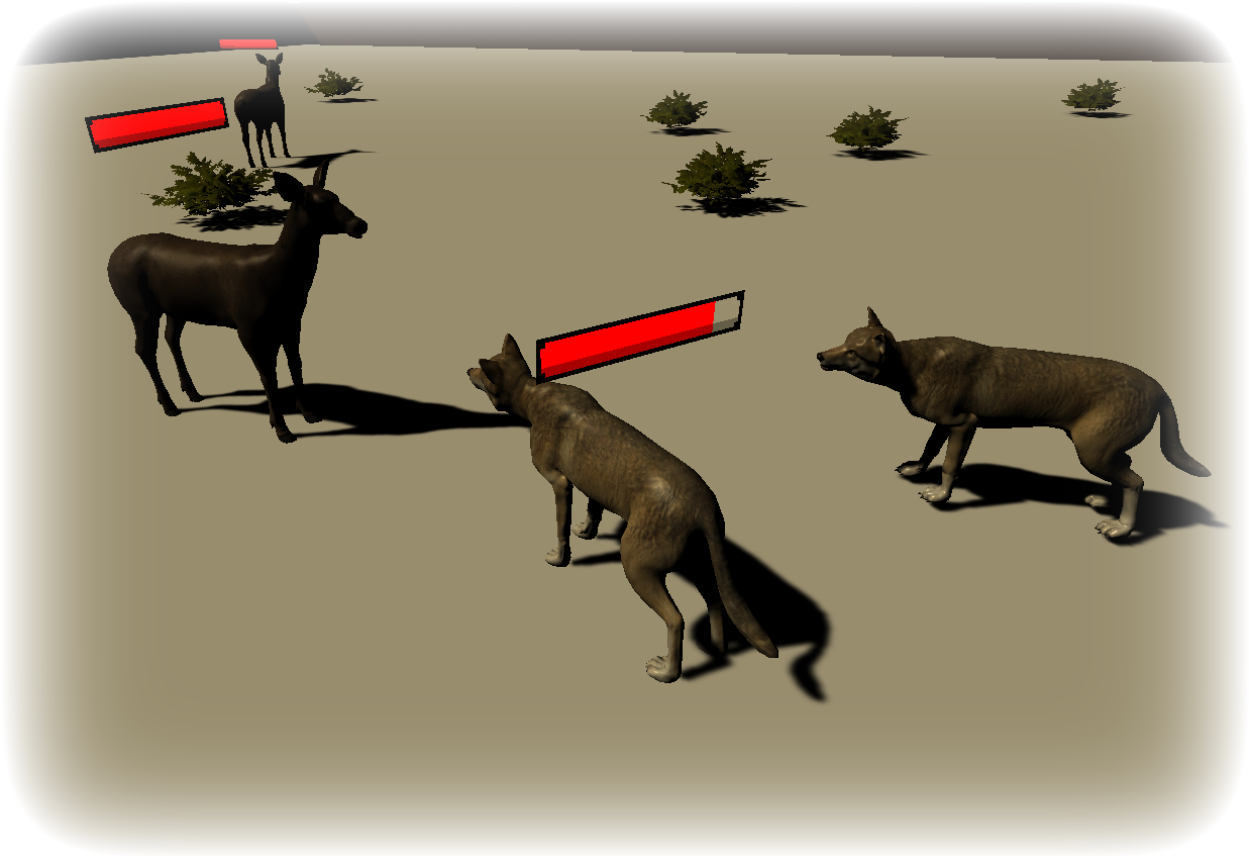

Multi-Agent Deep Reinforcement Learning in a Three-Species Predator-Prey Ecosystem

Using Multi-Agent Deep Reinforcement Learning to train agents in a three-species predator-prey simulated ecosystem and exploring the dynamics with respect to the Lotka-Volterra model of population dynamics.

Master's thesis in Data Science and Artificial Intelligence

TOBIAS KARLSSON

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2021

Master's thesis 2021

Multi-Agent Deep Reinforcement Learning in a Three-Species Predator-Prey Ecosystem

Using Multi-Agent Deep Reinforcement Learning to train agents in a three-species predator-prey simulated ecosystem and exploring the dynamics with respect to the Lotka-Volterra model of population dynamics.

TOBIAS KARLSSON

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2021

A Three Species predator prey system with Multi-Agent Reinforcement Learning
Using Multi-Agent Deep Reinforcement Learning to train agents in a three-species predator-prey simulated ecosystem and exploring the dynamics with respect to the Lotka-Volterra model of population dynamics.
TOBIAS KARLSSON

Supervisor: Claes Strannegård, Department of Computer Science and Engineering
Examiner: Marina Axelson-Fisk, Department of Mathematical Sciences

Master's Thesis 2021
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Three species predator prey ecosystem with grass, prey (deer) and predators (wolves).

Typeset in LaTeX
Gothenburg, Sweden 2021

A Three Species predator prey system with Multi-Agent Reinforcement Learning
Using Multi-Agent Deep Reinforcement Learning to train agents in a three-species predator-prey simulated ecosystem and exploring the dynamics with respect to the Lotka-Volterra model of population dynamics.
Tobias Karlsson
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

In computational biology, population dynamics in simulated ecosystems is one important topic. Standard mathematical tools of population dynamics such as systems of differential equations are typically incapable of accounting for a variety of important attributes, such as the intelligent and adaptive behavior of individual agents in complex environments. Thus, they are often insufficient to simulate dynamics in real-world ecosystems. In this thesis, a three-species predator-prey simulated ecosystem was implemented in the Unity game engine. Agents in the ecosystem were trained through multi-agent reinforcement learning. The population dynamics were then analysed with respect to the Lotka-Volterra predator-prey equations which are described by several parameters and assumptions regarding the responses of the parameters to changing population densities. The responses of the parameters were estimated through simulation experiments. It was found that the population dynamics of an ecosystem with trained agents exhibited Lotka-Volterra cycles where a random policy agent ecosystem failed to do so. Further, it was shown that the observed responses of the parameters did not fulfill the Lotka-Volterra assumptions, but rather showed properties that could be argued to be more realistic.
For the reinforcement learning, a reward system was introduced as the happiness network, which incorporated both the external and internal state of the animat, inspired by behavioral science of real-world animals. This reward system was shown to perform better than a simple reward system with a positive reward for eating food and a negative for dying, in some environments and was argued to have benefits in more complex ecosystems.

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

Acknowledgements

I would first and foremost like to thank my supervisor Claes Strannegård for giving me the opportunity to work with this project under a large influence over the direction of research and giving me consistent feedback on the work.
Secondly, a thank to Marina Axelson-Fisk for taking on the role as my examiner.
Thirdly, I would like to thank Marcus Hilding Södergren for his initial work with the implementation of the ecosystem in Unity and his help on getting me started with the project. Another big thanks to Birger Kleve, Hans Glimmerfors, Pietro Ferrari and Victor Skoglund which also worked on their theses within simulated ecosystems under the supervision of Claes, and where we through weekly meetings could give each other feedback and help.
Lastly, thanks to Johan Nicander and Rikard Eriksson for useful feedback on the thesis writing and my brother Andreas Karlsson for helping out with running simulations.

Tobias Karlsson, Rodellar, June 2021

Contents

List of Figures

List of Tables

1

Introduction

An animat was defined by Stuart Wilson as an artificial animal including a model of homeostasis [1]. Homeostasis denotes the regulation of internal variables, such as body temperature, pH, sodium level, potassium level, calcium level and blood sugar level. He proposed the idea of creating artificial intelligence (AI) by modeling animal behavior, the *animat path to AI*. The most prominent examples of agents with general intelligence are humans which are special kind of animals, making the animat path to AI a naturally interesting idea in the objective to develop artificial general intelligence (AGI).

Reinforcement Learning (RL) is an area of machine learning where the behaviors of agents are learnt through interaction with an environment. The area of RL has seen massive research efforts in the last decade with outstanding results in several different domains.

In simulated ecosystems, conventional mathematical tools are incapable of accounting for a variety of important attributes of the system, such as the intelligent and adaptive behavior of individual agents in complex environments. Despite recent attention in RL with its clear connection to modeling animal behavior, few studies have been made in the domain of ecosystem modelling.

In this thesis, we explore whether real-world dynamical properties rooted in the theory of the Lotka-Volterra (LV) predator-prey equations are being exhibited in simulated ecosystems with animats learning through multi agent RL. More precisely, we investigate the population dynamics in a three-species predator-prey system. The dynamics of simulated predator-prey systems are governed by a combination of the simulation parameters, such as the environmental and agent configurations, and the decision making abilities. By estimating the LV parameter responses to different environment configurations, we suggest how the simulated ecosystem may show more realistic dynamics by challenging the assumptions made by the classical LV model. Further, a reward system inspired by real life animals is incorporated. This reward system, referred to as the *happiness network*, combines external stimuli with internal homeostatic regulation.

By analysing the dynamics of an agent based simulated ecosystem with agents trained by RL with conventional mathematical models of computational biology, we hope to find similarities and differences between the two approaches which may be useful in the process of taking ecosystem simulations to more realistic levels.

1.1 Background and related work

A vast amount of work has been put into analyzing the LV equations and their different variants, both through the lens of mathematical analysis and numerical simulations. The LV models are based on several assumptions and are deterministic. One of the assumptions are infinite population sizes. The fact that real world ecosystems consist of finite populations with decision making of animals makes them non-deterministic. This stochasticity is better captured in an individual based model. Individual based models have been pointed out to play an increasingly important role in questions posed by complex ecological systems [2], where they also paraphrased the well-known paper by Dobzhansky [3] "Nothing makes sense in ecology except in the light of the individual". RL provides an intuitive framework to train and simulate individual based agents with complex decision making systems but well defined observation and action spaces.
Some researchers have been exploring the dynamics in two-species predator-prey systems with agents trained through RL. In [4], the authors showed that by using multi-agent reinforcement learning (MARL), they obtained cyclic dynamics. However, they used a random policy for the prey, which is not very realistic. As a further step, [5], investigated several more complex predator-prey simulated ecosystem variants where the prey were also trained agents. In their simplest environment, they showed that with an appropriate choice of parameters, they could obtain stable limit cycles as predicted by the LV equations. When a mating mechanism was introduced, where new animats were born due to inter-species meetings, they showed that they could obtain quasi-cycles as expected from a stochastic predator-prey system where the birth and death of individuals can be described as a stochastic process. Their environment is an unbounded grid world, where the prey and predators can observe a limited sized square around them, from a birds-eye-view[1]. The prey observed the energy levels of the predators. At each timestep, the agents could move to a nearby grid, and if a prey is within the hunting area of the predator, it got eaten up. The authors used deep recurrent Q-learning [6], to take into account the fact that the environment is a Partially Observable Markov Decision Process (POMDP), which is described in section 2.1.2.
Research with MARL in predator-prey systems have been done on other aspects as well. In [7], it was investigated whether group behaviors such as flocking and cross-species symbiotic partnerships, which are often observed in nature, could also be observed in a multi-species predator-prey system with agents trained through MARL. This is, to my knowledge, the only time before this thesis that a three-species predator-prey system have been explored with RL. Their simulations were run in two different environments, a 2D grid world with agents in the magnitude of thousands and a 3D game engine world with tens of agents. Each agent had a radius which it could observe things within. They observe their own velocities as well as the position of all other agents and their velocities. A policy network was trained for each species and their results showed that there is both intra and cross species coordination emerging, even with no explicit reward encouragement for these to occur, but simply independently incentivized self interested agents.

[1] Viewing the surrounding as though it was a bird, watching from above.

In [8], they present a method to create a mapping between the parameters of a simulated predator-prey system and the parameters of the LV model. This mapping may be useful to compare simulation results to that of the deterministic LV model. All the research for RL predator-prey systems mentioned above use a reward which considers the animals response to external changes, such as giving a positive reward for eating and a negative for dying. In [9], the authors suggest that this is only one part of the reward system of animals. They suggest combining this with the internal changes, referred to as homeostatic regulation, where each animal has several homeostatic variables which it wants to keep at their optimal values. In this thesis, we incorporate a reward system inspired by this, called the happiness network, where the reward at each step is the difference in happiness. The happiness is a combination of internal homeostatic regulation and external stimuli, such as the smell of food. Further, in contrast to the referenced predator-prey MARL systems mentioned above, we use observations resembling vision and smell from a first person perspective to make it more realistic.
An increasingly popular framework to train RL agents is to use the Unity game engine and the machine learning toolkit ML-agents [2]. This makes it easier to build visually rich environments to train agents with RL. Visually rich environments enables qualitative evaluation of the agents to a larger extent. One sidegoal of this thesis is to be part of providing a framework in Unity for future research of simulated ecosystems.

1.2 Aims and limitations

In this thesis, we begin by exploring whether we can obtain intelligent behaviors in a simulated predator-prey system with agents trained in a MARL framework and the Proximal Policy Optimization (PPO) RL algorithm. The intelligence is explored by investigating the survival capabilities of our agents compared with random policy agents. Further, we explore whether the simulated predator-prey systems exhibit cyclic population dynamics similar to that described by the LV equations. The LV model is based on several assumptions, such as the parameter responses to changing population densities. Through simulation experiments similar to that of [8], the parameter responses in the simulated ecosystem are estimated and compared with the assumptions.
The capability of the happiness network is evaluated by comparing the lifetime of agents trained with it in comparison to a more classic external reward system.
This thesis will not compare the performance of different RL algorithms. The reason for this, is mainly due to the computational cost of training a large multi agent system. This is also the reason for not experimenting with different hyperparameters for the training process. Rather, these are selected in accordance with common recommendations[3].

[2] `https://github.com/Unity-Technologies/ml-agents`
[3] The default configuration of the PPO implementation in stable-baselines3.

2
Theory

This section begins with discussing the idea of RL from a broader perspective and where it fits in among other machine learning paradigms. It is followed by a presentation of the Markov Decision Process formulation of a RL problem, which will lead to the definition of a Partially Observable Markov Decision Process. Then some main ideas of Multi Agent Reinforcement Learning (MARL) and the idea of centralized learning, decentralized execution is introduced. This will be followed by going through the inner workings of the Proximal Policy Optimization algorithm that is utilized in this thesis. After that, the Lotka-Volterra three-species predator prey model is presented. By the end of the section one can find a presentation of how the ecosystems are formalized in order to be described more precisely.

2.1 Reinforcement Learning

RL is one of the three machine learning paradigms. In RL, there is an agent which we want to take actions in such a way that some goal or goals are fulfilled. The agent observes the environment through some senses and takes an action that will maximize its cumulative reward. The rewards that is given to the agent is constructed to try to make the agent behave in a way that satisfy our goals, similar to how a pet trainer gives candy to its pet when it acts in wished for manner.
In supervised learning, there is a set of examples grouped with a label provided by an external supervisor. This label is the correct "action" given the corresponding example. The task of learning in this setting then is to extrapolate from these examples such that the actions are correct when new examples are presented. However, when learning through interaction, it has limitations due to the impracticalities associated with obtaining this training set of desired behavior that covers the whole observation space of the agent.
Using usupervised learning, we are not limited by an external supervisor. However, in this task, the goal is usually to try to find structure hidden in this unlabeled data. RL is about maximizing cumulative reward rather than finding the hidden structure, even though this second task might be important in order to achieve the first. Thus, RL is considered its own machine learning paradigm. The exploration-exploitation trade-off is a inherent challenge of RL. The agent, which wants to collect as much reward as possible needs to both exploit taking actions that it has discovered to be good, but also explore actions which it is unsure about, otherwise it might miss out on the very best actions. On the other hand, if it always selects to try out new things, it misses out on a lot of reward. This is the trade-off and every RL algorithm

needs to consider how to deal with it.
RL algorithms are usually one of two types, model-based or model-free, even though hybrids of the two which utilizes the strengths of both algorithms are beginning to emerge lately [10]. Briefly, in model-based methods, the agent selects its action by using its model predictions of next state and reward in order to calculate optimal actions. In model-free, the agent does not try to predict future states, rather it samples from its experiences. Selecting actions that has historically been good given a particular state.
One of the things that makes RL exciting is that it resembles how humans and animals learn and many algorithms are inspired by biological learning systems [11]. Thus, in the objective of achieving more general artificial intelligence, RL seems to be a logical and promising approach [12].

2.1.1 The Markov decision problem formulation

A RL problem that satisfies the Markov property is called a Markov decision process (MDP). Given a set of possible outcomes Ω for a random variable X, the Markov property is satisfied whenever

$$\Pr(X_n = x_n \mid X_{n-1}, \ldots, X_0 = x_0) = \Pr(X_n = x_n \mid X_{n-1} = x_{n-1}) \tag{2.1}$$

[13]. This assumption is used for many RL problems since it provides a clean mathematical framework and are often approximately true. A MDP is a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathbb{P}_a, \mathbb{R}_a)$ with

$$\mathbb{P}_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a) \tag{2.2}$$

being the probability that action a in state s at time t will take us to state s' at time $t + 1$.

$$\mathbb{R}_a(s, s') = \mathbb{E}[\Pr(r_{t+1} \mid s_{t+1} = s', s_t = s, a_t = a)] \tag{2.3}$$

being the expected reward r at time $t + 1$ after transitioning from s to s' through action a at time t [14]. A finite part of a MDP depicted as a decision network can be seen in fig. 2.1. How good is a particular state? This question is very important in RL and particularly for value-based methods, such as the classic Q-learning method [15]. The value function is the expected future reward by following a policy π. A policy is a probability distribution over every action in a particular state. The value of a state s under a policy π is denoted by $V_\pi(s)$ and is defined as

$$V_\pi(s) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k \mathbb{R}_{a_t}(s_{t+k}, s_{t+k+1})] \tag{2.4}$$

and we call it the state-value function for policy π. γ is the discount factor satisfying $0 \leq \gamma \leq 1$ which gives rewards in the near future larger weights. Similar to the state-value function, the action-value function for policy π is defined as

$$Q_\pi(s, a) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k \mathbb{R}_{a_t}(s_t, s_{t+1}) \mid a_t = a] \tag{2.5}$$

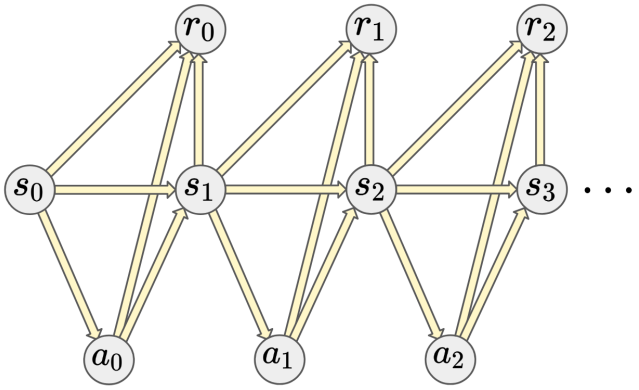

Figure 2.1: A finite part of a MDP depicted as a decision network.

which is the expected reward obtained by taking action a at time t and thereafter following the policy π [16].
Policy-based methods tries to improve the policy directly. We search for better policies in a policy space, which is usually approximated through some function approximator, such as neural networks.

$$\pi(\cdot \mid s) \approx \pi_\theta(\cdot \mid s)$$

By parameterizing the function, gradient methods can be applied to take a step in the direction which improves the long-term reward of the policy. This is called a policy gradient method. In this thesis, Proximal Policy Optimization (PPO) is used an described in section 2.1.6, which is a policy-based method.

2.1.2 Partially Observable Markov Decision Process

A generalisation of a MDP is a Partially Observable MDP (POMDP). A POMDP is a 6-tuple $(\mathcal{S}, \mathcal{A}, \mathbb{P}_a, \mathbb{R}_a, \Omega, \mathbb{O}_a)$. Where, the 4-tuple from an MDP is extended with Ω as the set of possible observations and $\mathbb{O}(o \mid s', a)$ as the conditional observation probability given the state s' and action a.

$$\mathbb{O}_a(o, s') = \Pr(o_{t+1} = o \mid s_{t+1} = s', a_t = a) \tag{2.6}$$

In a POMDP, The dynamics of the system is governed by a MDP, but the complete state of the system can not be observed by an agent. At each timestep, the environment is in some state $s \in \mathcal{S}$ and the agent takes an action $a \in \mathcal{A}$. Now the environment transitions to a new state s' with probability $P_a(s', s)$. The agent now makes a new observation $o \in \Omega$ of the updated environment, drawn from the distribution $\mathbb{O}_a(o, s')$. A finite part of a POMDP depicted as a decision network can be seen in fig. 2.2.

2.1.3 Multi-agent reinforcement learning

Many RL applications involve the participation of more than one agent, thus falling under the domain of Multi-Agent Reinforcement Learning (MARL). Even though

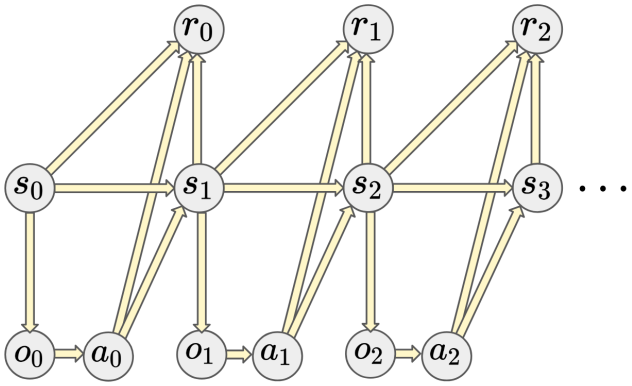

Figure 2.2: A finite part of a POMDP depicted as a decision network.

there are many successful empirical results, theoretical foundations are lacking, as concluded in [17]. MARL problems usually have a number of common challenges. The first being the fact that the objectives of the agents are not necessarily aligned, making the learning goals multi-dimensional. Secondly, the environment of an agent is non-stationary as all agents concurrently changes their policies in order to their individual interests. Additionally, the joint action space grows exponentially with the number of agents, possibly causing issues when it comes to scalability, referred to as the combinatorial nature of MARL in [18]. Finally, the structure of information in MARL is more complicated as the individual agents has limited access to others observations, which may lead to sub-optimal local policies. All these issues have not seen a lack of effort in trying to be addressed. However, much is yet to be conveyed when it comes to rigorous theoretical analyses of MARL.

2.1.4 Centralized learning, decentralized execution

Probably the most common learning scheme in MARL, is to use centralized learning with decentralized execution. This is when multiple agents observe and acts in an environment without explicit communication in between each other, no information is explicitly shared between them. But, the agents have a shared policy which is trained collectively. This can be put in contrast with a decentralized setting with networked agents, where agents share some information between them but there is no central controller, or a fully decentralized setting, where there is no central controller nor any information sharing between agents. These three options are illustrated from left to right in fig. 2.3. One of the reasons for using centralized learning when the agents are equal, is to speed up the training process. During each model update, a larger number of experience trajectories are sent to the model, compared to the option of assigning one individual model per agent. However, if the agents are non-equal, they can not share a brain.

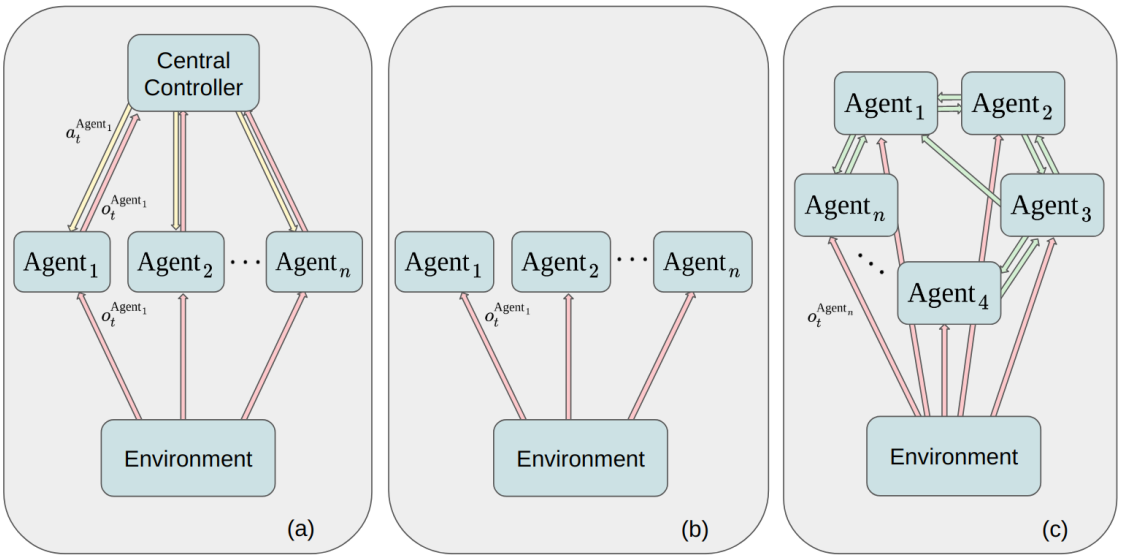

Figure 2.3: Three different information structures of MARL. (a) Centralized learning with decentralized execution. (b) Fully decentralized setting. (c) Decentralized setting with networked agents.

2.1.5 Decentralized POMDP

A class of challenging, yet very common, MARL problems are those where the environment is only partially observable to an agent (POMDPs). Theoretical analysis of these settings are yet in its infancy, compared with the MDP setting [17]. When the centralized learning, decentralized execution scheme is used in an POMDP, it is referred to as a Dec-POMDP [19]. A Dec-POMDP is formally defined as a 9-tuple $(\mathcal{S}, \mathcal{A}, \mathbb{P}_a, \mathbb{R}_a, \Omega, \mathbb{O}_a, \mathcal{D}, h, b_0)$. Here, we have further extended the POMDP defined in section 2.1.2 with the set of agents $\mathcal{D}$, b_0 and h. Here, b_0 is the initial state distribution of the environment and h the horizon of the problem. The horizon defines how many steps ahead that an agent tries to maximize its expected cumulative reward when deciding which action to take. All the sets $\mathcal{S}, \mathcal{A}$ and Ω are the joint sets of the agents. In a Dec-POMDP, the key idea is that the local observations of all agents are sent to a central controller, which returns a joint reward. The policy might or might not be shared among the agents. This way, the Dec-POMDP setting makes it possible to train a common policy through the local observations of the agents. The shared policy is possible since all the agents which share the central controller are equal.

2.1.6 Proximal Policy Optimisation

Compared with value-based methods, policy-based methods have better convergence guarantees [20] [21]. Proximal policy optimisation (PPO) [22] is a state-of-the-art policy-based RL algorithm which solves the convergence problem of some policy gradient methods. It relaxes a hard constraint on the second-order derivative matrix and imposes a penalty in the objective function. This way one can use a first-order optimizer like gradient descent which is less computationally demanding. Given that

we have parameterized a policy function π_θ and a value function V_θ with a shared parameter θ. And with $\hat{A}_t$ as the estimated advantage function at time t of

$$A_t(s_t, a_t) = Q(s_t, a_t) - V(s_t). \tag{2.7}$$

and a probability ratio of two successive policies

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}. \tag{2.8}$$

The surrogate objective function used in PPO with a clipped objective, is given by

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{2.9}$$

with ϵ as a hyperparameter. Thus, if

$$r_t(\theta) \notin [1-\epsilon, 1+\epsilon] \tag{2.10}$$

the estimated advantage function will be clipped. This way the incentive is removed to change a policy to much. The resulting objective function will be the lower (to obtain a pessimistic bound) of the clipped and unclipped. Using this minimisation, we only select the change in probability ratio when it actually makes the objective worse. When using a neural network architecutre, where the policy and value functions shares parameter θ, the authors recommend adding a value function error term to the surrogate objective, so that we also try to improve the approximation of the value function. Further, they recommend adding an entropy bonus, to encourage exploration. This way, we end up with the following objective function,

$$L(\theta) = \hat{\mathbb{E}}_t[L^{\text{CLIP}}(\theta) - c_1(V_\theta(s_t) - V_t^{\text{targ}})^2 + c_2 S[\pi_\theta](s_t)] \tag{2.11}$$

with c_1, c_2 being coefficients, S an entropy bonus and V_t^{targ} as the target value function given by

$$V_t^{\text{targ}} = r_t + V_\theta(s_{t+1}) \tag{2.12}$$

Now that we know the loss function being used, what does the training algorithm look like? Each iteration, all N actors collects T timesteps of data, then we use this NT timesteps of data to find the best parameter θ by maximizing the surrogate objective function $L(\theta)$ through Steepest Gradient Ascent or Adam Optimizer, for K epochs. Next iteration we will use this better parameter θ for the parameterized policy π_θ.

Algorithm 1: PPO, Actor-Critic style

Input: Initial policy π_{θ_0}
for *i = 1, 2, ...* **do**
  for *actor = 1, 2, ..., N* **do**
    Run policy $\pi_{\theta_{\text{old}}}$ for T steps;
    Calculate advantage estimates from the trajectories $\hat{A}_1, \ldots, \hat{A}_T$;
  end
  Optimize the surrogate objective function $L(\theta)$ with K epochs and a batch size of NT;
  $\theta_{\text{old}} \leftarrow \theta$;
end

In this algorithm, the advantage estimate can not look further than these T timesteps. Thus, the advantage estimate at time t is given by the difference of the value function at time t, with how much discounted reward we actually obtained in the interval $[t, T]$ plus the discounted value at time T of the state we ended up in.

$$\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t-1} r_{T-1} + \gamma^{T-t} V(s_T) \tag{2.13}$$

2.2 Lotka-Volterra Predator-Prey equations

The Lotka-Volterra predator-prey equations were independently proposed by Alfred J. Lotka in 1910 [23] and Vito Volterra in 1926 [24]. The equations are a pair of first order non-linear differential equations which describes the population dynamics of predator-prey systems. Since then, the model has been extended in various forms to in different ways consider the assumptions of the model which is often not realizable in real world predator prey systems. The model has also been extended to consider more than two species [25]. We will consider a three-species model consisting of a grass-prey-predator food chain. We put a logistic growth on the grass to incorporate the fact that our ecosystem is finite and this put an upper bound on the amount of grass that can grow within it. With $x(t), y(t)$ and $z(t)$ as the grass, prey and predator population size at time t and $a, b, c, d, e, f, g, x_c \geq 0$.

$$\frac{dx}{dt} = a(1 - \frac{x}{x_c})x - bxy \tag{2.14a}$$

$$\frac{dy}{dt} = -cy + dxy - eyz \tag{2.14b}$$

$$\frac{dz}{dt} = -fz + gyz \tag{2.14c}$$

The parameters are described as

- a, the growth rate of grass.
- b, the rate at which a prey eat the grass.
- c, the rate at which the prey starve in the absence of grass to eat.
- d, the growth rate of prey per grass.
- e, the rate at which a predator eat prey.
- f, the rate at which the predators starve in the absence of prey to eat.
- g, the growth rate of predators per prey.
- x_c, the grass carrying capacity of the ecosystem

Different choices of the parameters will yield different population dynamics and the analysis of this set of differential equations are a vast topic in itself. What we need to know for this thesis is that for some parameter configurations, the solution to these analytic solutions are cyclic. In fig. 2.4 an example of cyclic dynamics in a three-species predator prey system is shown. The three-species LV equations are based on several unrealistic assumptions.

1. The environment is constant and genetic adaptation is not assumed to be negligible.
2. Predators and prey have an infinite appetite.

3. The population sizes are infinite and continuous.

4. The grass growth rate is only dependent on the grass population size, but converges as the carrying capacity of the environment is reached ($a(1 - \frac{x}{x_c})x$).

5. The grass is eaten by the prey in proportion to the prey population size (the $-bxy$ term).

6. The prey and predator populations growth rate increases linearly with the food available (the dxy and gyz terms).

7. The prey can die from natural causes similar to the predators (the $-cy$ term), and through predation which is proportional to the predator population (the $-eyz$ term).

8. The predators can only die from natural causes (the $-fz$ term), and it is not dependent on the amount of food available.

Variants of the equations have been developed, which in different ways consider more realistic assumptions. Two major assumptions that every model needs to consider is the numerical and functional response, concepts first introduced by M. E. Solomon in [26]. The numerical response is the change in population density as a function of the food density of that species (assumption 6). It may be divided into two mechanisms, the demographic and aggregational responses. The first is the change in the reproduction due to changes in food density. The most simple model of prey and predator's demographic response is based on the assumption that reproduction rate is linearly proportional to the food. The aggregational response is the change in predator population due to immigration into an area with increased prey population and is not considered in the LV model above. The functional response is the predation rate as a function of food density, it is also assumed to have a linear response in the classic LV model (assumption 5 and 7). One of the most popular variations of the LV equations is the Rosenzweig–MacArthur model [27]. Here, they introduce a non-linear Holling type II functional response [28]. For this type of response, prey mortality from predation declines with prey density.

In fig. 2.5, the populations sizes of lynx and hare were reported on an annual basis in the Canadian Rockies during a 90 year period and analysed by [29]. As can be seen, it shows some cyclic dynamics but has some significant differences when compared with the analytic model.

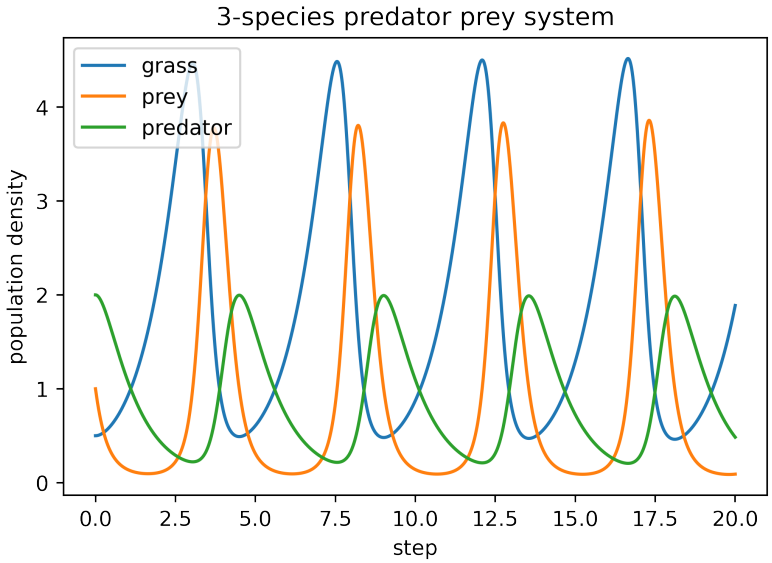

Figure 2.4: Example of a three-species predator prey system with cyclic dynamics. Here $x_0 = 0.5, y_0 = 1, z_0 = 2$ and $a = b = c = d = e = f = g = 1$, $x_c = 1000$.

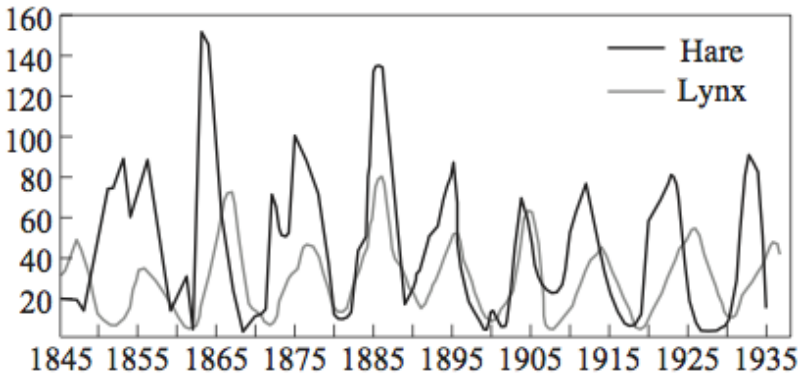

Figure 2.5: Example of real world data for a 2-species predator prey system.

2.3 Simulated Ecosystems

In this section, the mathematical formulation that will be used to describe our ecosystems is presented. The reward system denoted the happiness network is then introduced.

2.3.1 Formulation of an ecosystem

Similar to how they did in [30], we make a number of definitions that will be useful in order to formalize an ecosystem.

Definition 1. *We define an ecosystem as a pair* (A, E) *with* A *being the set of animats and* E *the environment consisting of all non-animat objects which can be divided into subsets if needed.*

Definition 2. *An animat is a pair* (G, P) *with* G *called the genotype and* P *the phenotype.*

The genotype is a property that is fixed throughout the lifetime while the phenotype is used to describe the part of the animat that is changing throughout the lifetime.

Definition 3. *A genotype consists of the following:*

(i) *A set of actions* $\mathcal{A}$.
(ii) *A set of sensors* $\mathcal{Z}$ *through which a set of objects* $\mathcal{B}_z$ *for* $z \in \mathcal{Z}$ *can be observed.*
(iii) *A set of events that will lead to death* $\mathcal{D}$.
(iv) *A set of physical properties* $\mathcal{F}$.
(v) *A set of homeostatic variables* $\mathcal{H}$. *Each homeostatic variable having a set of objects* $\mathcal{B}_h$ *which through a set of events* $\mathcal{E}_o$ *can alter the homeostatic variable.*
(vi) *A genotype policy network* $\pi_g(o)$ *which is a probability distribution over actions* $a \in \mathcal{A}$ *for each observation* $o \in \mathcal{O}$.
(vii) *A reward function* $\mathbb{R}_a(s, s')$.

Definition 4. *A phenotype consists of the following:*

(i) *A set of observations* $\mathcal{O}$ *dependent on the sensors, homeostatic variables and the ecosystem in which the animat lives.*
(ii) *An experience trajectory* $\mathrm{Exp} = [(o_1, a_1, r_1), \ldots, (o_{t-1}, a_{t-1}, r_{t-1})]$.
(iii) *A phenotype policy network* $\pi_p(o)$ *which is a probability distribution over actions* $a \in \mathcal{A}$ *for each possible observation* $o \in \mathcal{O}$.

The policy network in the genotype π_g, is the policy that an animat inherits at birth, while π_p in the phenotype is what a particular individual has learnt through its experience trajectory Exp by starting from π_g at birth.

2.3.2 Happiness network

In this section, two different approaches to modelling behavioral adaption are described and then we present the *happiness network*, which is an attempt to combine the two.

Most reinforcement learning models which aims to imitate behavioral adaption in animals, consider the animals response to external changes. This model assumes that animals objectives are to select actions in a way that maximizes the reward acquistion. That real world animals adapt to their environment by continuously updating their estimates of state-values and the teaching signal is suggested to be carried by the phasic activity of the midbrain dopamine neurons [31]. This signal is projected onto the striatum, where stimulus-response associations are encoded.

Others approach the behavioral adaptation in animals by considering homeostatic regulation. Here, the assumption is that in order to survive, a number of homeostatic variables should not deviate too far from their optimal setpoints. In most animals, there are many homeostatic variables such as body temperature, pH, sodium level, potassium level, calcium level and blood sugar level. These all have to be kept within a homeostatic range. If any of them are not within this range, the animal will die. Thus, the behavioral adaption in animals is to response appropriately to perturbations of these homeostatic variables. These models are called Negative-Feedback models of homeostatic regulation [32]. That these models play a vital

role for some behavioral adaptations is argued to be indisputable, a number of criticism have been raised though. For example, after some learning trials, rats that were trained to run down an alley for intragastic feeding of milk, did not show any motivation to do so, while rats that were trained with normal drinking of milk quickly showed this motivation.
Both of these approaches both have their shortcomings. The first one, not taking into account the internal state of the animal and the second one not taking into consideration that external stimuli is a strong reinforcing component in the process of learning. Thus, efforts have been made to take a more unified approach [9]. In this thesis, we also try to unify these two approaches by introducing the happiness network.
The happiness network consists of a happiness function and a reward function which takes the two most recent happiness function evaluations and returns a reward. The happiness function of an animat takes an observation from its sensors and the homeostatic variables as input and outputs an associated happiness. The happiness function can be constructed in many ways, our approach aims to take the internal state of the animat into account by making the animat less happy as homeostatic variables deviate from their optimal setpoints. The external observatory data is used to make the animat more happy when they make observations that are associated with events that bring the homeostatic variables closer to their setpoints. For example, the smell of food makes the animat happy when it is hungry while the smell of a predator makes it less happy. Let $\mathbf{o}_t$ be the vector containing the $n =\mid \mathcal{Z} \mid$ vector observations for each sensor at the current timestep t. The second term is a summation over all sensors and the set of objects observable by the sensor. The function f_{zb} is then the associated happiness for sensor z and object b when making observation $\mathbf{o}_t^z$. For example, z being smell and b being food.

$$\text{Happiness}(\mathbf{o}_t) = \sum_{i \in \mathcal{H}} \frac{\ln(1 + \alpha_{h^i} h_t^i)}{\ln(1 + \alpha_{h^i})} + \sum_{z \in \mathcal{Z}} \sum_{b \in \mathcal{B}_z} f_{zb}(\mathbf{o}_t)$$

with $h^i \in [0, 1]$ which has optimal value $h^{i,*} = 1$. This way, any deviation from the optimal homeostatic value decreases the happiness in an exponential way. The shape of the exponential decay is determined by the parameter α_{h^i}. The motivation for the happiness function is that the homeostatic function should change more rapidly when a homeostatic variable is close to critical, while on the other hand, changes in a homeostatic variable when we are close to the optimal value, doesn't change the homeostatic function by a large amount. For example, going from being thirsty to almost die due to dehydration, gives a greater sense of urgency than going from very well hydrated to slightly less so. This way, the animats obtain a greater reward for satisfying homeostatic needs when these are close to critical, otherwise they will die. In this work, we assume that all homeostatic variables are critical, meaning that if the variable becomes zero, the animat dies.
Now that we have defined the happiness function, we move on to how the rewards are constructed using the output from the former. The reward at the current timestep is simply the difference in happiness compared with the previous timestep. Simple and intuitive.

$$\mathbb{R}_a(s_{t-1}, s_t) = \text{Happiness}_t - \text{Happiness}_{t-1} \tag{2.15}$$

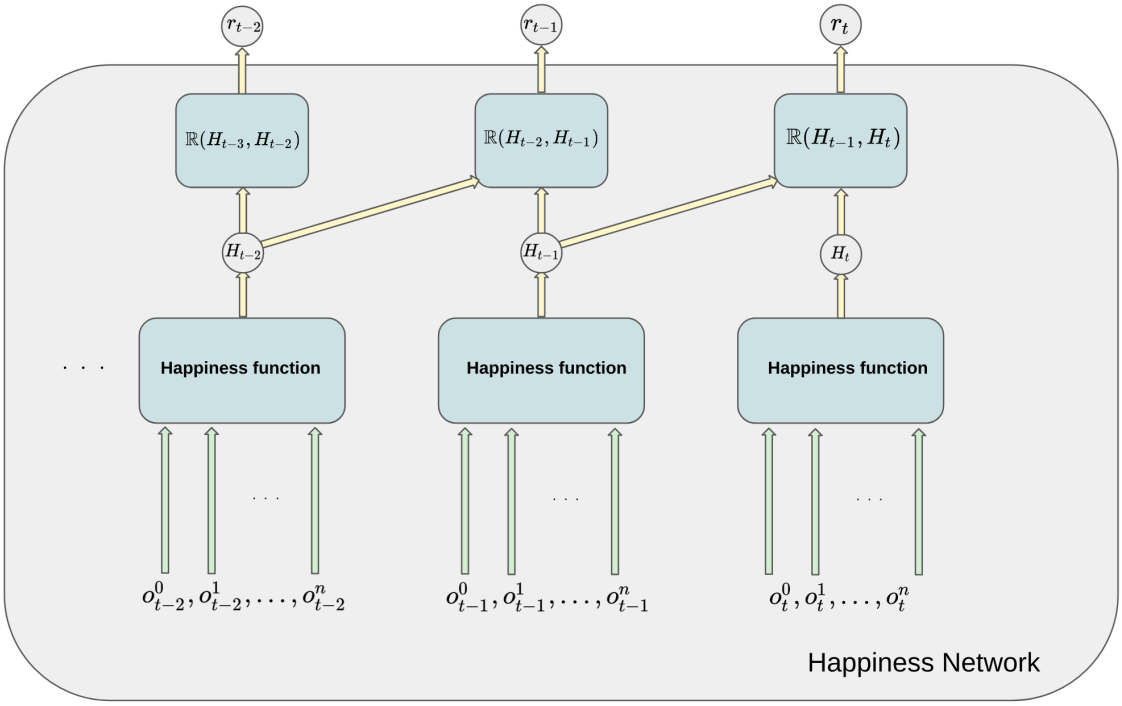

Figure 2.6: The happiness network

3
Methods

In this section, we will begin by describing how we do MARL in Unity. Then we will continue by going through the setup of the different agents, how they observed the environment and their reward functions. The different environment setups that were evaluated will be described.

3.1 Reinforcement Learning in a Unity simulation

ML-agents is an open source project which enables training of agents within the popular game engine Unity. In this thesis, we utilize part of ML-agents but with our own implementation of the decision making part (referred to as the Python Trainer). This is done in order to be able to customize the RL to a larger extent. The Python Trainer uses the OpenAI gym interface [1] which has become somewhat of a research standard in RL. We utilize implemented RL-algorithms from Stable Baselines3 [2].
The block diagram fig. 3.1 shows the flow of information from the environment to the Python Trainer. The environment contains a number of agents, whose observations are sent to its individual or shared behavior. The behavior can either perform inference using an existing model, use some heuristic or be sent through the Unity communicator via the Python API to the Python Trainer. The actions decided by the trainer is then propagated the opposite way to the agents. We also use custom sidechannels to obtain additional environment data and use tensorboard [3] to track the training progress in real time.
The interaction between the environment and the agent with respect to the sensory input, homeostatic variables, happiness network and policy network is displayed in fig. 3.2.

3.2 Sensors

Two different sensors, vision and smell are used by the animats to make a partial observation $\mathbf{o}_t$ of the state s_t of the ecosystem. The idea of combining multiple sensors is that vision may contain more useful information but may also be harder to interpret due to the high dimensionality. Smell on the other hand is a preprocessed

[1] `https://github.com/openai/gym`
[2] `https://github.com/DLR-RM/stable-baselines3`
[3] `https://github.com/tensorflow/tensorboard`

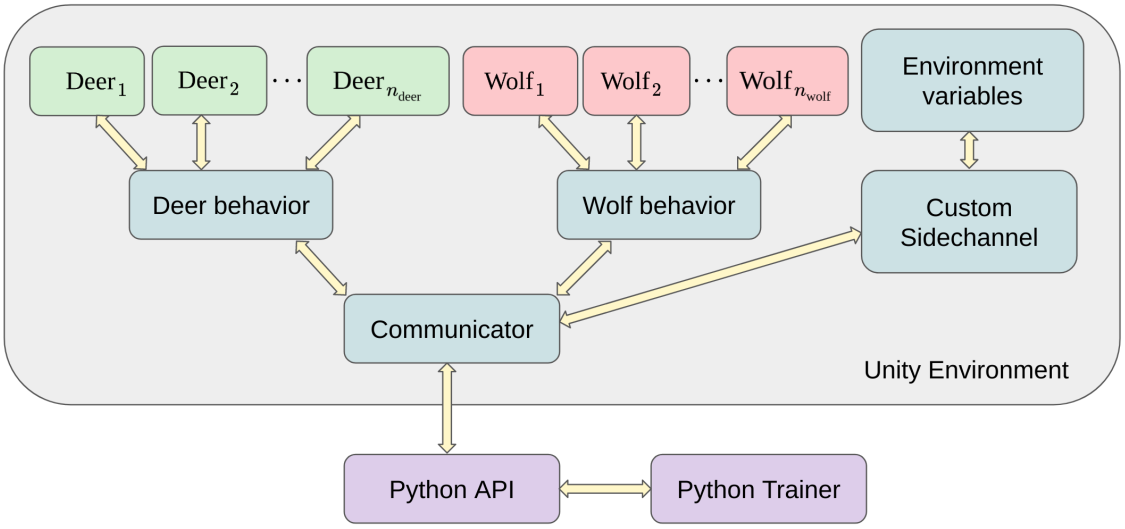

Figure 3.1: Block diagram for information flow between agents and the Python Trainer, a centralized learning, decentralized execution is utilized.

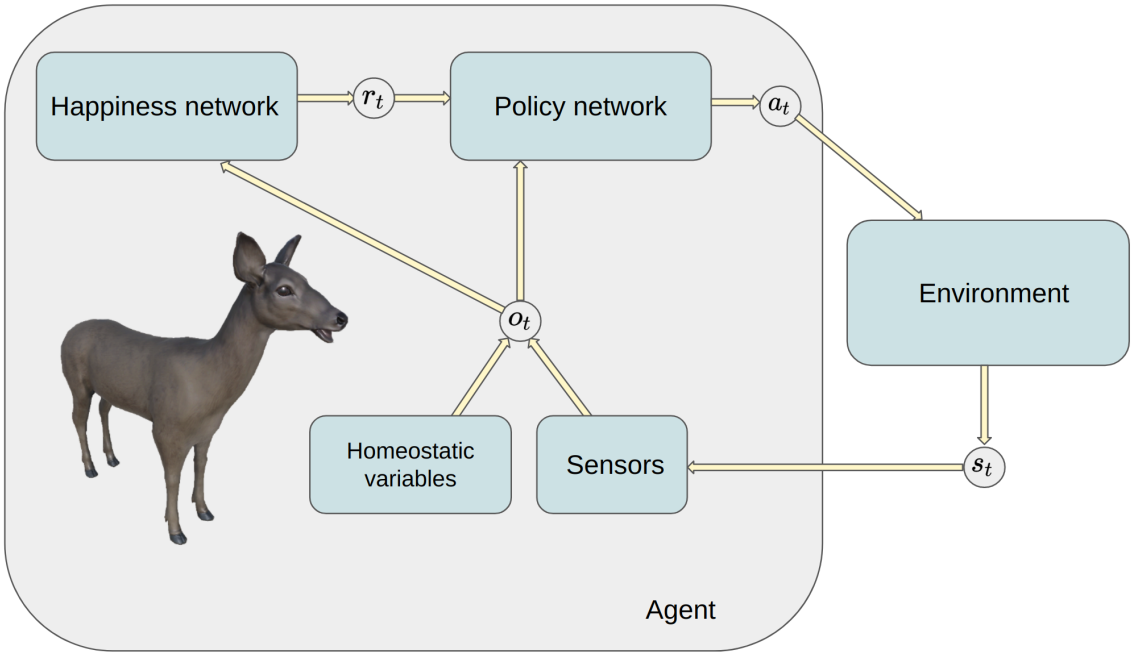

Figure 3.2: The interaction between the environment and the agent.

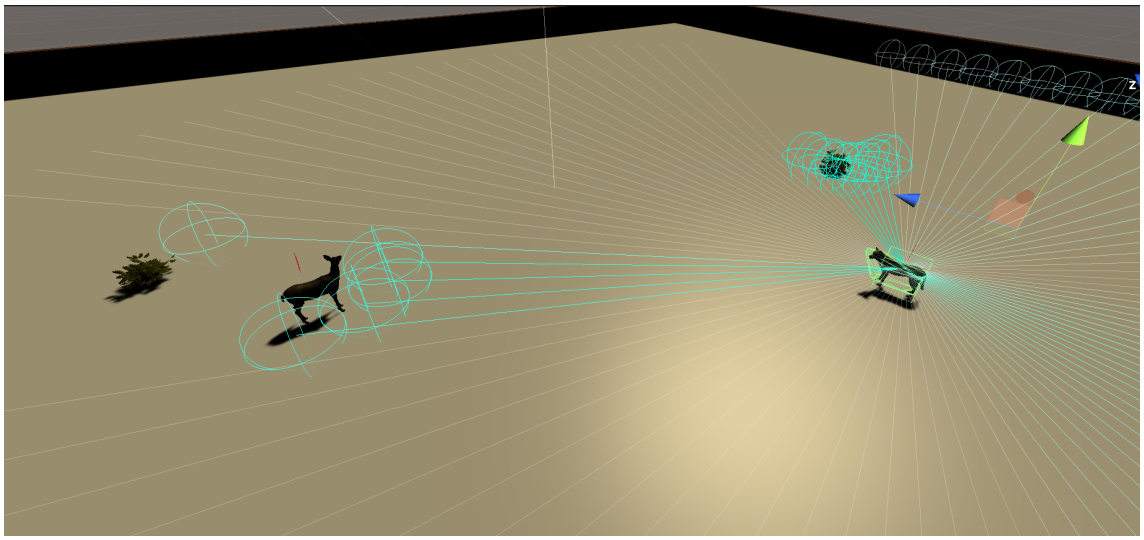

Figure 3.3: A wolf and its vision sensor, here observing the surrounding objects within its observation radius.

observation of significantly fewer dimensions and thus possibly easier to interpret for the animats. Further, smelling is not prohibited by obstructions. This way, we hope that the two sensors complement each other.
Many real life mammals rely on vision. Using raw images as sensory input in RL has some associated challanges. However, in Unity we can utilize the so called RayPerceptionSensor. This sensor which we call vision, shoots a number of rays in a desired direction and distance. If a ray hits an object it will report back the tag of this object if the tag is within the set of observable objects that we have chosen. The distance to the object is also registered as the ratio of the distance and the maximum distance. For example, if a ray hits a food object, the animat knows that it is a food object and we do not have to go through the hassle of first learning an animat how a food object looks like. This is very convenient.
Given a set of objects that are observable through vision $\mathcal{B}_{\text{vision}}$ and a set of rays $\mathcal{R}$ that together makes up the vision sensor, for each ray $r \in \mathcal{R}$ at timestep t, we will obtain an observation

$$\mathbf{o}_t^r = [i_t^1, \ldots, i_t^n, d_t] \tag{3.1}$$

with i_t^b indicating with a boolean whether the ray hit the object $b \in \mathcal{B}_{\text{vision}}$. The distance to the object d_t is observed as a ratio of the maximum distance of the ray. A complete vision observation at time t with is then

$$\mathbf{o}_t^{\text{vision}} = [\mathbf{o}_t^1, \mathbf{o}_t^2, \ldots, \mathbf{o}_t^{|\mathcal{R}|}].$$

The vision of the animats consists of 150 rays, evenly distributed at 360 degrees around the animat, see fig. 3.3 for an example. Vision of real life animals is not 360 degrees, but they can usually rotate their head and this is an simplification of this. The sensor for smelling works in a different way. An animal can sense the magnitude of a particular smell if it is present. By moving its head it detects differences in magnitude and thus a probable direction of the origin for the smell. We let each animat have a set of smellable objects $\mathcal{B}_{\text{smell}}$. We make the assumption that

an animat can find the magnitude m_t^b which gives how strong the smell of object $b \in \mathcal{B}_{\text{smell}}$ is. The magnitude has the property $0 \leq m_t^b \leq 1$ with a value of 1 when the smell in the olfactory sensor is at its maximum. The animat also observes the relative angle ϕ_t^b to this object. For each type of smell, the animat observes the smell magnitude of the object with maximum magnitude as well as the relative angle to this object. The magnitude is given by

$$m_t^b = 1 - \frac{d_t^b}{d_{\text{max}}^b} \tag{3.2}$$

with d_b being the distance to the closest object of this smell and d_{max}^b the maximum distance which the smell of b can be scented. The observation for a particular smell $b \in \mathcal{B}_{\text{smell}}$ is then

$$\mathbf{o}_t^b = [m_t^b, \phi_t^b] \tag{3.3}$$

and a complete smell observation

$$\mathbf{o}_t^{\text{smell}} = [\mathbf{o}_t^1, \mathbf{o}_t^2, \ldots, \mathbf{o}_t^{|\mathcal{B}_{\text{smell}}|}] \tag{3.4}$$

In order to know its own internal state, the animat needs to also observe its homeostatic variables. At each time step it observes the $\mid \mathcal{H} \mid$ homeostatic variables.

$$\mathbf{o}_t^{\text{homeostatic}} = [h_t^1, h_t^2, \ldots, h_t^{|\mathcal{H}|}] \tag{3.5}$$

A common issue in RL, and many other machine learning tasks, is how to deal with problems where a temporal memory might be helpful. For example, if an animat observes some other animat, it can't know in which direction the observed animat is moving, using only the most recent sensor observation data. The way this is dealt with in this implementation, is by stacking n consecutive observations. This way, a complete observation at time t is

$$\mathbf{o}_t = [\mathbf{o}_{t-n}^{\text{vision}}, \mathbf{o}_{t-n}^{\text{smell}}, \mathbf{o}_{t-n}^{\text{homeostatic}}, \ldots, \mathbf{o}_{t-1}^{\text{vision}}, \mathbf{o}_{t-1}^{\text{smell}}, \mathbf{o}_{t-1}^{\text{homeostatic}}, \mathbf{o}_t^{\text{vision}}, \mathbf{o}_t^{\text{smell}}, \mathbf{o}_t^{\text{homeostatic}}] \tag{3.6}$$

3.3 Simulations

The simulations runs in two phases, the first one being the pretraining and the second one, the dynamics simulation. During pretraining, the number of indiviudals of predators and prey are fixed and if an agent dies, its episode ends and it is respawned instantly. In contrast, during the dynamics simulation an animat is removed permanently when it dies. The number of agents for each speciesis fixed to $(x = 100, y = 100, z = 50)$ in the pretraining phase.
The motivaton for this pretraining approach instead of starting the dynamics simulations with untrained behaviors is, in the context of biology, that animals are not born with completely untrained brains. Rather, they are born with a large amount of reflexes and decision-making encoded in the genetics. This way our pretraining can be seen as the process of evolution shapes the baseline behavior of a particular specie.

With the pretrained models, the dynamics simulation can be initiated. The goal of this simulation is to record the population dynamics. In this phase, each species is still being trained and can thus adapt to the dynamics of the ecosystem. Each species is prevented from going extinct by adding a new individual if the number of animats of a species becomes zero, similar to how animals outside of the ecosystem can migrate into it and how they did in [5]. This is one way to deal with the fact that the LV model is continuous and the population density of a species can get close to zero but never go extinct.

3.4 Ecosystems

Ecosystem objects

The ecosystem is a pair (A, E), with A being the set of animats and E the rest of the environment. There are two type of animats in the environment.

$$\mathrm{A} = \{\text{deer}, \text{wolf}\}$$

with the deer being a prey and the wolf a predator. The grass, which is a species which dynamics we measure, is not an animat, but part of the environment. If there were multiple types of plants or other non-animal organisms with more complex inner workings, these could be divided into well defined sets with well defined properties. However, since we only consider one type of grass which only property is the ability to reproduce, we omit this to keep things more simple. The three species can be seen in fig. 2.4. The other objects in the environment are the ground and the walls. Thus,

$$\mathrm{E} = \{\text{grass}, \text{ground}, \text{wall}\}$$

The ground is flat square area with an edge length of 200, bounded by impassable walls. In fig. 3.5, the complete simulation environment is shown.

Actions

The set of actions are the same for both animats. At each decision step, the animat selects a value for each of the two actions in the sets

$$\mathcal{A} = \{\text{move}, \text{rotate}\}, \quad \text{move} \in \{0, 0.5, 1\}v_{\max}, \quad \text{rotate} \in \{-1, 1\}\theta_{\max}$$

with $v_{\max}$ being the maximum forward movement speed which is described by the physical properties of the genotype and $\theta_{\max}$ the maximum rotation speed of the animat which is also part of the genotype.

Policy networks

Each species j has a genotype policy network π_g^j which is obtained from the pretraining. In the dynamics simulations, each animat starts with this genotype policy networks, which is then altered during the lifetime and then referred to as the phenotype policy network π_p^j.

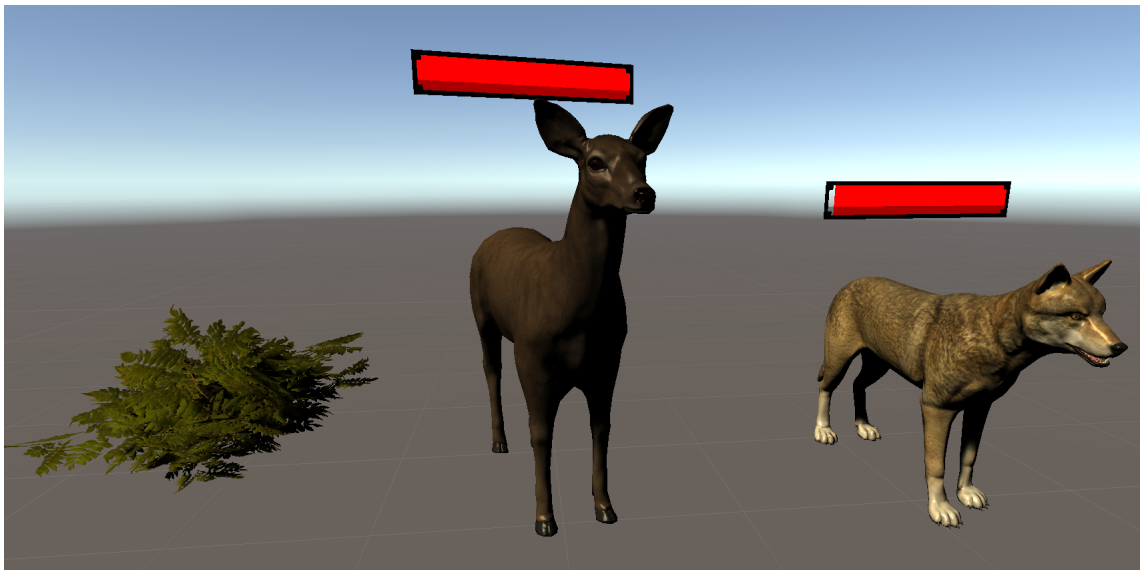

Figure 3.4: The three species in Unity, grass, a deer and a wolf. The red bars above displays the current energy levels of the animat.

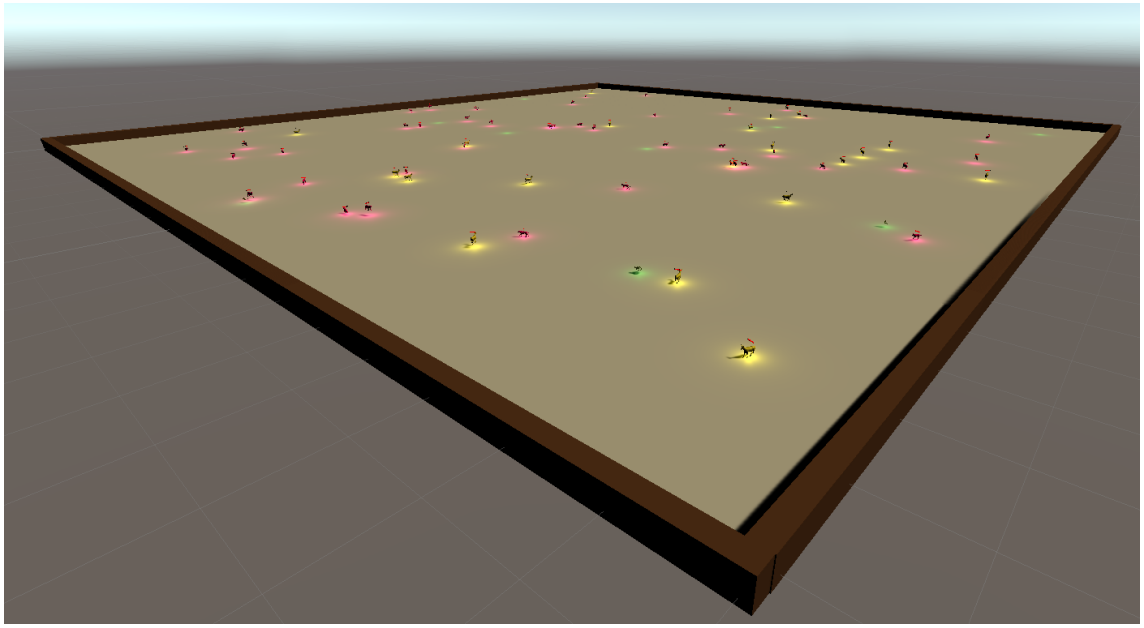

Figure 3.5: The simulation environment, the gameobjects of each specie glow in different colors to easier be distinguishable when watching the simulations.

Sensors

The set of sensors are, for both animats

$$\mathcal{Z} = \{\text{vision}, \text{smell}\}$$

where the animats have a set of observable objects $\mathcal{B}_{\text{vision}}$ for their vision and a set $\mathcal{B}_{\text{smell}}$ which they can smell. Each animat can with their vision observe all animats and objects. They can smell every other animat and the grass.

$$\mathcal{B}_{\text{vision}} = \{\text{A}, \text{E}\}, \quad \mathcal{B}_{\text{smell}} = \{\text{A}, \text{grass}\} \tag{3.7}$$

Homeostatic variables

The homeostatic variable of the deer and the wolves are just their need for energy.

$$\mathcal{H}_{\text{deer}} = \mathcal{H}_{\text{wolf}} = \{\text{energy}\}$$

with associated events

$$\mathcal{E}_{\text{energy}} = \{\text{eat}, \text{metabolism}, \text{predation}\}$$

Where the event of eating occurs when an animat collides with one of its eatable objects. Metabolism is going on all the time, but at a higher rate during movement. Predation is considered related to energy because the predator reduces the energy level of the prey to zero in a collision and this is what leads to the death of the prey. The set of objects associated with an event are specific for an animat type.

$$\mathcal{B}^{\text{deer}}_{\text{eat}} = \{\text{grass}\}, \quad \mathcal{B}^{\text{deer}}_{\text{predation}} = \{\text{wolf}\}, \quad \mathcal{B}^{\text{wolf}}_{\text{eat}} = \{\text{deer}\} \tag{3.8}$$

Death

Death occurs when any of the homeostatic variables become less than or equal to zero.

$$\mathcal{D} = \{\min_{h_i \in \mathcal{H}} \frac{\ln(1 + Ch_i)}{\ln(1 + C)} \leq 0\}, \tag{3.9}$$

Physical configurations

The wolves are 20% faster than the deer. This is a design choice made in order to compensate for the fact that hitting a moving target is hard, and this is what the wolves have to do when they chase a deer. If they had a similar speed, situations where a wolf chased a deer without getting any closer, frequently occurred. The physical configuration of each species is given as a set

$$\mathcal{F}_{\text{deer}} = \{\alpha^{\text{deer}}_{\text{metabolism}}, \alpha^{\text{deer}}_{\text{reproduction}}, \text{age}, d^{\text{wolf}}_{\text{max}}, v^{\text{deer}}_{\text{max}}, \theta^{\text{deer}}_{\text{max}}\} \tag{3.10}$$

$$\mathcal{F}_{\text{wolf}} = \{\alpha^{\text{wolf}}_{\text{metabolism}}, \alpha^{\text{wolf}}_{\text{reproduction}}, \text{age}, d^{\text{wolf}}_{\text{max}}, v^{\text{wolf}}_{\text{max}}, \theta^{\text{wolf}}_{\text{max}}\} \tag{3.11}$$

$$1.2v_{\max}^{\text{deer}} = v_{\max}^{\text{wolf}}, \quad \theta_{\max}^{\text{deer}} = \theta_{\max}^{\text{wolf}}. \tag{3.12}$$

The rate of energy decay ϵ_{energy} depends on the metabolism parameter $\alpha_{\text{metabolism}}$ and the choice of movement action by the animat. At each timestep, the base level of metabolism is 0.0004. Thus, an animat which does not move and begins with full energy level will consume all of it and die after 2500 time steps. However, if its chooses to run at maximum speed until death, it will die after slightly more than 300 time steps.

$$\alpha_{\text{metabolism}}^{\text{deer}} = \alpha_{\text{metabolism}}^{\text{wolf}} = 0.0004 \tag{3.13}$$

$$\epsilon_{\text{energy}} = \begin{cases} \alpha_{\text{metabolism}}, & \text{if move} = 0 \\ 2\alpha_{\text{metabolism}}, & \text{if move} = 0.5 \\ 8\alpha_{\text{metabolism}}, & \text{if move} = 1 \end{cases}$$

The reproduction probability $\alpha_{\text{reproduction}}$ specifies the probability that an animat will give birth to a child per time step. The grass also has a reproduction probability. All species spawn the child to a random unoccupied location in the environment. The reason that the reproduction probability of the wolves is lower is that they are apex-predators, i.e not predated by any other species. Thus, if they begin to grow faster than they die due to lack of prey, they will grow unboundedly even in the absence of prey since they are not dying from predation.

$$\alpha_{\text{reproduction}}^{\text{grass}} = 0.001, \quad \alpha_{\text{reproduction}}^{\text{deer}} = 0.001, \quad \alpha_{\text{reproduction}}^{\text{wolf}} = 0.0008 \tag{3.14}$$

The parameter $d_{\max}$ specifies the maximum distance at which the animat can observe, in other words, the range of the smell of vision sensors. The prey have a larger observation range than the predators. This is to give the prey the possibility of learning to avoid the predators before they are within the observable range of it. In fig. 3.6, the observation radius of a wolf and a predator can be seen.

$$d_{\max}^{\text{deer}} = 40, \quad d_{\max}^{\text{wolf}} = 30 \tag{3.15}$$

Happiness function

The first term of the happiness function in section 2.3.2

$$\sum_{i \in \mathcal{H}} \frac{\ln(1 + \alpha_{h^i} h_t^i)}{\ln(1 + \alpha_{h^i})} \tag{3.16}$$

reduces to

$$\frac{\ln(1 + \alpha_{\text{energy}} \text{energy})}{\ln(1 + \alpha_{\text{energy}})} \tag{3.17}$$

with $\alpha_{\text{energy}} = 10$. The smelling sensor is used for the second term in the happiness function. Increasing happiness when the smell of eatable objects increases. Decreasing happiness for increasing smell of predators. Further, the happiness dependence of a particular smell depends on the homeostatic variable associated with this smell h_t^b. For example, the smell of food makes the animat more happy when it is hungry.

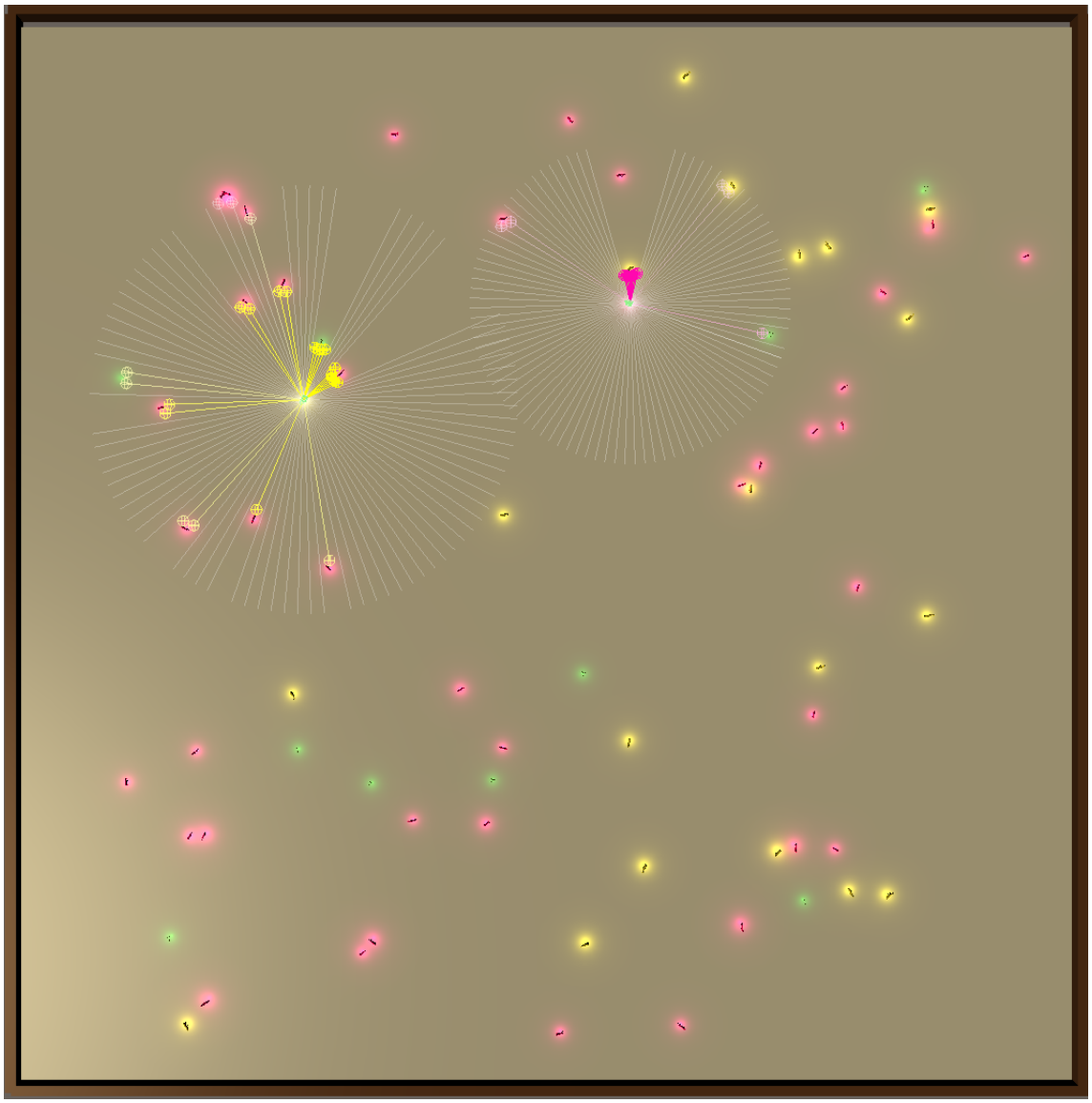

Figure 3.6: The simulation environment watched from above. The grass is green, prey yellow and predators pink. The left circle displays the vision sensor of the prey and the right, which is slightly smaller, that of a predator.

This is not the case for the smell of predator though, this smell is equally repulsive regardless of the homeostatic variables. The parameters $\alpha_{\text{goodsmell}}$ and α_{badsmell} are happiness weights for the good and bad smells. We use $\alpha_{\text{goodsmell}} = \alpha_{\text{badsmell}} = 0.01$.

$$f_{\text{smell},b}(\mathbf{o}_t) = \begin{cases} +\alpha_{\text{goodsmell}} m_t^b (1 - h_t^b), b \in \mathcal{B}_{\text{smell}} & \text{if } b \in \mathcal{B}_{\text{eat}} \\ -\alpha_{\text{badsmell}} m_t^b, b \in \mathcal{B}_{\text{smell}} & \text{if } b \in \mathcal{B}_{\text{predation}} \end{cases}.$$

A complement to the happiness function is used by giving a negative reward of -5 if the agent dies.

Mating

The mating mechanism is simplified and not based on two or more individuals meeting and deciding to mate, but rather an asexual one which depends on the population sizes. At each timestep and for each animat, we draw a uniformly random number $u \sim U(0, 1)$ and if $u < \alpha_{\text{reproduction}}$, we add a new animat of this specie.

Grass

In accordance with eq. (2.14), each grass object has a probability of $\alpha^{\text{grass}}_{\text{reproduction}}(1 - \frac{x}{x_c})$ to spawn a new grass object at a random location. The carrying capacity is determined by experimentally finding how many grass objects that can exist in the ecosystem before we can not find any free location to spawn a grass object to. We find that $x_c \approx 1000$.

3.5 Parameter estimation

The choice of parameters in the simulation have a large impact on the dynamics. In the LV equations, there are a number of parameters. The simulation parameters are listed in table 3.1. Given a particular simulation configuration, we want to estimate the LV parameters and their responses to changing population densities.
The parameters are estimated similar to how was done in [8]. The parameter a is exactly equal to the parameter $\alpha^{\text{grass}}_{\text{reproduction}}$. The parameter b is estimated by measuring the average number of grass that a prey eats per timestep for a number of fixed $(x, y = 20, z = 0)$ configurations, varying the fixed amount of grass x in each simulation. With $k(t)$ as the number of grass eaten by the y prey at timestep t, an estimation $\hat{b}$ is found by taking the average number of grass eaten per timestep in the T simulated timesteps divided by the grass density.

$$\hat{b} = \frac{1}{T} \sum_{t=1,2,\ldots,T} \frac{k(t)}{xy}. \tag{3.18}$$

Same way an estimation of e is done by measuring the average number of preys killed by predators per timestep for fixed $(x = 50, y, z = 20)$, varying the number of prey. With $p(t)$ as the number of prey predated at timestep t by the z predators,

$$\hat{e} = \frac{1}{T} \sum_{t=1,2,\ldots,T} \frac{p(t)}{yz}. \tag{3.19}$$

Simulation parameter	Description
$\alpha^{\text{grass}}_{\text{reproduction}}$	reproduction probability per individual per timestep
$\alpha^{\text{prey}}_{\text{reproduction}}$	
$\alpha^{\text{predator}}_{\text{reproduction}}$	
$\alpha^{\text{prey}}_{\text{metabolism}}$	base level energy consumption per timestep
$\alpha^{\text{predator}}_{\text{metabolism}}$	
$v^{\text{prey}}_{\text{max}}$	maximal speed
$v^{\text{predator}}_{\text{max}}$	
$\theta^{\text{prey}}_{\text{max}}$	maximal angular speed
$\theta^{\text{predator}}_{\text{max}}$	
π_{prey}	policy network
π_{predator}	
E	The simulation environment size, objects, etc.

Table 3.1: The different simulation parameters which have an effect on the dynamics.

The parameters c and d are estimated by considering

$$r = -c + dx \tag{3.20}$$

as the aggregate growth rate of the prey in the absence of any predators

$$\frac{dy}{dt} = ry \tag{3.21}$$

Which is a differential equation with the solution

$$y(t) = y_0 e^{rt}. \tag{3.22}$$

If we now measure the population size $y(t)$ for different fixed values of x, we can find the corresponding growth rate r by fitting the linear equation for the logarithmized population size.

$$\ln(y(t)) = \ln(y_0) + rt \tag{3.23}$$

This way, we obtain (r, x) tuples which can be used to fit a linear equation for eq. (3.20) which will give us the estimation of the constants c and d. Same way we estimate f and g by considering the growth of predators for a fixed number of prey.

4
Results

4.1 Lotka Volterra dynamics

In fig. 4.1, four population dynamics simulations of three-species predator-prey systems with equal parameter configurations and initial values ($x_0 = 10, y_0 = 20, z_0 = 40$) were run. The population dynamics exhibits cycle properties. As the figure shows, the simulations almost immediately diverges from each other due to their stochastic nature. However, some properties are similar for the simulations, such as the time period and average peak amplitude, displayed in table 4.1. In comparison, the population dynamics for the same environment configuration with random policy agents does not exhibit limit cycles, as seen in fig. 4.2. In this case, the grass grows quickly to near its carrying capacity, then it takes a long time before the prey population starts to grow due to their incapacitates of finding food. A large prey population can now coexist with the grass without swiping the environment clean of grass, as the trained agents do. The predators never get a chance to grow since the probability of successfully finding prey when selecting actions randomly, is too low.

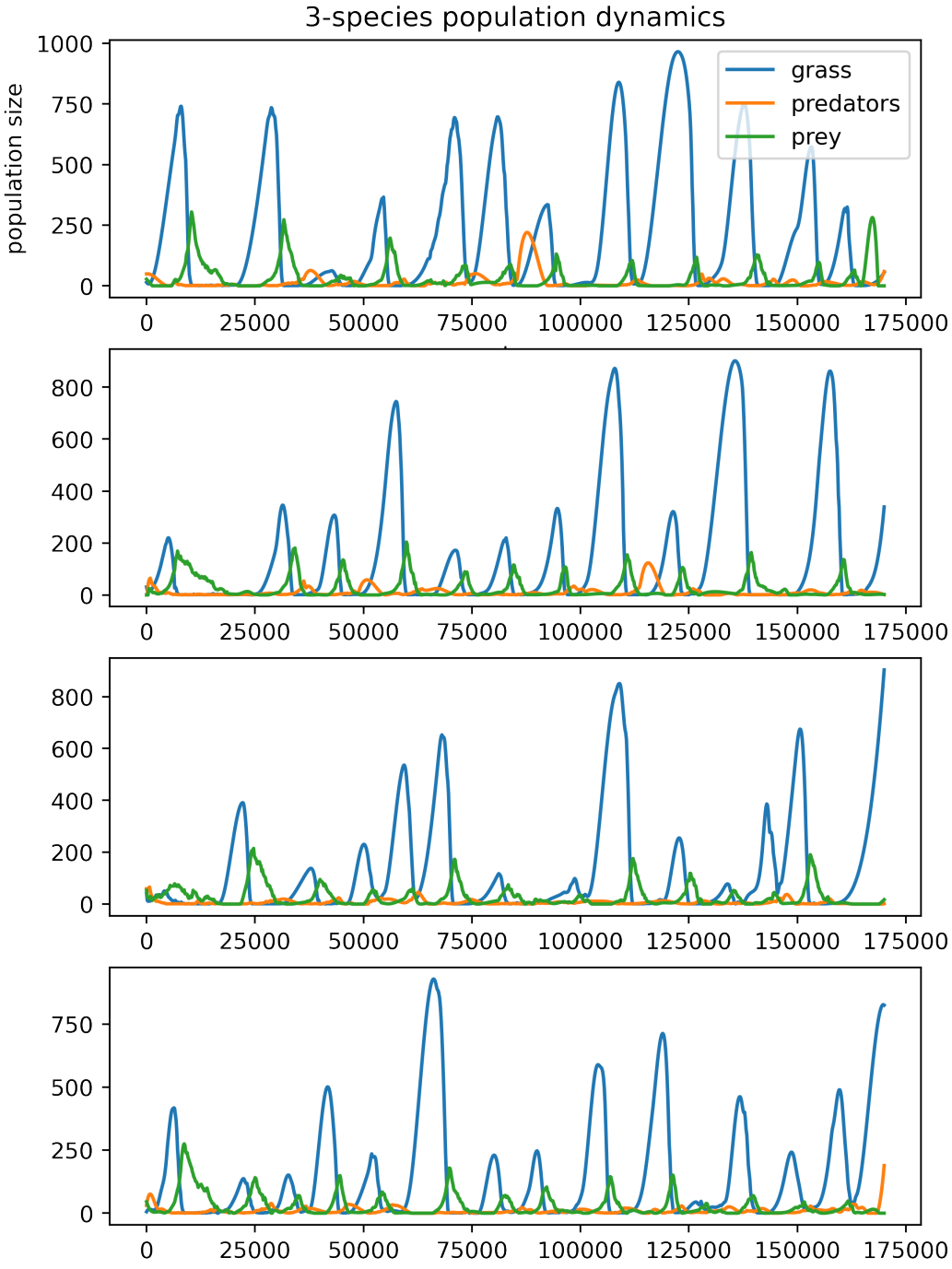

Figure 4.1: Population dynamics simulations of three-species predator-prey systems with equal simulation parameters and initial conditions. The variation is explained by the stochasticity.

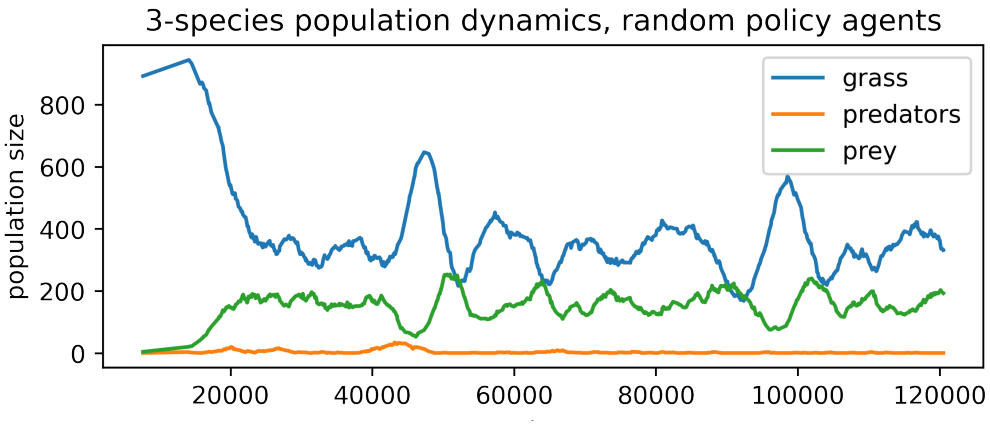

Figure 4.2: Population dynamics when using random policy agents.

Simulation	Species	Amplitude	Period
1	grass	638(203)	90(29)
	prey	152(75)	94(25)
	predators	51(58)	98(40)
2	grass	482(282)	90(27)
	prey	142(34)	90(28)
	predators	47(32)	120(53)
3	grass	423(235)	86(34)
	prey	123(57)	96(34)
	predators	25(16)	84(33)
4	grass	441(243)	74(16)
	prey	129(57)	70(22)
	predators	29(14)	68(33)

Table 4.1: Average peak amplitude and period length for the four simulations in fig. 4.1. The standard deviation is given in the parentheses.

4.2 Parameter estimation

One question we asked was whether an estimation of the parameters of the three-species LV model could be reconstructed from our simulation results and how the parameter estimations responded to changing population densities and thus, if the assumptions of the model regarding the responses were satisfied in the simulations. For the grass eating rate parameter b, which is the probability for a grass to be eaten by a prey per alive prey in the environment, the assumption is that this should be constant for different densities of grass, in other words, a linear functional response. If there is a higher grass density, the prey will have a higher grass eating rate, but for an individual grass its probability of being eaten remains the same. In fig. 4.3, this seems to be an assumption that holds to some extent. The slopes of the cumulative eating for the average prey in environments with different grass densities are similar and the variation does not seem to be related to the changes in grass density. The same measurements are made for e, the predation rates by predators in fig. 4.4. Here there are some differences in the predation rates for the different prey densities. A prey living in a higher prey density environment has a lower mortality rate than those in the lower density environments. Thus, this assumption of the LV model does not seem to hold in the simulation for this parameter.

The growth rate of the prey in the absence of predators $r_{\text{prey}} = -c + dx$ is estimated by measuring the exponential growth rate for different fixed grass densities in fig. 4.5. In the first figure, we initialize the environment with 10 prey and measure the population growth. In the second figure, the environment is initialized with 100 prey and we use lower grass densities to also be able to measure the decay. Since the measured population sizes are logarithmized, a straight slope indicate exponential growth and decay. This seems to be an assumption that holds. Each line in fig. 4.5 gives us a datapoint (r_{prey}, x) which is used to fit the linear equation for r_{prey} as a function of the grass density. The assumption for this demographic response in the LV model is a linear response. This assumption means that the growth rate increases linearly with increasing grass density. However, this seems to not hold in our simulation environment. In fig. 4.7, we note that for fixed $x > 20$, the growth rate starts to converge.

In the same way, we measure the growth rate parameters f, g for the predators by measuring the population growth in environment with fixed prey densities. Here, $r_{\text{predator}} = -f + gy$. In fig. 4.6, there seems like the exponential growth assumption holds with noisy but overall linear slopes of the logarithmized population sizes. However, even more apparent than for the prey, as can be seen in fig. 4.8, the growth rate parameter r_{predator} is not linearly dependant on the prey density. The data suggest that there are two domains, if $y < 20$ there is a constant negative growth rate and for $y > 20$ there is a constant positive growth rate, while close to $y = 20$ there is a quick transition between these two domains. Thus, the LV model assumption for the demographic response doesn't hold for these parameters either in the simulation environment.

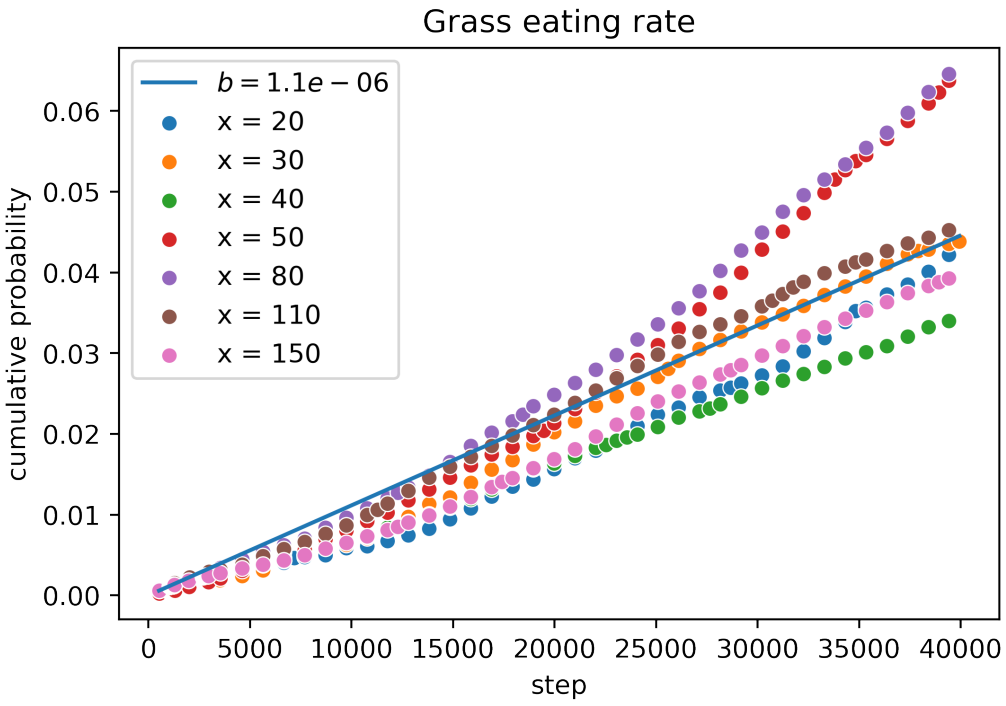

Figure 4.3: The parameter b is the rate per timestep at which a prey eats grass per available grass in the environment.

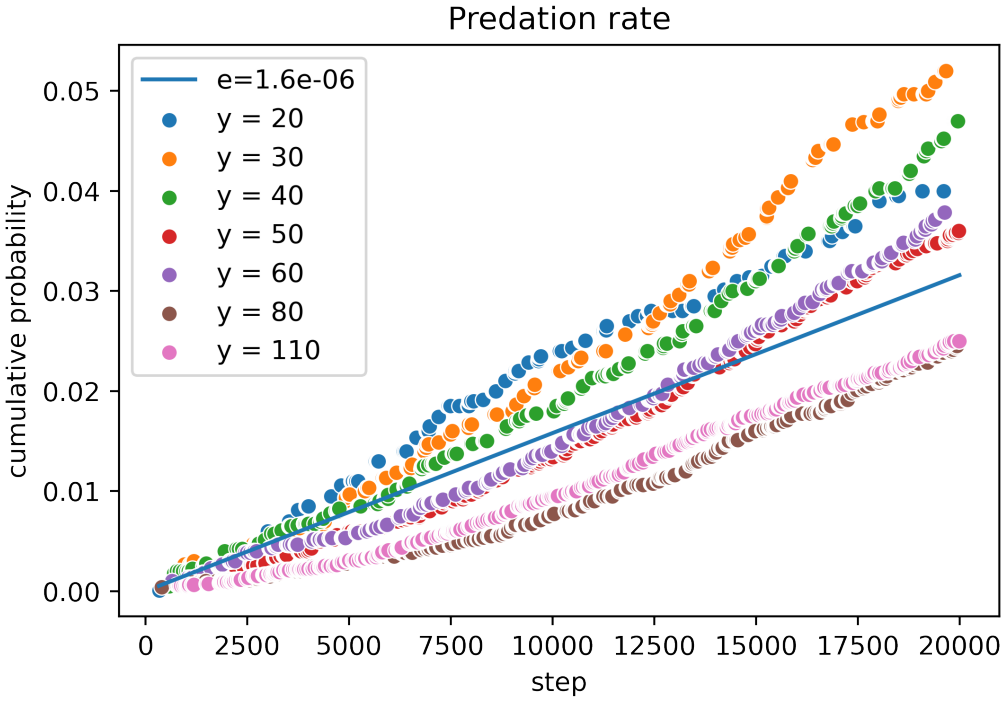

Figure 4.4: The parameter e is the rate per timestep at which a predator eats prey per available prey in the environment.

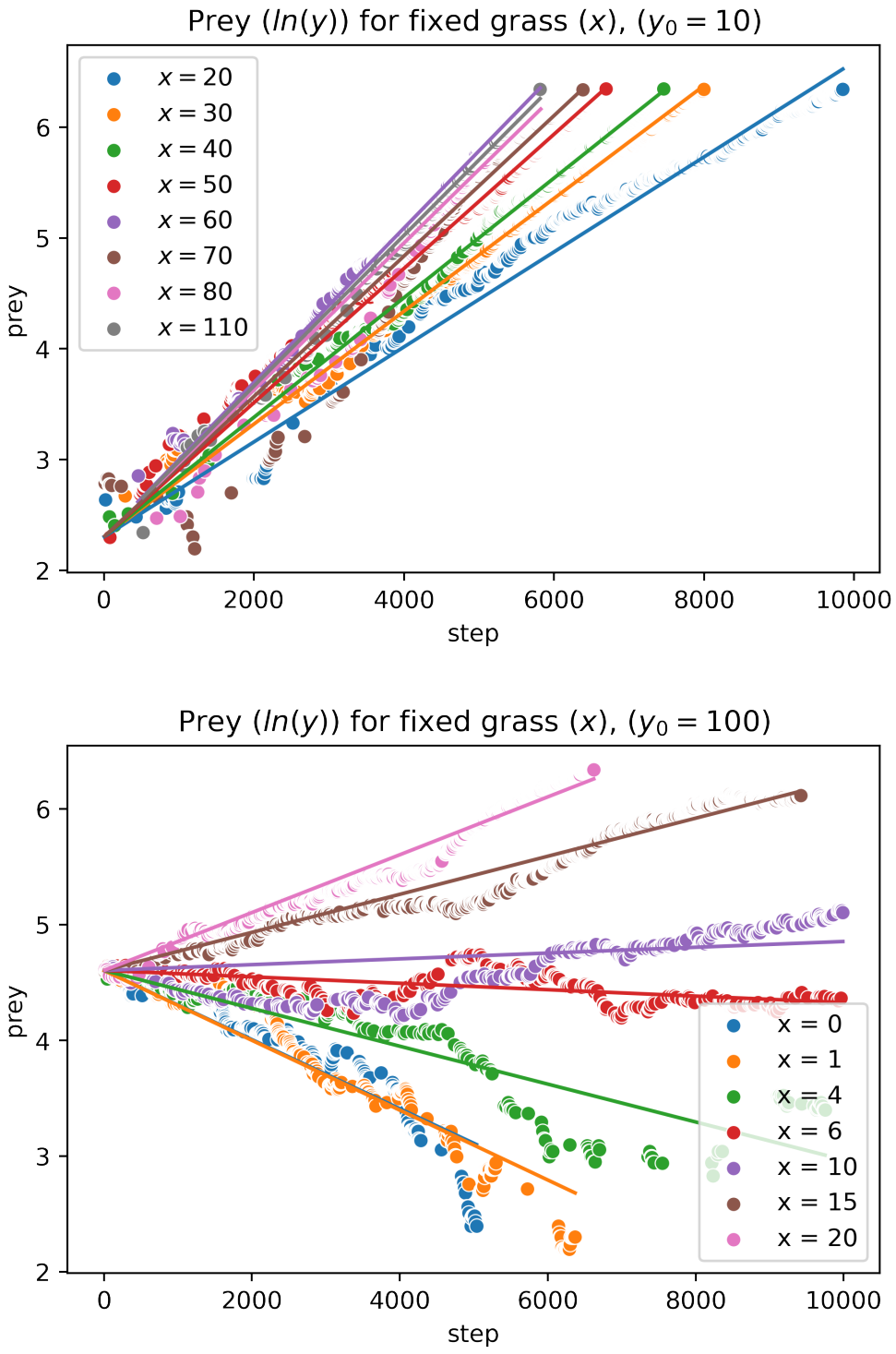

Figure 4.5: The growth and decay of prey in the absence of predation and with a fixed grass density.

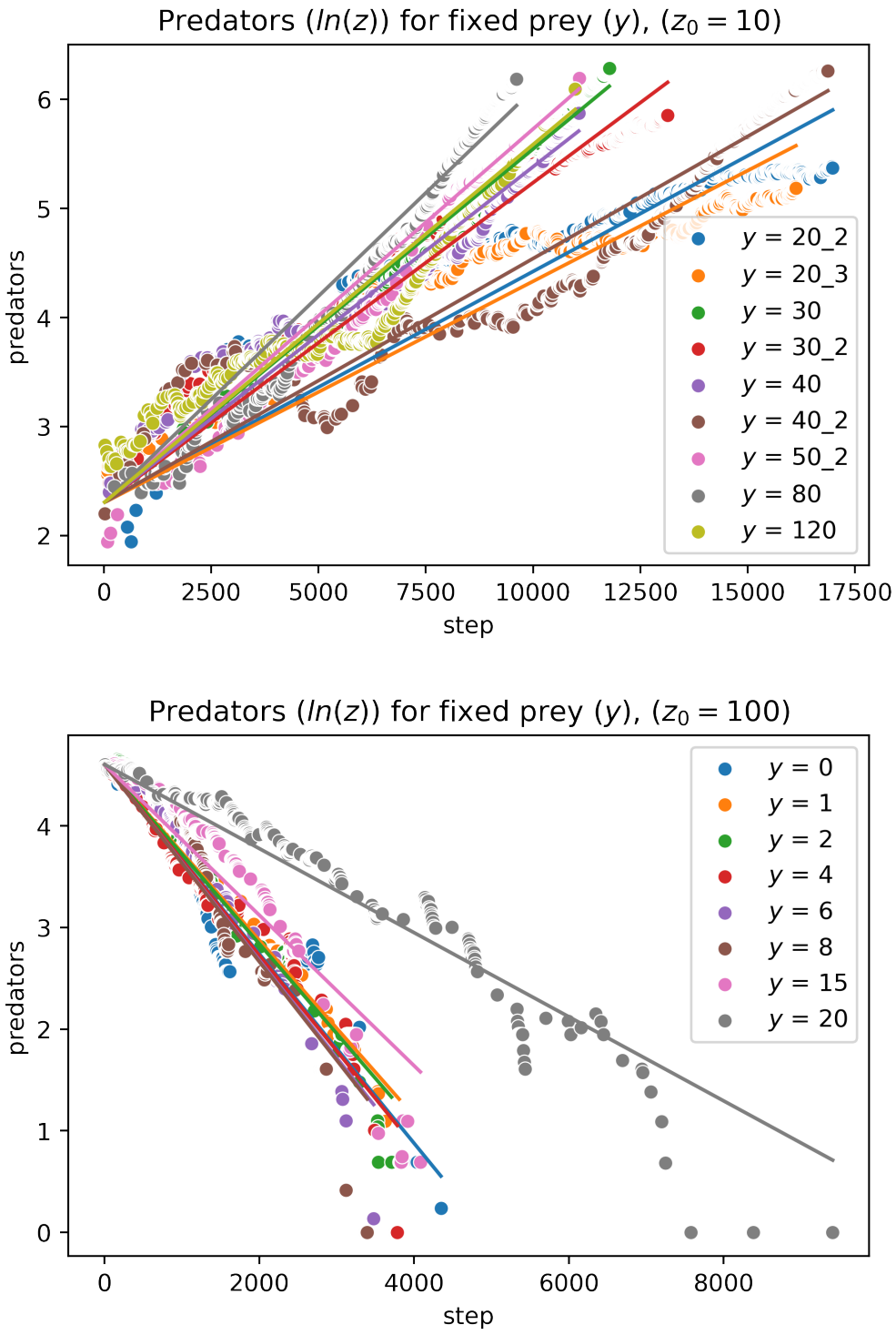

Figure 4.6: The growth and decay of predators with a fixed prey density.

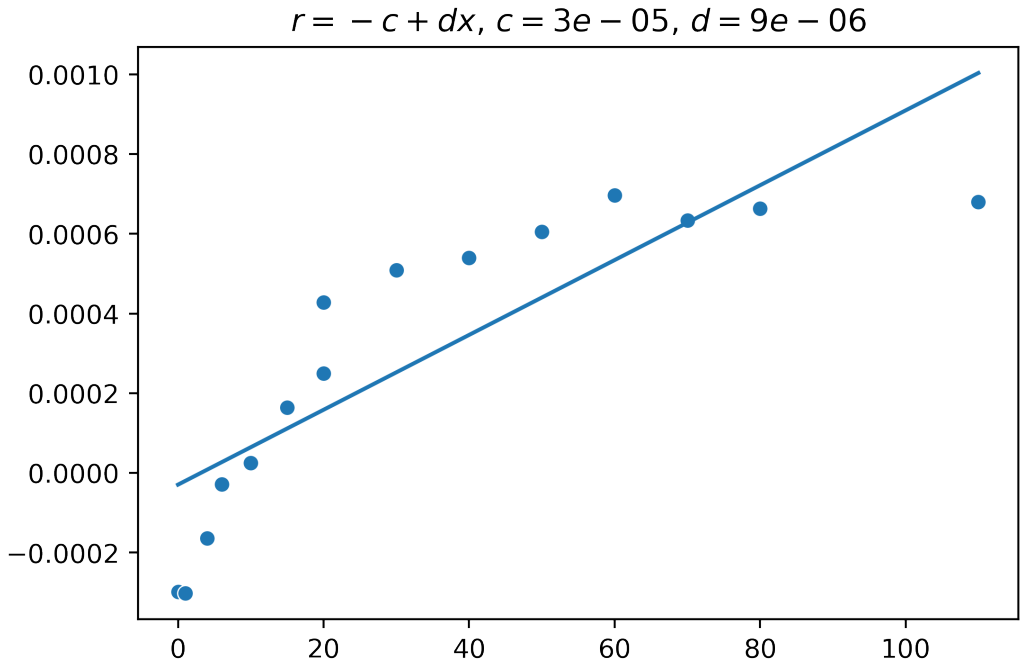

Figure 4.7: Using the exponential growth rate data from fig. 4.5, a linear equation for $r_{\text{prey}} = -c + dx$ is fitted.

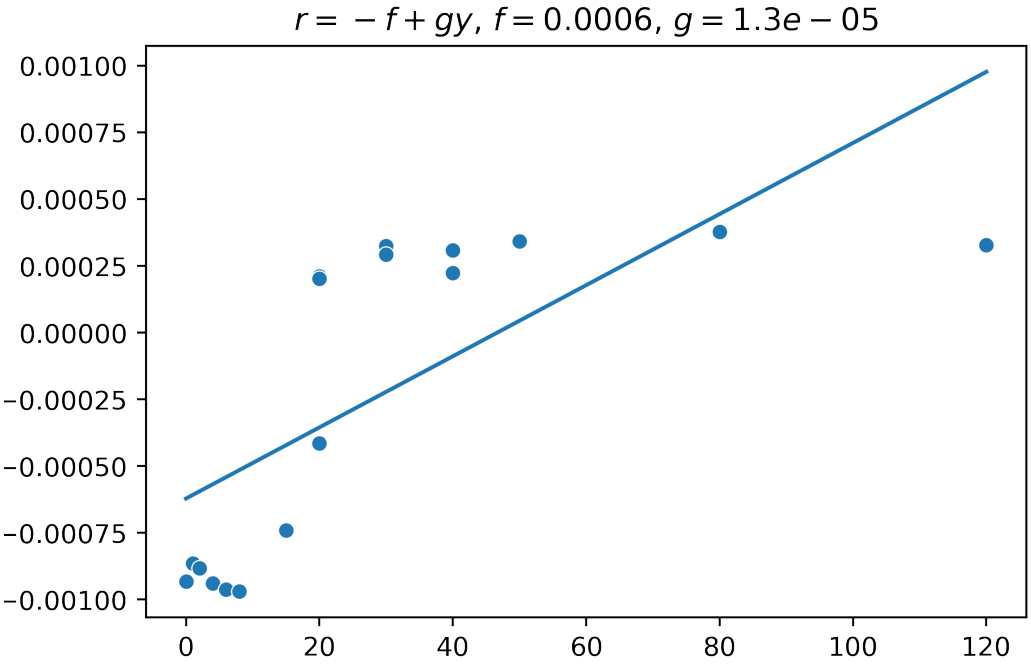

Figure 4.8: Using the exponential growth rate data from fig. 4.6, a linear equation for $r_{\text{predator}} = -f + gy$ is fitted.

4.3 Survival properties

The survival properties of the trained agents are evaluated by comparing them with random policy agents. In fig. 4.9, an environment with a fixed population sizes ($x = 100, y = 20, z = 20$), half of them with a trained policy and the other half with a random, is simulated. The average lifetime of the agents alive at a particular time step is measured for the respective agent type. The values in all coming average lifetime measurements are smoothed with a exponential moving average. The predators outperform the prey in this environment. The trained predators converge at around 5000 timesteps. The random policy predators does not succeed in finding prey and die due to lack of energy. The trained prey performs better than the random policy prey, but both dies at a rather young age due to predation. In fig. 4.10, we have removed the predators. Now the trained prey have mastered the environment and their lifetime grows without tendency to converge at 40000 steps. The random prey does not find any grass and dies due to starvation.
How do the average lifetime changes when the maximum speed $v_{\max}$ is increased? An increased speed typically means a higher metabolism. This way, there is a compromise between being able to move more quickly and a higher rate of energy usage. In fig. 4.11, a minor experiment has been performed where the maximum speed and metabolism is altered in an environment with a fixed amount of predators, which has the maximum speed $v_{\max} = 5$. A prey with a speed of 6 outperforms the slower prey, even when the metabolism is increased in a equal proportion. It seems like this speed advantage enables the prey to outrun chasing predators. The fast prey with a metabolism remaining at 0.2 is however doing better than the fast prey with the increased metabolism at 0.3. The reason why the average lifetimes are not all starting at zero is because the agents don't report the data on every step.
In fig. 4.12, the probability of predation for different speed and metabolism configurations are shown. The prey can die in two ways, from starvation or predation. The probability is found through summation of all deaths from predation divided by the total number of deaths up until a particular step. The fastest prey with a high metabolism has the lowest probability of predation.

4.4 The happiness network

An evaluation of the performance of the happiness network as an reward system is briefly performed. We trained the two animat species with two different reward functions, the happiness network and another where an extrinsic reward of $+1$ was added for the event of eating $\mathcal{E}_{\text{eat}}$ and a negative reward of -5 for dying. These were evaluated on three different fixed population size configurations ($x = 100, y = 50, z = 10$), ($x = 50, y = 50, z = 50$), ($x = 10, y = 50, z = 100$). In the first, the risk of predation is lower due to a lower predator density. Thus, the prey should come a long way with only learning to eat grass. In the second, there is a balance between the two needs. In the third, it is a high predator density. The priority should in the last be to learn escaping predators. This hypothesis is partly confirmed. In fig. 4.13, the extrinsic prey outperforms the one using the happiness network. For

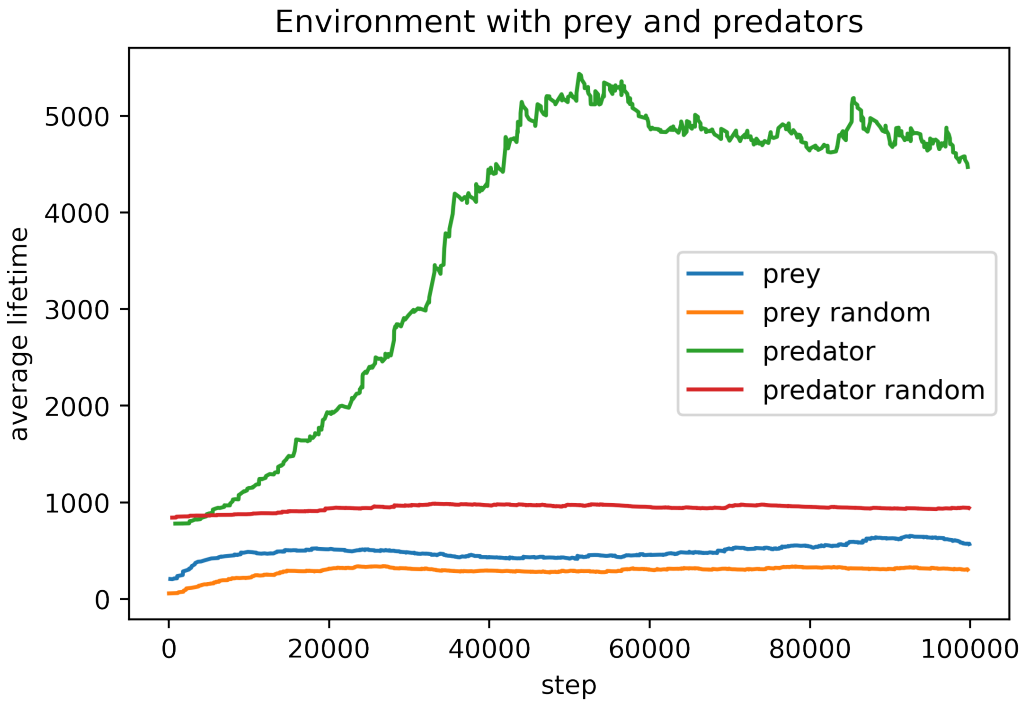

Figure 4.9: Average lifetime of agents alive in an environment with trained and random policy agents of both species.

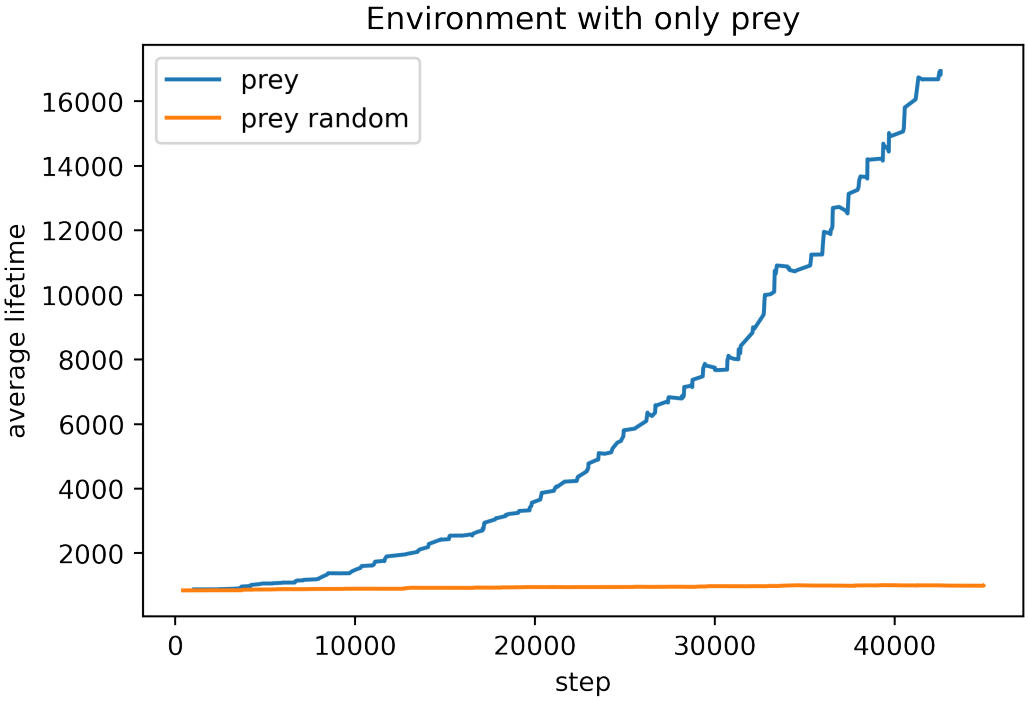

Figure 4.10: Average lifetime of agents alive in an environment with trained and random policy agents in an environment with no predators.

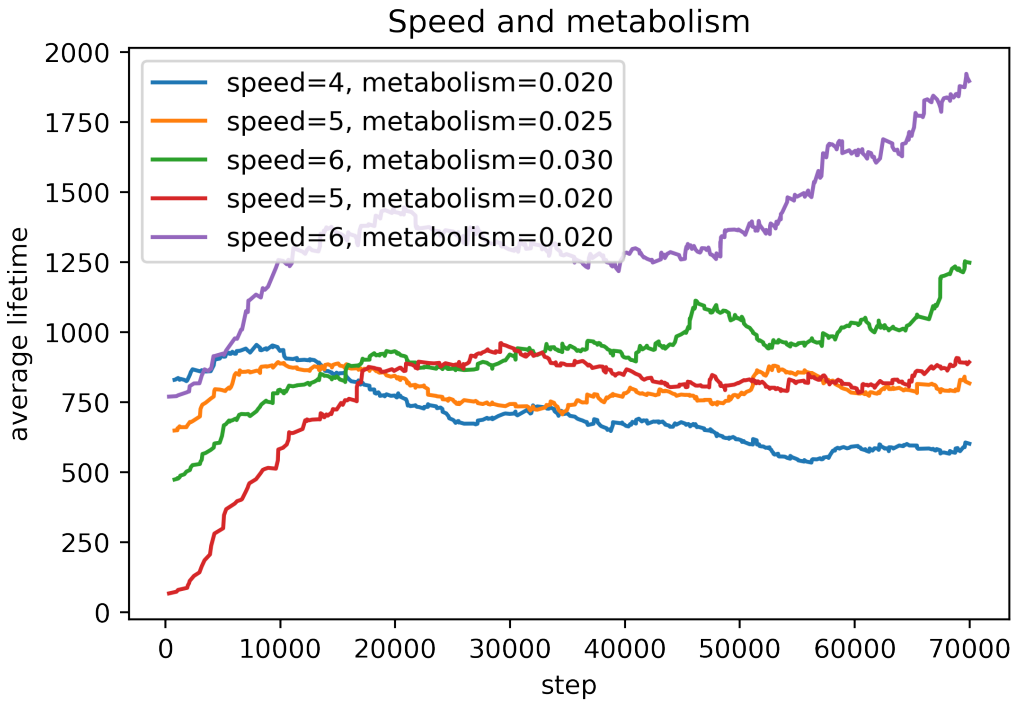

Figure 4.11: Average lifetime of prey with different maximum speed and metabolism configurations.

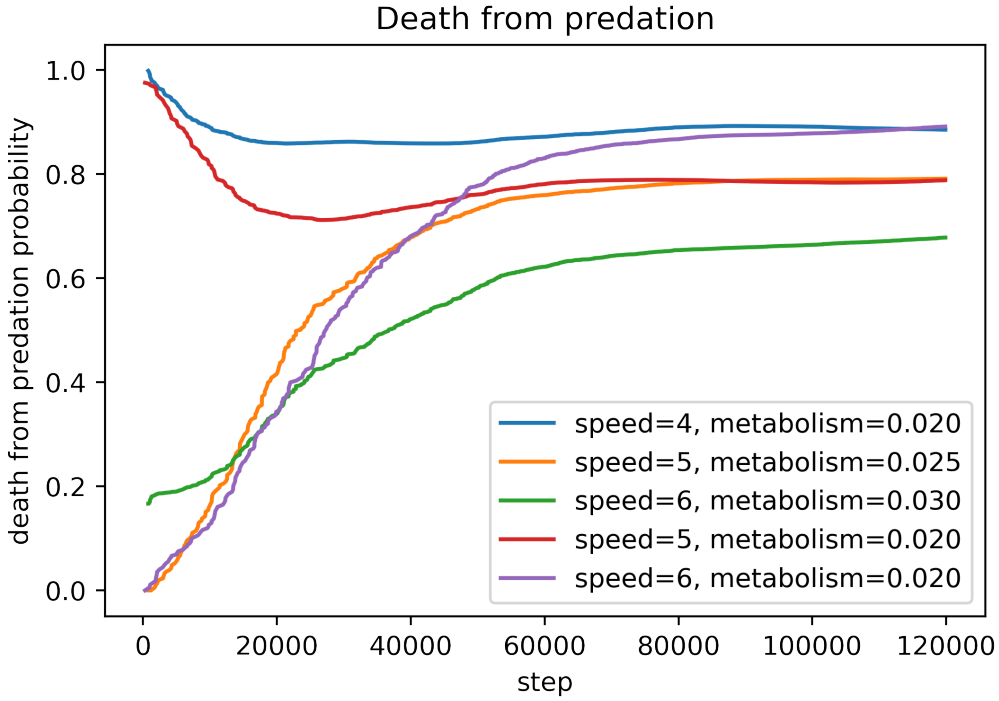

Figure 4.12: The prey can die from starvation or predation. This shows the probability of dying from predation for different speed and metabolism configurations.

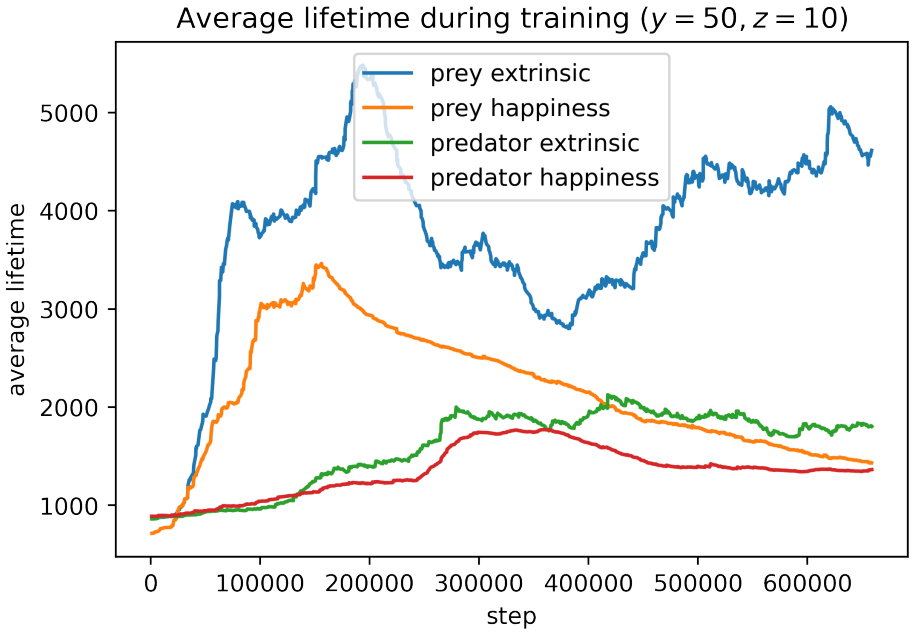

Figure 4.13: Comparison of the average lifetime during training for a pure extrinsic reward signal and the happiness network. In this training environment the predator population is lower than in the other two.

the predators, the extrinsic initially outperforms the happiness, probably due to the more simple reward signal which enables quicker learning. However, by the final timestep their lifetime is similar. For the second configuration in fig. 4.14, the prey which are trained with the happiness network performs better than the one using the pure extrinsic reward. In this environment, a balance between eating and escaping predators is needed and the extrinsic prey, which might have a larger focus on eating grass, has a much greater probability of dying from predation as can be seen in fig. 4.16. The wolf which only needs to find and eat prey, performs better with the extrinsic reward. Lastly, for the third configuration fig. 4.15, the performance of the prey are roughly equal at the final time step. This time, the death from predation probability is greater for the prey trained with the happiness network.

In all three training configurations, the lifetime of the prey for the extrinsic reward system reaches an initial peak and then starts to decrease. This is probably because the wolf needs more training steps to learn to kill prey since they have a harder task of hitting moving targets. In the first training configuration this is also apparent for the prey trained with the happiness network, but not for the other two configurations. This can either be because the prey trained with happiness is better to escape predators or because the predators trained with happiness is not equally good at hunting as their extrinsic counterpart.

The number of training steps differ between the three training environments because they are trained for a fixed amount of time and not the number of steps, and in environments with more agents, the time per step is larger.

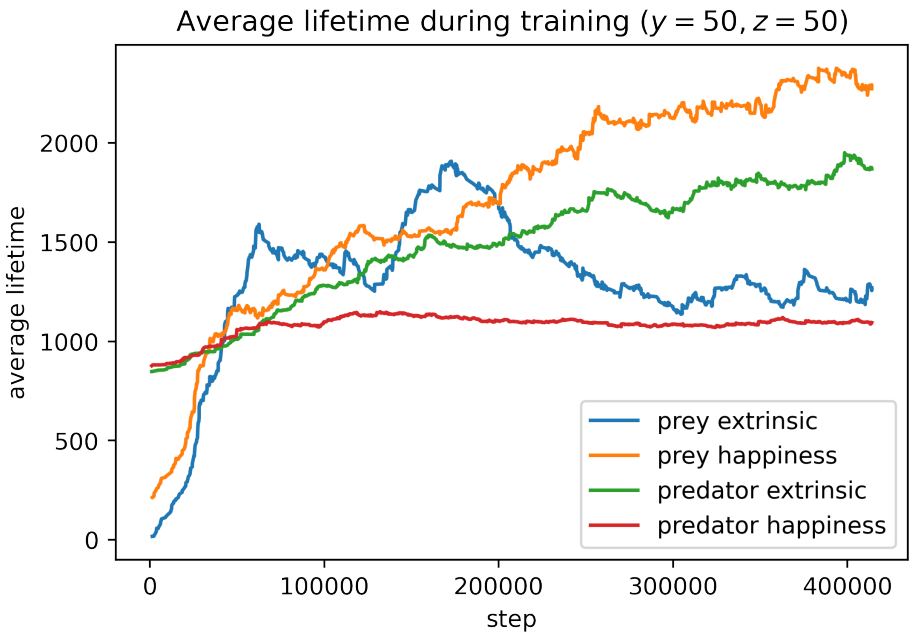

Figure 4.14: Comparison of the average lifetime during training for a pure extrinsic reward signal and the happiness network. In this training environment the predator population is lower than in the third but larger than in the first.

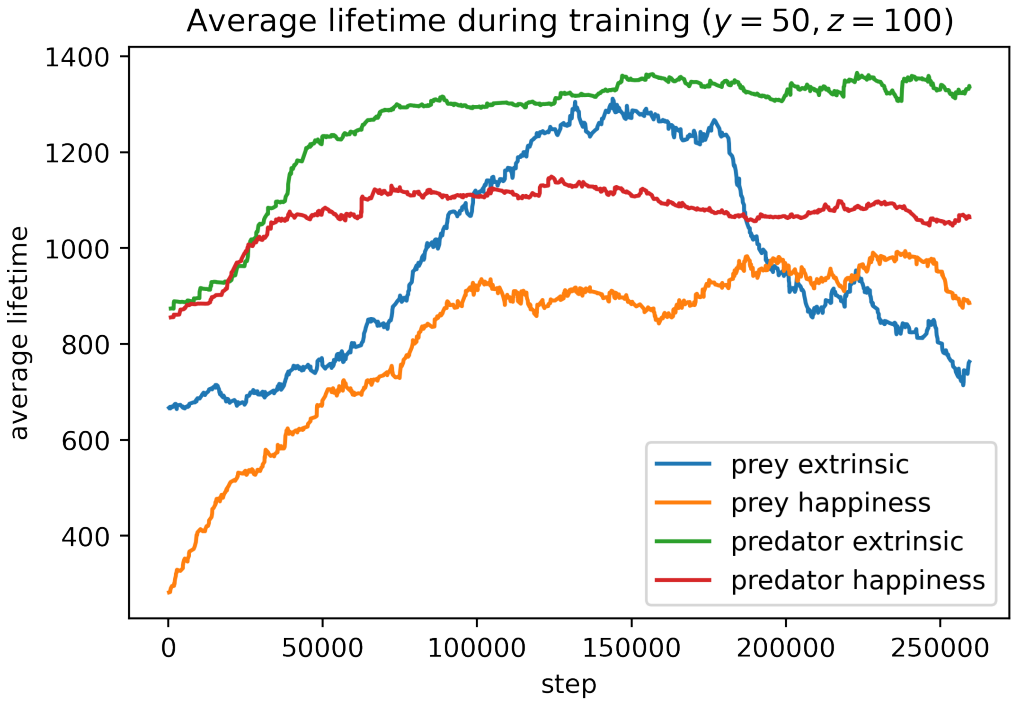

Figure 4.15: Comparison of the average lifetime during training for a pure extrinsic reward signal and the happiness network. In this training environment the predator population is larger than in the other two.

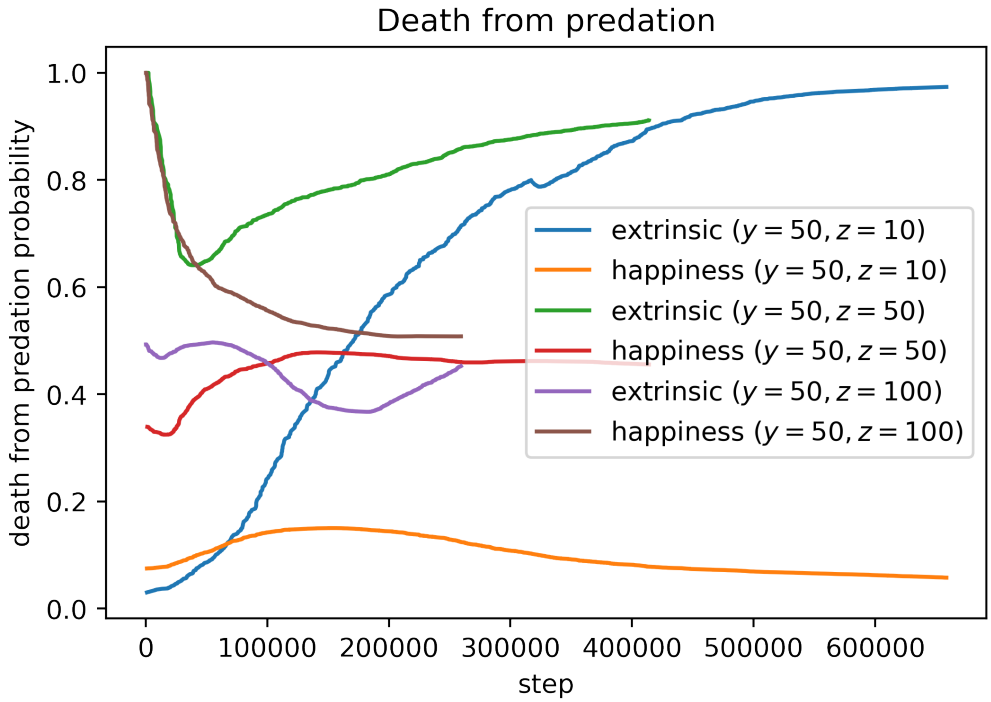

Figure 4.16: Comparison of cause of death for the prey in the three training environments.

5
Discussion

In this section, we will begin by discussing our findings for the main topic of the thesis, three-species predator-prey population dynamics with respect to the LV model and the parameter response estimations. Then the survival properties of the trained agents is discussed and how this could have been further investigated. The comparison of the happiness network with an extrinsic reward system will be considered. A section about RL in simulated ecosystems in general and the design of animats is included in the next section. The chapter ends with a brief discussion about ethical considerations and suggestions of future research.

5.1 Lotka Volterra dynamics

The LV model is a simplified model based on several unrealistic assumptions. Real world data, such as that shown in fig. 2.5, only exhibit accordance with the model to some extent. Therefore, a reflection during this thesis work has been what the actual ground truth is that we strive for in the efforts of trying to obtain realistic population dynamics. Previous work with simulated ecosystems and RL have focused on obtaining LV cyclic dynamics for two-species systems. Thus, having a similar approach for a three-species system seemed like a natural extension for this thesis. The result in fig. 4.1 shows typical LV cycles, which is significantly different than that obtained when using random policy agents in fig. 4.2. This indicates that the intelligence of the agents gives rise to dynamics that are more in accordance with the LV model. However, we can not rule out that the random policy agent simulation would give rise to cyclic dynamics under other simulation parameter choices.
In table 4.1, the amplitude and time period for each species in the four simulation experiments is shown. For each experiment, the period of each species is similar, except for the predators in the second simulation. The amplitudes varies slightly more in each experiment, and has a high variation within each experiment. This indicates that the amplitude of the peaks are more affected by stochasticity than the time period.
When estimating the parameters and their responses to changing population densities, some interesting result are being found which challenges the LV assumptions. Firstly, in fig. 4.4, the simulation measurements indicates that the parameter e, which is the probability for a particular prey to be eaten per time step per predator in the environment, is not constant for different prey densities. There seems like this probability is lower in high prey density environments. In other words, a prey is less likely to be eaten by a predator if there are many other prey. This is what the

more realistic Holling Type II functional response also expresses and makes sense from a real world ecosystem perspective. Typically, this is explained to happen because the time for predators to search prey decreases with increasing prey density, but the time for processing the prey (killing, eating) remains constant. Thus, in high prey density environments, most of the time of the predators is used for the processing, and the predation rate converges. I don't think that this is the primary reason for the observed response in our simulation environment though, as the processing occurs instantly. Rather, I think it is a result of the predators not having infinite appetite. When they feel full, they will not expect an equally large reward from killing prey and may have a smaller drive for hunting. In other words, a fixed predator population will only need to hunt an amount of prey such that its need for food is fulfilled. This is a behavior that the trained predators in the environment seems to exhibit but which the LV model doesn't assume. However, this behavior is not apparent for the prey eating rate parameter b from figure fig. 4.3. Why a similar eating behavior is not observed for the prey is not obvious, possibly it has something to do with the preys need to learn prioritizing eating and escaping predators and that this interferes with the learning of not needing to eat when already full.
Another interesting insight from the parameter estimation is the numerical responses for the growth and death parameters c, d and f, g of the prey and predators respectively. The assumption of the LV model is that the rate of population growth increases linearly with the amount of available food (grass and prey). In fig. 4.7 and fig. 4.8 this doesn't seem to be the case for our simulations. Rather, there seems that the growth rate converges at a food population of around 20 for both the prey and the predators. This also seems like a reasonable response. If a population has a sufficient amount of food available such that none starves, the population will not grow quicker if it is provided with additional food. This is the dynamics that our simulations seem to exhibit.
In the efforts of making better LV model variations, many more complex functional and numerical responses have been suggested. However, there is a trade-off between model complexity and simplicity. This kind of simulation may offer a natural way to explore responses in more complex ecosystems since real world data of the requested type may be limited. Through better response estimation, one might be able to model ecosystems more accurately. Ecosystem simulations like this might also be an alternative to directly simulate population dynamics without a need for constructing a mathematical model.

5.2 Survival properties

In fig. 4.9 and fig. 4.10, it is very clear that the trained animats have a much longer expected lifetime than the random agents. The first of the two figures shows that at a predator population of 20, both the trained and random policy prey will struggle and die from predation at a young age. The lifetime of the predators converges to around 5000 steps. The reason why the average lifetime converges and don't seem to grow without limit like the trained prey do in fig. 4.10, is probably because the prey density is too low. This makes it possible for predators to not find prey since they only partially observe the environment. This is not the case in fig. 4.10 since

the grass density is higher, and the prey seem to always be able to find grass before they starve.
Increasing the maximum speed of the prey increases its expected lifetime for this particular population density configuration. A higher maximum speed enables the prey to escape chasing predators, and as can be seen in fig. 4.12, death from predation is the most probable death cause. However, it seems to also be useful for finding grass. Comparing the slowest and fastest prey with equal metabolism, we can see that their death from predation probability is equal while the life expectancy of the later is significantly longer. This way, the number of deaths from both starvation and predation is lower for the faster predator. For the fastest prey with a metabolism of 0.02 and 0.03 respectively, the probability of dying from starvation is larger for the later, this is an expected result.
It would be interesting to make a more thorough analysis of the survival properties for different animat and ecosystem configurations. In real life animats, there is a compromise between moving quickly and usage of energy. The actual relationships between energy consumption and movement have not been studied for this analysis. The optimal configuration is probably depending on the environment in which the animal lives. These kinds of simulations may be able to give additional insights into questions like this.
Comparing the trained agents with random policy agents could have been further investigated in several ways. For example, a rule based agent which moved greedily towards the closest food object. Or agents trained with other RL algorithms through other MARL schemes. One could also compare agents with different sensors and reward systems, the later which is done briefly in section 4.4. An issue for these comparisons though is how to make fair comparisons. The comparison of lifetime for a particular agent should ensure that the environment in which it lives should be equal for all agents. And the conclusions made from such an experiment is maybe not valid for environments that differs from the measured one. The population dynamics simulations are non-stationary with changing population densities. Thus, the optimal policy may be different at different stages of the simulation and the analysis of which policy is the best is thus a time dependent question. Changing the physical configuration of an animat, such as increasing its speed or sensory range, changes what the animat can expect to happen given an action under a particular observation. Therefore, properties of the animat can't be perturbed too far away from the setup at which it has been trained without an expected loss of performance.

5.3 The happiness network

The main argument for the happiness network is the idea that the reward is greatest when actions are selected which improves the homeostatic variables that are furthest from their optimal values. This way, the animat learns to prioritize actions depending on its internal state. In comparison, a simple reward signal such as always rewarding eating is probably good to ensure that the energy homeostatic variable is fulfilled, but when multiple different homeostatic variables needs to be fulfilled, something more sophisticated might be useful. The performance of the happiness network was compared with a simple extrinsic reward system. For the

prey in the high grass density environment with few predators, the extrinsic reward performed very well, as expected. When the predator density was increased and the prey needed to both escape predators and eat grass, the happiness network yielded prey that survived the longest. This difference was not as large when the predator density was further increased. The reason for this might be that focusing on only escaping predators might be the best priority here, and the extrinsic trained prey learned to only focusing on this due to their -5 reward for death. The happiness network maybe instead learned that when the prey was hungry, the most urgent threat was from starvation and ignored the predators to some extent, which decreased the lifetime since they were actually the greatest threat.
The advantages of the happiness network are not really evaluated in this thesis due to using animats with only one homeostatic variable. Thus, the usage of it for optimal performance in this particular environment may be argued against. Further, the happiness network section 2.3.2 consists of both a response for the homeostatic variables which could be different for different variables, as well as a response to the sensory observation. And both responses includes different weights which may be crucial to obtain a optimal policy. Both of which are not tuned for optimal performance due to the complexity of training a MARL system, but rather chosen through reasoning. Another master thesis in our research group had the design of an happiness network as their primary topic.

5.4 Animats and ecosystem simulations

Even though the animats and the ecosystem simulation in this thesis are more complex than many previous ecosystem simulators, it is only in its infancy when it comes to modelling real world ecosystems. However, introducing additional homeostatic variables, sensory inputs, environmental complexity, etc. is not a far leap and it is possible that animats can be trained to survive in much more complex environments. After working with this thesis, it is our belief that in order to obtain intelligent behavior in more complex ecosystems, an appropriate reward system design is crucial. This reward system needs to ensure that the desired policy (the one which resembles real world animal behavior), is the one which maximizes the discounted cumulative reward. This seems to be a harder task than one might think. If the reward is too great for some subtask of the desired behavior, the animat learns to exploit this. For example, if the weights for happiness from smell is too great, it may learn that chasing good smells should be the primary goal, which is not necessarily the best behavior for survival. Another example, if the negative reward for the smell of predators are too great, the prey may only focus on avoiding predators and ignore their need for eating. Vice versa if the reward for fulfilling the homeostatic needs is too great, it may not prioritize escaping predators. When multiple homeostatic variables are introduced, this gets even more complex. Which homeostatic variable is most in need of being fulfilled now and to what extent of doing so does the environment currently provides? How to design a reward system which enable appropriate learning is not a straight forward task, but luckily, there are real-world animals to look for answers and this is what has been done in this thesis by introducing the happiness network. Another possibility is to use evolutionary tactics to

find happiness networks which yield animats with the best survival properties.
Reproduction is another concept that needs to be implemented to the ecosystems. In this thesis, asexual reproduction was used, which is not very realistic in a agent based ecosystem. More realistic reproduction was a topic of another master thesis in our research group. Here, reproduction through physical interaction and the physical growth of animats from infants to adults were implemented. The growth of grass was also done in a more realistic manner and eating grass done by prey by applying its eating action to it and multiple prey could share a grass resource until it was consumed. Similarly, when the predators killed a prey, it left a meat object behind which multiple predators could apply its eating action to until the meat source was consumed. They also used genetic algorithms to simulate evolution of physical properties.
Another feasible ideas for predator-prey ecosystems is to introduce fighting, which was done briefly by [5]. Terrain dependent movement and obstacles, marine environments, seasonal variations, communication of observations, etc. are all ideas that can be implemented in a rather straight forward way in the Unity environment. Whether training animats in those are a straight forward task or not is however a question which is harder to answer. This is one of the most interesting parts of simulated ecosystems of this kind, you can incrementally make them more realistic and at some point you might actually have agent based simulations where both the animats and the environment resemble their real-world counterpart.

5.5 Limitations

Using Unity as a simulation environment gives many advantages, such as an existing framework for building the environment in 3D, utilizing a physics engine and a continuous space where most previous work has been done in a gridworld. This enables qualitative analysis of the animat behaviors. However, it also has some limitations such as the speed of running simulations in comparison with less rich simulation environments. This way, exploring the simulation environment parameter space is very computationally demanding. Further, increasing the simulation environment size, which might yield less stochasticity in the population dynamics is not feasible since the speed decreases further and Unity seemed to be limited to deal with around 500 alive agents simultaneously, which is not a lot for population dynamics experiments.
The LV model has been studied in detail for a long time. Searching for the topic on google scholar yield 70 000 hits. Thus, finding relevant work and what has already been done is challenging. However, when it comes to LV in connection with RL, which is a small subset of all the work that have been put into analysing the LV model, we believe that most relevant work have been considered.

5.6 Ethics

This type of simulations might equally well be used to simulate human behavior in different environments. If the simulations at some point become realistic enough,

which they are far from being right now, there might be concerns as to how the simulations can be used for ill-intended purposes. Such as exposing realistic simulated humans to some scenario of interest in the simulation environment and be able to more accurately predict the outcome. Similarly, one might be able to train robots to do harm in the simulated environment and then launch them into the corresponding real-world ecosystem.
Another concern may be if these types of simulations are used at an too early stage to predict the outcome when one alters an ecosystem, when the simulation is not yet capable of doing so. An overly optimistic belief regarding the current capabilities of artificial intelligence is not uncommon today and I consider this scenario as a threat that is not unlikely to happen. Using ecosystem simulations that might be ill-intended and biased or with good intentions but inaccurate to motivate interventions to the environment which might have a negative impact.

5.7 Future research

Many ideas regarding how the animats and the ecosystem can be more realistic was brought up in section 5.4. As mentioned there, it is an seemingly unbounded number of real-world inspired ideas that can be introduced to the simulated ecosystems. However, the major problem is to train intelligent animats in increasingly complex environments. For this, there might exist better learning algorithms than that of this thesis. Model-based RL has seen a recent upswing and might be a promising direction to explore. As said in section 5.3, an appropriate design of reward system is crucial and becomes a more complex task as the animats have multiple objectives. How different sensory inputs complement each other, how they can be pre-processed for easier interpretation and their relation to RL may also be a topic for future research. Real-world animals has a large number of different sensory inputs which all are used to improve the survival properties of the animal.
Specifically for the studies of LV, there are several interesting further research questions. A similar study for the competitive LV model, where several animals compete over common resources of food, may be performed. Estimations of the functional and numerical responses of the parameters seems feasible in simulated ecosystems like this. Using it to estimate even more complex functional responses and their relationships with the current state-of-the-art mathematical models may be a future research topic.

6
Conclusion

In this work, we set out to investigate the population dynamics of a three-species predator-prey system with agents trained with multi agent reinforcement learning and compare the dynamics with the three-species Lotka Volterra predator-prey equations. It was shown that the dynamics of the trained agents exhibit the characteristic Lotka Volterra cycles when random policy agents did not. The functional and numerical responses of the parameters in the Lotka Volterra model was analysed and it was found that the responses in the simulated ecosystems did not fulfill the assumptions of the model, but rather showed other responses that could be argued to be more realistic.
The survival properties of the animats were analysed through comparison with random policy agents and by comparison of different physical configurations. It was shown that increasing the movement speed of the prey gave a longer expected lifetime, even when the rate of energy consumption (metabolism), was increased at an equal proportion. These type of simulations was argued to be useful for understanding the adaptation of properties in real-world animals.
A reward system which incorporated both the external and internal state of the animat was introduced as the happiness network. This reward system was shown to perform better in some environments than a simple reward system with positive reward for eating food and a negative for dying.

Bibliography

[1] S.W. Wilson. The animat path to ai, 1991.

[2] Donald L DeAngelis and Volker Grimm. Individual-based models in ecology after four decades. *F1000prime reports*, 6, 2014.

[3] Theodosius Dobzhansky. Nothing in biology makes sense except in the light of evolution. *The american biology teacher*, 75(2):87–91, 2013.

[4] Yaodong Yang, Lantao Yu, Yiwei Bai, Jun Wang, Weinan Zhang, Ying Wen, and Yong Yu. A study of ai population dynamics with million-agent reinforcement learning, 2018.

[5] Jun Yamada, John Shawe-Taylor, and Zafeirios Fountas. Evolution of a complex predator-prey ecosystem on large-scale multi-agent deep reinforcement learning, 2020.

[6] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps, 2017.

[7] Peter Sunehag, Guy Lever, Siqi Liu, Josh Merel, Nicolas Heess, Joel Z. Leibo, Edward Hughes, Tom Eccles, and Thore Graepel. Reinforcement learning agents acquire flocking and symbiotic behaviour in simulated ecosystems. *Artificial Life Conference Proceedings*, (31):103–110, 2019.

[8] Seongdong Kim, Christoph Hoffmann, and Varun Ramachandran. Analyzing the parameters of prey-predator models for simulation games. In *International Conference on Entertainment Computing*, pages 216–223. Springer, 2010.

[9] Mehdi Keramati and Boris Gutkin. A reinforcement learning theory for homeostatic regulation. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[10] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, and et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, Dec 2020.

[11] E. Neftci and B. Averbeck. Reinforcement learning in artificial and biological systems. *Nature Machine Intelligence*, pages 133–143, March 2019.

[12] Itamar Arel. *Deep Reinforcement Learning as Foundation for Artificial General Intelligence*, pages 89–102. 01 2012.

[13] Markov property. Markov property— Wikipedia, the free encyclopedia, 2020. [Online; accessed 10-january-2021].

[14] Richard Bellman. A markovian decision process. *Indiana Univ. Math. J.*, 6:679–684, 1957.

[15] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.
[16] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
[17] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms, 2019.
[18] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33(6):750–797, Oct 2019.
[19] Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. Springer Publishing Company, Incorporated, 1st edition, 2016.
[20] Kaiqing Zhang, Alec Koppel, Hao Zhu, and Tamer Başar. Global convergence of policy gradient methods to (almost) locally optimal policies, 2020.
[21] Boyi Liu, Qi Cai, Zhuoran Yang, and Zhaoran Wang. Neural proximal/trust region policy optimization attains globally optimal policy, 2019.
[22] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.
[23] Alfred J. Lotka. Contribution to the theory of periodic reactions. *The Journal of Physical Chemistry*, 14(3):271–274, 1910.
[24] Vito Volterra. Fluctuations in the abundance of a species considered mathematically 1, 1926.
[25] Erica Chauvet, Joseph E. Paullet, Joseph P. Previte, and Zac Walls. A lotka-volterra three-species food chain. *Mathematics Magazine*, 75(4):243–255, 2002.
[26] M. E. Solomon. The natural control of animal populations. *Journal of Animal Ecology*, 18(1):1–35, 1949.
[27] Michael L Rosenzweig and Robert H MacArthur. Graphical representation and stability conditions of predator-prey interactions. *The American Naturalist*, 97(895):209–223, 1963.
[28] C. S. Holling. Some characteristics of simple types of predation and parasitism. *The Canadian Entomologist*, 91(7):385–398, 1959.
[29] Charles Elton and Mary Nicholson. The ten-year cycle in numbers of the lynx in canada. *Journal of Animal Ecology*, 11(2):215–244, 1942.
[30] Claes Strannegård, Wen Xu, Niklas Engsner, and John A. Endler. Combining evolution and learning in computational ecosystems. *Journal of Artificial General Intelligence*, 11(1):1 – 37, 01 Jan. 2020.
[31] Wolfram Schultz, Peter Dayan, and P. Read Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997.
[32] John Eric Rayner Staddon. *Adaptive behavior and learning*. Cambridge University Press, 2016.