noise level is the standard deviation of a normal distribution with the mean value 0. The noise is added to the action value while training. As the learning rate is kept small (10^{-4}), the cumulative equity capital consistently rises over time . Although the training curves have similar forms, the learned strategy stabilized at different levels. From the figure, it's apparent that learned strategies with noises larger than 0.1 are better than those with lower noises. As the differences of these better strategies are marginal after epoch 500, even higher noise level will not improve the learned strategy. In this figure, two additional lines are drawn for comparison. The dashed line at the bottom indicates the cumulative equity capital of the agent in standard AgriPoliS, while the solid line shows the cumulative equity capital for the agent with the optimal fixed action value (β). This is an optimized value for (β) that leads to maximum cumulative equity but remains constant over the entire simulation. After about 550 epochs, the solid line is lower than four of the learning curves, which means that the DRL enhanced agent can get better results than any fixed bidding strategy. The two curves below the solid line are those with the smallest noises (0.05 and 0.01) and reflect that the agents are stuck at suboptimal strategies. This demonstrates the importance of the exploration with noise.

4.2 Action Sequences from Learned Policies

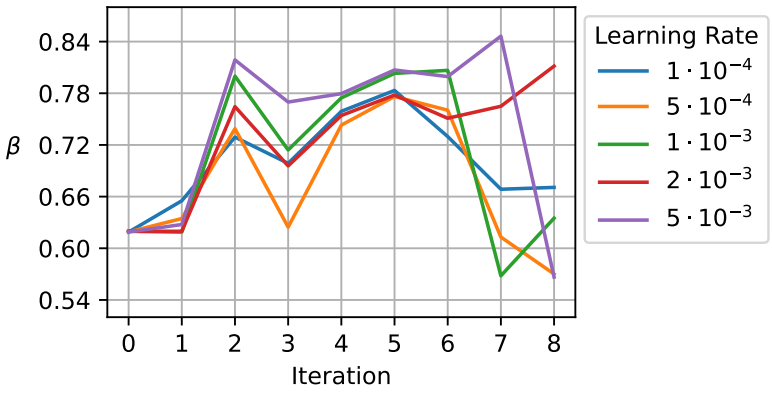
Figure 3 shows the sequences of actions (variation of β over time) for the learned policies corresponding to the variation of hyperparameters in Figure 2. The first observation is, the action sequences for all the strategies excluding the two with the smallest noise levels have similar curves. For example, they have two peaks at iteration 2 and 6. Second, as the Figure 3b indicates, the action sequences with different learning rates have larger variances, while resulting in comparable rewards (cf. Figure 2b). The variance is especially large at the end of the simulations. This indicates the importance of actions in earlier iterations while the impact of the actions in later iterations on the cumulative reward is relatively low. As the agent is reaching the terminal state, the action has no more long term influence. Third, the two action sequences with the lowest noise levels have very different characteristics than the other ones. As we know from Figure 2c, these two strategies lead even to a lower reward than the optimal fixed action strategy.

4.3 Quality of the Learned Policies

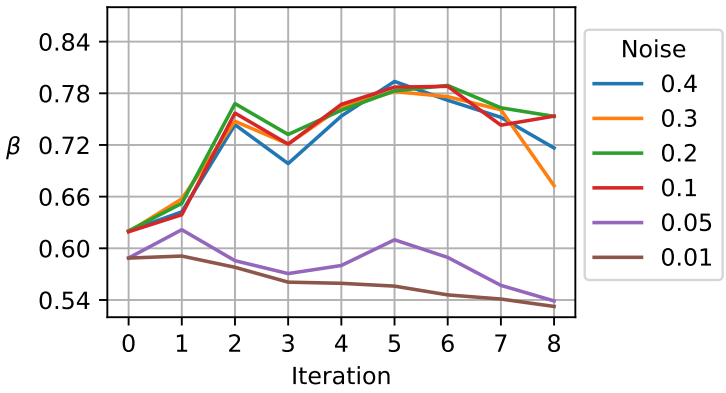
In Figure 2 the training curves stabilize at the latest after 600 epochs. To analyze the quality of the learned strategies, Figure 4 shows the average cumulative equity capital over the last 100 epochs with bars indicating the standard deviation. As the algorithm chooses the data from the play buffer randomly, the variance of the strategies is expected. The qualities with different initializations (Figure 4a) and different learning rates (Figure 4b) are good and comparable. From



(a) Different Initializations

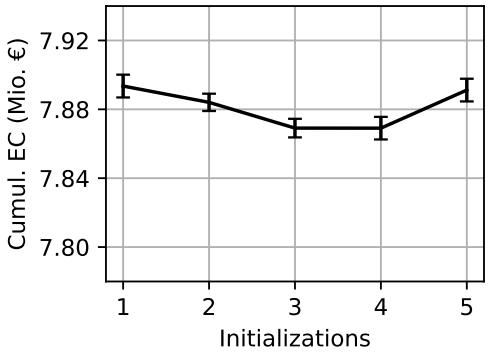


(b) Different Learning Rates

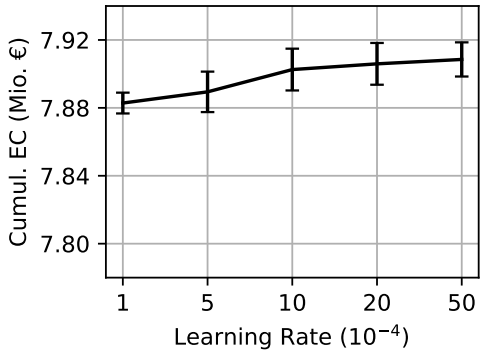


(c) Different Levels of Noise

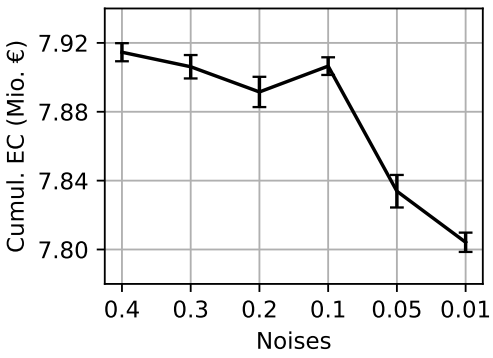
Figure 3: Action sequences of learned strategies with different hyperparameters. $0 \leq \beta \leq 1$ is the bidding parameter



(a) Different Initializations



(b) Different Learning Rates



(c) Different Levels of Noise

Figure 4: Average cumulative equity capital over the last 100 epochs depending on different hyperparameters

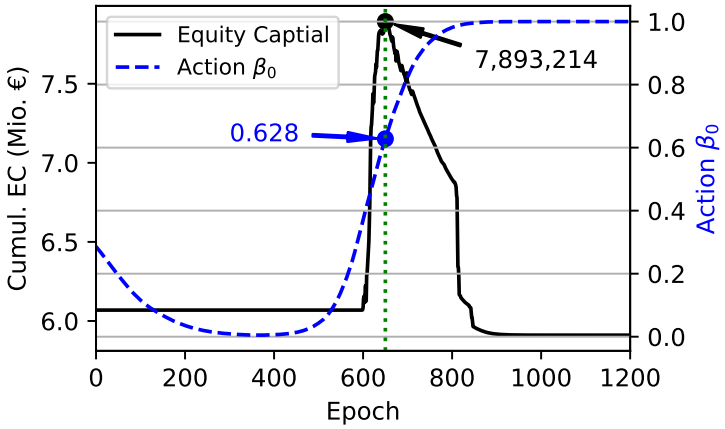


Figure 5: Training with DDPG algorithm

Figure 4b it seems the larger the learning rates the better the learned strategy. But this is only due to the slower learning at lower learning rates. This observation therefore indicates that for some hyperparameter settings the learned strategies can still be improved with more epochs. This is confirmed by training with 3000 epochs, although the improvement is marginal. In Figure 4c, the quality differences are obvious between the first four noises (0.4, 0.3, 0.2, 0.1) and the last two (0.05, 0.01). The differences among the strategies with higher noises are marginal.

4.4 Experiments with DDPG

For RL with continuous action space, the Deep Deterministic Policy Gradient (DDPG) [SLH+14] algorithm is successfully applied for several environments in the Gymnasium library [TTK+23]. It uses a value network to direct the learning process of the policy network. At first glance it seemed suitable for our problem, taking the equity capital of every iteration (r_i) as the reward of every state transition. But the DDPG algorithm converged to very poor results. The reason is, although the r_i contribute to the reward, it cannot be considered itself as the reward for every action. Nevertheless, the training with this algorithm has some interesting points. In Figure 5 we show the typical training curve along with the first action value (β_0) of the corresponding learned policy.

The training curve is characterized with 4 phases separated by the epochs 600, 650 and 900. In the first phase (from the beginning to epoch 600), although the cumulative equity capital keeps at a constant low value, the policy network is actually updating, as the corresponding action values indicate. The constancy of the cumulative equity capital with varying action values is due to the characteristics of the auction process, as only a successful bid can influence the reward. In the second phase (from epoch 600 to 650), the agent learns rapidly, obtaining the largest equity value of €7,893,214. The action value grows correspondingly from a very low value to 0.628. In the third phase (from epoch 650 to 900), the agent forgets progressively what it has learned and the equity value reaches the minimum. Interestingly, the action value grows further to the maximal value of 1.0. In the last phase (from epoch 900) the values do not change anymore. The

thinking-learning-forgetting-sleeping pattern of the training curve comes from the structure of the DDPG algorithm and the misinterpretation of the equity capital of every iteration as reward of the action as the DDPG uses gradient ascent, which depends directly on the output of the value network. By updating the value network, the target value $v + r_i$ is always larger than v, therefore larger action values are prioritized and the policy network will be updated to output larger action values. This is a positive feedback cycle, which causes the steady growth of the action value. But this steady growth is not related to any growth of the cumulative equity capital and therefore does not lead to successful learning in our case.

5 Discussion and Conclusion

This study pioneers the integration of Deep Reinforcement Learning (DRL) into the Agent-Based Model AgriPoliS, empowering agents with learning capabilities. That means that an agent in AgriPoliS is enhanced with learning ability. By establishing the framework using the three components Machine Learning Unit (MLU), communication unit COM-MQ and environment unit APS-ENV, ML algorithms are developed independently of AgriPoliS. Since the data is (de)serialized and communicated through a reliable communication component, we have the freedom to exploit the ML ecosystem of Python and especially PyTorch, while keeping AgriPoliS in C++.

The training results demonstrate the superiority of the DRL-enhanced agent over the standard agent in AgriPoliS, reflected in the higher cumulative equity capital. Our analysis of hyperparameter variations reveals the critical influence of noise on the learning process, emphasizing the necessity of providing sufficient exploration space for successful learning.

Although our study showcases the feasibility of our approach, certain limitations warrant consideration. The deliberately small size of the simulated region, containing only seven agents, restricts the generalizability of findings to larger, more complex agricultural landscapes. Moreover, the substantial training time, approximately 14.5 hours for 1000 epochs, poses a practical challenge.The unique characteristics of the RL problem within AgriPoliS, including the inability to reset the environment during learning, the fixed number of transitions from start to terminal state, the definition of the state values and the rewards of transitions, present additional challenges. Notably, the absence of rewards for every action complicates the applicability of some RL algorithms, e.g. DDPG. This problem has parallels to strategic games such as the game Go, where rewards only accrue after a series of actions. The findings from our study therefore apply primarily to problems characterized by delayed effects of strategic decisions and spatial interactions, such as those involving competition for limited natural resources.

Building upon the insights gained from this study, future research endeavors aim to train agents in more realistic agricultural regions, thereby enhancing the model's applicability and relevance. Furthermore, exploring the training of multiple DRL agents in complex landscapes presents an exciting avenue for research, promising insights into the multifaceted economic, societal, and environmental consequences of intelligent decision-making in agriculture.

Acknowledgements: We acknowledge financial support from the German Research Foundation (DFG) through Research Unit 2569 "Agricultural Land Markets – Efficiency and Regula-

tion".

Bibliography

[An12] L. An. Modeling human decisions in coupled human and natural systems: Review of agent-based models. *Ecological Modelling* 229:25–36, 2012. Modeling Human Decisions.
doi:10.1016/j.ecolmodel.2011.07.010

[Bal97] A. Balmann. Farm-based modelling of regional structural change: A cellular automata approach. *European Review of Agricultural Economics* 24(1):85–108, 03 1997.
doi:10.1093/erae/24.1.85

[BDM22] M. Buchholz, M. Danne, O. Musshoff. An experimental analysis of German farmers' decisions to buy or rent farmland. *Land Use Policy* 120:106218, 2022.
doi:10.1016/j.landusepol.2022.106218

[BGM+21] A. Balmann, M. Graubner, D. Müller, S. Hüttel, S. Seifert, M. Odening, J. Plogmann, M. Ritter. Market power in agricultural land markets: concepts and empirical challenges. *Agricultural Economics* 70(4):213–235, 2021.
doi:10.30430/gjae.2021.0117

[Bon02] E. Bonabeau. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences* 99(suppl_3):7280–7287, 2002.
doi:10.1073/pnas.082080899

[BPR+11] F. E. Bert, G. P. Podestá, S. L. Rovere, Ángel N. Menéndez, M. North, E. Tatara, C. E. Laciana, E. Weber, F. R. Toranzo. An agent based model to simulate structural and land use changes in agricultural systems of the argentine pampas. *Ecological Modelling* 222(19):3486–3499, 2011.
doi:10.1016/j.ecolmodel.2011.08.007

[BW14] W. Britz, C. Wieck. Analyzing structural change in dairy farming based on an Agent Based Model. Technical paper 2014:1, Institute for Food and Resource Economics University of Bonn, 2014.
https://api.semanticscholar.org/CorpusID:208012431

[CH12] A. T. Crooks, A. J. Heppenstall. *Introduction to agent-based modelling*. Springer Netherlands, Dordrecht, 2012.
doi:10.1007/978-90-481-8927-4_5

[DSW01] A. De Janvry, E. Sadoulet, W. Wolford. *Access to land and land policy reforms*. Volume 3. UNU World Institute for Development Economics Research Helsinki, 2001.

[EE19] European Commission, Eurostat. Farm structure survey. Apr. 2019.
https://ec.europa.eu/eurostat/web/microdata/farm-structure-survey

[EEM20] European Commission, Eurostat, F. Mari. *The representativeness of the Farm Accountancy Data Network (FADN) – Some suggestions for its improvement – 2020 edition*. Publications Office, 2020.
doi:doi/10.2785/06861

[Eps99] J. M. Epstein. Agent-based computational models and generative social science. *Complexity* 4(5):41–60, 1999.
doi:10.1002/(SICI)1099-0526(199905/06)4:5¡41::AID-CPLX9¿3.0.CO;2-F

[Gra18] M. Graubner. Lost in space? The effect of direct payments on land rental prices. *European Review of Agricultural Economics* 45(2):143–171, 2018.
doi:10.1093/erae/jbx027

[HAB19] F. Heinrich, F. Appel, A. Balmann. Can land market regulations fulfill their promises? Forland-working paper 12, Humboldt Universität zu Berlin, 2019.
doi:http://dx.doi.org/10.18452/20890

[Hap04] K. Happe. *Agricultural policies and farm structures - Agent-based modelling and application to EU-policy reform*. PhD thesis, Leibniz Institute of Agricultural Development in Transition Economies (IAMO), 2004.

[HBB+18] R. Huber, M. Bakker, A. Balmann, T. Berger, M. Bithell, C. Brown, A. Grêt-Regamey, H. Xiong, Q. B. Le, G. Mack, P. Meyfroidt, J. Millington, B. Müller, J. G. Polhill, Z. Sun, R. Seidl, C. Troost, R. Finger. Representation of decision-making in European agricultural agent-based models. *Agricultural Systems* 167:143–160, 2018.
doi:10.1016/j.agsy.2018.09.007

[HKB06] K. Happe, K. Kellermann, A. Balmann. Agent-based analysis of agricultural policies: an illustration of the agricultural policy simulator AgriPoliS, its adaptation and behavior. *Ecology and Society* 11(1), 2006.
http://www.jstor.org/stable/26267800

[HOB13] S. Hüttel, M. Odening, A. Balmann. Agricultural land markets – Recent developments and determinants. *German Journal of Agricultural Economics*, 2013.
https://www.gjae-online.de/articles/agricultural-land-markets-recent-developments-and-determinants/

[KSB08] K. Kellermann, C. Sahrbacher, A. Balmann. Land markets in agent-based models of structural change. 692-2016-47500:18, 2008.
doi:10.22004/ag.econ.6647

[LB18] F. Lehn, E. Bahrs. Analysis of factors influencing standard farmland values with regard to stronger interventions in the German farmland market. *Land Use Policy* 73:138–146, 2018.
doi:10.1016/j.landusepol.2018.01.020

[LHP+15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
doi:10.48550/arXiv.1509.02971

[OH20] M. Odening, S. Hüttel. Introduction to the special issue 'agricultural land markets – recent developments, efficiency and regulation'. *European Review of Agricultural Economics* 48(1):4–7, 2020.
doi:10.1093/erae/jbaa023

[OVG+20] O. A. Osoba, R. Vardavas, J. Grana, R. Zutshi, A. Jaycocks. Modeling agent behaviors for policy analysis via reinforcement learning. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. Pp. 213–219. 2020.
doi:10.1109/ICMLA51294.2020.00043

[Rah10] S. Rahman. Determinants of agricultural land rental market transactions in Bangladesh. *Land Use Policy* 27(3):957–964, 2010.
doi:https://doi.org/10.1016/j.landusepol.2009.12.009

[RG11] S. F. Railsback, V. Grimm. *Agent-based and individual-based modeling: A practical introduction*. Princeton University Press, 2011.
http://www.jstor.org/stable/j.ctt7sns7

[SB18] R. S. Sutton, A. G. Barto. *Reinforcement Learning: An introduction*. MIT press, 2018.

[SHM+16] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot et al. Mastering the game of Go with deep neural networks and tree search. *Nature* 529(7587):484–489, 2016.
doi:10.1038/nature16961

[SLH+14] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller. Deterministic policy gradient algorithms. In Xing and Jebara (eds.), *Proceedings of the 31st International Conference on Machine Learning*. Proceedings of Machine Learning Research 32(1), pp. 387–395. PMLR, Bejing, China, 22–24 Jun 2014.
https://proceedings.mlr.press/v32/silver14.html

[TTK+23] M. Towers, J. K. Terry, A. Kwiatkowski, J. U. Balis, G. d. Cola, T. Deleu, M. Goulão, A. Kallinteris, A. KG, M. Krimmel, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, A. T. J. Shen, O. G. Younis. Gymnasium. Mar. 2023.
doi:10.5281/zenodo.8127026

[ZVC23] W. Zhang, A. Valencia, N.-B. Chang. Synergistic integration between Machine Learning and agent-based modeling: A multidisciplinary review. *IEEE Transactions on Neural Networks and Learning Systems* 34(5):2170–2190, 2023.
doi:10.1109/TNNLS.2021.3106777