Optimal Control of Lettuce Greenhouse Horticulture using Model-Free Reinforcement Learning

An investigation on the effect of short-term weather forecast horizons

Y.T.G. Jansen

A thesis presented for the degree of Master of Science

in Artificial Intelligence

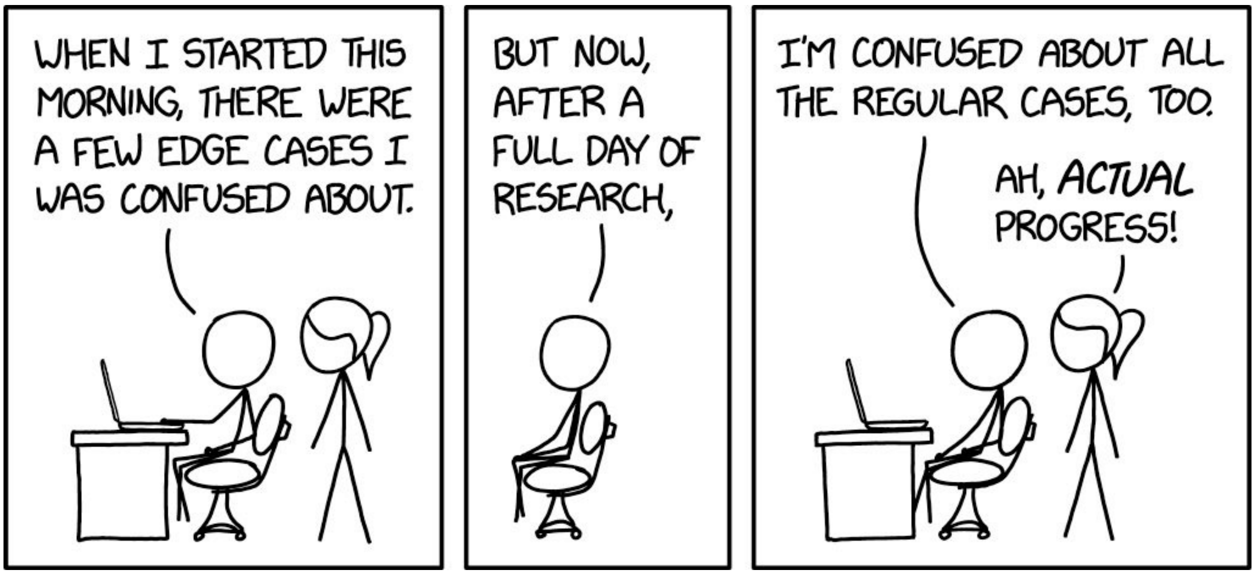
Supervisor Dr. N. Alechina
Intelligent Systems group
AI & Data Science division
Department of Information and Computing Sciences
Faculty of Science
Utrecht University, Netherlands

in collaboration with

Supervisor Dr. S. Boersma
Biometris group
Wageningen University & Research, Netherlands

Acknowledgements

First of all I would like to thank my supervisor Natasha Alechina as well as my daily supervisor Sjoerd Boersma for their guidance and support during the thesis process. The figure below describes my thesis experience very well and I particularly appreciated my supervisors' flexibility and patience with regards to this. They helped me to better put things in perspective and to improve my decision making on what is relevant for the main aim of the story, and what is less so; details that are very interesting - and ever so slightly relevant - but are less important, or even distracting, when looking at the bigger picture. (I am aware of the irony here.)


WHEN I STARTED THIS MORNING, THERE WERE A FEW EDGE CASES I WAS CONFUSED ABOUT.
BUT NOW, AFTER A FULL DAY OF RESEARCH,
I'M CONFUSED ABOUT ALL THE REGULAR CASES, TOO.
AH, *ACTUAL* PROGRESS!

I am also very grateful for the support by dr. ir. Congcong Sun and dr. Gert Kootstra of the Agricultural Biosystems Engineering group at Wageningen University & Research, they also helped me develop a more pragmatic approach in my work. I also want to thank dr. Shihan Wang from the Intelligent Systems group at Utrecht University for her helpful and constructive feedback as second examiner. Finally, I would like to thank all my friends for their support - I could not have done it without them. I want to specifically mention to Raoul Buurke, Walter van Rijen and Wiebe de Vries. The final mention goes to Niels Wouda, who provided me with valuable feedback on multiple occasions.

Abstract

A greenhouse is an important growing system that can provide a controlled climate environment and allow for crop growth in a changing outdoor climate. Due to the high energy cost, labor and resource scarcity, optimal and automated control of greenhouse horticulture is becoming more and more important with the aim of optimizing resource usage while maximizing crop production. Outdoor weather is a critical disturbance when controlling greenhouse climate and it complicates the modelling and optimization processes.

With the development of Artificial Intelligence (AI) and improved sensing techniques, Reinforcement Learning (RL) is getting more and more attention due to its learning-based control strategies, independent from having a good model. Up to now, most of the RL applications in greenhouse climate control do not consider outdoor weather forecast while making control decisions, which means plenty of useful information is missed and this might lead to control actions which are not optimal. Therefore in this project, we investigated how weather forecast horizons will affect optimal control of greenhouse horticulture using reinforcement learning.

After going through different deep RL approaches, Soft Actor-Critic (SAC) and Twin-Delayed Deep Deterministic Policy Gradient (TD3) stand out because of their capacity to consider continuous state-action space. As the weather prediction will mainly work well in the short-term due to forecast uncertainty, moreover, long-term weather forecast will bring various unnecessary noise and a large state space. As a result, our work mainly focused on short-term weather forecast horizons of 0, 3, 7, 11, 15, 19, 23 time steps of fifteen minutes. To investigate how horizons can affect the control performance, these seven different horizons were used in experiments using the state-of-the-art continuous control algorithms, SAC and TD3.

After demonstrating the proposed approaches in a lettuce greenhouse, we found that SAC consistently performed better than TD3 with higher rewards in terms of crop production and net profit, while resource use was comparable. Furthermore, inclusion of weather forecasts proved essential for both algorithm learning stability, as well as its training and generalization performance, resulting in increased yields and net profits, while reducing resource use and the amount of indoor climate constraint violations. Moreover, we can also conclude that four hours of weather forecast is the best option. Longer predictions did not increase performance, whereas using shorter forecasts quickly degraded performance.

Contents

Figure 1
Large-scale lettuce crop production in greenhouse.[1]

1 Introduction

Between 720 and 811 million people are estimated to be undernourished in 2020 and this number has slowly been increasing again since 2014 after years of decreasing (FAO et al. (2021)). With current global population levels estimated at 7.7 billion people and projected to grow to 9.7 billion in 2050, demands for fresh and nutritional foods are rapidly increasing (United Nations et al. (2019)). Additionally, resources such as fresh water (Marcelis et al. (2019)), oil and gas reserves, and arable land are becoming increasingly scarce while the effect of global warming becomes more visible. Consequently, it becomes more and more important to produce high-quality fresh food in a resource-efficient manner. Greenhouses can offer a helping hand here due to having high crop production rates per covered surface area and being water-efficient (Stanghellini (2014)). At the same time, the global greenhouse surface area is increasing rapidly (van Rijswick (2018)). The downside is that they have a relatively high energy consumption and require a relatively high initial investment (Graamans et al. (2018)).

Greenhouses serve to protect crops from external disturbances, such as adverse weather effects, i.e., snow, high wind speeds, and extreme relative humidity and temperature situations. It also protects crops against diseases, pests and insects, while allowing to make use of radiation from the sun for heating and photosynthesis. A greenhouse provides an environment where climate conditions can be controlled optimally for optimal crop growth. This allows for both higher quality and larger quantities of crop production compared to open field crop production (van Straten and van Henten (2010)), as well as growing crops in a cost- and sustainable, resource-efficient manner. Another benefit is that it allows crop production during the entire year and in locations where it may have been impossible without a greenhouse (Bakker et al. (1995)). A visual representation of greenhouse horticulture with lettuce can be seen in Fig. 1.

Indoor climate variables that can be measured and controlled using the sensors and actuators include temperature, humidity and carbon dioxide levels. Additionally, the weather disturbances (outside weather) can be exploited. Especially the energy of the sun is useful. An example is using the incoming radiation and outside temperature to heat up the greenhouse instead of turning on the heater or the use of cooler outside air for lowering the indoor temperatures.

During the day, photosynthetically active radiation (PAR) from incoming solar radiation enables the photosynthesis process in crops where CO_2 and H_2O are used and converted into glucose. This is subsequently used by the plant to grow (additionally oxygen O_2 is produced and H_2O is released due to transpiration, increasing the humidity concentration in the greenhouse) (Farquhar et al. (1980); Marcelis et al. (2006)). How much photosynthesis takes places depends on the amount of incoming radiation, the indoor temperature and the CO_2 concentration. During the night there is no sunlight, so no crop growth

[1] From *https://www.stockfood.com/images/00228603-Lettuce-growing-in-greenhouse*

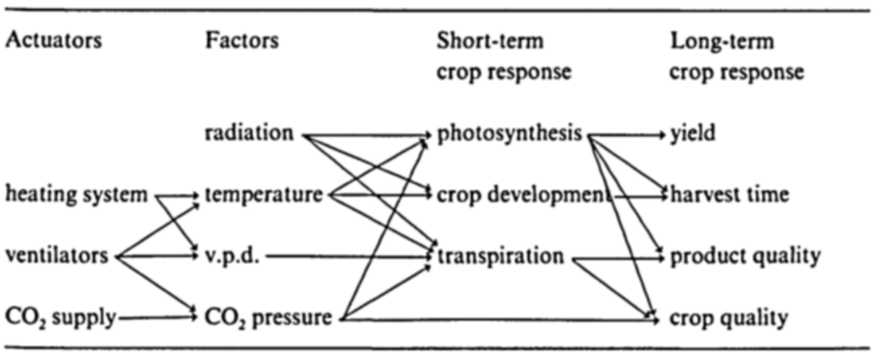


Figure 2
Causal relations in greenhouse system, from Challa (1990). V.p.d. is the abbreviation for vapour pressure deficit.

takes place and additional heating is necessary to maintain acceptable indoor temperatures (which are lower during the night than during the day as no photosynthesis will take place regardless).

The greenhouse for crop production is a complex system that can approximately be divided in two subsystems: the greenhouse indoor climate and the crop system (van Henten and Bontsema (2009)). The dynamics of the former are affected by the outside weather, control inputs by the greenhouse actuators and by the crops present in the greenhouse via, for example, (evapo)transpiration and CO_2 uptake. The dynamics of the latter are affected by the current state of the crops, the greenhouse climate as well as the absence or presence of light Fig. 2.

Greenhouse crop process models have been developed to understand, describe and estimate effects of the greenhouse climate on crop growth using, for example, Ordinary Differential Equations (ODEs). Exact modelling approaches depend on the use-case. In this specific work, a lettuce greenhouse model developed by (van Henten (1994b, 2003)) is used. Use of a crop model, together with developed models for describing the greenhouse climate allows for automated decision making on greenhouse climate set-points and crop growth processes using control methods. Implementation of automated decision making algorithms in the control loop requires linking the greenhouse sensors and actuators to suitable greenhouse crop and climate models. An overview of dynamic greenhouse climate models can be found in (Lopez-Cruz et al. (2018); Katzin et al. (2022))

Growing crops in a greenhouse requires active climate control, preferably in an automated or semi-automated fashion. This climate control system can, for example, increase or decrease the temperature of the greenhouse based on changing outside temperatures via heating, increase carbon dioxide levels (CO_2) in the greenhouse, or decrease the indoor relative humidity by heating or ventilation. The goal of these actions is to optimise long-term crop growth, while maintaining favorable climate conditions for the crop and minimising resource costs.

In practice, settings for climate control systems are largely based on simple, rule-based heuristics, weather forecasts and crop growers' past experiences (van Beveren et al. (2015a,b)). Growers decide on set-points[2] for the various greenhouse climate variables. It is not always possible to maintain optimal conditions for each variable as the system is complex and highly coupled and may lead to conflicting behaviours from the various actuators and unachievable set-points. As such, a band around the set-points is usually included to allow for deviations and ease trade-offs between control inputs. The instructions are passed to a climate computer where simple low-level feedback controllers - proportional- integral-derivative (PID) (and variants)- are used to control the actuators for set-points regulation (van Straten and van Henten (2010)). These do not require a model of the environment, are easy to implement and robust, but can only respond to disturbances post-hoc. An extension of feedback control is feed forward control that has access to anticipated disturbances, but this already requires a model to estimate effects of control inputs and future disturbances on the current state (van Mourik et al. (2021); van Straten and van Henten (2010)).

The control decisions that have to be made are highly complex and grower decisions regularly leave room for improvements as human behaviour is often sub-optimal (Kahneman (2003)). This complexity also means that vast amounts of experience is required and it is becoming increasingly difficult to find enough skilled laborers for crop management (Sparks (2018)). This is where automation can provide relief by providing decision support for or partially replacing the crop growers, or both. The recent Autonomous Greenhouse Challenges indicate a promising future for automated greenhouse management, with intelligent systems outperforming human crop growers in crop production quantity, quality

[2] A set-point is the desired value for a given variable.

and resource-efficiency (Hemming et al. (2019, 2020)). The long term aim is to produce robust and explainable control systems that allow for resource efficient and fully autonomous greenhouses that handle both climate regulation and crop growth control strategies (Verdouw et al. (2021); Knibbe et al. (2022)). Then, only a single remote crop grower is required to supervise multiple greenhouses at once.

Currently proposed automated control strategies in scientific literature mainly focus on controlling the climate in the greenhouse via model predictive control (MPC) in simulation studies. It is a model-based method that has full access to - and makes use of - greenhouse climate and crop models, and assumes knowledge on the weather forecasts. See, for example, (van Henten and Bontsema (2009); Ooteghem (2010); van Straten and van Henten (2010); van Beveren et al. (2015a,b); Chen et al. (2018); Xu and van Willegenburg (2018); Kuijpers et al. (2021, 2022); Boersma et al. (2022a)). These proposed solutions optimizes a given objective function that is defined over a prediction horizon and is often used to optimize on set-points, but can also be used for set-point regulation (van Henten and Bontsema (2009); van Straten and van Henten (2010); van Mourik et al. (2021)). It repeatedly receives state-feedback from the environment and then plans the control inputs for a given horizon using the model of the environment and the anticipated future disturbances to estimate the development of the indoor dynamics (van Mourik et al. (2021); van Straten and van Henten (2010)). While these studies are promising, MPC does have several downsides: First of all, it uses feedback from the simulator to estimate future developments of the state-variables for planning, which means that it is sensitive to small one-step prediction errors that originate from model biases or the reality gap,.This can lead to compounding prediction errors and a significantly degraded prediction accuracy. Secondly, as it plans a series of actions for a given prediction horizon, MPC is sensitive to getting stuck in local optima when the included model is nonlinear. Additionally, finding optimal trajectories becomes difficult to impossible as the solution space increases exponentially with increasing model complexity (Knibbe et al. (2022)). Finally, a mathematical model of the most important processes in the greenhouse and the crop is required. Such a model is typically difficult to identify correctly, requiring historical data that may not be available, and is often too complex to actually evaluate. Although MPC provides great insight in greenhouse control, it is unfortunately not often used in practice (van Beveren et al. (2015b)), and Van Straten and van Henten hypothesized that self-learning and self-optimizing systems would lead to faster implementation of optimal control algorithms in practice (van Straten and van Henten (2010)). Thus, using an alternative method that does not assume access to an underlying mathematical model may be appropriate.

Reinforcement Learning (RL) provides methods to develop learning and optimizing decision-making algorithms and has seen success in other fields; playing Atari games (Mnih et al. (2013, 2015)), playing Go and beating the human world champion (Silver et al. (2016)). Other examples are playing Chess Hassabis (2017)), autonomous driving (Waldrop (2015)), navigating stratospheric balloons (Bellemare et al. (2020)) and robotics (Goldberg (2019); Tan et al. (2018); Haarnoja et al. (2018a)). It has also been used in some agricultural applications, such as nutrient application (Overweg et al. (2021); Gautron et al. (2022); Turchetta et al. (2022)) and irrigation control Saikai et al. (2023)) in open field horticulture, as well as in robotics. However, not much work exists on the application of RL to greenhouse control as of yet.

Two Autonomous Greenhouse Challenges (AGC) were held by WUR, in 2018 and 2020 respectively. Their aim was to grow crops using autonomously controlled greenhouses with the objective of maximizing net profit. Five teams participating in each challenge, as well as a group of human crop growers as reference. The first challenge focused on growing cucumbers and the second challenge focused on growth of cherry tomatoes. A variety of model-free and model-based AI control algorithms were used by the teams to decide on optimal climate set-points, including RL algorithms. Realisation of the set-points was achieved using low-level PI- and P-controllers. Model-based Bayesian RL and model-free Deep Deterministic Policy Gradient (DDPG) were used by two of the teams during the first challenge, and an unspecified and unexplained algorithm was used during the second challenge. In both challenges teams pre-trained their algorithms using crop and greenhouse simulators and these were subsequently applied to a real greenhouse crop growing case (Hemming et al. (2019, 2020)). Little detail is provided on exact approaches and methods used as these contain sensitive information of businesses involved, but datasets of the weather as well as the realized greenhouse climate and crop data of the teams involved are freely available.

Advances in using Reinforcement Learning for optimal control in greenhouses in scientific literature are made in Wang et al. (2020); Zhang et al. (2021); An et al. (2021); Cao et al. (2022); Ajagekar and You (2022), each with an economic objective function with the exception of the last. In Wang et al. (2020) the static datasets from the first Autonomous Greenhouse Challenge were used to train a control system that decides on hourly CO_2 temperature and relative humidity set-points using Behaviour Cloning and the Deep-Deterministic Policy Gradient (DDPG) algorithm, performance evaluation of the final control system was done on the same weather dataset. The work by Zhang et al. (2021) uses model-based RL and decided on hourly actions for the entire next day for indoor temperature and CO_2, as well as starting and end times for both irrigation and lighting. In An et al. (2021), synthetic data from a cherry tomato simulator is used to first learn surrogate models for the greenhouse climate and crop dynamics. Its performance to generalize is verified using a static dataset from the second Autonomous Greenhouse Challenge. Subsequently these surrogate models are used to train control systems that decides on hourly set-points for CO_2 temperature and relative humidity, as well as whether additional lighting is used or not using SAC and as an Elitist Genetic Algorithm (EGA). The work of Cao et al. (2022) builds on the previously mentioned work. It adds functionality to perform online calibrations of the surrogate models using newly obtained real data and re-optimization of the control system using the latest model versions.[3] Ajagekar and You (2022) focuses on temperature control in a greenhouse with tomato crops, although crop dynamics are ignored. Heating or cooling actions are performed on an hourly basis to maintain temperature within bounds. A variant of Deep Q-Networks is used and DDPG functions as baseline. Optimization is done by balancing energy use and maintenance of desired condition.

None of these works include weather forecasts in their approach, which is an important factor to take into account when optimizing control decisions. As such, this work investigates the effect of short-term weather forecast lengths on performance of low-level control systems using two state-of-the-art model-free RL control algorithms in Soft Actor-Critic (SAC) (Haarnoja et al. (2018c)) and Twin-Delayed Deterministic Policy Gradient (TD3) (Fujimoto et al. (2018)). As the focus is on the control of the greenhouse climate and crop, a very simple crop in lettuce is used.

The following research questions are answered in this work:

- How do various weather forecast horizons affect controller performance?
- How do SAC and TD3 results compare?

The outline of this work is as follows, the used greenhouse and crop model are explained in Chapter 2. Chapter 3 focuses on Reinforcement Learning, providing preliminaries, descriptions of the two algorithms and finishes by describing the environment and reward structure. The experiment setup and simulation results are presented in Chapter 4. Chapter 5 consists of the discussion, summarized key findings, concluding remarks and finishes with recommendation for future work.

[3] These last two works claim to have extended the datasets with synthetic datasets. However, their reference is missing in the first paper and incorrect in the second. Regardless, the real-world test claimed to have been performed in the second paper is not feasible. Algorithm selection for the real-life experiments was based on data from the second Autonomous Greenhouse Challenge, but this data was not yet available at the time as the challenge was still ongoing.

2 Lettuce Greenhouse

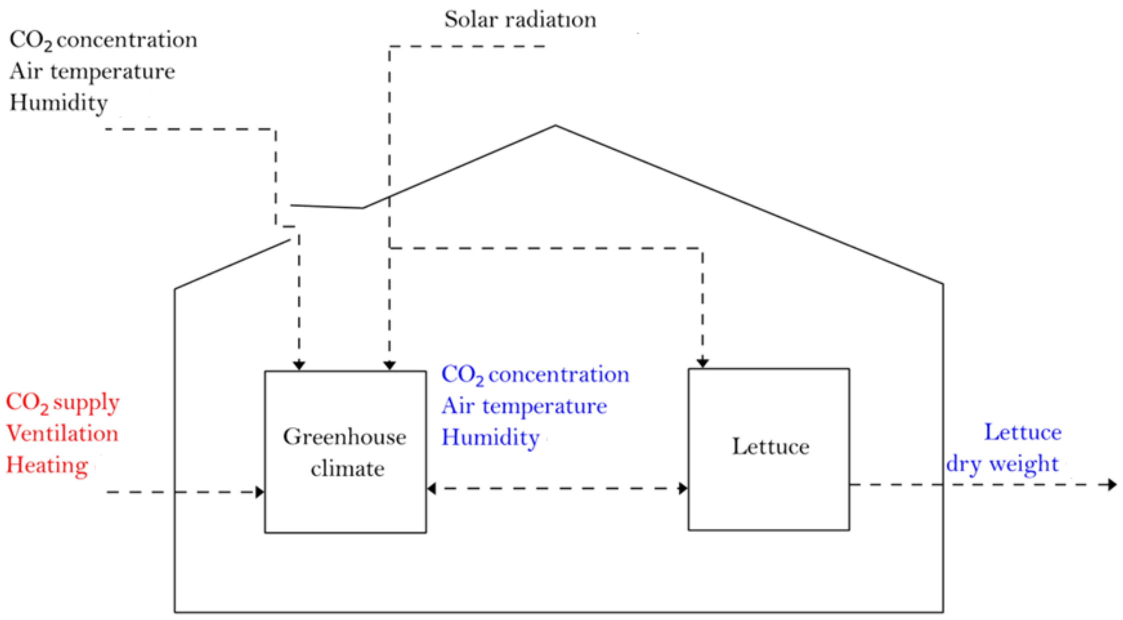


Figure 3
Schematic overview of the environment of van Henten (1994a, 2003), adapted by Sjoerd Boersma. Control inputs are given in red, system measurements in blue and the external disturbances are in black. Information flows and directions are indicated by the dashed arrows.

2.1 Greenhouse lettuce model description

A validated model for climate dynamics and lettuce growth in greenhouses by van Henten (1994b, 2003) is used in this work. (Additional background on greenhouse modelling can be found in subsection 6.1.) A visual overview of the model is given in Fig. 3. Control inputs are given in red, system measurements in blue and the external disturbances are in black. The dashed lines indicate interactions and their directions. The model is discretized using the explicit fourth order Runge-Kutta method (Press et al. (1986)), the general form of the resulting discrete-time model is as follows:

$$\begin{aligned} x(k+1) &= f(x(k), u(k), d(k); c), \\ y(k) &= g(x(k); c), \end{aligned} \tag{1}$$

where $k \in \mathbb{Z}_{\geq 0}$ is the the discrete time, $x(k) \in \mathbb{R}^4$ represents the state variables and its dynamics over time are represented by $f(\cdot)$, $u(k) \in$ represents the used control signals at a given time step, $d(k) \in \mathbb{R}^4$ represents the weather disturbances at each time step k and $c \in \mathbb{R}^{29}$ consists of the greenhouse climate and lettuce crop model constants. $y(k) \in \mathbb{R}^4$ consists of the sensor measurements, which are obtained from the state variables via $g(\cdot)$. $f(\cdot)$ and $g(\cdot)$ both represent non-linear functions that are explained in 2.2. An overview of the variable symbols introduced in this chapter and their descriptions is given in Table 1. The vectors are defined as:

$$\begin{aligned} x(k) &= \begin{pmatrix} x_W & x_{CO_2} & x_T & x_h \end{pmatrix}^T, \\ u(k) &= \begin{pmatrix} u_{CO_2} & u_v & u_q \end{pmatrix}^T, \\ d(k) &= \begin{pmatrix} d_{I_o} & d_{CO_2} & d_T & d_h \end{pmatrix}^T, \\ y(k) &= \begin{pmatrix} y_W & y_{CO_2} & y_T & y_{RH} \end{pmatrix}^T, \end{aligned} \tag{2}$$

where the time dependencies (current time step k) of the vector components have been left out for the sake of readability (in other words, $x_W = x_W(k)$, $d_{I_o} = d_{I_o}(k)$ and so on). x_W describes the crop dry matter content in $kg \cdot m^{-2}$, i.e. the crop weight where all water content has been removed from the crops. Dry weight is used as it is considered a more accurate measurement of biomass. Crop water contents fluctuates with environmental conditions (crop moisture contents and consequently fresh weight will be lower if hot and dry conditions have been experienced recently, for example). The greenhouse CO_2 density x_{CO_2} in kg $\cdot$ m^{-3}, greenhouse air temperature x_T in °C and the absolute humidity content of the air in the greenhouse x_h in kg $\cdot$ m^{-3} are also included in $x(k)$. The control signals $u(k)$ consist of CO_2 injection rate u_{CO_2} in mg $\cdot$ m^{-2} $\cdot$ s^{-1}, ventilation rate u_v in mm $\cdot$ s^{-1} and heating supply u_q in W $\cdot$ m^{-2}. The four weather disturbances included in $d(k)$ are the incoming radiation d_{I_o} in W $\cdot$ m^{-2}, outside carbon dioxide density d_{CO_2} in kg $\cdot$ m^{-3}, outside temperature d_T in °C and outside humidity content d_h in kg$\cdot$ m^{-3}. The measured state observations $y(k)$ consist of lettuce dry weight y_W in kg $\cdot$ m^{-2}, indoor CO_2 concentration y_{CO_2} in $10^3\cdot$ ppm, observed temperature y_T in in °C, and finally the relative humidity y_{RH} in %. This contains very similar information to $x(k)$, but reports the indoor carbon dioxide concentration and humidity in units used by standard measurement sensors.[4]

Table 1

Main symbols.

Category	Symbol	Description
State variables	x_W	crop dry matter content in kg $\cdot$ m^{-2}
	x_{CO_2}	Greenhouse CO_2 density in kg $\cdot$ m^{-3}
	x_T	Greenhouse air temperature in °C
	x_h	Greenhouse air absolute humidity content in kg $\cdot$ m^{-3}
Control inputs	u_{CO_2}	CO_2 injection rate u_{CO_2} in mg $\cdot$ m^{-2} $\cdot$ s^{-1}
	u_v	Ventilation rate u_v in mm $\cdot$ s^{-1}
	u_q	Heating supply u_q in W $\cdot$ m^{-2}
Disturbances	d_{I_o}	Incoming radiation d_{I_o} in W $\cdot$ m^{-2}
	d_{CO_2}	Outside carbon dioxide density d_{CO_2} in kg $\cdot$ m^{-3}
	d_T	Outside temperature d_T in °C
	d_h	Outside absolute humidity content d_h in kg$\cdot$ m^{-3}
Measurement variables	y_W	Lettuce dry weight in kg $\cdot$ m^{-2}
	y_{CO_2}	Greenhouse CO_2 concentration in ppm
	y_T	Greenhouse air temperature in °C
	y_{RH}	Greenhouse air relative humidity in %
Processes	$\phi_{\text{phot, c}}$	The gross canopy photosynthesis rate in kg $\cdot$ m^{-2} $\cdot$ s^{-1}
	$\phi_{\text{resp, W}}$	Respired dry matter due to the maintenance respiration of the crop in kg $\cdot$ m^{-2} $\cdot$ s^{-1}
	$\phi_{\text{resp, c}}$	Amount of CO_2 released due to crop respiration in kg $\cdot$ m^{-2} $\cdot$ s^{-1}
	$\phi_{\text{vent, c}}$	Mass exchange of CO_2 through the vents and due to leakage in kg $\cdot$ m^{-2} $\cdot$ s^{-1}
	$Q_{\text{vent, q}}$	Energy exchange through the vents and by transmission through the greenhouse cover in W $\cdot$ m^{-2}
	$Q_{I_o\text{, q}}$	Heating due to incoming solar radiation in W $\cdot$ m^{-2}
	$\phi_{\text{transp, h}}$	Canopy transpiration in kg $\cdot$ m^{-2} $\cdot$ s^{-1}
	$\phi_{\text{vent, h}}$	Mass exchange of water vapour through the vents and via leakage in kg $\cdot$ m^{-2} $\cdot$ s^{-1}

2.2 Model equations

Descriptions of the symbols describing processes in the model can be found in Table 1, the model constants and their descriptions, values and units can be found in Table 2. The state dynamics in continuous form are described by the following set of deterministic ordinary differential equations:

$$\frac{x_W}{dt} = c_{\alpha\beta}\phi_{phot,c}(t) - \phi_{\text{resp, W}}(t), \tag{3}$$

where $t \in \mathbb{R}$ is the continuous time, $c_{\alpha\beta}$ is a yield factor for the crop, $\phi_{phot,c}(t)$ is the gross canopy photosynthesis rate $\phi_{\text{phot,c}}(t)$ in kg $\cdot$ m^{-2} $\cdot$ s^{-1} and $\phi_{\text{resp, W}}(t)$ is the respired dry matter due to the maintenance

[4] While sensors exist that can measure either absolute or relative humidity, the added value of measuring both is very limited. In other works one is used and is used to estimate the other (van Beveren et al. (2015a,b); van Henten and Bontsema (2009)). Constraints will be imposed on relative humidity and as such including this information has added value (see subsection 3.4).

respiration[5] of the crop in $\mathrm{kg} \cdot \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}$,

$$\frac{\mathrm{d}x_{\mathrm{CO_2}}(t)}{\mathrm{d}t} = \frac{1}{c_{\text{cap, c}}}\Big(- \phi_{\text{phot,c}}(t) + \phi_{\text{resp, c}}(t) + u_{\mathrm{CO_2}}(t) \cdot 10^{-6} - \phi_{\text{vent,c}}(t)\Big), \tag{4}$$

where $c_{\text{cap, c}}$ is the volumetric $\mathrm{CO_2}$ capacity of the greenhouse air in $\mathrm{m}^3 \cdot$ [air] m^{-2} [greenhouse (gh)] [6], $\phi_{\text{resp, c}}(t)$ is the amount of $\mathrm{CO_2}$ released by the crop by respiration in $\mathrm{kg} \cdot \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}$, and $\phi_{\text{vent,c}}(t)$ is the mass exchange of $\mathrm{CO_2}$ through the vents and due to leakage in $\mathrm{kg} \cdot \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}$,

$$\frac{\mathrm{d}x_{\mathrm{T}}(t)}{\mathrm{d}t} = \frac{1}{c_{\text{cap, q}}}\Big(u_{\mathrm{q}}(t) - Q_{\text{vent, q}}(t) + Q_{\mathrm{I_o,q}}(t)\Big), \tag{5}$$

where $c_{\text{cap, q}}$ is the effective heat capacity of the greenhouse air in J m^{-2} [gh] °C^{-1}, $Q_{\text{vent, q}}(t)$ is the energy exchange through the vents and by transmission through the greenhouse cover in $\mathrm{W} \cdot \mathrm{m}^{-2}$ and $Q_{\mathrm{I_o,q}}(t)$ is heating due to incoming solar radiation in $\mathrm{W} \cdot \mathrm{m}^{-2}$,

$$\frac{\mathrm{d}x_{\mathrm{h}}(t)}{\mathrm{d}t} = \frac{1}{c_{\text{cap, h}}}\big(\phi_{\text{transp,h}}(t) - \phi_{\text{vent,h}}(t)\big), \tag{6}$$

where $c_{\text{cap, h}}$ is the volumetric humidity capacity of the greenhouse air in $\mathrm{m}^3 \cdot$ [air] m^{-2} [greenhouse (gh)][6], $\phi_{\text{transp,h}}(t)$ is the canopy transpiration.[7] in $\mathrm{kg} \cdot \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}$ and $\phi_{\text{vent,h}}(t)$ is mass exchange of water vapour through the vents and via leakage in $\mathrm{kg} \cdot \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}$.

2.2.1 Processes

The gross canopy photosynthesis rate $\phi_{\text{phot,c}}(t)$ in $\mathrm{kg} \cdot \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}$ is described as:

$$\phi_{\text{phot,c}}(t) = \big(1 - e^{-c_{\text{LAI, W}} x_{\mathrm{W}}(t)}\big) \frac{c_{\mathrm{I_o}}^{\text{phot}} d_{\mathrm{I_o}}(t) \cdot \varphi(t)}{c_{\mathrm{I_o}}^{\text{phot}} d_{\mathrm{I_o}}(t) + \varphi(t)}, \tag{7}$$

where $c_{\text{LAI, W}}$ is the effective canopy surface in $\mathrm{m}^2 \cdot \mathrm{kg}^{-1}$. The first term (between brackets) is the effective surface of the crop per square meter and the second term represents the maximum possible photosynthesis, where $c_{\mathrm{I_o}}^{\text{phot}}$ is the light use efficiency in $\mathrm{kg} \cdot [\mathrm{CO_2}]\ \mathrm{J}^{-1}$ and $\varphi(t)$ is extracted for sake of readability and is unitless:

$$\varphi(t) = \big(- c_{\mathrm{CO_2},1}^{\text{phot}} x_{\mathrm{T}}(t)^2 + c_{\mathrm{CO_2},2}^{\text{phot}} x_{\mathrm{T}}(t) - \ c_{\mathrm{CO_2},3}^{\text{phot}}\big)\big(x_{\mathrm{CO_2}}(t) - c^{\text{phot}}\big), \tag{8}$$

where $c_{\mathrm{CO_2},1}^{\text{phot}}$, $c_{\mathrm{CO_2},2}^{\text{phot}}$ and $c_{\mathrm{CO_2},3}^{\text{phot}}$ are parameters to describe the effect of temperature on the gross canopy photosynthesis in $\mathrm{m} \cdot \mathrm{s}^{-1} \cdot$ °C^{-2}, $\mathrm{m} \cdot \mathrm{s}^{-1} \cdot$ °C^{-1} and $\mathrm{m} \cdot \mathrm{s}^{-1}$ respectively, and c^{phot} is the $\mathrm{CO_2}$ compensation point in $\mathrm{kg} \cdot \mathrm{m}^{-3}$.[8] The maintenance respiration of the crop dry matter $\phi_{\text{resp, W}}(t)$ in $\mathrm{kg}\ \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}$ described as:

$$\phi_{\text{resp, W}}(t) = c_{\text{resp, W}} \cdot \phi_{\text{resp}}(t), \tag{9}$$

where $c_{\text{resp, W}}$ is the crop dry matter respiration rate in s^{-1} and $\phi_{\text{resp}}(t)$ is the maintenance respiration in $\mathrm{kg} \cdot \mathrm{m}^{-2}$:

$$\phi_{\text{resp}}(t) = x_{\mathrm{W}}(t) \cdot c_{Q_{10},\text{resp}}^{(x_{\mathrm{T}}(t)-25)/10}, \tag{10}$$

where $c_{Q_{10},\text{resp}}$ is the maintenance respiration Q_{10} factor. It describes how crop maintenance respiration halves for every temperature decrease of 10 °C and how it doubles for every increase of 10 °C. $\mathrm{CO_2}$ production by crop respiration $\phi_{\text{resp, c}}(t)$ in $\mathrm{kg} \cdot \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}$ is described by:

$$\phi_{\text{resp, c}}(t) = c_{\text{resp, CO}_2} \cdot \phi_{\text{resp}}(t), \tag{11}$$

where $c_{\text{resp, CO}_2}$ is the respiration coefficient for crop $\mathrm{CO_2}$ release in s^{-1}. The mass exchange of $\mathrm{CO_2}$ via the vents and due to leakage $\phi_{\text{vent,c}}(t)$ in $\mathrm{kg} \cdot \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}$ is described by:

$$\phi_{\text{vent,c}}(t) = \big(u_{\mathrm{v}}(t) \cdot 10^{-3} + c_{\text{leak}}\big)\big(x_{\mathrm{CO_2}}(t) - d_{\mathrm{CO_2}}(t)\big), \tag{12}$$

[5] Maintenance respiration refers to all respiration that does not lead to a net increase in crop dry matter. The energy it provides is used for internal processes to maintain itself.

[6] In other words, it is the amount of air present per square meter in the greenhouse and equals the height of the greenhouse.

[7] Canopy refers to the part of the crop that is above ground.

[8] This is the minimum $\mathrm{CO_2}$ concentration above which net crop photosynthesis occurs. I.e., the crop $\mathrm{CO_2}$ uptake during photosynthesis is larger than the amount released due to respiration.

where c_{leak} is the ventilation leakage through the greenhouse cover in $\text{m}^{-3} \cdot$ [air] $\text{m}^{-2} \cdot$ [gh] s^{-1}. Energy exchange through the vents and by transmission through the greenhouse cover $Q_{\text{vent, q}}(t)$ in $\text{W} \cdot \text{m}^{-2}$ is defined by:

$$Q_{\text{vent, q}}(t) = (c_{\text{cap, v}} u_{\text{v}}(t) + c_{\text{g\_o}})(x_{\text{T}}(t) - d_{\text{T}}(t)), \tag{13}$$

where $c_{\text{cap, v}}$ is the heat capacity of the greenhouse air in $\text{J} \cdot \text{m}^{-3} \cdot$ [gh] $°\text{C}^{-1}$ and $c_{\text{g\_o}}$ is the transmission of energy through the cover in $\text{W} \cdot \text{m}^{-2} \cdot$ [gh] °C. Heating due to incoming solar radiation $Q_{\text{I}_\text{o},\text{q}}(t)$ in W $\cdot\ \text{m}^{-2}$ is described as:

$$Q_{\text{I}_\text{o},\text{q}}(t) = c^{\text{rad}}_{\text{o\_g}} d_{\text{I}_\text{o}}, \tag{14}$$

where $c^{\text{rad}}_{\text{o\_g}}$ is unitless and describes the fraction of solar radiation that provides heating for the greenhouse. The canopy transpiration $\phi_{\text{transp,h}}(t)$ in $\text{kg} \cdot \text{m}^{-2} \cdot \text{s}^{-1}$ is defined as:

$$\phi_{\text{transp,h}}(t) = c^{\text{evap}}_{\text{c\_a}} \left(1 - e^{-c_{\text{LAI, W}} x_{\text{W}}(t)}\right) \cdot \left(\frac{c^{\text{sat}}_{\text{H}_2\text{O},1}}{c_{\text{R}}(x_{\text{T}}(t) + c_{\text{T}})} e^{\frac{c^{\text{sat}}_{\text{H}_2\text{O},2} x_{\text{T}}(t)}{x_{\text{T}}(t) + c^{\text{sat}}_{\text{H}_2\text{O},3}}} - x_{\text{h}}(t) \right), \tag{15}$$

where $c^{\text{evap}}_{\text{c\_a}}$ is the mass transfer coefficient for water vapor between crop canopy and greenhouse air in m s^{-1}, c_{R} is the gas constant in $\text{J K}^{-1}\ \text{kmol}^{-1}$, c_{T} is used to convert °C to K and is in K, and $c^{\text{sat}}_{\text{H}_2\text{O},1}$, $c^{\text{sat}}_{\text{H}_2\text{O},2}$, $c^{\text{sat}}_{\text{H}_2\text{O},3}$ are parameters used to compute saturated water vapor pressure, with units in J m^{-3}, unitless and °C, respectively. The mass $\phi_{\text{vent,h}}(t)$ mass exchange of water vapour through the vents and via leakage in $\text{kg} \cdot \text{m}^{-2} \cdot \text{s}^{-1}$ is defined by:

$$\phi_{\text{vent,h}}(t) = \left(u_{\text{v}}(t) \cdot 10^{-3} + c_{\text{leak}}\right)\left(x_{\text{h}}(t) - d_{\text{h}}(t)\right). \tag{16}$$

2.2.2 Measurement output

The measurement equations that convert state variables into standard sensor units, where necessary, are defined as follows:

$$\begin{aligned} y_{\text{W}}(t) &= x_{\text{W}}(t), \\ y_{\text{CO}_2}(t) &= g_h\left(x_{\text{T}}(t), x_{\text{CO}_2}(t)\right), \\ y_{\text{T}}(t) &= x_{\text{T}}(t), \\ y_{\text{RH}}(t) &= g_{\text{CO}_2}\left(x_{\text{T}}(t), x_{\text{h}}(t)\right). \end{aligned} \tag{17}$$

Measurements for dry weight and temperature are identical to the state variables, however this is not the case for CO_2 and humidity. The indoor CO_2 concentration is computed from the indoor temperature and CO_2 density via g_{CO_2}:

$$g_{\text{CO}_2}\left(z_{\text{T}}(t), z_{\text{CO}_2}(t)\right) = 10^3 \frac{R\left(z_{\text{T}}(t) + c_{\text{T}}\right)}{P M_{\text{CO}_2}} \cdot z_{\text{CO}_2}(t), \tag{18}$$

where $z_{\text{T}}(t)$ and $z_{\text{CO}_2}(t)$ represent variables describing *an* air temperature in °C and *an* air CO_2 density in $\text{kg} \cdot \text{m}^{-3}$, respectively. The function yields *an* air CO_2 concentration value in $10^3 \cdot \text{ppm}$. R is the molar gas constant in $\text{J} \cdot \text{mol}^{-1}\ \text{K}^{-1}$, P is the pressure (of 1 standard atmosphere (atm)) in Pa, M_{CO_2} is the molar mass of CO_2 in $\text{kg} \cdot \text{mol}^{-1}$. Similarly, the indoor relative humidity is computed from the indoor temperature and absolute humidity density via g_h:

$$g_h\left(z_{\text{T}}(t), z_{\text{h}}(t)\right) = \frac{R\left(z_{\text{T}}(t) + c_{\text{T}}\right)}{c^{\text{sat}}_{\text{H}_2\text{O},4} \exp\left(\frac{c^{\text{sat}}_{\text{H}_2\text{O},5} z_{\text{T}}(t)}{z_{\text{T}}(t) + c^{\text{sat}}_{\text{H}_2\text{O},6}}\right)} \cdot z_{\text{h}}(t), \tag{19}$$

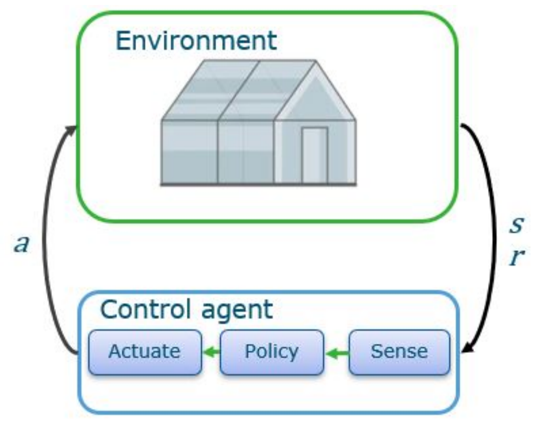
where z_{h} is *a* variable describing *an* air humidity density in $\text{kg} \cdot \text{m}^{-3}$. The function output is *an* air relative humidity value in %. $c^{\text{sat}}_{\text{H}_2\text{O},4}$, $c^{\text{sat}}_{\text{H}_2\text{O},5}$ and $c^{\text{sat}}_{\text{H}_2\text{O},6}$ are parameters used to compute saturated water vapor pressure, their units are Pa, unitless and °C, respectively.

Table 2
Model constants in order of appearance.

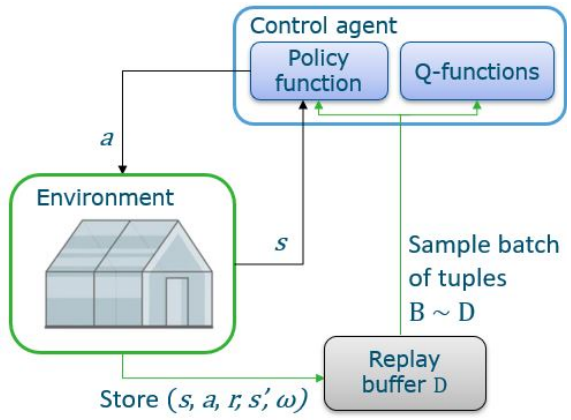
Symbol	Description	Value	Unit
$c_{\alpha\beta}$	Yield factor	0.544	-
$c_{\text{cap, c}}$	Volumetric CO_2 capacity of the greenhouse air	4.1	m^3 [air] m^{-2} [gh]
$c_{\text{cap, q}}$	Effective heat capacity of the greenhouse air	30000	J m^{-2} [gh] $^\circ\text{C}^{-1}$
$c_{\text{cap, h}}$	Volumetric humidity capacity of the greenhouse air	4.1	m^3 [air] m^{-2} [gh]
$c_{\text{LAI, W}}$	Effective canopy surface	53	m^{-2} [L] kg^{-1} [dw]
$c^{\text{phot}}_{\text{I}_\text{o}}$	Light use efficiency	$3.55 \cdot 10^{-9}$	kg [CO_2] J^{-1}
$c^{\text{phot}}_{\text{CO}_2,1}$	Parameterizes temperature influence on gross canopy photosynthesis	$5.11 \cdot 10^{-6}$	m s^{-1} $^\circ\text{C}^{-2}$
$c^{\text{phot}}_{\text{CO}_2,2}$	Parameterizes temperature influence on gross canopy photosynthesis	$2.30 \cdot 10^{-4}$	m s^{-1} $^\circ\text{C}^{-1}$
$c^{\text{phot}}_{\text{CO}_2,3}$	Parameterizes temperature influence on gross canopy photosynthesis	$6.29 \cdot 10^{-4}$	m s^{-1}
c^{phot}	Carbon dioxide compensation point	$5.2 \cdot 10^{-5}$	kg [CO_2] m^{-3} [air]
$c_{\text{resp, W}}$	Crop dry matter respiration rate	$2.65 \cdot 10^{-7}$	s^{-1}
$c_{Q_{10,\text{ resp}}}$	Maintenance respiration Q_{10} factor	2	-
$c_{\text{resp, c}}$	Coefficient for CO_2 release rate due to respiration	$4.87 \cdot 10^{-7}$	s^{-1}
c_{leak}	Ventilation leakage trough the greenhouse cover	$7.5 \cdot 10^{-6}$	m s^{-1}
$c_{\text{cap, v}}$	Heat capacity per volume unit of greenhouse air	1290	J m^{-3} [gh] $^\circ\text{C}^{-1}$
$c^{\text{g\_o}}$	Overall heat transmission through cover	6.1	W m^{-2} [gh] $^\circ\text{C}^{-1}$
$c^{\text{rad}}_{\text{o\_g}}$	Solar heat load coefficient	0.2	-
$c^{\text{evap}}_{\text{c\_a}}$	Mass transfer coefficient for water vapor between leaf and air	$3.6 \cdot 10^{-3}$	m s^{-1}
$c^{\text{sat}}_{\text{H}_2\text{O},1}$	Parameterizes saturation water vapor pressure	9348	J m^{-3}
c_{R}	Gas constant	8314	J K^{-1} kmol^{-1}
c_{T}	For conversion from °C to K	273.15	K
$c^{\text{sat}}_{\text{H}_2\text{O},2}$	Parameterizes saturation water vapor pressure	17.4	-
$c^{\text{sat}}_{\text{H}_2\text{O},3}$	Parameterizes saturation water vapor pressure	239	°C
R	Molar gas constant	8.3144598	J mol^{-1} K^{-1}
P	Pressure (of 1 standard atmosphere (atm))	101325	Pa
M_{CO_2}	Molar mass of CO_2	0.0441	kg mol^-1
$c^{\text{sat}}_{\text{H}_2\text{O},4}$	Parameterizes saturation water vapor pressure	610.78	Pa
$c^{\text{sat}}_{\text{H}_2\text{O},5}$	Parameterizes saturation water vapor pressure	17.2694	-
$c^{\text{sat}}_{\text{H}_2\text{O},6}$	Parameterizes saturation water vapor pressure	238.3	°C

3 Reinforcement Learning

3.1 RL preliminaries



(a)



(b)

Figure 4
Basic Reinforcement Learning feedback loop (a; left) and the offline learning feedback loop (b; right) that is used in the algorithms used in this work, where data is logged in a replay buffer (external data set) that is then used for learning, rather than learning from the reward signal directly.

The Reinforcement Learning problem is approached as a Markov Decision Process (MDP). This can be defined as a tuple $(\mathcal{S}, \mathcal{A}, P, R)$, where $\mathcal{S} \in \mathcal{R}^m$ represents the continuous state space and m is the dimension of the state space, $\mathcal{A} \in \mathcal{R}^n$ represents the continuous action space of the agent and n is the dimension of the action space, P the transition function $p(s'|s,a)$ that maps a state-action pair (s,a) to the next state s', and R the reward function $R(r|s,a,s')$ that returns a scalar reward value $r \in \mathcal{R}$ for taking action a in state s and ending up in the next state s'.

Reinforcement learning requires an agent and an environment for it to interact with. The environment has a state S, and at each discrete time step k the environment returns a state observation $s_k \in \mathcal{S}$ to the agent. The agent then selects and executes some action a from the action space using its policy π, which maps state observations to actions (or action distributions). Based on this interaction, the environment is updated using the transition function and the agent receives a scalar reward r_k as well as the new observed state of the environment s_{k+1}. See Fig. 4a for an example. The goal in reinforcement learning is to learn the optimal policy function π^*_θ, with parameters θ, that maximizes the expected sum of discounted future rewards, also referred to as the expected *return*:

$$G_k = \sum_{i=k}^{i=T} \left[\gamma^{i-k} r(s_i, a_i)\right], \tag{20}$$

where $\gamma \in [0,1)$ is the discount factor that indicates the relative importance of expected rewards in the distant future and T is the time step at which the episode ends. The optimal policy is then defined as:

$$\pi^*_\theta = \arg\max_{\pi_\theta} J(\pi_\theta) = \arg\max_{\pi_\theta} \sum_{i=k}^{i=T} \mathop{\mathbb{E}}_{\tau\sim\rho_{\pi_\theta}} [G_k], \tag{21}$$

where T is the final time step, $\mathbb{E}[G_k]$ is the expected return and τ is the trajectory of states and state-action pairs that follows from selecting actions using policy π_θ, starting in the first state s_0 that is sampled from an initial-state distribution $\rho(s_0)$. The expected return G_k for executing a given action a in a given state s, ending up in the next state s' and subsequently following a policy π is learned or estimated via the Q-function:

$$Q^{\pi_\theta}(s,a) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta} [G_k|s,a]\,. \tag{22}$$

The Bellman equation relates the Q-value for a given state-action pair (s,a) to the Q-value of the ensuing state-action pair (s',a') (Bellman (1957)). This means that $Q^\pi(s,a)$ can be rewritten to the following:

$$Q^\pi(s,a) = \mathop{\mathbb{E}}_{\tau\sim\pi} \left[r(s,a) + \gamma Q^\pi(s',a')\right], \quad a' \sim \pi(\cdot|s'), \tag{23}$$

where the Q-value for a given state-action pair (s, a) and given a fixed policy π can be learned iteratively based on the immediate expected received reward $r(s, a)$ and the expected future discounted rewards (Q-value) for the next state-action pair (s', a'), where action a' is taken from policy π.

3.1.1 Deep Reinforcement Learning algorithms for continuous control

Current state-of-the-art *Deep Reinforcement Learning algorithms* (DRL) for continuous control are Soft Actor-Critic (SAC) (Haarnoja et al. (2018c)) and Twin-Delayed Deterministic Policy Gradient (TD3) (Fujimoto et al. (2018)). Both of which are model-free, offline and off-policy actor-critic algorithms that build on the Deep Deterministic Policy Gradient (DDPG) algorithm (Lillicrap et al. (2016)). As such, the general structure for these algorithms is very similar, albeit slightly different than what was introduced so far due to their off-policy and offline learning nature. Their general structure will be introduced first, followed by the algorithm specific details of SAC and then TD3. (See subsection 6.2 in the Appendix for more background on DRL algorithms and why these algorithm were selected.)

Algorithm properties Both SAC and TD3 make use of two parameterized Q-functions Q_{ϕ_1} and Q_{ϕ_2}, which are also referred to as the *critics*, and one parameterized policy function π_θ, which is also referred to as the *actor*. Each of these functions is modelled with their own neural network, with parameters ϕ_1, ϕ_2 and θ, respectively. The actor maps state observations to actions as is normal, but the critics now have an additional role besides learning of the Q-function: they provide feedback on performed actions to the actor, which it in turn uses to help learn and improve the policy function. SAC and TD3 both make use of target networks for the Q-functions, these are parameterized by ϕ_1^- and ϕ_2^-, respectively. TD3 makes use of an additional target network for the policy function π_{θ^-}, parameterized by θ^-. Their use is to stabilize the learning process (Mnih et al. (2015); Fujimoto et al. (2018)). The target networks are initialized by copying the parameters of the respective main network they act as target for.

Both algorithms, while their exact approaches differ, make use of a stochastic *behaviour policy* $\pi_\theta(\cdot|s)$ to interact with the environment for the sake of exploration of the state space and to learn about the greedy deterministic *exploitation policy* $\pi_\theta(s)$ that is used when evaluating the algorithm policy. This is referred to as off-policy learning. We differentiate between a *data collection* mode, where the agent interacts with the environment, and a *learning* mode, where the various algorithm functions and parameters are updated. See Fig. 4b for a visual illustration.

Data collection The agent receives an observation of the state s_k from the environment and selects an action a_k based on a policy π_θ, the environment transitions to the next state s_{k+1} and terminates if a terminal state is reached as per usual. However, the reward r_k that is emitted by the environment is not immediately used by the agent to learn.[9] Instead, a replay buffer $\mathcal{D}$ is used that contains a history of interactions and transitions. The environment interaction is logged in a (s, a, r, s', ω) tuple that is then stored in the replay buffer[10], where ω represents the *done* signal of the environment that indicates episode termination. It has a default value of 0, and it takes a value of 1 if s' is a terminal time step. Also note that the subscripts k in the tuple have been removed and that the notation for the next state s_{k+1} has been replaced with s', as it is not important in the replay buffer from which exact time step k in the episode the tuple originates. Regardless, if this is important, information on the time step should be included in the state observation.

Learning procedure Learning of the Q-functions and the policy function is done in an offline manner by uniformly sampling batches of tuples $\mathcal{B}$ from the replay buffer, these are then used to learn the critics and the actor via their respective loss functions and stochastic gradient descent. The implementation details for both the learning of the Q-functions, as well as the learning of the policy differ between the two algorithms and these are explained in their respective sections. The target networks are updated via *Polyak updates*; in other words, the parameters of a target network slowly track the parameters of their main network counterpart.

[9] An environment is not necessarily required due to the offline and off-policy learning nature of the algorithms; in that case, a (static) data set of logged transitions is required instead. These approaches can also be combined to supplement each other, using a data set while also allowing for the gathering of new data via interactions with the environment. While the policy with which the data is obtained is not very relevant, the quality and diversity of the data in the data set are major factors that affect the learning process and the quality of the final Q-functions and the policy function. This is not something that is delved into further in this work.

[10] The size of the replay buffer is a hyperparameter that needs to be set and the buffer works with a First In, First out principle once it is full. The added value of using a replay buffer is that it improves learning efficiency: it improves data-use efficiency as it allows learning from previous experiences; additionally,it improves learning stability as it avoids biasing learning towards recent samples and breaks temporal correlations between samples; sampling from a larger data history makes the learning process more smooth as even small Q-value changes due to, for example, a bias towards recent samples or temporal correlations can lead to large changes in policy behaviour and as such could greatly affect the data distribution and cause instability in or even degradation of the learning process; and finally it makes it less likely for the policy to get stuck in local optima (Mnih et al. (2015)).

3.2 SAC

SAC makes use of soft Q-learning and a stochastic policy $\pi_\theta(\cdot|s)$. In soft Q-learning, an additional reward term for entropy $\alpha H(\pi(\cdot|s)$ is introduced in the reward function. Consequently, the optimal policy aims to maximize future expected return as well as policy entropy. In other words, it aims to maximize rewards, while acting as randomly as possible. This term helps with both exploration and regularization[11]. As such, the optimal policy π^*_θ is as follows:

$$\pi^*_\theta = \arg\max_{\pi_\theta} \sum_{i=k}^{i=T} \gamma^{i-k} \mathop{\mathbb{E}}_{\tau\sim\rho_{\pi_\theta}} \big(r(s,a) + \kappa H(\pi_\theta(\cdot|s)\big), \tag{24}$$

where κ is a temperature coefficient that states the relative importance of the entropy term with respect to the reward term, this coefficient is tuned automatically during learning. Indeed, setting κ to 0 returns the original reward function. The Soft Q-function, i.e., entropy-regularized Q-function with parameters ϕ is as follows:

$$Q^{\pi_\theta}_\phi(s,a) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}[r(s,a) + \gamma\big(Q^{\pi_\theta}_\phi(s',\tilde{a}') - \kappa \ln \pi_\theta(\tilde{a}'|s')\big), \quad \tilde{a}' \sim \pi_\theta(\cdot|s'), \tag{25}$$

where $\tilde{a}'$ is an action performed in state s' and sampled using the current behaviour policy $\pi_\theta(\cdot|s')$. Remember that approximating the Q-function is done by sampling batches from the replay buffer, as a result the entropy reward for the first time step is excluded as this action comes from the replay buffer and was sampled using a different policy. $\ln \pi_\theta(\tilde{a}'|s')$ represents the log-likelihood of taking a specific action $\tilde{a}'$ in state s'. The notations $\tilde{a}$ and $\tilde{a}'$ will be used to differentiate actions that are selected using the current behaviour policy from those that come from the replay buffer (a and a') and were obtained using a previous iteration of the learned policy or from another policy.

SAC makes use of two parameterized Q-functions to approximate the soft Q-function, Q_{ϕ_1} and Q_{ϕ_2}, which are also referred to as the *critics*, and one parameterized policy function π_θ, which is also referred to as the *actor*. Each of these functions is modelled with their own neural network, with parameters ϕ_1, ϕ_2 and θ, respectively. Each critic has a target network, $Q_{\phi_1^-}$ and $Q_{\phi_2^-}$, these are parameterized by ϕ_1^- and ϕ_2^- and initialized by copying the parameters of the function they act as target for.

Whereas the direct output of a deterministic policy function is an action vector, that is not the case for stochastic policies. Instead, the initial function outputs are two vectors describing the mean μ_θ and the (natural) log standard deviation $\ln \sigma_\theta$ for the policy distribution given a state s. The natural log is easily transformed to the standard deviation σ_θ by exponentiation. However, rather than sampling an action $\tilde{a}$ from the distribution directly, SAC makes use of the reparameterization trick by introducing a transformation function f^{S}_θ:

$$\begin{aligned}\tilde{a} &= f^{\mathrm{S}}_\theta(\epsilon; s) \\ &= \tanh\left(\mu_\theta(s) + \sigma_\theta(s) \odot \epsilon\right), \quad \epsilon \sim \mathcal{N}(0,1),\end{aligned} \tag{26}$$

where $\tilde{a}$ is the resulting action and ϵ is a noise vector drawn from a spherical Gaussian. The output is squashed with $\tanh$ to ensure actions stay within bounds ($[-1,1]$). The reasoning for the use of this reparameterization approach will be explained when policy learning is discussed. The definition for the stochastic behaviour policy $\pi_\theta(\cdot|s)$, with as output action $\tilde{a}$, is then as follows:

$$\pi_\theta(\cdot|s) = \pi_\theta(f^{\mathrm{S}}_\theta(\epsilon; s)|s), \quad \epsilon \sim \mathcal{N}(0,1), \tag{27}$$

and the greedy deterministic policy $\pi_\theta(s)$ used during evaluation episodes is:

$$\pi_\theta(s) = a = \tanh(\mu_\theta). \tag{28}$$

3.2.1 Learning procedure:

Learning of the critics is done by minimizing residuals between target values and the predicted Q-values for each Q-network independently. As such, the loss functions for the Q-networks are:

$$L(\phi_i) = \mathop{\mathbb{E}}_{(s,a,r,s',\omega)\sim\mathcal{D}}\left[\frac{1}{2}\big(Q_{\phi_i}(s,a) - \hat{q}(r,s',\omega)\big)^2\right], \quad \text{for } i \in \{1,2\}, \tag{29}$$

[11] Regularization refers to limiting or avoiding, or both, of overfitting to specific data. Overfitting can occur when a model has too many variables to fit the available data which allows too much flexibility and can generate a fit to specific details, patterns or relations in the data where there should be none, leading to poor generalization on unseen data. This can, for example, lead to the incorrect formation of very narrow peaks in the Q-function estimators that affect both target values and can be exploited by the policy function, resulting in poorer quality of learning (Goodfellow et al. (2016))

where the target Q-values $\hat{q}$ used are shared between the two networks and is computed as follows:

$$\hat{q}(r, s', \omega) = r + (1-\omega)\gamma \min_{i=1,2} Q_{\phi_i^-}(s', \tilde{a}') - \kappa \log \pi_\theta(\tilde{a}'|s')), \quad \tilde{a}' \sim \pi_\theta(\cdot|s'), \tag{30}$$

where r is short for $r(s, a, s')$ and $\tilde{a}'$ is taken from the current behaviour policy $\pi_\theta(\cdot|s')$. Only the lowest predicted Q-values for the following state-action pairs $(s', \tilde{a}')$ out of the two target Q-network predictions are used to compute the target values for the loss function of both Q-functions. The parameters of the critics are then updated by minimizing their respective loss functions via stochastic gradient descent.

The policy loss function to be minimized makes use of the minimum predicted Q-value between the two Q-function approximators and is as follows:

$$L_\pi(\theta) = \mathop{\mathbb{E}}_{\substack{s\sim\mathcal{D},\\ \tilde{a}\sim\pi_\theta}} \left[\kappa \log \pi_\theta(\tilde{a}|s) - \min_{i=1,2} Q_{\phi_i}(s, \tilde{a})\right], \tag{31}$$

which is then rewritten using the previously mentioned reparameterization trick, so replacing $\tilde{a}$ with $f^s(\epsilon; s)$. This replaces the expectation over the action distribution, which depends on θ, with an expectation over noise that does not depend on θ, reducing variance in the policy gradient estimator. The policy is then updated by minimizing the following loss function using stochastic gradient descent:

$$L_\pi(\theta) = \mathop{\mathbb{E}}_{\substack{s\sim\mathcal{D},\\ \epsilon\sim\mathcal{N}}} \left[\kappa \log \pi_\theta(f_\theta^S(\epsilon; s)|s) - \min_{i=1,2} Q_{\phi_i}(s, f_\theta^S(\epsilon; s))\right]. \tag{32}$$

The temperature parameter κ is tuned automatically during the algorithm learning process by minimizing the target entropy loss function:

$$L(\kappa) = \mathop{\mathbb{E}}_{\substack{s\sim\mathcal{D},\\ \tilde{a}\sim\pi_\theta}} [-\kappa \log \pi_\theta(\tilde{a}, s) - \kappa\bar{H}], \tag{33}$$

where $\bar{H}$ is the minimum target entropy threshold. The temperature parameter is then updated using stochastic gradient descent.

Updating of the target Q-networks is done based on weights of the target networks and the main Q-networks:

$$\phi_i^- \leftarrow \eta\phi_i + (1-\eta)\phi_i^- \qquad \text{for } i \in \{1, 2\}, \tag{34}$$

where $\eta = 0.005$ is the Polyak parameter that indicates the relative importance of the main and target networks during target network updates.

3.3 TD3

TD3 learns a deterministic policy $\pi_\theta(s)$. It makes use of two parameterized Q-functions Q_{ϕ_1} and Q_{ϕ_2}, which are also referred to as the *critics*, and one parameterized policy function π_θ, which is also referred to as the *actor*. Each of these functions is modelled with their own neural network, with parameters ϕ_1, ϕ_2 and θ, respectively. Each critic has a target network, $Q_{\phi_1^-}$ and $Q_{\phi_2^-}$, these are parameterized by ϕ_1^- and ϕ_2^- , the actor also has a target network $\pi_{\theta^-}^-$ with parameters θ^-. Target networks are initialized by copying the parameters of the function they act as target for.

Action noise is added to the output of the deterministic policy during training for the sake of exploration, this results in the following behaviour policy:

$$\pi_\theta(\cdot|s) = \text{clip}\left(\pi_\theta(s) + \epsilon, a^{\text{min}}, a^{\text{max}}\right), \quad \epsilon \sim \mathcal{N}(0, \tilde{\sigma}), \tag{35}$$

where $\tilde{\sigma}$ is the standard deviation used to draw samples for the action noise covariance matrix, this a hyperparameter that is tuned in this work and the value will be given later, and a^{min} and a^{max} represent the action constraint of -1 and 1, respectively, and ensure actions remain within the valid action range.

3.3.1 Learning procedure

The critics are learned by minimizing the following loss function for each Q-network separately:

$$L(\phi_i) = \mathop{\mathbb{E}}_{(s,a,r,s',\omega)\sim\mathcal{D}} \left[\left(Q_{\phi_i}(s, a) - \hat{q}(r, s', \omega)\right)^2\right], \quad \text{for } i \in \{1, 2\}. \tag{36}$$

where the target Q-values $\hat{q}$ used are shared between the two networks. Only the lowest predicted Q-values for the following state-action pairs $(s', \tilde{a}')$ out of the two target Q-network predictions are used to compute the target values for the loss function of both Q-functions. This helps alleviate the Q-value overestimation that was problematic in its predecessor DDPG. Target values are computed as follows:

$$\hat{q}(r, s', \omega) = r + \gamma(1-\omega) \min_{i=1,2} Q_{\phi_i^-}\big(s', a'\big), \quad \tilde{a}' \sim \pi_{\theta^-}^-(\cdot|s'), \tag{37}$$

where the action $\tilde{a}'$ is taken from the behaviour policy of the target network $\pi_{\theta^-}^-(\cdot|s')$:

$$\pi_{\theta^-}^-(\cdot|s') = \text{clip}\left(\pi_{\theta^-}^-(s') + \text{clip}(\epsilon, -c^-, c^-), a^{\text{min}}, a^{\text{max}}\right), \quad \epsilon \sim \mathcal{N}(0, \tilde{\sigma}^-) \tag{38}$$

where the target policy $\pi_{\theta^-}^-$ is used to compute the Q-learning target and (clipped) Gaussian noise is added to the target policy to smooth the Q-function estimate. The standard deviation $\tilde{\sigma}^-$ used for the added target policy noise that is drawn from a spherical Gaussian has a default value of 0.2, c^- represents the clip value of 0.5 for the added target action noise to ensure the target action remains close to the deterministic target action. [12]

The policy and all of the target networks are updated less frequently, once every two critic updates, than the critic networks to reduce negative effects of errors in the Q-value estimates on policy learning. The policy is learned by maximizing Q_{ϕ_1}:

$$J(\theta) = \mathop{\mathrm{E}}_{s\sim\mathcal{D}}\left[Q_{\phi_1}(s, \pi_\theta(s))\right]. \tag{39}$$

The weights of the critic target networks are updated as in SAC:

$$\phi_i^- \leftarrow \eta\phi_i + (1-\eta)\phi_i^- \quad \text{for } i \in \{1, 2\}, \tag{40}$$

and the weights for the policy target network are updated in a similar manner:

$$\theta^- \leftarrow \eta\theta + (1-\eta)\theta^-. \tag{41}$$

[12] Overfitting to narrow spikes in Q-value estimates is a problem when deterministic policies are used to compute target values, making them sensitive to errors in the Q-function approximators that can be exploited by the policy, result in erroneous behaviour or lead to high-variance in the target values. The solution proposed by the authors is to smooth the target policy as a regularization method. This means that action values for a narrow area around the deterministic target actions are fitted during training with the idea being that similar actions should have similar Q-value estimates, resulting in a more smooth Q-function.

3.4 Environment description

This section aims to provide an overview of the environment model transferred from the process model presented in Chapter 2, so that the RL agent can interact with. This environment description involves a comprehensive description of several crucial elements: the observation space returned to the agent, the action space available to the agent, the transition function governing state changes, and the underlying structure of the reward function.

Observation space The observation space can be denoted by the tuple $S = (Y, X_s, U, D, \hat{D}_f.^{[13]}, K)$, where $y \in Y \subseteq \mathbb{R}^4$ are the greenhouse measured variables, $x_s \in X_s \subseteq \mathbb{R}^2$ are the estimated state variables for indoor CO_2 density and absolute humidity, $d \in D \subseteq \mathbb{R}^4$ represents the current weather, $\hat{d}_f \in \hat{D}_f \subseteq \mathbb{R}^{4 \cdot \mathrm{N}_p}$ represents the predicted weather forecast where N_p is the number of future steps in the prediction horizon, $u \in U \subseteq \mathbb{R}^3$ represent the control signals and $k \in K \subseteq \mathbb{Z}_{\geq 0}$ the time step. The time interval between two discrete time steps Δt is 900 seconds, or 15 minutes. This value is used as the time scale of the greenhouse indoor climate is between 15 minutes and 1 hour, as per Straten et al. (2000). A state observation $s(k)$ is denoted as $(y(k), x_s(k), u(k-1), d(k), \hat{d_f}(k)^{[13]}, k(k))$.

The measured indoor variables $y(k)$ are described in subsection 2.2. We assume measurements are accurate - that is, there is no noise in the measurements. x_s consists of the indoor CO_2 density ($x_{\mathrm{CO_2}}(k)$) and the absolute humidity ($x_{\mathrm{h}}(k)$), this is the subset of the state variable vector described in subsection 2.2 that does not overlap with the measurement variables. We assume these can be estimated correctly from the measurements by rewriting their respective equations in the measurement output. [14]

The previous control signals $u_i(k-1)$ are also included as they are only allowed to change a maximum of 10% of their maximum values at each time step. The weather observation at the current time step $d(k)$ as described in subsection 2.2, as well as the weather forecast $\hat{d_f}(k)$ are included in the observation space. We assume the weather forecast is accurate and the weather time-series develops as per the forecast. $\hat{d_f}$ is then defined as follows:

$$\hat{d}_f(k) = \begin{pmatrix} d(k+1)^T & d(k+2)^T & \cdots & d(k+\mathrm{N}_p)^T \end{pmatrix}^T. \tag{42}$$

The agent is considered time-aware, i.e., the discrete time step k since the start of the episode is included in the observation space. The size of the observation space is then $14 + 4 \cdot N_p$. It is assumed that the information in the observation space is perfectly accurate — that is, there is no noise in the measurements and estimations as mentioned previously, and the weather time-series develops as per the forecast.

Action space The action space ($a \in A \subseteq [-1, 1]^3$) is continuous. New control signals $u_(k)$ are obtained by mapping the action vector to the maximum allowed change in control signal per time step, $\delta u^{\max} = 0.10 \cdot u^{\max}$, this is then added to the previous control signals and the result is clipped to the physical ranges of the actuator. This translates to the following equation:

$$u(k) = clip\left(u(k-1) + a(k) \cdot \delta u^{\max}, u^{\min}, u^{\max}\right). \tag{43}$$

The control input constraints are as follows:

$$\begin{aligned} u_{\mathrm{CO_2}} &\in [0, 1.2]\ \mathrm{mg} \cdot \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}, \\ u_{\mathrm{v}} &\in [0, 7.5]\ \mathrm{mm} \cdot \mathrm{s}^{-1}, \\ u_{\mathrm{q}} &\in [0, 150]\ \mathrm{W} \cdot \mathrm{m}^{-2}, \end{aligned} \tag{44}$$

and as such $u^{\min}$ and $u^{\max}$ are defined as:

$$\begin{aligned} u^{\min} &= \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T, \\ u^{\max} &= \begin{pmatrix} 1.2 & 7.5 & 150 \end{pmatrix}^T. \end{aligned} \tag{45}$$

Transition function The environment transition starts with updating the control signals based on actions selected by the agent, this develops as per Eq. (43). Subsequently, the state variables x and the sensor measurements y transition according to the discretized model Eq. (1) that has been discussed in subsection 2.1. The current and forecast weather data in the observation space are deterministic and slide through the given weather data set and finally the time step increases incrementally by one. Pseudocode for the transition function is given in Algorithm 1.

Reward function The agent is rewarded for maximizing its profit, while minimizing constraint violations. Optimizing net profit requires careful balancing of income by crop production and the costs

[13] The forecast is excluded from the observation space and transition function if no forecast horizon is used.

[14] $x_{\mathrm{CO_2}}$ and x_{h} are estimated by rewriting Eq. (18) and Eq. (19), respectively. This is referred to as soft sensing (van Mourik et al. (2021)).

Algorithm 1 Transition function pseudocode

```
1: procedure
2: $u(k) \leftarrow clip\big(u(k-1) + 0.1 \cdot a(k), u^{\min}, u^{\max}\big)$
3: $x(k+1) \leftarrow f\big(x(k), u(k), d(k); c\big)$
4: $y(k+1) \leftarrow g\big(x(k+1); c\big)$
5: $x_s(k+1) \leftarrow x_{2,4}(k+1)$
6: $d(k+1) \leftarrow d(k+1)$
7: $\hat{d_f}(k+1) \leftarrow \hat{d_f}(k+1)$[13]
8: $k(k+1) \leftarrow k(k) + 1$
9: $s(k+1) \leftarrow \big(y(k+1), x_s(k+1), u(k), d(k+1), \hat{d_f}(k+1), k(k+1)\big)$
```

of using resources. Maximizing net profit is a commonly used objective in optimal control of greenhouse climate using RL (Hemming et al. (2019, 2020); Wang et al. (2020); Zhang et al. (2021); An et al. (2021); Cao et al. (2022)). Furthermore, models do not completely describe the true system over the entire state-space and negative effects of unfavourable conditions such as extremely high temperatures, CO_2 concentrations or relative humidity levels on crop growth are not fully captured by the model. As a result, the controller will not avoid these states, so it is necessary to impose constraints on the indoor climate variables to ensure that undesirable conditions are avoided. This means that the reward function is to be augmented with penalty terms for indoor climate constraint violations, and the structure of the reward function is now similar to that of an objective function used for MPC (van Henten and Bontsema (2009)).[15] The full reward function for taking an action $a(k)$ given an observed state $s(k)$ at time step k and ending up in the next state $s(k+1)$ following the transition function is defined as follows:

$$R(k) = \beta_1 c_{let,p}\phi_{\text{let}}(k+1) - \beta_2 c_{\text{CO}_2}\frac{\Delta t}{1e6}u_{\text{CO}_2}(k) - \beta_3 c_q \frac{\Delta t}{3.6e6}u_{\text{q}}(k+1) + p_{\text{CO}_2}(k+1) + p_T(k+1) + p_h(k+1). \quad (46)$$

The first term in the reward function describes income due to crop growth, where $c_{\text{let, p}}$ is the price per produced lettuce head and $\phi_{\text{let}}(k+1)$ represents the difference in lettuce head yield between the current and the next observed state. The second and third term assigns economic costs to the control inputs, where c_{CO_2} is the cost per kg of CO_2 used and c_q the price per kWh energy used. Prices for crops and resources are assumed to be static. β_1, β_2 and β_3 are constants that can be altered from their default values of 1 in order to prioritize crop growth, or to reduce environmental impacts by increasing costs associated with CO_2 usage, for example. p_{CO_2}, p_T and p_h are penalties for CO_2, temperature and humidity boundary violations, respectively, and are computed for indoor conditions at $s(k+1)$. The values associated with the constants in the reward function and the upcoming penalty functions are given in Table 3.

The difference in lettuce head yield between two time steps is estimated from crop dry weight as follows:

$$\phi_{\text{let}}(k) = \text{c}_{\text{let, hw}} \cdot dy_{\text{W}}(k) \cdot \text{c}_{\text{let, head}} \quad (47)$$

where $dy_{\text{W}}(k)$ is the difference in crop dry weight between a current and the preceding time step - $dy_{\text{W}}(k) = y_{\text{W}}(k) - y_{\text{W}}(k-1)$, and $\text{c}_{\text{let, hw}}$ is the ratio of crop dry weight to harvestable crop weight that is used to compute the increase in harvestable yield. This is then converted to an increase in lettuce heads produced via the constant $\text{c}_{\text{let, head}}$ that describes the harvestable weight of a singular, grown lettuce head. The dry weight: harvestable weight ratio varies in reality as it includes the crop fresh weight which depends on environmental conditions, but for practical reasons a constant ratio is assumed. The same holds for the assumption on the weight of a lettuce head. The purpose of this conversion is to have a rough translation of crop dry weight to income. The CO_2 upper boundary violation penalty term is computed as follows:

$$p_{CO_2}(k) = \begin{cases} c_{pC} \cdot (y_{\text{CO}_2}(k) - y_{\text{CO}_2}^{\max}), \text{ if } y_{\text{CO}_2}(k) > y_{\text{CO}_2}^{\max} \\ 0, \text{ otherwise,} \end{cases} \quad (48)$$

where c_{pC} in € $\cdot 10^3$ ppm^{-1} is a constant used to penalize upper CO_2 boundary violations. Temperature boundary violation penalties depend on whether it is an upper or lower boundary violation:

$$p_T(k) = \begin{cases} -c_{p_{Tl}} \cdot \big(y_{\text{T}}^{\min} - y_{\text{T}}(k)\big), \text{ if } y_{\text{T}}(k) < y_{\text{T}}^{\min} \\ -c_{p_{Tup}} \cdot \big(y_{\text{T}}(k) - y_{\text{T}}^{\max}\big), \text{ if } y_{\text{T}}^{\max} < y_{\text{T}}(k) \\ 0, \text{ otherwise,} \end{cases} \quad (49)$$

[15] These penalty terms were not used in the previous works on RL for greenhouse control, as their control algorithms decided on indoor climate set-points, rather than direct actuator control. Regulation of set-points is done using simple PID-controllers (or variants thereof).

where $c_{p_{Tlwb}}$ and $c_{p_{Tupb}}$ are the constants used to penalize lower and upper temperature boundary violations in € · C^{-1}, respectively. Finally, relative humidity constraint violations are penalized via the following equation:

$$p_h(k) = \begin{cases} c_{p_h} \cdot (y_{\text{RH}}(k) - y_{\text{RH}}^{\max}), \text{ if } y_{\text{RH}}(k) > y_{\text{RH}}^{\max} \\ 0, \text{ otherwise,} \end{cases} \tag{50}$$

where c_{p_h} in € · %$^{-1}$ is the constant to penalize upper relative humidity boundary violations. Note that all penalty constants are multiplied with their respective units and a euro constant to ensure units used across all terms in the reward function are consistent. Their values are found empirically.

Initial conditions The initial state for the indoor conditions are set and the control signals for the first time step of each run are also predefined. Their values are given in Eq. (51). The first control signals are performed automatically when the environment is reset, i.e., a first step is always performed with $a = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T$ and this transition is not logged in the replay buffer. The subsequent control signals for each of the actuators are decided on by the agent.

$$\begin{aligned} x(0) &= \begin{pmatrix} 0.0035 & 0.001 & 15 & 0.008 \end{pmatrix}^T, \\ y(0) &= g(x(0); c)^T, \\ u(0) &= \begin{pmatrix} 0 & 0 & 50 \end{pmatrix}^T. \end{aligned} \tag{51}$$

Table 3
Reward function constants and environment constraints.

Constant	Value	Unit
$c_{\text{let, p}}$	0.50	€· head^{-1}
$c_{\text{let, hw}}$	17	kg · kg dw^{-1}[16]
$c_{\text{let, head}}$	0.25	kg · head^{-1}
c_{CO_2}	0.1906[17]	€· kg CO$_2^{-1}$
c_q	0.1281	€· kWh^{-1}
$c_{p_{p_C}}$	$\frac{1}{20}$	€ · 10^3 ppm^{-1}
$c_{p_{T_{lwb}}}$	$\frac{1}{300}$	€· °C^{-1}
$c_{p_{T_{Upb}}}$	$\frac{1}{200}$	€· °C^{-1}
$c_{p_{p_h}}$	$\frac{1}{50}$	€ · %$^{-1}$
$y_{\text{CO}_2}^{\max}$	1.6	$\cdot 10^3$ ppm
$y_{\text{T}}^{\min}$	15	°C
$y_{\text{T}}^{\max}$	25	°C
$y_{\text{RH}}^{\max}$	90	%

[16] Taken from Baeza et al. (2021).
[17] CO$_2$ price from van Henten and Bontsema (2009) in Dutch Guilder converted to euros is used, without rounding decimals: $(42e-2/2.20371)$.

4 Results

4.1 Experiment setup

The problem is considered episodic, one crop cycle of forty days is assumed with discrete time steps of fifteen minutes and the episodes terminates after exactly fourty days at $k = 3840$ when the crops are harvested. Training is done on 2014 climate data and policy generalization performance is tested using 2020 climate data. For each algorithm the effect of a variety of weather prediction horizons $N_p \in \{0, 3, 7, 11, 15, 19, 23\}$ on profit was be explored, where N_p is the number of time steps ahead of the current time k.

Every experiment is performed five times using different seeds ($\in \{27, 28, 42, 43, 94\}$). Each run starts with a warm-up phase consisting of three episodes where the agent takes random, uniformly sampled actions to collect data for the replay buffer. No learning happens during this warm-up phase. This is followed by the learning-phase that consists of 47 training episodes where the behaviour policy is used to interact with the environment and learning occurs. Every training episode is followed by two evaluation episodes where the learned greedy deterministic policy is applied to both the training and test weather data. These trajectories are not added to the replay buffer. Results of the evaluation episodes across the five seeds are logged and shown in the results section.

During runs the observations returned to the agent and samples drawn from the replay buffer used to train the actor and the critics are normalized with a running mean and standard deviation and then clipped to $[-10, 10]$ to avoid extreme values, as per the *VecNormalize* wrapper of Stable-Baselines3. (s, a, r, s') tuples stored in the replay buffer are not normalized. Additionally, the state variables x were clipped to minimum values of $\begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}$ and maximum values of $\begin{pmatrix} 400 & 0.00400 & 40 & 0.051 \end{pmatrix}$. While the warm-up phase still results in extreme indoor conditions with large constraint violations and thus initially in low quality data in the replay buffer, this was found to be useful as it helps avoid extremely poor quality data (i.e., where very large penalties occur) from entering the replay buffer during the warm-up phase.[18] We found this to be helpful in agent learning to avoid premature policy convergence and biasing of the policy to the initial samples experienced.[19]

4.2 Weather data

Training data Weather data for March 2nd - April 11th 2014 from the VenLow greenhouse in Bleiswijk (Kempkes et al. (2014)) is used to train the control agents. The dataset consists of samples captured at a five-minute time intervals and contains the outdoor global radiation [$\mathrm{W} \cdot \mathrm{m}^{-2}$], temperature [°C], relative humidity [%] and CO_2 concentration [ppm]. CO_2 data is converted in to CO_2 densities in kg $\cdot\ \mathrm{m}^{-3}$ and relative humidity in to absolute humidity in kg $\cdot\ \mathrm{m}^{-3}$ by rewriting equations Eq. (18) and Eq. (19). The five-minute data is then resampled to 15 minutes.

Test data Weather data for March 2nd - April 11th 2020 from the Second Autonomous Greenhouse Challenge is used as test data[20]. This data comes from measurements at the same location as the traing data, Bleijswijk. Radiation, temperature and relative humidity are reported at five-minute time intervals and was processed in the same manner as the training data. Two hours of data in the early morning of March 29th 2020 is missing and filled using linear interpolation. CO_2 data is not reported, as such the outside CO_2 density trajectory from 2014 data was reused instead. This assumption leads to a slight underestimation of outside CO_2 levels as CO_2 levels are rising globally (Friedlingstein et al. (2022)). The flow of CO_2 between the greenhouse and the outside world depends on the difference in CO_2 densities between the two. These differences are generally large due to CO_2 injection in greenhouses, and the slight underestimation is of little effect on this scale. It does still mean that CO_2 losses to the outside world are ever so slightly overestimated in our experiments. However, underestimation of outdoor CO_2 levels will always be the case when training on past data with aim of implementation in the future.

Visualized time-series for both weather sequences can be found in the appendix (Fig. 10).

4.3 Hyperparameter tuning

Agent and network hyperparameters were found empirically and are specified in Table 4. The TD3 algorithm adds an additional hyperparameter for the standard deviation used for the action noise. This is tuned for each prediction horizon N_p individually using values in $\{0.1, 0.2, \ldots, 0.6\}$ and the value

[18] Note that the maximum clip values for crop dry weight and for absolute humidity are way higher than can ever be reached but are reported for sake of completeness.

[19] We explored using different numbers of episodes in the warm-up phase and adjusting the number of learning episodes accordingly to keep the total number of episodes in an experiment at 50.

[20] Available at https://www.kaggle.com/datasets/piantic/autonomous-greenhouse-challengeagc-2nd-2019

Table 4
Agent hyperparameters

	Value
Learning rate α	$1{\cdot}10^{-3}$
Hidden units	32
Hidden layers	2
Discount factor γ	0.8
Activation function	ReLU
Minibatch size	64
Replay buffer size	100k

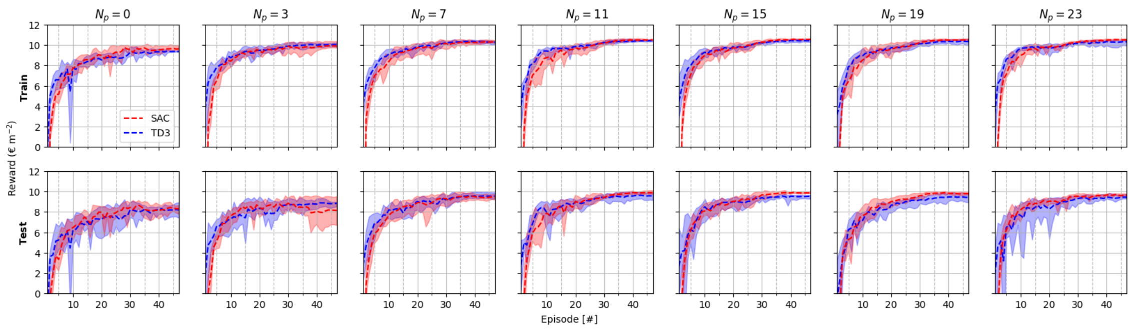


Figure 5
Learning curves describing total received reward on the training weather data set (upper row) and test weather data (lower row) using the greedy policy for SAC and TD3. The initial random action episodes are not included. The dotted line represents mean value per evaluation episode, outer lines represent minimum and maximum observed values across the five seeds.

that yields the best final episode performance on the test data is according to the criterion $\bar{R} - \sigma_R$ is reported, where $\bar{R}$ and σ_R refer to the average and standard deviation of the final rewards across the seeds, respectively. This is done due to a preference for a more consistent and robust performance that is less sensitive to randomness across the seeds during the learning process, as is done Duan et al. (2016). Default values of the Stable-Baselines3 implementations are used for hyperparameters that are not directly specified here (Raffin et al. (2021)). Adam optimizer is used (Kingma and Ba (2014)) with a linear learning schedule, where the learning schedule decreases linearly from its starting value to zero between the first and final training step.

4.4 Simulation results

Overall algorithm performance on the reward function is presented and discussed first. This is followed by an in-depth investigation of how the weather forecast horizons and the algorithms affect the final observed net profit, crop yield, control usage, and violations of the indoor climate constraints. A time series for the policy rollout for a single episode is presented in the final subchapter.

Note that a direct comparison with other works is neither feasible nor sensible, as controller performance is highly dependent on greenhouse location, type, available sensors and actuators, as well as crop species, weather conditions, used simulation settings and the used reward or objective function (van Beveren et al. (2015a); Knibbe et al. (2022)).

4.4.1 Overall performance

The central question discussed in this chapter is:

- *What is the effect of the prediction horizon on overall controller performance?*

Reward learning curves are presented in Fig. 5 and final values are reported in Table 5. Do note that differences in weather play a major role in the achievable reward signal and is one of the main reasons for the absolute differences between training and test results. This also means that it is more difficult to quantify overfitting to the training data compared to traditional Machine Learning approaches. We define a *generalization error* as the difference between the average final train and test performance. This quantification allows us to perform a relative comparison of overfitting between our experiments.

Table 5
Rewards for the final learned policies by SAC and TD3 on the fixed weather data applied to training and testing datasets over five seeds for various weather forecast horizons. Mean and (standard deviation) values are reported.

	0	3	7	11	15	19	23
SAC							
Train	9.62 (0.14)	9.91 (0.23)	10.29 (0.10)	10.52 (0.05)	10.56 (0.04)	10.51 (0.05)	10.51 (0.06)
Test	8.30 (0.30)	8.14 (0.99)	9.46 (0.21)	9.89 (0.13)	9.87 (0.06)	9.81 (0.09)	9.64 (0.07)
TD3							
Train	9.36 (0.04)	10.03 (0.14)	10.27 (0.14)	10.43 (0.06)	10.42 (0.10)	10.34 (0.19)	10.32 (0.19)
Test	8.20 (0.54)	8.83 (0.15)	9.55 (0.20)	9.57 (0.25)	9.53 (0.13)	9.43 (0.30)	9.44 (0.16)

Experiment time across five seeds for one weather forecast scenario took approximately 2 hours and 30 minutes in total for SAC experiments and 1 hour and 50 minutes in total for TD3 experiments. TD3 training time is shorter compared to SAC as it performs only half the total policy and target updates due to its policy and target network update delay.

From the learning curves, we see both training and test performances improve during the entire learning procedure, with the exception of the experiment with a 3-step forecast using SAC. This experiment shows a degradation in test performance for at least one of its seeds in the final ten episodes. This does not show in the training learning curve, indicating performance degradation is due to overfitting. TD3 generally performs better in the first five to ten episodes but is then overtaken by SAC. Asymptotic performance for both algorithms and on both the train and test weather sequences improves with increasing forecast horizons from 0 to 15 steps. We see that the generalization error decreases with increasing forecast horizons for both algorithms, indicating that the use of forecasts improves generalization performance and reduces the observational overfitting that occurs at lower forecast horizons. Additionally, variance across the seeds lowers with increased horizons lengths. This highlights the importance of using forecasts for both training and generalization performance as well as well as reducing algorithm sensitivity to random seeds.

Overfitting when no forecast or a minimal forecast is used makes sense when returning to the basics of RL. The control agent has to estimate the future discounted reward for a state-action pair and future rewards are heavily dependent on future weather, on which the agent has very little information in this case. So, while the agent may be able to estimate the future rewards adequately in the training case, it implicitly does so based on the static weather sequence used for training and it does not have enough information in the observed state to generalize this as well to different weather sequences. Generalization performance can also suffer when no or limited forecast information is provided due to increased risk that information on crop dry weight and time-awareness is misused by the control agent to differentiate between very similar weather observations due to the use of a static weather sequence.

For forecast horizons of 11 steps or more, we see that SAC performs better than TD3 on both the train and test data. It shows lower variance across the various seeds used in the learning procedure as well as better asymptotic performance. Optimal performance is obtained with a forecast of 15 steps for both algorithms.

A degradation in generalization performance is visible for SAC at $N_p = 23$ compared to $N_p = 19$ while train performance remains similar, indicating that the agent is now overfitting to the noise in the observed forecast. This is most likely the result of (1) the discount factor, and (2) using a static weather data set. It is important to take the discount factor of 0.80 used to train the algorithms into consideration here, as this means that rewards 11, 15, 19 and 23 steps in the future are assigned $8.6\%, 3.5\%, 1.4\%$ and 0.6% of their actual reward value when estimating the Q-value, respectively. Especially in the case of the latter two, it means that - when adding additional forecast knowledge in the state observation - this is barely used in the Q-function and may clutter the observation space with redundant information or noise. This affects the learning process speed and quality and increases the risk of overfitting.

4.4.2 Application

As we are dealing with a multi-objective reward function, it is important to also investigate the individual components of the reward function to gain a better understanding of agent behaviour and how trade-offs are made (Dulac-Arnold et al. (2019)) and what this means in the application context. The following questions are examined in this chapter to investigate the results from an application point of view:

- *What is the effect of the prediction horizon on net profit?*
- *What is the effect of the prediction horizon on crop dry weight yield?*
- *What is the effect of the prediction horizon on total control use?*
- *What is the effect of the prediction horizon on climate constraint violations?*

The latter two questions are answered for each of the control actions and climate constraints separately. The chapter is concluded with a time-series analysis of a policy from the best performing agent setup.

Table 6
Net profits in euros per m^2 for the final learned policies by SAC and TD3 on the fixed weather data applied to training and testing datasets over five seeds for various weather forecast horizons. Mean and (standard deviation) values are reported.

	0	3	7	11	15	19	23
SAC							
Train	10.73	10.86	11.11	11.26	11.28	11.28	11.28
	(0.11)	(0.24)	(0.11)	(0.07)	(0.07)	(0.04)	(0.08)
Test	10.49	10.20	11.30	11.61	11.59	11.59	11.56
	(0.33)	(0.82)	(0.22)	(0.11)	(0.02)	(0.07)	(0.13)
TD3							
Train	10.46	10.89	11.11	11.29	11.23	11.23	11.30
	(0.04)	(0.13)	(0.17)	(0.03)	(0.09)	(0.06)	(0.09)
Test	10.51	10.75	11.41	11.44	11.46	11.48	11.66
	(0.28)	(0.14)	(0.18)	(0.12)	(0.19)	(0.12)	(0.16)

What is the effect of the prediction horizon on net profit?
The net profit depends on resource costs and income from crop yield, these will be discussed later on in this chapter. Final net profits are reported in Table 6, the learning curves are located in the appendix. Net profit increases with increasing horizon length up to an 11-step horizon, after which it plateaus in both the train and test case for SAC. TD3 shows similar behaviour and net profit in the train case plateaus at the same level, but net profit on the test case keeps improving slightly with forecast horizon. TD3 net profit in the test case shows larger variance between seeds for horizons of 11 steps or more and is generally lower than net profits obtained with SAC, with the exception of the longest forecast explored. Higher net profit values can be obtained when neglecting climate constraints and this will be touched upon in the discussion.

Table 7
Crop dry weight yield in $g \cdot m^{-2}$ (instead of $kg \cdot m^{-2}$ for sake of readability) for the final learned policies by SAC and TD3 on the fixed weather data applied to training and testing datasets over five seeds for various weather forecast horizons. Mean and (standard deviation) values are reported.

	0	3	7	11	15	19	23
SAC							
Train	426.39	424.91	428.82	432.97	432.96	433.43	433.15
	(3.60)	(4.62)	(3.25)	(1.67)	(2.00)	(1.11)	(1.97)
Test	443.63	433.33	456.66	465.94	463.73	464.05	464.20
	(10.00)	(19.51)	(6.34)	(2.23)	(0.75)	(2.06)	(4.42)
TD3							
Train	416.04	425.90	428.22	433.08	430.72	431.80	433.45
	(2.81)	(3.66)	(5.34)	(0.72)	(3.19)	(2.10)	(2.63)
Test	445.17	446.57	459.56	461.10	458.95	461.15	466.36
	(7.65)	(5.73)	(4.96)	(3.35)	(5.82)	(4.38)	(4.71)

What is the effect of the prediction horizon on crop dry weight yield?
Final crop yields are reported in Table 7, the learning curves are located in the appendix. Note that values are reported in grams rather than kilograms for sake of readability. SAC yield is higher when

no horizon is used is than when a three-step horizon is used, this is due to higher CO_2 injection in the no-horizon case and will be elaborated on when control use is discussed. SAC yields for horizons of 11 steps or greater are generally stable in both the training and test cases. TD3 generally reports slightly higher yields in the test case compared to SAC for forecasts of 7 steps or fewer, but reports yields that are 3-5 grams lower for the remaining forecasts with exception of the longest forecast where a slightly higher yield is found. Using the reward function, this roughly translates to a difference of 10.2 - 17 cents m^{-2} in income. Finally, variance between seeds in the test case and for horizons larger than 7 steps is lower for SAC than for TD3.

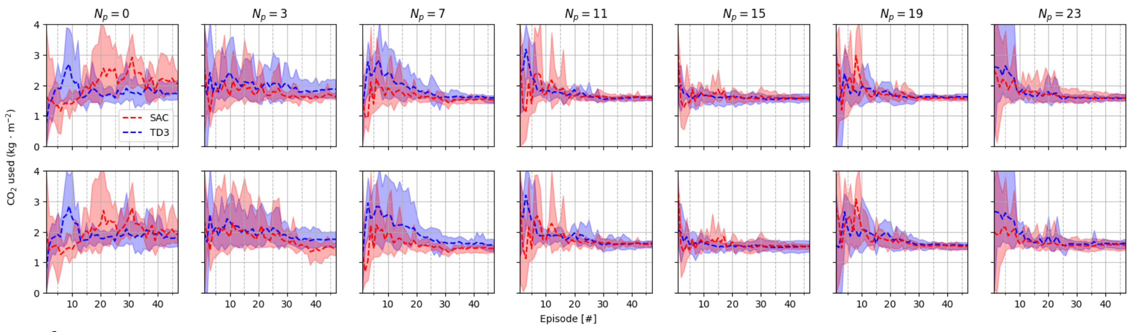


Figure 6
Learning curves describing total CO_2 use per episode on the training weather data set (upper row) and test weather data (lower row) using the greedy policy for SAC and TD3. The initial random action episodes are not included. The dotted line represents mean value per evaluation episode, outer lines represent minimum and maximum observed values across the five seeds.

Table 8
CO_2 usage in kg · m^{-3} for the final learned policies by SAC and TD3 on the fixed weather data applied to training and testing datasets over five seeds for various weather forecast horizons. Mean and (standard deviation) values are reported.

	0	3	7	11	15	19	23
SAC							
Train	2.06 (0.20)	1.63 (0.08)	1.52 (0.06)	1.58 (0.05)	1.57 (0.05)	1.58 (0.06)	1.58 (0.08)
Test	1.91 (0.23)	1.45 (0.14)	1.45 (0.08)	1.60 (0.06)	1.54 (0.06)	1.53 (0.08)	1.57 (0.09)
TD3							
Train	1.72 (0.22)	1.87 (0.24)	1.59 (0.09)	1.60 (0.05)	1.57 (0.09)	1.62 (0.06)	1.58 (0.07)
Test	1.76 (0.20)	1.75 (0.20)	1.57 (0.13)	1.62 (0.08)	1.52 (0.12)	1.56 (0.08)	1.62 (0.13)

What is the effect of the prediction horizon on control use?

CO_2 use:
Total CO_2 use for the final policies are given in Table 8 and learning curves are shown in Fig. 6. Use of forecasts leads to reduced CO_2 use compared to the case without a forecast, but the extent of this still shows some variance. SAC reports an immediate decrease in use when horizons of 3 and 7 steps are used. It shows no clear pattern in total CO_2 use for the longer forecast horizons in both training and test cases. TD3 shows an increase in CO_2 use in the training case and a similar use in the test case when a horizon of 3 steps is used compared to no horizon. Use for a 7-step horizon decreases to levels similar for the SAC horizons of 11 steps or longer. CO_2 use for the longer forecasts remains similar but with some variation and no clear pattern exists. SAC reports smaller between-seed variances than TD3, but these are still relatively large in magnitude (for example in the test case for $N_p = 15$, the standard deviation of 0.06 on a value of 1.54 equals almost 4 percent). The learning curves show that using forecasts has additional benefits as it helps stabilize the learning procedure and reduces sensitivity to seeds used.

Ventilation use:
Total ventilation use for the final policies are given in Table 9 and learning curves are shown in Fig. 7. SAC ventilation use in both the train and test case reduces when increasing the forecast horizon to 7 steps, it then plateaus and remains stable for the longer horizons. A similar trend is observed for TD3,

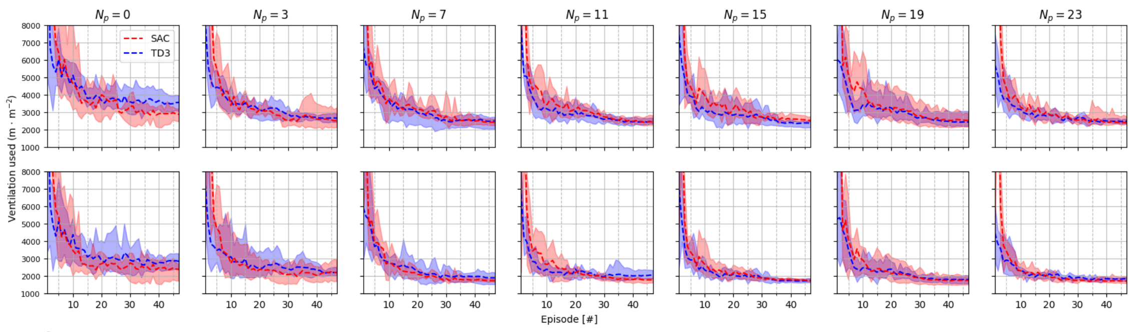


Figure 7
Learning curves describing total ventilation use per episode on training weather data set (upper row) and test weather data (lower row) using the greedy policy for SAC and TD3. The initial random action episodes are not included. The dotted line represents mean value per evaluation episode, outer lines represent minimum and maximum observed values across the five seeds.

Table 9
Total ventilation used in $m \cdot m^{-2}$ for the final learned policies by SAC and TD3 on the fixed weather data applied to training and testing datasets over five seeds for various weather forecast horizons. Mean and (standard deviation) values are reported.

	0	3	7	11	15	19	23
SAC							
Train	2882 (266)	2520 (392)	2417 (148)	2446 (211)	2530 (113)	2534 (272)	2399 (180)
Test	2375 (346)	2215 (475)	1728 (55)	1774 (103)	1740 (55)	1746 (171)	1708 (121)
TD3							
Train	3565 (316)	2660 (157)	2494 (237)	2460 (137)	2378 (130)	2435 (316)	2479 (93)
Test	2843 (341)	2195 (170)	1895 (235)	2039 (179)	1729 (88)	1794 (148)	1829 (66)

but the total ventilation use plateau is higher in the test case and is less stable. Variations between seeds vary for both algorithms, indicating that ventilation control is hard to learn. The learning curves show similar behaviour to those for CO_2 use in that it helps stabilize learning and reduce variance between seeds.

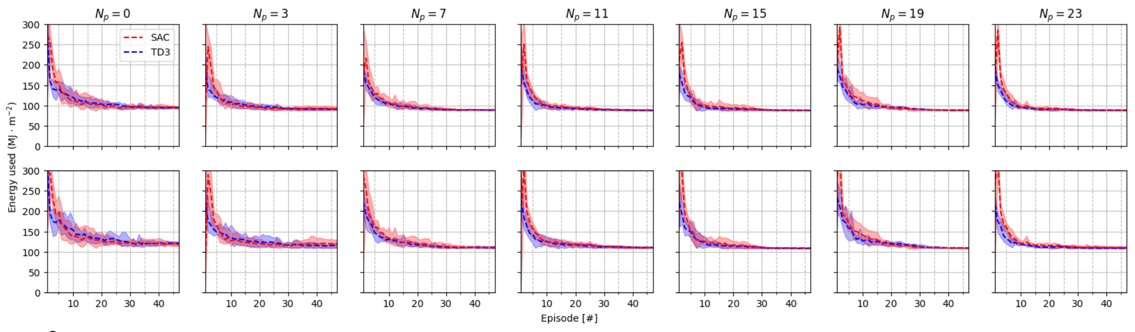


Figure 8
Learning curves describing total energy use per episode on training weather data set (upper row) and test weather data (lower row) using the greedy policy for SAC and TD3. The initial random action episodes are not included. The dotted line represents mean value per evaluation episode, outer lines represent minimum and maximum observed values across the five seeds.

Energy use:
Total energy use for the final policies are given in Table 10 and learning curves are shown in Fig. 8. For reference, 100 MJ translates to a cost of €3.55. The effect of forecast length on energy use is very limited. SAC results for a 3-step horizon and no horizon are similar in both the train and test case. Extending

Table 10
Energy usage in $MJ \cdot m^{-2}$ for the final learned policies by SAC and TD3 on the fixed weather data applied to training and testing datasets over five seeds for various weather forecast horizons. Mean and (standard deviation) values are reported.

	0	3	7	11	15	19	23
SAC							
Train	94.74 (1.83)	92.00 (2.86)	89.33 (0.43)	88.66 (0.82)	88.25 (0.33)	88.63 (0.65)	88.36 (0.56)
Test	118.83 (3.80)	119.62 (7.49)	110.88 (0.99)	110.33 (1.45)	109.26 (0.55)	109.57 (0.75)	110.38 (1.59)
TD3							
Train	94.38 (1.41)	90.93 (1.31)	88.44 (0.74)	87.86 (0.44)	87.69 (0.31)	88.32 (1.21)	88.20 (0.97)
Test	120.47 (2.44)	115.16 (3.82)	110.00 (1.83)	110.48 (0.62)	108.27 (0.38)	109.55 (1.18)	109.10 (1.11)

the forecast to 7 steps leads to the largest reduction in energy use in the test case for SAC, but total energy use in the train case lowers marginally. A plateau is reached and further extension of the forecast horizon does not affect energy use for SAC on either weather sequence. TD3 energy usage across the weather sequences and forecast horizons is very similar to SAC. The learning curves show that use of a small forecast horizon helps stabilize learning early on compared to no forecast and that overall learning of heating control goes smoothly.

What is the effect of the prediction horizon on climate constraint violations?
All final indoor climate constraint violations are reported in Table 11. Reported values represent the summed total boundary violations across all discrete time steps in the fourty day episode in their respective units.[21] The learning curves can be found in the appendix.

CO_2:
Adding a 3-step forecast halves the total CO_2 boundary violations compared to no forecast for the SAC algorithm in the test weather case. TD3 shows most CO_2 constraint violations in both the train and the test case when no forecast is used, as well as when forecasts of 3, 19 and 23 steps are used. No further clear patterns are visible for SAC and TD3. It is, however, important to keep in mind that these values represent the summed violations across the fourty-day time-series and the small values of violations by both algorithms are essentially trivial as long as they are spread out over the time-series and not the results of a single event.

Lower temperature boundary:
SAC lower temperature boundary violations plateau in both the training at around 15 °C and test case at around 245 °C once a forecast horizon of 11 steps is used with limited gain from longer forecasts. TD3 shows a reduction in violations when adding a 3-step horizon compared to the case with no horizon. No clear pattern in violations for longer horizons, with total violations varying between 30 and 67 °C and 65 and 110 °C in the train and test cases, respectively. Additionally, TD3 shows larger variations between seeds for horizons of 11 steps and greater compared to SAC.

Upper temperature boundary:
Upper boundary violations are tricky, in the sense that they are not always avoidable as no active cooling is present in the greenhouse system and heating by outside temperature and the sun can lead to boundary violations. SAC reports total upper temperature boundary violations of 141.71 °C and 316.68 °C when no horizon is used for the train and test set, respectively. This lowers to an optimum of 110.30 °C and 286.93 °C at a horizon of 11 steps. Results for 15 steps are essentially the same, but violations increase again with increasing horizon length when larger horizons than 15 steps are used. TD3 reports total upper temperature boundary violations of 102.88 °C and 282.06 °C when no horizon is used for the train and test set, respectively. The best upper temperature maintenance for TD3 is only marginally better and is found when a horizon of 3 steps is used: 102.84 °C and 273.97 °C for the train and test case, respectively. Performance is marginally worse when horizons of 7 and 11 steps are used, after which test set performances degrade again to levels worse than when no horizon is used.

Overall TD3 reports lower upper temperature violation values and smaller between-seed variations than SAC and is better and more robust at maintaining upper temperature boundaries than SAC.

[21] Values are taken and summed at the 15 minute time intervals and are not integrated. In other words, if the upper temperature boundary is violated by 2°C in two consecutive time steps, this results in an increase of 4 °C in the total episodic constraint violation value, rather than the integrated violation between the two time steps that is adjusted for the time steps of 15 minutes.

Table 11
Total amount of boundary constraint violations in their respective unit of the final learned policies by SAC and TD3 applied to training and testing datasets over five seeds for various weather forecast horizons. Mean and (standard deviation) values are reported.

		0	3	7	11	15	19	23
SAC								
CO_2	Train	0.70	1.11	0.73	0.42	0.53	0.42	0.29
		(0.49)	(0.50)	(0.74)	(0.29)	(0.48)	(0.16)	(0.19)
	Test	1.93	0.93	0.30	0.80	0.95	0.60	0.86
		(1.91)	(0.67)	(0.27)	(0.26)	(0.69)	(0.65)	(0.86)
T_{min}	Train	64.80	46.58	21.84	14.18	13.19	16.63	16.85
		(24.47)	(28.13)	(9.05)	(8.52)	(5.59)	(8.22)	(9.88)
	Test	91.94	94.28	54.84	24.73	21.54	23.29	27.24
		(52.64)	(81.61)	(26.41)	(15.18)	(4.36)	(15.04)	(9.24)
T_{max}	Train	141.71	121.54	116.02	110.30	105.84	113.92	113.82
		(18.39)	(12.89)	(11.32)	(7.08)	(9.31)	(13.10)	(4.67)
	Test	316.68	306.23	292.45	286.93	286.99	297.79	304.38
		(39.81)	(21.84)	(12.68)	(10.94)	(13.94)	(13.17)	(11.29)
RH	Train	1.79	0.70	0.50	0.39	0.00	0.22	0.84
		(2.94)	(0.94)	(0.83)	(0.55)	(0.00)	(0.27)	(1.02)
	Test	4.21	2.58	3.38	2.35	2.25	3.22	7.30
		(7.10)	(3.13)	(2.04)	(2.80)	(2.07)	(2.99)	(4.01)
TD3								
CO_2	Train	1.04	0.87	0.51	0.59	0.75	0.85	1.37
		(0.90)	(0.67)	(0.34)	(0.35)	(0.45)	(0.62)	(0.86)
	Test	2.37	1.79	0.42	0.97	0.89	2.11	2.86
		(1.33)	(1.61)	(0.17)	(0.99)	(0.49)	(2.54)	(1.38)
T_{min}	Train	100.84	43.81	41.04	42.73	30.91	62.75	66.54
		(11.52)	(6.76)	(9.07)	(18.68)	(9.70)	(49.27)	(66.20)
	Test	162.78	74.28	84.39	80.54	65.39	93.32	109.17
		(81.37)	(33.58)	(18.50)	(51.36)	(12.89)	(64.63)	(46.88)
T_{max}	Train	102.88	102.84	111.09	109.83	107.33	102.28	107.73
		(12.27)	(9.07)	(9.56)	(7.98)	(7.53)	(4.58)	(9.03)
	Test	282.06	273.97	282.28	279.01	290.31	290.31	297.30
		(20.24)	(8.39)	(9.89)	(9.77)	(13.33)	(10.57)	(8.33)
RH	Train	3.65	1.75	0.02	1.14	0.47	0.17	1.25
		(3.72)	(2.43)	(0.03)	(1.17)	(0.94)	(0.22)	(1.43)
	Test	6.23	4.61	1.40	1.84	4.91	3.34	5.43
		(4.71)	(3.34)	(0.96)	(1.90)	(3.70)	(1.88)	(5.05)

Additionally, unlike SAC, it manages to do so when no horizon or a very small horizon is provided.

Relative humidity:

Both algorithms adhere to the relative humidity boundary well, both in training and test cases. Total summed violations decrease from 1.79% and 4.21% when no horizon is used to an optimum of 0.00% and 2.25% for the train and test cases respectively when a horizon of 15 steps is used for SAC. Larger horizons lead to performance degradation again, with test performance for a horizon of 23 steps showing more violations than when no horizon is used.

TD3 starts with summed violations of 3.65% and 6.23% when no horizon is used. For this algorithm no clear pattern exists between prediction horizon and maintaining relative humidity conditions within boundaries in the train and test data cases. Regardless and as mentioned before, these are summed boundary violations across fourty days, which means that the very small values reported here are trivial and boundaries are maintained well.

4.4.3 Time-series

A visualization of time-series for indoor measurements, the disturbances, control inputs and the reward function is given in Fig. 9. Climate constraints are maintained well, with some minor exceptions for indoor CO_2 as well as the upper temperature constraint on warmer days near the end of the episode. Relative humidity generally is around 80 to 85% during the night, with lower values during the day due to increased temperatures. Clear diurnal cycles are visible in CO_2 and heating usage: CO_2 is injected during the day and additional heating is used during the night in order to satisfy the lower temperature constraint. Day-time additional heating only occurs on days where heating due to the sun is not enough to satisfy the minimum boundary constraint and no more heating is used than is necessary to maintain the minimum required temperature. Other greenhouse and reinforcement learning works (Wang et al. (2020); Zhang et al. (2021); An et al. (2021); Cao et al. (2022)) make an additional distinction between a daytime and nighttime energy price with the latter being lower as was also used during the Autonomous Greenhouse Challenges (Hemming et al. (2019, 2020)). We experimented with various higher and lower static energy prices and found total energy use and temperature profiles to be consistent regardless of price indicating that (1) daytime temperature profiles are mostly dependent on external disturbances and (2) use of dynamic energy prices would not make a difference, given these climate constraints and this lettuce-greenhouse-model. Ventilation is mostly used during the afternoon to dehumidify or to cool the greenhouse, or both. Ventilation use during the last six days is much increased compared to earlier days in order to compensate for increased external heating. While ventilation at warmer days leads to increased CO_2 losses to the outside world, the agent still considers it beneficial to maximize CO_2 injection in to the system. This is clearly visible during the last days where indoor CO_2 values drop rapidly once ventilation is used during the afternoon but no reduction in CO_2 injection occurs.

The reward profile shows small negative rewards during the nights due to heating requirements and positive rewards during the day due to crop growth, as expected. The reward profile shows a daytime trend similar to the incoming radiation trend, except for the warmer days where upper temperature constraints are violated or large CO_2 losses due to ventilation lead to reduced crop growth, or both.

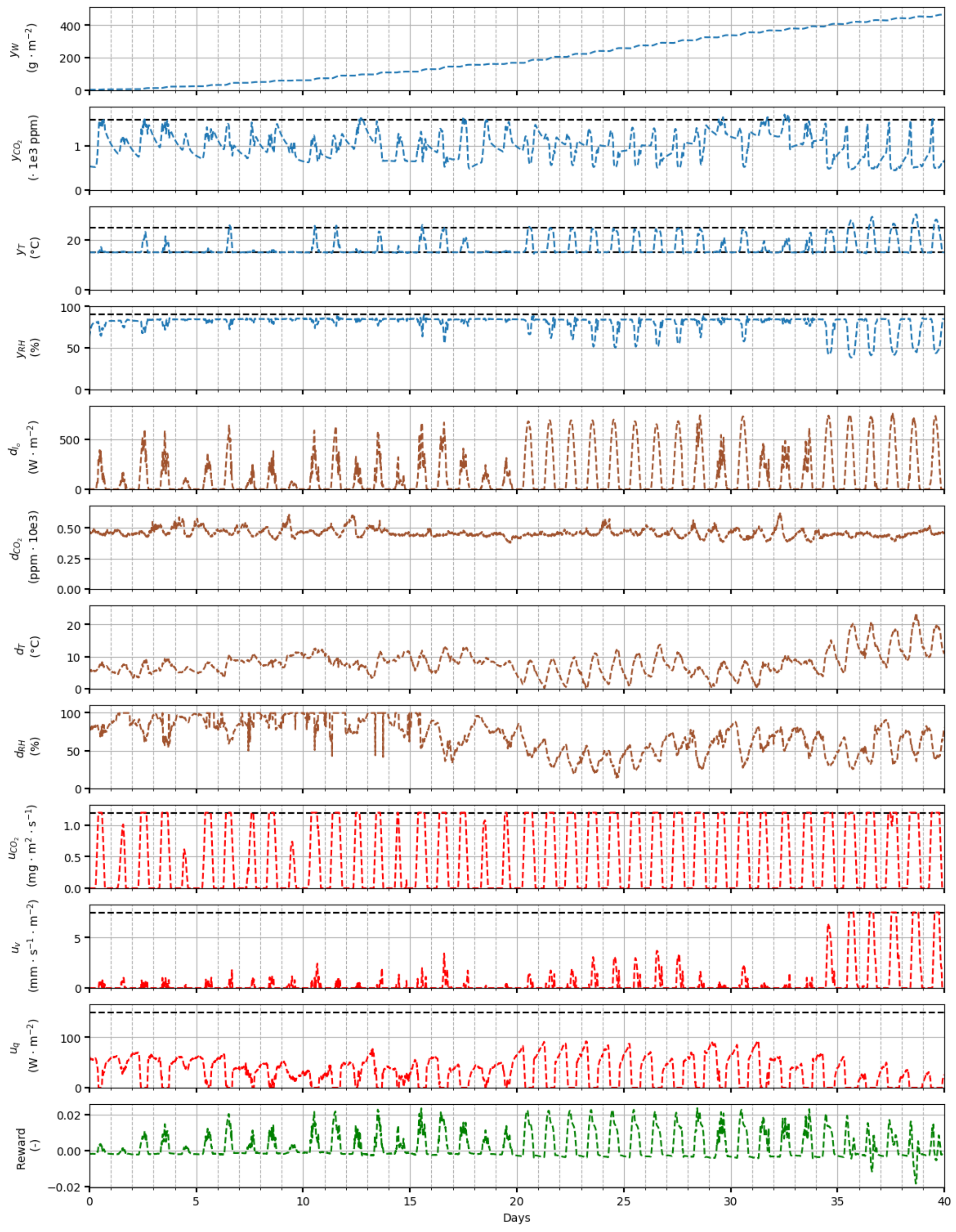


Figure 9
Time-series of indoor conditions, external disturbances, the policy rollout and the reward function found for a single seed and with a prediction horizon of 15 steps using SAC. Measurement variables are represented with blue lines, disturbances by brown lines, control inputs with red lines and the overall reward with a green line. Constraints are give with a dashed black line when relevant.

5 Discussion & Conclusion

We introduced the use of Reinforcement Learning algorithms for optimal control of greenhouse climate conditions in a greenhouse with lettuce. The state-of-the-art continuous control algorithms, SAC and TD3, were used and demonstrated that they can learn sensible control strategies using limited weather forecasts and in limited amounts of time, although at high CO_2 losses to the outside world on warmer days because of ventilation. Policies trained on just a single fixed fourty day weather data trajectory already generalize well to new weather data without any further training.

We found that inclusion of weather forecast in the agent observation space provides very clear benefits compared to the case without a forecast, both from a RL and an application point-of-view. The former will be discussed first, the latter follows. The work is concluded with a brief discussion of the limitations and suggestions for future work.

5.1 RL perspective

We found the generalization error to be largest when no forecast or a 3-step forecast were used. When the control agent has access to no or very limited forecast information, it increases the risk of overfitting to the specific training data due to the use of a static weather sequence. The high expressive power of neural networks used in Deep Reinforcement Learning agents means that they are at risk of overfitting to their training environment (Zhao et al. (2019); Song et al. (2019)). For example, information on crop dry weight or time-awareness, or both, can be misused by the control agent during training to differentiate between similar weather conditions in the training data for its Q-function and policy estimators. While this is beneficial for training performance, it does result in poor generalization performance.

Increases in test performance were larger than increases in training performance with increasing forecast horizon lengths. Furthermore, it also resulted in lower variance between the various seeds. This highlights the importance of including weather forecasts, not only for overall performance on a static weather dataset, but also in algorithm generalization performance. [22] The importance of forecast data in the observation for generalization purposes makes sense, as future rewards are heavily dependent on future weather. The agent needs access to information on the weather forecast in order to adequately estimate future rewards for actions and to decide on whether for example preemptive actions are necessary to avoid constraint violations, or what the best time is to ventilate the greenhouse for dehumidification while minimizing energy and CO_2 losses, or alternatively whether injecting additional CO_2 in to the greenhouse is beneficial.

Optimal performance is obtained with a forecast of 15 steps for both algorithms. A degradation in generalization performance while train performance remains similar is visible for SAC at $N_p = 23$ compared to $N_p = 19$, indicating that the agent is now overfitting. This is mostly likely the result of the used discount factor of 0.80, where rewards at 19 and 23 steps in the future are assigned 1.4% and 0.6% of their actual reward value in the Q-value, respectively. This means that - when adding forecast knowledge in the state observation - it provides little information for and is barely used in estimating the Q-function. In addition, it increases the dimensionality of the observation space and may clutter it with redundant information or noisy patterns. This increases the learning complexity, affects the learning quality and increases the risk of overfitting as a static dataset is used for training.

A direct comparison with other works is not feasible as controller performance is highly dependent on greenhouse location, type and available sensors and actuators, as well as crop species, weather conditions, used simulation settings (van Beveren et al. (2015a); Knibbe et al. (2022)) and the used reward or objective function. However, a single static weather dataset to train and evaluate on is also used in previous works (Wang et al. (2020); An et al. (2021); Cao et al. (2022)) and additional crop information was included in the observation space, algorithm hyperparameters were unfortunately not reported. While testing on training data already likely leads to overoptimistic results, we show that this in fact can be quite the difference. The actions in Zhang et al. (2021) are daily decisions on hourly indoor climate set-points for the next 24 hours and the observation space consists of the past 24 hours of weather data and indoor conditions at hourly intervals. Use of past weather provides more relevant information for future weather estimation to the agent compared to the other three works as it can implicitly estimate future weather based on the weather of the previous day, but it also runs the risk of overfitting to the static weather dataset.

One downside of the criterion used for hyperparameter tuning of the TD3 algorithm is that we inherently bias the results towards the test data set. Reinforcement Learning traditionally does not make use of a training, test and validation set as it usually 'just' interacts with an environment, but

[22] Ideally a relative improvement compared to a case without forecast is defined, but this is of little value given the high variance in no-forecast experiments.

the dependence on external disturbances in our case means that their use is valuable. While the results in this work should preferably also be evaluated on an additional validation data set for verification purposes, it is highly unlikely that it affects conclusions with regards to algorithm performance and the effect of weather forecast on control policies.

TD3 has more hyperparameters related to exploration and regularization that were not tuned during our work. Additional tuning can lead to improved TD3 performance, but comes at the cost of significantly increased computation times. More information on this is given in the appendix (see subsection 6.5). While this task was fairly simple and obtaining hyperparameters that result in good performance is easier, this becomes more complicated with increased task complexity as the range of suitable hyperparameters becomes smaller. This means that hyperparameter tuning becomes more important to obtain desirable performance (Gu et al. (2016)). Consequently, the better performance, lower sensitivity to stochasticity during the learning procedure and the simplicity of tuning SAC means the latter is the preferred algorithm for training reliable control agents.

On a similar note, it is very likely that additional hyperparameter tuning of, for example, the discount factor and network architecture for each combination of algorithm and weather forecast length separately will lead to improvements in generalization performance and increased learning stability. While properly quantifying effects of each forecast horizon separately remains difficult due to the hyperparameter sensitivity present in DRL, we have demonstrated the importance of using weather forecasts.

5.2 Application perspective

We found using a weather forecast increased net profit and crop yield, while also reducing resource use and with fewer climate constraint violations occurring. SAC reports higher net profits and crop yields on the test data for forecast horizons of 11, 15 and 19 steps. The TD3 experiment with a 23-step forecast reports both the highest net profits and highest total crop yield on the training and test data out of all experiments, while this is not visible in the overall reward values. However, this is also the experiment with the largest amount of CO_2 boundary constraint violations and negative effects of too high CO_2 concentrations are not fully expressed in the model, so more violations of the CO_2 constraint caused the higher observed yields and net profits.

Overall, we found SAC is better at managing lower temperature as well as the CO_2 boundary constraints, but has marginally more upper boundary violations. Relative humidity constraint violations between the algorithms are similar. However, in reality the total violation values are small and are trivial as long as they are not the result of single events. Constraint management is also what makes this a tricky problem to solve as it is not possible to enforce constraints in Reinforcement Learning, so careful tuning of constraint violation parameters is required. Resource use may be suboptimal if constraints are too strict, however if constraints are too lenient the agent may decide to ignore them altogether. It may consider to choose not to ventilate to compensate for high temperatures or relative humidity levels in order to preserve high indoor CO_2 levels, or it may choose to increase indoor CO_2 to much higher levels than the desired constraint, effectively exploiting weaknesses in the reward function.

5.3 Limitations and future work

The use of a static weather set between March 2nd and April 11th 2014 to train on limits this study to a meteorological spring season. Additionally, it does not capture capture seasonal patterns and annual differences in climate conditions. We expect that the overfitting effect found in the SAC experiment with a 23-step forecast may reduce when dynamic weather with more variation in conditions is used. On the other hand, added variation will likely have a negative effect on experiments where no or a very short forecasts are used. This is due to the increased uncertainty and variance in weather trajectory developments beyond their forecast, as this information is required to adequately estimate Q-values. Additionally, it diminishes the ability of the agent to overfit to the training weather data.

We also assumed the weather forecast and indoor measurements to be perfect, which is not be the case in reality. Kuijpers et al. (2022) found negative effects of uncertainties in short-term weather forecast to be very limited at around a 2% reduction in net profit when using a MPC control algorithm for tomato crop growth, but how this affects performances of lettuce and RL algorithms is not researched as of yet. Adding forecast and measurement errors means the agent will have to estimate the true forecast and measurements from the given noisy forecast and measurements. This changes the problem into a Partially Observable Markov Decision Process (POMDP). This also means that past observations become more important to improve estimates of the current state (Duan et al. (2016)), either by the agent itself or via supplementary state estimation techniques (for examples, see van Mourik et al. (2021); Boersma et al. (2022b)). Additionally, information on estimated forecast and measurements errors can be added to the observation space to help the agent. How and how much this negatively affects the learning process and agent behaviour will depend on the shape and size of the noise, as well as how the forecast errors are distributed over time.

Whereas MPC is able to decide control actions based on dynamic climate constraints, as well as dynamic crop and resource pricing and pricing forecasts, this is only feasible for RL if this is also done during training. This increases both the observation space significantly and much increases learning complexity. This is brings us to further improvements, such as to make the agent constraint or price-aware, or both, by including these (static) values in the observation space.[23] This should reduce learning complexity if static values are used, increase and improve learning speed for RL algorithms as well as adaptability to situations with, for example, different lettuce and energy prices than were assumed in this work.

Furthermore, the specific greenhouse and crop model parameters used in the environment were assumed as latent variables and were not given to the control agent as the goal was to learn control policies without access to a model or prior knowledge of the environment. Effects of the outside climate on the indoor climate and the interaction with the crop in the greenhouse vary greatly and while most relevant parameters can be named, their values are greenhouse- and crop-specific. Some are easy to measure, for example the greenhouse height, but they are generally unknown. A suggestion for future work is to combine RL with system identification to improve generalization abilities and transferability to different environments. This can be done by including rough estimates of some of the more important constants in the observation space. Two particularly interesting follow-ups would then be to (1) include rough estimates of some of the more important constants in the observation space and supplement this with online parameter estimation methods to perform online corrections on parameter estimations for samples stored in the replay buffer and (2) to implement domain randomization during the learning procedure. Varying model parameters used during the learning episodes improves agent generalization and helps bridge the 'reality gap' where policies learned using a simulator may not transfer to the real world (Peng et al. (2018)). This results in essentially creating a controller metamodel that can then be used to adapt to new environments using relatively few new (real world) data samples (Finn et al. (2017)). However, this does require that the agent is provided with an observation and action history in their observation space in order for it to infer (hidden) model parameters (Duan et al. (2016)).

[23] A brief additional discussion on the used observation space can be found in the appendix, see subsection 6.6.

References

Achiam, J. (2018). Benchmarks for Spinning Up Implementations — Spinning Up documentation. Accessed on 2022-03-15.

Ajagekar, A. and You, F. (2022). Deep reinforcement learning based automatic control in semi-closed greenhouse systems. *IFAC-PapersOnLine*, 55:406–411.

An, Z., Cao, X., Yao, Y., Zhang, W., Li, L., Wang, Y., Guo, S., and Luo, D. (2021). A Simulator-based Planning Framework for Optimizing Autonomous Greenhouse Control Strategy. *Proceedings of the International Conference on Automated Planning and Scheduling*, 31:436–444.

Andrychowicz, O. A. M., Baker, B., Chociej, M., Józefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., and Zaremba, W. (2018). Learning dexterous in-hand manipulation. *International Journal of Robotics Research*, 39:3–20.

Baar, J. V., Sullivan, A., Cordorel, R., Jha, D., Romeres, D., and Nikovski, D. (2018). Sim-to-real transfer learning using robustified controllers in robotic tasks involving complex dynamics. *Proceedings - IEEE International Conference on Robotics and Automation*, 2019-May:6001–6007.

Babaeizadeh, M., Frosio, I., Tyree, S., Clemons, J., and Kautz, J. (2016). GA3C: gpu-based A3C for deep reinforcement learning. *CoRR*, abs/1611.06256.

Baeza, E., Dijkxhoorn, Y., Logatcheva, K., Hennen, W., Splinter, G., Stanghellini, C., and Hemming, S. (2021). *Business case for large scale crop production in greenhouse facilities in Iceland for the global market*. Number WPR-1049 in Report / Stichting Wageningen Research, Wageningen Plant Research, Business Unit Greenhouse Horticulture. Project number: 3742295600. - Theme: Energy and Climate.

Bakker, J., Bot, G., Challa, H., and van de Braak, N. (1995). *Greenhouse Climate Control. An integrated approach*. Wageningen Pers.

Bellemare, M. G., Candido, S., Castro, P. S., Gong, J., Machado, M. C., Moitra, S., Ponda, S. S., and Wang, Z. (2020). Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature 2020 588:7836*, 588:77–82.

Bellman, R. (1957). *Dynamic Programming*. Dover Publications.

Boersma, S., Sun, C., and van Mourik, S. (2022a). Robust sample-based model predictive control of a greenhouse system with parametric uncertainty. *IFAC-PapersOnLine*, 55:177–182.

Boersma, S., Van Mourik, S., Xin, B., Kootstra, G., and Bustos-Korts, D. (2022b). Nonlinear Observability Analysis and Joint State and Parameter Estimation in a Lettuce Greenhouse using Ensemble Kalman Filtering. *IFAC-PapersOnLine*, 55(32):141–146.

Cao, X., Yao, Y., Li, L., Zhang, W., An, Z., Zhang, Z., Xiao, L., Guo, S., Cao, X., Wu, M., and Luo, D. (2022). igrow: A smart agriculture solution to autonomous greenhouse control.

Challa, H. (1990). *Crop growth models for greenhouse climate control.*, pages 125–145. Simulation monographs. Pudoc.

Chen, L., Du, S., He, Y., Liang, M., and Xu, D. (2018). Robust model predictive control for greenhouse temperature based on particle swarm optimization. *Information processing in agriculture, vol. 5(3), pp. 329-338*.

Choab, N., Allouhi, A., Maakoul, A. E., Kousksou, T., Saadeddine, S., and Jamil, A. (2019). Review on greenhouse microclimate and application: Design parameters, thermal modeling and simulation, climate controlling technologies. *Solar Energy*, 191:109–137.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. (2016). Benchmarking deep reinforcement learning for continuous control. *33rd International Conference on Machine Learning, ICML 2016*, 3:2001–2014.

Dulac-Arnold, G., Mankowitz, D., and Hester, T. (2019). Challenges of real-world reinforcement learning.

FAO, IFAD, UNICEF, WFP, and WHO (2021). *The State of Food Security and Nutrition in the World 2021: Transforming food systems for food security, improved nutrition and affordable healthy diets for all*. FAO, Rome, Italy.

Farquhar, G. D., von Caemmerer, S., and Berry, J. A. (1980). A biochemical model of photosynthetic co2 assimilation in leaves of c3 species. *Planta*, 149:78–90.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks.

Friedlingstein, P., O'sullivan, M., Jones, M. W., Andrew, R. M., Gregor, L., Hauck, J., Quéré, C. L., Luijkx, I. T., Olsen, A., Peters, G. P., Peters, W., Pongratz, J., Schwingshackl, C., Sitch, S., Canadell, J. G., Ciais, P., Jackson, R. B., Alin, S. R., Alkama, R., Arneth, A., Arora, V. K., Bates, N. R., Becker, M., Bellouin, N., Bittig, H. C., Bopp, L., Chevallier, F., Chini, L. P., Cronin, M., Evans, W., Falk, S., Feely, R. A., Gasser, T., Gehlen, M., Gkritzalis, T., Gloege, L., Grassi, G., Gruber, N., Özgür

Gürses, Harris, I., Hefner, M., Houghton, R. A., Hurtt, G. C., Iida, Y., Ilyina, T., Jain, A. K., Jersild, A., Kadono, K., Kato, E., Kennedy, D., Goldewijk, K. K., Knauer, J., Korsbakken, J. I., Landschützer, P., Lefèvre, N., Lindsay, K., Liu, J., Liu, Z., Marland, G., Mayot, N., Mcgrath, M. J., Metzl, N., Monacci, N. M., Munro, D. R., Nakaoka, S. I., Niwa, Y., O'brien, K., Ono, T., Palmer, P. I., Pan, N., Pierrot, D., Pocock, K., Poulter, B., Resplandy, L., Robertson, E., Rödenbeck, C., Rodriguez, C., Rosan, T. M., Schwinger, J., Séférian, R., Shutler, J. D., Skjelvan, I., Steinhoff, T., Sun, Q., Sutton, A. J., Sweeney, C., Takao, S., Tanhua, T., Tans, P. P., Tian, X., Tian, H., Tilbrook, B., Tsujino, H., Tubiello, F., Werf, G. R. V. D., Walker, A. P., Wanninkhof, R., Whitehead, C., Wranne, A. W., Wright, R., Yuan, W., Yue, C., Yue, X., Zaehle, S., Zeng, J., and Zheng, B. (2022). Global carbon budget 2022. *Earth System Science Data*, 14:4811–4900.

Fujimoto, S., van Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. *CoRR*, abs/1802.09477.

Gautron, R., Padrón, E. J., Preux, P., Bigot, J., Maillard, O.-A., and Emukpere, D. (2022). gym-dssat: a crop model turned into a reinforcement learning environment.

Goldberg, K. (2019). Robots and the return to collaborative intelligence. *Nature Machine Intelligence 2019 1:1*, 1:2–4.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Graamans, L., Baeza, E., Van Den Dobbelsteen, A., Tsafaras, I., and Stanghellini, C. (2018). Plant factories versus greenhouses: Comparison of resource use efficiency. *Agricultural Systems*, 160:31–43.

Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., and Levine, S. (2016). Q-Prop: Sample-Efficient Policy Gradient with An Off-Policy Critic. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*.

Haarnoja, T., Ha, S., Zhou, A., Tan, J., Tucker, G., and Levine, S. (2018a). Learning to walk via deep reinforcement learning.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018b). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. (2018c). Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905.

Hassabis, D. (2017). Artificial intelligence: Chess match of the century. *Nature 2017 544:7651*, 544:413–414.

Hemming, S., de Zwart, F., Elings, A., Righini, I., and Petropoulou, A. (2019). Remote Control of Greenhouse Vegetable Production with Artificial Intelligence—Greenhouse Climate, Irrigation, and Crop Production. *Sensors*, 19(8):1807.

Hemming, S., Zwart, F. d., Elings, A., Petropoulou, A., and Righini, I. (2020). Cherry Tomato Production in Intelligent Greenhouses—Sensors and AI for Control of Climate, Irrigation, Crop Yield, and Quality. *Sensors*, 20(22):6430.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2019). Deep Reinforcement Learning that Matters. *arXiv:1709.06560 [cs, stat]*. arXiv: 1709.06560.

Kahneman, D. (2003). Maps of bounded rationality: Psychology for behavioral economics. *American Economic Review*, 93(5):1449–1475.

Katzin, D., van Henten, E. J., and van Mourik, S. (2022). Process-based greenhouse climate models: Genealogy, current status, and future directions. *Agricultural Systems*, 198:103388.

Kempkes, F. L. K., Janse, J., and Hemming, S. (2014). Greenhouse concept with high insulating double glass with coatings and new climate control strategies; from design to results from tomato experiments. *ISHS Acta Horticulturae*.

Kingma, D. P. and Ba, J. L. (2014). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

Knibbe, W. J., Afman, L., Boersma, S., Bogaardt, M.-J., Evers, J., van Evert, F., van der Heide, J., Hoving, I., van Mourik, S., de Ridder, D., and de Wit, A. (2022). Digital twins in the green life sciences. *https://doi.org/10.1080/27685241.2022.2150571*, 94:249–279.

Koos, S., Mouret, J.-B., and Doncieux, S. (2013). The transferability approach: Crossing the reality gap in evolutionary robotics. *IEEE Transactions on Evolutionary Computation*, 17(1):122–145.

Kuijpers, W. J., Antunes, D. J., van Mourik, S., van Henten, E. J., and van de Molengraft, M. J. (2022). Weather forecast error modelling and performance analysis of automatic greenhouse climate control. *Biosystems Engineering*, 214:207–229.

Kuijpers, W. J. P., Katzin, D., van Mourik, S., Antunes, D. J., Hemming, S., and van de Molengraft, M. J. G. (2021). Lighting systems and strategies compared in an optimally controlled greenhouse. *Biosystems Engineering, vol. 202, pp. 195-216*.

Kuijpers, W. J. P., van de Molengraft, M. J. G., van Mourik, S. van 't Ooster, A., Hemming, S., and van Henten, E. J. (2019). Model selection with a common structure: Tomatocrop growth models.

Biosystems Engineering, vol. 187, pp. 247-257.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N. M. O., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971.

Lopez-Cruz, I., Fitz-Rodríguez, E., Raquel, S., Rojano-Aguilar, A., and Kacira, M. (2018). Development and analysis of dynamical mathematical models of greenhouse climate: A review. *European Journal of Horticultural Science*, 83:269–279.

Maestrini, B., Mimić, G., van Oort, P. A., Jindo, K., Brdar, S., van Evert, F. K., and Athanasiados, I. (2022). Mixing process-based and data-driven approaches in yield prediction. *European Journal of Agronomy*, 139:126569.

Marcelis, L. F., Broekhuijsen, A. G., Meinen, E., Nijs, E. M., and Raaphorst, M. G. (2006). Quantification of the growth response to light quantity of greenhouse grown crops. *Acta Horticulturae*, 711:97–103.

Marcelis, L. F. M., Costa, J. M., and Heuvelink, E. (2019). Achieving sustainable greenhouse production: present status, recent advances and future developments. pages 1–14.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous Methods for Deep Reinforcement Learning. *arXiv:1602.01783 [cs]*. arXiv: 1602.01783.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. A. (2013). Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Norton, T., Sun, D. W., Grant, J., Fallon, R., and Dodd, V. (2007). Applications of computational fluid dynamics (cfd) in the modelling and design of ventilation systems in the agricultural industry: A review. *Bioresource Technology*, 98:2386–2414.

Ooteghem, R. J. V. (2010). Optimal control design for a solar greenhouse. *IFAC Proceedings Volumes*, 43:304–309.

Overweg, H., Berghuijs, H. N. C., and Athanasiadis, I. N. (2021). Cropgym: a reinforcement learning environment for crop management.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. (2018). Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1986). *NUMERICAL RECIPES Third Edition-Numerical Recipes: The Art of Scientific Computing, Third Edition*.

Raffin, A., Kober, J., and Stulp, F. (2021). Smooth Exploration for Robotic Reinforcement Learning. *arXiv:2005.05719 [cs, stat]*. arXiv: 2005.05719.

Rasheed, A., San, O., and Kvamsdal, T. (2019). Digital twin: Values, challenges and enablers.

Roy, J. C., Boulard, T., Kittas, C., and Wang, S. (2002). Convective and ventilation transfers in greenhouses, part 1: The greenhouse considered as a perfectly stirred tank. *Biosystems Engineering*, 83:1–20.

Saikai, Y., Peake, A., and Chenu, K. (2023). Deep reinforcement learning for irrigation scheduling using high-dimensional sensor feedback.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. (2015). Trust region policy optimization. *CoRR*, abs/1502.05477.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Driessche, G. V. D., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms. *31st International Conference on Machine Learning, ICML 2014*, 1.

Song, X., Jiang, Y., Tu, S., Du, Y., and Neyshabur, B. (2019). Observational overfitting in reinforcement learning.

Sparks, B. (2018). What is the current state of labor in the greenhouse industry? *Greenhouse Grower*. Accessed on 2021-10-27.

Stanghellini, C. (2014). Horticultural production in greenhouses: Efficient use of water. *Acta Horticulturae*, 1034:25–32.

Straten, G. V., Challa, H., and Buwalda, F. (2000). Towards user accepted optimal control of greenhouse climate. *Computers and Electronics in Agriculture*, 26:221–238.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, USA.

Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., Bohez, S., Vanhoucke, V., Brain, G., and Deepmind, G. (2018). Sim-to-real: Learning agile locomotion for quadruped robots.

Turchetta, M., Corinzia, L., Sussex, S., Burton, A., Herrera, J., Athanasiadis, I. N., Buhmann, J. M., and Krause, A. (2022). Learning long-term crop management strategies with cyclesgym. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

United Nations, Department of Economic and Social Affairs, and Population Division (2019). *World urbanization prospects: the 2018 revision*. OCLC: 1120698127.

van Beveren, P. J., Bontsema, J., van Straten, G., and van Henten, E. J. (2015a). Minimal heating and cooling in a modern rose greenhouse. *Applied Energy*, 137:97–109.

van Beveren, P. J., Bontsema, J., van Straten, G., and van Henten, E. J. (2015b). Optimal control of greenhouse climate using minimal energy and grower defined bounds. *Applied Energy*, 159:509–519.

van Henten, E. J. (1994a). *Greenhouse climate management: an optimal control approach*. PhD thesis, University Wageningen.

van Henten, E. J. (1994b). Validation of a dynamic lettuce growth model for greenhouse climate control. *Agricultural Systems*, 45(1):55–72.

van Henten, E. J. (2003). Sensitivity analysis of an optimal control problem in greenhouse climate management. *Biosystems Engineering, vol. 85(3), pp. 355–364*.

van Henten, E. J. and Bontsema, J. (2009). Time-scale decomposition of an optimal control problem in greenhouse climate management. *Control Engineering Practice*, 17:88–96.

van Mourik, S., van der Tol, R., Linker, R., Reyes-Lastiri, D., Kootstra, G., Groot Koerkamp, P., and van Henten, E. J. (2021). Introductory overview: Systems and control methods for operational management support in agricultural production systems. *Environmental Modelling & Software, vol. 139, pp. 105031*.

van Rijswick, C. (2018). World vegetable map 2018 more than just a local affair. Accessed on 2022-03-20.

van Straten, G. and van Henten, E. J. (2010). Optimal greenhouse cultivation control: survey and perspectives. *IFAC conference*.

Verdouw, C., Tekinerdogan, B., Beulens, A., and Wolfert, S. (2021). Digital twins in smart farming. *Agricultural Systems*, 189.

Waldrop, M. M. (2015). Autonomous vehicles: No drivers required. *Nature*, 518:20–23.

Wang, L., He, X., and Luo, D. (2020). Deep reinforcement learning for greenhouse climate control.

Xu, D. amd Du, S. and van Willegenburg, G. (2018). Adaptive two time-scale receding horizon optimal control for greenhouselettuce cultivation. *Computers and Electronics in Agriculture, vol. 146(3), pp. 93-103*.

Zhang, L., Xu, Z., Xu, D., Ma, J., Chen, Y., and Fu, Z. (2020). Growth monitoring of greenhouse lettuce based on a convolutional neural network. *Horticulture Research 2020 7:1*, 7:1–12.

Zhang, W., Yao, Y., Xiao, X., Balasubramanian, V. N., and Tsang, I. (2021). Robust model-based reinforcement learning for autonomous greenhouse control. *Proceedings of Machine Learning Research*, 157:2021–2021.

Zhao, C., Sigaud, O., Stulp, F., and Hospedales, T. M. (2019). Investigating generalisation in continuous deep reinforcement learning.

6 Appendix

6.1 Digital Twins in Greenhouse Horticulture

The goal is to have digital twins that give real-time information on the indoor climate and crop status and perform, amongst others, climate and crop control automatically, but also includes harvesting via robots and the use of drones for monitoring for example (Verdouw et al. (2021); Knibbe et al. (2022)). This requires precise tools to estimate and perform online updates on the relevant greenhouse and crop system parameters in order to develop an accurate model, correction of measurement errors and estimation of not directly measurable states from the measurements data, and for prediction of future developments, as well as algorithms that can optimize control actions within the greenhouse.

6.1.1 Greenhouse and crop modelling approaches

Greenhouse climate dynamics are modelled in a variety of ways and the level of complexity and detail depends on the use-case (for example describing, understanding or control of the climate). Computation Fluid Dynamics (CFD) are used to model the indoor climate profile using partial different equations and are mostly used to investigate spatio-temporal dynamics (Norton et al. (2007); Choab et al. (2019)) as temperatures within a greenhouse can fluctuate by 1.5 - 2 °C, this variance occurs even at similar heights (van Beveren et al. (2015a)). These however, are not very suitable for optimal control applications as these models are too computationally expensive and slow, while often also being intractable for mathematical solvers used in model-based control methods. Instead simplified climate models are used that assume a perfectly stirred tank, i.e., it is assumed that climate conditions are homogeneous within the greenhouse, a greenhouse is assumed to be infinitely large and wall-effects are neglected (Roy et al. (2002)). These generally consist of an energy balance for temperature and, if they are included in the model, a mass balance(s) for CO_2 or the humidity, or both (Lopez-Cruz et al. (2018); Katzin et al. (2022)). In some cases these models function as an emulator for a more complex model, but they also exist as standalone models (Knibbe et al. (2022)). These models trade some model accuracy for significant increases in computation speed, making the use of solvers feasible and have been found to perform well for control applications. Some first-principles-based methods to describe the climate dynamics are mechanistic process models described by ordinary differential equations or difference equations. Alternatively data-driven machine learning methods have also been used to develop surrogate models of the dynamics (Maestrini et al. (2022)). Where mechanistic models are considered to generalize better to other situations (i.e., a wide variety of greenhouse environments), data-driven machine learning methods reduce computation time and can perform better in the specific case their data originates from (Rasheed et al. (2019)).

Similarly, a wide variety of crop models exist. In one approach the crop biomass (dry weight of the crops) is aggregated in to one single state and this is then used to model overall behaviour of the crop given environmental conditions (Roy et al. (2002)), which works best for simpler crops such as lettuce. More complex crops such as tomatoes or cucumber require a lot more information to describe (such as a number of leaves, trusses and fruits, for example) and have clearly distinguished development stages. More detail is at times included by describing for example the total leaf biomass and total crop biomass per plant: each leaf and fruit of a plant can be modelled separately, which is called the ”small leaf, small fruit” approach, or alternatively leaf mass and fruit mass can be aggregated per plant, which is referred to as the ”big leaf, big fruit” approach (Kuijpers et al. (2019)).

6.1.2 Suitability for RL

While the use of high-fidelity models of the greenhouse indoor climate and for the (separate) crops to represent the real greenhouse environment is desirable in the long-term, it is not practical for finding optimal control policies due to two main drawbacks:

- Computational complexity
- Very high dimension of state-space

The first point leads to very long computational times. The second point makes optimization problems much more difficult. For model-based control algorithms such as model predictive control and model-based RL algorithms it may mean that they are not able to find tractable solutions or that they may not be able to find solutions in time (and thus be unable to meet real-time requirements and constraints) due to the size of the solution space (Knibbe et al. (2022); Dulac-Arnold et al. (2019)). For model-free RL algorithms both drawbacks make the learning procedure much more difficult and time-consuming.

As a result, the decision is made in optimal control to make use a low-dimensional models. These models still capture the most fundamental features of the problem, while vastly reducing problem complexity and computation times, thus simplifying the optimization problem in the process (Rasheed et al. (2019)). Policies found by training on the simple model can subsequently be adjusted for and applied to a higher-fidelity model in order to validate them. If results are not satisfying, the simplified model needs to be updated and a new controller has to be trained. This is an iterative cycle that ends once acceptable results are obtained in the high-fidelity environment (Knibbe et al. (2022)). This is very much a modelling problem with the aim of finding the optimal model complexity for capturing the relevant processes while still being simple enough to allow for optimization algorithms to work. While important to touch on briefly, this is beyond the scope of this work.

As the focus in this work is on the effect of weather forecasts on optimal control of the greenhouse climate and crop, the decision was made to go a simple crop in lettuce. As real-life data is scarce and expensive, the decision was made to make use of mechanistic models for the agents to interact with. Generating synthetic data is frequently done in domains where real world data is scarce, expensive to obtain or when training in the real world raises safety concerns or takes too long due to physical limitations, for example in robotics (Koos et al. (2013); Peng et al. (2018); Tan et al. (2018); Andrychowicz et al. (2018); Baar et al. (2018)). A simple lettuce crop model was used, where crop biomass is described by a single state. Similarly, a greenhouse model that assumes the indoor climate conditions to be homogeneous is used. As a result, the decision was made to use the greenhouse with lettuce model by van Henten (1994b, 2003) that is elaborated on in section 2.

Furthermore, previous works where Reinforcement Learning is applied to greenhouse optimal control focused on learning policies based on external datasets consisting of static trajectories for indoor crop status, climate conditions and actuator inputs as well as outdoor climate conditions and without further interaction with an environment (Wang et al. (2020); Zhang et al. (2021); An et al. (2021); Cao et al. (2022)). In our work the decision is made to learn from scratch instead, using only the simulation environment. This allows the agent more flexibility with regards to exploration of the state-space and avoids implementing a bias towards the experienced control trajectories when using a static, offline data set.

6.1.3 Types of decision support

Automated decision support in greenhouse horticulture can be provided in three different ways (van Mourik et al. (2021)). These can be translated in to the Markov Decision Process (MDP) format that is used in RL, and are as follows:

- Monitoring: State estimation based on sensor measurements or images, or both, as well as online model parameter correction or calibration if that information is used by the control agent. The aim is to minimize the overall state estimation error. This includes measurement errors due to sensor noise and estimations of states that are not directly measurable using available sensors (Duan et al. (2016)). This requires use of state estimation methods, such as Kalman-Filtering (for examples, see van Mourik et al. (2021); Boersma et al. (2022b)). Furthermore, crop state estimation can be aided by computer vision algorithms (see Zhang et al. (2020) for an example).
- Prediction: This consists of the prediction of future rewards by estimating the value- and action-value-functions (Sutton and Barto (2018)), as well as system identification in the form of the estimation of transition functions $\mathbb{P}$ for indoor climate and crop dynamics in order to predict trajectory developments (Duan et al. (2016); Sutton and Barto (2018); Lopez-Cruz et al. (2018); Knibbe et al. (2022)). The latter also requires online calibration of the environment model parameters using a feedback loop with the system as some of the parameters that are assumed constant in models are in reality dynamic over a crop cycle, thus requiring online corrections (Katzin et al. (2022)). Such a model can be, for example, a mathematical process model or a surrogate model using neural networks. Additionally, this also includes online correction of weather forecasts based on observed weather (for example, see Kuijpers et al. (2022)).
- Control: Policy π optimization that maps state observations s to a set of control actions a and does so in a way that optimizes performance for a given objective function (Sutton and Barto (2018)).

Our work focuses on learning control policies using model-free RL. State and parameter estimation and calibration, as well as learning of the transition dynamics are beyond the scope of this work, but do provide an interesting basis for future work.

6.2 Algorithm choice

Reinforcement Learning algorithms that use discrete state- or action-spaces are not feasible. With discrete state-spaces the Markov property is no longer satisfied as the guarantee that transition (s, a) leads to state s′ is lost. Discretizing the action-space for multiple actuators in the system leads to a rapidly increasing number of combinations that have to be visited during the learning process (curse of dimensionality). With developments in deep learning in the early 2010s, Deep Q-Network (DQN) was developed in 2013 that can learn discrete policies from raw and high-dimensional (image) data using neural networks as function approximators and without the use of handcrafted features as was the norm at the time (Mnih et al. (2013, 2015)). Because precise, continuous control of the actuators is desirable and to limit negative effects of the curse of dimensionality, Deep Reinforcement Learning (DRL) algorithms with neural networks as function approximators and continuous action- and state-spaces will be used.

6.2.1 Deep Reinforcement Learning Algorithms

Actor-critic algorithms are most suitable RL algorithms to tackle continuous control tasks. They consist of an actor that learns a policy and takes the actions in the environment, and a critic that learns the value-function and gives feedback to the actor.

In the last decade research on actor-critics has progressed by essentially extending DQN - that only works for discrete actions (Mnih et al. (2013, 2015)) - with Deterministic Policy Gradient (DPG), a policy-based method that can handle continuous action-spaces (Silver et al. (2014)). This resulted in the Deep Deterministic Policy Gradient algorithm (DDPG) that also makes use of the important experience replay buffer introduced in DQN for offline learning and is able to handle continuous action-spaces (Lillicrap et al. (2016)). Learning policies in greenhouse control is a complex task and training times are expected to be long. As a result the increased sampling- and computational-efficiency of offline learning actor-critic algorithms is preferred over policy-based as well as other actor-critic algorithms that learn on-policy and online, such as the Trust Region Policy Optimization (TRPO) (Schulman et al. (2015)), Advantage actor-crtic (A2C) (Mnih et al. (2016)), Asynchronous Advantage actor-Critic (A3C) (Babaeizadeh et al. (2016)) and Proximal Policy Optimization Algorithms (PPO) (Schulman et al. (2017)).

Both Twin-Delayed Deep Deterministic Policy Gradient (TD3) (Fujimoto et al. (2018)) and Soft-Actor Critic (SAC) (Haarnoja et al. (2018b,c)) build on DDPG and are considered state-of-the-art for continuous control problems, though the latter was found to perform slightly better in four of five control tasks (Achiam (2018)). The main difference between SAC and TD3 being the fact that the former makes use of a stochastic actor during training as well as an additional entropy term in the reward term to make use of the maximum entropy principle, both of which help improve exploration and robustness as well as reducing variance. It does mean that it not only tries to maximize future reward, but a combination of entropy and future reward instead. TD3 chooses deterministic actions greedily instead and requires the use of added action noise during the learning process for the sake of exploring.

6.3 Weather time-series

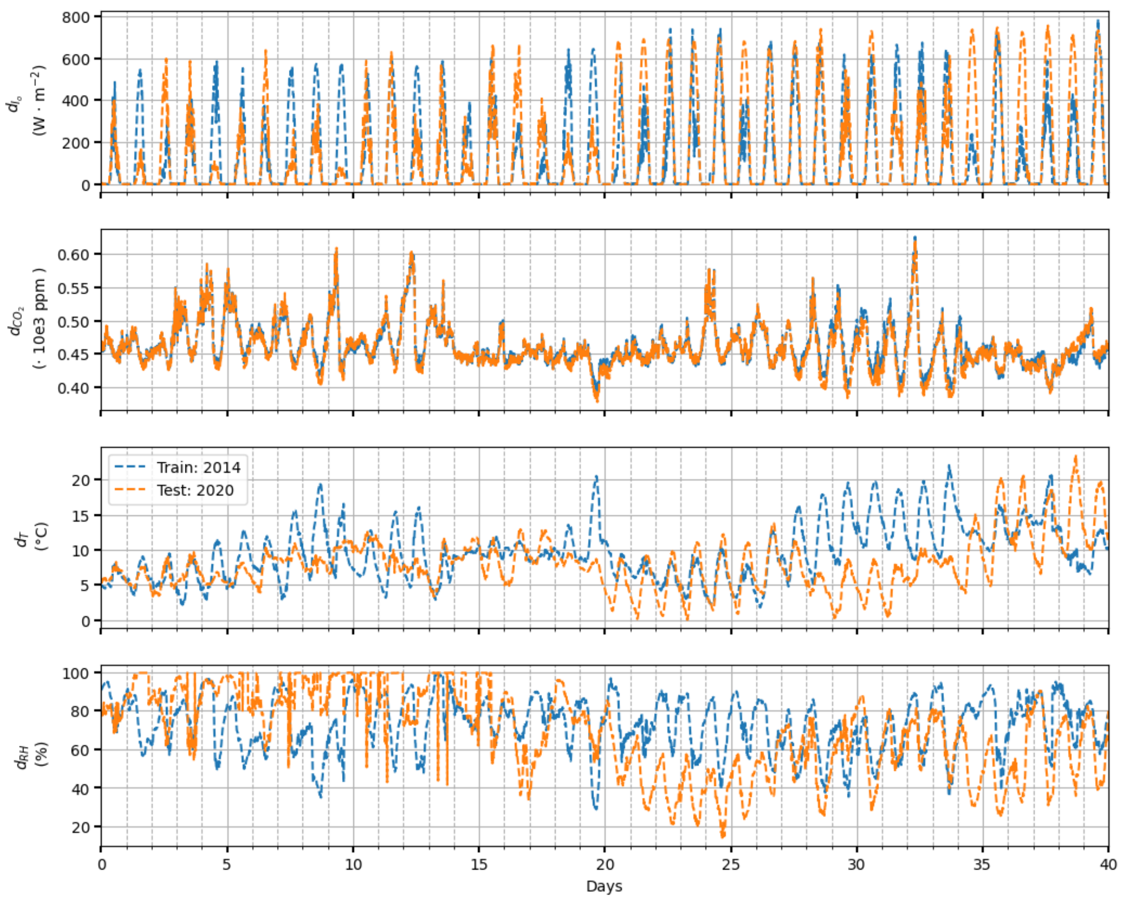


Figure 10
Time-series of train and test weather data used. The blue and orange dashed line represents the data used for training and testing, respectively.

6.4 Learning curves

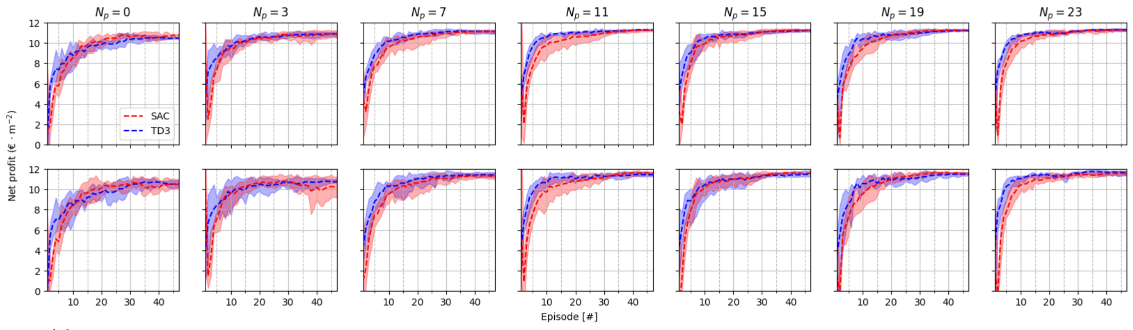


Figure 11
Learning curves describing total net profit in euros · m^{-2} on the training weather data set (upper row) and test weather data (lower row) using the greedy policy for SAC and TD3. The initial random action episodes are not included. The dotted line represents mean value per evaluation episode, outer lines represent minimum and maximum observed values across the five seeds.

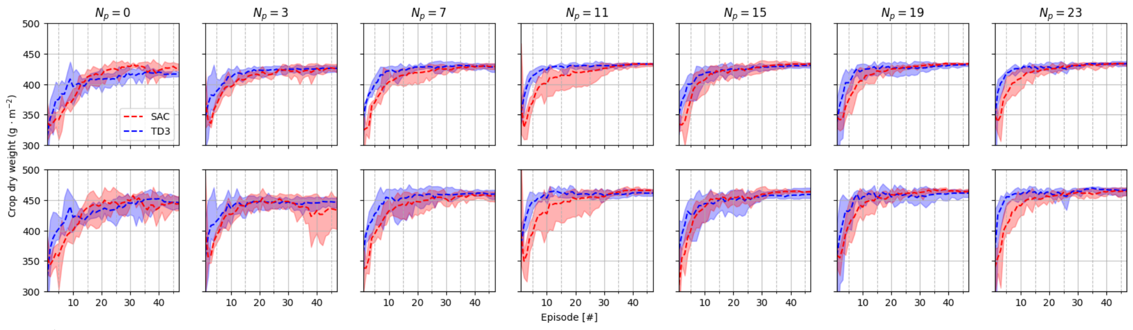


Figure 12
Learning curves describing the final crop yield in g · m^{-2} on the training weather data set (upper row) and test weather data (lower row) using the greedy policy for SAC and TD3. The initial random action episodes are not included. The dotted line represents mean value per evaluation episode, outer lines represent minimum and maximum observed values across the five seeds.

6.4.1 Constraint violations

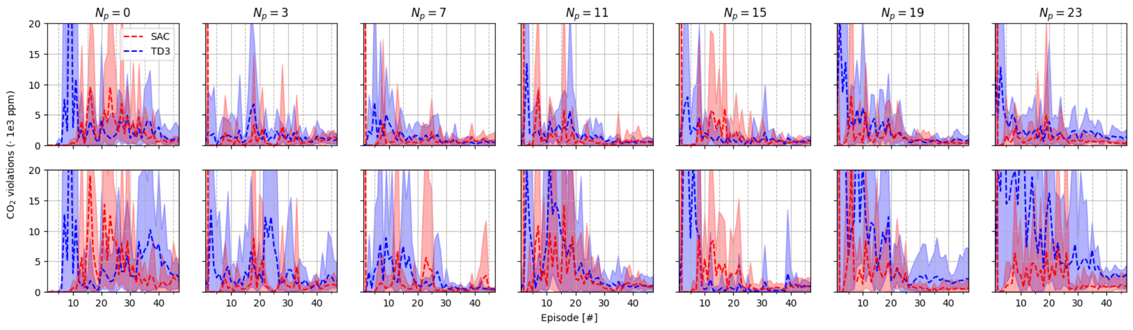


Figure 13
Learning curves describing the summed total CO_2 violations in ppm · $10e^3$ on training weather data set (upper row) and test weather data (lower row) using the greedy policy for SAC and TD3. The initial random action episodes are not included. The dotted line represents mean value per evaluation episode, outer lines represent minimum and maximum observed values across the five seeds.

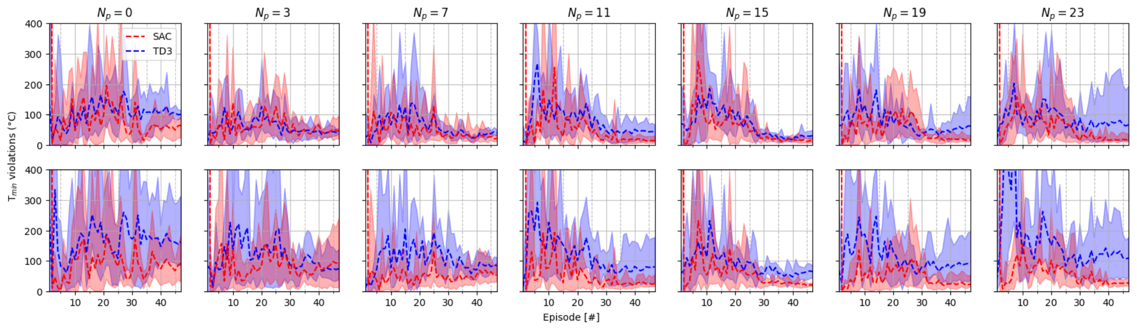


Figure 14
Learning curves describing the summed total lower temperature boundary violations in °C on the training weather data set (upper row) and test weather data (lower row) using the greedy policy for SAC and TD3. The initial random action episodes are not included. The dotted line represents mean value per evaluation episode, outer lines represent minimum and maximum observed values across the five seeds.

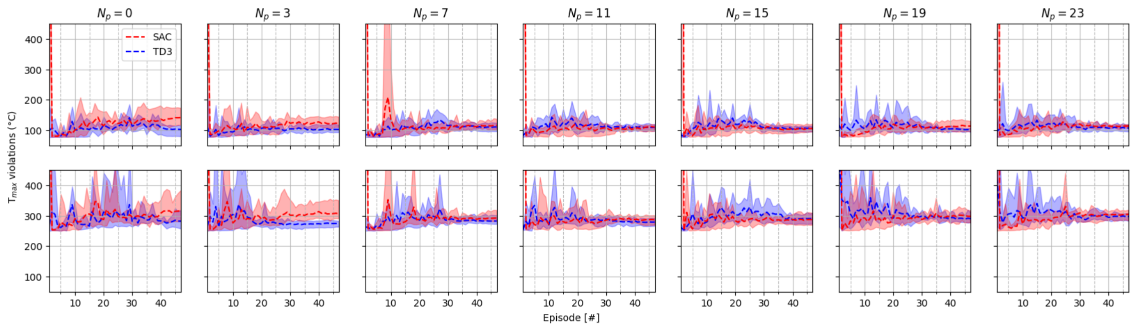


Figure 15
Learning curves describing the summed total upper temperature boundary violations in °C on the training weather data set (upper row) and test weather data (lower row) using the greedy policy for SAC and TD3. The initial random action episodes are not included. The dotted line represents mean value per evaluation episode, outer lines represent minimum and maximum observed values across the five seeds.

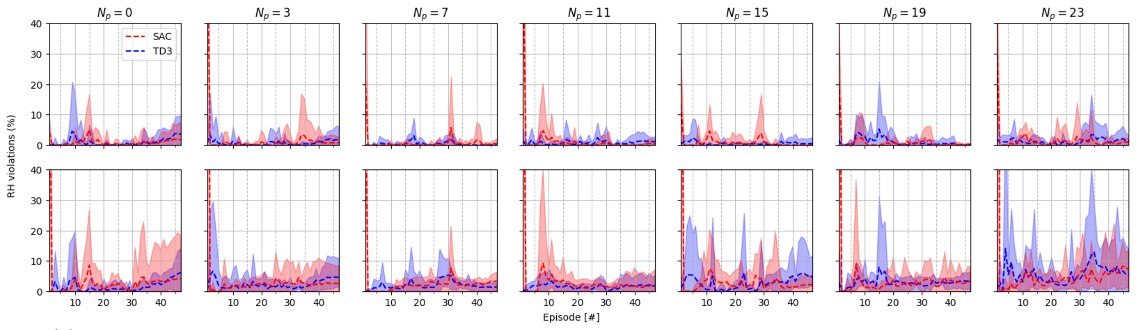


Figure 16
Learning curves describing the summed total relative humidity boundary violations in % on the training weather data set (upper row) and test weather data (lower row) using the greedy policy for SAC and TD3. The initial random action episodes are not included. The dotted line represents mean value per evaluation episode, outer lines represent minimum and maximum observed values across the five seeds.

6.5 Hyperparameter tuning: Limitations and discussion

Agent hyperparameters are unfortunately not reported in these previous works (Wang et al. (2020); Zhang et al. (2021); An et al. (2021); Cao et al. (2022)), nor are learning curves reported and are experiments performed across multiple seeds. This means that no indication exists whether they just got lucky during experiments because of the seed or not, whether a policy would generalize adequately to unseen data and how much variation in policy behaviour would exist when repeated with differing seeds, something that we have shown experiments with no forecast are likely very sensitive to.

With the used experiment setup TD3 was found to finish the learning procedure faster, as it performs only half the total policy and target updates due to the implemented update delay compared to SAC but performed worse. However, TD3 has more hyperparameters related to exploration and regularization that are not tuned in this work (target action noise and its clip value, policy and target network update frequency). Tuning these could improve performance for the current experiment setup but that comes at the cost of additional and exponentially increasing experimentation time.

Additionally, hyperparameters were set for each of the prediction horizon cases (with the exclusion of action noise for the TD3 algorithm). Alternatively, each prediction horizon case can be seen as a different environment and hyperparameter tuning should be done separately. Lower discount factors should increase stability for shorter forecast horizon cases as it reduces variance in Q-function and policy function estimators that originate from differences in weather beyond their forecast horizon, alternatively a larger discount factor may reduce risk of overfitting to noise in the observations at longer forecast horizons. Other optimal hyperparameters may differ too due to differences in observation space information content and size. This comes at the cost of significantly increased computation time and makes one-to-one comparisons of results more difficult.

While properly quantifying effects of each forecast horizon remains difficult due to the hyperparameter sensitivity present in DRL, we have demonstrated the importance of using forecasts. Additionally, better performance and the simplicity of tuning SAC means it is the preferred algorithm for training more reliable control agents that are less sensitive to hyperparameter tuning and stochasticity during the training procedure.

6.5.1 Reward scaling

It is known for the predecessor to TD3, the DDPG algorithm, that performance and stability are very sensitive to the used reward scale and this optimal reward scaling parameter depends on the used environment (Duan et al. (2016); Henderson et al. (2019)). This sensitivity also holds for the SAC algorithm (Haarnoja et al. (2018b,c)), so it is very likely that this also the case for the TD3 algorithm. No work in greenhouse control has explored this as of yet and is interesting for future research.

6.6 Observation space

MPC optimal control approaches make use of access to the state variables, model equations and constraints to plan an optimal action sequence for a prediction horizon. While constraints can be set on both state variables and measurement variables, they are only applied on the variables present in the system equations, which measurements not always are. This means that measurement constraints are converted to their relevant state variables to set constraints on these instead. For example, a constraint on relative humidity is converted to the temperature-adjusted humidity density - so in the same unit as the state variable for humidity, and this is then used as a constraint vector in the optimization procedure (see van van Henten and Bontsema (2009) for an example). Because of this approach in MPC - implicit access to both the state and measurement variables - the decision was made to include the state variables in the observation space as well in this work. However, further preliminary research on the observation space using only the SAC algorithm suggests that inclusion of estimated state variables for CO_2 and absolute humidity is unnecessary.