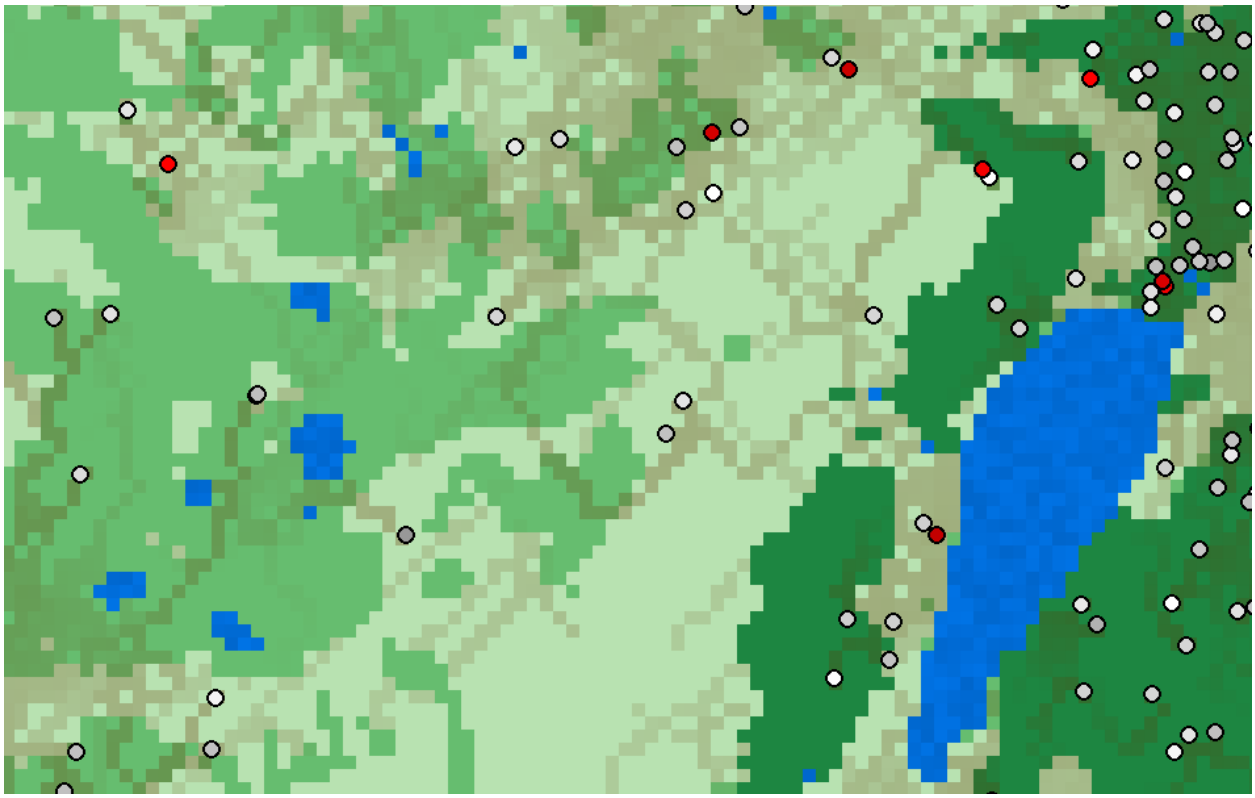

Multi-agent Reinforcement Learning for predator-prey ecosystem

Simulating predator prey ecosystem in a 2D based environment using Multi-agent Reinforcement Learning

Master's thesis in Engineering Mathematics and Computational Science

Michal Palak

DEPARTMENT OF MATHEMATICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Multi-agenent Reinforcement Learning for predator-prey ecosystem

Simulating predator prey ecosystem in a 2D grid based environment using Multi-agent Reinforcement Learning

Michal Palak

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Engineering Mathematics
Chalmers University of Technology
Gothenburg, Sweden 2026

Simulating predator prey ecosystem
in a 2D grid based environment using Multi-agent Reinforcement Learning
MICHAL PALAK

Supervisor: Claes Strannegård, Department of Computater Science and Engineering
Examiner: Petter Mostad, Department of Mathematics

Master's Thesis 2026
Department of Engineering Mathematics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: 2D grid based predator prey ecosystem.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Simulating predator prey ecosystem in a 2D grid based environment using Multi-agent Reinforcement Learning

MICHAL PALAK
Department Engineering Mathematics
Chalmers University of Technology

Abstract

Agent-based ecosystem models provide a flexible way to study population dynamics by simulating the behaviour of individual organisms. However, manually specifying realistic animal behaviour can be difficult, especially when agents must balance multiple needs and interact with other species. This thesis investigates the use of multi-agent reinforcement learning for simulating a simplified predator-prey ecosystem in a two-dimensional grid-based environment.

A custom ecosystem environment was developed in which prey and predator agents interact with spatially distributed resources, including multiple grass types and water reservoirs. The model includes survival constraints based on energy, thirst, age, movement, reproduction, predation, and resource consumption. In contrast to simpler predator-prey simulations, the environment requires prey agents to balance grazing and drinking, creating a simple form of migration between food and water resources. Age-dependent movement speed was also introduced as a way to model increased vulnerability among young and old individuals.

Several learning configurations were evaluated. In particular, the thesis compares a standard survival reward with a homeostatic reward based on internal energy and water levels, as well as two training setups for handling agent death. The results indicate that the homeostatic reward might improve early learning however the results were not statistically significant. The respawn based training environment significantly improves training efficiency compared with the standard setup. Comparisons between PPO, TRPO, TQC and hand coded agents showed broadly similar performance after extended training. Trained agents were also evaluated on satellite-derived terrain maps, where no significant reduction in performance measure was observed compared with Perlin noise-generated maps.

The results suggest that multi-agent reinforcement learning can be used to generate stable predator-prey dynamics in a spatially structured ecosystem while producing useful simulation statistics such as population trends, survival distributions, reproduction patterns, and spatial movement heatmaps.

Keywords: multi-agent reinforcement learning, predator-prey ecosystem, agent-based modelling, reinforcement learning, sustainability, ecosystem simulation.

Keywords: multi-agent reinforcement learning, predator-prey, lotka-volterra.

Acknowledgements

I would like to express my sincere gratitude to Claes Strannegård and Niklas Engsner, for their guidance, support, and valuable feedback throughout the process of writing this thesis. Their expertise, patience, and encouragement have been extremely helpful and deeply appreciated.

I would also like to thank Petter Mostad for their valuable insights, time, and guidance. Their comments and perspective have contributed significantly to the quality and completion of this thesis.

Finally, I would like to thank my partner, for her constant support, patience, and encouragement during this journey.

Michal Palak, Gothenburg, May 2026

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

RL	Reinforcement learning
MARL	Multi agent reinforcement learning
PPO	Proximal policy optimization
TRPO	Trust region policy optimization

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Indices

i, j	Row and column indices of a grid cell in the ecosystem.
k, l	Row and column indices of another grid cell, often used for neighbouring or destination cells.
t	Discrete time step in the simulation or reinforcement learning process.
p	Index for plant or grass resource type, where $p \in \{1, 2, 3\}$.
a	Action index in reinforcement learning contexts.

Sets

$\mathcal{S}$	Set of states in a Markov decision process.
$\mathcal{A}$	Set of possible actions available to an agent.
Ω	Set of possible observations in a partially observable Markov decision process.
$\mathcal{T}$	Set of discrete time steps in the simulation.
$\mathcal{P}$	Set of plant resources in the ecosystem.
$\mathcal{Z}$	Set of animal species in the ecosystem, consisting of prey and predator agents.

Parameters

n	Number of rows in the two-dimensional grid environment.
m	Number of columns in the two-dimensional grid environment.

g_p	Regrowth rate of plant resource p.
$E_{\max}$	Maximum energy capacity of an animal agent.
$W_{\max}$	Maximum water capacity of an animal agent.
$A_{\max}$	Maximum age of an animal agent.
food_E	Amount of energy gained from consuming a food resource.
food_W	Amount of water gained from consuming a food resource.
r	Observation radius of an agent.
α	Natural prey growth rate in the Lotka–Volterra predator–prey model.
β	Predation rate coefficient in the Lotka–Volterra predator–prey model.
γ_{LV}	Natural predator death rate in the Lotka–Volterra predator–prey model.
δ_{LV}	Predator growth rate per prey consumed in the Lotka–Volterra predator–prey model.
γ	Discount factor used in reinforcement learning.
ϵ	Clipping parameter used in the PPO objective function.
η	Normalization constant used when updating belief states in a POMDP.

Variables

R^k	Resource matrix for resource type k.
$R^k_{ij,t}$	Amount of resource k in grid cell (i, j) at time step t.
$R^p_{ij,t}$	Amount of plant resource p in grid cell (i, j) at time step t.
$R^s_{ij,t}$	Number of prey agents in grid cell (i, j) at time step t.
$R^w_{ij,t}$	Number of predator agents in grid cell (i, j) at time step t.
$R^a_{ij,t}$	Water resource value in grid cell (i, j) at time step t.
E_t	Energy level of an animal agent at time step t.
W_t	Water level of an animal agent at time step t.
A_t	Age of an animal agent at time step t.
x	Prey population in the Lotka–Volterra model.
y	Predator population in the Lotka–Volterra model.
s_t	State of the environment at time step t.
a_t	Action selected by an agent at time step t.

o_t	Observation received by an agent at time step t.
r_t	Reward received at time step t.
π	Policy used by an agent to select actions.
$\pi(a\|s)$	Probability of selecting action a in state s under policy π.
$V^\pi(s)$	State-value function under policy π.
$Q^\pi(s,a)$	Action-value function under policy π.
$A^\pi(s,a)$	Advantage function under policy π.
$b(s)$	Belief state, representing a probability distribution over possible states.
$P_a(s,s')$	Transition probability from state s to state s' after taking action a.
$R_a(s,s')$	Expected immediate reward after transitioning from state s to state s' due to action a.
$O_a(o,s')$	Observation probability of receiving observation o after reaching state s' due to action a.
θ	Parameters of the policy network or actor.
w	Parameters of the value function or critic.
δ_t	Temporal-difference error at time step t.

Contents

List of Figures

List of Tables

1
Introduction

Ecosystems and their biodiversity offer invaluable resources, such as clean water, pollination, construction materials, textiles, food, and pharmaceuticals. Yet, the looming extinction of over a million species in the coming years threatens these natural contributions, potentially leading to significant impacts on biodiversity, climate, and human living conditions globally [1]. This calls for immediate, science-driven policy actions to protect biodiversity while ensuring sustainable human development. Computational models of ecosystems play a pivotal role in designing and predicting the outcomes of these policies, offering insights into ecosystem dynamics and the implications of human interventions like recreation, farming, urban development, and forestry. These models, especially agent-based models (ABMs) that simulate individual animal behaviours, are crucial for understanding population dynamics and, by extension, ecosystem functions and structures. However, the traditional approach of manually coding animal behaviours in ABMs is increasingly challenging due to the extensive pre-knowledge and time required. This project proposes leveraging machine learning, particularly Multi-Agent Reinforcement Learning (MARL), as an innovative solution to automate the development of ecosystem models from data. While ML has revolutionized many areas of life sciences, its application in biodiversity science and, specifically, in ABM-based ecosystem modelling, remains under explored. Our investigation focuses on the potential of MARL to model ecosystems, emphasizing animal behaviour's role in agent-based models and its impact on biodiversity and population dynamics.

1.1 Background and related work

Mathematical models and simulations used to gain insight into population dynamics and animal behaviour have been studied for many years. As early as in 1925, the Lotka–Volterra model was introduced to describe the population dynamics of predator–prey systems, demonstrating cyclical behaviour in the populations of both predators and prey. Since then, many mathematical models have been proposed that aim to improve upon earlier ones by incorporating additional features to make population dynamics more realistic.

This top-down modelling approach has been questioned, with some arguing that individual-based models can better capture the stochastic that emerges in such systems [2]. The individual-based approach focuses on simulating the behaviour of each individual within the system. By simulating many individuals simultaneously, population-level patterns can emerge naturally. This approach also makes it easier

to incorporate different mechanisms at the individual level, thereby increasing model complexity without requiring direct assumptions about their effects on population dynamics.
As simulations grow more complex, new methods capable of capturing increasing levels of complexity are being incorporated. One such method is reinforcement learning. In [3], the authors demonstrated that a predator–prey system in which predator decisions were controlled by a reinforcement learning algorithm, while prey actions followed a random policy, exhibited cyclical behaviour similar to that observed in the Lotka–Volterra model. In [4], a more complex model in which prey were also trained as reinforcement learning agents showed that, for a certain set of parameters, a stable limit cycle could again be observed, further demonstrating cyclical population behaviour. Multi-agent reinforcement learning models of predator–prey systems have also been studied in two-dimensional grid-world environments, where the authors demonstrated the emergence of animal behaviours such as flocking [5].

2

Aims and limitations

The objective of this thesis is to design and implement a reinforcement learning based framework for simulating a two dimensional ecosystem with multiple interacting agents. The work focuses on constructing an environment that captures essential ecological properties, including resource distribution, agent needs, and interactions among agents. In contrast to previous work in this area, this thesis aims to create an environment in which prey are required to migrate between water reservoirs and grazing fields. This migration represents a next step in increasing the complexity of the simulations and is intended to capture simple migration dynamics.

Within this environment, a multi-agent reinforcement learning framework is developed to enable agents to learn adaptive behaviours that support survival and coexistence. We note that reinforcement learning models that capture the need for migration can be highly complex, therefore the implementation is intentionally restricted to a simplified migration mechanism involving water reservoirs. Additionally, the thesis introduces a mechanism that captures prey vulnerability to predation when individuals are weakened due to age, considering both young and old individuals.

The thesis further investigates the effectiveness of different learning strategies through experimental evaluation, with the goal of identifying methods that facilitate stable learning and long-term coexistence of the two species. In the experimental evaluation, long-term coexistence is quantified using sustainability measure defined as the number of simulation time steps for which the predator prey ecosystem remains alive during an evaluation episode. To assess the robustness and generalization capabilities of the proposed approach, trained agents are evaluated in previously unseen environments. Finally, example statistics are collected from the simulations, including agent movement patterns, survival duration, and population trends, in order to demonstrate the analytical potential of the proposed framework.

3

Theory

This chapter introduces the main theoretical concepts used in this thesis. We first describe the classical Lotka–Volterra predator-prey model, since it provides a useful reference point for understanding population dynamics. We then introduce reinforcement learning, Markov decision processes, multi-agent reinforcement learning, and the policy optimization algorithms used in the experiments. Last we shortly describe Perlin noise algorithm that we use to generate terrain.

3.1 Lotka–Volterra

The Lotka–Volterra equations [6], also known as the predator-prey equations, are a classical mathematical model used to describe the interaction between two species in an ecosystem. One species is treated as prey and the other as predator.
We denote the prey population by x and the predator population by y. The basic assumptions of the Lotka–Volterra model are:

- The prey population grows exponentially in the absence of predators.
- The predator population decreases in the absence of prey. Encounters with prey contribute positively to predator reproduction, and the predator population grows when prey density is sufficiently high.
- The environment does not change over time and there is no migration.

With these assumptions, the interaction between the two populations can be written as the following system of differential equations:

$$\begin{aligned} \frac{dx}{dt} &= \alpha x - \beta xy, \\ \frac{dy}{dt} &= -\gamma y + \delta xy. \end{aligned}$$

Here, $\frac{dx}{dt}$ and $\frac{dy}{dt}$ describe how the prey and predator populations change over time. The parameter α is the natural growth rate of the prey population when no predators are present. The parameter γ is the natural death rate of predators when no prey is available. The parameter β describes how often predators catch prey, while δ describes how much the predator population grows from consuming prey.
The prey equation consists of two terms. The term αx increases the prey population at a rate proportional to its current size. The term βxy decreases the prey population due to predation. This term depends on both x and y, because predation requires encounters between prey and predators.

The predator equation also consists of two terms. The term $-\gamma y$ decreases the predator population when there is not enough prey. The term δxy increases the predator population based on the number of encounters between prey and predators.

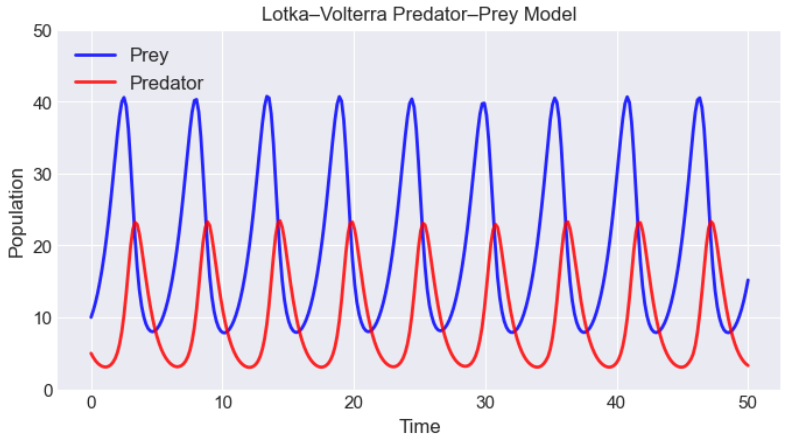

Figure 3.1: Oscillatory population dynamics in the Lotka–Volterra predator-prey model, showing alternating peaks and declines of prey (blue) and predator (red) populations over time.

The Lotka–Volterra model often leads to oscillations in the two populations, as shown in Figure 3.1. When prey is abundant, the predator population can grow because there is enough food. As the predator population grows, more prey are eaten, which causes the prey population to decrease. When there is less prey, the predator population also decreases due to lack of food. This then allows the prey population to recover, and the cycle can start again.
For example, we can think of rabbits as prey and foxes as predators. If there are many rabbits, the fox population can increase. However, when the number of foxes increases, they eat more rabbits, which reduces the rabbit population. When fewer rabbits are available, the fox population decreases. After that, the rabbit population can increase again because there are fewer foxes. This creates a repeating cycle.
In this thesis, the Lotka–Volterra model is not used directly to control the simulation. Instead, it is used as a theoretical reference for predator-prey dynamics. The model developed in this thesis is more detailed in some ways, since it describes individual animals, spatial movement, food, water, age, and predation. However, the Lotka–Volterra model is still useful because it shows the kind of population oscillations that can appear in predator-prey systems.

3.2 Reinforcement Learning

Reinforcement learning is a machine learning approach where an agent learns by interacting with an environment. At each time step, the agent observes some information about the environment, chooses an action, and receives a reward. The goal of the agent is to learn a policy that gives high long-term reward.
In this thesis, reinforcement learning is used to model animal behaviour. Instead of manually defining all movement rules for the animals, the agents learn actions that

help them survive. For example, prey agents need to learn how to find food and water while avoiding predators, while predator agents need to learn how to find and catch prey.

3.2.1 Markov Decision Process

A Markov decision process, or MDP, is a mathematical framework for modelling decision-making over time. It is commonly used to describe reinforcement learning problems. An MDP consists of:

- A set of states S, which describe possible situations in the environment.
- A set of actions A, which describe the possible actions available to the agent.
- Transition probabilities $P_a(s, s')$, which describe the probability of moving from state s to state s' after taking action a.
- A reward function $R_a(s, s')$, which describes the reward received after moving from state s to state s' due to action a.

The transition probability can be written as

$$P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a).$$

The reward function can be written as

$$R_a(s, s') = \mathbb{E}[r_{t+1} \mid s_t = s, s_{t+1} = s', a_t = a].$$

The goal in an MDP is to find a policy π that chooses actions in a way that maximizes the expected future reward. A policy can be seen as the strategy used by the agent. The expected future reward is usually discounted by a factor γ, where rewards received sooner are weighted more than rewards received later.
The value of a state s under policy π is described by the value function

$$V_\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{a_{t+k}}(s_{t+k}, s_{t+k+1}) \mid s_t = s \right].$$

This value function describes how good it is for the agent to be in a certain state when it follows policy π.
In many realistic problems, the agent cannot observe the full state of the environment. This is also true in this thesis, since animals do not have access to complete information about the whole ecosystem. For example, a sheep can only observe resources and animals within a limited range. This type of problem can be described as a partially observable Markov decision process, or POMDP.
A POMDP extends an MDP by adding observations. It can be described as a tuple $(S, A, P_a, R_a, \Omega, O_a)$, where Ω is the set of possible observations and $O_a(o, s')$ is the probability of observing o after action a has led to state s':

$$O_a(o, s') = \Pr(o_{t+1} = o \mid s_{t+1} = s', a_t = a).$$

Since the agent cannot observe the true state directly, it must act based on the information available in its observation.

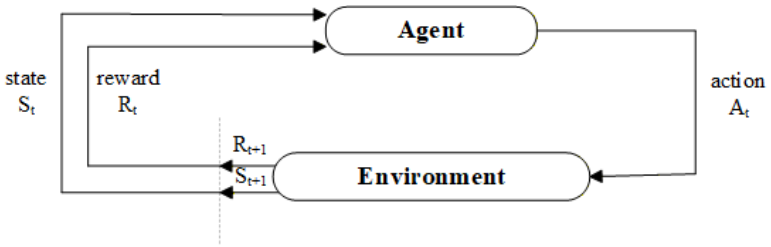

Figure 3.2: Agent-environment interaction in a Markov decision process [7].

3.2.2 Multi-agent Reinforcement Learning

Multi-agent reinforcement learning extends reinforcement learning to settings with several agents acting in the same environment. In this case, the outcome for one agent depends not only on its own action, but also on the actions of other agents. This is important for predator-prey ecosystems. A prey agent changes the environment by moving, eating grass, drinking water, reproducing, or being eaten. A predator agent changes the environment by moving and hunting prey. Since all agents act in the same ecosystem, their decisions affect each other.
In a multi-agent setting, the state transition and reward may depend on the joint action of all agents. If there are several agents, the joint action can be written as

$$\mathbf{a}_t = (a_t^1, a_t^2, \ldots, a_t^N),$$

where N is the number of agents and a_t^i is the action of agent i at time t.
Multi-agent reinforcement learning problems can be cooperative, competitive, or mixed. In cooperative problems, agents share the same goal. In competitive problems, one agent's success may be another agent's failure. Predator-prey systems are mixed, because prey agents benefit from avoiding predators, while predators benefit from catching prey. At the same time, the whole ecosystem can still show stable population behaviour when both species survive over time.
A key challenge in multi-agent reinforcement learning is that the environment becomes non-stationary from the perspective of a single agent. This means that the environment appears to change during training, because the other agents are also learning and changing their behaviour. This makes learning more difficult than in a single-agent problem.

3.3 On-policy Actor-Critic Methods

A policy is the strategy used by an agent to choose actions. In reinforcement learning, a policy maps states or observations to actions. A policy can be deterministic or stochastic.
A deterministic policy chooses one action for each state:

$$a = \pi(s).$$

A stochastic policy gives a probability distribution over possible actions:

$$\pi(a \mid s).$$

In this thesis, the algorithms used are based on actor-critic methods. Actor-critic methods use two main components: an actor and a critic. The actor represents the policy and decides which action the agent should take. The critic estimates how good the current state or action is, and is used to improve the policy.
The actor is usually represented by a policy $\pi(a \mid s; \theta)$, where θ are the parameters of the policy network. The objective of the actor is to adjust θ so that the expected return becomes larger.
The policy gradient theorem gives a way to update the policy parameters:

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi \left[\nabla_\theta \log \pi(a \mid s; \theta) A^\pi(s, a) \right].$$

Here, $J(\theta)$ is the expected return and $A^\pi(s, a)$ is the advantage function. The advantage function describes how much better an action is compared to the average action in the same state.
The critic estimates a value function. The state-value function is written as

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid S_t = s \right].$$

It describes the expected future reward from state s when following policy π. Another common value function is the action-value function,

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid S_t = s, A_t = a \right].$$

This describes the expected future reward after taking action a in state s and then following policy π.
The critic is trained by minimizing the difference between its estimated value and a target value. A common loss function is

$$L(w) = \mathbb{E}_\pi \left[\left(r_{t+1} + \gamma V(s_{t+1}; w) - V(s_t; w) \right)^2 \right],$$

where w are the critic parameters.
The difference

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}; w) - V(s_t; w)$$

is called the temporal-difference error. It measures how different the received reward and predicted future value are from the current value estimate. This signal is used both to train the critic and to guide the actor update.
On-policy actor-critic methods train the policy using data collected from the current policy. This is important for algorithms such as PPO and TRPO, which update the policy while controlling how much it changes between updates.

3.4 TRPO

Trust Region Policy Optimization, or TRPO, is a reinforcement learning algorithm introduced by Schulman et al. [8]. The main idea behind TRPO is to improve the policy while preventing updates that are too large.

Large policy updates can make learning unstable. A policy that performs well before an update may perform much worse after the update if the parameters are changed too much. TRPO addresses this problem by using a trust region. The trust region limits how far the new policy is allowed to move away from the old policy.
TRPO optimizes the following objective:

$$\max_{\theta} \mathbb{E}_{\pi_{\theta_{\text{old}}}} \left[\frac{\pi_\theta(a \mid s)}{\pi_{\theta_{\text{old}}}(a \mid s)} A^{\pi_{\theta_{\text{old}}}}(s,a) \right].$$

This objective encourages actions that have positive advantage under the old policy. However, the update is constrained by the KL divergence between the old and new policies:

$$\mathbb{E}_{\pi_{\theta_{\text{old}}}} \left[D_{\text{KL}} \left(\pi_{\theta_{\text{old}}}(\cdot \mid s) \parallel \pi_\theta(\cdot \mid s) \right) \right] \leq \delta.$$

Here, D_{KL} is the Kullback–Leibler divergence and δ controls the maximum allowed policy change. The advantage function $A^{\pi_{\theta_{\text{old}}}}(s,a)$ estimates whether an action is better or worse than expected under the old policy.
The update procedure in TRPO can be described as:

1. Estimate the policy gradient using sampled trajectories.
2. Approximate the natural policy gradient while respecting the trust region.
3. Update the policy parameters without exceeding the KL divergence constraint.

3.5 PPO

Proximal Policy Optimization, or PPO, is a reinforcement learning algorithm introduced by Schulman et al. [9]. PPO builds on the same idea as TRPO, namely that policy updates should not be too large. However, PPO uses a simpler optimization procedure.
Instead of enforcing a hard KL divergence constraint, PPO usually uses a clipped objective function. This makes the algorithm easier to implement and tune compared to TRPO, while still keeping policy updates within a reasonable range.
The PPO-Clip objective is written as

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

where

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}.$$

Here, $r_t(\theta)$ is the probability ratio between the new policy and the old policy. The term A_t is the advantage estimate at time t, and ϵ is a hyperparameter that defines the clipping range.
The clipping mechanism limits how much the probability ratio can change. If the new policy changes too much compared to the old policy, the objective is clipped. This discourages very large policy updates and helps keep training stable.
PPO is widely used because it gives a practical balance between performance, stability, and implementation simplicity. It is less computationally demanding than TRPO, while still controlling the size of policy updates.

3.6 Perlin noise

Perlin noise is a well known algorithm that makes it possible to generate noise that looks more naturally than ordinary random values. It is often used, for example, to generate terrain, textures, clouds, or other spatial patterns [10]. In this master's thesis, Perlin noise was used to generate distribution of resources in the ecosystem. It was chosen because real environments do not look completely random. In nature, larger connected areas often occur, such as grass fields or water reservoirs. They don't appear at random in each separate grid cell, but instead form connected clusters and continuous regions.
The main idea of Perlin noise is to create random patterns, that allow for smooth transitions and formation of clusters. The point is not for every tile on the map to have a completely different random value, because it would result in a chaotic pattern. In Perlin noise, points that are close to each other are assigned similar values in order to create smooth transitions and more naturally looking patterns.
Perlin noise can create patterns of varying complexity by adjusting the frequency parameter which controls how often the pattern changes. Low frequency creates large, smooth, connected regions, while high frequency creates smaller and more detailed areas.
Because of this, different frequencies are used to create different types of environments with varying complexity. For example, a frequency of 2 produces broad regions with slow transitions, while frequencies such as 8 or 16 create more fragmented patterns with many smaller clusters.

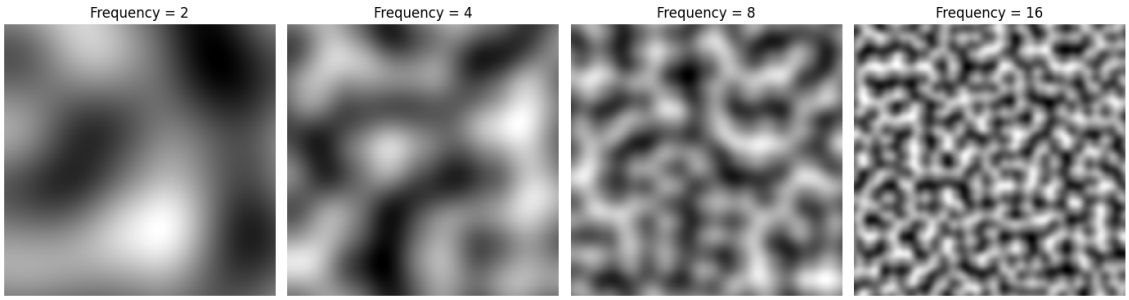

Figure 3.3: The figure shows Perlin noise at different frequencies. When the frequency is low, for example 2, the pattern has large and smooth patches. The higher the frequency, the more details appear in the image. At a frequency of 16, the pattern is much finer and more "rough."

4

Methods

In this section we explain and motivate our approach of modelling ecosystem. Because the main focus of this study is to investigate use of Reinforcement Learning algorithm in modelling more complex animal behavior we need to create an environment with right amount of complexity. This is not an easy task, because an environment that is to easy might not fully leverage the advantage given by Reinforcement Learning. On the other hand an environment that is too complex might be infeasible to implement and test due to limitations in time and resources. Therefore we note this as a limitation in our work, because different level of ecosystems complexity may lead to different conclussions.

4.1 Modelling the ecosystem and RL - framework

4.1.1 Modelling ecosystem

In order to model the ecosystem, we need a structured way to quantify and represent the various resources that make up the environment. A resource can be anything that plays a role in the ecosystem, such as plants, nutrients in the soil, or even living organisms. For example, animals like fish or rabbits can also be considered resources, as they interact with and influence the ecosystem dynamics.

To represent these resources in a two-dimensional space, we adopt a straightforward grid-based approach, where each resource is stored in a separate matrix. This results in a discretized ecosystem consisting of $n \times m$ grid cells, where n represents the number of rows and m represents the number of columns. Each cell corresponds to a specific area within the ecosystem, with its size determined by the scale at which we wish to model the environment. The ecosystem is thus described by a set of resource matrices $R^k \in \mathbb{R}^{n \times m}$, where R^k_{ij} represents the quantity of resource k in cell (i, j).

For example, if we want to model a 2×2 km ecosystem, we can define any number of rows and columns based on the desired level of precision. If we set $n = 2$ and $m = 2$, each of the four grid cells in our model represents a 1×1 km section of the ecosystem. Using this representation, we narrow our focus to six key resources: three types of grass, a water resource, and two resources representing predators and prey.

Each resource in the model requires a set of rules that govern its changes over discrete time steps. These rules determine how resources evolve and interact with other elements in the system. For instance, some resources may follow diffusion models

to simulate the spread of nutrients, which in turn influence plant growth. However, since our primary focus is on using reinforcement learning (RL) to model animal behavior, we use a simplified approach for plant growth. Each of the three grass species, which serve as a food source for prey, regrows at different rates following the equation:

$$R^p_{ij,t+1} = \min(R^p_{ij,t} + g_p, 1)$$

where g_p is the growth rate of plant p. This means that when prey consumes a grass patch, it regrows linearly with factor g_p until it reaches its full capacity $R^p_{ij,t} = 1$. This dynamic introduces a possibility for prey to choose between better or worse grazing grounds depending on the grass growth rate, which allows more complex animal behavior. In order to model animal behavior within our ecosystem, we need

Figure 4.1: Visual representation of wolf (red dot) and sheep (white dot) in the simulated ecosystem.

a way to represent their movement, energy consumption, and survival constraints in a structured manner. Animals are dynamic entities that move continuously across the grid-based environment. To capture this, we use a discrete grid representation where each cell corresponds to a specific location in the ecosystem. When an animal moves from one grid cell to another, the grid is updated to reflect the change in position. Animal movement speed varies with age, with younger and older animals

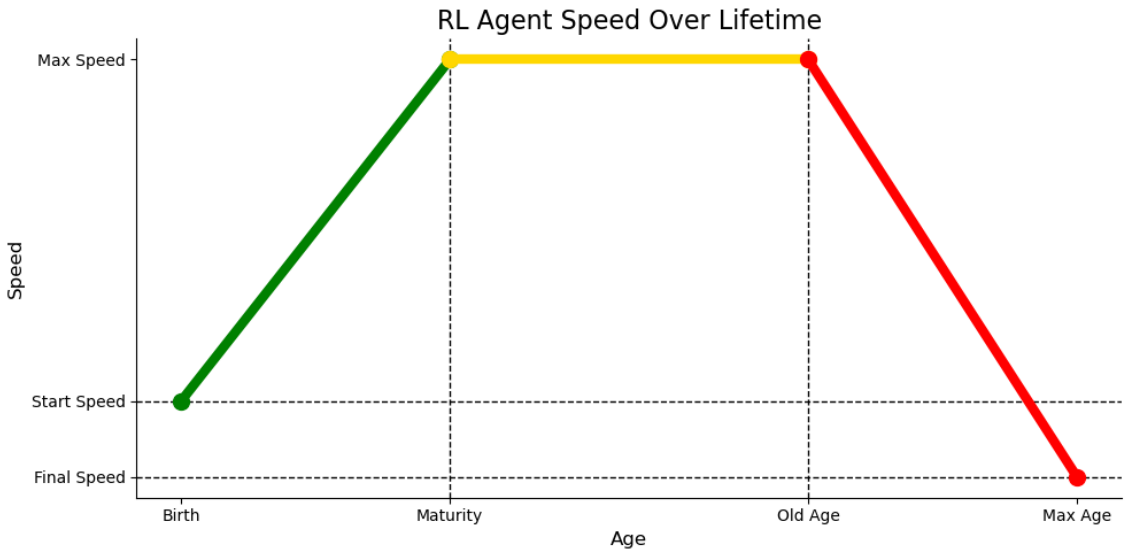

Figure 4.2: Visual representation of agents speed as a function of age

generally moving more slowly. Speed increases linearly as the animal develops, reaches a peak at maturity, and then decreases linearly as the animal ages.

Since movement plays a key role in the ecosystem dynamics, the speed variation introduces diversity in how animals interact with their surroundings. This mechanism also adds a naturally occuring vulnerabiulity of young and old individuals to predation.
Apart from movement, animals require energy to survive. Each animal has an energy value that decreases by 1 unit per time step, and if it reaches zero, the animal dies. Energy has a maximum capacity, and animals can replenish it by consuming food resources available in the ecosystem. This creates a dependency between animal survival and resource availability.

Figure 4.3: Visual representation of ageing animals. When animals are born they start with the color on the left and as they age their color shifts towards the darker shade as they reach their maximum age.

In addition to energy, animals also require water. Each animal has a water value that depletes over time, and if it reaches zero, the animal dies. Water can be replenished by drinking, which in this model occurs when an animal is adjacent to a water source or consumes food that contains water. However, animals cannot enter water cells; if an animal moves into a water cell, it dies. This restriction introduces complexity in movement decisions, as animals must navigate their environment carefully to balance their need for food, water, and survival.
By structuring animal behavior in this way, we create an ecosystem where movement, resource consumption, and survival constraints interact dynamically, leading to emergent behaviors that reflect real world ecological principles. The distribution of resources is generated using Perlin noise algorithm [10], a well known algorithm used for terrain generations. Example the ecosystem can be seen in the figure 4.4 above where we can see different grass species represented by different shades of green and blue patches of water reservoirs. In addition the green grass patches get darker when consumed by a sheep and comes back to it's original color as it regrows.

4.2 RL - modelling approach

We use the PettingZoo library to train our autonomous agents, as it is one of the most commonly used multi-agent reinforcement learning (RL) frameworks. This means that experience collected from many sheep and wolf agents during training is used to update the neural network policy. After training, the agents use this trained policy to choose actions, but they do not update the policy during the simulation. This means that the learning should not be interpreted as an individual sheep or wolf learning from its own experience. It should rather be seen as a proxy for adaptation on the species level, similar to how behaviour in nature can be shaped by evolution.

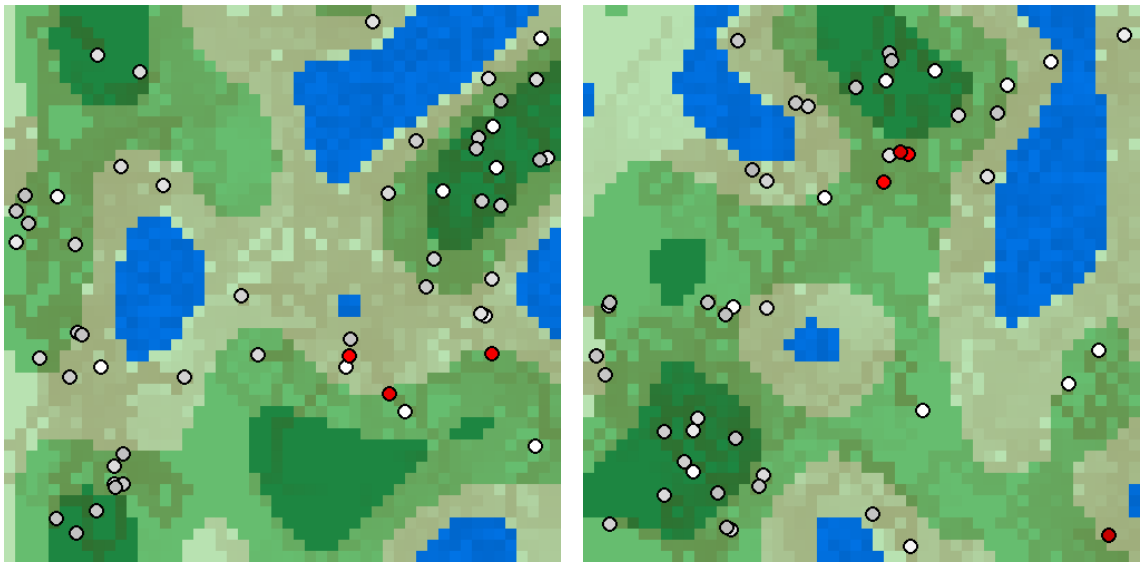

Figure 4.4: Examples of ecosystems with resource distributions generated using Perlin noise. Different shades of green represent different grass species, while blue patches indicate water reservoirs.

For our experiments, we focus on three specific algorithms implemented in PettingZoo: PPO, TRPO, and TQC. These algorithms have been shown to perform well in various RL settings, and our goal is to determine which one works best for our setup.
We train our models on the previously described environments using two different configurations, as detailed in Section 4.3. Additionally, we evaluate two different reward functions commonly used in ecosystem modeling. The first reward function, referred to as the "survival" reward (also know as heartbeat reward), simply grants an agent one point per timestep it survives. The total cumulative reward for an agent is therefore equal to the total number of timesteps it remains alive.
The second reward function takes into account not only survival but also the agent's well-being. In this case, an agent that is starving or is dehydrated receives a lower reward, reflecting the idea that survival alone is not sufficient—the agent should also strive for a sustainable and healthy existence. This approach models the principle that real animals aim not only to survive but to maintain a certain quality of life. Of course, in nature, an animal's well-being is influenced by factors beyond just food and water, such as social interactions and status within a group. However, in our simplified model, we focus solely on survival and resource consumption, as these are the most relevant aspects for our current study. Investigating additional factors could be an interesting direction for future research.
To train a neural network that makes decisions for our agents, we need to define its input, output, and reward structure. This means specifying what information an animal can perceive from its environment, how it processes this information, and what actions it can take.
The input provided to an agent must be realistic, meaning it should reflect sensory or internal information that a real-world animal would have access to. For example, an

agent can perceive its surroundings visually, but it should not have direct knowledge of the exact location of the nearest food source. Additionally, the input must be structured in a way that facilitates learning, as we are working with relatively small neural networks and aim for efficient training and inference.
We categorize inputs into two main types: internal inputs and sensory inputs.
Internal inputs relate to the agent's internal state. We include age, energy, and thirst as internal inputs, as these factors directly impact survival. Since all the RL algorithms we use follow an actor-critic approach, these inputs are particularly useful for the critic network, which evaluates how good a given state is. Consider, for example, an agent using the survival reward function. If a sheep agent has internal inputs (Age: 120, Energy: 30, Water: 100), we can estimate that its minimum expected reward is 120. Furthermore, assuming no external threats (e.g., predation), the sheep can survive for at least 30 more timesteps since it has sufficient energy and water. Accurate state evaluation is crucial in a multi-agent RL setup, where assessing the long-term impact of an action can be challenging. Sensory inputs

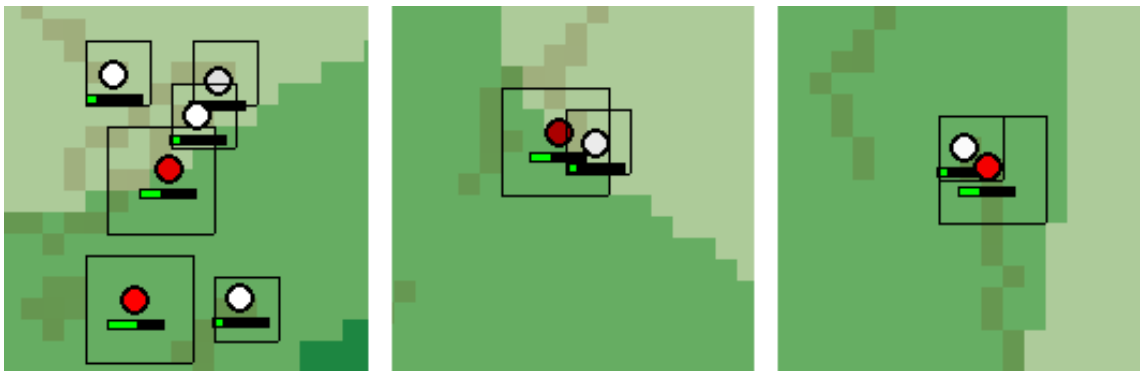

Figure 4.5: Graphical representation of wolf and sheep observations spaces. Observation space is represented with the black square around animals. (Left) None of the animals see each other. (Center) Sheep is within wolfs observation space. (Right) Both wolf and sheep are within each others observation space

include vision and smell. Vision is modelled by providing the agent with a subset of the resource grids within a fixed range of its position. Different species have different vision ranges: sheep can see a 3×3 grid around them, while wolves have a larger 5×5 vision range, reflecting their role as predators.
Smell is implemented in two ways. The first and simpler approach provides wolves with a directional vector that represents direction to all sheep within the ecosystem, weighted by distance cubed as in equations below.

$$A_j = \sum_{i=1}^{n_{prey}} \frac{v_{ji}}{\|v_{ji}\|^3}, \quad \forall j \in \{1, ..., n_{prey}\} \tag{4.1}$$

$$S_j = \frac{A_j}{\|A_j\|}, \quad \forall j \in \{1, ..., n_{predator}\} \tag{4.2}$$

Where n_{sheep} and n_{wolf} is the number of sheep and wolves in the ecosystem, v_{ji} is the vector pointing from predator j to prey i, A is the aggregated weighted vector and S

is the final smell vector that is simply A normalized. This feature is included only for wolves to give them a slight advantage during training, as predators generally face more difficulty in learning optimal policies. By structuring the input space in this

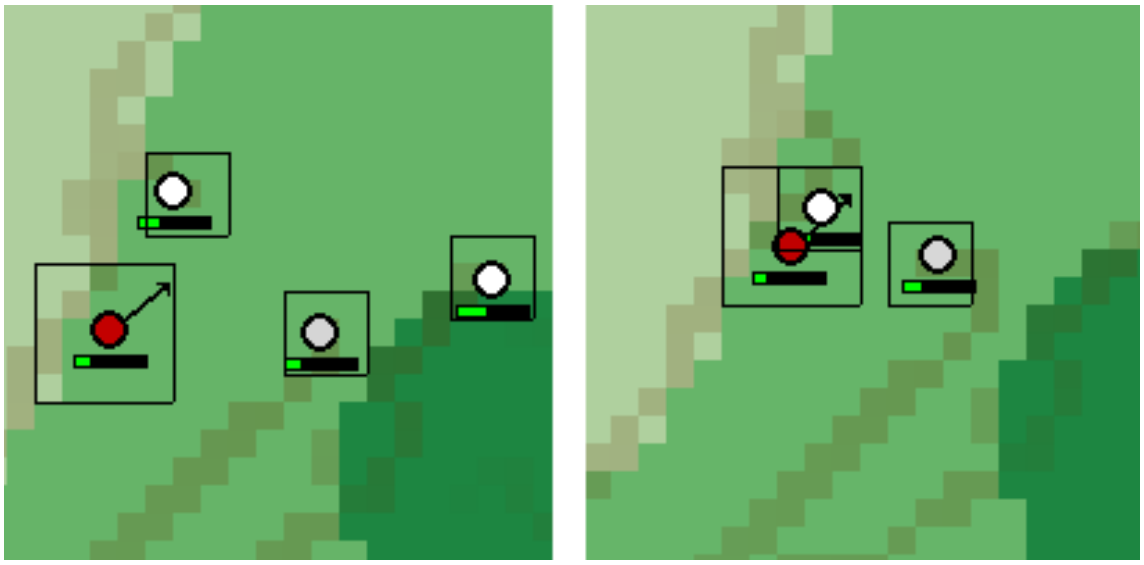

Figure 4.6: Graphical representation of wolf smell. Direction of smell is represented with the black arrow pointing from the wolf. Direction of the arrow is affected more by sheep that are closer

way, we ensure that agents have access to relevant information while maintaining a level of realism in their decision-making process. This approach allows us to train models that capture key ecological dynamics while remaining computationally efficient.
The genral smell that is used by both sheep and wolves is modelled by a 3x3 grid values of Gaussian kernel for a specific resource (see figure below).

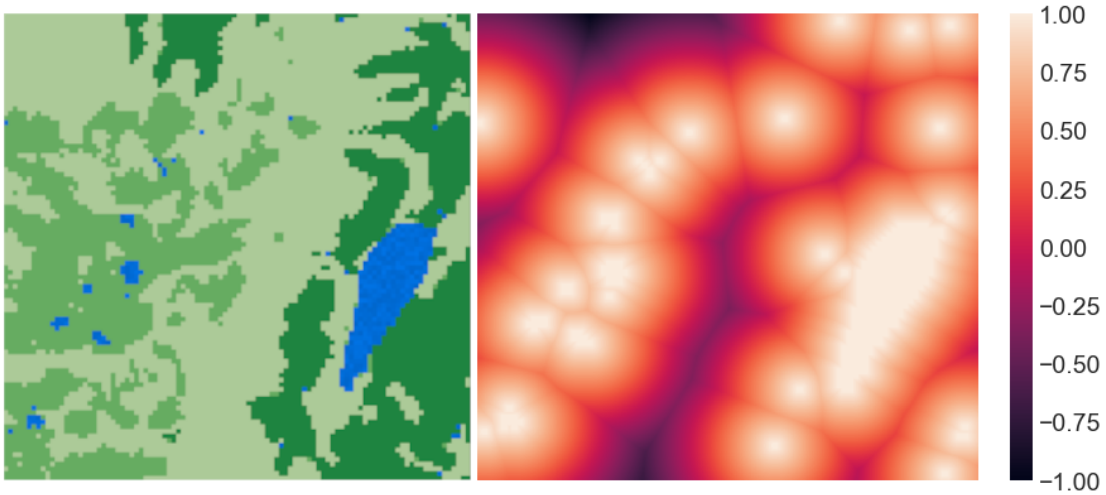

Figure 4.7: Gaussian Kernel representation (right) of water resource from one of the maps used (left)

this approach gives a simple and efficient way to provide sheep with informations about where different types of resources are as well as for both species to know where the water is. Of course this also gives some additional information for wolves and informations for sheep on where the wolves are.
We are using one neural network to model sheep and wolf brain. This is because we think that in our model sheep and wolves need to learn quite similar thing and that the network will be able to generalize quite good for different species. That is why we also add a input that tells the animal or the network if it takes a action for a sheep or a wolf.

4.3 Experimental evaluation

In this section we describe what methods we used in comparison of different setups to train our RL agents. We describe in detail motivation for the steps taken and give important details regarding training.

4.3.1 Sustainability measure

In the experimental evaluation, model performance is assessed using a measure called sustainability. In this thesis, sustainability is defined as the total number of simulation time steps during which both species remain alive. If either species goes extinct, the simulation is terminated.
This termination criterion is used because the purpose of the model is to study predator prey dynamics rather than the survival of a single species in isolation. If wolves go extinct and the simulation continues with only sheep, the sheep population is mainly limited by the available resources in the ecosystem, and the simulation no longer represents a predator prey system. Conversely, if sheep go extinct, wolves typically die shortly afterwards due to lack of food. Ending the simulation when one species goes extinct therefore provides a direct measure of whether both species are able to coexist over time.
In the comparison experiments, evaluation episodes are capped at 4000 time steps. This cap is used due to computational limitations and because simulations that reach 4000 time steps are often stable. However, the simulation framework itself does not require this limit, simulations can be run for as long as both species remain alive.

4.3.2 Training setups

In PettingZoo, we need to define the total number of agents that will be active during training. While this works well for most multi-agent RL setups, it poses a challenge in our case since we want agents to be able to die for example, from starvation or other causes.
One way to handle dead agents in PettingZoo is by using a technique called "black death", where all inputs for a dead agent are set to zero. However, this approach may impact both the performance and training time of our RL algorithm, as the presence of inactive agents could introduce inefficiencies in learning.

To evaluate whether black death is the best approach, we compare it to an alternative simulation method. In this method, we use an environment that closely mirrors the original one, maintaining the same map setup and the same rules governing resources and interactions. The key difference lies in how we handle dead agents: instead of setting their inputs to zero, we allow them to respawn randomly on an available free square. This ensures that the total number of agents remains constant throughout training.
Of course, this method introduces its own limitations, as it is less biologically realistic as real animals do not instantly reappear after dying. However, we believe that this approach still captures most of the essential ecosystem dynamics while offering a potential solution to the inefficiencies introduced by the black death method. Comparing both approaches will allow us to assess their impact on learning performance and determine which one is better suited for our setup.

4.3.3 Comparison of different reward types

The survival reward also known as heart beat reward has been shown to be effective. However, variations of survival-like rewards have also been tested and have demonstrated promising results. In our study, we compare the survival reward with our custom-designed homeostatic reward, which was briefly introduced in Section 4.3. Here, we provide a more detailed motivation for its structure, as outlined in the following equation.

$$f(x,a) = \begin{cases} -\left(\frac{x}{a}\right)^2 + 2\left(\frac{x}{a}\right) & , x \leq a \\ 1 & , x > a \end{cases} \tag{4.3}$$

Which is simply a quadratic function with it's maximum at $x = a$ as shown in the figure below.

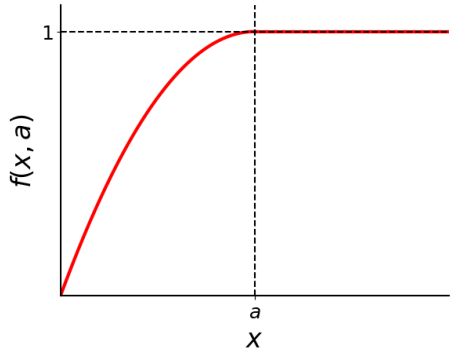

Figure 4.8: Function used to model homeostatic reward described in equation 4.3.

And by using $f(x,a)$ and agents internal values for water W and energy E we obtain:

$$\text{Homeostatic Reward} = f(W, 0.8) + f(E, 0.8) \tag{4.4}$$

The core idea behind the homeostatic reward is to help the critic function better evaluate the impact of eating and drinking on an agent's survival. Unlike the survival

reward, which only incentivize survival, the homeostatic reward directly ties the reward function to an agent's internal states, energy and water levels. In essence, we aim to encourage the RL agent to maintain homeostatic balance, as real animals automatically regulate their internal states to survive.
A simple way to model this would be to assign a reward proportional to the sum of an agent's energy and water levels. However, this approach does not account for the fact that animals prioritize different needs depending on their current state. For example, an agent with low water but moderate energy should be more strongly incentivized to drink rather than eat.
To address this, we cap the maximum reward an agent can receive from energy and water at a certain threshold (set to 0.8 in our case). Additionally, we scale the reward so that the difference in reward becomes larger when the agent is hungrier or thirstier. This ensures that agents are more sensitive to extreme conditions and prioritize actions accordingly, as reflected in the equation above.

4.3.4 Different algorithms

There are many reinforcement learning algorithms that may perform better or worse in our scenario. Therefore, it is important to evaluate whether different algorithms lead to differences in simulation performance.
Due to constraints imposed by the framework used and implementation challenges, we limited our study to PPO, TRPO, and TQC. These algorithms are closely related: both PPO and TQC are derived from TRPO and can be considered variations of it. We therefore investigate whether these variations result in meaningful performance differences in our scenario.
For all experiments, the default Stable-Baselines3 implementations and base network architectures were used. In addition, we evaluate a simplified neural network consisting of a single linear layer using the state-of-the-art PPO algorithm. This allows us to assess whether reducing network complexity can yield benefits in terms of computational speed and model simplicity.

4.3.5 Out of distribution

As previously mentioned, the agents are trained in environments generated using the Perlin noise algorithm. However, real-world ecosystems differ significantly and are far more complex. While certain aspects of natural environments can be approximated using Perlin noise, it cannot capture the full complexity of real ecosystems. To evaluate whether the trained agents can survive in a more complex environment, we test them on terrains derived from satellite images. These terrains are collected from Stelvio National Park in Italy using Google Earth imagery [11]. The images are converted into maps suitable for our simulation by matching the RGB values in the images to the closest corresponding values used in the simulation. As a result, green land areas such as forests or plains are represented by grass types in the simulation, while dark blue regions corresponding to lakes are converted into water reservoirs. This approach has limitations, as it does not account for several important factors, such as altitude. The purpose of this method is solely to test agent performance

in an environment where the spatial distribution of resources more closely resembles that of the real world. We evaluate this by testing sustainability across ten different terrain patches collected from Google Earth and comparing the results to sustainability measured in Perlin noise–generated environments.

5
Results

5.1 Reward

Reward types were investigated by training 5 models for each of two different reward types and saving them at different time steps during training. These models were then compared using the sustainability measure. Each saved model was tested 5 times and the results can be seen in Figure 5.1 below. We can see that the Wellness reward results in faster learning and better results at 4 million timesteps.

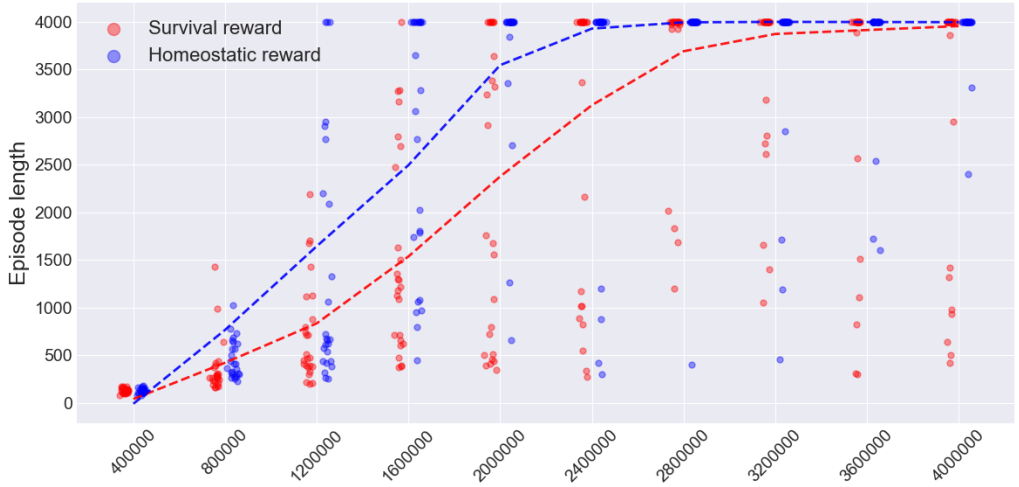

Figure 5.1: Sustainability as a function of RL training progress for the Survival reward and Homeostatic reward. The x-axis shows the number of training time steps completed before the saved model was evaluated. The y-axis shows sustainability, measured as the length of an evaluation episode in simulation time steps. Evaluation episodes were capped at 4000 time steps; therefore, a value of 4000 means that the simulation survived until the maximum evaluation length. Each dot represents one evaluation simulation of a saved model at a given training checkpoint. For each reward type, 5 independently trained models were saved at several checkpoints and each saved model was evaluated 5 times. Dashed lines show LOESS-smoothed trends for each reward type.

Resulting sustainability under the two reward types was compared using exact two-sided permutation tests. Because results from the same trained model are dependent across training checkpoints, the comparison was performed at the model level rather than by permuting individual episodes. Training speed was defined as the first check-

point at which a model reached a predefined success criterion for two consecutive checkpoints.
In the main analysis, a checkpoint was considered successful if at least 3 out of 5 evaluation episodes reached an episode length of 4000. In the sensitivity analysis, a checkpoint was considered successful if the mean episode length across the 5 evaluation episodes was above 3000. For each trained model, the first checkpoint satisfying the corresponding criterion for two consecutive checkpoints was recorded, and the two reward types were then compared using an exact two-sided permutation test. The results are shown in Table 5.1 below.

Analysis	**p-value**
Main analysis	0.0830
Sensitivity analysis	0.3755

Table 5.1: P-values from exact two-sided permutation tests for the reward comparison experiment. The main analysis used the criterion of at least 3 of 5 episodes reaching 4000 for two consecutive checkpoints. The sensitivity analysis used the criterion of mean episode length above 3000 for two consecutive checkpoints.

5.2 Training environment

Training setups were investigated by training 5 models for each of two different training setups and saving them at different time steps during training and comparing these models using sustainability measure. Each model was tested 5 times and results can be seen in Figure 5.2 below where we can see that the trainenv setup results in much better performance in training time. We can see that the training seems to have converged after around 2M steps while the normal training environment does not seem to converge even after 4M training steps.
Resulting sustainability under the two different training environments was compared using exact two-sided permutation tests. Because results from the same trained model are dependent across training checkpoints, the comparison was performed at the model level rather than by permuting individual episodes. Training speed was defined as the first checkpoint at which a model reached a predefined success criterion for two consecutive checkpoints.
In the main analysis, a checkpoint was considered successful if at least 3 out of 5 evaluation episodes reached an episode length of 4000. In the sensitivity analysis, a checkpoint was considered successful if the mean episode length across the 5 evaluation episodes was above 3000. For each trained model, the first checkpoint satisfying the corresponding criterion for two consecutive checkpoints was recorded, and the two training setups were then compared using an exact two-sided permutation test. The results can be seen in Table 5.2 below.

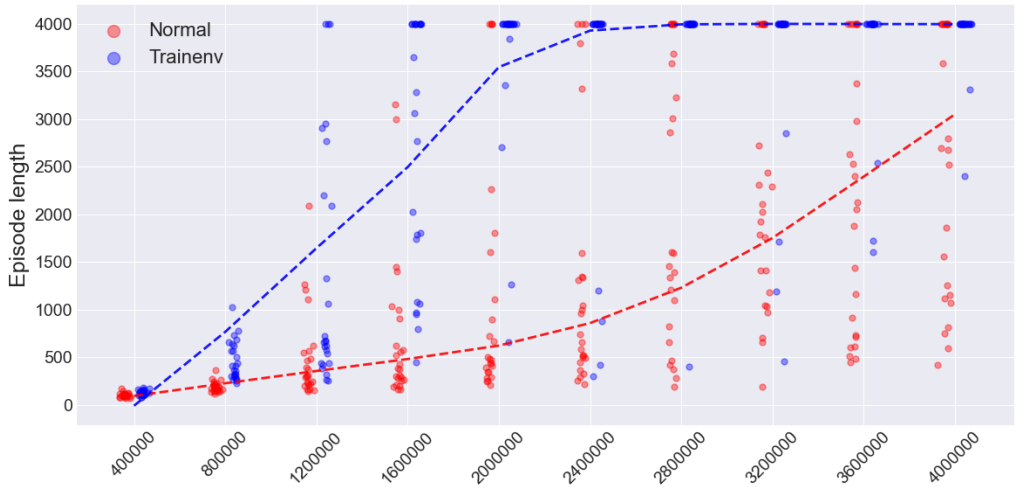

Figure 5.2: Sustainability as a function of RL training progress for two different training setups (Trainenv and Normal). The x-axis shows the number of training time steps completed before the saved model was evaluated. The y-axis shows sustainability, measured as the length of an evaluation episode in simulation time steps. Evaluation episodes were capped at 4000 time steps; therefore, a value of 4000 means that the simulation survived until the maximum evaluation length. Each dot represents one evaluation simulation of a saved model at a given training checkpoint. For each reward type, 5 independently trained models were saved at several checkpoints and each saved model was evaluated 5 times. Dashed lines show LOESS-smoothed trends for each reward type.

Analysis	**p-value**
Main analysis	0.0356
Sensitivity analysis	0.0119

Table 5.2: P-values from exact two-sided permutation tests for the training environment comparison. The main analysis used the criterion of at least 3 of 5 episodes reaching 4000 for two consecutive checkpoints. The sensitivity analysis used the criterion of mean episode length above 3000 for two consecutive checkpoints.

5.3 Algorithm

Performance of different algorithms was evaluated by training different models for 4M steps using different reinforcement learning algorithms and comparing them based on sustainability measure capped at 4000 steps. Results can be seen in Figure 4.4 below. We can see that most algorithms perform similarly well after 4M steps, although PPO2 and Hand coded show lower performance than the other algorithms.

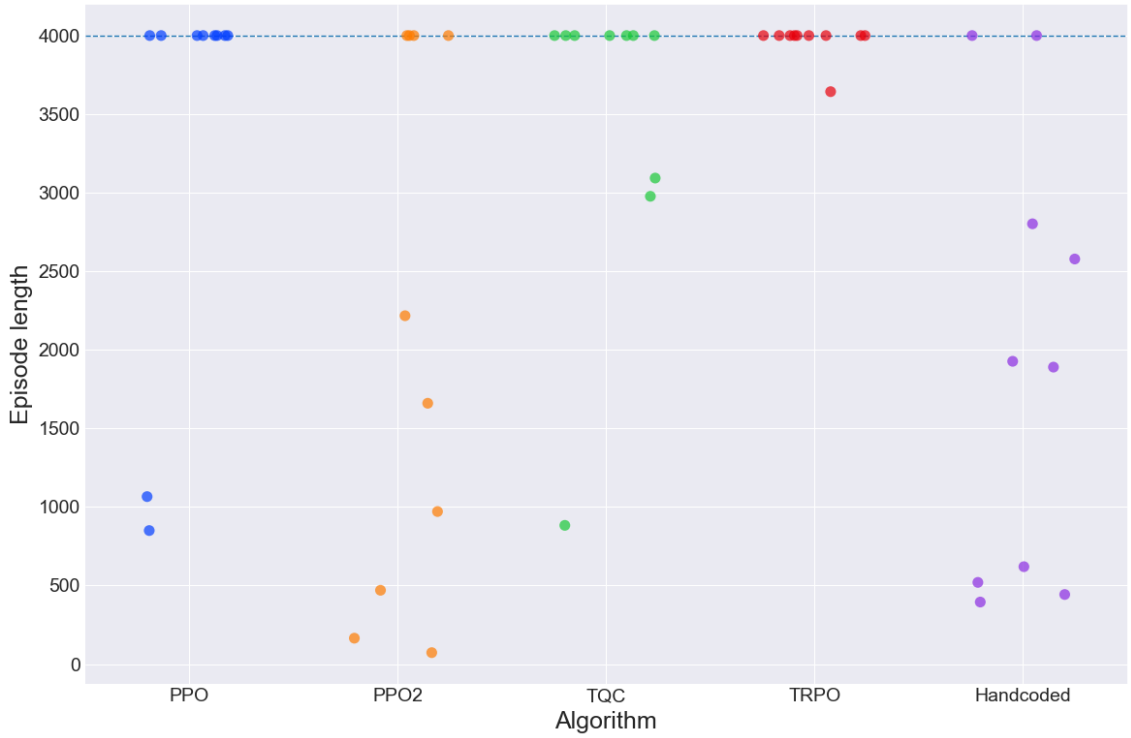

Figure 5.3: Sustainability for different algorithms. The x-axis shows the algorithm used to control the agents: PPO, PPO2, TQC, TRPO, or the hand-coded. For the reinforcement-learning methods, the policy was evaluated after 4 million training time steps. The y-axis shows sustainability, measured as the length of an evaluation episode in simulation time steps. Evaluation episodes were capped at 4000 time steps; therefore, a value of 4000 means that the ecosystem survived until the maximum evaluation length. Each dot represents one independent evaluation simulation for the corresponding method.

Resulting sustainability for different algorithms was compared using exact two-sided permutation tests based on binary success. A run was considered successful if the episode length reached 4000 time steps. PPO was used as the reference algorithm and compared separately against PPO2, TQC, TRPO and Hand coded. The test statistic was the difference in win rate between PPO and the comparison algorithm. Holm correction was applied across the four pairwise comparisons. Results can be seen in Table 5.3 below. Although PPO had a higher win rate than PPO2 and Hand coded, and a lower win rate than TRPO, none of these differences were statistically significant after Holm correction.

Comparison	Other win rate	Difference	p-value	p-value*
PPO vs PPO2	0.40	0.40	0.170	0.509
PPO vs TQC	0.70	0.10	1.000	1.000
PPO vs TRPO	0.90	-0.10	1.000	1.000
PPO vs Handcoded	0.20	0.60	0.023	0.092

Table 5.3: Results of exact two-sided permutation tests comparing PPO against each alternative algorithm using binary success as the outcome, where success was defined as reaching an episode length of 4000. Differences are reported as PPO win rate minus the comparison algorithm win rate. * Holm adjusted p-value.

5.4 Out of distribution

All models were trained on worlds generated using Perlin noise. In order to evaluate model performance on maps that resemble the real world, 10 maps were generated for our world based on Google Maps images collected in Stelvio National Park. Results can be seen in Figure 4.4 below.

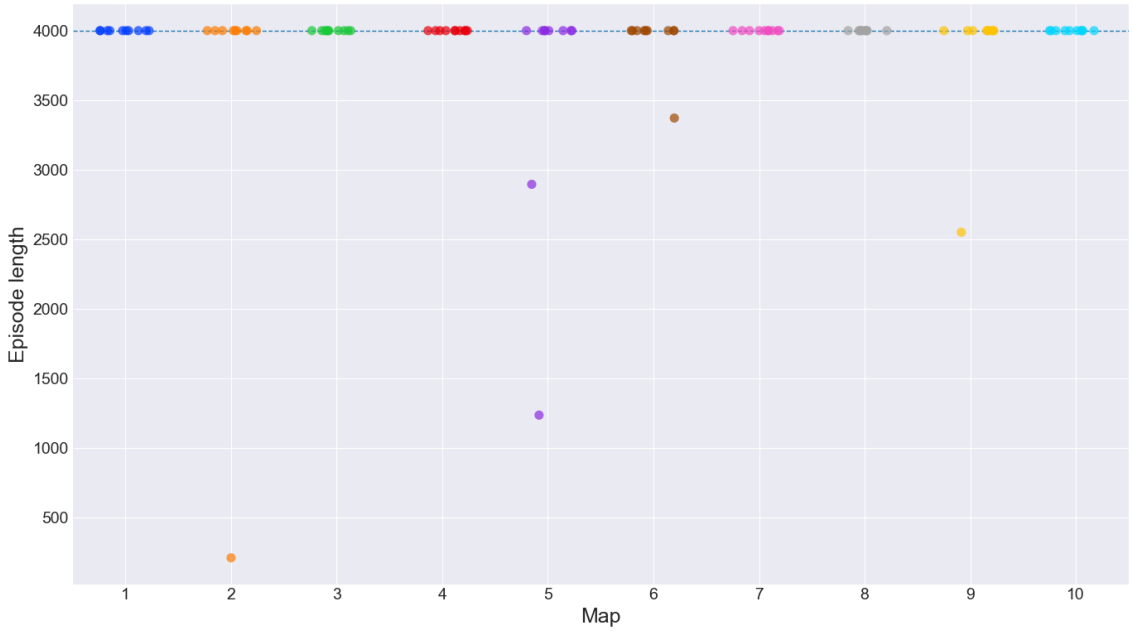

Figure 5.4: Out-of-distribution sustainability on satellite-derived terrain maps. The x-axis shows the map index, where each number corresponds to one real-world terrain patch generated from Google Earth imagery from Stelvio National Park. The y-axis shows sustainability, measured as the length of an evaluation episode in simulation time steps. Evaluation episodes were capped at 4000 time steps; therefore, a value of 4000 means that the simulation survived until the maximum evaluation length. Each dot represents one independent evaluation simulation of the trained agents on the corresponding map.

Resulting sustainability for different maps was compared using exact two-sided permutation tests based on binary success. A run was considered successful if the

episode length reached 4000 time steps. The Perlin noise map was used as the reference and compared separately against each of the 10 real world maps. The test statistic was the difference in win rate between the Perlin noise map and the comparison map. Holm correction was applied across the ten pairwise comparisons. Results can be seen in Table 5.4 below where we can see that there is no significant difference between the Perlin noise map and the real world maps.

Comparison	Other win rate	Difference	p-value	p-value*
Map 1	1.00	-0.20	0.474	1.000
Map 2	0.90	-0.10	1.000	1.000
Map 3	1.00	-0.20	0.474	1.000
Map 4	1.00	-0.20	0.474	1.000
Map 5	0.80	0.00	1.000	1.000
Map 6	0.90	-0.10	1.000	1.000
Map 7	1.00	-0.20	0.474	1.000
Map 8	1.00	-0.20	0.474	1.000
Map 9	0.90	-0.10	1.000	1.000
Map 10	1.00	-0.20	0.474	1.000

Table 5.4: Results of exact two-sided permutation tests comparing the Perlin noise map against each real world map using binary success as the outcome, where success was defined as reaching a value of 4000. Differences are reported as Perlin noise map win rate minus the comparison map win rate. * Holm adjusted p-value.

5.5 Simulation statistics

In this section we presnt resulting statistics that can be generated using a final model trained on 4M steps using homeostatic reward trainenv and PPO algorithm. Statistics were collected during a longer episode that lasted 10 000 steps.

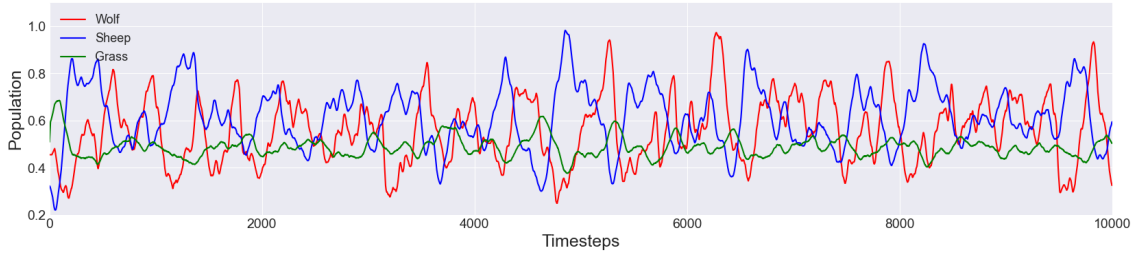

Figure 5.5: Dynamics of the simulated ecosystem over a 10,000 step evaluation episode. The curves show 30 step moving averages of the fractions of maximum wolf and sheep populations, and avergae of the fraction of available grass resources. The coupled fluctuations indicate predator prey interactions, where changes in prey abundance affect predator abundance, while grazing pressure causes grass availability to decrease and recover over time.

Population trajectories exhibit stable oscillatory behaviour, with predator and prey

populations fluctuating around bounded levels while remaining coupled to resource availability. Similarly grass resources recover and decline in response to grazing.

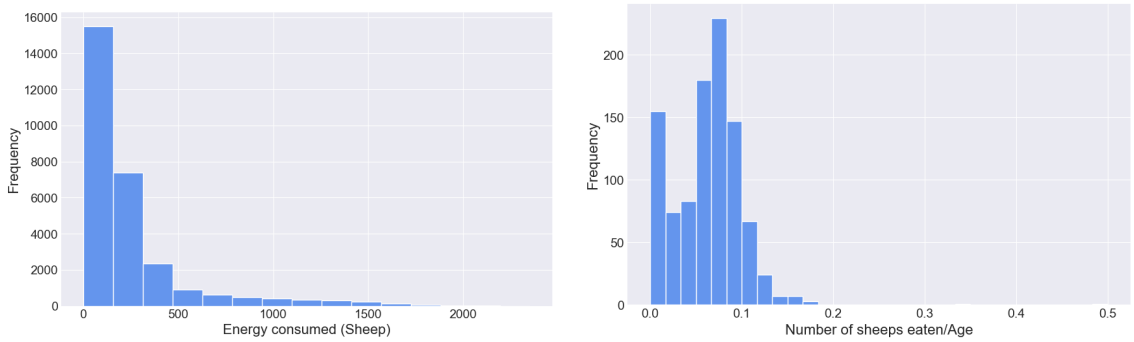

Figure 5.6: Individual level feeding and predation statistics collected during the 10,000 step simulation. The left histogram shows the total amount of energy consumed by individual sheep from grass resources, while the right histogram shows the number of sheep eaten by individual wolves. The distributions illustrate variation in resource intake and hunting success among agents.

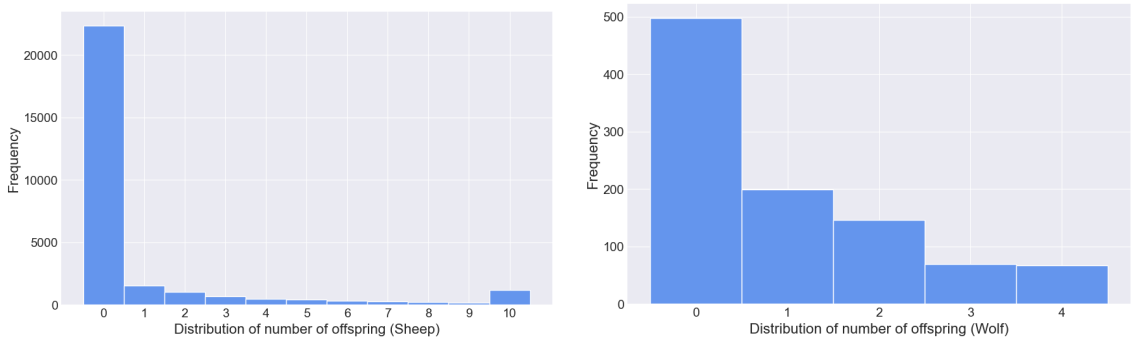

Figure 5.7: Distribution of reproduction of sheep and wolves during the 10,000 step simulation. The histograms show how many offspring were produced by individual agents. Most agents produced few or no offspring, while a smaller number reproduced more frequently, indicating unequal reproductive success within both populations.

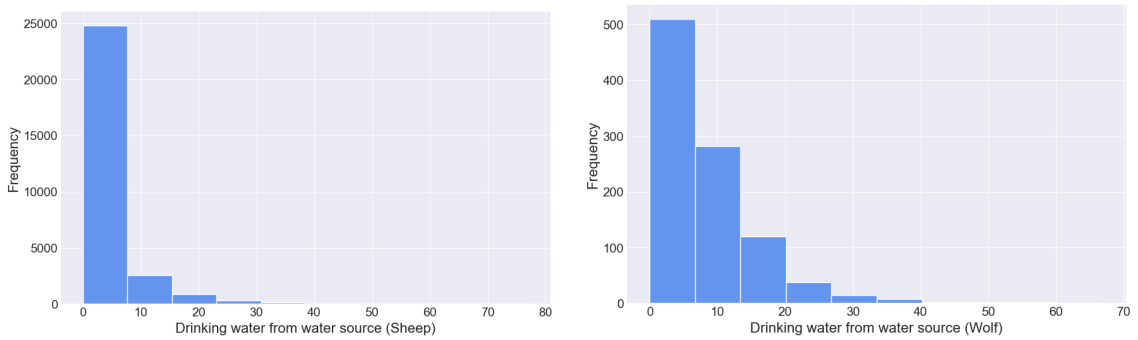

Figure 5.8: Distribution of drinking events from water sources for sheep and wolves during the 10,000 step simulation. The histograms show how often individual agents replenished water by reaching cells adjacent to water reservoirs. The results illustrate differences in water-use behaviour and reflect the need for agents to balance movement between grazing, hunting, and drinking locations.

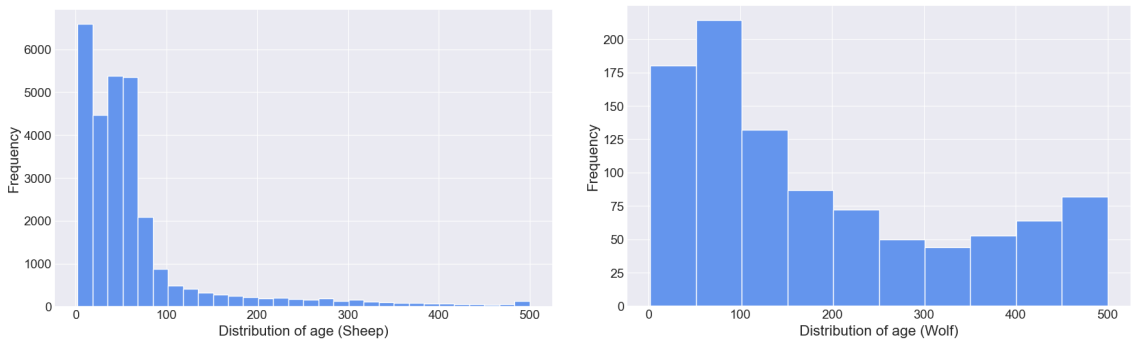

Figure 5.9: Distribution of final ages for sheep and wolves observed during the 10,000 step simulation. Final age corresponds to the age of an agent when it died. The distributions summarize survival outcomes for the two species and show how age-dependent mortality, predation, starvation, dehydration, and the maximum-age constraint shape population structure.

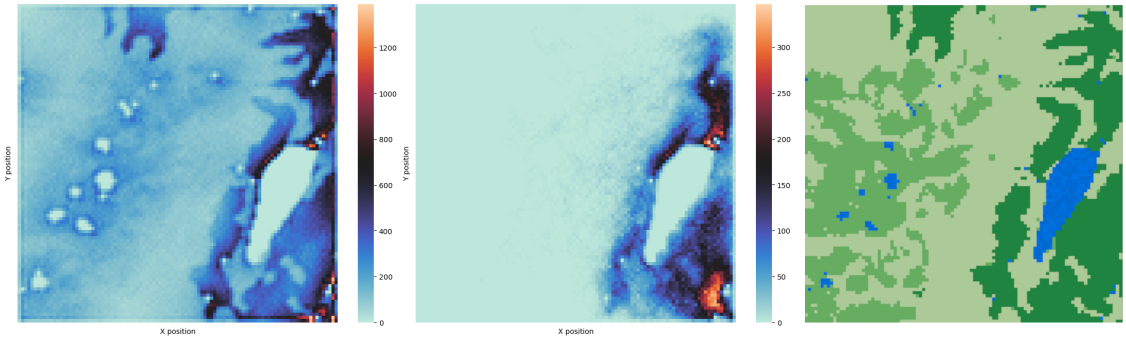

Figure 5.10: Spatial movement heatmaps for sheep and wolves together with the underlying resource map used in the simulation. The left heatmap shows the frequency of sheep positions, the centre heatmap shows the frequency of wolf positions, and the right panel shows the map with grass resources and water reservoirs. Darker regions in the heatmaps indicate areas visited more frequently. The patterns show that sheep concentrate near productive grazing areas and water sources, while wolves tend to occupy regions with high sheep density.

Finally, spatial heat maps reveal non uniform movement patterns, with prey population mainly occupying land cover types where grass regenerates the fastest (darkest green patches on the reference map figure 5.10). Additionally we can see that prey population prefers positions close to the water reservoirs (see darker shades of blue around water sources). Predators seems to prefer environment where the population of prey is the highest, and rarely visits left part of the map where the density of sheep population is small (see middle heat map figure 5.10).

6

Discussion

This chapter discusses the results presented in Chapter 5 in relation to the objectives of the thesis. The aim is to interpret the observed outcomes, highlight their implications for multi-agent reinforcement learning in ecosystem modeling, and outline key limitations and directions for future work.

6.1 Reward

The comparison between the survival reward and the homeostatic reward shows that adding internal state information to the reward function can make learning faster and help the agent survive better over time. The models trained with the homeostatic reward usually reached higher sustainability earlier during training. This suggests that rewarding agents for keeping their energy and water levels balanced helps them learn more stable and useful behaviors.

However, the permutation tests did not provide strong statistical evidence for difference between the two reward types. In the main analysis, where training speed was defined as the first checkpoint at which at least 3 out of 5 evaluation episodes reached an episode length of 4000 for two consecutive checkpoints, the difference between reward types was not statistically significant in the two-sided test. The same was true in the sensitivity analysis based on the mean episode length exceeding 3000 for two consecutive checkpoints. This suggests that although the wellness reward shows a descriptive advantage, the evidence for a consistent difference in training speed is limited in this experiment.

One possible explanation is that the survival reward, despite its simplicity, already provides a strong survival signal in this relatively constrained environment. As a result, agents can still learn viable survival strategies without explicitly modeling internal homeostasis. Also, the sample size in this experiment was relatively small, with only 5 trained models for each reward type. This limits the statistical power of the permutation tests and makes it harder to find clear differences, even when the general trends suggest that differences may exist. Still, the faster convergence seen with the wellness reward suggests that reward shaping inspired by biological ideas can improve sample efficiency. This is especially useful when training agents in more complex environments or in settings that require more computational resources.

6.2 Training environment

Training inside the simulated training environment leads to faster learning compared to the standard setup. This effect is expected, as the simulated environment avoids training on huge proportion of empty observations that arise when agents die. As a result, a greater fraction of training data corresponds to meaningful interactions between agents and their environment, making the learning process more data efficient.

However, an important limitation of this comparison that we need to keep in mind is that the training efficiency is measured in terms of the number of training steps rather than the actual computation time. In the simulated environment, more agents are alive at any given time, which increases the computational cost per step. This slight difference in computational efficiency wouldn't necessarily undermine the significance of the result but rather reduce the total effect. Thus the actual effect on computational time is less than the actual values presented in results section.

Another limitation concerns the ecological realism of the simulated environment. By maintaining a constant number of agents through respawning, the population density remains fixed throughout training. This results in a more densely populated environment than would naturally occur when agents are allowed to die permanently. In contrast, the standard environment allows population density to fluctuate over time, exposing agents to a wider range of ecological scenarios, including low-density states that may require different survival strategies.

Ideally, training would focus exclusively on observations from living agents while still allowing agents to die permanently, thereby preserving realistic population dynamics without introducing large regions of uninformative observation space. Such an approach could enable agents to learn behaviors that adapt to changing population densities while retaining the data efficiency benefits observed in the simulated environment. Developing training schemes that achieve this balance remains an important direction for future work.

6.3 Algorithm

No statistically significant performance differences were observed between the algorithms evaluated in this study when performance was measured using binary success, defined as reaching an episode length of 4000. PPO was compared separately against PPO2, TQC, TRPO, and the Handcoded controller using exact two-sided permutation tests, and none of these comparisons remained statistically significant after Holm correction. This suggests that, under the chosen experimental conditions, the algorithms achieved broadly comparable success rates after 4 million training steps.

It is important to note that this conclusion is conditional on both the fixed number of training steps used in the evaluation and the binary definition of success. While the results suggest that the algorithms perform similarly at 4 million training steps in terms of how often they reach the maximum episode length, differences may emerge if sustainability were evaluated across a wider range of training durations or using the full episode length rather than only a binary success criterion. Some

algorithms may converge faster or exhibit different stability properties earlier or later in training, which is not captured by the present analysis.
Another important limitation is the relatively small sample size, with only 10 runs evaluated for each algorithm. This reduces the statistical power of the permutation tests and makes it harder to detect differences even when descriptive differences in win rate are present. For example, PPO achieved a higher win rate than PPO2 and Handcoded, while TRPO achieved a slightly higher win rate than PPO, but these differences were not statistically significant after correction for multiple comparisons.
A further limitation is that the algorithms considered are not fully diverse in their learning strategies. PPO2, PPO, TQC, and TRPO are all reinforcement learning based approaches, and PPO, PPO2, and TRPO are particularly closely related. This reduces the breadth of algorithmic comparison and may partly explain why large differences were not observed. The inclusion of the Handcoded controller provides a useful non-learning baseline, but a broader comparison with more distinct learning algorithms could give a more complete picture.

6.4 Out of distribution

The out-of-distribution evaluation shows no significant difference in sustainability between Perlin noise–generated environments and environments derived from real-world satellite imagery. This suggests that the learned policies generalize reasonably well to new spatial resource configurations, at least in terms of overall survival.
Note that the real world maps used in these experiments don't account for the altitude changes and the distribution of resource in different real world environments might look completely different. For example an environment like swamp or desert might lead to completely different results. However training of the agents on worlds with different Perlin Noise parameters that are specifically designed for these different real world environments could be a solution.

6.5 Simulation statistics

The presented simulation statistics primarily serve to demonstrate the types of data that can be extracted from multi-agent reinforcement learning–based ecosystem simulations. Rather than being used for quantitative ecological inference, these statistics illustrate the analytical potential of the proposed framework. For example, the spatial distributions of animals visualized through heatmaps provide insight into how agents utilize the environment over time. Interesting thing to notice here is that the predators are hunting close to the water source which would mirror the natural behavior observed in nature where lions often hunt close to the water source [12].
Such information could be useful for studying migration patterns, both in response to resource depletion and to reproductive behaviour. With additional constraints and environmental factors such as seasonal changes or resource variation this type of analysis could be extended to investigate more complex movement dynamics. However, incorporating such mechanisms is beyond the scope of the current work and

represents a potential direction for future research. The population dynamics shown in the time-series plots exhibit clear cyclic behaviour, which is consistent with the oscillatory patterns predicted by the classical Lotka–Volterra model. This suggests that, despite the much more complex predator prey relation than in Lotka-Voltera model, the system still exhibits oscillations that could possibly be captured by a simpler model. Additional statistics, including distributions of age, offspring count, and resource consumption, further illustrate how behaviour on individual level result in patterns seen among whole population. In future studies, such statistics could be used to calibrate model parameters against real world data, thereby improving biological realism.

6.6 Sustainability

One important limitation of this thesis is how we measure performance. In this work, the main measure used is sustainability, which mainly tells us whether the simulated ecosystem survives for a long time. For example, a model is considered successful if the simulation reaches the maximum episode length. This is useful when comparing different reward types, training setups and algorithms, because a policy that quickly leads to extinction is clearly not useful for long term simulations. However, sustainability does not necessarily mean that the simulation is ecologically realistic. A model can produce stable predator-prey dynamics while still relying on simplified assumptions, unrealistic parameter values or behaviours that would not occur in a real ecosystem. At the same time, a real ecosystem is not always stable. Real ecosystems can collapse or go through large temporary changes due to changes in climate, resources, diseases or human intervention. Therefore, the results in this thesis should mainly be interpreted as showing that MARL can generate persistent predator-prey dynamics in the proposed artificial environment. They should not be interpreted as evidence that the model can predict real ecological responses.
The same limitation applies to qualitative patterns observed in the simulations. For example, the model produces oscillatory predator-prey dynamics and predators tend to spend more time in areas where prey density is high. These behaviours are interesting because they are similar to patterns observed in predator-prey systems. However, reproducing some expected patterns does not mean that all behaviours in the simulation are realistic. In order to make stronger claims about ecological realism, the model would need to be validated against real world data. This could include movement trajectories, population time series, predation rates, reproduction rates, age distributions or observed responses to environmental changes.

7
Conclusions

In this thesis we designed and implemented a reinforcement learning based framework for simulating predator-prey dynamics in a two dimensional grid based ecosystem. The main focus of the work was to investigate whether multi-agent reinforcement learning can be used to model more complex animal behaviour in an environment where agents need to interact with resources, avoid starvation and dehydration, and survive together with another species.

The ecosystem was modelled using a grid based representation with several different resources, including grass resources, water reservoirs, prey agents and predator agents. This allowed us to create an environment where prey agents are required to move between grazing fields and water reservoirs in order to survive. In addition, age dependent movement speed was introduced in order to model the vulnerability of young and old individuals. These mechanisms make the environment more complex than a simple predator-prey model, while still keeping it simple enough to train and evaluate reinforcement learning agents.

The results show that the proposed framework is able to produce stable long term simulations where both predator and prey populations can survive for extended periods of time. The population dynamics observed in the simulations show oscillatory behaviour, which is similar to the behaviour described by classical predator-prey models such as the Lotka–Volterra model. However, in contrast to such models, the framework developed in this thesis also makes it possible to study individual level behaviour, such as movement patterns, resource consumption, survival time and reproduction.

Different reward types were compared in order to investigate how reward design affects learning. The homeostatic reward showed better performance during earlier parts of training compared to the heartbeat reward. This suggests that including internal states such as energy and water directly in the reward function can help the agents learn useful behaviour faster. However, after longer training, the difference between the reward types becomes less clear, which indicates that the heartbeat reward can also be sufficient in this environment when enough training is provided.

Different training setups were also compared. The training environment where dead agents were respawned resulted in faster learning compared to the normal setup. This is most likely because the agents receive more useful observations during training, instead of many zero observations from dead agents. At the same time, this training setup is less biologically realistic, since real animals do not respawn after death. Therefore, this method should mainly be seen as a practical way to improve training efficiency rather than as a realistic model of ecosystem dynamics.

The comparison of different reinforcement learning algorithms showed that PPO,

TRPO and TQC all achieved relatively similar performance after 4 million training steps. This suggests that, for the environment considered in this thesis, the choice between these algorithms is not the most important factor once enough training has been performed. We also observed that a simpler PPO model with a smaller neural network could still achieve competitive performance, which indicates that the level of behavioural complexity required in this environment may not require a very large model.

The out of distribution experiments showed that agents trained on Perlin noise generated maps were also able to survive on maps generated from real world satellite images. No significant difference in sustainability was observed between the Perlin noise map and the real world maps used in the experiments. This suggests that the agents learned behaviours that generalize to new resource distributions. However, the real world maps used in this thesis are still simplified, since they do not include important factors such as altitude, seasonal changes, terrain difficulty or differences between real vegetation types.

Finally, the simulation statistics show that the framework can be used to collect different types of information about the simulated ecosystem. Examples include population trends, distributions of final age, number of offspring, drinking behaviour, food consumption and spatial heatmaps. These statistics demonstrate that the framework can be used not only to measure sustainability, but also to study how individual agent behaviour leads to population level patterns.

Overall, the results indicate that multi-agent reinforcement learning is a promising approach for modelling predator-prey ecosystems in a grid based environment. The framework developed in this thesis is still simplified, but it captures several important aspects of ecosystem dynamics, including resource dependence, predation, migration between water and food resources, reproduction and age based vulnerability. Future work could extend the model by including more realistic terrain, seasonal changes, additional species, more complex resource dynamics, separate neural networks for different animal types and calibration against real world ecological data.

Bibliography

[1] Alexandre Antonelli. *The Hidden Universe: Adventures in Biodiversity*. University of Chicago Press, 2022.

[2] Donald L. DeAngelis and Volker Grimm. "Individual-based models in ecology after four decades". In: *F1000Prime Reports* 6 (2014), p. 39. DOI: 10.12703/P6-39.

[3] Yaodong Yang et al. "A Study of AI Population Dynamics with Million-Agent Reinforcement Learning". In: *arXiv preprint* (2018). arXiv: 1803.06787.

[4] Jun Yamada, John Shawe-Taylor, and Zafeirios Fountas. "Evolution of a Complex Predator–Prey Ecosystem on Large-Scale Multi-Agent Deep Reinforcement Learning". In: *arXiv preprint* (2020). arXiv: 2003.07005.

[5] Peter Sunehag et al. "Reinforcement Learning Agents Acquire Flocking and Symbiotic Behaviour in Simulated Ecosystems". In: *Artificial Life Conference Proceedings*. Vol. 31. MIT Press, 2019, pp. 103–110.

[6] Alfred James Lotka. *Elements of Physical Biology*. Williams & Wilkins, 1925.

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

[8] John et al. "Trust Region Policy Optimization". In: *arXiv preprint: 1502.05477* (2015).

[9] John et al. "Proximal policy optimization algorithms". In: *arXiv preprint: 1707.06347* (2017).

[10] Ken Perlin. "An image synthesizer". In: *ACM Siggraph Computer Graphics* 19.3 (1985), pp. 287–296.

[11] Google LLC. *Satellite imagery*. Google Earth. 2026.

[12] Cornelius J. Louw, Sam M. Ferreira, and Jason P. Marshal. "Water dependence structures predation risk for large herbivores in insular protected areas". In: *Mammalian Biology* 102.5–6 (2022), pp. 1783–1792. DOI: 10.1007/s42991-022-00278-8.

.1 Appendix

.1.1 grid based ecosystem

We start by explaining implementation of the grid based ecosystem, which consists of $n \times m$ cells, where n is the number of rows and m is the number of columns. Each cell represents a discretized ecosystem and can represent arbitrary length depending on scale of ecosystem that we want to study. For example centimiters when studying microbial ecosystem or kilometers when studying a forest or a sea. The ecosystem is represented by a set of resources which are represented with matrices $R^k \in \mathbb{R}^{n*m}$ where R^k_{ij} is the amount of resource k in cell (i, j). R can represent variaty of resource. For example it can represent the distribution of certain type of plant over an area, a water source or location of animals. In addition the ecosystem is described by set of interactions between resource R^k which are rules for how resource values in each cell $R^k_{ij,t}$ changes over time $t \in T$. This can for example be a diffusion process where dynamic of R^k depends only on itself or even predator prey interaction where prey get's eaten if it happen to be in the same cell as predator. In this study we focus on modelling predator prey dynamics, however we note that this representation of ecosystem is very general and can be used to model complex interactions that occur in nature.

.1.2 Predator-prey ecosystem dynamics

In this section we describe how the predator-prey ecosystem used in this study is modelled with respect to grid based ecosystem described in previous section.
The ecosystem consists of six different types of resources. Three plant resources $R^p, p \in \{1, 2, 3\}$ which are food source for prey agents, a sheep resource R^s representing the prey, a wolf resource R^w representing the predator, and a water resource R^a. Plant resources have fixed initial position and do not spread to any other grid cells. We can think of them as three types of food source for prey population. These resources have a maximum capacity of $R^p_{i,j} \leq 1$ and grow linearly with time i.e $R^p_{ij,t+1} = min(R^p_{ij,t} + g_p, 1)$ where g_p is the growth rate of plant p. In addition, plant resources can be consumed by prey population which means that if sheep resource is present in the same cell then plant resource is reduced to 0. Thus the general formula for updating plant resource can be written as

$$R^p_{ij,t+1} = \begin{cases} min(R^p_{ij,t} + g_p, 1) & , R^s_{ij,t} = 0 \\ 0 & , R^s_{ij,t} = 1 \end{cases} \tag{.1}$$

Water resource R^a is a static resource which have fixed initial position and remains the same over the whole simulation $R^a_{ij,t+1} = R^a_{ij,t}$. Reason for this is that water resource in this context represents a lake or a river and is thus considered static.
Sheep and wolf resources follow more complex dynamics where each R^w_{ij} and R^s_{ij} take integer values and represent number of each animal species in particular grid cell. In this context each animal is considered an individual agent which can change resource matrix in three different ways.

Moving from cell i, j to cell k, l:

$$R^{s/w}_{ij,t+1} = R^{s/w}_{ij,t} - 1, R^{s/w}_{kl,t+1} = R^{s/w}_{kl,t} + 1 \tag{.2}$$

Dying while being at cell i, j:

$$R^{s/w}_{ij,t+1} = R^{s/w}_{ij,t} - 1 \tag{.3}$$

Or getting an offspring while being at cell i, j:

$$R^{s/w}_{ij,t+1} = R^{s/w}_{ij,t} + 1 \tag{.4}$$

Where each of three scenerios are based on agent mechanics described in next section.

.2 Agent mechanics

We have two types of agents, wolf(predator) and sheep(prey). All agents have associated three values, energy $E \in (0, E_{max})$, water $W \in (0, W_{max})$ and age $A \in (0, A_{max})$. If any of $E = 0, W = 0$ or $A = A_{max}$ occurs then agent dies. In order to survive agents need to move within the ecosystem to find food and water. Both E and W decreases with one each time step and increase with energy or water found in resources at agents current cell. In addition W can be filled to W_{max} if agent is located in a cell adjacent to water resource R^a. Which can be described as follows for agent in cell i, j and food resource R^{food} where $food = s$(sheep) for wolf and $food = p$(plant) for sheep

$$E_{t+1} = max(min(E_t - 1 + food_E * R^{food}_{ij,t}, E_{max}), 0) \tag{.5}$$

$$W_{t+1} = \begin{cases} max(min(W_t - 1 + food_W * R^{food}_{ij,t}, W_{max}), 0), & if \sum_{k=i-1}^{i+1} \sum_{l=j-1}^{j+1} R^a_{kl} = 0 \\ W_{max} & if \sum_{k=i-1}^{i+1} \sum_{l=j-1}^{j+1} R^a_{kl} > 0 \end{cases} \tag{.6}$$

In addition animals can drown which means that if an agent moves from cell i, j to cell k, l occupied by water resource then agent dies

$$R^{s/w}_{ij,t+1} = R^{s/w}_{ij,t} - 1, R^{s/w}_{kl,t+1} = 0, if R^a_{kl,t} = 1 \tag{.7}$$

Sheep agents can also die from predation by wolf which means that if sheep agent moves from cell i, j to cell k, l where wolf agent is present then

$$R^{s}_{ij,t+1} = R^{s}_{ij,t} - 1, R^{s}_{kl,t+1} = 0, if R^{w}_{kl,t} > 0 \tag{.8}$$

.3 RL implementation

All agents use the same neural network to choose action with one of the parameters in observation space telling which animal type it is(wolf/sheep). Action space is continuous with following components $a_t \in (x, y, v), 0 \leq x, y, v \leq 1$, where x and y are components of the direction vector and v is velocity. Observation space consists of several

components that differ between animal types O_t = (vision, internal parameters, smell). Vision consists of resource values $R_{kl} : k \in (i - r, i + r), l \in (j - r, j + r)$ including all resources where $r = 1$ for sheep and $r = 2$ for wolf. Internal parameters consists of age A, water W, energy E, position within the grid, last action a_{t-1} and animal type. smell...

.3.1 Reward types

.3.2 Different training setups

Variable	**HR (95% CI)**	**p-value**
Training time = 800 000	0.01447 (0.00451–0.04644)	<0.005
Training time = 1 200 000	0.00463 (0.00139–0.01534)	<0.005
Training time = 1 600 000	0.00208 (0.00062–0.00702)	<0.005
Training time = 2 000 000	0.00111 (0.00032–0.00385)	<0.005
Training time = 2 400 000	0.00048 (0.00013–0.00179)	<0.005
Training time = 2 800 000	0.00030 (0.00008–0.00117)	<0.005
Training time = 3 200 000	0.00028 (0.00007–0.00110)	<0.005
Training time = 3 600 000	0.00029 (0.00007–0.00117)	<0.005
Training time = 4 000 000	0.00040 (0.00011 –0.00152)	<0.005
Homeostatic reward	1.07768 (0.61540–1.88721)	0.33
Homeostatic reward*Training time = 800 000	0.55240 (0.25031–1.21910)	0.142
Homeostatic reward*Training time = 1 200 000	0.43460 (0.19372–0.97500)	0.043
Homeostatic reward*Training time = 1 600 000	0.30398 (0.12745–0.72498)	0.007
Homeostatic reward*Training time = 2 000 000	0.15775 (0.05049–0.49284)	<0.005
Homeostatic reward*Training time = 2 400 000	0.30861 (0.08513–1.11880)	0.074
Homeostatic reward*Training time = 2 800 000	0.11023 (0.01280–0.94904)	0.045
Homeostatic reward*Training time = 3 200 000	0.51345 (0.13307–1.98114)	0.333
Homeostatic reward*Training time = 3 600 000	0.34850 (0.08061–1.50662)	0.158
Homeostatic reward*Training time = 4 000 000	0.16368 (0.03202–0.83668)	0.030

Table .1: Cox regression comparing different reward types

Variable	**HR (95% CI)**	**p-value**
Training time = 0.8×10^6	0.0450 (0.0249–0.0814)	<0.005
Training time = 1.2×10^6	0.0083 (0.0042–0.0162)	<0.005
Training time = 1.6×10^6	0.0041 (0.0021–0.0083)	<0.005
Training time = 2.0×10^6	0.0019 (0.0009–0.0039)	<0.005
Training time = 2.4×10^6	0.0017 (0.0008–0.0036)	<0.005
Training time = 2.8×10^6	0.0009 (0.0004–0.0020)	<0.005
Training time = 3.2×10^6	0.0010 (0.0005–0.0021)	<0.005
Training time = 3.6×10^6	0.0009 (0.0004–0.0020)	<0.005
Training time = 4.0×10^6	0.0007 (0.0003–0.0015)	<0.005
Trainenv	0.167 (0.1296–0.2153)	<0.005

Table .2: Cox regression comparing different training setups

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY