Informatics

Crop rotation optimization for organic farming systems by combining model-based reinforcement learning methods with symbolic planning

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Data Science

eingereicht von

Magnus Wagner, M.Sc.
Matrikelnummer 12034922

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ. Prof. Dr. Allan Hanbury
Mitwirkung: Dipl.-Ing. Mag.rer.soc.oec. Dr.techn. Stefan Fenz

Wien, 29. März 2024

[illegible]

Magnus Wagner

Allan Hanbury

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Informatics

Crop rotation optimization for organic farming systems by combining model-based reinforcement learning methods with symbolic planning

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Data Science

by

Magnus Wagner, M.Sc.

Registration Number 12034922

to the Faculty of Informatics

at the TU Wien

Advisor: Univ. Prof. Dr. Allan Hanbury

Assistance: Dipl.-Ing. Mag.rer.soc.oec. Dr.techn. Stefan Fenz

Vienna, March 29, 2024

[illegible]

Magnus Wagner

Allan Hanbury

Declaration of Authorship

Magnus Wagner, M.Sc.

I hereby declare that I have written this work independently, have given full details of the sources and aids used, and have marked places in the work—including tables, maps and illustrations—which are taken from other works or from the Internet, either verbatim or in spirit, as borrowed, in any case indicating the source.

Vienna, March 29, 2024

[illegible]

Magnus Wagner

Kurzfassung

In der modernen Landwirtschaft ist die Kultivierung einer Fruchtfolge verschiedener Pflanzen auf einem Feld für viele Landwirte der Standard. Richtig angewandt fördert dieses Vorgehen nicht nur günstige Bodenbedingungen, sondern vermeidet auch das Auftreten von Schädlingen und Unkraut und steigert die Ernteerträge. Über viele Jahre hinweg haben Forschende versucht, das intuitive Wissen der Landwirte und den aktuellen Stand der Agronomie in Regelwerke und Modelle zu überführen, um optimale Abfolgen für die Fruchtfolge weltweit generieren zu können. Während regelbasierte Ansätze darauf abzielten, Sequenzen zu schaffen, die keine Fruchtfolgeregeln verletzen, berücksichtigten mathematische Modelle zusätzlich Durchschnittserträge und Marktbedingungen zur Gewinnoptimierung. In jüngster Zeit haben Fortschritte in der Anwendung von Reinforcement Learning (RL) auf dieses Problemfeld vielversprechende Ergebnisse erzielt. Dank der Fähigkeit der Modelle, iterativ von ihrer Umgebung zu lernen, konnten für komplexe mehrjährige Simulationsbedingungen optimale Fruchtfolgen gefunden werden, welche mit bereits verwendeten Anbauplänen übereinstimmten. Ein wesentlicher Nachteil bestand jedoch darin, dass die Agenten mehrere tausend Trainingsepisoden benötigten, um zu einer stabilen und optimalen Leistung zu konvergieren. Um dieses Problem zu lösen, erweitert dieses Projekt den vorgeschlagenen RL-Ansatz, um die Effizienz der Agenten erheblich zu verbessern. Darüber hinaus sollen die Modelle in der Lage sein, mit exogener Unsicherheit wie variierenden Erträgen und Marktbedingungen umzugehen. Auch im untrainierten Zustand sollten von den Agenten keine schlechteren Entscheidungen getroffen werden als von erfahrenen Landwirten, um eine Akzeptanz bei der Nutzung der Agenten in Entscheidungsunterstützungssystemen zu erreichen. Die implementierte Lösung wurde in einer Simulationsumgebung trainiert und evaluiert. Die Simulation wurde mit Fokus darauf erstellt, die relevantesten exogenen Effekte auf Ernteertrag und Gewinn abzubilden. Die Ergebnisse zeigen, dass es möglich ist, RL-Agenten mit hoher Proben-Effizienz zu konstruieren, die in Entscheidungsunterstützungssystemen für Landwirte eingesetzt werden können. Durch die Verwendung eines Soft Actor-Critic-Lernalgorithmus, die Kombination mit einem symbolischen Planer zur Beschränkung der Pflanzenauswahl auf passende Optionen und die Stabilisierung der frühen Trainingsphasen mit Wissen aus anderen Fruchtfolge-Experimenten zeigt der Agent stetige Leistungsverbesserungen bei geringem Risiko für den Landwirt, ungeeignete Pflanzen empfohlen zu bekommen. Dieses Ergebnis stärkt die Position von Reinforcement Learning als eine sinnvolle Option im Bereich der Fruchtfolge-Optimierung und eröffnet neue Wege für zukünftige Forschung.

Abstract

In modern farming, cultivating a sequence of varying crops on a plot of land is the standard for many farmers. This behaviour leads to better soil conditions, avoids pest and weed occurrence and increases crop yields if done correctly. Since many years, researchers have tried to convert the intuitional knowledge of farmers and the current state-of-the-art of agronomy into rule systems and models that can generate optimal crop rotation sequences for fields around the world. Rule-based approaches often focus on creating sequences not violating any agronomical or crop rotation rules, mathematical models additionally consider average yields and market conditions to optimize for profit. Recent advances in applying Reinforcement Learning to this problem domain demonstrated promising results. Due to the ability of the agents to learn iteratively from environments, optimal crop cultivation sequences could be obtained for complex multi-year simulation conditions which could match cultivation plans that were already in use. A major downside was however the agents' need for several thousand episodes of training to converge to a stable and optimal performance. To address this issue, this thesis project extends the proposed Reinforcement Learning approach to improve the sample efficiency of the agents significantly. Additionally, the agents should be able to deal with exogenous uncertainty like yield fluctuations and price and cost variability. Even in the untrained state, the agents should not make worse decisions than experienced farmers in order to achieve acceptance in the use of the agents in decision support systems before they have been able to adapt to the respective environment. The implemented solution was trained and evaluated on a simulation environment built to encompass the most relevant exogenous conditions affecting crop yield and farming profit. The results from the evaluation demonstrate that it is possible to construct RL agents with a high sample efficiency which can be deployed in decision support systems for farmers. By using a Soft Actor-Critic learning algorithm, combining it with a symbolic planner to restrict the crop selection to viable choices and stabilizing early training with knowledge from other crop rotation experiments, the agent shows steady performance improvements with a low risk for farmers to receive recommendations for unsuitable crops on their fields. This result fortifies the position of Reinforcement Learning as a viable option to address the problem of crop rotation optimization and opens new paths for future research.

Contents

CHAPTER 1

Introduction

1.1 Context & Motivation

Agriculture plays a central role in the nutrition of a large majority of people living on our planet. It exists for more than 10.000 years in our world and has always been a subject of improvement and optimization to achieve higher yields, farm more sustainably and generate tastier and healthier products [Bri23]. There exist many levers to improve crop yields and to use available resources more efficiently. Those include soil treatment, water resource management, activity and resource scheduling as well as fertilizer and pesticide usage. Modern decision support systems (DSS) for agriculture try to aggregate those different fields into complete tools which support farmers during their everyday business. They assist farmers with irrigation decisions, deliver weather predictions, simulate crop growth under selected conditions and help coordinating and scheduling production resources. In the context of Industry & Agriculture 4.0, those tools have access to an abundance of data which can either be gathered locally via sensors and cameras or can be accessed from databases containing regional data about the weather, soil types and market conditions. More sophisticated systems can for example coordinate autonomous drones to detect pest and weed infestations and locally apply pesticides or gather and evaluate live sensor data from the fields [ZMBM20].

An important decision farmers need to make several times each year is what to grow next on their plot of land. Intuitively, the crop with the highest profit margin would be the best decision. Despite that, humans have already realized in the early days of agriculture and farming practice that specific sequences of crops have higher yields than just planting single crops on the same field repeatedly. Practices were already written down during the Roman Times and nowadays, there exist many example sequences and rule sets that farmers can orient on to plant crops and fruits with the highest yields in the right order [Fra05]. Optimizing crop sequences to achieve the highest potential profit falls in the problem domain of crop rotation planning. Having a DSS propose the optimal next crop to plant is therefore a building block improving some aspects of farming which can be integrated with other tools to optimize agricultural production.

Due to the advantages for the soil and it being a more sustainable approach towards planting crops and fruits, crop rotation planning is often combined with organic farming. Organic farming in Austria falls under EU guidelines. Among other things there are strict rules forbidding certain types of fertilizers, pesticides and seeds which can as a consequence inhibit the production of optimal yields [uR18]. Organic farming is however connected with many benefits also attributed to crop rotation management like environmental protection and a higher resilience to environmental changes [JAT^{+}17]. From a financial perspective, organic farming practices are more expensive in labour costs but on average lead to a price premium on the markets due to consumer behaviour [CR15]. Additionally, they are subsidized by the Austrian government and the EU [Rec22],[ECD18]. Under those circumstances, about 26% of all agricultural area in Austria is already used for organic farming and surpasses the EU "Farm-to-Fork" strategy to use 25% of the agricultural area organically until 2030 [Com20]. Using agricultural expert knowledge and agronomical guidance to optimize crop selection for organic farming can be another step to increase the attractiveness of organic farming when compared to conventional farming.

The main problem with the previously mentioned rule sets is though, that they are too generic for the diversity of different farm and market conditions. As an example, they usually do not count in farm size, number of fields, soil characteristics, climate and weather zones. Additionally, the crop rotation plans must be tailored to the farmer's available labour and equipment resources, as some crop types might be too much effort for a small size farm. Legal requirements and regulations are another constraint that farmer's must adhere to depending on their location [PKB21]. Current rule-based algorithms for crop rotation planning therefore lack generalizability. In the field of decision support systems, there are many alternatives to rule-based algorithms though. Some of the more prominent examples are quantitative optimization techniques like mathematical modelling or linear programming, custom scoring systems or evolutionary algorithms [SWT23]. In addition, there are recent advances in using Deep Reinforcement Learning (Deep RL) agents for crop rotation optimization [FNFW23]. They can be trained on a simulation environment to predict optimal crop sequences and are able to adapt to real conditions when being used at an actual farm.

1.2 Problem Definition & Research Goals

A key issue with the previously mentioned quantitative optimization techniques is, that they generalize similarly bad for the multitude of possible farm characteristics and regional conditions which influence crop suitability and yield. Most techniques rely on a set of input data including average yields for different crops, static market prices and static effects on yield from crop rotation rules being followed or broken. Knowledge is often gained from case studies on single farms over several years or from aggregated data. Studies from different regions indicate large variances in the observed results [SSS11]. Additionally, many proposed methods do not include uncertainty impacting variance in observed yields which represents a risk for farmers [ACC00]. To combat those concerns, a model is necessary that can adapt to the real environment and represent yields accurately on the farm-level instead of only using aggregates. It needs to be able to consider uncertainty when proposing crops to grow and should be able to generalize over different regions. A thorough check of the literature reveals that linear programming approaches from the literature do not offer this set of features and evolutionary algorithms similarly only allow optimizing goals for a fixed set of input data. The RL algorithm proposed by Fenz et al. (2023) allows the user to let the model train further on the target environment and therefore has an advantage in terms of adaptability [FNFW23]. A problem is though that many Reinforcement Learning agents learn slowly and have a low sample efficiency as they are usually trained until convergence on simulation environments with computational resources being the only limiting factor. For the deployment of RL agents in real environments, this sample efficiency needs to be improved drastically as only one training sample can be generated per year and field and a typical number of training steps for Reinforcement Learning would lead to the model only adapting to the actual environment after hundreds of years. In summary, we can define the following problems to be addressed by this master thesis project:

1. A crop rotation optimization model must be able to deal with uncertainty in the input data and must be adaptable to be useable by farmers for individual farm characteristics and different regions.
2. While the model is not adapted yet to individual farm characteristics due to the lack of empirical data, it must follow expert knowledge and avoid adverse crop rotation sequences.
3. The adaptation process to individual farm characteristics must happen in a reasonable amount of time to make model usage viable for farmers.

One solution to those problems would be to pretrain the agent on a simulation environment representative for the region the farm is located. This was also used by Fenz et al (2023) to generate crop rotation sequences suitable to be used by farmers in Austria [FNFW23]. Afterwards, the trained model could be applied to select consequent crops to grow for farms from that region and could be adapted individually for each farm setting with continued learning from actual experience. Another idea to improve sample efficiency is to use an approach called model-based Reinforcement Learning. Here, the RL agent is additionally being trained on simulated trajectories produced by a transition model of the environment instead of only using real experience for training. The transition model itself is regularly updated with samples from the training environment. A model-based RL method suitable for the crop rotation optimization problem was proposed by Janner et al (2019) [JFZL19]. It uses an ensemble of Probabilistic Neural Networks to predict next states and employs trajectory sampling with short rollouts as a planning method to train a soft actor-critic agent selecting the next best action for the current state. The method is called Model-based Policy Optimization (MBPO) and shows state-of-the-art performance for uncertain environments with a lower number of environmental samples than comparable model-free algorithms. A further feature to improve the sample efficiency of the RL agent could be the usage of a hybrid model which combines the reasoning over knowledge from crop rotation rules and recommendations based on expert knowledge with an RL agent. The adaptation of the agent to the real environment can then be sped up by limiting exploration to actions not breaking any rules. This will on one hand accelerate the assessment of favourable actions for the specific farm environment but can on the other hand prevent the selection of crops detrimental to the crop rotation and the yields of following crops. The approach to formalize expert knowledge into constraints for the RL agent used for this master thesis project is called answer set programming (ASP). It is a form of declarative programming and can be utilized to find solutions satisfying all rules and constraints of a problem formulation [GKK$^+$11]. In the context of crop rotation optimization, solutions would represent suitable crop rotation sequences the RL agent can explore. As a last proposed feature, updates from neighbouring farms can be utilized to accelerate model training. Agents could have access to more data from similar conditions at the same time which would further improve the learning speed.

The construction of a hybrid model incorporating the proposed features is specified in more detail in the methodical section of this thesis. The goal of this project is to evaluate them and their effect on performance. Performance is defined in more detail in Section 3.3 and encompasses the average farming profit per crop rotation, the total farming profit over the training run and the profit stability during deployment of the agent. The evaluation is defined by the following research questions:

- RQ 1: To which extent can performance be improved by using the proposed features in comparison to baselines?
  - 1.1: By how much do hybrid systems combining a symbolic planner with an RL agent show a better performance on crop rotation problems than simple RL agents without rule-based planning?
  - 1.2: To which extent can the use of more modern RL learning algorithms like soft actor-critic learning improve performance when compared to deep Q learning?
  - 1.3: In which situations and by how much do hybrid systems combining a symbolic planner with an RL agent show a better performance on crop rotation problems than only using a symbolic planning system like answer set programming to select crops to grow?
  - 1.4: To which extent can performance be improved by using model-based RL when compared to model-free methods?
  - 1.5: To which extent can agent updates from neighbouring farms improve performance in comparison to agents only updating from their own farm's experience?

  - 1.6: To which extent can agents already pretrained on other environments improve performance when compared to agents without pretraining?

- RQ 2: To which extent can the use of probabilistic action selection by the RL agent improve diversity of crop selection when compared to deterministic algorithms?

1.3 Structure of the work

The content of this master thesis is structured in the following way. Chapter 2 (Background and Related Work) delves into comprehensive and recent research on symbolic planning algorithms, reinforcement learning and hybrid systems that integrate both concepts. The discussion encompasses the distinction between model-free and model-based reinforcement learning, the underlying principles of function approximation and policy gradient methods and a review of recent studies about the evaluation of RL models. Beyond establishing the foundational methods for the proposed hybrid model, the chapter explores the theoretical and literary landscape surrounding crop rotation planning and yield optimization. Additionally, a section is dedicated to various methods in the literature aimed specifically at addressing the crop rotation optimization problem, along with an exploration of research in related fields such as crop yield prediction and predicting the next crop in a sequence of previous crops. In Chapter 3 (Experiment Design), the focus lies on presenting the experimental setup in detail. This includes a detailed explanation of the simulated environment employed for model training and evaluation, an exploration of the algorithms underpinning the proposed hybrid model, and a thorough examination of the evaluation procedure. The origins of assumptions and settings in the simulation environment are outlined, with a clear exposition of their impact on model rewards. Following this, the model characteristics and features are specified in detail. It is explicitly stated how different types of crop rotation rules and constraints are depicted in an Answer Set Program, which algorithms and techniques are used to implement the RL agent, how both systems are combined and why the complete model might perform better than the selection of baselines. Additionally, various strategies for utilizing available experience are introduced, along with a description of their implementation in the agents. Subsequently, the evaluation approach to answer the research questions and to obtain significant results is defined. The section outlines the baseline models used in the experiments, the performance indicators adopted as measures and further experiment details. This information, supplemented by additional details in the appendix, ensures the possibility for complete reproducibility of the results. In Chapter 4 (Results), the outcomes of the study are presented and discussed in relation to the research questions. Plausible explanations for the observed results are theorized. In Chapter 5 (Summary), the findings from this project are summarized, contextualized within the current research field, and the limitations of this work are stated. Possible future research objectives building upon the findings from this project are identified.

CHAPTER 2

Background and Related Work

In the following section, recent research about symbolic planning algorithms, reinforcement learning and hybrid systems is discussed. The specifications of models and evaluation metrics used in the project implementation are explained in more detail. In addition, the chapter explores the topical domain of crop rotation planning and yield optimization. A focus lies on methods addressing the crop rotation optimization problem, accompanied by an exploration of research in related fields such as crop yield prediction and temporal crop prediction. Those related fields are thematized to understand their relevance and potential for crop rotation optimization algorithms.

2.1 Symbolic Planning

For the crop rotation optimization problem, it is key to find crop rotation sequences following expert knowledge recommendations and constraints. This type of problem is from the family of Boolean satisfiability problems with the objective to find stable models. Approaches to solve this problem are called satisfiability solvers [GKK$^+$11]. They provide a combinatorial reasoning and search platform based on propositional logic. The goal for the development of SAT solvers in the past years has been that they offer a highly expressive knowledge representation while still being able to solve problems in worst-case polynomial time. While polynomial time can still be problematic for large problems, it was shown that most random-generated satisfiability problems could be solved in close to linear time [MSL$^+$92]. An SAT solving approach well suited for knowledge representation and reasoning is called answer set programming (ASP). ASP is a form of declarative programming to be able to solve NP-hard search problems. It originates from the stable model semantics of logic programming introduced by Gelfold & Lifschitz (1988) [GL88]. The user describes the problem with a high-level representation language which is automatically translated into a low-level propositional representation. The problem description contains rules and constraints whereas a collection of rules results in a unique stable model after solving if a solution is possible. A stable model therefore describes a solution to the problem satisfying all rules and constraints.

A reward-winning solver for answer set programming named *clingo* was developed by Gebser et al. (2011) [GKK$^+$11]. It is part of the Potsdam answer set solving collection (*Potassco*) which is continuously being developed since its first introduction. *Clingo* mainly consists of two parts, a grounder named *gringo* [GKO$^+$09] and a solver named *clasp* [GKNS07]. The user defines the problem via the input language of *gringo*. The grounder then translates the problem definition into a ground logic program which is read by the solver *clasp*. The program is consequently simplified through preprocessing by removing redundancies and solved afterwards. Possible solutions are presented to the user.

The following paragraph introduces basic concepts in the input syntax for clingo aligned with the current user guide for clingo [KKS12]. Answer set programs are usually defined via an initial instance and an encoding applying to every instance. The instance describes the specific setup, the encoding represents the rules constraining the solution to the problem. In the field of crop rotation planning, the instance would for example be the definition of the starting conditions with previously planted crops, market prices and soil characteristics. The encoding would contain definitions about what a crop is, what the possible actions are and how different subsequent actions are constrained by crop rotation rules. These parts of the program are defined via facts, rules and constraints:

$$\textbf{Fact: } A_0.$$
$$\textbf{Rule: } A_0 :- L_1, \dots, L_n$$
$$\textbf{Constraint: } :- L_1, \dots, L_n$$

A fact A_0 or the head A_0 of a rule are atoms in the form of a constant or function. The body behind the ":–"-symbol is a set of literals each being either positive or negative. Facts are unconditionally true. Rules are true if all positive literals in the body are true and all negative literals (with a "not"-prefix) are satisfied. Constraints filter solution candidates. If a solution candidate does not fulfil all constraints, it is not an answer set / solution to the problem. In the head of a rule or fact, several atoms can be divided by the ";"-symbol, which means that the head holds true if at least one of the atoms is true. The *clingo* syntax supports many additional constructs to aid the user in defining the answer set program. Some examples are intervals and the pooling of terms, aggregates like sums over numerical variables and conditional literals. Furthermore, it is possible to optimize numerical variables in the answer set program via maximization or minimization.

2.2 Reinforcement Learning

2.2.1 Background

Reinforcement Learning (RL) as a practice had a revival in the early 1980s and focuses on trial-and-error learning of a policy on the set of optimal control problems. A popular former method to solve those problems is called Dynamic Programming and was introduced by Richard Bellman [Bel54],[Bel66]. Bellman also introduced the concept of stochastic optimal control problems named Markov decision processes (MDPs) [Bel57]. Until now, most RL problems are formulated as MDPs. In an MDP, an agent starts in an environmental state s and can select an action a. The agent then receives a reward r from the environment and the environment updates to be in a new state s'. The interaction between agent and environment leads to a sequence of actions and states called trajectory [SB18].

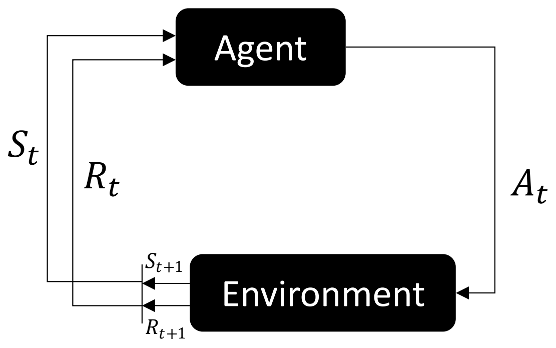


Figure 2.1: Agent-environment interaction in an MDP [SB18].

As Markov decision processes are stochastic, an action in a state can result in different following states. The probability of entering those states is depicted by the four-argument dynamics function p shown in formula 2.1.

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r | s, a) \;=\; 1 \tag{2.1}$$

By continuously interacting with the environment, the agent learns to adapt its policy to achieve its objective, which is typically reward maximization. The total reward can be divided into the immediate reward obtained from the environment and potential future rewards. If no difference is made between those two, the expected return of a policy p on a finite MDP can be formulated like in formula 2.2 where T is the final step of the process:

$$G_t \doteq R_{t+1} + R_{t+2} + \;\cdots\; + R_T \tag{2.2}$$

For continuing tasks, the expected reward could easily be infinite. To avoid this, a discounting factor γ can be introduced to discount future rewards:

$$G_t \doteq \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.3}$$

Many RL methods involve the estimation of a value function which defines a value for each state under a policy π:

$$v_\pi(s) \leftarrow \sum_a \pi(s,a)\left[r(s,a) + \gamma \sum_{s'} p(s'|a,s)\, v_\pi(s')\right] \tag{2.4}$$

It calculates the expected value for a state over all possible actions with their selection probabilities represented by policy π, the resulting immediate rewards r and the discounted values of future states s' with their transition probabilities p. The value function therefore represents the expectation value of immediate and future rewards. As an alternative to the value-function, action-state combinations can be described with an action-value function q under policy π. The difference to the value function is that the action-value function describes the value of selecting action a in state s. While the difference is mainly formal, the action-value function depicted in formula 2.5 should be mentioned as it is used by many popular RL algorithms:

$$q_\pi(s,a) \leftarrow r(s,a) + \sum_{s'}\left[p(s'|a,s) \sum_{a'} \pi(a'|s')\, q_\pi(s',a')\right] \tag{2.5}$$

To find the value function under an optimal policy when a perfect model of the environment is known, dynamic programming can be equipped. It can be counted as an early variant of model-based RL [BD15],[SB18],[Wat89]. Dynamic programming was originally defined by Bellman for a wide range of problems [Bel54]. In the context of reinforcement learning, the algorithm alternatively evaluates the current policy until the value function converges and then improves the policy to select the best action under the new value function for each state. Even if the policy evaluation step is not run until convergence itself, this procedure guarantees convergence to an optimal policy. While convergence is guaranteed, environments with huge state spaces face the issue that it is only achieved after a long computation time due to the algorithm's need to visit every state. For many problems though, observing every possible state is not necessary as many reachable states will lead to suboptimal rewards and do not need to be explored in detail. Additionally, dynamic programming can only be used if a perfect model of the environment is known, which is unrealistic for most real-world applications.

2.2.2 Model-free Reinforcement Learning

To address this problem and to decrease the immense computational effort, Monte Carlo (MC) methods can be utilized [KW09]. Instead of visiting every state, the algorithm only samples trajectories from the environment and learns from those by averaging rewards for each state-action pair. Formula 2.6 describes how to do value updates in an incremental way with α as the learning rate:

$$q_{n+1}(S_t, A_t) = q_n(S_t, A_t) + \alpha[G_t - q_n(S_t, A_t)] \tag{2.6}$$

The action value is iteratively updated with the error term multiplied by α. The error term is the difference between the experienced reward G_t after finishing the episode and the expected value $q_n(S_t, A_t)$ inferred from the action-value function. To promote exploration and guarantee convergence in the limit while using a sampling approach, several methods have been developed. A simple option is to select the starting state for each episode randomly [SB18]. Another popular option is to use a behaviour policy for action selection that is different from the target policy which is optimized. This approach is called off-policy learning. A suitable behaviour policy for off-policy learning could be an ϵ-greedy policy. This policy selects the estimated best action most of the time but selects a random action with probability ϵ. By doing so, exploration of all states is guaranteed. A problem with Monte Carlo methods is though, that the model must wait before finishing an episode to learn. Therefore, MC methods cannot be used for continuous learning. As a solution, temporal difference (TD) methods can be applied [Sut88]. In contrast to Monte Carlo methods using the total reward from an episode to update the value-function, TD methods can already update the policy after each step by estimating the total reward via bootstrapping. The incremental learning is depicted formula 2.7:

$$q_{n+1}(S_t, A_t) = q_n(S_t, A_t) + \alpha[R_{t+1} + \gamma q_n(S_{t+1}, A_{t+1}) \; - \; q_n(S_t, A_t)] \tag{2.7}$$

The total episodic reward is exchanged with a sum of the immediate reward R_{t+1} and the discounted estimation $\gamma q_n(S_{t+1}, A_{t+1})$ of the action-value of the next state . In practice, TD learning converges faster than MC methods due to the intra-episode learning. An alternative to TD learning which is still widely used in its variants is Q-Learning. It was introduced by Watkins in 1989 and is a form of off-policy TD learning. The agent selects an action based on an ϵ-greedy policy but the action-values are updated for a greedy policy as shown in formula 2.8 [Wat89].

$$q_{n+1}(S_t, A_t) = q_n(S_t, A_t) + \alpha[R_{t+1} + \gamma \; max_a \; q_n(S_{t+1}, a) \; - \; q_n(S_t, A_t)] \tag{2.8}$$

There is a variant of Q-Learning worth mentioning. Q-Learning suffers from an effect called maximization bias which leads to the agent overestimating actions when rewards are positive by chance. A solution to this issue is to train two action-value functions independent of each other alternately. During the policy evaluation step, the bootstrapped estimate is used from the respective other action-value function and is therefore unbiased. In practice, convergence is improved for many examples and the method is available in most popular reinforcement learning libraries.

2.2.3 Function Approximation & Policy Gradient Methods

The previously mentioned algorithms work well for tabular environments where each state receives a single value from the value function. If states are represented by feature vectors over several dimensions, an alternative value function must be used which is in a parameterized functional form. A simple example would be a linear function mapping the feature vector of a state to a single value by multiplying it with a weight vector.

As the number of weights represents the number of dimensions which is typically much smaller than the number of possible states, updates from single experiences will affect the estimated values of many states. This generalization can make the learning more efficient. As an example, the update for single-step TD Learning of a value function is adapted to be similar to formula 2.9 [Sut88],[Li17]:

$$w_{n+1} = w_n + \alpha[R_{t+1} + \hat{v}\,(S_{t+1}, w_n) - \hat{v}\,(S_t, w_n)]\nabla\hat{v}(S_t, w_n) \tag{2.9}$$

The weights of the value function are modified via stochastic gradient descent to minimize the error between the value estimation and the bootstrapped value. In the specific case of combining function approximation with TD learning, one would speak of semi-gradient descent due to the target also being an estimation [SB18]. Instead of relying on a linear function to be able to represent values in a complex environment, it is also possible to use other ways to transform the input feature vector. One example are artificial neural networks. The weights of the neural network are learned and updated via backpropagation [LTHS88]. A popular example from recent research has been demonstrated in 2015, when Mnih et al. used a convolutional neural network to transform pixel-based images and possible actions into action-values with an algorithm named Deep Q Learning (DQN) [MKS$^+$15]. The crop rotation optimization algorithm by Fenz et al. also used Deep Q Learning to learn and predict action-values across the state space [FNFW23].

While the previously described methods learn an action-value function and choose the action with the highest value in each state to follow an optimal policy, there is also another way to select actions. Policy gradient methods learn a parameterized probabilistic policy instead of a parameterized value function and select actions by sampling from the policy. This represents another way of function approximation as the probability distribution of selecting an action in each state is calculated from a function transforming the input feature vector. The objective to maximize performance is achieved by continuously adapting the function parameters θ via gradient ascent [SB18]:

$$\theta_{n+1} = \theta_n + \alpha\widehat{\nabla J(\theta_n)} \tag{2.10}$$

$\widehat{\nabla J(\theta_n)}$ is an estimate of the gradient of the performance with respect to θ. The advantage of policy gradient methods is that the learned policies are not deterministic like a greedy policy. Instead, actions with almost the same values have almost the same probability to be selected by the agent after soft-maxing the action preferences. A version of this approach is the REINFORCE algorithm proposed by Williams (1992) [Wil92]. It utilizes an episodic Monte-Carlo approach by generating a complete episode and updates the policy approximation parameters in the following way for each timestep t in the episode with length T:

$$\theta_{n+1} = \theta_n + \alpha\gamma^t G\nabla ln\pi(A_t|S_t, \theta_n) \tag{2.11}$$

where

$$G = \sum_{k=t+1}^{T} y^{k-t-1} R_k. \tag{2.12}$$

With two changes, this method can be transformed into the Actor-Critic (AC) algorithm first investigated by Witten (1977) [Wit77] and further developed by Barto et al. (1983) [BSA83]. At first, a baseline can be subtracted from each reward to reduce variance. It has been shown that the estimated value function for each state is a suitable baseline as it is flexible, varies with state and represents the average value of a state. Actions performing better than the average receive a positive reward, actions performing worse receive a negative reward which leads to a lower probability for action selection. The second change is, that the AC algorithm adopts bootstrapping by using estimated values during one-step updates alike to TD learning. For each step t during an episode, the one-step return δ_t is calculated [KT99]:

$$\delta_t = R_{t+1} + \hat{v}\left(S_{t+1}, w_n\right) \; - \; \hat{v}\left(S_t, w_n\right) \tag{2.13}$$

It is used to update the value function weights w (like in value function approximation) and the policy parameters θ:

$$w_{n+1} \; = \; w_n \; + \; \alpha_w \delta_t \nabla \hat{v}\left(S_t, w_n\right) \tag{2.14}$$

$$\theta_{n+1} = \theta_n + \alpha_\theta \gamma^t \delta_t \nabla ln \pi(A_t|S_t, \theta_n) \tag{2.15}$$

A modern variant of the actor-critic algorithm which is used for this master thesis project is the soft actor-critic (SAC) algorithm. It was developed by Haarnoja et al. (2018) [HZAL18]. The objective function for the policy depicted in formula 2.16 is adapted to include the entropy H of the policy π_θ in addition to the rewards obtained by following the policy:

$$J\left(\pi\right) = \sum_{t=0}^{T} E_{(s_t, a_t) \sim \rho_\pi} \left[r(s_t, a_t) + \alpha H\left(\pi_\theta\left(\cdot|s_t\right)\right)\right] \tag{2.16}$$

Here, θ are the parameters of the policy and ρ_π denotes the state- and state-action-marginals of the trajectory distribution induced by policy π. The inclusion of entropy in the objective prevents agents from optimizing their policy to be almost deterministic. Probabilities for sub-optimal actions are higher which promotes exploration. In practice, the new part of the objective is integrated in the following way: During the policy evaluation step representing the critic, the soft Q-value can be updated iteratively with the following formula for individual states and actions:

$$Q_{n+1}(s_t, a_t) \;=\; r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\ p}[V_n(s_{t+1})] \tag{2.17}$$

where

$$V_n(s_t) = \mathbb{E}_{a_t \sim \pi}[Q_n(s_t, a_t) \;-\; log\ \pi(a_t|s_t)] \tag{2.18}$$

Here, the value function V contains the entropy part of the objective. The new policy is calculated in the policy improvement step by minimizing the Kullback-Leibler divergence between a policy from a set of possible policies Π corresponding to Gaussians and the normalized exponential Q-function:

$$\pi_{n+1} = \underset{\pi\prime \in \Pi}{arg\ min}\ D_{KL}\left(\pi\prime(\cdot|s_t)||\frac{exp(Q_{\pi_n}(s_t, \cdot))}{Z_{\pi_n}(s_t)}\right) \tag{2.19}$$

A repeated alternation between policy evaluation and policy improvement leads to improvement in the tabular case. For the non-tabular case including function approximation, which is used in the project, the updates must be applied to the function parameters instead of updating the value functions and the policy directly for each state. During each gradient step, the following gradients multiplied with a learning rate are subtracted from the parameters:

$$\text{Value-Function: } \hat{\nabla}_\varphi J_V(\varphi) = \nabla_\varphi V_\varphi(s_t)(V_\varphi(s_t) \;-\; Q_\phi(s_t, a_t) + log\ \pi_\theta(a_t|s_t)) \tag{2.20}$$

$$\text{Action-Value-Function: } \hat{\nabla}_\phi J_Q(\phi) = \nabla_\phi Q_\phi(s_t, a_t)(Q_\phi(s_t, a_t) \;-\; r(s_t, a_t) \;-\; \gamma V_{\bar{\varphi}}(s_t)) \tag{2.21}$$

$$\text{Policy-Gradient: } \hat{\nabla}_\theta J_\pi(\theta) = \nabla_\theta log\pi_\theta(a_t|s_t) + (\nabla_{a_t} log\pi_\theta(a_t|s_t) - \nabla_{a_t} Q_\phi(s_t, a_t)) \nabla_\theta f_\theta(\epsilon_t; s_t) \tag{2.22}$$

The reparameterization trick is used to obtain an unbiased gradient estimator which makes the policy gradient independent of the parameters of the policy function: The action a_t in the policy is transformed with a neural network $a_t = f_\theta(\epsilon_t; s_t)$ where ϵ_t is an input noise vector sampled from a fixed distribution. Additionally, instead of using the current value-function to update the action-value-function, an exponentially moving average $\bar{\varphi}$ of the value network weights was used to stabilize training. They also apply double Q-Learning to avoid overestimating the action-values during training. Further details can be found in the original paper by Haarnoja et al. (2018) [HZAL18]. In comparison to other actor-critic methods like TRPO [SLA+15], PPO [SWD+17] or A3C [MBM+16], the SAC algorithm offers a higher stability and a better sample efficiency while learning which is mainly due to the algorithm using off-policy learning and the addition of entropy maximization in the objective.

2.2.4 Model-based Reinforcement Learning

In contrast to model-free algorithms, model-based methods like dynamic programming rely on samples simulated by a model representing the environment when updating the policy. The model estimates the dynamics function of the MDP to predict the reward and next state from a previous state and a selected action. Current popular state-of-the-art models show that they can compete with model-free methods while having a better sample efficiency [PKP20]. When agents learn from a transition model instead of the real environment, it is called *planning*. An algorithm combining planning and direct learning from experience was introduced by Sutton in 1991 and is named *Dyna-Q* [Sut90],[Sut91]. The algorithm uses ϵ-greedy action selection and learns via Q-Learning from the experience in a tabular setting. Additionally, a dynamics model is adapted to represent the newly acquired transition too. After learning from real experience, the agent learns for several steps from randomly selected previously observed states with previously taken actions from those states. The model selects a previously experienced transition for this state-action combination randomly to simulate the transition. In non-tabular scenarios, the model would be trained to predict reward and next state from the randomly selected previous state and action. As the agent is not limited to learning from experience, *Dyna-Q* is more sample-efficient than common model-free algorithms. On the downside, the dynamics model can be biased which will lead to a biased value function and policy when learning from model samples. Instead of planning with random transitions or in another variant planning with random trajectories from experience, a model-based learner can also plan at decision time. This type of planning always uses the current state as a starting position and starts sampling trajectories from there. Decision-time planning is especially useful in applications without the need for fast responses which would also be the case for a crop rotation planner [SB18]. The sampling of trajectories is usually guided by some kind of heuristic. ϵ-greedy algorithms could already be seen as a heuristic, but more sophisticated methods have become more popular over time. A well-known example is the TD-Gammon bot by Tesauro [Tes94]. In later versions, it uses a game-specific heuristic to plan games of Backgammon by selecting the own best action depending on the values of the available opponents' actions afterwards. The authors also used trajectory sampling as a computationally more efficient variant, which means that they did not analyse each possible trajectory over n steps but only sampled n times from all possible trajectories [TG96]. Another popular rollout strategy is called Monte Carlo Tree Search (MCTS). It was used by Silver et al. (2016) to train an agent able to reliably beat human grandmasters in the game Go [SHM^{+}16]. The key idea behind the rollout strategy is to build a tree of actions starting with the current state and expanding it repeatedly following a tree policy. The tree policy is backed up by running simulations from the newly expanded state to the end of each episode to receive a reward. Albeit highly popular model-based RL algorithms like AlphaGo can be trained on simulated game environments with fully known rule sets [SHS^{+}17], RL agents learning on dynamics models reflecting real and highly complex environments need to consider the uncertainty in the predicted next states and rewards [PKP20]. An agent possessing this capability was recently developed by Janner et al (2019). It uses an ensemble of Probabilistic Neural Networks to predict next states and applies trajectory sampling with short rollouts as a planning method. The method is called Model-based Policy Optimization (MBPO) and shows state-of-the-art performance for uncertain environments with a low number of environmental samples when compared to model-free algorithms [JFZL19]. The main features distinguishing it from other models are a) the usage of an ensemble of probabilistic networks to predict environment dynamics and b) limited rollout lengths to disconnect the model horizon from the task horizon. The idea of using a bootstrapped ensemble of probabilistic networks was introduced by Chua et al. (2018) in their PETS (Probabilistic Ensembles with Trajectory Sampling) algorithm [CCML18]. Each probabilistic network parametrizes a Gaussian distribution with diagonal covariance which can be sampled to predict the next state:

$$p_\theta(s_{t+1}, r|s_t,\ a_t) = \mathcal{N}(\mu_\theta(s_t, a_t), \Sigma_\theta(s_t, a_t)) \tag{2.23}$$

The approach addresses both aleatoric uncertainty and epistemic uncertainty. Aleatoric uncertainty describes the inherent uncertainty of the system. Epistemic uncertainty describes the uncertainty of the trained dynamics function stemming from a lack of data. In contrast to aleatoric uncertainty, epistemic uncertainty can be reduced with a larger number of samples [CCML18]. The use of probabilistic neural networks in the MBPO and PETS algorithms is appropriate to represent aleatoric uncertainty. By using bootstrapped ensembles of probabilistic networks, epistemic uncertainty can be represented. The parametric approach using neural networks is an alternative to earlier works addressing uncertainty via Gaussian processes, which is non-parametric Bayesian method [WHF05],[DR11]. While Gaussian processes outperform parametric approaches for many types of environments due to them representing epistemic uncertainty more successfully, parametric approaches revealed to be more suitable for high-dimensional environments [MBP+23]. The MBPO algorithm mainly differs from the PETS algorithm by using limited rollouts with random starting states and by simply applying the current policy and dynamics model to sample trajectories. In the PETS algorithm, a particle-based variant of model-predictive control (MPC) is used as the rollout method [CA07]. As an agent, the MBPO algorithm adopts a soft-actor critic algorithm.

2.2.5 Evaluation of Reinforcement Learning models

Modern Reinforcement Learning algorithms still have many issues with instability over extrinsic factors like hyperparameters or codebases and intrinsic factors like random seeds and environment characteristics. A study by Henderson et al. (2018) showed a large variance in performance when varying those factors for modern actor-critic methods like TRPO, PPO and DDPG [HIB+18]. The standard practice is to evaluate the models with different randomly generated environments and get aggregated measures over all validation runs. Most papers report their results from those runs as average point estimates when comparing performance to baselines. Henderson et al. recommend to additionally run significance tests like 2-sample t-tests or Kolmogorov-Smirnov tests and evaluate bootstrap percent differences with confidence intervals to compare performance between algorithms. In a paper by Colas et al. (2018), they recommend running at least 20 different random seeds per test, using low significance levels, and performing Welch t-tests instead of comparing bootstrap confidence intervals [CSO18]. Chan et al. (2020) propose including reliability measures to complement typical performance measures when comparing RL algorithms. They divide *reliability* into *dispersion* and *risk* and suggest different procedures when testing those during and after training and across training runs [CFC+19]. Dispersion is measured as the interquartile range (IQR – the difference between the 25th and 75th percentile) over sliding windows during the training run, as the IQR across training runs and as the IQR across evaluation runs with a fixed policy after training. Risk is determined by the *conditional value at risk* (CVaR), which can also be measured during training (e.g. on first-order differences), across training runs and across evaluation runs. CVaR is defined as the average performance over runs in the lowest α-quantile. Agarwal et al. (2021) suggest not using mean performance across runs as a measure at all [ASC+21]. They find the interquartile mean to be more robust towards outliers while still being more statistically efficient than the median. By combining many of those suggested measures in a performance profile, it is possible to present performance, variance in performance and algorithm reliability in a more precise way.

2.3 Hybrid Systems

For sequential decision-making, Rule-based Planning and Reinforcement Learning have been two popular approaches that differ a lot in how they address the problem. Planning has been favoured in the 20th century when the integration of expert-knowledge into automated systems seemed promising. Reinforcement Learning however focuses on learning from experience without necessarily knowing anything about the environment beforehand. In the field of agricultural crop rotation planning, it is key to improve the sample efficiency of trained models as only a tiny amount of actual data can be gathered from a single plot of land over several years. Reducing the need for data by integrating expert knowledge into the model therefore seems to be a reasonable idea. To bridge the gap between expert systems and purely data-driven models, current research proposes different ways on how to mix both approaches into hybrid-models. Some approaches focus on reward-shaping to include expert knowledge into the RL agent training [NBM+19],[DHFLHvH22], others use rule-based planners as slow but more precise systems than the trained RL agent for time-critical applications [GSR21],[GCF+22]. Another idea is to use symbolic planning on high-level abstractions of the environment and solve each low-level step derived from the symbolic plan with RL which is a concept similar to Hierarchical Reinforcement Learning [GK05],[ESL19],[óIYIM19],[BM03]. A further paper suggests to use symbolic planning to reduce the size of the exploration space for the RL agent [LIS16]. A reward-shaping approach by Noothigattu et al. (2019) uses a hybrid strategy of two policies, one policy learning from crafted rewards that represent human expert knowledge and another pure RL policy which learns to maximize game rewards obtained from experience. A contextual bandit acts as a meta controller and learns which policy to choose for action selection in which situation [NBM+19]. In a paper by den Hengst et al. (2021), they utilize symbolic planning to calculate a potential for each state based on an abstract representation of the MDP with safety constraints [DHFLHvH22]. This potential is included in the learning process of an RL agent by shaping rewards. Actions leading to more progress towards a goal defined by the symbolic planner receive a higher reward. Gulati et al. (2021) built a system which decides on either using an RL agent, representing a fast system, or an MCTS planner, representing a slow system with better performance [GSR21]. The decision on which system to choose is made depending on how much time is available to the agent to select the optimal action. In a game of Pac-Man, they hand-craft this condition to be based on the proximity of enemies to the agent. Another paper uses a similar technique but uses a meta-controller which assesses the confidence of the fast system and chooses to run the slow system if there is enough time, and the confidence of the fast system is low [GCF+22]. Earlier research by Grounds & Kudenko (2005) combines an automated STRIPS planner with a Q-Learning agent [GK05],[FN71]. The planner defines a sequence of high-level operations. The RL agent learns to solve each operation on a low-level. A similar approach was pursued by ón Illanes et al. (2019) [óIYIM19]. They use a symbolic planner to create a high-level partially ordered plan out of options based on the temporal abstraction framework introduced by Sutton et al. (1999) [SPS99]. A hierarchical RL agent subsequently learns to follow those options on a lower level. Research by Leonetti et al. (2016) utilizes answer set programming as a type of symbolic planning to create a sub-selection of possible strategies from expert knowledge [LIS16]. The reinforcement learning agent is only allowed to select actions following these strategies which reduces the possible exploration space and therefore improves sample efficiency. An integrated approach of a planner and an RL agent comparable to some variants of model-based reinforcement learning was proposed by Anthony et al. (2017) [ATB17]. They use an MCTS planning agent as an expert policy and let it predict reward targets on which a policy neural network can be trained. The policy network learns to predict the action selection probabilities to be equal to the ratio of actions selected during the MCTS planning for each state. Additionally, the trained policy neural network influences selection probabilities during MCTS planning by adapting the upper confidence bounds of state values. After enough planning steps, a value network is trained on the targets from the MCTS planning too. The predicted values from the value network are used as values for each node during MCTS planning instead of the previously used upper confidence bounds. The trained hybrid model could significantly outperform human experts and pure RL algorithms in a game of Hex. Most approaches in the literature can either demonstrate an improved performance or a better sample efficiency when integrating expert knowledge via planning into the learning process of RL algorithms. A downside was shown when the constraints from expert knowledge prevented the RL agent to explore actions leading to an optimal policy. In the context of crop rotation planning, it is therefore important to lower the impact of expert knowledge in parts of the state space where the agent already gathered much experience.

2.4 Crop Rotation Planning

Crop rotation planning as a practice aids the farmer to select the right crop depending on the current situation on his land. Different types of crops have different soil condition preferences and modify soils towards optimal conditions for other crops when being cultivated. While many farms around the world decide to run monocultures of the financially most attractive crops in the region, this practice can diminish ground fertility over a long period of time and reduce yields due to sub-optimal conditions. Already in the early days of agriculture and farming practice, humans have realized that specific sequences of crops have higher yields than just planting single crops on the same plot repeatedly. Specific practices were already written down during the Roman Times and nowadays, there exist many example sequences and rule sets that farmers can orient on to plant crops and fruits with the highest yields in the right order [Fra05]. Those rule sets are typically learnt from experience and represent expert knowledge. Schöning et al (2023) state that "calculating an optimal crop rotation is a highly complex task, which depends on various factors, ranging from biological essentials to socioeconomic circumstances" [SWT23]. In conventional farming, disadvantages from monocultures can be balanced out by providing the soil with the necessary nutrients from fertilizers. This can however lead to nutrient leaching into ground water and affect the environment negatively. Therefore, the EU government and many other countries in the world regularly introduce further restrictive legislations to stop ground water pollution and promote organic farming without the excessive use of fertilizers with financial subsidies. Due to that, organic farming accounts for about 26% of all agricultural land in Austria, which has already surpassed the EU's "Farm-to-Fork" strategy to use 25% of the agricultural area organically by 2030 [Com20]. In the following section, it will be analysed what the characteristics of successful crop rotation management systems are and which rules and expert knowledge are found in the literature to determine fitting crop sequences. Additionally, different types of developed crop rotation optimization algorithms from the literature are described with their advantages and disadvantages.

2.4.1 Crop Rotation Management Systems

For many large farms in the world, crop rotation management systems play a huge role in supporting the farmer making decisions about managing his farm. Some tools only focus on the crop selection aspect while others include decision support for optimal cultivation dates and scheduling of supportive farm operations like ploughing, tilling, or applying fertilizer and pesticides. A large variety of crop rotation management tools can be found in the literature. They are based on different optimization or planning methods, different types of input data or were developed with different objectives. There exist sophisticated and expensive CRM tools with large-scale commercial farms as their target group. They are often part of entire software ecosystems for farmers which supply them with guidance and data-driven insights for the daily farm work [ZMBM20]. For small-scale farmers, those tools are often not affordable, and they rely their decision-making on experience or cheaper tools promoted by academia or governmental bodies. The adoption of those software tools is influenced by a variety of factors. The tools need to select crops suitable to the farm which also achieve high gross margins. They need to be easy to use for IT laymen and should not be too expensive to justify their purchase. In addition, they should be adaptable enough to match the farmers' habits [SWT23]. Many tools take this into account and predict appropriate crop sequences while considering external uncertainty like weather conditions and dynamic crop markets. Margins and yields are sometimes predicted by using average regional yields, costs and prices. In other tools, yields are simulated with a simulation software like DSSAT [HPB+19]. Crop succession effects are either predefined by experts, represented by rules and agronomic filters, or determined by aggregated indicators which typically adapt predicted yields to be higher or lower depending on the previously cultivated crops [DSG+12]. For optimization, many algorithms rely on mathematical modelling techniques like linear programming. Some older research utilizes constraint-based sequence generation while newer methods experiment with evolutionary algorithms or reinforcement learning to address the combinatorial problem. To enhance the ease of use, some tools opt for automatic data gathering and only need a location or ZIP code to extract regional data from other sources. Other tools necessitate the user to add data about soil characteristics, water content, fertilizer use and planting dates and allow pre-selecting crop options to restrain the search space. Most methods have net income as the only optimization target, others try to include more metrics like crop diversity, cost minimization and nutrient balance. As outputs, the tools deliver crop sequences for specified fields. Some tools also include a custom spatial allocation of crops onto the total farmable area. Visualizations about crop yield predictions, allocation maps and legislation compliance are sometimes part of the usable tool too [SWT23]. From a critical point of view, many models lack important features to be fully used as reliable and user-friendly crop management systems. In an older review by Dury et al. (2012), these issues are addressed in detail [DSG+12]. One drawback is that using net income as the sole optimization target does not reflect farmers' decisions as they usually consider several different objectives and do a multi-criteria assessment before deciding for a new crop to grow. Additionally, even when models consider uncertainty for factors like weather or market dynamics, those factors are represented as static probabilities which do not change over time. The authors estimate this to be unrealistic as market conditions usually change continuously and the climatic circumstances are subject to change as well with climate change elevating temperatures around the world and increasing probabilities for extreme weather events. Another downside of most tools is that they are specialized for single regions of the world and lack adaptability to other regions. Some do not offer open-source access to change the internal model settings, others are partially adaptable, but the necessary input data format is not found for other regions and would need a fundamental data preparation step before usage. The authors suggest improving future models by formalizing the cropping plan decision via an integrative modelling framework which considers all levels of temporal and spatial dimensions instead of relying on static and deterministic procedures. They propose to build a better understanding of crop production dynamics and include the typical farmer's decision process when constructing models.

2.4.2 Crop Rotation Rules

The effects from previous crops on succeeding cultivation and the needs of different plants to achieve optimal yields have been studied thoroughly. Nowadays, there exists a plethora of guidelines aggregated by researchers or governmental bodies to support farmers in their decision making. For this master thesis, those guidelines and discoveries can be used to guide decision making in an automated fashion. The following section is split into the topics *cover crops*, *nutrient effects*, *water & irrigation management*, *soil management* and *pest & weed management* and will be summarized into general crop rotation principles. The subtopics are not strictly isolated but rather show dependencies and synergies between different farming practices. There are many other factors influencing the optimization of farming operations which are not directly related to crop rotation planning and are therefore beyond the scope of this chapter. This aggregation of knowledge shall help in understanding why specific rules and constraints were included in the model to predict beneficial crop sequences.

Cover Crops

In many farming guidelines and support books, there are chapters focusing on the use of cover crops and legumes in organic farming and the benefits when being integrated into crop rotations [MVE+00]. In organic farming, they are often planted after cash crops to prepare the ground for the next crop. In colder climate conditions with less time to grow between harvest and winter, it is also common practice to interseed cover crops into already well matured cash crops when moisture levels are high enough and the right tools are available on the farm. Cover crops can also be grown for a whole year or even several years to let the soil recover after years of intensive cultivation. The cultivation of cover crops for one or multiple years is often combined with their usage as forage or green manure and can yield a profit. It was regularly shown in different research settings that cover crops improve yields of follow-up cash crops by 10-20% on average although this value can vary a lot depending on soil conditions and on the type of farming operations carried out on the field. In general, wetter regions had more constant responses than drier regions, especially when break crops replaced fallow in dry conditions [KCKL08]. The reasons for that are manifold. Cover crops prevent erosion of soils when compared to leaving the land bare by reducing fallow periods which can lead to soil degradation. They improve biological and physical properties of the soil by producing biopores and supply important nutrients like nitrogen, phosphorus and potassium in the form of soil organic matter. Additionally, they fixate nitrogen from previous crop residue, prevent nitrate leaching into ground water and reduce nitrous oxide from emitting into the atmosphere [RHB+16]. Many cover crops are competitive against different weed types and prevent their growth. Furthermore, they break pest cycles and can improve soil water availability [KCKL08],[BJL+11]. Often, a mix of different cover crops is sown to benefit from their diverse effects on the soil. Those mixtures are more flexible as less fertility-dependent plants in the mix grow better on low-fertility soils whereas plants with higher nitrogen needs will dominate in high-fertility soils. Cover crops can be divided into legumes, grass cover-crops and other types like the botanical family of brassicas or buckwheat. Some legumes like soybeans or common beans can be grown during summer and are killed by harsh winter conditions. Others like hairy vetch, winter field peas or crimson clover survive in colder climates to deliver a soil cover over the winter months. Some of them like red clover, white clover or alfalfa are used as perennial cover crops for longer periods without the cultivation of a cash crop. Grass cover crops are usually more resistant than legumes but offer less nutritional value when used as forage. They include annual cereals like rye or winter wheat or forage grasses like ryegrass and are more likely to increase soil organic matter when grown to full maturity. Brassicas like rapeseed, oilseed radish or mustard are planted in late fall and are especially good against pests in the soil like root pathogens or nematodes due to their biofumigation potential [MVE+00].

Nutrient Effects

Nitrogen, phosphorus, and potassium are minerals most commonly deficient in soils. There are other essential minerals necessary to grow plants, but they are less common to deplete in organic agriculture [MVE+00]. A good strategy is to regularly test mineral concentrations for those three elements in the soil and balance nutrient import and export to keep levels within an optimal range. In organic farming, nutrients can be added to the ground by planting crops with a nutrient surplus like legumes. Applying animal manure or silage is an alternative if available. Those amendments release their nutrients slowly during decomposition. They are usually not as effective as synthetic fertilizers but often leave the soil in a healthier state afterwards. Some care is recommended with animal composts as a regular fertilization as their use can lead to high phosphorous levels in the soil which can be detrimental to some plants [BJL+11]. Additionally, nutrients can be washed out of the soil and leach into ground water. This effect is stronger if nutrients are available in high concentrations due to a sequence of crops with a nutrient surplus or due to over-fertilization. An EU directive set limits to the amount of nitrogen per liter of ground water which resulted in national laws enforcing those limits with potential fines for crossing the thresholds [Com91]. To combat nutrient loss, an avoidance of longer periods of bare land is recommended. Cultivated plants can pick up nutrients through their roots and thereby fixate them in the ground [AE08]. Forage crops or cover crops can be turned into silage to apply this "green manure" full of nutrients at a later point in time. Alternatively, they can be tilled into the ground right before another crop is planted which can pick up the nutrients from the decomposing plant residue. The actual amount of nutrients a crop can offer to the next crop is determined by the amount of nitrogen the crop could pick up and fixate, by the maturity of the legume when it is incorporated into the ground, whether the whole plant or only the root system stays in the field and by the environmental conditions affecting the decomposition rate of the residue. The farmer should not wait too long before sowing a follow-up crop to avoid nutrient loss again. Maintaining a soil pH in an optimal range is an additional support to prevent nutrient leaching. When nitrogen levels are too high, risk for diseases increases as well and lodging can occur, which is the bending of grain crop stems near ground level and results in significant yield loss. A paper by Bachinger et al. (2003) offers a calculation help to determine the amount of nitrogen import and export for different soil qualities and crop types [BZ03]. They include nitrogen fixation, leaching and volatilization, degressive nitrogen mineralization of crop residues, manure effects and rainfall as influencing factors in their calculations. For cover crop mixtures, the authors adapt their calculation to include the share of legumes growing in the plot, as a higher share will lead to a higher amount of nitrogen fixation.

Water & Irrigation Management
Water & irrigation management is another building block in organic agriculture. In the US, average crop yields of irrigated farms are greater than yields on dryland farms by 118% for wheat and 30% for corn on average [MVE$^+$00]. This large difference stems from the crops' disparate water needs. Crops with high-water needs like rice, soybean or wheat are usually grown in regions with good access to water resources whereas crops like potatoes or corn are also favoured in drier regions. Farmers should additionally consider the water needs of cover crops, as a cover crop consuming a high amount of water might leave the soil too dry for the next crop to grow well. Using water from rainfall is typically the best-case scenario as an intensive artificial irrigation can lead to an accumulation of salts in the soil, is more expensive and necessitates a large source of water close to the farm.

Soil Management
Different soil types are classified by their ratio of sand, clay and silt. Heavier soils contain a higher share of clay and silt while lighter soils consist primarily out of sand. In heavier soils, soil aggregates can develop more easily which capture nutrients and store water. However, heavier soils face a higher risk of soil compaction when heavy machinery is operated on them. This can be detrimental to plant growth due to the roots of following crops not reaching the deeper layers of the soil. Sandier soils on the other hand have a much lower water capacity which can be problematic in drier regions or during periods of drought. To sustain a healthy soil, there are several management practices recommended. It is beneficial to enrich the soil with nutrients and soil organic matter through the incorporation of crop residues. This loosens up the ground by improving the microbiome which also results in the aggregation of soil particles, a better drainage and aeration. Some crops leave more organic matter in the soil than others. Annual row-crops which are transported away for sale after harvest leave the ground with less organic matter than before their cultivation. Moreover, they have a lower density of roots in the ground and during harvest, exposed parts of the soil are subject to traffic by agricultural machinery and to rainfall [BJL$^+$11]. Perennial legumes, grasses and legume-grass forage crops increase soil organic matter instead. A thorough recovery of soils is generally achieved with perennial cover crops like alfalfa. Generally, if only parts of the plant are harvested and the rest is incorporated into the ground, more soil organic matter is available afterwards. Additionally, soils on organic farms demonstrate higher soil quality represented by higher water storage capacity and more soil organic matter when compared to conventional farms. The main reason for that is the use of diverse crop rotations with cover crops and applications of organic compost [LD99]. Furthermore,

healthy soils should contain a minimal number of pests, be free of toxins and have sufficient depth before a compact soil layer [MVE+00]. Acidic soils should be treated with limestone to raise the pH to an optimal level. Conventional tillage and the use of heavy machinery should be limited as it speeds up soil degradation and compaction. A higher soil compaction will negatively influence aeration and drainage, can lead to higher pest damage, shallower plant roots and necessitates further tillage. To combat this, a regular integration of crops with deep rooting systems into crop rotations is recommended, as this loosens up the deeper layers of the soil and they move nutrients from the deeper layers into the plant's top growth [BJL+11]. Different types of organic matter have different decomposition rates which affects how quickly their nutrients become available to the plants. The decomposition rate is also affected by the carbon to nitrogen ratio of the amendment, the temperature and moisture levels and the soil type. While green manures and silage decompose faster, composts are more stable with a slower decomposition rate.

Pest & Weed Occurence

To avoid the occurrence of pests and weeds, the soil should stay healthy and nutrient levels should be kept in balance. As a general guideline, crops susceptible to pests should not be grown in the following years after pest occurrence to let the pests slowly die off. Pests which can infect a multitude of hosts can persist in the field through a variety of crops though, and a farmer must sometimes use other management techniques to prevent further spread and yield loss. This can include the utilization of organic pesticides or the beneficial biochemical effects of specific crops against several pest types. For example, some crops propagate beneficial fungi spore counts which support the avoidance of parasitic infections. Others are competitive when growing next to weeds and keep weed seed numbers low. A high biological activity in the soil also reduces weed seed numbers [MVE+00]. Some cover crops can behave as unwanted "volunteer" weeds in the next season if allowed to grow seeds. They need to be tilled into the soil before seeding or can be killed by harsh winter conditions when grown as cover crops over the colder months [BJL+11]. If weed infestation still occurs after those counter-measures, it can still be suppressed by growing a perennial crop or by mowing repeatedly to prevent seeding.

Crop Rotation Principles

Most of the knowledge from the previous chapters can be distilled into generalist crop rotation recommendations which is necessary to formalize constraints and rules for an automated model to follow. In this section, we present them as recommendations, the specific constraints for the hybrid model are formulated in the section "Experiment Design". Those recommendations can be found in an aggregated form in several books about crop rotation principles and organic farming [MVE+00],[BJL+11],[Nan16]:

- Nutrient levels should stay balanced by growing a mixture of crops importing nutrients into and crops exporting nutrients from the soil. Regular soil testing can help determine the need for further amendments.

- In organic crop rotations, a pattern of nutrient- and soil-building crops should be followed by crops with a high nutrient demand. After this step, less nutrient-demanding crops should be grown before returning to soil-building crops.
- Crops should be grown both in warm and cold seasons as this will reduce the time for weeds to grow.
- Winter cover crops should be grown before summer cash crops as they can be tilled in right before cultivating the new crop. Winter-killed cover crops should be grown before early-season crops as yield losses due to competition can be avoided.
- The same crop should not be grown on the same plot of land in consecutive years to avoid pest infestation. This also holds true for crops from the same botanical family.
- If a specific pest has already occurred, avoid potential host plants for the next years.
- At least one deep-rooted crop (e.g. alfalfa, sunflowers) should be grown during a rotation to gather nutrients and water from deeper soil layers and improve physical soil properties.
- Multiple crops leaving lots of residues should be grown during a crop rotation to promote the incorporation of organic matter into the soil.
- In drier regions or during periods without a lot of rainfall or artificial irrigation, not too many crops with a high-water demand should be grown in sequence to avoid follow-up crops lacking water to grow.
- When crops in a rotation sequence are mainly grown for soil-building or nutrient-building purposes (e.g. clover/rye) without any harvested goods being sold, the incurred costs should be lower than the additional gain in yield from following crops.
- In specific market conditions, breaking crop rotation rules can be beneficial but often creates more risk in the long run.

To support farmers besides recommending crop rotation practices, many books and recommendation sheets provided by governmental bodies give an overview about the characteristics and needs of different crops. Characteristics relevant for a crop rotation model found in those sources are the botanical family of the plant, the harvested part of the plant, net removal or gain of nitrogen, phosphorus and potassium, cold and drought tolerance, required soil pH, soil type preference and the degree of weed competition. Additionally, aggregated lists describing all hosts for common pests can be found in the literature [BJL^{+}11],[Fre03],[BZ07]. Another helpful tool is an indicator matrix which rates the suitability of pre-crop-post-crop sequences [Kol08b]. The suitability is usually a combination of previously mentioned effects including planting and harvesting dates, cross-infections of pests, weed suppression and nutrient balance effects. Those sheets often contain recommended breaks between growing the same crop on the same plot of land [JC19]. A recommendation document by Stein-Bachinger & Reckling (2013) gives insight into the effects of different crops on the soil, the erosion risk when growing a plant and characteristics for different types of manures [SBRG13].

2.4.3 Crop Yield Prediction & Simulation

Several papers in the literature describe methods to predict the yield of grown crops under specified conditions. Yield prediction is important under the context of crop rotation planning as only with precise yield predictions, the net profit of growing and selling a crop can be calculated accurately. As the focus of this work lies in developing a crop rotation optimization model, only the model types and the features used to predict crop yields are analysed in this section without going into too much detail. A review by van Klompenburg et al. (2020) states that the most popular types of input data to accurately predict yields can be grouped into the following feature groups: *Soil information* includes soil type, pH value, cation exchange capacity and the area of production. *Plant features* contain the type of plant, the crop density, the average growth and weight of the plant. *Weather conditions* are described by the amount of rainfall, the humidity, the intensity of solar radiation and the wind speed. Most papers assessed by the review use *nutrient concentrations* of nitrogen and potassium in the soil to predict yields. Concentrations of other elements like magnesium, phosphorus, zinc, sulphur, boron, calcium and manganese were used by some researchers but less frequently. Other popular features were the type and amount of irrigation and fertilization [VKKC20]. With regard to the models used, the review states the tendency of older research towards linear regression and classical machine learning models like random forest regression, support vector machines and gradient boosting tree regression [SDH$^+$18],[CUM19]. More complex models in use are artificial neural networks. Some methods use Long-Short-Term-Memory (LSTM) networks to include a temporal dimension in sequential predictions [SAC$^+$20],[WTD$^+$18],[JHZ$^+$20], others combine it with convolutional neural networks (CNN) to infer from visual data like satellite images [WHFY20],[YSH$^+$19],[TOdSMJZ20]. LSTMs can become especially useful to pick up temporal crop rotation effects from previous crops planted on the same plot. Although using image data can be helpful when predicting yields after cultivation, it will not help predictions made before a crop was planted. One approach uses Reinforcement Learning to predict crop yield but cannot demonstrate significant performance improvements when compared to an LSTM with the same input data [EV20].

2.4.4 Crop Rotation Optimization Methods

Crop rotation optimization methods focus on recommending the next crop to plant or to deliver complete crop rotation plans for farmers to follow. They pursue the same objective as this thesis project. In the following section, popular crop rotation models are presented in detail by outlining their methodical approach, by describing the data used to calibrate the model and by assessing their applicability for farmers.

Rule-Based Methods

As crop rotation planning has existed for a long time, there is also a lot of less recent research targeting this problem. Many older methods approach the problem by converting expert knowledge into rules readable by symbolic planning systems or constraints for mathematical modelling formalizations. Nevo et al. (1994) propose an integrated system which combines linear programming with a PROLOG-based constraint logic to do the planning. The logic system contains the expert knowledge which cannot be easily integrated into the linear programming problem. The system optimizes for net profit and adapt yields when logical rules are fulfilled or broken [NOP94]. Another popular approach was proposed by Dogliotti et al. (2003). Their model is primarily rule-based and filters all possible crop sequences to only keep those which fulfill all rules. Their ruleset contains timing constraints about optimal sowing and harvesting dates and intercrop periods. Additionally, they include maximum growing frequencies for crops in a rotation and the filter balances nutrient uptake and loss as well as the effects of crops on soil health. The tool then calculates the yields for all suitable crop rotations by using average values of gross margin for each fruit from the region and proposes the sequences with the highest expected yields to the farmer [DRVI03]. A similar approach was pursued by Bachinger & Zander (2007). In addition to filtering suitable crop rotations, they include other crop production activities like tillage, pre- & post-crop activities or manuring and combine them into operational sequences. Filters including those activities like "no manure after a legume-crop" are used as well. The model then predicts expected yields, postharvest NO_3-leaching, weed & pest risks, soil quality and the nutrient balance for each sequence. Yields are calculated with a polynomial trend function for different crops with parameters derived from yield data and expert knowledge. The calculation of nitrogen balances is the same as the one used in the recommendation sheet by Stein-Bachinger & Reckling (2013) [SBRG13]. Economic performance is calculated as a formula connecting costs, yield and prices. As inputs, it only uses the German soil index, the mean annual precipitation and the mean precipitation during the winter half of the year to make data input easy for farmers. The tool is able to select economically and agronomically sustainable crop rotations suitable to the conditions of farms in the region of the study [BZ07]. A newer publication by Reckling et al. (2016) formalizes this approach further and proposes a cropping system assessment framework which can generalize over regions when suitable input data is found [RHB+16]. They tested it by evaluating the effects of legumes on sustainability factors like nitrogen leaching and volatilization by comparing cropping systems with and without legume crops in different European regions.

Mathematical Modelling

In the past 20 years, there has been an abundance of research focusing on crop rotation optimization with mathematical modelling as its method. Mathematical modelling is a form of quantitative optimization which uses rules in form of equations and constraints to represent real-world problems. It is therefore similar to purely rule-based approaches in how it handles constraints. Instead of only looking for potential solutions satisfying all constraints, mathematical modelling can be used to maximize or minimize selected targets. A simple version of the method is proposed by dos Santos et al. (2010) [dSCAS10]. Several crops have known demands which need to be fulfilled. Crops have different planting and harvesting dates and they include constraints like the need for cover crops, fallow periods and minimum breaks between two cultivations of the same crop. While fulfilling all constraints, the solver optimizes for yield. The method used is a column generation heuristic. Further research by the same authors introduces the aspect of adjacency constraints when planning for multiple adjacent plots at the same time [dSMAS11]. Another method includes similar constraints but optimizes for minimal area of land use while fulfilling all constraints. It utilizes a heuristic to accelerate the optimization of possible crop sequences [APS15]. Detlefsen & Jensen (2007) show, that the issue of having an overly large number of possible crop sequences can be improved by using network flows. Although the method only makes use of a limited number of features, it shows that the model still performs relatively fast with an increasing number of potential crops to select [DJ07]. Research by Filippi et al. (2017) considers uncertainty in its linear programming model by including market price and yield variability. Instead of optimizing for a static gross margin, the model optimizes the conditional value at risk (CVaR) which is the average yield over the worst quantile of simulation runs. By doing so, the model can output crop sequences with the lowest amount of risk, which might be the preference of some farmers [FMS17]. Another linear programming model which is used in a complete crop rotation management system was developed by Pahmeyer et al. (2019). Besides typical agronomical constraints, they include regional legislation and automate data acquisition to improve user experience. Additionally, they allow manual value adjustment for users to customize the tool [PKB21]. A method proposed by Boyabathli et al. (2019) extends the concept of crop rotation management by letting the model propose an allocation of farmland. They incorporate revenue uncertainty into the model to be able to show the importance of growing more than one crop at the same time to mitigate risk [BNZ19]. A robust optimization method used by Fikry et al. (2021) must deal with uncertainty in market demand and water availability besides yield and price uncertainty. Market demand is a constraint they introduce which forces the model to plan in minimum amounts of some crops [FEG21].

Evolutionary Algorithms & Meta-Heuristics

Many of the presented mathematical modelling approaches rely on linear solvers or heuristics to find optimal crop sequences. However, the size of the search space can be too large if many different crops are available to be selected by the farmer. To address this issue, some researchers focus on using meta-heuristics like evolutionary algorithms. Pavón, Brunelli & von Lücken (2009) show that the crop rotation optimization problem can be solved with multi-objective evolutionary algorithms. Besides maximizing net profit, their model can minimize total investment cost and economic risk, maximize nutrient accumulation in soils and maximize crop diversification in subsequent seasons and on adjacent parcels. The model uses average yield data from a single farm for five different crops. The authors test three different evolutionary algorithms and compare the result with a linear programming method. The models output a pareto-optimal solution set so that the user can decide afterwards which objectives are more important [PBvL09]. An extension of this work from 2021 includes more objectives like the minimization of fallow periods and adds nutritional demands for each crop and soil treatment costs [vLAR21]. Both papers show that evolutionary algorithms outperform the linear programming method in terms of finding diverse and optimal solutions. Another study from 2011 shows similar results when using pareto-based evolutionary algorithms to optimize plant selection for greenhouses in Spain [MBG+11]. In 2014, Chetty & Adewumi used another meta-heuristic method to solve the crop planning problem. They employed swarm intelligence techniques like the firefly algorithm and glowworm swarm optimization to optimize revenue and minimize costs under restricted water usage [CA13].

Reinforcement Learning

While most of the previously mentioned papers include expert knowledge via constraints and optimize for yield or revenue, they often do not consider the effects of previous crops on the following crops' yields. Many methods have the static constraint to not grow the same crop continuously, some use a constraint to make sure that crops with high nutrient needs are grown after crops with a high nutrient import. While this static approach guarantees that solutions do not break constraints, the effect on yield if breaking a constraint is rarely included in the models. Particularly for soil building measures, short-term effects can rarely be observed but a healthy soil will improve yields in the long run [MVE+00]. A suitable approach to address this problem can be Reinforcement Learning. Due to the usage of value functions representing discounted future gains instead of only focusing on the next reward, Reinforcement Learning models learn to plan strategies which are also suitable across longer sequences of cultivated crops. The concept of Reinforcement Learning only showed up sparely in the context of crop rotation optimization. An already mentioned paper used it for yield prediction [EV20], another publication from 2004 mentions using it without going into too much methodical detail [OWC04]. Recent research by Fenz et. al (2023) applied the method to predict optimal crop rotation sequences [FNFW23]. The authors let a deep Q learning agent train on two different simulation environments. One environment is built upon information from a slightly modified crop succession indicator matrix by Kolbe (2006) [Kol06]. The second environment uses a crop suitability matrix based on predictions of the Normalized Difference Vegetation Index (NDVI) representing yield performance. The matrix was generated by Fenz et al. (2023) via an XGBoost regressor predicting NDVI values from clustered weather and soil information by training with NDVI values for the same fields from previous years in Austria [FNH+23]. The model adds constraints to the training by shaping rewards in the simulation environment. Average yields obtained from Austrian farms are used as a baseline. They are increased depending on the pre-crop effect obtained from the suitability matrix or penalized if the pre-crop-post-crop combination is not recommended. Additionally, crops increase or decrease a pre-defined nitrogen balance according to average values from the literature. If the nitrogen balance falls below zero, another penalty is added to the reward. The crop rotation sequences proposed by both models were validated by expert farmers. Many sequences were realistic with some unrealistic pre-crop-post-crop combinations in between.

2.4.5 Crop Rotation Prediction from Previous Crops

Besides optimizing crop selection for the next season's yield, there has also been research about the prediction of crop types planted on fields using the crop types of previous years as input. The research focuses on predicting complete cover maps for selected regions. Sources for input data are usually from Land Parcel Identification Systems which detect crop types via satellite images after the crops have already grown. They are typically published yearly by government agencies for different regions. A method by Schönhart et al. (2011) integrates this observed land use data with agronomic criteria from expert knowledge and applies the trained model to more than 500 farms in Austrian's region called "Mostviertel". The crop rotations are selected by having the highest agronomic score based on regionally specific crop rotation tables. The predictions are then evaluated by their area-weighted deviations from reality [SSS11]. Research from 2015 used historical cover maps to predict crop cover in France for the following years via Markov logic models [OID15]. For those models, historical crop rotations are used as weighted rules. If selecting a crop in the next year matches many of those rules, there is a high probability for the choice to be correct. Using this approach, an accuracy of about 60% was achieved to predict the next crop correctly. Newer research with the same goal made use of deep learning to predict the next states. An approach by Zhang et al. (2019) generates a binary encoding of crops planted in previous years for each pixel of the input data and predicts the next crop with a fully connected network using this input data [ZDLG19]. A more sophisticated approach by Yaramasu et al. (2020) utilizes a spatio-temporal autoencoder. A pretrained convolutional neural network is used to learn spatial patterns and encode the input data into a lower-dimensional space representing spatial information about the fields and crops used. The temporal information is added by encoding the spatial representations from several years into a single tensor of spatial-temporal features. Afterwards, a deconvoluting decoder architecture uses this information to predict the crop type for each pixel in an input-sized image again [YBP20]. Both of those newer approaches show better performance than the Markov logic model but still suffer from noise inside of the input data where some crops were not identified correctly from satellite images. A helpful idea was realized in research by Abernethy et al. (2023). They start by aggregating pixels from input data into custom-shaped polygons representing individual plots of land. By doing so, they remove noise and lower the amount of redundant training data. A gradient-boosting decision tree ensemble model is used afterwards to predict the next crop type for each field [ABB$^+$23]. These more modern techniques show promising results with 70-90% accuracy when predicting the next crop, depending on region and crop type. Higher results are typically achieved for regions with simpler crop rotations (e.g. corn-soybean-corn) and crops that are planted more commonly. Being able to predict the next crops on fields before the new season starts is helpful when estimating the total amount of produce being offered on the markets after the harvest. For individual fields, this type of crop selection might be able to predict suitable crops according to previous and common crop rotations. It however does not take into account field characteristics and dynamic factors like weather, costs and market prices. This might lead to the predicted crops being a sub-optimal choice. It could rather be helpful as a support model to estimate the supply of different crops in agricultural markets when deciding for a new crop to plant. Crop types with a high estimated supply might be unattractive to grow for an individual farmer. Conversely, a well trained crop rotation optimization model for individual farms might make the region-wide predictions more precise.

CHAPTER 3

Experiment Design

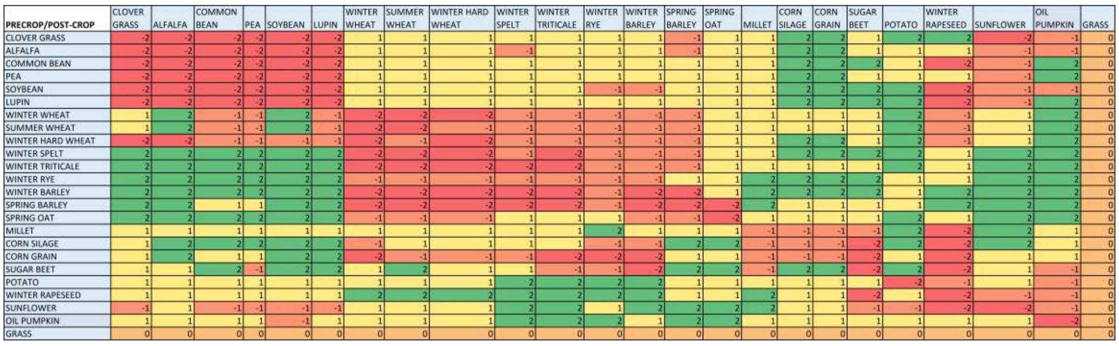
In this chapter, the evaluation approach to answer the research questions and to obtain significant results from the experiments run during this project is discussed. It contains a description of the simulation environment implemented to run experiments and an explanation from which data sources the assumptions and values in the environment stem. Furthermore, it is specified in detail which models are used as baselines in the experiments, which performance indicators are used as measures and further experiment details are stated to give the reader a thorough understanding of the work. Thus, and with further information in the appendix, a complete reproducibility of the results is possible.

3.1 Simulation Environment

A simulation environment to run experiments should be able to reflect all relevant effects from a real environment on crop yield and the farm's profits. This is necessary to make sure that agents performing greatly under simulation conditions can perform comparably well in real-life conditions when being used for decision support. To achieve this, the simulation environment used for all experiments is an extended more complex version of the simulation environment used by Fenz et al. (2023) [FNFW23]. Their environment relies on the previously selected crop as the key information to select the next crop. It calculates the total reward for a crop rotation by adding up the individual crops' static yields multiplied with positive multipliers and penalty factors. Positive multipliers and penalties are obtained by mapping the pre-crop-post-crop pairs to the Kolbe matrix, which gives each combination a rating regarding its effect on yield [Kol06]. Factors between 0.8 and 1.2 are in use. In addition, their environment tracks nitrogen levels starting from an initial value for each crop rotation sequence representing an episode and penalizes crop rotations heavily when the nitrogen levels go below 0. Furthermore, the maximum number of individual crops in a crop rotation are counted as well as minimum crop breaks for individual crops. It is also tracked if root crops are followed by non-root crops and if the maximum frequency of legume cultivation is surpassed. Crop rotations breaking any of those rules result in a high static negative penalty added onto the total yield.

The environment used for this project reuses or extends the concepts in the following way:

- **Crop suitability factor:** The suitability of pre- and post-crop combinations is again rated by a slightly updated Kolbe matrix penalizing non-suitable combinations with a yield factor of 0.8 and rewarding highly beneficial combinations with a factor of 1.2 [Kol06],[LfU08]. Less suitable combinations are penalized with a factor of 0.9, slightly beneficial combinations receive a factor of 1.1. The suitability matrix is depicted in Figure 3.1.



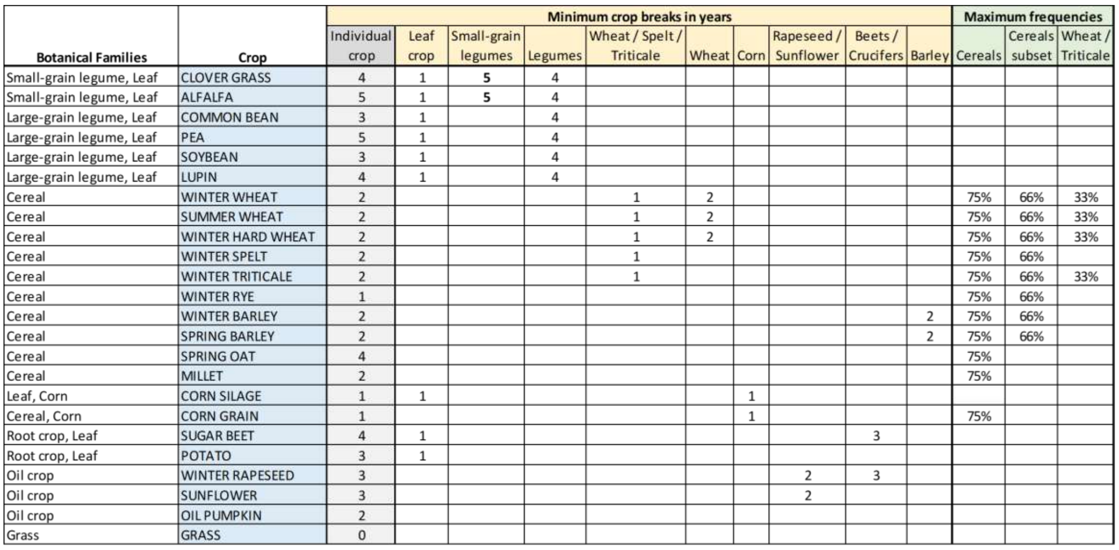
PRECROP/POST-CROP	CLOVER GRASS	ALFALFA	COMMON BEAN	PEA	SOYBEAN	LUPIN	WINTER WHEAT	SUMMER WHEAT	WINTER HARD WHEAT	WINTER SPELT	WINTER TRITICALE	WINTER RYE	WINTER BARLEY	SPRING BARLEY	SPRING OAT	MILLET	CORN SILAGE	CORN GRAIN	SUGAR BEET	POTATO	WINTER RAPESEED	SUNFLOWER	OIL PUMPKIN	GRASS
CLOVER GRASS	-2	-2	-2	-2	-2	-2	1	1	1	1	1	1	1	-1	1	1	2	2	1	2	2	-2	-1	0
ALFALFA	-2	-2	-2	-2	-2	-2	1	1	1	-1	1	1	1	-1	1	1	2	2	1	1	1	-1	-1	0
COMMON BEAN	-2	-2	-2	-2	-2	-2	1	1	1	1	1	1	1	1	1	1	2	2	2	1	-2	-1	2	0
PEA	-2	-2	-2	-2	-2	-2	1	1	1	1	1	1	1	1	1	1	2	2	1	1	1	-1	2	0
SOYBEAN	-2	-2	-2	-2	-2	-2	1	1	1	1	1	-1	-1	1	1	1	2	2	2	2	-2	-1	-1	0
LUPIN	-2	-2	-2	-2	-2	-2	1	1	1	1	1	1	1	1	1	1	2	2	2	2	-2	-1	2	0
WINTER WHEAT	1	2	-1	-1	2	-1	-2	-2	-2	-1	-1	-1	-1	-1	1	1	1	1	1	2	-1	1	2	0
SUMMER WHEAT	1	2	-1	-1	2	-1	-2	-2	-1	-1	-1	-1	-1	-1	1	1	1	1	1	2	-1	1	2	0
WINTER HARD WHEAT	-2	-2	-1	-1	-1	-1	-2	-1	-2	-1	-1	-1	-1	-1	1	1	2	2	1	2	-1	1	2	0
WINTER SPELT	2	2	2	2	2	2	-2	-2	-2	-1	-2	-1	-1	-1	1	1	2	2	2	2	1	2	2	0
WINTER TRITICALE	2	2	2	2	2	2	-2	-2	-2	-2	-2	-1	-1	-1	1	1	1	1	1	2	1	2	2	0
WINTER RYE	2	2	2	2	2	2	-1	-1	-1	-1	-1	-1	-1	1	1	2	2	2	2	1	1	2	2	0
WINTER BARLEY	2	2	2	2	2	2	-2	-2	-2	-2	-2	-1	-2	-2	1	2	2	2	2	1	2	2	2	0
SPRING BARLEY	2	2	1	1	2	2	-2	-2	-2	-2	-2	-1	-2	-2	-2	2	1	1	1	1	2	2	2	0
SPRING OAT	2	2	2	2	2	2	-1	-1	-1	1	1	1	-1	-1	-2	1	1	1	1	2	1	2	2	0
MILLET	1	1	1	1	1	1	1	1	1	1	1	2	1	1	1	-1	-1	-1	-1	2	-2	2	1	0
CORN SILAGE	1	2	2	2	2	2	-1	1	1	1	1	-1	-1	2	2	-1	-1	-1	-2	2	-2	2	1	0
CORN GRAIN	1	2	1	1	2	2	-2	-1	-1	-1	-2	-2	-2	1	1	-1	-1	-1	-2	1	-2	1	1	0
SUGAR BEET	1	1	2	-1	2	2	1	2	1	1	-1	-1	-2	2	2	-1	2	2	-2	2	-2	1	-1	0
POTATO	1	1	1	1	1	1	1	1	1	2	2	2	2	1	1	1	1	1	1	-2	-1	1	-1	0
WINTER RAPESEED	1	1	1	1	1	1	2	2	2	2	2	2	2	1	1	2	1	1	-2	1	-2	-1	-1	0
SUNFLOWER	-1	1	-1	-1	-1	-1	1	1	1	2	2	1	2	2	2	2	1	1	-1	-1	-2	-2	-1	0
OIL PUMPKIN	1	1	1	1	-1	1	1	1	1	2	2	2	1	2	2	1	1	1	1	1	1	1	-2	0
GRASS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3.1: Kolbe matrix rating each pre-crop-post-crop combination. The values multiplied with 10% are used as factors increasing or decreasing the yield obtained from the following crop. For example, a value of 2 leads to a 20% yield increase for the following crop.

- **Crop break factors:** Each crop and specific groups of crops are assigned minimum breaks from the literature. If those minimum breaks are violated by cultivating the same crop or a crop from one of those groups too early again, the expected yield is reduced through a factor representing the severity of the violation. The severity factor is calculated by a heuristic which follows the following logic examples:
  - When a crop is cultivated at time t but has already been cultivated at time $t-1$ with a minimum crop break of 2 years, the yield is penalized more severely than when the crop has only been cultivated at time $t-2$.
  - When a crop is cultivated at time t but has already been cultivated at time $t-1$ with a minimum crop break of 2 years, the yield is penalized more severely than when the minimum crop break would only be 1 year.

- When a crop is cultivated at time t but has already been cultivated twice at time $t-1$ and $t-2$ with a minimum crop break of 2 years, the yield is penalized more severely than when the crop has been cultivated only once at time $t-1$.

The complete heuristic can be found in the source code in the provided git repository [Wag23]. The underlying data about minimum crop breaks for different crop types was obtained from the literature [LB16],[SBRG13]. It is summarized in Figure 3.2.



		Minimum crop breaks in years										Maximum frequencies		
Botanical Families	Crop	Individual crop	Leaf crop	Small-grain legumes	Legumes	Wheat / Spelt / Triticale	Wheat	Corn	Rapeseed / Sunflower	Beets / Crucifers	Barley	Cereals	Cereals subset	Wheat / Triticale
Small-grain legume, Leaf	CLOVER GRASS	4	1	**5**	4									
Small-grain legume, Leaf	ALFALFA	5	1	**5**	4									
Large-grain legume, Leaf	COMMON BEAN	3	1		4									
Large-grain legume, Leaf	PEA	5	1		4									
Large-grain legume, Leaf	SOYBEAN	3	1		4									
Large-grain legume, Leaf	LUPIN	4	1		4									
Cereal	WINTER WHEAT	2				1	2					75%	66%	33%
Cereal	SUMMER WHEAT	2				1	2					75%	66%	33%
Cereal	WINTER HARD WHEAT	2				1	2					75%	66%	33%
Cereal	WINTER SPELT	2				1						75%	66%	
Cereal	WINTER TRITICALE	2				1						75%	66%	33%
Cereal	WINTER RYE	1										75%	66%	
Cereal	WINTER BARLEY	2									2	75%	66%	
Cereal	SPRING BARLEY	2									2	75%	66%	
Cereal	SPRING OAT	4										75%		
Cereal	MILLET	2										75%		
Leaf, Corn	CORN SILAGE	1	1					1						
Cereal, Corn	CORN GRAIN	1						1				75%		
Root crop, Leaf	SUGAR BEET	4	1							3				
Root crop, Leaf	POTATO	3	1											
Oil crop	WINTER RAPESEED	3							2	3				
Oil crop	SUNFLOWER	3							2					
Oil crop	OIL PUMPKIN	2												
Grass	GRASS	0												

Figure 3.2: Minimum crop breaks in years and maximum frequencies for all crops.

- **Crop maximum frequency factors:** Each crop and specific groups of crops are assigned maximum frequencies from the literature [NW15],[JC19]. The exact values can be found in Figure 3.2. If those maximum frequencies are violated by cultivating the same crop or a crop from one of those groups too often, the expected yield is reduced through a factor representing the severity of the violation. The severity factor f_{mf} is calculated as depicted in equation 3.1 for single crop maximum frequencies, where c is the count of the crop in the time window, c_{max} is the maximum count in the time window and w is the window length. It is only applied if a maximum frequency rule is violated in the current step:

$$f_{mf} = 1 - \left(\frac{c - c_{max}}{w} \right), \text{ if } c > c_{max} \tag{3.1}$$

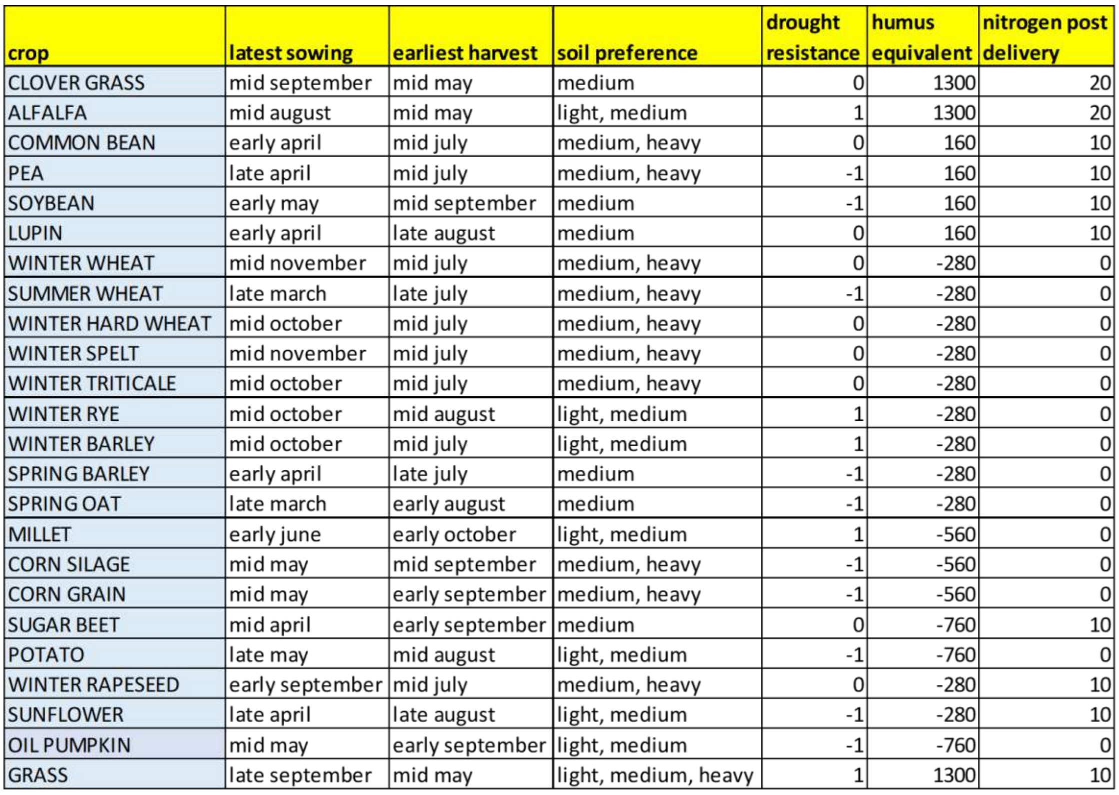
This makes sure that a stronger violation of the rule leads to a heavier penalization. A similar logic applies to maximum frequencies related to groups of crops and can be found in the source code [Wag23].

- **Nitrogen, phosphorus and potassium effects:** During the time crops grow, they consume minerals from the air and the soil to insert into their biological matter. Organic farmers selling off their harvested crops typically combat this draining effect by fertilizing the ground before each cultivation. The three main minerals necessary to sustain a fertile soil are nitrogen, phosphorus and potassium. In the simulation environment, it is assumed that missing amounts of those minerals are added via fertilization before each new crop cultivation. In addition, the postdelivery effect of high humus ratios and pre-crop effects on nitrogen levels are included in the simulation environment. As different crop types have different nutrient needs, it can be beneficial for the agent to select less nutrient-demanding crops or even nutrient-donating crops like legumes when fertilization costs are high. Another defined restriction is that nitrogen fertilization is limited to 170kg N per hectare and season, which is in accordance with Austrian agricultural laws [inf23]. If a crop needs more nitrogen than what is available in the soil after fertilization, the yield is limited by the available nitrogen amount. Information about the nutrient needs of different crop types was extracted from the literature and farming guideline sheets [KDO+22].

Furthermore, other environment effects representing real-life conditions for farms are introduced:

- **Crop sowing date factor:** Crop types have different common sowing dates in a specific region. While there is a certain time window for them to obtain optimal yields, missing that window will leave the crop with less time to grow sufficiently and can lead to lower yields. Additionally, all crop types have an earliest harvesting date as the crop would not be developed enough for harvest before that time. Due to this, some pre-crop-post-crop combinations automatically cause a yield reduction in the post-crop due to the second crop's latest sowing date being before the first crops earliest harvesting date. The cultivation and harvesting dates were obtained from several different farming guidelines and are set as static values in the simulation environment for each crop type [agr23],[get23],[Ste24]. For each week the second crop cannot be planted after its latest sowing date, the yield is reduced by 20% with a maximum reduction of 100% after 5 weeks. Albeit only using a heuristic, yield reduction values are in accordance with values from the literature when considering that yield reduction is only applied when the crop is cultivated after the latest possible date from the recommended range [RKE09]. The specific weeks for sowing and harvest are depicted in Figure 3.3.

- **Humus factor:** For different soil types, there is a minimum viable amount of humus to obtain optimal crop yields. Lower humus ratios leave the soil in a barren state. Some crop types have a humus building effect, others are detrimental to the amount of humus in the soil. It is therefore key for a farmer to balance out those effects and keep a fertile soil. The humus building and leaching effects for all



crop	latest sowing	earliest harvest	soil preference	drought resistance	humus equivalent	nitrogen post delivery
CLOVER GRASS	mid september	mid may	medium	0	1300	20
ALFALFA	mid august	mid may	light, medium	1	1300	20
COMMON BEAN	early april	mid july	medium, heavy	0	160	10
PEA	late april	mid july	medium, heavy	-1	160	10
SOYBEAN	early may	mid september	medium	-1	160	10
LUPIN	early april	late august	medium	0	160	10
WINTER WHEAT	mid november	mid july	medium, heavy	0	-280	0
SUMMER WHEAT	late march	late july	medium, heavy	-1	-280	0
WINTER HARD WHEAT	mid october	mid july	medium, heavy	0	-280	0
WINTER SPELT	mid november	mid july	medium, heavy	0	-280	0
WINTER TRITICALE	mid october	mid july	medium, heavy	0	-280	0
WINTER RYE	mid october	mid august	light, medium	1	-280	0
WINTER BARLEY	mid october	mid july	light, medium	1	-280	0
SPRING BARLEY	early april	late july	medium	-1	-280	0
SPRING OAT	late march	early august	medium	-1	-280	0
MILLET	early june	early october	light, medium	1	-560	0
CORN SILAGE	mid may	mid september	medium, heavy	-1	-560	0
CORN GRAIN	mid may	early september	medium, heavy	-1	-560	0
SUGAR BEET	mid april	early september	medium	0	-760	10
POTATO	late may	mid august	light, medium	-1	-760	0
WINTER RAPESEED	early september	mid july	medium, heavy	0	-280	10
SUNFLOWER	late april	late august	light, medium	-1	-280	10
OIL PUMPKIN	mid may	early september	light, medium	-1	-760	0
GRASS	late september	mid may	light, medium, heavy	1	1300	10

Figure 3.3: Crop options from the simulation environment with characteristics used during simulation.

crop types were obtained from the literature [Kol08a],[FBM20]. They are listed in Figure 3.3. Additionally, organic fertilization leads to an increase in soil organic matter and a higher humus ratio which is considered in the calculation of the new humus value after cultivation. Each created simulation environment starts with an initial humus ratio common for the respective ground type. If the humus ratio goes below a certain threshold, the cultivated crops' yields are negatively affected. The penalty is more severe the further the actual humus ratio deviates from the threshold. Detrimental effects on yield are in accordance with literature and are defined to be most severe at a 70% yield loss [Kol12],[Wie24].

- **Ground type and humidity:** Different crop types have different ground type and humidity needs. Some crops are more drought resistant than others; some crops need heavier soils than others [JC19]. The simulation environment can therefore be set to one of three ground types (light, medium, heavy) and one of two humidity settings (dry, humid). Crop yields are adapted depending on the crop's suitability to the environment's ground type. The experiments are run for all combinations of those settings to determine if some agents perform better in more restrictive conditions like a light soil und dry weather.

Additionally, the following effects are included to introduce **uncertainty** into the environment:

- Profit calculation: Instead of optimizing for yields, the environment returns profit per hectare calculated by multiplying price and yield and subtracting variable and fixed costs.
- To introduce uncertainty into the environment, yields, prices and costs are sampled from distributions. The same crop rotation in the same environment can therefore lead to different profits in each run. This increases complexity for the agents but represents real life conditions in a more realistic way.
- Yield uncertainty: Average yields for an environment are sampled from normal distributions representing the crop yields observed in Austria in the previous eight years [GHL+23]. Those average yields represent a plot of land's condition for crops to grow. During each step in an environment, actual yields are sampled again from a normal distribution using these average yields as a mean and a lower standard deviation than the one used to sample average yields. This leads to relatively stable yet partially uncertain yields during the crop rotation sequence.
- Price & cost uncertainty: Prices, variable cost factors and fixed costs are determined for each crop individually at the creation of an environment by sampling initial values from distributions representing those parameters for crops in Austria from the previous eight years [GHL+23]. For each step in the environment, prices, variable cost factors and fixed costs are simulated with the Geometric Brownian Motion process method which is commonly used as a simple method to simulate stock prices and growth scenarios [HB16]. Each price or cost movement is characterized by its percentage drift μ and its percentage volatility σ. Those parameters are calculated in the following way for a sequence of n values:

$$\mu = \frac{1}{n-1}\sum_{i=1}^{n-1}\delta_i, \text{ with } \delta_i = \frac{x_{i+1}-x_i}{x_i} \tag{3.2}$$

$$\sigma = \frac{1}{n-1}\sum_{i=1}^{n-1}(\delta_i - \mu) \tag{3.3}$$

The next simulated value S_{t+1} is calculated as depicted in formula 3.4:

$$S_{t+1} = S_t \cdot \exp\left(\left(\mu - \frac{\sigma^2}{2}\right) + \sigma N(0,1)\right) \tag{3.4}$$

Average prices, costs and profits from Austria which are used as baseline values are shown in Figure 3.4. Some crops have a higher profit potential but also represent higher costs and therefore a higher risk. If the yield is heavily reduced for those crops, it leads to a high financial loss for the farmer.

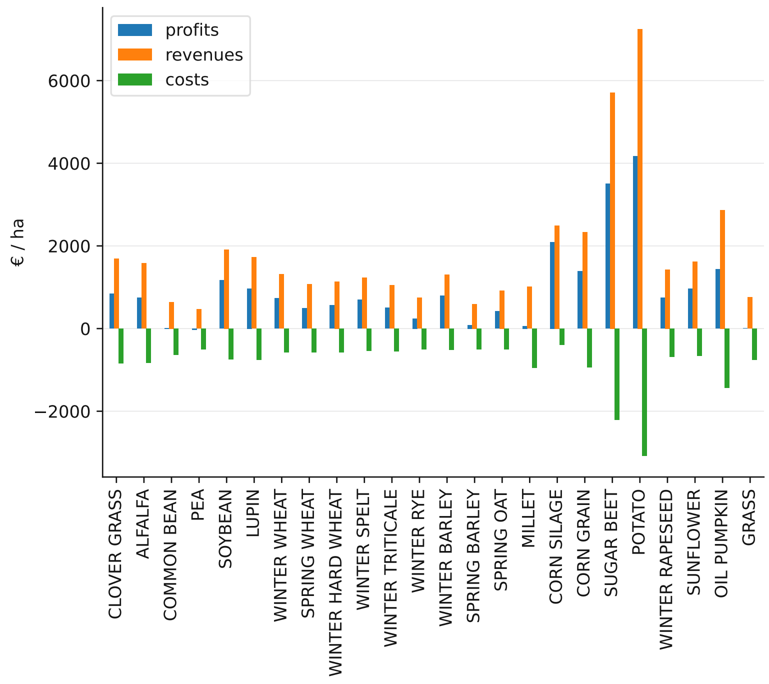
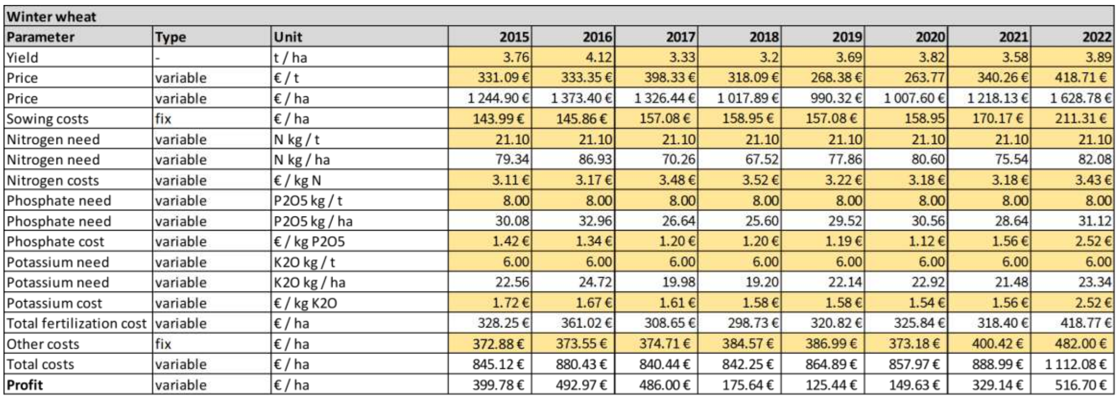


Figure 3.4: Average profits, revenues and costs per hectare for each crop option.

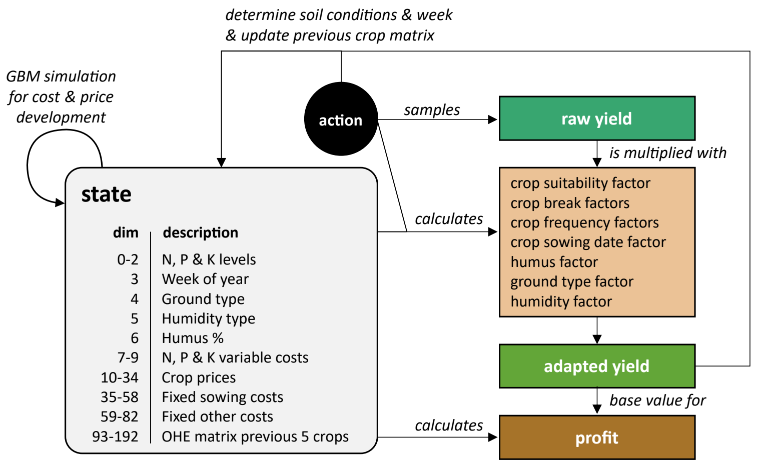
An example for price and cost information as well as nutrient needs for winter wheat in Austria between 2015 and 2022 can be examined in Figure 3.5. The data was gathered from the Austrian agricultural profit margin database and the Bavarian agricultural institute for each individual crop [GHL+23],[KDO+22]. The yellow fields are original data, the white fields exhibit values calculated from the original data.



Winter wheat										
Parameter	**Type**	**Unit**	**2015**	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**	**2022**
Yield	-	t / ha	3.76	4.12	3.33	3.2	3.69	3.82	3.58	3.89
Price	variable	€ / t	331.09 €	333.35 €	398.33 €	318.09 €	268.38 €	263.77	340.26 €	418.71 €
Price	variable	€ / ha	1 244.90 €	1 373.40 €	1 326.44 €	1 017.89 €	990.32 €	1 007.60 €	1 218.13 €	1 628.78 €
Sowing costs	fix	€ / ha	143.99 €	145.86 €	157.08 €	158.95 €	157.08 €	158.95	170.17 €	211.31 €
Nitrogen need	variable	N kg / t	21.10	21.10	21.10	21.10	21.10	21.10	21.10	21.10
Nitrogen need	variable	N kg / ha	79.34	86.93	70.26	67.52	77.86	80.60	75.54	82.08
Nitrogen costs	variable	€ / kg N	3.11 €	3.17 €	3.48 €	3.52 €	3.22 €	3.18 €	3.18 €	3.43 €
Phosphate need	variable	P2O5 kg / t	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00
Phosphate need	variable	P2O5 kg / ha	30.08	32.96	26.64	25.60	29.52	30.56	28.64	31.12
Phosphate cost	variable	€ / kg P2O5	1.42 €	1.34 €	1.20 €	1.20 €	1.19 €	1.12 €	1.56 €	2.52 €
Potassium need	variable	K2O kg / t	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00
Potassium need	variable	K2O kg / ha	22.56	24.72	19.98	19.20	22.14	22.92	21.48	23.34
Potassium cost	variable	€ / kg K2O	1.72 €	1.67 €	1.61 €	1.58 €	1.58 €	1.54 €	1.56 €	2.52 €
Total fertilization cost	variable	€ / ha	328.25 €	361.02 €	308.65 €	298.73 €	320.82 €	325.84 €	318.40 €	418.77 €
Other costs	fix	€ / ha	372.88 €	373.55 €	374.71 €	384.57 €	386.99 €	373.18 €	400.42 €	482.00 €
Total costs	variable	€ / ha	845.12 €	880.43 €	840.44 €	842.25 €	864.89 €	857.97 €	888.99 €	1 112.08 €
Profit	variable	€ / ha	399.78 €	492.97 €	486.00 €	175.64 €	125.44 €	149.63 €	329.14 €	516.70 €

Figure 3.5: Price & cost information, nutrient needs and average profit of winter wheat in Austria between 2015 and 2022 [GHL+23].

The complete simulation process is shown schematically in Figure 3.6. The step function receives an action as input. A raw yield value is sampled for the selected crop. Then, the factors representing soil conditions and potentially violated crop rules are calculated. The raw yield is multiplied with all factors to obtain the adapted yield. This yield value is multiplied with the current crop price and current costs are subtracted which results in the profit. After each step, the soil conditions are updated according to the selected action and adapted yield. Prices and costs are updated via a Geometric Brownian Motion process. The flattened matrix representing the last five cultivated crops is updated to include the newly selected crop. With each step from the environment, the agent receives a new reward represented by the profit and moves to a new state. State information contains nutrient levels, the current week in the year, the ground and humidity type, the humus ratio, variable costs for organic fertilizers, crop prices, fixed sowing and other costs and the one-hot encoded matrix representing the last five cultivated crops:



dim	description
0-2	N, P & K levels
3	Week of year
4	Ground type
5	Humidity type
6	Humus %
7-9	N, P & K variable costs
10-34	Crop prices
35-58	Fixed sowing costs
59-82	Fixed other costs
93-192	OHE matrix previous 5 crops

Figure 3.6: Single step of simulation environment.

The simulation environment allows the agent to select from 24 different crops as actions. The range of crops encompasses legumes like clover, alfalfa, peas or soybeans, cereals like wheat, rye, corn or barley, root crops like potatoes or sugar beet and other crops like sunflowers or rapeseed. The crops are selected to span a wide range of characteristics regarding their soil-building effects, water needs, cultivation dates and nutrient needs. Further information about the crop characteristics can be found in Figure 3.3. To stay within the scope of a master thesis project, it was decided to not include cover crops in the simulation. Although an addition of cover crops would lead to further optimization potential, it would also add more restrictions to the crop rotations as the cultivation of cover crops would reset the minimum breaks between individual crops and botanical families.

3.2 Model Specifications

In this chapter, the algorithms and features behind the proposed model are explained. The simulation environment is used as a training setting for a variety of different RL models including some or all of those features. The following sections describe the RL learning algorithms in use and how they are combined with a dynamics model for model-based RL. It is also explicitly stated how different types of crop rotation rules and constraints are depicted in an answer set program, how both systems are combined and why the complete model should perform better than the selection of baselines.

- **Deep Q Learning (DQN):** The DQN agent was already applied by Fenz et al. to a less complex simulation environment and is commonly used for discrete action spaces and large state spaces [FNFW23]. As a type of temporal difference learning, it predicts the action values of all possible actions with a neural network and learns from obtained rewards and its own estimations of action values in future states. The agent makes use of off-policy learning by slowly filling a replay buffer with experiences made in the simulation environment and learning the model weights during each step with a sampled batch of former experience from the buffer. The action selection is performed with an ϵ-greedy policy selecting the action with the highest action value in most cases and sometimes picking a random action with a chance ϵ.
- **Soft Actor-Critic (SAC):** The SAC algorithm is part of the policy gradient family of algorithms with a probabilistic action selection. It maximizes the entropy-regularized advantage function during learning and outputs a probability distribution over the actions, as described in section 2.2.c. The selected action is sampled from the distribution. As the original SAC algorithm was designed for a continuous action space, some adaptions needed to be made to the cost functions as proposed by the literature [Chr19].

The learning algorithms are further adapted by adding variations from the literature that proved to be beneficial towards sample efficiency and stability. One of those variations is to add **soft updates** [MKS+15]. Applying this variation, there is a distinction between local and target Q networks for the DQN algorithm and between local and target critic networks for the SAC algorithm. Local networks are trained directly by minimizing the loss function via gradient descent. The bootstrapped action values $Q_\phi(S_{t+1})$ for those updates are inferred from the target network. After each training step, the target network weights are updated to be a weighted average of the current target network weights and the updated local network weights. The weighting factor τ determines the strength of the update with $0 < \tau < 1$. Lower values for τ result in a slower alignment between local and target network weights and make training more stable. Another option to increase stability is to use **double Q-learning** instead of only having a single network to predict Q-values [VHGS16]. In the literature, it is shown that Q-value estimations from single estimators are commonly overestimated. By having two networks initialized and trained separately, the target Q-value can be the minimum of both network's estimations and the bias can be reduced. For this project, soft updates for both the DQN and the SAC agent are used as well as double-Q-learning for the SAC agent. As a result, two local and two target networks are used by the SAC agent. To improve sample efficiency during training, agents are not trained directly on the current experience but learn on a batch of previous experiences gathered in a **replay buffer** [Lin92]. Experiences saved in this buffer represent a state S_t, action a, the received reward r, the next state S_{t+1}, and a flag d signalling if the episode is finished or not. By doing so, the networks can make use of batch-wise gradient descent. While experiences can be sampled uniformly from the replay buffer in theory, there is a more efficient method to sample experience called **prioritized experience replay** [SQAS15]. Here, state transitions are not sampled from all experiences uniformly. Experienced transitions with a higher expected learning effect are sampled with a higher probability. The learning effect is represented by the temporal difference (TD) error for each sample calculated during the last training step it was used for. As bias is introduced into the learning process by not sampling uniformly, the technique is used together with importance sampling to correct for bias. Prioritized experience replay is affected by two parameters α and β. The sampling probabilities P are calculated as depicted in formula 3.5 with the priority π representing the last TD error obtained from using transition i as a target:

$$P\left(i\right) = \frac{p_i^\alpha}{\sum_k p_k^\alpha} \tag{3.5}$$

A higher value for α results in samples with a higher priority being sampled even more often. The importance sampling weights multiplied with the TD errors during training on the sampled batch are calculated as depicted in formula 3.6 with N being the current size of the replay buffer:

$$w_i = \left(\frac{1}{N \cdot P(i)}\right)^\beta \tag{3.6}$$

A higher value for β results in a larger bias correction with $\beta = 1$ being the maximum. Both the DQN and the SAC algorithm use prioritized experience replay to train on the replay buffer. The complete algorithm for the DQN agent representing a full training run is depicted as pseudo-code in algorithm 3.1:

The Q network parameters are updated with experience from the replay buffer as soon as there are enough samples to fill a batch with size n. The update function is depicted in algorithm 3.2:

Algorithm 3.1: DQN Learning Algorithm with Prioritized Replay and Soft Updates

```
1  Initialize $\epsilon$-decay schedule, soft update parameter $\tau \in [0, 1]$, $\beta$-annealing schedule,
     Q local network $Q^l_\phi$, Q target network $Q^t_\psi$, environment dataset $D_{env}$, initial
     state $S_t$
2  for $N$ epochs do
3      Initialize state $S_t$
4      Calculate $\epsilon$ from $\epsilon$-decay schedule
5      for $E$ steps do
6          Sample random number $e$ uniformly from range $[0, 1]$
7          if $e \leq \epsilon$ then
8              Select crop $a_t$ randomly from possible crops
9          end
10         else
11             Select crop $a_t$ according to the formula $a_t = \arg\max_a Q^l_\phi(S_t, a)$
12         end
13         Cultivate crop $a_t$ in simulation environment; Obtain profit $r$, next state
             $S_{t+1}$, and episode finish indicator $d$; Add full transition $(S_t, a_t, r, S_{t+1}, d)$
             to $D_{env}$ with priority $p = p_{max}$ where $p_{max}$ is the current highest priority
             in $D_{env}$
14         Update local Q network parameters $\phi$ on environment data $D_{env}$ with
             prioritized replay (Algorithm 3.2)
15         Soft-update Q target network parameters: $\psi = \tau\phi + (1 - \tau)\psi$
16         $S_t = S_{t+1}$
17     end
18 end
```

Algorithm 3.2: Update function of DQN learning algorithm with prioritized replay

```
1  Calculate $\beta$ from $\beta$-annealing schedule
2  Sample batch with size $n$ from $D_{env}$ with probability $P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$ for each
     sample $i$
3  Calculate sampling weight for each sample $i$: $w_i = [P(i) \cdot \text{len}(D_{env})]^{-\beta}$ and
     normalize the weights $\bar{w_i} = \frac{w_i}{\max_{j \in n} w_j}$
4  for each sample $i$ with data $(S_{t,i}, a_{t,i}, r_i, S_{t+1,i}, d_i)$ do
5      Calculate Q learning error:
         $\delta_i = r_i + (1 - d_i)\gamma \max_a Q^t_\phi(S_{t+1,i}, a) - Q^l_\phi(S_{t,i}, a_{t,i})$
6      Update the priority of the sample to be equal to the Q learning error
7  end
8  Calculate loss from Q learning errors: $L = \frac{1}{n} \sum_i^n (\delta_i \cdot \bar{w_i})^2$
9  Update local Q network parameters $\phi$ via gradient descent with loss $L$
```

Here, the parameters α and β are used as described in the previous section. The bootstrapped action value for the next state is only used for the target function if the finish indicator d_i of sample i is not 1. For the SAC agent, the algorithm is similar in many steps except for action selection and agent update depicted in algorithm 3.3:

Algorithm 3.3: SAC learning algorithm with prioritized replay and soft updates

```
1  Initialize temperature $\kappa$-decay schedule, soft update parameter $\tau \in [0,1]$,
     $\beta$-annealing schedule, local critic networks $Q^l_{\phi_1}$ and $Q^l_{\phi_2}$, target critic networks
     $Q^t_{\psi_1}$ and $Q^t_{\psi_2}$, actor network $\Gamma_\chi$, environment dataset $D_{env}$
2  for $N$ epochs do
3      Initialize state $S_t$
4      Calculate temperature $\kappa$ from $\kappa$-decay schedule
5      for $E$ steps do
6          Sample crop $a_t$ from probability distribution $\Gamma_\chi(S_t)$
7          Cultivate crop $a_t$ in simulation environment; Obtain profit $r$, next state
             $S_{t+1}$, and episode finish indicator $d$; Add full transition $(S_t, a_t, r, S_{t+1}, d)$
             to $D_{env}$ with priority $p = p_{\max}$, where $p_{\max}$ is the current highest priority
             in $D_{env}$
8          Update local critic network parameters $\phi_1$ and $\phi_2$ and actor network
             parameters $\chi$ on environment data $D_{env}$ with prioritized replay
             (Algorithm 3.4)
9          Soft-update target critic network parameters for both networks:
             $\psi_j = \tau\phi_j + (1-\tau)\psi_j$ for $j \in \{1,2\}$
10         $S_t = S_{t+1}$
11     end
12 end
```

The update function for the SAC agent is depicted in algorithm 3.4. It proved to be helpful to avoid importance sampling during the update step. The decay schedules are dependent on the number of episodes n_{run} set for the training run. They are defined in formula 3.7 with n being the current episode number and the other parameters set in the hyperparameter configuration. The schedule begins with a value close to ι_{start} and converges towards ι_{end} with an increasing n.

$$f(n, \iota_{end}, \iota_{start}, n_{run}) = \iota_{end} + (\iota_{start} - \iota_{end})e^{-\frac{n}{\iota_{decay}}} \tag{3.7}$$

The β-annealing schedule depicted in formula 3.8 is only dependent on hyperparameter ι_{rate}. The schedule slowly converges to 1 with an increasing episode number n:

$$f(n, \iota_{rate}) = 1 - e^{-\iota_{rate} \cdot n} \tag{3.8}$$

Algorithm 3.4: Update function of SAC learning algorithm with prioritized replay

```
1  Calculate $\beta$ from $\beta$-annealing schedule.
2  Sample batch with size $n$ from $D_{env}$ with the probability $P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$ for each sample $i$
3  Calculate sampling weight for each sample $i$: $w_i = [P(i) \cdot \mathrm{len}(D_{env})]^{-\beta}$ and normalize the weights $\bar{w_i} = \frac{w_i}{\max_{j \in n} w_j}$
4  for each sample $i$ with data $(S_{t,i}, a_{t,i}, r_i, S_{t+1,i}, d_i)$ do
5      Calculate soft state values: $\vartheta_i = \Gamma_\chi(S_{t+1,i})^T \left[\min\left(Q^t_{\psi_1}(S_{t+1,i}), Q^t_{\psi_2}(S_{t+1,i})\right) - \kappa \log \Gamma_\chi(S_{t+1,i})\right]$
6      Calculate critic learning errors: $\delta_{i,j} = r_i + (1 - d_i)\gamma\vartheta_i - Q^t_{\phi_j}(S_{t,i}, a_{t,i})$ for $j \in \{1, 2\}$
7      Calculate minimum delta: $\Delta_{i,\min} = \min(\delta_{i,1}, \delta_{i,2})$
8      Update the priority of the sample to be equal to the minimum delta: $p_i = \Delta_{i,\min}$
9  end
10 Calculate critic losses from critic learning errors: $L_{Q,j} = \frac{1}{n}\sum_i^n(\delta_{i,j})^2$ for $j \in \{1, 2\}$
11 Update local critic network parameters $\phi_j$ via gradient descent with losses $L_{Q,j}$ for $j \in \{1, 2\}$
12 Calculate actor loss: $L_\Gamma = \frac{1}{n}\sum_i^n \Gamma_\chi(S_{t+1,i})^T \left[\kappa \log \Gamma_\chi(S_{t,i}) - \min\left(Q^t_{\phi_1}(S_{t,i}), Q^t_{\phi_2}(S_{t,i})\right)\right]$
13 Update actor network parameters $\chi$ via gradient descent with loss $L_\Gamma$
```

3.2.1 Model-based Reinforcement Learning

An option to improve sample efficiency is to use model-based reinforcement learning. The theory behind model-based RL is, that the agent has access to additional experience simulated by a dynamics model trained to predict the profit and the next state from an environment state and the crop selected for cultivation. The method implemented for this project is called Model-based Policy Optimization (MBPO) and was introduced by Janner et al. in 2019 [JFZL19]. Some adaptions are made to fit the context but the main features of using an ensemble of neural networks to predict environment dynamics and the limitation of rollout length are kept. Instead of parametrizing a Gaussian distribution for each state, an ensemble of Bayesian neural networks (BNNs) is used to predict the target values directly. This keeps the original idea of probabilistic modelling while allowing a larger number of input dimensions than the original MBPO algorithm and was already proposed in 2016 by Gal et al. to improve other Bayesian approaches in model-based RL [GMR16]. The pseudocode for the model-based training is aligned with the algorithm developed by Janner et al. and shown in algorithm 3.5 for the SAC agent:

Algorithm 3.5: Model-Based Policy optimization with Bayesian Neural Networks and a SAC learning algorithm

```
1  Initialize temperature $\kappa$-decay schedule, soft update parameter $\tau \in [0, 1]$,
     $\beta$-annealing schedule, local critic networks $Q^l_{\phi_1}$ and $Q^l_{\phi_2}$, target critic networks
     $Q^t_{\psi_1}$ and $Q^t_{\psi_2}$, actor network $\Gamma_\chi$, environment dataset $D_{env}$, ensemble of $m_{dyn}$
     predictive models $\Lambda_{l,\omega_l}$, model dataset $D_{model}$
2  Create an experience dataset $D_{pretrain}$ filled with $M$ transitions of experience
     obtained from randomly selecting crops filtered by crop rotation rules to
     cultivate on the simulation environment
3  Pretrain each model $\Lambda_{l,\omega_l}$ on different sampled batches from $D_{pretrain}$ for
     $N_{pretrain}$ steps via gradient descent
4  for $N$ epochs do
5      Initialize state $S_t$
6      Calculate $\kappa$ from $\kappa$-decay schedule
7      for $E$ steps do
8          Sample crop $a_t$ from probability distribution $\Gamma_\chi(S_t)$
9          Cultivate crop $a_t$ in simulation environment; Obtain profit $r$, next state
             $S_{t+1}$, and episode finish indicator $d$; Add full transition $(S_t, a_t, r, S_{t+1}, d)$
             to $D_{env}$ with priority $p = p_{\max}$, where $p_{\max}$ is the current highest priority
             in $D_{env}$
10         Update local critic network parameters $\phi_1$ and $\phi_2$ and actor network
             parameters $\chi$ on environment data $D_{env}$ with prioritized replay
11         Soft-update target critic network parameters: $\psi_j = \tau\phi_j + (1 - \tau)\psi_j$ for
             $j \in \{1, 2\}$
12         Train each model $\Lambda_{l,\omega_l}$ on different sampled batches from $D_{env}$ for $N$ steps
             via gradient descent
13         for $M$ model rollouts do
14             Sample $s_t$ uniformly from $D_{env}$
15             Perform $k$-step model rollout starting from $s_t$ using crop selection from
                 actor network $\Gamma_\chi$ and a randomly selected dynamics model $\Lambda_{l,\omega_l}$ for
                 each step; add the obtained experience to $D_{model}$
16         end
17         for $X$ training steps do
18             Update local critic network parameters $\phi_1$ and $\phi_2$ and actor network
                 parameters $\chi$ on model data $D_{model}$ without prioritized replay
19             Soft-update target critic network parameters: $\psi_j = \tau\phi_j + (1 - \tau)\psi_j$ for
                 $j \in \{1, 2\}$
20         end
21         $S_t = S_{t+1}$
22     end
23 end
```

The update function of the SAC agent when training on model data does not use prioritized replay and works similarly otherwise. The variant with the DQN agent differs in step 8 and 15 where action-selection is ϵ-greedy and in step 10 and 18 where only the action-value function is updated instead of actor and critic. By using an ensemble of dynamics models and training them with different bootstrapped samples from D_{env}, epistemic uncertainty can be modelled. The aleatoric uncertainty is represented by the probabilistic nature of BNNs that are used as dynamics models. Although agents training on actual experience make use of prioritized experience replay, experience from the model memory D_{model} is sampled uniformly to make sure that high loss samples are not prioritized. This avoids that inferred transitions far from reality are not weighted more heavily due to the high divergence between the agent prediction and the simulated target. As it is realistic to assume that experience from the target field and neighbouring fields as well as characteristics and knowledge from the literature were already gathered before training an agent on a new field, the dynamics models used during the experiments are pretrained on experience from 500 10-step episodes simulated on the same environment with random actions filtered by crop rotation rules. Those experiences represent previous years of cultivation from the region without the use of a decision support system but farmer's knowledge about crop rotation rules from the literature.

3.2.2 Symbolic Planning

Another feature of the proposed hybrid model is to use symbolic planning to reduce the number of viable crop options the agent can select from. The planning step used for the models could therefore also be seen as a filtering mechanism. The filter determines from the current state which crops would be suitable to cultivate without breaking any crop rotation rules or other restrictions. The filtering query is written in the syntax of the answer set solving framework *clingo* [GKK+11]. It checks if any minimum crop breaks or maximum frequencies would be violated by cultivating a crop, if there is enough time to sow the crop and if the humus ratio in the soil after cultivation is higher than the minimum threshold. As a result, all viable crops are returned. The following examples in table 3.1 show how information from crop rotation rules is depicted in the *clingo* syntax. Information about crop properties, maximum frequencies and minimum breaks is defined as facts. The current state is encoded into facts as well. The filtering rules are defined as rules. The actual code can be found in the accompanying GitHub repository [Wag23].

During action selection, the possible actions for the agent are filtered by this method with a chance represented by parameter λ. λ decreases over time during training to allow more experienced agents to perform more exploration. If no possible actions are found after filtering, the agent selects an action without the filter. Aside from the action selection step, the filtering is used during the agent update step. For the DQN agent, the target value for the update is the sum of the reward and the discounted action-value of the greedy action for the next state. Instead of simply using the greedy action, the *greedy action after filtering* is used to calculate the action-value for the next state. Algorithm 3.6 describes a training run for hybrid DQN agents with symbolic planning. The difference

Table 3.1: Description of *clingo* syntax used to represent crop rotation rules.

Type	Natural language description	Clingo syntax
General Facts		
Static Fact	There are 24 different crops to select from.	action(0..23).
Static Fact	The last 5 years are defined as timestamps.	previous_time(-5..-1).
Static Fact	The humus equivalent property of crop 0 (clover grass) has value 1300.	property(0, humus_equivalent, 1300).
Static Fact	Crop 0 has a minimum crop break of 2 years.	cropbreak(0, single, -1). cropbreak(0, single, -2).
Static Fact	The botanical groups for maximum frequencies (MF) are “ge”, “geohnemaishaferhirse”, “weizentriticale”.	mfgroup((ge;geohnemaishaferhirse;weizentriticale)).
Static Fact	Crop 10 is in the MF groups “ge”, “geohnemaishaferhirse” and “weizentriticale”.	crop_mfgroups(10, (ge;geohnemaishaferhirse;weizentriticale)).
Static Fact	MF Group “weizentriticale” contains crops 6, 7, 8 & 10. The range for the MF group “weizentriticale” spans the last 3 years.	mf_group_block(weizentriticale, (6;7;8;10), -3..-1).
Static Fact	The maximum number of crops from the MF group “weizentriticale” cultivated in the time range of the MF group is 1.	mf_group_block_max_count(weizentriticale, 1).
Static Fact	Crop 0 is in the crop break (AP) groups “blatt”,”fl”,”l”.	apgroups(0, (blatt;fl;l)).
Static Fact	AP Group “fl” contains crops 0 & 1. The minimum break for the AP group “fl” is the last 5 years.	ap_group_block(fl, (0;1), -5..-1).
Facts from the current state		
Dynamic Fact	The current week is week 20.	week_info(20).
Dynamic Fact	The ground type is 0 (medium).	ground_type_info(0).
Dynamic Fact	The humidity type is 0 (dry).	drywet_info(0).
Dynamic Fact	The current humus equivalent soil ratio is 2570. The humus equivalent minimum threshold is 2000.	humus_info(2570,2000).
Dynamic Fact	Crop 0 was cultivated three years ago.	previous_actions_info(-3,0)
Minimum crop break filtering		
Static Rule	If a crop A was planted during the time range of its individual crop break, it is “blocked_by_previous”.	blocked_by_previous(A) :- action(A), previous_actions_info(X,A), cropbreak(A, single, X).
Static Rule	If a crop A is in any AP group APG and was planted during the time range of the crop break of APG, it is “blocked_by_ap_group”.	blocked_by_ap_group(A) :- apgroups(A,APG), previous_actions_info(X,Y), ap_group_block(APG, Y, X).
Maximum frequency group filtering		
Static Rule	If a crop A is in the MF group MFG and was planted at time Y within the time range of MFG, the group block of MFG at time Y is active.	mf_group_block_active(MFG, Y) :- mf_group_block(MFG, A, Y), previous_actions_info(Y,A).
Static Rule	The count of MF group MFG is equal to the number of active group blocks of MFG.	count_mf_group(MFG, C) :- C = #count {Y : mf_group_block_active(MFG, Y)}, mfgroup(MFG).
Static Rule	Any crop A is blocked by an MF group MFG if the count C of the MF group is equal or higher than the maximum count of the MF group. Any crop A is generally “blocked_by_mf_group” if it is blocked by at least one MF group.	blocked_by_mf_group(A) :- action(A), crop_mfgroups(A, MFG), count_mf_group(MFG, C), mf_group_block_max_count(MFG, MC), C+1 > MC.
Property filters		
Static Rule	Any crop A is “blocked_by_week” if it is a winter crop and its latest sowing week is equal or smaller than the current week.	blocked_by_week(A) :- action(A), property(A, latest_sowing, LS), week_info(W), W > LS-1, property(A, summercrop, SC), SC == 0.
Static Rule	Any crop A is “blocked_by_humus” if its humus equivalent HE is smaller than 0 and the sum of HE and the current humus equivalent ratio HL is smaller or equal than the humus equivalent minimum threshold.	blocked_by_humus(A) :- action(A), property(A, humus_equivalent, HE), humus_info(HL, HML), HL+HE <= HML, HE<0.
Action filtering rule		
Static Rule	Any crop A is a solution candidate if it is not “blocked_by_previous”, not “blocked_by_ap_group”, not “blocked_by_mf_group”, not “blocked_by_week” and not “blocked_by_humus”.	immediate_candidate(A) :- action(A), not blocked_by_previous(A), not blocked_by_ap_group(A), not blocked_by_mf_group(A), not blocked_by_week(A), not blocked_by_humus(A).

to DQN agents without symbolic planning is marked in bold.

Algorithm 3.6: DQN Learning Algorithm with Symbolic Planning, Prioritized Replay and Soft Updates

```
1  Initialize $\epsilon$-decay schedule, $\lambda$-decay schedule; soft update parameter $\tau \in [0,1]$,
     $\beta$-annealing schedule; Q local network $Q^l_\phi$, Q target network $Q^t_\psi$, environment
     dataset $D_{env}$
2  for $N$ epochs do
3      Initialize state $S_t$
4      Calculate $\epsilon$ from $\epsilon$-decay schedule; Calculate $\lambda$ from $\lambda$-decay schedule
5      for $E$ steps do
6          Sample random numbers $e$ and $l$ uniformly from range $[0,1]$
7          if $l \leq \lambda$ then
8              $\mathcal{A}' = \mathcal{A}$, with $\mathcal{A}$ as the action space spanning all possible crops
9          end
10         else
11             Filter action space $\mathcal{A}$ with the symbolic planner to receive a
                 subset $\mathcal{A}'$.
12         end
13         if $e \leq \epsilon$ then
14             Select crop $a_t$ randomly from $\mathcal{A}'$.
15         end
16         else
17             Select crop $a_t$ according to the formula $a_t = \arg\max_{a \in \mathcal{A}'} Q^l_\phi(S_t, a)$
18         end
19         Cultivate crop $a_t$ in simulation environment; Obtain profit $r$, next state
             $S_{t+1}$ and episode finish indicator $d$; Add full transition $(S_t, a_t, r, S_{t+1}, d)$
             to $D_{env}$ with priority $p_i = p_{max}$ where $p_{max}$ is the current highest
             priority in $D_{env}$
20         Update local Q network parameters $\phi$ on environment data $D_{env}$
             with prioritized replay (Algorithm 3.7)
21         Soft-update Q target network parameters $\psi = \tau\phi + (1-\tau)\psi$
22         $S_t = S_{t+1}$
23     end
24 end
```

The update function is adapted as shown in algorithm 3.7.

A similar approach is carried out for the SAC agent. Action selection is restricted by the filter too. Actions that are filtered out by the answer set solver are assigned probabilities of 0 in the modified policy. During critic updates, the soft state-values are calculated with the adapted policy $\pi'(S_{t+1,i})$ as shown in algorithm 3.8.

Algorithm 3.7: Update function of DQN learning algorithm with symbolic planning and prioritized replay

```
1  Calculate $\beta$ from $\beta$-annealing schedule
2  Sample batch with size $n$ from $D_{env}$ with probability $P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$ for each
     sample $i$
3  Calculate sampling weight for each sample $i$: $w_i = [P(i) \cdot \text{len}(D_{env})]^{-\beta}$ and
     normalize the weights $\bar{w}_i = \frac{w_i}{\max_{j \in n} w_j}$
4  for each sample $i$ with data $(S_{t,i}, a_{t,i}, r_i, S_{t+1,i}, d_i)$ do
5      Filter action space $\mathcal{A}$ with the symbolic planner based on next state
         $S_{t+1,i}$ to receive a subset $\mathcal{A}'$
6      Calculate Q learning error:
         $\delta_i = r_i + (1 - d_i)\gamma \max_{\mathbf{a} \in \mathcal{A}'} \mathbf{Q}^{\mathbf{t}}_{\phi}(\mathbf{S_{t+1,i}}, \mathbf{a}) - Q^l_\phi(S_{t,i}, a_{t,i})$
7      Update the priority of the sample to be equal to the Q learning error
8  end
9  Calculate loss from Q learning errors: $L = \frac{1}{n} \sum_i^n (\delta_i \cdot \bar{w}_i)^2$
10 Update local Q network parameters $\phi$ via gradient descent with loss $L$
```

Algorithm 3.8: Update function of SAC learning algorithm with symbolic planning and prioritized replay

```
1  Calculate $\beta$ from $\beta$-annealing schedule.
2  Sample batch with size $n$ from $D_{env}$ with the probability $P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$ for each
     sample $i$
3  Calculate sampling weight for each sample $i$: $w_i = [P(i) \cdot \text{len}(D_{env})]^{-\beta}$ and
     normalize the weights $\bar{w}_i = \frac{w_i}{\max_{j \in n} w_j}$
4  for each sample $i$ with data $(S_{t,i}, a_{t,i}, r_i, S_{t+1,i}, d_i)$ do
5      Filter action space $\mathcal{A}$ with the symbolic planner based on next state
         $S_{t+1,i}$ to receive a subset $\mathcal{A}'$
6      Calculate crop probabilities for next state $\pi(S_{t+1,i}) = \Gamma_\chi(S_{t+1,i})$; set
         the probability of each crop not in $\mathcal{A}'$ to 0 to obtain $\pi'(S_{t+1,i})$;
         normalize $\pi'(S_{t+1,i}) = \frac{\pi'(S_{t+1,i})}{\sum_k^m \pi'_k(S_{t+1,i})}$ with $m$ being the number of
         crops
7      Calculate soft state values:
         $\vartheta_i = \pi'(\mathbf{S_{t+1,i}})^T \left[\min\left(Q^t_{\psi_1}(S_{t+1,i}), Q^t_{\psi_2}(S_{t+1,i})\right) - \kappa \log\left(\pi'(\mathbf{S_{t+1,i}})\right)\right]$
8      Calculate critic learning errors: $\delta_{i,j} = r_i + (1 - d_i)\gamma\vartheta_i - Q^t_{\phi_j}(S_{t+1,i}, a_{t,i})$ for
         $j \in \{1, 2\}$
9      Calculate minimum delta: $\Delta_{i,\min} = \min(\delta_{i,1}, \delta_{i,2})$
10     Update the priority of the sample to be equal to the minimum delta:
         $p_i = \Delta_{i,\min}$
11 end
12 Calculate critic losses from critic learning errors: $L_{Q,j} = \frac{1}{n} \sum_i^n (\delta_{i,j})^2$ for $j \in \{1, 2\}$
13 Update local critic network parameters $\phi_j$ via gradient descent with losses $L_{Q,j}$
     for $j \in \{1, 2\}$
14 Calculate actor loss:
     $L_\Gamma = \frac{1}{n} \sum_i^n \Gamma_\chi(S_{t+1,i})^T \left[\kappa \log \Gamma_\chi(S_{t,i}) - \min\left(Q^t_{\phi_1}(S_{t,i}), Q^t_{\phi_2}(S_{t,i})\right)\right]$
15 Update actor network parameters $\chi$ via gradient descent with loss $L_\Gamma$
```

3.2.3 Neighbour experiments

As it would be common to exchange information about suitable crops with your neighbours or even share experience and data about previously cultivated crops, it is examined if an agent can learn from experience obtained from neighbouring fields to improve sample efficiency further. Another use case would be a farmer with more than one plot of land. To evaluate this research question, it is assumed that neighbouring plots of land have similar soil and humidity conditions and the average yields obtained from cultivated crops are alike as well. To simulate differences between different plots of land, the average yields of the neighbouring fields are adapted as depicted in formula 3.9:

$$\mu_{nb} = N\left(\mu_{orig}, (1-\rho_{nb}) \cdot \sigma_{orig}\right), \; 0.5 < \rho < 0.99 \tag{3.9}$$

This results in sampled yields from neighbouring environments with a higher similarity ρ to the target environment to be closer in their distribution to the target environment's yield distribution than neighbouring environments with a lower similarity. Experience is generated by creating neighbour environments with random similarities to the target environment and then simulating 10-year long crop sequences on those neighbouring environments until a replay buffer with 5000 experienced transitions is filled. Actions are selected randomly from the action space filtered by the answer set solver to represent action selection by a farmer with knowledge in crop rotation rules. The obtained transitions are saved into a prioritized replay buffer with the similarity ρ_{nb} as the priority. This leads to experience from more similar neighbour environments to be sampled more often and to be weighted more heavily in the calculation of the actor and critic losses. Agent updates from neighbour experience are carried out after each normal training step to stabilize learning. As it is also realistic to assume that experience from neighbouring fields was already gathered before training an agent on a new field, the experience is used to pretrain the agents too before they start training on the actual environment. The full training run algorithm in pseudo-code is depicted in algorithm 3.9, the neighbour experience filling step in algorithm 3.10 and the update function from the neighbour experience buffer in algorithm 3.11.

Algorithm 3.9: SAC learning algorithm with prioritized replay, soft updates, and neighbor experience

```
1  Initialize temperature $\kappa$-decay schedule, soft update parameter $\tau \in [0, 1]$,
     $\beta$-annealing schedule; local critic networks $Q^l_{\phi_1}$ and $Q^l_{\phi_2}$, target critic networks
     $Q^t_{\psi_1}$ and $Q^t_{\psi_2}$, actor network $\Gamma_\chi$, environment dataset $D_{env}$, neighbor experience
     dataset $D_{neighbor}$
2  **Fill neighbor experience buffer** $D_{neighbor}$ **with experience** (Algorithm 3.10)
3  **for** $\mathbf{N_{nb\_training}}$ training steps **do**
4      **Update local critic network parameters** $\phi_1$ **and** $\phi_2$ **and actor
         network parameters** $\chi$ **on neighbor experience data** $D_{neighbor}$ **with
         prioritized replay**
5  **end**
6  **for** $N$ epochs **do**
7      Initialize state $S_t$
8      Calculate $\kappa$ from $\kappa$-decay schedule
9      **for** $E$ steps **do**
10         Sample crop $a_t$ from probability distribution $\Gamma_\chi(S_t)$
11         Cultivate crop $a_t$ in simulation environment; Obtain profit $r$, next state
             $S_{t+1}$, and episode finish indicator $d$; Add full transition $(S_t, a_t, r, S_{t+1}, d)$
             to $D_{env}$ with priority $p_i = p_{max}$, where $p_{max}$ is the current highest
             priority in $D_{env}$
12         Update local critic network parameters $\phi_1$ and $\phi_2$ and actor network
             parameters $\chi$ on environment data $D_{env}$ with prioritized replay
             (Algorithm 3.11)
13         **Update local critic network parameters** $\phi_1$ **and** $\phi_2$ **and actor
             network parameters** $\chi$ **on neighbor experience data** $D_{neighbor}$
             **with prioritized replay**
14         Soft-update target critic network parameters $\psi_j = \tau\phi_j + (1 - \tau)\psi_j$ with
             $j \in \{1, 2\}$
15         $S_t = S_{t+1}$
16     **end**
17 **end**
```

Algorithm 3.10: Neighbour experience filling step

```
1  for $i$ in range($N_{neighbours}$) do
2      Create a copy $NBE$ of the simulation environment with average crop yields
        $\mu_{orig}$
3      Sample random number $\rho_{nb}$ uniformly from range $[0.5, 0.99]$
4      Determine new average yields for each crop $k$ in NBE:
        $\mu_{nb,k} = N\left(\mu_{orig,k}, (1-\rho_{nb}) \cdot \sigma_{orig,k}\right)$
5      Initialize state $S_t$ in $NBE$
6      for $E$ steps do
7          Filter action space $\mathcal{A}$ with the symbolic planner to receive a subset $\mathcal{A}'$
8          Select crop $a_t$ randomly from $\mathcal{A}'$
9          Cultivate crop $a_t$ in NBE; Obtain profit $r$, next state $S_{t+1}$, and episode
            finish indicator $d$
10         Add full transition $(S_t, a_t, r, S_{t+1}, d)$ to $D_{neighbour}$ with priority $p_i = \rho_{nb}$
            $S_t = S_{t+1}$
11     end
12 end
```

Algorithm 3.11: Update function of SAC learning algorithm with prioritized replay from neighbour experience buffer

```
1  Sample batch with size $n$ from $D_{neighbour}$ with the probability $P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$ for
    each sample $i$
2  for each sample $i$ with data $(S_{t,i}, a_{t,i}, r_i, S_{t+1,i}, d_i)$ do
3      Calculate soft state values:
        $\vartheta_i = \Gamma_\chi(S_{t+1,i})^T \left[\min\left(Q^t_{\psi_1}(S_{t+1,i}), Q^t_{\psi_2}(S_{t+1,i})\right) - \kappa \log \Gamma_\chi(S_{t+1,i})\right]$
4      Calculate critic learning errors: $\delta_{i,j} = r_i + (1-d_i)\gamma\vartheta_i - Q^t_{\phi_j}(S_{t+1,i}, a_{t,i})$ for
        $j \in \{1,2\}$
5  end
6  Calculate critic losses from critic learning errors: $L_{Q,j} = \frac{1}{n}\sum_i^n (\delta_{i,j} \cdot P(i))^2$ for
    $j \in \{1,2\}$
7  Update local critic network parameters $\phi_j$ via gradient descent with losses $L_{Q,j}$
    for $j \in \{1,2\}$
8  Calculate actor loss:
    $L_\Gamma = \frac{1}{n}\sum_i^n P(i) \cdot \Gamma_\chi(S_{t+1,i})^T \left[\kappa \log \Gamma_\chi(S_{t,i}) - \min\left(Q^t_{\phi_1}(S_{t,i}), Q^t_{\phi_2}(S_{t,i})\right)\right]$
9  Update actor network parameters $\chi$ via gradient descent with loss $L_\Gamma$
```

In summary, the agent training process is depicted in Figure 3.7. The state S_t is fed into the RL agent and the automated planner. The planner also receives a list of constraints c to adhere to. The RL agent infers a probabilistic policy π for this state. The automated planner identifies actions which would violate constraints and filters them out of the restricted action space $\mathcal{A}'$. The restricted actions are penalized in the policy to result in an adapted policy $\pi\prime$. During simulation, an action a_t is sampled from the adapted policy. The environment returns a reward r and a new state S_{t+1}. The full transition including previous state, selected action, reward, next state and filtered action space is added to the replay buffer. Additionally, the previous state S_t, the selected action a_t, the next state S_{t+1} and the reward r are used to update the dynamics model of the environment. With the help of the hybrid model for action selection, the dynamics model can plan for k-length predicted trajectories and fill the model experience buffer. The RL agent is updated by training on experience batches from the real experience buffer, the model experience buffer and the neighbour experience buffer.

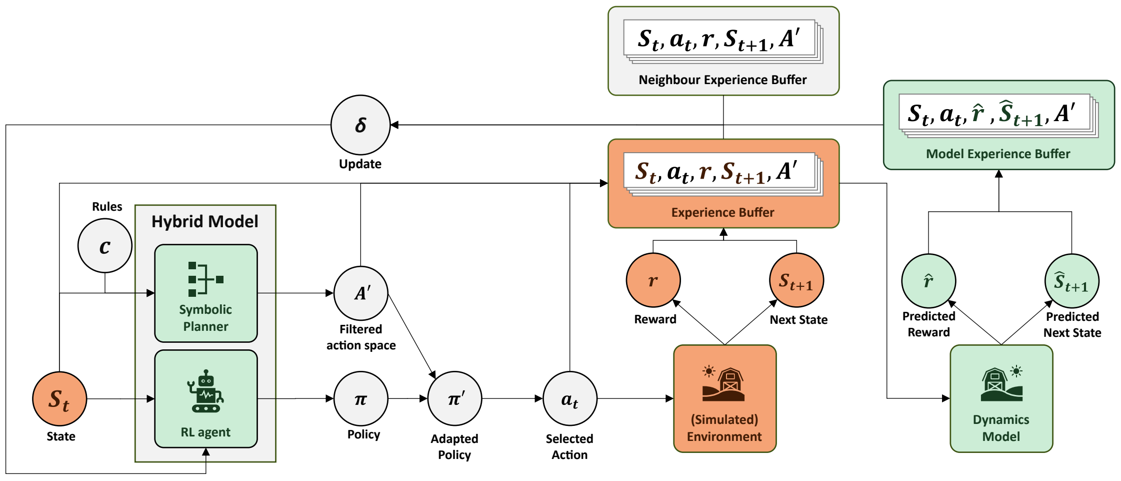
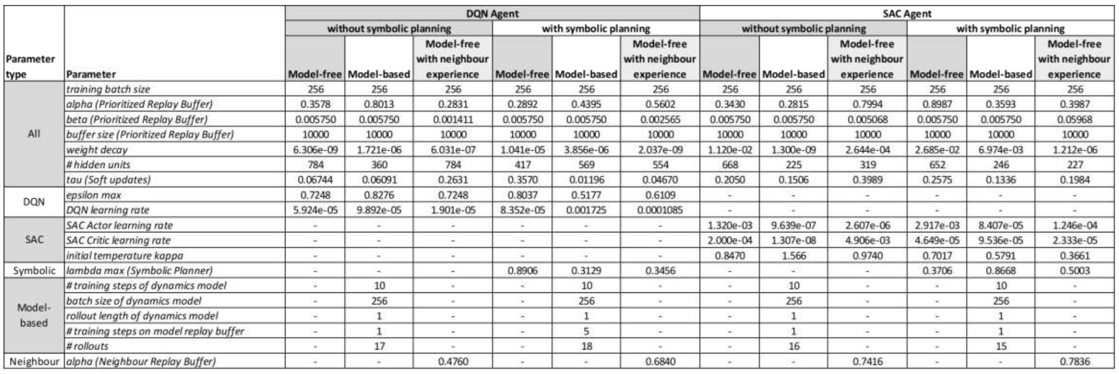


Figure 3.7: Schematic representation of the hybrid model selecting crops and receiving updates.

3.3 Evaluation

To evaluate the proposed models and answer the research questions, several experimental setups need to be tested. The first research question is about comparing performance between different model specifications. Performance can be measured in many ways for RL agents, as was presented in Section 2.2.5. A performance profile made from a set of different performance measures gives insight under which circumstances models perform better or worse and which models are most suitable under real life conditions. The model trainings are run for 180 episodes with each model setting. The hyperparameters for evaluation are determined by running an *optuna* hyperparameter optimization pipeline for 1 hour with model-free and for 2 hours with model-based agents (due to a longer computation time). The pipelines are trained for 90 episodes in each run and new hyperparameter settings are tested until the time limit is reached. The optimal hyperparameters can be observed in Figure 3.8.

After hyperparameter optimization, training performance evaluation is carried out with the best hyperparameter configuration for 20 different seeds per agent type in the same



Parameter type	Parameter	DQN Agent						SAC Agent					
		without symbolic planning			with symbolic planning			without symbolic planning			with symbolic planning		
		Model-free	Model-based	Model-free with neighbour experience	Model-free	Model-based	Model-free with neighbour experience	Model-free	Model-based	Model-free with neighbour experience	Model-free	Model-based	Model-free with neighbour experience
All	*training batch size*	256	256	256	256	256	256	256	256	256	256	256	256
	alpha (Prioritized Replay Buffer)	0.3578	0.8013	0.2831	0.2892	0.4395	0.5602	0.3430	0.2815	0.7994	0.8987	0.3593	0.3987
	beta (Prioritized Replay Buffer)	0.005750	0.005750	0.001411	0.005750	0.005750	0.002565	0.005750	0.005750	0.005068	0.005750	0.005750	0.05968
	buffer size (Prioritized Replay Buffer)	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
	weight decay	6.306e-09	1.721e-06	6.031e-07	1.041e-05	3.856e-06	2.037e-09	1.120e-02	1.300e-09	2.644e-04	2.685e-02	6.974e-03	1.212e-06
	# hidden units	784	360	784	417	569	554	668	225	319	652	246	227
	tau (Soft updates)	0.06744	0.06091	0.2631	0.3570	0.01196	0.04670	0.2050	0.1506	0.3989	0.2575	0.1336	0.1984
DQN	*epsilon max*	0.7248	0.8276	0.7248	0.8037	0.5177	0.6109	-	-	-	-	-	-
	DQN learning rate	5.924e-05	9.892e-05	1.901e-05	8.352e-05	0.001725	0.0001085	-	-	-	-	-	-
SAC	*SAC Actor learning rate*	-	-	-	-	-	-	1.320e-03	9.639e-07	2.607e-06	2.917e-03	8.407e-05	1.246e-04
	SAC Critic learning rate	-	-	-	-	-	-	2.000e-04	1.307e-08	4.906e-03	4.649e-05	9.536e-05	2.333e-05
	initial temperature kappa	-	-	-	-	-	-	0.8470	1.566	0.9740	0.7017	0.5791	0.3661
Symbolic	*lambda max (Symbolic Planner)*	-	-	-	0.8906	0.3129	0.3456	-	-	-	0.3706	0.8668	0.5003
Model-based	*# training steps of dynamics model*	-	10	-	-	10	-	-	10	-	-	10	-
	batch size of dynamics model	-	256	-	-	256	-	-	256	-	-	256	-
	rollout length of dynamics model	-	1	-	-	1	-	-	1	-	-	1	-
	# training steps on model replay buffer	-	1	-	-	5	-	-	1	-	-	1	-
	# rollouts	-	17	-	-	18	-	-	16	-	-	15	-
Neighbour	*alpha (Neighbour Replay Buffer)*	-	-	0.4760	-	-	0.6840	-	-	0.7416	-	-	0.7836

Figure 3.8: Optimal hyperparameter configurations for all model settings.

environment setting with the same ground type and humidity and allows for a robust significance testing as proposed by Colas et al [CSO18]. Additionally, each model setting is trained for 900 episodes on 5 different seeds and a single environment setting to see if they converge to an even higher average performance when being trained for much longer. All model settings are compared against random action selection and filtered random action selection to observe the impact of training RL agents instead of only following a simple rule-based approach or not following any rules at all. The performance measures selected for analysing training performance are the following:

- **Mean episodic profit**: Main performance measurement.
- **Conditional Value at Risk (CvaR)**: Average value of the lowest 30% quantile of total profits to analyse risk when training this model setting.
- **Normalized interquartile range (IQR) of performance**: Indicator of relative profit/performance variance during and across training runs.

The mean episodic profit is additionally summed up to obtain the cumulative profit. The performance is averaged over all runs with the same agent type. The CVaR is only calculated for the cumulative profit over runs to compare between average runs and the worst 30% of runs. The interquartile range of performance is calculated for low-pass-filtered profits across runs according to the proposed evaluation of reliability by Chan et al. (2019) [CFC+19]. By using a low-pass filter before calculating the IQR, the effect of intra-run variance is reduced. The low-pass filter in use is calculated in the following way with r_i being the total profit of episode i. The parameter η is set to 0.05 to achieve a strong intra-run variance reduction without losing too much information. The calculation of the low-pass-filtered reward is depicted in formulas 3.10 and 3.11.

$$r_i = \alpha r_{i-1} + \beta(r_i - r_{i-1}) \tag{3.10}$$

$$\alpha = \frac{(2-\eta)}{(2+\eta)}, \quad \beta = \frac{\eta}{(2+\eta)} \tag{3.11}$$

The second reliability measure proposed by them is to calculate the IQR for detrended performance during runs. Detrending is carried out by calculating the difference between one episode's profit and the next as depicted in formula 3.12.

$$\Delta_i = r_i - r_{i-1} \tag{3.12}$$

Afterwards, the interquartile range of detrended performance over a time window of 15 episodes in each run is calculated. Both the IQR across and the IQR during runs are normalized by dividing them by the 95th percentile of total profits. This is done to ensure that only variance relative to the total profit is compared as a higher absolute fluctuation is expected if agents demonstrate a higher performance. The described measures are compared visually to evaluate trends in model performance. Additionally, mean values and distributions for the following episode ranges are obtained via bootstrap sampling.

- **First 20 episodes**: Indicators on how well and stable an agent learns without much experience.
- **Last 30 episodes**: Indicators on how well the algorithm converges towards viable and stable policies in later stages of training.
- **Total run**: Indicators for overall performance and reliability.

By sampling from the original runs with replacement, it is possible to generate confidence intervals for the obtained metrics. The number of bootstrap samples is 50.000 for each agent type. The interquartile range across runs is calculated over each sample of runs individually. To analyse the significance in metric differences, permutation tests are used [Goo13]: All tests between the mean values of two different agent types are run by setting the mean difference of their metrics across all runs as a threshold. Then, the metrics from each run of both agent types are concatenated, permuted and split in half to obtain samples under the null hypothesis that the per-run metrics from both agent types stem from the same distribution. This is repeated 1000 times and the number of permuted mean differences more extreme in the target direction than the previously set threshold is counted. Dividing this number by 1000 results in the p-value for the hypothesis test. P-values smaller than 0.05 are interpreted as significant, values smaller than 0.01 are interpreted as highly significant. For the IQR across runs, a variant of this test is used with the IQRs being calculated after the permutation and splitting step. Another relevant evaluation is to test how well the agents perform when being pretrained on one environment and then being deployed on another environment. To test this, agents pretrained for 180 episodes on one environment are run on 5 other environments for 90 episodes each. Performance measures for these tests are again mean profit, conditional value at risk and the normalized interquartile profit range during and across runs. Results from this evaluation show which agent types are most suitable for pretraining agents that can be used on other environments. A last step during evaluation is to test how diverse crop selection is under different model settings. This evaluation is used to answer research question 2. Crop diversity is measured as the Shannon entropy over all selected actions during training. It is evaluated across full training runs and for the first 20 and last 30 episodes.

CHAPTER 4

Results & Discussion

In the following section, the outcomes of the study are presented. At first, an individual training run is examined in detail and it is shown how the proposed model architecture fares when compared against a fixed crop rotation suitable for the environment. Then, agent performance is compared for all agent types and subsequently discussed regarding the research questions. Plausible explanations for the observed results are theorized.

4.1 Detailed examination of individual training runs

At first, we examine a single training run of the proposed hybrid model to understand the level of profit variance the agent must deal with while learning a beneficial policy. In Figure 4.1, it can be observed how the agent slowly improves after the first 30 episodes and rather consistently outperforms the random action selection across the whole training run. All episode results after episode 35 exhibit an at least positive total reward. The batch size for all experiments was set to 256. As the agents do not train before having enough experience in the buffer to fill one batch, the actual learning and policy updating only starts after episode 25. On the right side of Figure 4.1, the episodic profits from applying the same crop sequence in every episode can be observed. The crop sequence in use was obtained by running a complete training run for 500 episodes on the same environment and then finding the crop sequence resulting in the highest episodic profit. The sequence in use cultivated the following crops in order: potato, sugar beet, grass, potato, grass, potato, sugar beet, potato, grass, potato. Although the fixed crop sequence policy regularly exhibits high episodic profits in our simulation environment, it does not adapt its crop selection to different prices and costs and might have issues in real market conditions. This would in this example be the case if the market price for potatoes and sugar beet decreases significantly.

When examining individual episodes of the SAC agent with symbolic planning and neighbour experience closer in Figures 4.2, 4.3 and 4.4, a more detailed image of agent

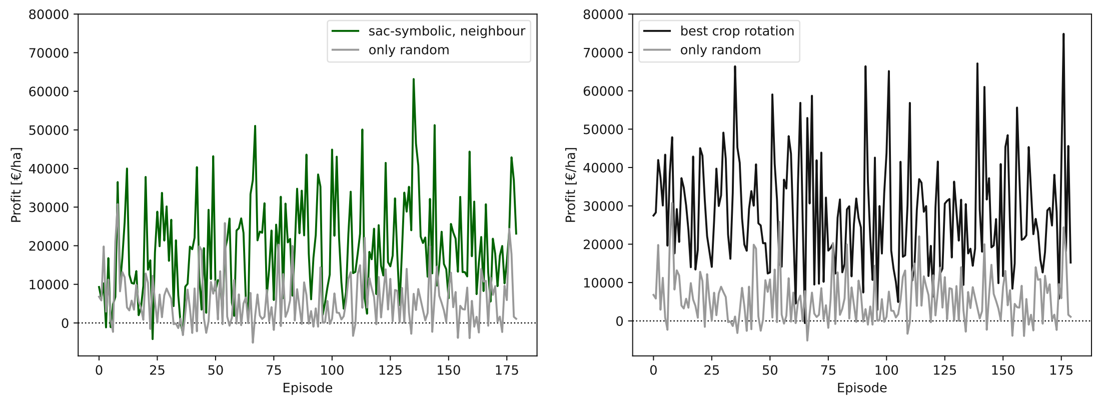


Figure 4.1: *Left*: Single training run of SAC agent using symbolic planning and neighbour experience in an example environment with a medium ground type and wet conditions when compared to random action selection. *Right*: Performance comparison of selecting the same crop sequence in each training episode against random crop selection. The crop sequence in use was the one with the highest profit in a training run over 500 episodes with the same agent in the same environment.

behaviour appears. During both episodes, the agent starts by cultivating cash crops like winter wheat, potato or sugar beet. Crop rotation rules are generally adhered to, which might be either due to the action filtering mechanism of the symbolic planner or the agent already having learned not to break crop rotation rules due to reduced yields. In episode 135, the agent however violates the minimum break rule to cultivate potatoes twice in year 1 and year 4 and then again in year 6 and year 8. The resulting profits are high which justifies the behaviour economically.

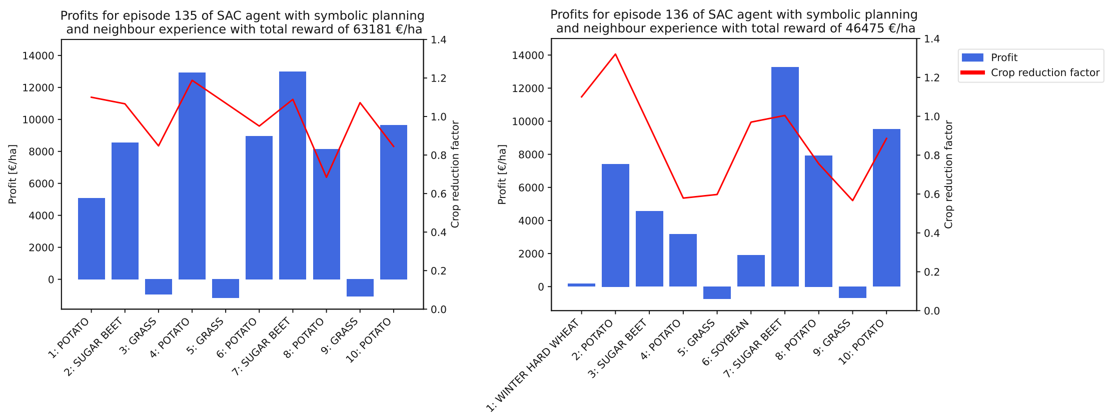
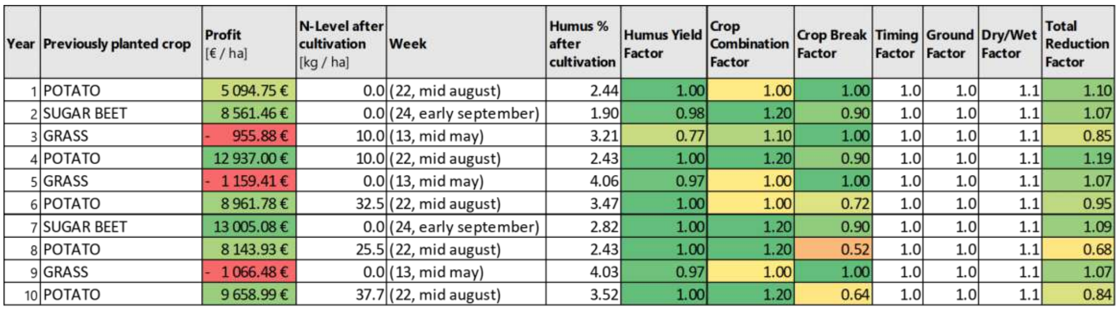


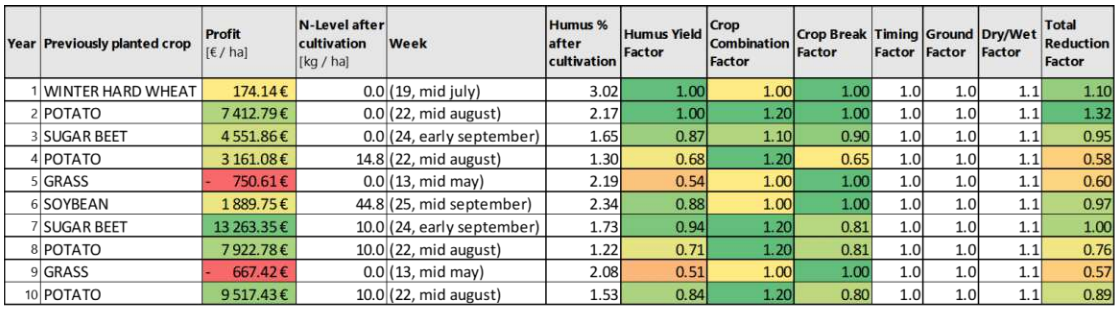
Figure 4.2: *Left*: Individual profits during episode 135 of the example environment. *Right*: Individual profits during episode 136 of the example environment.

In both episodes, a sequence of cash crops lowering the humus soil ratio is followed by



Year	Previously planted crop	Profit [€ / ha]	N-Level after cultivation [kg / ha]	Week	Humus % after cultivation	Humus Yield Factor	Crop Combination Factor	Crop Break Factor	Timing Factor	Ground Factor	Dry/Wet Factor	Total Reduction Factor
1	POTATO	5 094.75 €	0.0	(22, mid august)	2.44	1.00	1.00	1.00	1.0	1.0	1.1	1.10
2	SUGAR BEET	8 561.46 €	0.0	(24, early september)	1.90	0.98	1.20	0.90	1.0	1.0	1.1	1.07
3	GRASS	- 955.88 €	10.0	(13, mid may)	3.21	0.77	1.10	1.00	1.0	1.0	1.1	0.85
4	POTATO	12 937.00 €	10.0	(22, mid august)	2.43	1.00	1.20	0.90	1.0	1.0	1.1	1.19
5	GRASS	- 1 159.41 €	0.0	(13, mid may)	4.06	0.97	1.00	1.00	1.0	1.0	1.1	1.07
6	POTATO	8 961.78 €	32.5	(22, mid august)	3.47	1.00	1.00	0.72	1.0	1.0	1.1	0.95
7	SUGAR BEET	13 005.08 €	0.0	(24, early september)	2.82	1.00	1.20	0.90	1.0	1.0	1.1	1.09
8	POTATO	8 143.93 €	25.5	(22, mid august)	2.43	1.00	1.20	0.52	1.0	1.0	1.1	0.68
9	GRASS	- 1 066.48 €	0.0	(13, mid may)	4.03	0.97	1.00	1.00	1.0	1.0	1.1	1.07
10	POTATO	9 658.99 €	37.7	(22, mid august)	3.52	1.00	1.20	0.64	1.0	1.0	1.1	0.84

Figure 4.3: Individual crops' profits, week of year, nitrogen level and humus ratio after cultivation as well as reduction factors obtained from the simulation environment after cultivating the crops in episode 135 of the example environment.



Year	Previously planted crop	Profit [€ / ha]	N-Level after cultivation [kg / ha]	Week	Humus % after cultivation	Humus Yield Factor	Crop Combination Factor	Crop Break Factor	Timing Factor	Ground Factor	Dry/Wet Factor	Total Reduction Factor
1	WINTER HARD WHEAT	174.14 €	0.0	(19, mid july)	3.02	1.00	1.00	1.00	1.0	1.0	1.1	1.10
2	POTATO	7 412.79 €	0.0	(22, mid august)	2.17	1.00	1.20	1.00	1.0	1.0	1.1	1.32
3	SUGAR BEET	4 551.86 €	0.0	(24, early september)	1.65	0.87	1.10	0.90	1.0	1.0	1.1	0.95
4	POTATO	3 161.08 €	14.8	(22, mid august)	1.30	0.68	1.20	0.65	1.0	1.0	1.1	0.58
5	GRASS	- 750.61 €	0.0	(13, mid may)	2.19	0.54	1.00	1.00	1.0	1.0	1.1	0.60
6	SOYBEAN	1 889.75 €	44.8	(25, mid september)	2.34	0.88	1.00	1.00	1.0	1.0	1.1	0.97
7	SUGAR BEET	13 263.35 €	10.0	(24, early september)	1.73	0.94	1.20	0.81	1.0	1.0	1.1	1.00
8	POTATO	7 922.78 €	10.0	(22, mid august)	1.22	0.71	1.20	0.81	1.0	1.0	1.1	0.76
9	GRASS	- 667.42 €	0.0	(13, mid may)	2.08	0.51	1.00	1.00	1.0	1.0	1.1	0.57
10	POTATO	9 517.43 €	10.0	(22, mid august)	1.53	0.84	1.20	0.80	1.0	1.0	1.1	0.89

Figure 4.4: Individual crops' profits, week of year, nitrogen level and humus ratio after cultivation as well as reduction factors obtained from the simulation environment after cultivating the crops in episode 136 of the example environment.

cultivating grass as an intermediate crop not affecting any minimum breaks but instead building up the soil humus ratio. Immediately after grass cultivation, another cash crop is planted in episode 135 whereas soybeans are cultivated in episode 136 which have a positive effect on the humus soil ratio and improve the yield of the following sugar beet cultivation. Both example episodes show a general trend towards cultivating the main cash crops sugar beet and potato and then cultivating crops from other botanical families until the cash crops can be cultivated again without violating minimum crop breaks. While the behaviour follows a similar pattern in both episodes, the total episodic profit is different. A reason for that might be that the available cash crops have lower prices over many time steps in episode 136 and the complete episode has a lower profit potential overall. This theory is underlined by examples like year 2 in episode 136 where the profit from potato cultivation is rather low even though the yield factor is high with a value close to 1.3. This is in contrast to year 8 of episode 135 where the profit is about 500 €/ha higher even though the yield factor is as low as 0.76.

While these examples help to understand how agents might behave during episodes, it is necessary to aggregate average performance and reliability metrics across runs to observe general trends. In Figure 4.5, the mean episodic training rewards of the SAC agent with symbolic planning and neighbour experience is compared to random crop selection and the best crop sequences obtained from the first 5, 10, 20, 50, 100 and 500 training episodes. It is apparent that the agent only finds crop sequence highly suitable for the environment after enough training time. Only after about 150 episodes, the SAC agent does not improve anymore. It then exhibits equal mean rewards as the best crop sequence obtained after 100 episodes and clearly outperforms sequences obtained from earlier points during the training run. By only focusing on fixed crop sequences during farming, a huge learning potential would therefore be dismissed.

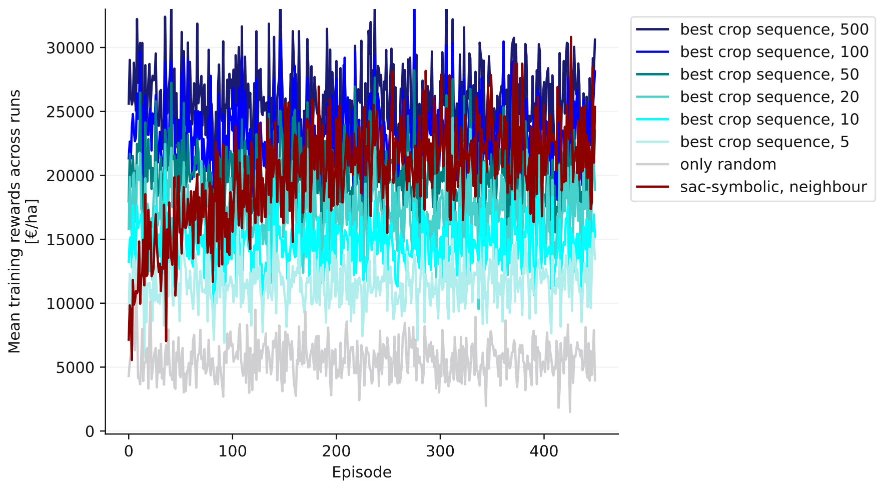


Figure 4.5: Mean training rewards across 20 different runs for the SAC agent with symbolic planning and neighbour experience when compared to random crop selection and the best crop rotation sequences obtained from the first 5, 10, 20, 50, 100 and 500 episodes.

4.2 General comparison of different agent types

In Figure 4.6, the bootstrapped sample distribution of mean total profits after 180 episodes are shown for different agent types. As an example, all hybrid agents using a symbolic planner (suffix "symbolic") show a consistently better mean cumulative performance after 180 episodes than their non-symbolic counterparts. When taking a look at the mean cumulative profits plotted against the episode numbers in Figure 4.7, it is noticeable that hybrid agents using a symbolic filtering mechanism gain a head start by immediately performing well from the start whereas RL agents without symbolic planning need more episodes to find actions resulting in higher profits. A possible explanation is the restricted exploration ability of hybrid agents. While non-symbolic RL agents without any experience try out highly unsuitable actions, the hybrid agents have a higher probability to immediately find rewarding actions and receive a positive reinforcement from them. Additionally, agent updates during training are more efficient as the bootstrapped action-values for the next states only include actions following crop rotation rules. Due to that, those values become more precise in earlier episodes of the training run as the actions from the filtered action space are sampled more often. Furthermore, a negative bias from the bootstrapped action values during early training is avoided for the SAC agents when unsuitable actions might still have a high probability to be selected. The highest difference in mean cumulative profits is observed between symbolic and non-symbolic model-based RL agents. The non-symbolic model-based versions do not seem to learn at all with the SAC agent performing just as bad as random action selection and the model-based DQN agent performing even worse. This effect could stem from the dynamics model learning from state transitions with a high ratio of badly performing early training episodes. Due to the low number of experiences with high profits in the training batches, the dynamics model possibly does not generalize well for suitable state-action pairs. This could result in the agents only receiving negative reinforcement and getting stuck in local optima. Additionally, it might be hard to find a suitable hyperparameter configuration for those agent types and the time limit of two hours resulted in a non-performing configuration. In contrast, the action filtering mechanism of the hybrid models allows the dynamics models to train on more transitions with high profits which might lead to a better precision when predicting beneficial state-action pairs.

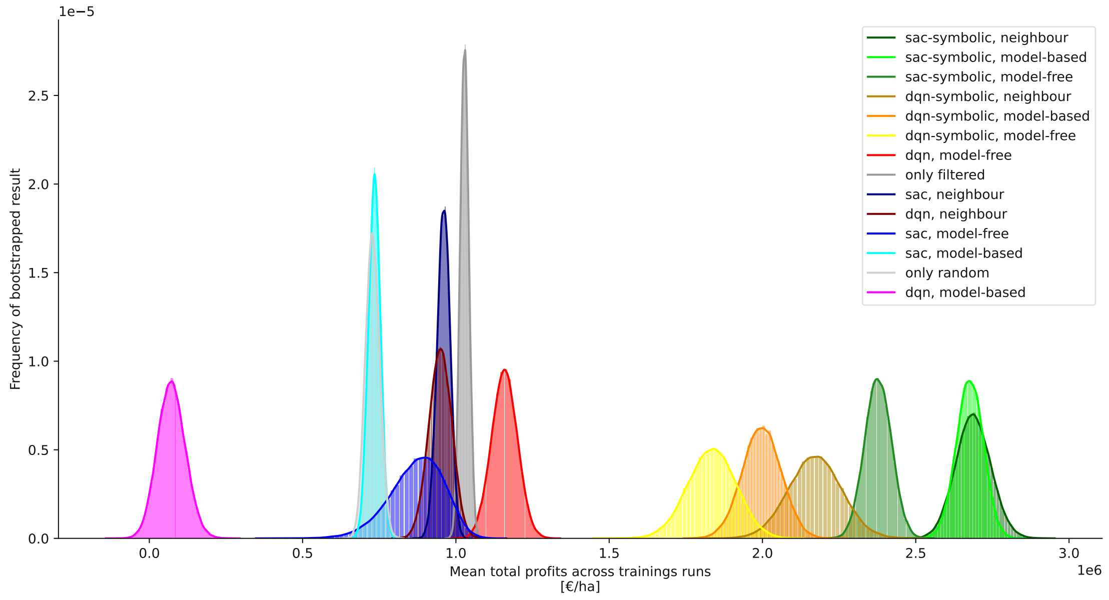


Figure 4.6: Bootstrapped distribution of mean total profits after 180 episodes across training runs.

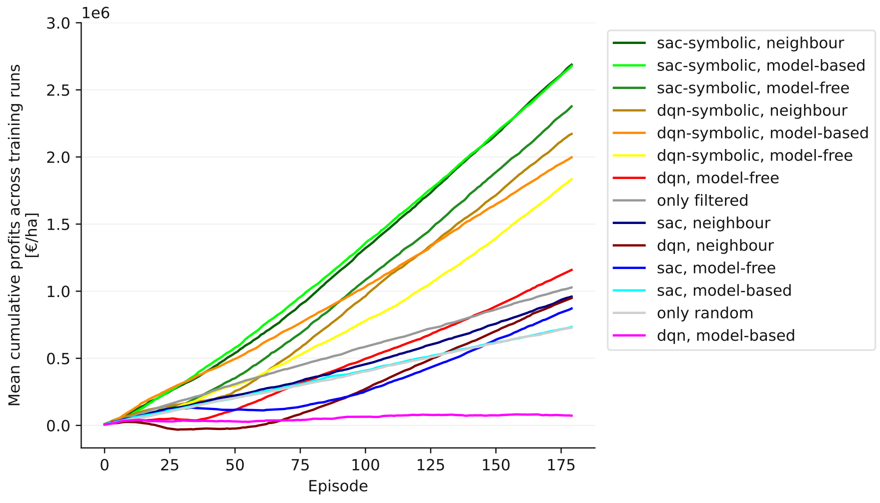


Figure 4.7: Mean cumulative profits across training runs for all agent types.

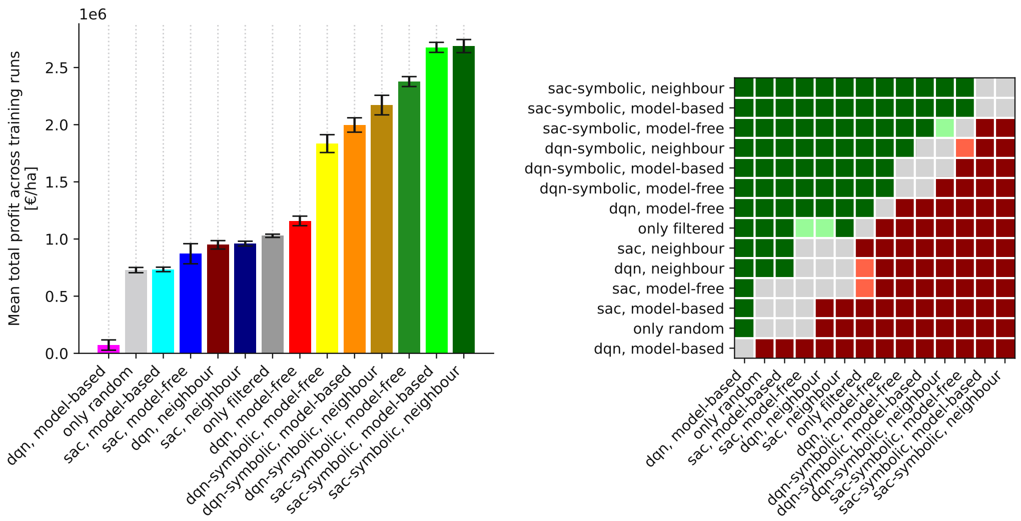


Figure 4.8: Mean bootstrapped total profits across runs for all agent types (*left*) and significance indicators for pair-wise performance differences of mean bootstrapped total profits across runs for all agent types (*right*). Dark green symbolizes that the agent from the row has a highly significantly better performance than the agent from the column with a p-value < 0.01. Light green marks a significant positive difference with a p-value smaller than 0.05. Dark and light red fields represent the mirrored relationship and are included to make the plot easier to read. Grey fields mark two agents which were not significantly different in performance during the permutation test.

It is necessary to check the visual results for statistically significant differences. In Figure 4.8, the mean bootstrapped total profits across runs including their confidence intervals are shown for all agent types. The significant differences in mean total profits between two agents are marked in the right chart. In the context of comparing symbolic to non-symbolic agents, it can be observed that all symbolic agents perform better with a highly significant difference than their non-symbolic counterparts.

Examining the CVaR cumulative profits in Figure 4.9, a similar image appears. The mean cumulative profits for the lowest 30% quantile are lower than the mean cumulative profits across all runs but the relative differences between different agent types stay alike. This shows that most agents exhibit a stable performance across runs. Runs with a lower cumulative reward might additionally be affected by lower prices and higher costs even though the agents still select the best possible actions.

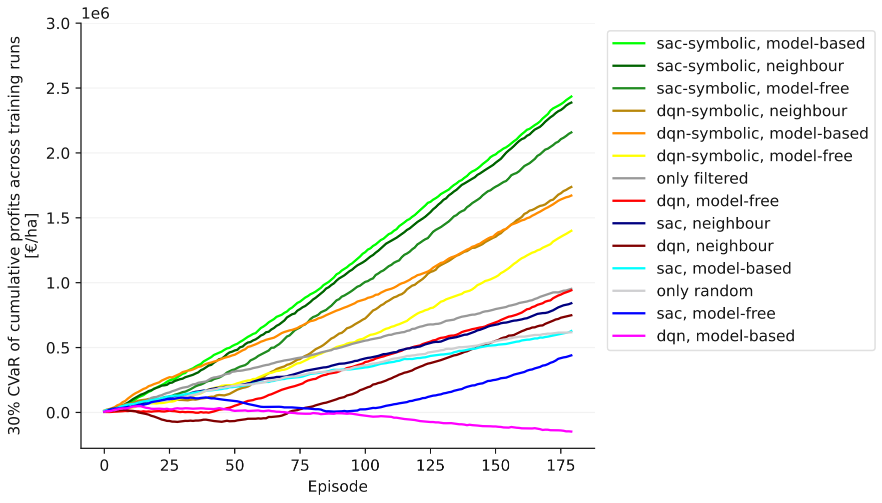


Figure 4.9: CVaR of cumulative profits across training runs for all agent types.

To illuminate this more precisely, we examine the cumulative performance over 900 episodes in Figure 4.10. Over a longer period, the performance of most agents remains stable. Within the DQN-based agents, the model-free symbolic agent and the model-free non-symbolic agent gradually improve performance. This results in the model-free non-symbolic agent to overtake the model-based symbolic agent and the symbolic agent using neighbour updates after about 500 episodes. The theory behind is that model-free algorithms are less stabilized by neighbour updates or the dynamics model generalizing for unseen transitions. While this instability is adverse in the early stage of training, it is useful much later when other algorithms already converged to a stable but not necessarily optimal policy.

In Figure 4.11, the mean profits over the first 20 episodes across runs are shown with significance indicators for pairwise performance differences obtained by permutation tests. The variance across bootstrap samples is relatively low which is confirmed by the significance indicators. In general, agents learning fast and stable, which includes most symbolic agents, perform well over the first 20 episodes and show significant performance

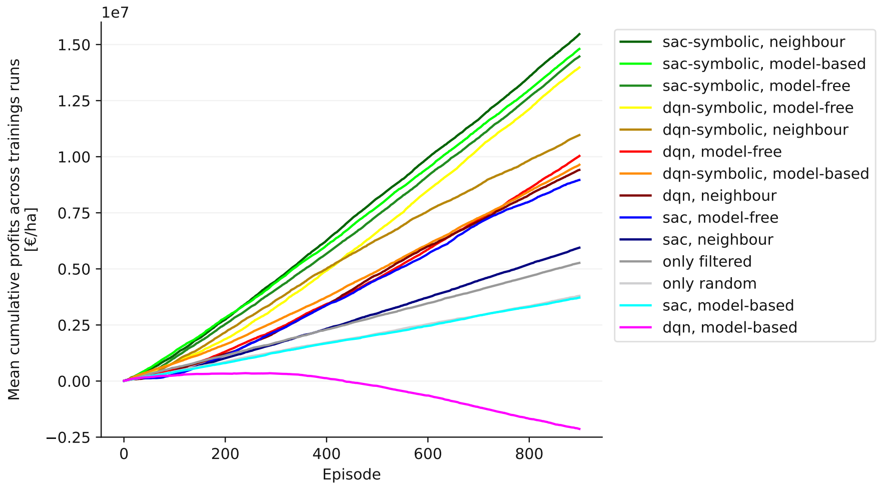


Figure 4.10: Mean cumulative profits across runs for all agent types over 900 episodes.

differences to agents not picking up good policies fast. Symbolic model-free agents seem to converge more slowly to good performances than agent types using neighbour experience or a pretrained dynamics model. Many agents in the middle of the field do not show any significant differences in performance towards each other over the early episodes. Some outliers like the three non-symbolic DQN agents perform particularly bad which shows that the DQN algorithm has trouble finding suitable actions early without any external guidance.

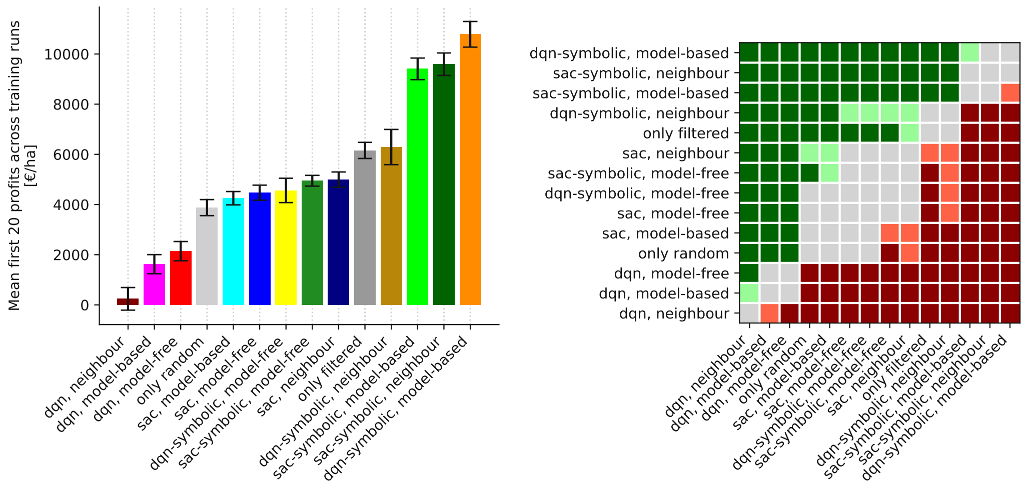


Figure 4.11: Mean bootstrapped profits over the first 20 episodes across runs and significance indicators for pair-wise performance differences of mean bootstrapped profits over the first 20 episodes across runs for all agent types.

Over the last 30 episodes of the training runs, this image changes, as depicted in Figure 4.12. The non-symbolic DQN agents (with the model-based variant as an exception) picked up a viable strategy and were able to perform better than their non-symbolic SAC counterparts. In the top of the field, the symbolic SAC agents showed a similar and significantly better performance than all other agent types with symbolic DQN agents right behind. Non-symbolic agents in general were significantly worse than symbolic agents in mean performance over the last 30 episodes of training runs.

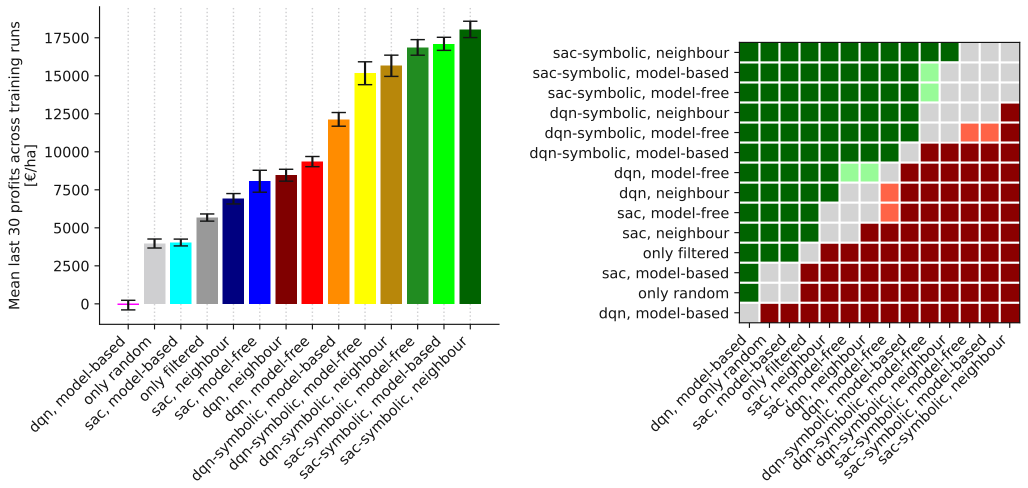


Figure 4.12: Mean bootstrapped profits over the last 30 episodes across runs and significance indicators for pair-wise performance differences of mean bootstrapped profits over the last 30 episodes across runs for all agent types.

Regarding the reliability of agents, no obvious trends between different agent types can be observed. Agents that generally perform badly seem to have a slightly higher normalized IQR of performance across runs, which can be observed in Figure 4.13 and Figure 4.14. A reason might be that most of those agents' training runs are stuck in a cycle of only selecting the same action for each step in an episode. As some actions lead to higher costs on average while others represent lower costs, the range of performance across runs is higher than for agents, which select the most suitable actions for each run and find optimized crop rotation sequences for their respective environments. Similarly, agents performing badly have slightly higher normalized IQRs of performance during runs depicted in Figure 4.15 and Figure 4.16. This could be the result of agents not reacting to different prices and costs which vary during and across episodes. A general baseline of variance between and during runs will also be the result of varying prices and costs for different crop types which represents a realistic scenario in real life. Well performing agents can guarantee a suitable crop selection even if exogenous conditions are not optimal. This behaviour is demonstrated by many of the symbolic agents. Another insight gained by visualizing the normalized IQRs of performance across runs is that all agent types show a higher performance variance across runs during the early episodes of training. It shows that agents become more stable after several episodes of training and converge to stable and similar performances across runs in the long term.

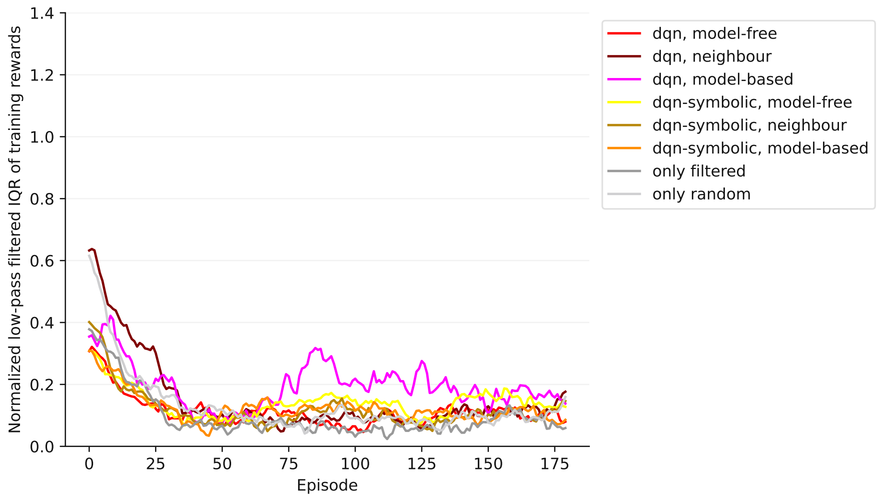


Figure 4.13: Mean normalized low-pass-filtered IQR of training rewards across runs for all DQN-based agents.

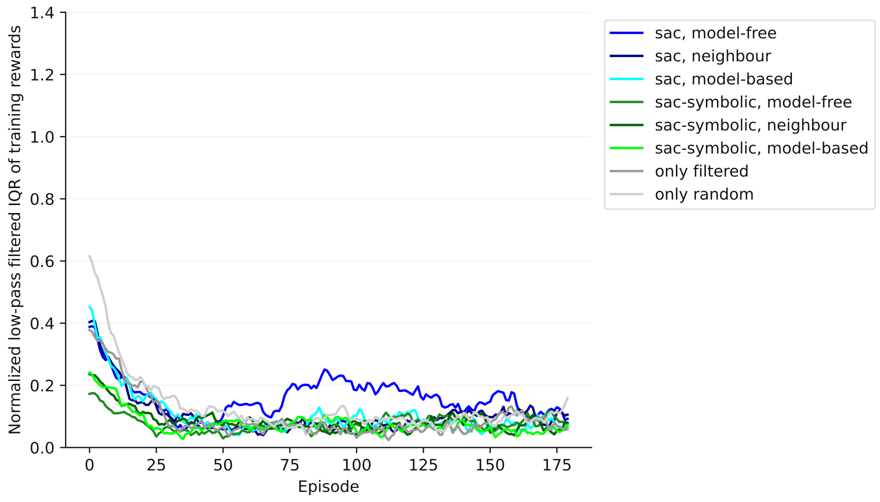


Figure 4.14: Mean normalized low-pass-filtered IQR of training rewards across runs for all SAC-based agents.

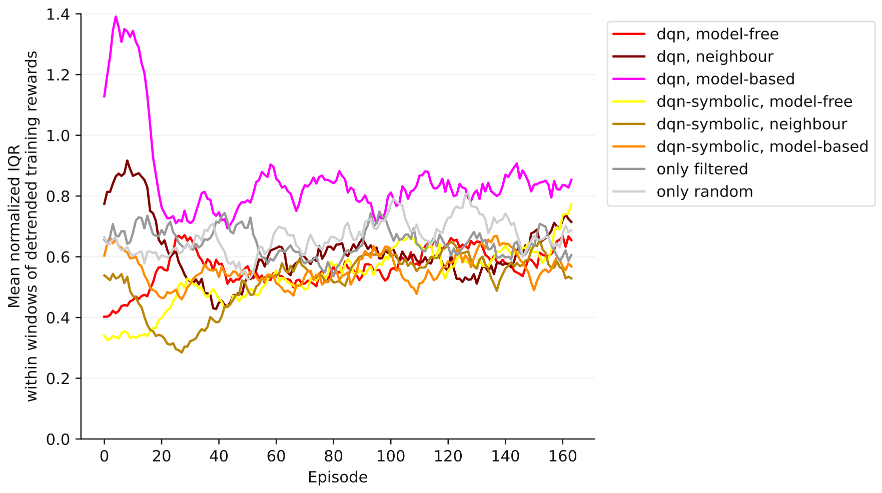


Figure 4.15: Mean normalized IQR within windows of detrended training rewards during runs for all DQN-based agents.

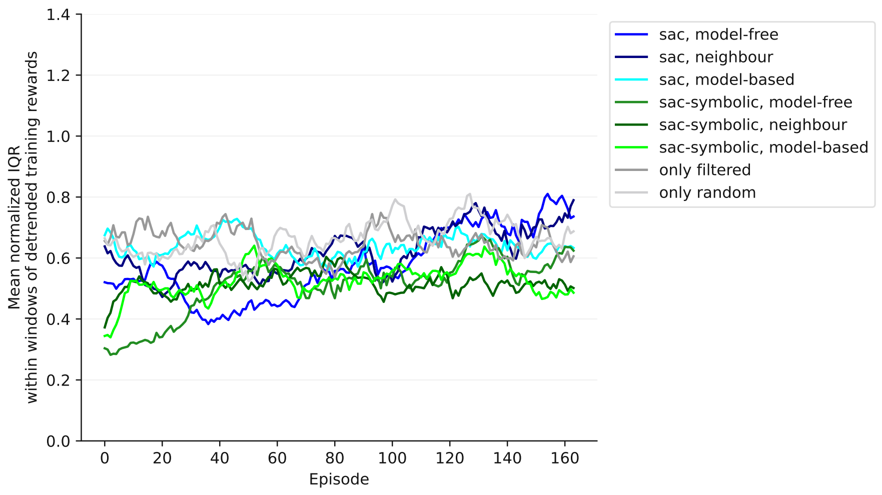


Figure 4.16: Mean normalized IQR within windows of detrended training rewards during runs for all SAC-based agents.

The results depicted in the previous plots were obtained from environments with a medium ground type and a high humidity. Other settings should result in the agents selecting different crops more suitable to the changed environment. A comparison of applying random action selection to environments with different soil types and humidity is shown in Figure 4.17. The highest profits are obtained within humid environments and heavy soils. Dry environments generally have a lower expected yield and therefore perform worse. In Figure 4.18, the performance curves for agents training on an environment with bad conditions with a light soil and low humidity can be examined. The relative differences between the mean total profits of the agents are similar to training on better environmental conditions with the exception that all non-symbolic agents obtain a lower mean cumulative reward after 180 episodes than the filtered random action selection. A reason might be that unrestricted exploration leads to more unfavourable actions in harsher environments.

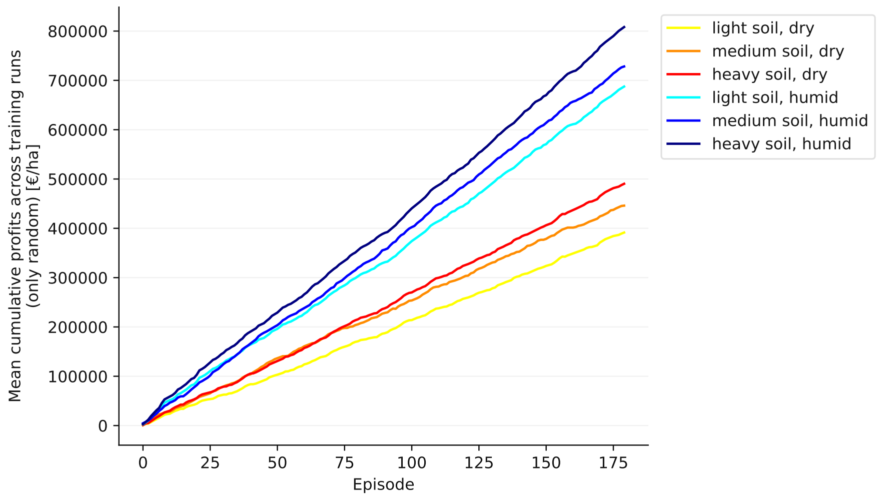


Figure 4.17: Mean cumulative training rewards across runs with random action selection for different environment settings.

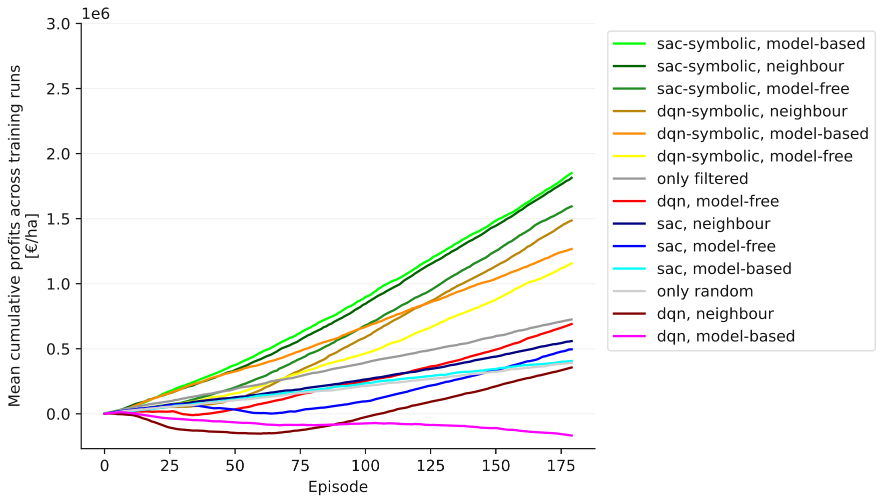


Figure 4.18: Mean cumulative training rewards across runs for all agents with light soil and low humidity.

After obtaining a general overview about the performance and reliability of different algorithms and agent features, a more detailed comparison is structured into sections related to the individual research questions.

4.3 Research question 1.1: Profit and reliability differences between hybrid systems and pure RL agents

Research question 1.1 is about analysing performance differences between hybrid systems combining a rule-based planner with an RL agent and pure RL agents. In Figure 4.19, it becomes clear that all symbolic agent versions perform significantly better than their non-symbolic counterparts when comparing total profits across runs. The performance differences are particularly high for the model-based versions. Significant differences are symbolized by the horizontal lines under the pairs of bars. Dark green represents the mean of the right bar being significantly higher than mean of the left bar with a p-value smaller than 0.01. Light green represents the same relationship with a p-value smaller than 0.05. Light red and dark red represent the opposite relationship with p-values of 0.05 and 0.01 respectively. Examining the average episodic performances during the first 20 and last 30 episodes of training runs in Figure 4.20 explains the difference in performance in more detail. The difference already appears in the first 20 episodes for all agents except the model-free SAC versions. Especially the symbolic agents using neighbour experience or a dynamics model profit from those features whereas the non-symbolic variants do not experience any significant improvements from them. In the late episodes, this advantage is expanded. Particularly the model-free non-symbolic variants improve their mean episodic performance greatly.

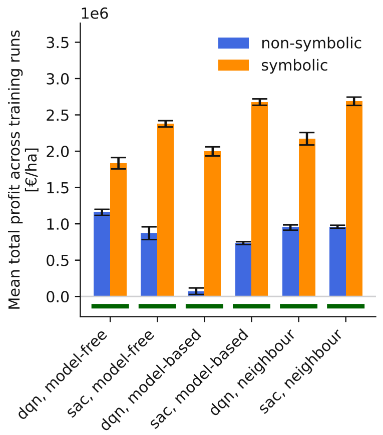


Figure 4.19: Direct comparison of mean total profits across runs from symbolic and non-symbolic versions of different agent types.

Regarding the reliability measures, no trend between symbolic and non-symbolic agent types can be determined for the normalized IQR of performance across runs depicted in Figure 4.21. For the first 20 episodes and last 30 episodes, no observable significant

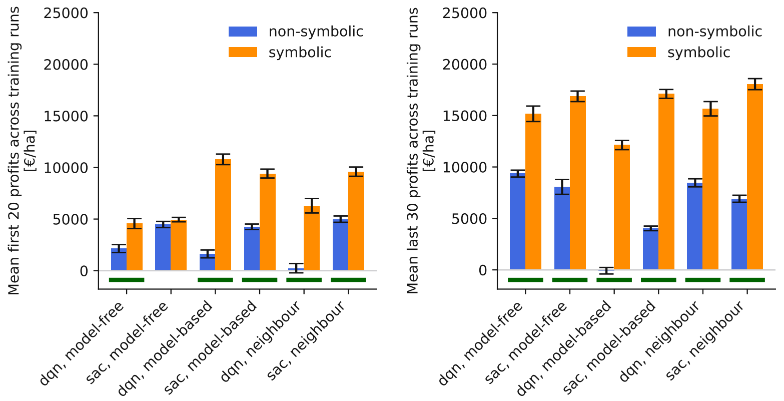


Figure 4.20: Direct comparison of average training rewards per episode for the first 20 (*left*) and last 30 episodes (*right*) of training runs from symbolic and non-symbolic versions of different agent types.

difference between most symbolic and non-symbolic agents is detected as the confidence intervals are too large.

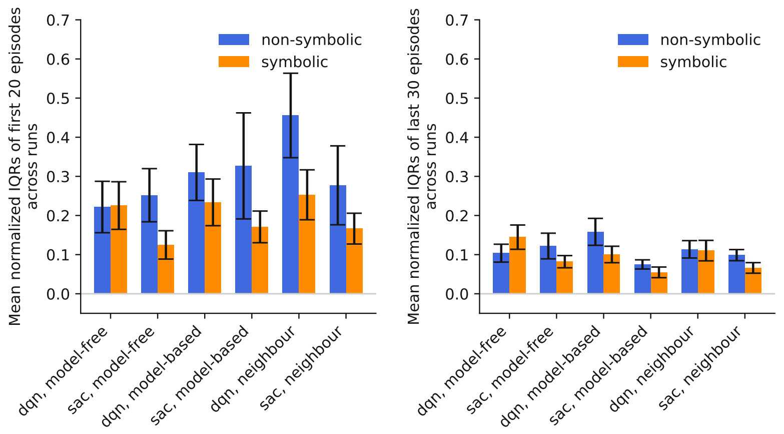


Figure 4.21: Direct comparison of mean normalized IQRs across runs for the first 20 (*left*) and last 30 episodes (*right*) of training runs from symbolic and non-symbolic versions of different agent types.

When comparing the IQRs of detrended performance during runs in Figure 4.22, there is an at least significant difference between the symbolic and non-symbolic versions of each agent type which can be explained by the more stable training performance of symbolic agents in the early training episodes. In the last 30 episodes, this significant difference is still observed for some agent types like the model-based DQN agent, while other agent types show no significant difference in IQRs anymore after training performance has stabilized.

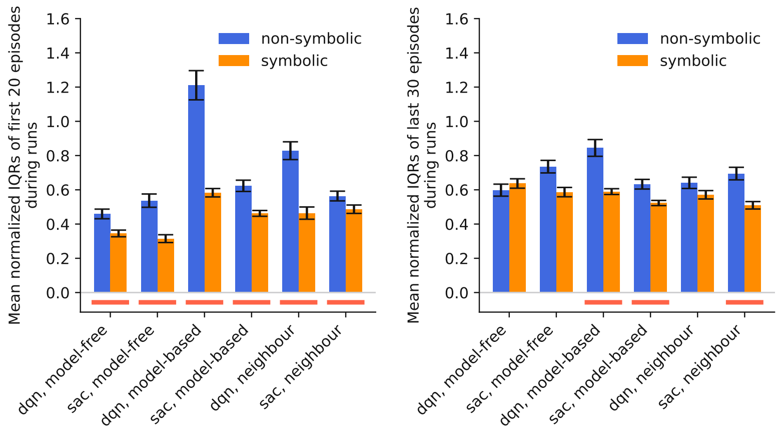


Figure 4.22: Direct comparison of mean normalized IQRs within windows of detrended training rewards during runs for the first 20 (*left*) and last 30 episodes (*right*) of training runs from symbolic and non-symbolic versions of different agent types.

4.4 Research question 1.2: Profit and reliability differences between Deep Q Learning (DQN) and Soft-Actor Critic (SAC) agents

Research question 1.2 focuses on investigating the performance disparities between DQN agents and their counterparts utilizing the SAC algorithm. It is expected that the more modern SAC algorithm is superior to the DQN agent in performance even though it is modified to work for discrete settings for which it was not originally designed for. This expected trend is confirmed by examining the bootstrapped mean total profits depicted in Figure 4.6 and in more detail in Figure 4.23, particularly for symbolic RL agents. Symbolic SAC agents, encompassing model-free and model-based agents and those incorporating neighbour experience, exhibit significantly higher mean total profits than their corresponding DQN counterparts.

Furthermore, DQN agents display a higher variance across bootstrap samples, indicating a less stable learning process. Analysis of mean total reward curves in Figure 4.10 reveals a swifter learning process for symbolic SAC agents compared to symbolic DQN agents. Upon comparing symbolic model-free SAC and DQN variants in Figure 4.24, no significant difference in mean performance is observed for the initial 20 episodes though. For symbolic model-based variants, the DQN agent even outperforms the SAC agent. However, the symbolic SAC agent using neighbour experience already outperforms the respective DQN agent significantly in the first 20 episodes. Over the last 30 episodes, all three symbolic SAC agents consistently demonstrate significantly higher mean performance than their DQN counterparts, a trend confirmed in longer training runs where SAC agents maintain their advantage.

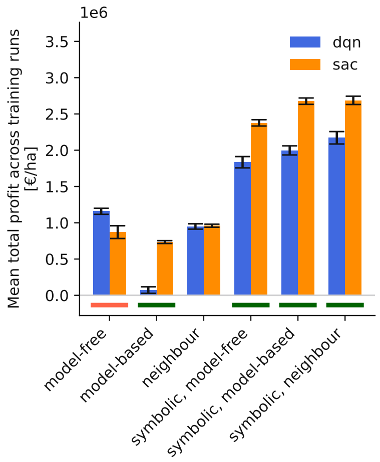


Figure 4.23: Direct comparison of mean total profits across training runs from DQN and SAC agents.

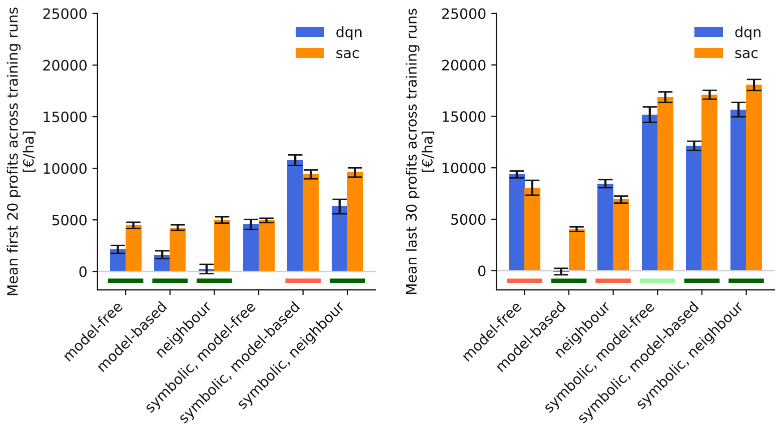


Figure 4.24: Direct comparison of average training rewards per episode for the first 20 (*left*) and last 30 episodes (*right*) of training runs from DQN and SAC agents.

In contrast, for non-symbolic agents, which are generally underperforming compared to their symbolic counterparts, this pattern is reversed. While non-symbolic model-free SAC agents with and without neighbour experience initially perform well with a significantly higher mean performance than their DQN counterparts, this advantage diminishes during training. Non-symbolic DQN agents without a dynamics model exhibit significantly higher mean performance over the last 30 episodes of the training runs. Especially model-free DQN agents exhibit improvement over longer training runs of 900 episodes, converging to a higher average performance per episode. This phenomenon may be linked to the more random nature of ϵ-greedy exploration in DQN agents for long training runs compared to the probabilistic action selection of SAC agents, allowing for ongoing optimization potential in late training stages. Model-based non-symbolic variants, on the other hand, perform equally or worse than random action selection, showing no discernible learning process. Reliability differences between DQN and SAC agents are minimal, with DQN agents displaying a slightly higher IQR of performance across runs depicted in Figure 4.25. Due to the high variance between permutation tests, most differences are not significant. Only the model-free agents with neighbour experience exhibit a significant difference over the first 20 episodes.

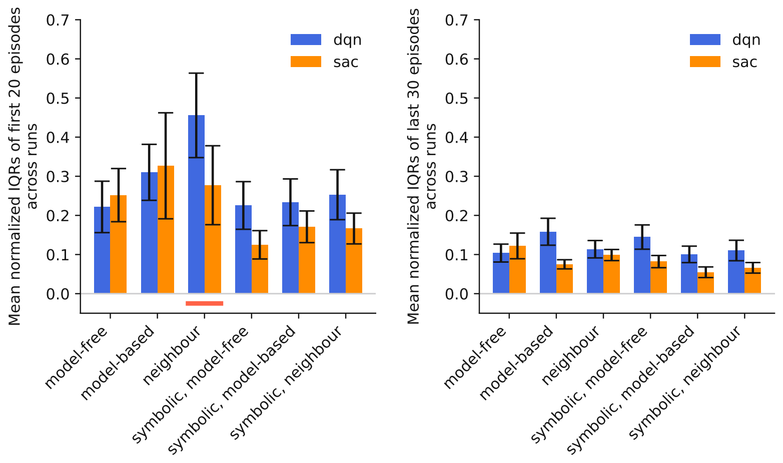


Figure 4.25: Direct comparison of mean normalized IQRs across runs for the first 20 (*left*) and last 30 episodes (*right*) of training runs from DQN and SAC agents.

When focusing on the IQR of performance during runs in Figure 4.26, a mixed impression appears. DQN agents again exhibit a slightly higher normalized IQR during runs over the first 20 episodes with the model-free non-symbolic agents being an exception. The symbolic agents also do not exhibit large differences. Over the last 30 episodes, the differences become mostly smaller.

In summary, the SAC algorithm proves to be more adept at learning from the crop rotation environment, consistently delivering superior performance across the training runs. Particularly the symbolic variants demonstrate rapid learning with few samples and converge to high average performances in the long run. Notably, without the use of a symbolic planner to filter suitable actions, the model-free DQN agent and the DQN agent with neighbour experience converge to a higher mean performance than non-symbolic SAC agents after an initially more unstable start.

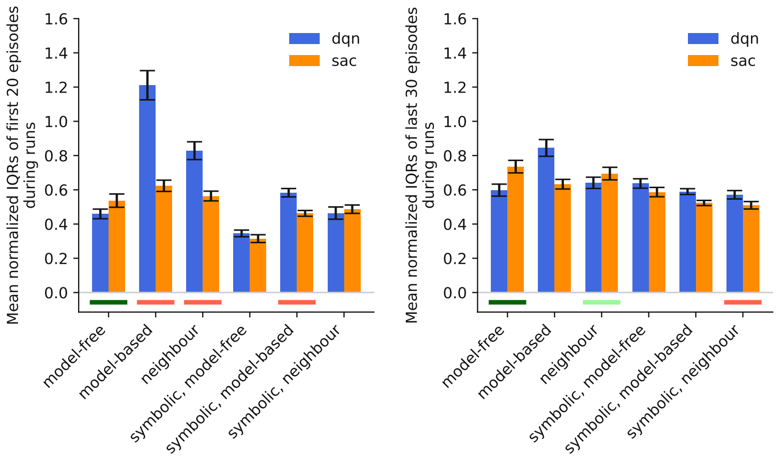


Figure 4.26: Direct comparison of mean normalized IQRs within windows of detrended training rewards during runs for the first 20 (*left*) and last 30 episodes (*right*) of training runs from DQN and SAC agents.

4.5 Research question 1.3: Profit and reliability differences between hybrid RL agents and symbolic planning

Research Question 1.3 aims to scrutinize performance variations among hybrid RL agents, such as the symbolic model-free DQN and SAC agents, in comparison to a purely symbolic planner. Figure 4.6 provides a visual representation of the bootstrapped mean total profits for these agents and the symbolic planner named "only filtered." Notably, there is a substantial disparity in total profits, with the hybrid agents demonstrating a significant performance superiority over the filtering mechanism based on crop rotation and cultivation rules (Figure 4.6). Although the symbolic planner outperforms the model-free hybrid agents significantly in the first 20 episodes (Figure 4.11), its average performance stagnates and is surpassed by the learning agents after about 30 episodes during training. Over the last 30 episodes, the mean performance of the hybrid agents is significantly higher than that of the symbolic planner. This trend is even more pronounced in extended training runs spanning 900 episodes, where symbolic RL agents continuously enhance their performance across the majority of episodes (Figure 4.10). Turning attention to reliability, an analysis of the normalized IQRs of performance across training runs reveals no significant difference between the symbolic planner and the hybrid agents. However, the hybrid agents exhibit a significantly lower normalized IQR of detrended performance during runs compared to the symbolic planner. It is important to note that this difference may be attributed to normalization, given the utilization of higher 95%-quantiles of performance by the hybrid agents for this purpose.

In summary, the findings underscore the substantial performance advantages of hybrid RL agents over the symbolic planner. While the symbolic planner initially outperforms the model-free hybrid agents, their learning capabilities enable them to surpass the symbolic planner in later episodes. This performance superiority is particularly pronounced in extended training runs, demonstrating the continuous improvement of symbolic RL agents.

4.6 Research question 1.4: Profit and reliability differences between model-based and model-free agents

Research question 1.4 focuses on examining disparities in performance between model-based and model-free agent types. An examination of the bootstrapped mean total profits after 180 episodes (Figure 4.6) reveals that model-based symbolic agents outperform their model-free symbolic counterparts. However, it is observed that model-based non-symbolic agents exhibit no discernible improvement in performance and, in fact, manifest poorer performance than their model-free non-symbolic counterparts.

Statistical evaluation of these performance distinctions underscores the significance of our findings. Specifically, the symbolic model-based SAC agent demonstrates a superior performance, with high statistical significance, in comparison to its model-free variant when assessing total profits in Figure 4.27. Conversely, the model-based non-symbolic DQN agent exhibits a significantly inferior performance compared to its model-free counterpart. For both symbolic DQN and non-symbolic SAC variants, no statistically significant differences in performance between the model-free and model-based versions are evident across all runs.

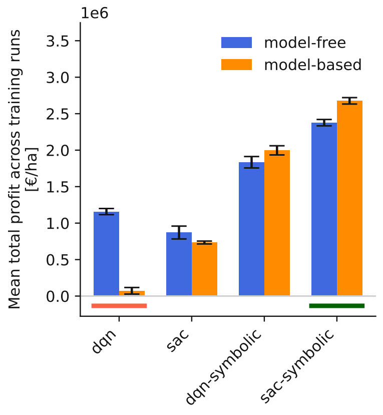


Figure 4.27: Direct comparison of mean total profits across training runs from model-free and model-based agents.

A closer inspection of average profits during the initial 20 and final 30 episodes of each run, as presented in Figure 4.28, provides a nuanced perspective. Model-based symbolic variants of the SAC and DQN agent swiftly acquire a useful policy, demonstrating significantly better performance in the initial 20 episodes (Figure 4.11) compared to their model-free counterparts. The main reason behind that could be the pretraining of the dynamics model which leads to the simulated trajectories being precise and aiding the agent in learning policies faster. As training progresses, model-free variants begin learning suitable policies, with the model-free symbolic DQN agent exhibiting significantly higher average performance in the last 30 episodes of the training run. This trend holds true for longer training runs with 900 episodes, where it outperforms its model-based counterpart by a substantial margin. For non-symbolic agents, there is no significant average performance difference in the first 20 episodes. However, both model-free variants significantly outperform their model-based counterparts in the last 30 episodes (Figure 4.12). A reason might be that the non-symbolic agents usually gather much unrewarding experience in early episodes which is then used to train the dynamics model. Without the filtering mechanism of the symbolic planner, no contrasting positive experiences are gathered and the dynamics model never learns to simulate rewarding transitions.

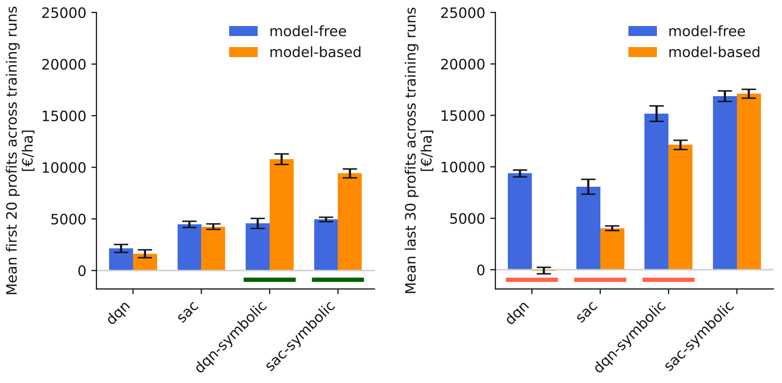


Figure 4.28: Direct comparison of average training rewards per episode for the first 20 (*left*) and last 30 episodes (*right*) of training runs from model-free and model-based agents.

An examination of the reliability of model-based agents yields additional insights. There are no significant differences in the IQR of performance across runs between model-based and model-free agents, irrespective of observing the first 20 or the last 30 episodes (Figure 4.29). Considering the intra-run IQRs of performance in Figure 4.30, model-based agents consistently exhibit significantly higher values than their model-free counterparts over the first 20 episodes. This might be influenced by the dynamics model's need to adapt from pretrained experience to the current environment. During this adaptation period, the simulated transitions might vary more strongly from actual experience than in the middle of the training run. During the last 30 episodes, this effect is not observed anymore although the model-free non-symbolic DQN agent demonstrates a significantly lower intra-run IQR of performance than its model-based counterpart. This is probably affected by the generally inferior performance of the model-based non-symbolic DQN agent with a continuously deteriorating mean performance over the training run.

In summary, it can be asserted that the superior performance of model-based symbolic agents stems from their early utilization of the dynamics model to refine policies. Over

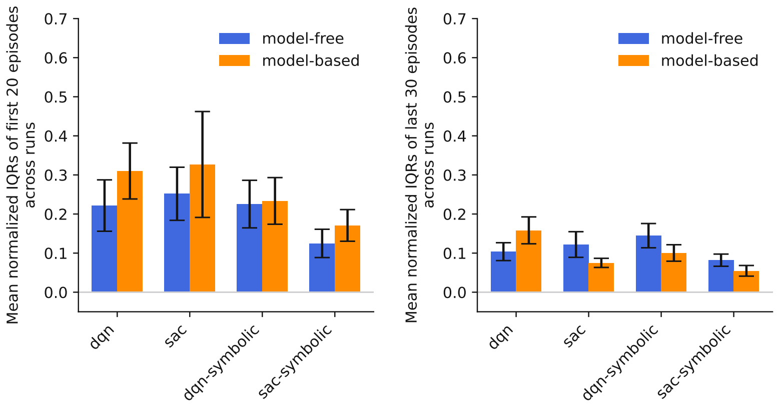


Figure 4.29: Direct comparison of mean normalized IQRs across runs for the first 20 (*left*) and last 30 episodes (*right*) of training runs from model-free and model-based agents.

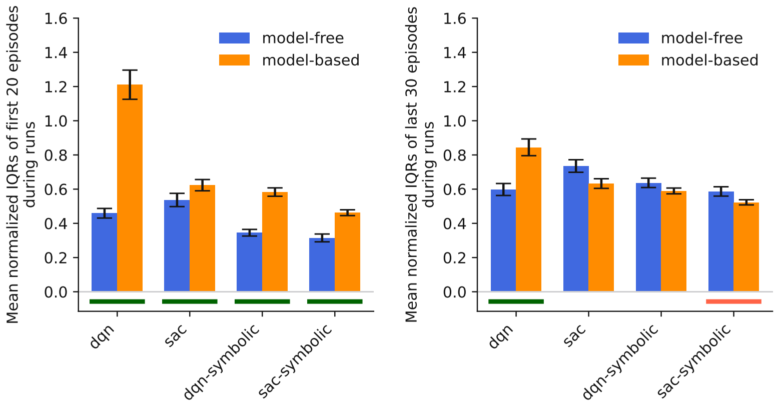


Figure 4.30: Direct comparison of mean normalized IQRs within windows of detrended training rewards during runs for the first 20 (*left*) and last 30 episodes (*right*) of training runs from model-free and model-based agents.

time, model-free variants learn policies that can match or even surpass their model-based counterparts, as exemplified by the symbolic DQN agent. In contrast, non-symbolic agents do not exhibit proficiency in learning suitable policies when coupled with a dynamics model. Although the theoretical expectation is that using a dynamics model would enhance the reliability and stability of agents, the observed interquartile ranges of performance during runs suggest that pretrained dynamics models contribute to increased performance variance in early episodes. This might primarily be due to the need for fine-tuning of the dynamics model in the early episodes of the training runs which can lead to partially misleading agent updates.

4.7 Research question 1.5: Profit and reliability improvements when using experience from neighbouring farms

In research question 1.5, the objective is to discern the impact of agents leveraging experience acquired from adjacent plots or neighbouring farmers within their vicinity. An evaluation of the bootstrapped mean total profits across all runs, depicted in Figure 4.6, illuminates the influence of neighbour experience on symbolic and non-symbolic agents. Both symbolic agents exhibit a significant performance boost when incorporating neighbour experience, evidenced by a notably higher total reward across all episodes compared to their counterparts devoid of updates from neighbouring farms. Conversely, the non-symbolic agents encounter challenges in learning from transitions not directly obtained from their own environment which is alike to observations made with model-based variants. Specifically, the non-symbolic DQN agent demonstrates a significantly lower mean total reward when utilizing neighbour experience. In contrast, the non-symbolic SAC agent with neighbour experience exhibits a marginally improved and less variable mean total reward, although this improvement is not statistically significant compared to its variant without neighbour experience.

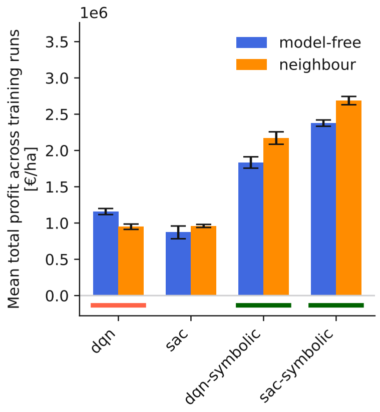


Figure 4.31: Direct comparison of mean total profits across training runs from agents with and without utilization of neighbour experience.

A closer examination of cumulative reward curves in Figure 4.7 unveils distinct performance profiles. The symbolic DQN agent utilizing neighbour experience initially outperforms its variant relying solely on direct updates, experiences a brief dip in average performance, but subsequently demonstrates accelerated improvement in the middle of the training run. Over the initial 20 training episodes, it achieves a significantly superior average performance compared to its version without neighbour updates, but by the end of the training run, the average performance aligns with that of its variant without neighbour updates, with no significant difference observed in Figure 4.32. The symbolic SAC agent consistently exhibits more stable learning with neighbour updates, outperforming its counterpart without such updates significantly in the first 20 episodes but showing no significant performance difference by the end of the training run. Conversely, the non-symbolic DQN agent performs significantly better without neighbour updates throughout the entire training run. For the non-symbolic SAC agent, no significant differences in performance are noted over the first 20 and last 30 episodes. Examining longer training runs reveals that the stabilization effect of using neighbour updates leads to worse average performance in the later stages of training. Only the symbolic SAC agent benefits from the neighbour updates even in the very late stage of training.

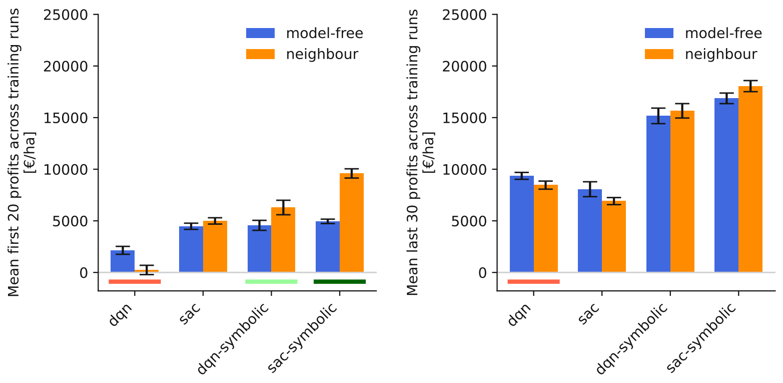


Figure 4.32: Direct comparison of average training rewards per episode for the first 20 (*left*) and last 30 episodes (*right*) of training runs from agents with and without utilization of neighbour experience.

Considering reliability, no significant differences in IQRs of performance across runs (Figure 4.33) are observed with one exception: the non-symbolic DQN agent using neighbour experience exhibits a significantly higher IQR across runs for the first 20 episodes of training. This discrepancy may be attributed to performance deterioration in the early stages of training in some runs, leading to heightened variance in performance across runs. Analysing intra-run performance variance in Figure 4.34, the utilization of neighbour experience results in significantly higher IQRs of detrended performance in the early episodes for all agents except the non-symbolic SAC agent. This phenomenon is explained by the need for agents to adapt from neighbour experience to experience obtained in their own environment, introducing performance differences between episodes. In later episodes of the training run, this effect has mostly diminished.

In summary, leveraging neighbour experience proves advantageous for agents already learning in a stable manner by using a symbolic planner. However, as agents accumulate more experience from their actual environment over time, this advantage diminishes for the symbolic DQN agent. Additionally, the performance variance within runs during early training is higher when using neighbour experience as agents need to adapt from neighbour experience to their own farm's experience.

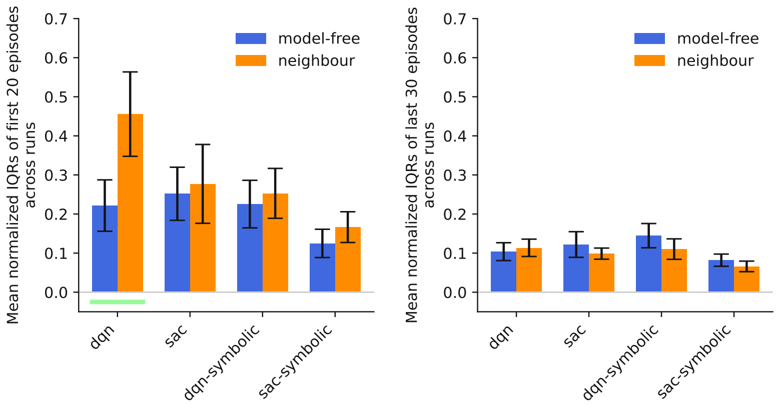


Figure 4.33: Direct comparison of mean normalized IQRs across runs for the first 20 (*left*) and last 30 episodes (*right*) of training runs from agents with and without utilization of neighbour experience.

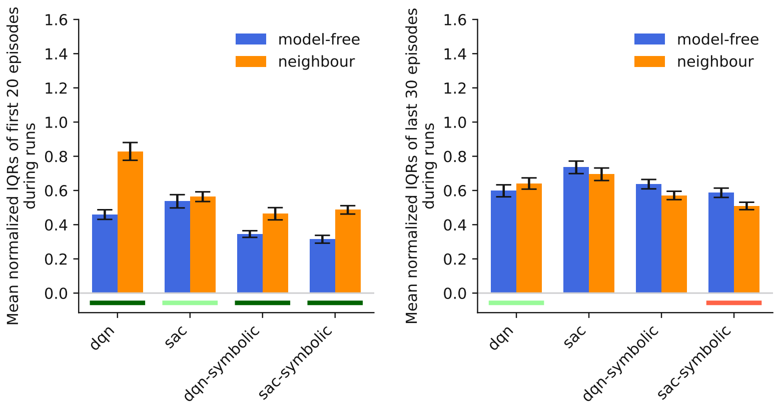


Figure 4.34: Direct comparison of mean normalized IQRs within windows of detrended training rewards during runs for the first 20 (*left*) and last 30 episodes (*right*) of training runs from agents with and without utilization of neighbour experience.

4.8 Research question 1.6: Profit and reliability improvements when comparing agents with or without pretraining

Research question 1.6 investigates the comparison between agents pretrained on a distinct environment and subsequently deployed on the target environment against agents without any pretraining. Pretrained agents underwent 180 episodes of learning on an environment with a similar ground type and humidity but different average yields. Following this pretraining, they were deployed on the target environment and continued training for an additional 90 episodes. Analysing the means of cumulative profits across runs on the target environment provides insights into the advantages of allowing agents to pretrain before deployment. Symbolic agents in particular benefit significantly from pretraining, likely attributed to their ability to learn a beneficial policy during pretraining, a benefit not assured for non-symbolic agents with more unstable training performances. Figure 4.35 and Figure 4.36 illustrate that symbolic agents which acquired reasonable policies during pretraining could build upon them in the subsequent test runs.

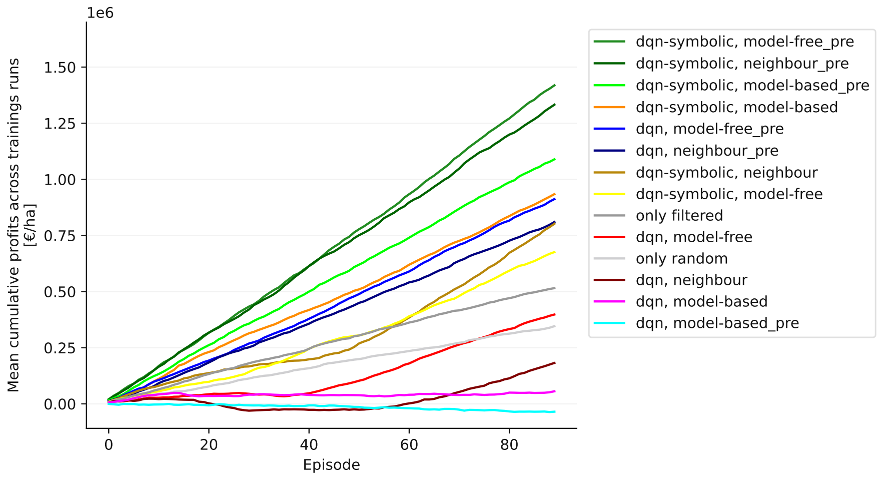


Figure 4.35: Mean cumulative profits across training runs for all DQN agents with and without pretraining.

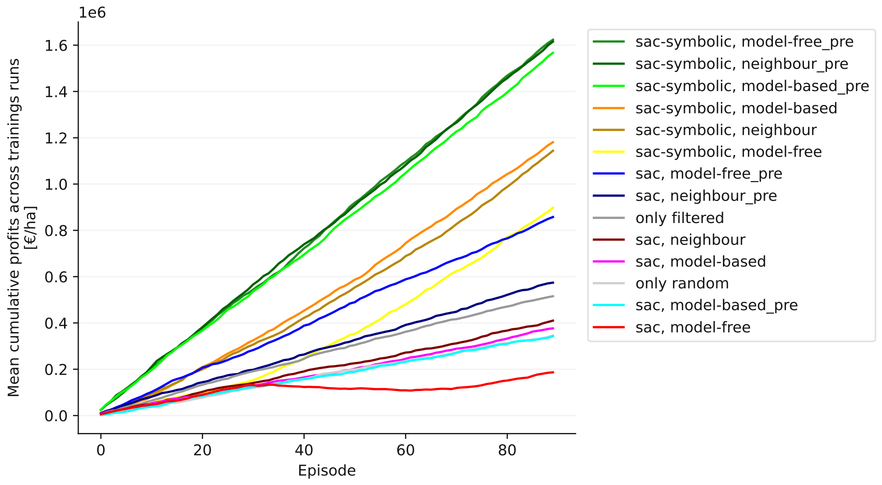


Figure 4.36: Mean cumulative profits across training runs for all SAC agents with and without pretraining.

Significant total reward differences between pretrained and non-pretrained counterparts are observed for those agents, with model-free agents exhibiting the highest performance disparities due to their greater instability but more thorough exploration which is advantageous over extended training periods and when adapting between pretraining and target environment policies. (Figure 4.37). Non-symbolic agents present a more nuanced perspective. Agents that struggled to find suitable policies during pretraining, such as model-based DQN and SAC agents, performed even worse with pretraining. Conversely, agents with an initially unstable start during pretraining but improvement in later steps, such as model-free DQN and SAC agents with and without neighbour experience, leveraged the pretraining experience in target runs, exhibiting significant differences in total profits between pretrained and non-pretrained agents.

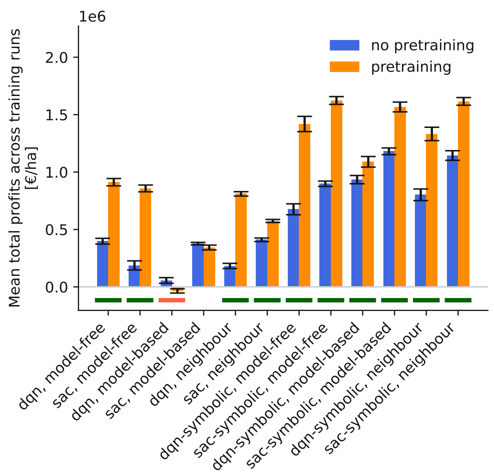


Figure 4.37: Direct comparison of mean total profits across training runs from all agents with and without pretraining.

Confirmation of this impression is found in mean profits over the first 20 episodes in Figure 4.38, revealing significant positive differences between pretrained and non-pretrained agents for all symbolic agents except the model-based DQN agent. Among non-symbolic agents, only the model-based variants exhibit a negative effect from pretraining. In the last 30 episodes, albeit smaller, differences in mean performance between pretrained and non-pretrained agents remain significant, with only the symbolic model-based DQN agent and the DQN agent with neighbour experience showing no significant differences.

While the IQRs of performance across runs (Figure 4.39) show no stable trend between pretrained and non-pretrained agents, the IQRs of performance during runs provide a different perspective in Figure 4.40. Non-symbolic agents without pretraining display a higher performance variance during early episodes, indicating the need to learn a suitable policy. Conversely, symbolic model-free agents exhibit a higher performance variance during early episodes when pretrained than when not pretrained. These differences diminish during the late episodes of the target run, with pretrained symbolic model-free agents even displaying a lower performance variance than their non-pretrained counterparts.

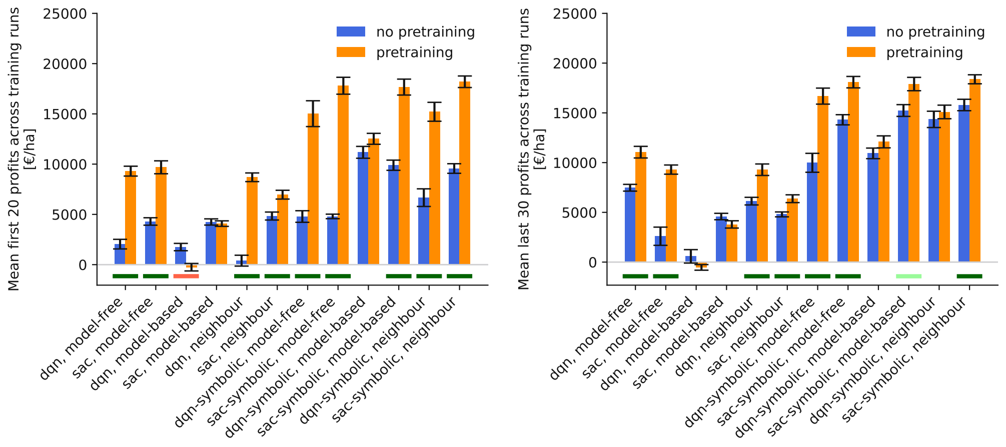


Figure 4.38: Direct comparison of average training rewards per episode for the first 20 (*left*) and last 30 episodes (*right*) of training runs from all agents with and without pretraining.

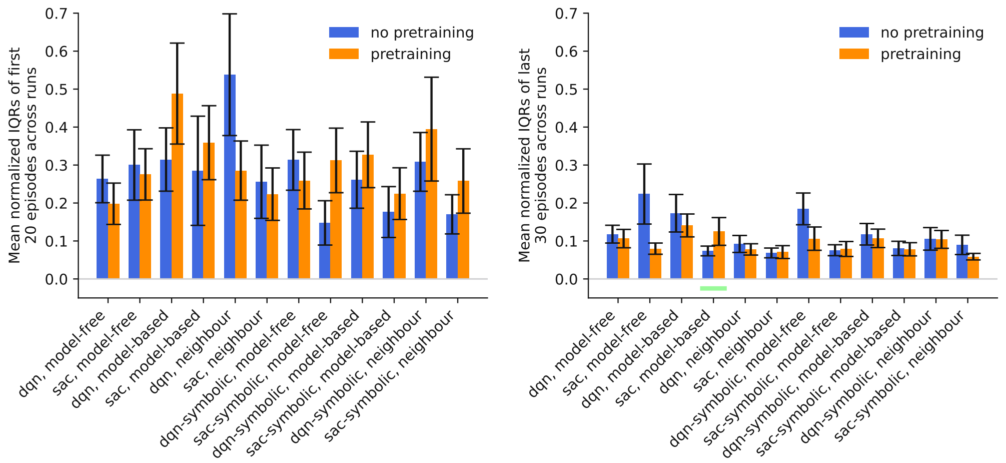


Figure 4.39: Direct comparison of mean normalized IQRs across runs for the first 20 (*left*) and last 30 episodes (*right*) of training runs from all agents with and without pretraining.

In summary, pretraining proves highly beneficial for agents capable of learning rewarding policies in another environment. Agents with more unstable learning or no learning at all struggle to derive significant benefits from additional learning time. The positive effect of pretraining remains impactful during later stages in the target environment for most agents.

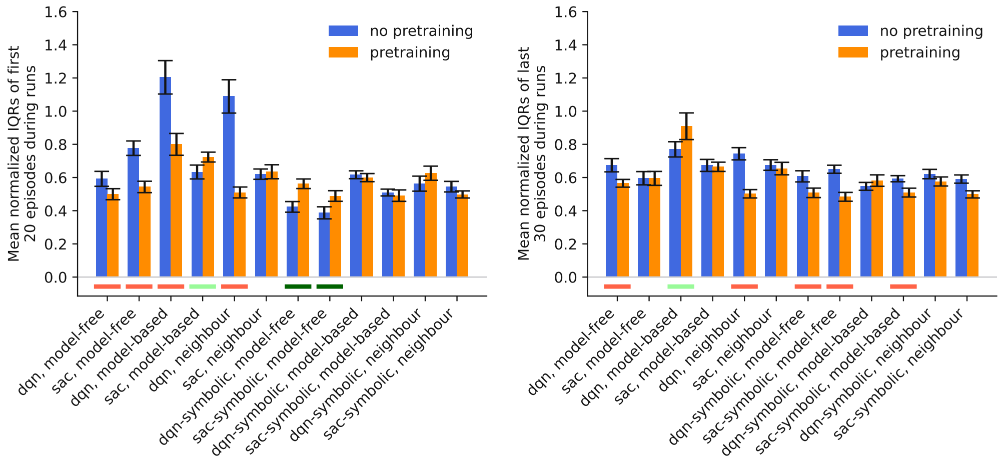


Figure 4.40: Direct comparison of mean normalized IQRs within windows of detrended training rewards during runs from agents with and without utilization of neighbour experience.

4.9 Research question 2: Comparison of crop selection diversity between SAC and DQN agents

For the final research question, we investigate whether the utilization of the probabilistic action selection of policy gradient reinforcement learning agents, such as the SAC algorithm, results in a higher crop diversity compared to the use of semi-deterministic agents like the DQN agent, which selects random actions with a probability ϵ. Crop diversity in a training run is quantified using the Shannon entropy, with the highest possible entropy nearing 3.18, reflecting equal selection probabilities for each crop. Entropy values below 2.5 indicate a scenario where only half of the possible crops are equally selected, while the other half is never chosen. The overall high entropy values are attributed to agents being trained on various 10-step episodes, each starting with different prices and costs, rendering different crops viable in different episodes. A first comparison between DQN and SAC agents in Figure 4.41 suggests that non-symbolic SAC agents exhibit a slightly higher entropy across runs than their DQN counterparts, with mixed results for symbolic agents. Notably, non-symbolic agents, in general, demonstrate a significantly broader diversity of crops, consistent with their less restricted action space and the option to select actions breaking crop rotation rules. Plotting the mean total rewards against the mean entropy across runs in Figure 4.42 demonstrates a correlation between a lower entropy and a higher total performance. Agents with a well-learned policy tend to stick to it, resulting in a lower crop diversity. Symbolic SAC agents stand out with a higher reward-to-entropy ratio than the average agent, suggesting that SAC agents with probabilistic action selection can achieve a high crop diversity while maintaining high total profits. In contrast, DQN agents are positioned at or below the regression line, indicating a lower reward-to-entropy ratio than the average.

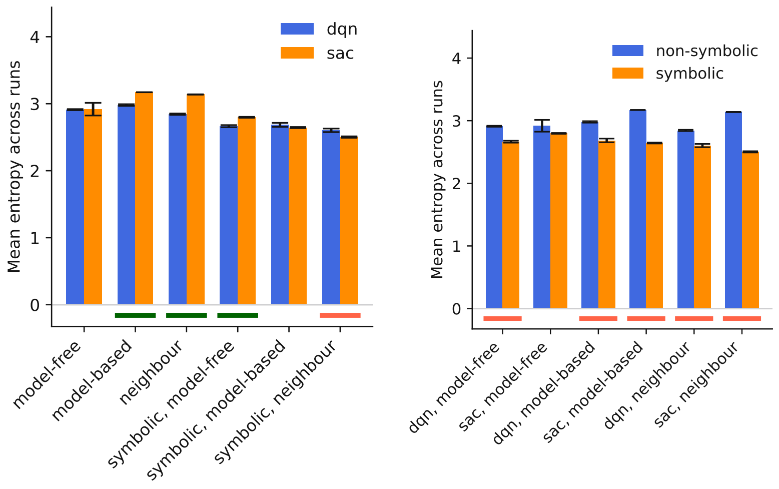


Figure 4.41: Direct comparison of mean entropy across runs from all agents.

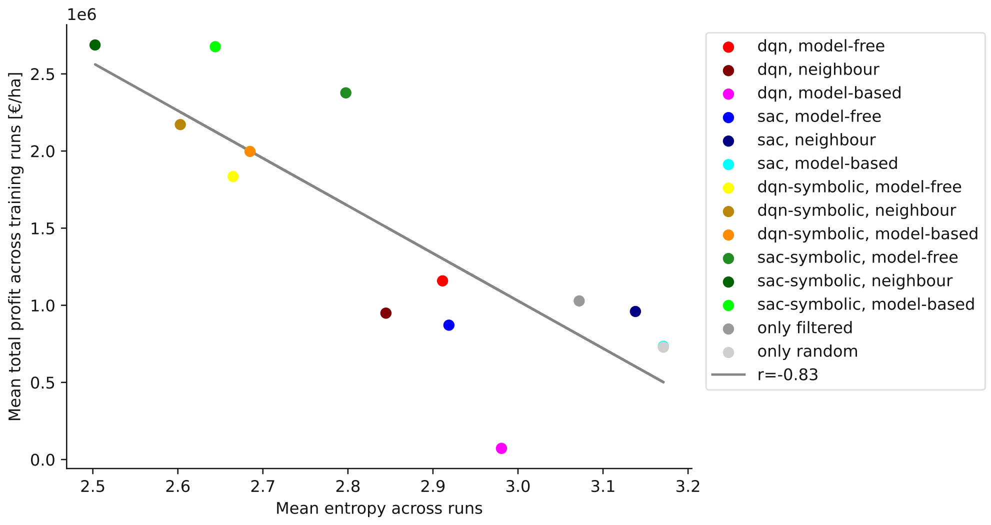


Figure 4.42: Scatter plot of mean total profits across training runs against mean entropies across runs for all agents.

A detailed examination of entropy across runs for the first 20 and the last 30 episodes provides further insights into agents' action selection behaviour in Figure 4.43. SAC agents consistently exhibit a higher mean entropy across runs in the initial 20 episodes compared to their DQN counterparts. In the last 30 episodes, mean entropy values largely align with those measured over all episodes. The higher entropy in the early episodes is an effect of the probabilistic policy which has not yet learned higher weights for more rewarding actions. Conversely, even with a high initial ϵ value in DQN training, the tendency toward exploration decreases over the first 20 episodes, resulting in a lower crop diversity. This stronger inclination toward exploration may also contribute to symbolic SAC agents learning suitable policies more rapidly, evident in their higher total profits compared to symbolic DQN agents.

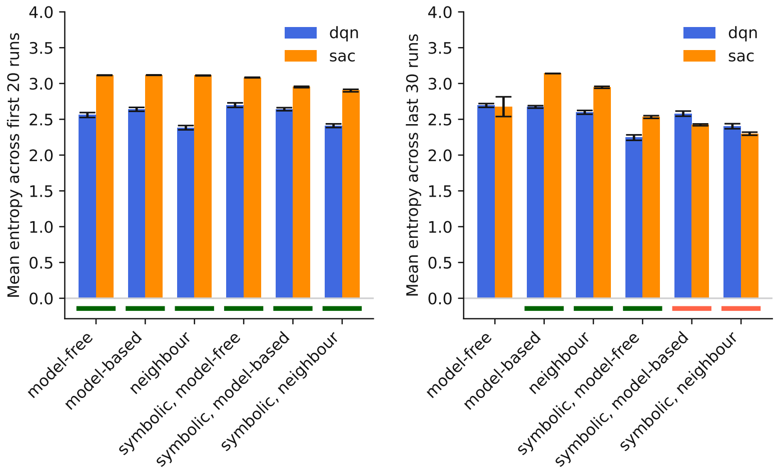


Figure 4.43: Direct comparison of mean entropy across runs for the first 20 (*left*) and last 30 (*right*) episodes from all agents.

In summary, a clear trend emerges, indicating that SAC agents generate a higher crop diversity during training with their probabilistic action selection than the DQN agents. Furthermore, a correlation is established between high total profits and lower mean entropies across runs, underscoring that agents with well-established policies tend to select rewarding actions more frequently, whereas agents struggling to find effective strategies continue to explore different actions in the later stages of training.

CHAPTER 5

Summary

In this final chapter, the findings from the project are summarized in section 5.1, contextualized within the current research field in section 5.2, and the limitations of this work are stated in section 5.3. Possible future research objectives building upon the findings from this project are identified. The results lead to a positive recommendation towards the use of hybrid reinforcement learning agents in decision support systems to recommend crops to farmers. Some uncertainty remains due to the use of a simulation environment instead of training agents on a real environment, but the results promote future research following a similar direction.

5.1 Conclusion

In this section, the results and insights obtained during this project are summarized. Previous approaches addressing crop rotation optimization focused on static problem formulations and environments and could give general guidance on which crop rotation sequences are suitable to maximize yields and profit. Many utilized rule-based algorithms or mathematical modelling to solve the problem. A more modern approach by Fenz et al. introduced Reinforcement Learning to the problem field [FNFW23]. Their solution showed great performance in selecting suitable crops to cultivate next and in generating viable crop rotation sequences after being trained on a simulation environment. However, the agent needed many episodes of rather unstable learning to converge to this performance in most training runs. As a single step in an environment represents a full year, an agent's need for several 1000 episodes to converge to a good performance makes it unviable to train under real-life conditions. Furthermore, the simulation environment the RL agents trained on had static rewards and did not incorporate yield and price uncertainty from real-life environments. The goal of this thesis project was therefore to find out if it is possible to implement an RL agent that can adapt to individual farm conditions, can deal with exogenous uncertainty, is sample efficient enough to reduce the need for real experience by a large factor and only select crops during exploration which would not violate any expert's advice or contradict common farming knowledge. The proposed solution which was implemented and evaluated during this thesis project contained the following features:

- Application of a modern Reinforcement Learning algorithm to learn which crops are optimal to grow in different states.
- Integration of a symbolic planner into the Reinforcement Learning process to guide exploration and avoid highly detrimental actions.
- Addition of a dynamics model which can simulate further transitions for the agent to learn from and incorporate the uncertainty of a real environment in its simulations.
- Utilization of experience from other farms to speed up learning.
- Pretraining of agents on simulated experience before deployment.

The implemented solutions containing those features were compared with baselines to evaluate the effectiveness of the proposed features on performance and reliability. Across various simulated environments, all proposed features demonstrated enhancements in performance throughout different training runs. Furthermore, the majority of these features exhibited stable learning patterns, with runs in the lowest 30% quantile of performance only marginally lagging behind the agents' mean performances across all runs. However, some agent types did not acquire any knowledge about which crops are beneficial towards farming profit. Delving deeper into the evaluation, all research questions were answered thoroughly. Research question 1 focused on potential performance improvements when using the proposed features in comparison to a baseline and was divided into six sub-questions:

RQ 1.1: By how much do hybrid systems combining a symbolic planner with an RL agent show a better performance on crop rotation problems than simple RL agents without rule-based planning?
The biggest positive impact on performance was achieved by the introduction of a symbolic planner to restrict the exploration space and improve the precision of next-state target value predictions in the RL algorithms. Agents learned significantly faster by selecting beneficial actions early and more often. A negative bias for all action-values from the early selection of many unrewarding actions could therefore be avoided. This assumedly had a positive impact on the precision of policy evaluation during the training runs. Although it was theorized that the restriction of the action space would lead to a lower mean performance after convergence than when using unrestricted agents, the hybrid agents outperformed their non-symbolic counterparts even in late episodes of training runs.

RQ 1.2: To which extent can the use of more modern RL learning algorithms like soft actor-critic learning improve performance when compared to deep Q learning?
The comparison between soft actor-critic and deep Q learning agents demonstrated that the use of the SAC algorithm proved to be more beneficial in combination with a symbolic planner. Particularly during late stages of training, it exhibited a significantly better mean performance than its DQN counterpart with a rather equal performance during early episodes of training runs. Without the restricted action space, it was the opposite way. The SAC agents performed better in early stages while becoming relatively worse to the DQN variants in late training episodes. The theory behind is that non-symbolic SAC agents already have a more guided exploration in early episodes due to the difference in action selection probabilities while the DQN agent explores completely randomly if the ϵ-condition is met. In terms of reliability, both learning algorithms exhibited similar results.

RQ 1.3: In which situations and by how much do hybrid systems combining a symbolic planner with an RL agent show a better performance on crop rotation problems than only using a symbolic planning system like answer set programming to select crops to grow?
The symbolic planner was used to filter for actions that would not violate any crop rotation rules or have a detrimental effect on soil conditions. To differentiate between the performance improvement achieved by using this filtering mechanism and the performance improvement from the RL agent learning on the environment, a comparison was made between randomly selecting actions from the filtered action space as a baseline and letting the RL agents select actions from the filtered action space. Regarding total performance, all hybrid RL agents outperformed the baseline significantly which demonstrates a positive effect of using RL agents beyond rule-based planning. While the baseline exhibited a constant mean performance, the RL agents improved over the course of the training runs. However, the model-free symbolic RL agents were outperformed by the random filtered action selection in the first 20 episodes, when the agents did not yet learn enough from the environment to select beneficial actions. Hybrid agents utilizing neighbour experience or a dynamics model did not have those cold start difficulties and immediately exhibited a higher mean performance than the baseline.

RQ 1.4: To which extent can performance be improved by using model-based RL when compared to model-free methods?
In research question 1.4, it was evaluated by which amount performance can be improved by supporting the RL algorithms with a pretrained dynamics model simulating experiences that the agents can use to train more. In theory, using model-based RL helps to stabilize training although it can hamper performance improvement if the dynamics model does not reflect the real environment precisely enough. The results showed that the non-symbolic model-based agents performed particularly badly. A reason might be that the dynamics model is only fed unrewarding experiences and generalizes badly for potentially rewarding state-action pairs. The agents then only receive negative reinforcement and never become able to differentiate between good and bad actions. In contrast, the

symbolic model-based agents performed well. Their main advantage was observed in the early episodes during training runs where they allowed the agent to already train while the replay buffer for real experiences was being filled. In later episodes, this advantage diminished and the stabilization effect led to the model-based agents converging to lower mean performances than their model-free counterparts.

RQ 1.5: To which extent can agent updates from neighbouring farms improve performance in comparison to agents only updating from their own farm's experience?
The effects from using experience from neighbouring farms were in many ways similar to the effects from using a dynamics model. Non-symbolic agents had trouble making use of neighbour experience while symbolic agents could stabilize and improve their own learning during the early stages of training. In contrast to the agents using a dynamics model, many agents utilizing neighbour experience did not exhibit negative effects from this feature in later episodes. A reason for that could be that direct experience from neighbouring fields and plots of land will never be extremely far from reality whereas unrealistic transitions simulated by a faulty dynamics model would have a much bigger negative impact on learning.

RQ 1.6: To which extent can agents already pretrained on other environments improve performance when compared to agents without pretraining?
In research question 1.6, the effect of pretraining agents on another environment before deployment was evaluated. Agents that were pretrained on a different environment and that were able to learn a viable policy for this environment could transfer their embedded knowledge to other environments and profit from the pretraining. A high mean performance in the last episodes of the pretraining run allowed the agents to transfer the knowledge to the new environment with only little adaptation needed. Agents with a rather unstable training like non-symbolic or model-free agents could benefit the most from the pretraining step as the early instability during learning in the target environment could be avoided. Additionally, SAC agents could profit more from pretraining than their DQN counterparts.

RQ 2: To which extent can the use of probabilistic action selection by the RL agent improve diversity of crop selection when compared to deterministic algorithms?
Research question 2 was about evaluating if the probabilistic action selection mechanism of SAC agents led to a higher crop diversity which is generally thought to be beneficial for the soil condition and the ecosystem around the plot of land. The crop diversity was measured via the Shannon entropy. The results demonstrated that most training runs resulted in a relatively high crop diversity when compared to a monoculture. A general trend was that higher training performances resulted in a lower crop diversity. A reason might be that well-performing agents focus on selecting the best options while disregarding sub-optimal crops. Agents with a less viable policy select more exploring options which results in a higher crop diversity over the run. In addition to this information, the

research question could be answered and the theory behind it could be confirmed: it was observed that SAC agents have a higher reward-to-entropy ratio than DQN agents which means that they select a larger variety of crops while maintaining a high performance. Particularly during early episodes, the crop diversity of SAC agents is significantly higher.

To summarize the findings, it can be noted that all proposed features of the hybrid RL algorithm showed better performance when compared to a baseline not containing the respective features. The symbolic planner in particular proved to a useful addition to stabilize training in early episodes and could improve the cumulative profits of the agent during deployment significantly. The more modern SAC algorithm proved to be better than the DQN baseline, especially when combined with the symbolic planner. The addition of a dynamics model simulating transitions proved to be useful in early episodes but was detrimental in later training. A similar observation was made with the introduction of neighbour experience. An advantage of this feature was however that no negative effects were observed in the later episodes of training runs. Pretraining the agents proved to be beneficial for all agents that were able to improve during training runs. Regarding reliability of different agent types, it could be observed that agents with a lower cumulative performance exhibited a slightly higher normalized performance variance across runs. This variance was especially high during early episodes of training runs. Instability during runs was also relatively high for many agents. An explanation is the inherent uncertainty of the simulation environment which leads to different profits under the same conditions and cultivated crop sequences. Hybrid agents could significantly lower this intra-run variance with the support of a symbolic planner. The use of neighbour experience made the first 20 episodes in training runs less stable as agents needed to adapt from the foreign experience to its own farm's experience.

A recommended setting to apply agents to real-life environments would be to choose an agent combining the soft actor-critic learning algorithm with a symbolic planner reflecting expert knowledge and crop rotation rules. If possible, it would be beneficial to include information about cultivated crops, yields and field conditions from the region and integrate it into the agent training in form of a replay buffer for neighbour experience. Another option to guarantee a higher stability and good performance immediately after deployment would be to pretrain the agent in a simulation environment which is as realistic as possible in predicting transitions and yields from different crops. If the adoption of such a model as a decision support system would be more widespread, reusing an agent already deployed and trained on another plot of land with similar conditions would also be a viable strategy. The model-based approach exhibited mixed results. Although the cumulative performance and particularly the mean performance in early episodes was significantly better when using a dynamics model, this was only achievable after pretraining the dynamics model before the training run. The stabilization effects which were the main reason to incorporate a dynamics model into the algorithm could not be observed. It is therefore questionable if the use of a model-based approach with a dynamics model pretrained on experience is superior to using the available experience to pretrain the agent directly. The comparison between using neighbour experience and using a dynamics model showed that only the use of a dynamics model led to a performance stagnation of SAC agents in the late stage of training. It is not recommended to immediately deploy model-free hybrid agents to a real environment without any kind of pretraining on a semi-realistic simulation environment. The results demonstrated that model-free agents had cold start issues even with the use of a symbolic planner. The main reason behind that is the empty replay buffer which can only be used for training after the number of gathered experiences is larger than the sample batch size.

5.2 Contributions to Research Field

Farmers around the world need to consider every year which crops to plant on their field. To make a thoughtful decision, they consider many exogenous factors like soil and climate conditions, weed & pest occurrences in previous years, market prices and costs, and the crop cultivation histories of their fields. Present research tries to support farmers in this endeavour by creating models and planners which can generate viable crop rotation sequences and optimize crop selection for yield and profit. As many approaches use static information and rule-based systems, their useability in reactive decision support systems is limited. A recent advance by Fenz et al. in using Reinforcement Learning to find optimal crop rotation sequences showed promising results [FNFW23]. However, their goal to generate suitable sequences after training the RL agents until convergence interfered with the ability to integrate those agents directly into systems adapting to live data and changing conditions. The need for several thousand training episodes until convergence made the agents unfit for decision support in an untrained state. In this thesis, it was researched if it is possible to improve the sample efficiency and stability of the current state-of-the-art RL algorithm in such a way that it might be suitable to be integrated into a live support system for crop rotation decisions. Several features were proposed and evaluated: The use of more modern RL algorithms, a hybrid approach combining reinforcement learning and symbolic planning, model-based reinforcement learning, the utilization of foreign experience and the pretraining of agents before deployment. As the evaluation on actual plots of land would take many years, the agents were trained and tested on simulation environments comparable to the one used by Fenz et al. Since the simulation environment they used did not encompass the full complexity of real-life conditions and all effects of different crop sequences, it was modified and extended with new crop rotation and soil condition effects derived from the literature. The results from the evaluation demonstrated that it is indeed possible to construct RL agents which could be deployed as decision support systems for farmers. By using a soft actor-critic learning algorithm, combining it with a symbolic planner to restrict the crop selection to viable choices and stabilizing early training with knowledge from other crop rotation experiments, the agent showed steady performance improvements with a low risk for the farmer to cultivate highly unsuitable crops on their fields. This result fortifies the position of reinforcement learning as a viable solution to the field of crop rotation optimization and opens new paths for future research.

5.3 Limitations and Future Work

While the results of this project are promising, some methodical details leave room for uncertainty which can be closed by future research. To begin with, there will always be room for improvement when using simulation environments. Real-life effects on crop yields and market conditions are more complex than their reflected versions in simulations. Many exogenous factors like weather, market dynamics, soil characteristics or water and nutrient availability can be quantified in some level of detail but will always behave differently and less predictable in the real world than in a simulated system. As an example, the simulated rules and assumptions with an effect on crop yield used in the project's experiments were mostly static and linear; the factors for yield reductions induced by the violation of crop rotation rules were only deducted by logical reasoning and were not extracted from reliable experimental research. Future research could address these issues and focus on implementing a more precise simulation environment to see if the results from this project can be reproduced in those scenarios too. Many crop dynamics tools like DSSAT are available open source to simulate crop yields under different conditions and could be used as further input for a training environment [HPB+19]. Besides potential improvements in precision, an extended simulation environment could encompass the cultivation of cover crops between main crops, a more detailed water and soil management and the addition of suitable farming operations for specific crop types. This extension would increase complexity a lot but also offer more guidance for inexperienced farmers when the agents are performing well.

Another field of evaluation would be to test agents pretrained on simulation environments using the proposed hybrid agent on small test plots of land in long-term experiments and find out if they reproduce the results obtained from this thesis project. A further limitation of this work stems from the assumption that experience from neighbouring fields to pretrain the model or a dynamics model in a model-based approach would already be available in the format used in this project. While there are possible ways to measure soil conditions like nutrient balance and humus ratio, the measurement process is often tedious and expensive. Many farmers therefore rely solely on their experience, market prices and the weather to select crops to cultivate and could not offer any substantial amount of reliable experience to use for pretraining. A dense aggregation of quantitative results from field experiments and case studies from the literature could be another future work to create a shared and central knowledge base available for future research efforts. Another advantage of this would be, that the standardized data could act as a benchmark to compare different proposed solutions in the field of crop rotation optimization. Further work in this research domain could encompass an enhancement of RL training algorithms, the integration of more model features improving sample efficiency, training speed and stability and a more standardized evaluation framework considering farmers' needs and success criteria for a viable decision support system.

List of Figures

List of Tables

List of Algorithms

Bibliography

[ABB$^+$23] Jonathon Abernethy, Peter Beeson, Claire Boryan, Kevin Hunt, and Luca Sartore. Preseason crop type prediction using crop sequence boundaries. *Computers and Electronics in Agriculture*, 208:107768, 2023.

[ACC00] Matthew L Adams, Simon Cook, and Robert Corner. Managing uncertainty in site-specific management: what is the best model? *Precision agriculture*, 2(1):39–54, 2000.

[AE08] Margrethe Askegaard and Jørgen Eriksen. Residual effect and leaching of N and K in cropping systems with clover and ryegrass catch crops on a coarse sand. *Agriculture, ecosystems & environment*, 123(1-3):99–108, 2008.

[agr23] Ernte 2023: Gerste, Weizen, Raps, Mais und weitere Feldfrüchte. AGRAVIS, 2023. *Accessed online (2024-02-18)*: `https://www.agravis.de/de/pflanzenbau/ernte/`.

[APS15] Laurent Alfandari, Agnès Plateau, and Xavier Schepler. A branch-and-price-and-cut approach for sustainable crop rotation planning. *European Journal of Operational Research*, 241(3):872–879, 2015.

[ASC$^+$21] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

[ATB17] Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. *Advances in neural information processing systems*, 30, 2017.

[BD15] Richard E Bellman and Stuart E Dreyfus. *Applied dynamic programming*, volume 2050. Princeton university press, 2015.

[Bel54] Richard Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515, 1954.

[Bel57] Richard Bellman. A Markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

[Bel66] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[BJL$^+$11] Jody Bolluyt, Sue Ellen Johnson, Peter Lowy, Margaret Tuttle McGrath, Charles L Mohler, Anusuya Rangarajan, Kimberly A Stoner, Eric Toensmeier, and Harold van Es. Crop rotation on organic farms: A planning manual, 2011. *Natural Resource, Agriculture, and Engineering Service (NRAES).*

[BM03] Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13:341–379, 2003.

[BNZ19] Onur Boyabatlı, Javad Nasiry, and Yangfang Zhou. Crop planning in sustainable agriculture: Dynamic farmland allocation in the presence of crop rotation benefits. *Management Science*, 65(5):2060–2076, 2019.

[Bri23] Encyclopædia Britannica. How agriculture and domestication began, 2023. *Accessed online (2024-02-18)*: `https://www.britannica.com/topic/agriculture/How-agriculture-and-domestication-began`.

[BSA83] Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

[BZ03] Johann Bachinger and Peter Zander. Planungswerkzeuge zur Optimierung der Stickstoffversorgung in Anbausystemen des Ökologischen Landbaus-Standort-und vorfruchtabhängige Kalkulation der N-Salden von Anbauverfahren. *Ressortforschung für den ökologischen Landbau 2002*, pages 21–30, 2003.

[BZ07] Johann Bachinger and Peter Zander. ROTOR, a tool for generating and evaluating crop rotations for organic farming systems. *European Journal of Agronomy*, 26(2):130–143, 2007.

[CA07] E.F. Camacho and C.B. Alba. *Model predictive control.* Springer Science & Business Media, 2007.

[CA13] Sivashan Chetty and Aderemi O Adewumi. Comparison study of swarm intelligence techniques for the annual crop planning problem. *Ieee transactions on evolutionary Computation*, 18(2):258–268, 2013.

[CCML18] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

[CFC+19] Stephanie CY Chan, Samuel Fishman, John Canny, Anoop Korattikara, and Sergio Guadarrama. Measuring the reliability of reinforcement learning algorithms. *arXiv preprint arXiv:1912.05663*, 2019.

[Chr19] Petros Christodoulou. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207*, 2019.

[Com91] European Commission. Council directive 91/676/eec of 12 december 1991 concerning the protection of waters against pollution caused by nitrates from agricultural sources. eu publications office, 1991. *Accessed online (2024-02-18)*: `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:31991L0676`.

[Com20] European Commission. Farm to Fork Strategy. EU Publications Office, 2020. *Accessed online (2024-02-18)*: `https://food.ec.europa.eu/horizontal-topics/farm-fork-strategy_en#Publications`.

[CR15] David W Crowder and John P Reganold. Financial competitiveness of organic agriculture on a global scale. *Proceedings of the National Academy of Sciences*, 112(24):7611–7616, 2015.

[CSO18] Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. How many random seeds? statistical power analysis in deep reinforcement learning experiments. *arXiv preprint arXiv:1806.08295*, 2018.

[CUM19] Phusanisa Charoen-Ung and Pradit Mittrapiyanuruk. Sugarcane yield grade prediction using random forest with forward feature selection and hyper-parameter tuning. In *Recent Advances in Information and Communication Technology 2018: Proceedings of the 14th International Conference on Computing and Information Technology (IC2IT 2018)*, pages 33–42. Springer, 2019.

[DHFLHvH22] Floris Den Hengst, Vincent François-Lavet, Mark Hoogendoorn, and Frank van Harmelen. Planning for potential: efficient safe reinforcement learning. *Machine Learning*, 111(6):2255–2274, 2022.

[DJ07] Nina K Detlefsen and Allan Leck Jensen. Modelling optimal crop sequences using network flows. *Agricultural Systems*, 94(2):566–572, 2007.

[DR11] Marc Deisenroth and Carl E Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[DRVI03] S Dogliotti, WAH Rossing, and MK Van Ittersum. ROTAT, a tool for systematically generating crop rotations. *European Journal of Agronomy*, 19(2):239–250, 2003.

[dSCAS10] Lana Mara R dos Santos, Alysson M Costa, Marcos N Arenales, and Ricardo Henrique S Santos. Sustainable vegetable crop supply problem. *European Journal of Operational Research*, 204(3):639–647, 2010.

[DSG$^+$12] Jérôme Dury, Noémie Schaller, Frédérick Garcia, Arnaud Reynaud, and Jacques Eric Bergez. Models to support cropping plan and crop rotation decisions. a review. *Agronomy for sustainable development*, 32:567–580, 2012.

[dSMAS11] Lana Mara Rodrigues dos Santos, Philippe Michelon, Marcos Nereu Arenales, and Ricardo Henrique Silva Santos. Crop rotation scheduling with adjacency constraints. *Annals of Operations Research*, 190:165–180, 2011.

[ECD18] Directorate-General for Agriculture European Commission and Rural Development. CAP explained : direct payments for farmers 2015-2020. EU Publications Office, 2018. *Accessed online (2024-02-18)*: `https://data.europa.eu/doi/10.2762/572019`.

[ESL19] Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

[EV20] Dhivya Elavarasan and PM Durairaj Vincent. Crop yield prediction using deep reinforcement learning model for sustainable agrarian applications. *IEEE access*, 8:86886–86901, 2020.

[FBM20] Andreas Fliessbach, Else K Bünemann, and Paul Mäder. Humus-Referenzwert für Ackerböden-Leitfaden für die Entwicklung eines Referenzwertes für den Vollzug. 2020. *Fachgruppe Vollzug Bodenbiologie (VBBio)*.

[FEG21] Ibrahim Fikry, Amr Eltawil, and Mohamed Gheith. A robust crop rotation optimization model with water scarcity and net return uncertainty considerations. *IEEE Access*, 9:128938–128950, 2021.

[FMS17] Carlo Filippi, Renata Mansini, and Elisa Stevanato. Mixed integer linear programming models for optimal crop selection. *Computers & Operations Research*, 81:26–39, 2017.

[FN71] Richard E Fikes and Nils J Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[FNFW23] Stefan Fenz, Thomas Neubauer, Jürgen Kurt Friedel, and Marie-Luise Wohlmuth. AI-and data-driven crop rotation planning. *Computers and Electronics in Agriculture*, 212:108160, 2023.

[FNH+23] Stefan Fenz, Thomas Neubauer, Johannes Heurix, Jürgen Kurt Friedel, and Marie-Luise Wohlmuth. AI-and data-driven pre-crop values and crop rotation matrices. *European Journal of Agronomy*, 150:126949, 2023.

[Fra05] CA Francis. Crop Rotations. Encyclopedia of Soils in the Environment 318–322. doi: 10.1016. Encyclopedia entry, B0-12-348530-4/00253-8, 2005.

[Fre03] Bernhard Freyer. *Fruchtfolgen:[konventionell, integriert, biologisch]; 116 Tabellen*. Ulmer, 2003.

[GCF+22] Marianna B Ganapini, Murray Campbell, Francesco Fabiano, Lior Horesh, Jon Lenchner, Andrea Loreggia, Nicholas Mattei, Taher Rahgooy, Francesca Rossi, Biplav Srivastava, et al. Combining fast and slow thinking for human-like and efficient navigation in constrained environments. *arXiv preprint arXiv:2201.07050*, 2022.

[get23] Getreide und andere Kulturpflanzen, 2023.

[GHL+23] Gerhard Gahleitner, Karin Heinschink, Siegbert Linder, Richard Maria, Thomas Skidmore, Franz Hunger, and Gerald Biedermann. Interaktive Deckungsbeiträge und Kalkulationsdaten. *Bundesanstalt für Agrarwirtschaft und Bergbauernfragen Österreich*, 2023.

[GK05] Matthew Grounds and Daniel Kudenko. Combining reinforcement learning with symbolic planning. In *European Symposium on Adaptive Agents and Multi-Agent Systems*, pages 75–86. Springer, 2005.

[GKK+11] Martin Gebser, Benjamin Kaufmann, Roland Kaminski, Max Ostrowski, Torsten Schaub, and Marius Schneider. Potassco: The Potsdam answer set solving collection. *Ai Communications*, 24(2):107–124, 2011.

[GKNS07] Martin Gebser, Benjamin Kaufmann, André Neumann, and Torsten Schaub. clasp: A conflict-driven answer set solver. In *Logic Programming and Nonmonotonic Reasoning: 9th International Conference, LPNMR 2007, Tempe, AZ, USA, May 15-17, 2007. Proceedings 9*, pages 260–265. Springer, 2007.

[GKO+09] Martin Gebser, Roland Kaminski, Max Ostrowski, Torsten Schaub, and Sven Thiele. On the input language of ASP grounder gringo. In *Logic Programming and Nonmonotonic Reasoning: 10th International Conference, LPNMR 2009, Potsdam, Germany, September 14-18, 2009. Proceedings 10*, pages 502–508. Springer, 2009.

[GL88] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *ICLP/SLP*, volume 88, pages 1070–1080. Cambridge, MA, 1988.

[GMR16] Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. Improving PILCO with Bayesian neural network dynamics models. In *Data-efficient machine learning workshop, ICML*, volume 4, page 25, 2016.

[Goo13] Phillip Good. *Permutation tests: a practical guide to resampling methods for testing hypotheses.* Springer Science & Business Media, 2013.

[GSR21] Aditya Gulati, Sarthak Soni, and Shrisha Rao. Interleaving fast and slow decision making. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1535–1541. IEEE, 2021.

[HB16] John C Hull and Sankarshan Basu. *Options, futures, and other derivatives.* Pearson Education India, 2016.

[HIB$^+$18] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[HPB$^+$19] Gerrit Hoogenboom, Cheryl H Porter, Kenneth J Boote, Vakhtang Shelia, Paul W Wilkens, Upendra Singh, Jeffrey W White, Senthold Asseng, Jon I Lizaso, L Patricia Moreno, et al. The DSSAT crop modeling ecosystem. In *Advances in crop modelling for a sustainable agriculture*, pages 173–216. Burleigh Dodds Science Publishing, 2019.

[HZAL18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[inf23] Informationsblatt ÖPUL 2023: Umweltgerechte und biodiversitätsfördernde Bewirtschaftung. Agrarmarkt Austria, Oktober 2023. *Accessed online (2024-02-18)*: `https://www.ama.at/getattachment/3a7e8e0e-0dbf-4bdc-8ae9-e29c261c05bb/06_1A_Umweltgerechte_und_biodiversitatsfordernde_Bewirtschaftung_(UBB)_2023_10.pdf`.

[JAT$^+$17] Zeynab Jouzi, Hossein Azadi, Fatemeh Taheri, Kiumars Zarafshani, Kindeya Gebrehiwot, Steven Van Passel, and Philippe Lebailly. Organic farming and small-scale farmers: Main opportunities and challenges. *Ecological economics*, 132:144–154, 2017.

[JC19] B Jeangros and N Courvoisier. Optimale Fruchtfolgen im Feldbau. *Agrarforschung Schweiz*, 10:7–8, 2019.

[JFZL19] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

[JHZ+20] Hao Jiang, Hao Hu, Renhai Zhong, Jinfan Xu, Jialu Xu, Jingfeng Huang, Shaowen Wang, Yibin Ying, and Tao Lin. A deep learning approach to conflating heterogeneous geospatial data for corn yield estimation: A case study of the US Corn Belt at the county level. *Global change biology*, 26(3):1754–1766, 2020.

[KCKL08] John Kirkegaard, Olaf Christen, Joseph Krupinsky, and David Layzell. Break crop benefits in temperate wheat production. *Field crops research*, 107(3):185–195, 2008.

[KDO+22] R Knöferl, M Diepolder, K Offenberger, S Raschbacher, M Brandl, A Kavka, L Hippich, R Schmücker, C Sperger, and S Kalmbach. Leitfaden für die Düngung von Acker-und Grünland: Gelbes Heft, hg. v. *Bayerische Landesanstalt für Landwirtschaft (LfL). Freising-Weihenstephan*, 15, 2022.

[KKS12] Roland Kaminski, Benjamin Kaufmann, and T Schaub. Answer set solving in practice. *Morgan & Claypool Publishers*, 2012.

[Kol06] H Kolbe. Fruchtfolgegestaltung im ökologischen und extensiven Landbau: Bewertung von Vorfruchtwirkungen. *Pflanzenbauwissenschaften*, 10(2):82–89, 2006.

[Kol08a] Hartmut Kolbe. Einfache Verfahren zur Berechnung der Humusbilanz für konventionelle und ökologische Anbaubedingungen. 2008.

[Kol08b] Hartmut Kolbe. Fruchtfolgegrundsätze im ökologischen Landbau. 2008.

[Kol12] H Kolbe. Zusammenführende Untersuchungen zur Genauigkeit und Anwendung von Methoden der Humusbilanzierung im konventionellen und ökologischen Landbau. *Bilanzierungsmethoden und Versorgungsniveau für Humus; Schriftenr; Landesamt für Umwelt, Landwirtschaft und Geologie (LfULG): Dresden, Germany*, 19:4–85, 2012.

[KT99] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[KW09] Malvin H Kalos and Paula A Whitlock. *Monte carlo methods*. John Wiley & Sons, 2009.

[LB16] Günter Leithold and Konstantin Becker. Planungshilfe zur Einordnung von Leguminosen in Öko-Fruchtfolgen. 2016. *Justus-Liebig-Universität Gießen.*

[LD99] MA Liebig and JW Doran. Impact of organic production practices on soil quality indicators, 1999. *Wiley Online Library.*

[LfU08] Fruchtfolgegrundsätze im Ökologischen Landbau, 2008. Artikeldetails: 1. Auflage, Redaktionsschluss: 01.06.2008, Publikationsart: Broschüre, *Accessed online (2024-02-18)*: `https://publikationen.sachsen.de/bdb/artikel/13610`.

[Li17] Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

[Lin92] Long-Ji Lin. *Reinforcement learning for robots using neural networks.* Carnegie Mellon University, 1992.

[LIS16] Matteo Leonetti, Luca Iocchi, and Peter Stone. A synthesis of automated planning and reinforcement learning for efficient, robust decision-making. *Artificial Intelligence*, 241:103–130, 2016.

[LTHS88] Yann LeCun, D Touresky, G Hinton, and T Sejnowski. A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, volume 1, pages 21–28, 1988.

[MBG$^+$11] Antonio L Márquez, Raúl Baños, Consolación Gil, María G Montoya, Francisco Manzano-Agugliaro, and Francisco G Montoya. Multi-objective crop planning using pareto-based evolutionary algorithms. *Agricultural Economics*, 42(6):649–656, 2011.

[MBM$^+$16] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[MBP$^+$23] Thomas M Moerland, Joost Broekens, Aske Plaat, Catholijn M Jonker, et al. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1):1–118, 2023.

[MKS$^+$15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[MSL$^+$92] David Mitchell, Bart Selman, Hector Levesque, et al. Hard and easy distributions of SAT problems. In *Aaai*, volume 92, pages 459–465, 1992.

[MVE+00] Fred Magdoff, Harold Van Es, et al. *Building soils for better crops*, volume 4. Sustainable Agriculture Network Beltsville, 2000.

[Nan16] Dilip Nandwani. *Organic farming for sustainable agriculture*, volume 9. Springer, 2016.

[NBM+19] Ritesh Noothigattu, Djallel Bouneffouf, Nicholas Mattei, Rachita Chandra, Piyush Madan, Kush R Varshney, Murray Campbell, Moninder Singh, and Francesca Rossi. Teaching AI agents ethical values using reinforcement learning and policy orchestration. *IBM Journal of Research and Development*, 63(4/5):2–1, 2019.

[NOP94] Amnon Nevo, Ramchand Oad, and Terence H Podmore. An integrated expert system for optimal crop planning. *Agricultural systems*, 45(1):73–92, 1994.

[NW15] Landwirtschaftskammer Nordrhein-Westfalen. Tabellen Fruchtfolge, 2015. *Accessed online (2024-02-18)*: `https://www.landwirtschaftskammer.de/Landwirtschaft/ackerbau/fruchtfolge/index.htm`.

[OID15] Julien Osman, Jordi Inglada, and Jean-François Dejoux. Assessment of a Markov logic model of crop rotations for early crop mapping. *Computers and Electronics in Agriculture*, 113:234–243, 2015.

[óIYIM19] Le ón Illanes, Xi Yan, Rodrigo Toro Icarte, and Sheila A McIlraith. Symbolic planning and model-free reinforcement learning: Training taskable agents. 2019. *University of Toronto*.

[OWC04] Shichao Ou, Xinghua Wang, and Liancheng Chen. The application of reinforcement learning in optimization of planting strategy for large scale crop production. In *2004 ASAE Annual Meeting*, page 1. American Society of Agricultural and Biological Engineers, 2004.

[PBvL09] Ruth Pavón, Ricardo Brunelli, and Christian von Lücken. Determining optimal crop rotations by using multiobjective evolutionary algorithms. In *Knowledge-Based and Intelligent Information and Engineering Systems: 13th International Conference, KES 2009, Santiago, Chile, September 28-30, 2009, Proceedings, Part I 13*, pages 147–154. Springer, 2009.

[PKB21] Christoph Pahmeyer, Till Kuhn, and Wolfgang Britz. 'Fruchtfolge': A crop rotation decision support system for optimizing cropping choices with big data and spatially explicit modeling. *Computers and Electronics in Agriculture*, 181:105948, 2021.

[PKP20] Aske Plaat, Walter Kosters, and Mike Preuss. Deep model-based reinforcement learning for high-dimensional problems, a survey. *arXiv preprint arXiv:2008.05598*, 2020.

[Rec22] Thomas Rech. Aktionsprogramm Biologische Landwirtschaft 23+. Österreichisches Bundesministerium für Land- und Forstwirtschaft, Regionen und Wasserwirtschaft, 2022. *Accessed online (2024-02-18)*: `https://info.bml.gv.at/themen/landwirtschaft/bio-lw/bio-aktionsprogramm.html`.

[RHB+16] Moritz Reckling, Jens-Martin Hecker, Göran Bergkvist, Christine A Watson, Peter Zander, Nicole Schläfke, Frederick L Stoddard, Vera Eory, Cairistiona FE Topp, Juliette Maire, et al. A cropping system assessment framework—evaluating effects of introducing legumes into crop rotations. *European Journal of Agronomy*, 76:186–197, 2016.

[RKE09] Joel K Ransom, Hans Kandel, and Gregory Endres. Replanting or Late Planting Crops. 2009. *North Dakota State University*.

[SAC+20] Raí A Schwalbert, Telmo Amado, Geomar Corassa, Luan Pierre Pott, PV Vara Prasad, and Ignacio A Ciampitti. Satellite-based soybean yield forecast: Integrating machine learning and weather data for improving crop yield prediction in southern Brazil. *Agricultural and Forest Meteorology*, 284:107886, 2020.

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[SBRG13] Karin Stein-Bachinger, Moritz Reckling, and Artur Granstedt. *Pflanzenbau & Tierhaltung*, volume I of *Kreislauforientierte oekologische Landwirtschaft - Handlungsempfehlungen für Landwirte und Berater*. ZALF, 2013.

[SDH+18] Ayush Shah, Akash Dubey, Vishesh Hemnani, Divye Gala, and DR Kalbande. Smart farming system: Crop yield prediction using regression techniques. In *Proceedings of International Conference on Wireless Communication: ICWiCom 2017*, pages 49–56. Springer, 2018.

[SHM+16] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[SHS+17] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[SLA$^+$15] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[SPS99] Richard S Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[SQAS15] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[SSS11] Martin Schönhart, Erwin Schmid, and Uwe A Schneider. CropRota–A crop rotation model to support integrated land use assessments. *European Journal of Agronomy*, 34(4):263–277, 2011.

[Ste24] Landwirtschaftskammer Steiermark. Anbau- und Kulturanleitungen, 2024. *Accessed online (2024-02-18)*: `https://stmk.lko.at/anbau-und-kulturanleitungen+2400+2575912`.

[Sut88] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44, 1988.

[Sut90] Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pages 216–224. Elsevier, 1990.

[Sut91] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

[SWD$^+$17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[SWT23] Julius Schöning, Paul Wachter, and Dieter Trautz. Crop rotation and management tools for every farmer?: The current status on crop rotation and management tools for enabling sustainable agriculture worldwide. *Smart Agricultural Technology*, 3:100086, 2023.

[Tes94] Gerald Tesauro. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation*, 6(2):215–219, 1994.

[TG96] Gerald Tesauro and Gregory Galperin. On-line policy improvement using Monte-Carlo search. *Advances in neural information processing systems*, 9, 1996.

[TOdSMJZ20] Danilo Tedesco-Oliveira, Rouverson Pereira da Silva, Walter Maldonado Jr, and Cristiano Zerbato. Convolutional neural networks in predicting cotton yield from images of commercial fields. *Computers and Electronics in Agriculture*, 171:105307, 2020.

[uR18] Europäisches Parlament und Rat. VERORDNUNG (EU) 2018/848 DES EUROPÄISCHEN PARLAMENTS UND DES RATES über die ökologische/biologische Produktion und die Kennzeichnung von ökologischen/biologischen Erzeugnissen sowie zur Aufhebung der Verordnung (EG) Nr. 834/2007 des Rates, 2018. *Accessed online (2024-02-18)*: `https://eur-lex.europa.eu/legal-content/DE/TXT/?uri=CELEX%3A32018R0848`.

[VHGS16] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[VKKC20] Thomas Van Klompenburg, Ayalew Kassahun, and Cagatay Catal. Crop yield prediction using machine learning: A systematic literature review. *Computers and Electronics in Agriculture*, 177:105709, 2020.

[vLAR21] Christian von Lücken, Angel Acosta, and Norma Rojas. Solving a many-objective crop rotation problem with evolutionary algorithms. In *Intelligent Decision Technologies: Proceedings of the 13th KES-IDT 2021 Conference*, pages 59–69. Springer, 2021.

[Wag23] Magnus Wagner. cro symbolic mbpo. Github repository, 2023. *Accessed online (2024-02-18)*: `https://github.com/MagnusWagner/cro_symbolic_mbpo`.

[Wat89] Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989. *King's College, Cambridge United Kingdom*.

[WHF05] Jack Wang, Aaron Hertzmann, and David J Fleet. Gaussian process dynamical models. *Advances in neural information processing systems*, 18, 2005.

[WHFY20] Xinlei Wang, Jianxi Huang, Quanlong Feng, and Dongqin Yin. Winter wheat yield prediction at county level and uncertainty analysis in main wheat-producing regions of China with deep learning approaches. *Remote Sensing*, 12(11):1744, 2020.

[Wie24] Martin Wiesmeier. Ertragswirkung unterschiedlicher Humusgehalte, 2024. *Accessed online (2024-02-18)*: `https://www.lfl.bayern.de/iab/boden/031146/index.php`.

[Wil92] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

[Wit77] Ian H Witten. An adaptive optimal controller for discrete-time Markov environments. *Information and control*, 34(4):286–295, 1977.

[WTD+18] Anna X Wang, Caelin Tran, Nikhil Desai, David Lobell, and Stefano Ermon. Deep transfer learning for crop yield prediction with remote sensing data. In *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies*, pages 1–5, 2018.

[YBP20] Raghu Yaramasu, Varaprasad Bandaru, and Koutilya Pnvr. Pre-season crop type mapping using deep neural networks. *Computers and Electronics in Agriculture*, 176:105664, 2020.

[YSH+19] Qi Yang, Liangsheng Shi, Jinye Han, Yuanyuan Zha, and Penghui Zhu. Deep convolutional neural networks for rice grain yield estimation at the ripening stage using UAV-based remotely sensed images. *Field Crops Research*, 235:142–153, 2019.

[ZDLG19] Chen Zhang, Liping Di, Li Lin, and Liying Guo. Machine-learned prediction of annual crop planting in the US Corn Belt based on historical crop planting maps. *Computers and Electronics in Agriculture*, 166:104989, 2019.

[ZMBM20] Zhaoyu Zhai, José Fernán Martínez, Victoria Beltran, and Néstor Lucas Martínez. Decision support systems for agriculture 4.0: Survey and challenges. *Computers and Electronics in Agriculture*, 170:105256, 2020.