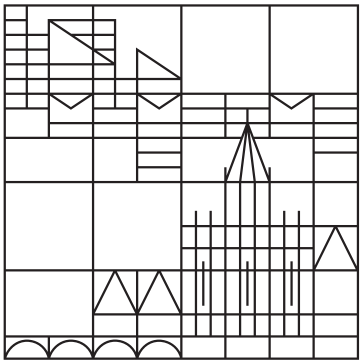
MAX PLANCK INSTITUTE
OF ANIMAL BEHAVIOR

Universität
Konstanz



Exploring Predator-Prey Dynamics from Videos using Generative Adversarial Imitation Learning

Jannik Wirtheim
Matricle No.: 1398359
M.Sc. Social and Economic Data Science

Department of Computer and Information Science
Chair of Cyber-physical Systems

1. Supervisor:	Prof. Dr. Heiko Hamann
2. Supervisor:	Prof. Dr. David Garcia
External Supervisor:	Dr. Liang Li

`https://github.com/wirthy21/Master-Thesis`

Konstanz, 2026

Jannik Wirtheim

Exploring Predator-Prey Dynamics from Videos using Generative Adversarial Imitation Learning

Abstract

Collective motion in swarm systems exhibits complex behavioral patterns emerging from local interactions and external influences. Learning policies that reproduce such dynamics remains challenging, especially in predator–prey settings where heterogeneous roles co-evolve competitively. This thesis investigates whether predator–prey behavior can be learned directly from real aquarium recordings of one predator and a school of prey. Expert trajectories are extracted using a customized detection and tracking pipeline and transformed into structured tensors. Separate modular policies for predator and prey are trained co-evolutionarily within a Generative Adversarial Imitation Learning framework using evolutionary strategies, with role-specific discriminators operating on latent transition representations learned by a self-supervised transition encoder. Performance is evaluated using swarm-level metrics, complemented by Monte-Carlo trajectory prediction to analyze uncertainty and policy visualizations to explain behavior in specific situations. Results show partial imitation: the learned policies reproduce basic group motion, but fail to capture inter-group dynamics such as consistently pursuit and coordinated avoidance. The thesis discusses causes, including data imbalance and architectural choices, and outlines directions to improve robust predator–prey imitation from videos.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AN	Attention Network
BC	Behavioral Cloning
DeepSORT	Deep Simple Online and Real-Time Tracking
EMA	Exponential Moving Average
ES	Evolutionary Strategy
GAIL	Generative Adversarial Imitation Learning
GAN	Generative Adversarial Network
IL	Imitation Learning
IRL	Inverse Reinforcement Learning
JS	Jensen–Shannon divergence
MARL	Multi Agent Reinforcement Learning
MC	Monte Carlo Simulation
MDP	Markov Decision Process
MLP	Multilayer Perceptron
MSE	Mean Squared Error
PIN	Pairwise Interaction Network
RL	Reinforcement Learning
RMSProp	Root Mean Square Propagation
VICReg	Variance-Invariance-Covariance Regularization
YOLO	You Only Look Once

CHAPTER 1

Introduction

Collective animal behavior represents one of the most fascinating examples where complex global patterns emerge as a result of simple interaction rules among individuals. Understanding the principles that drive coordinated movements holds important implications for research fields such as robotics, biology, and evolutionary science (Li et al., 2023; Yifan Wu et al., 2025). Early approaches to modelling collective motion relied on hand-crafted interaction rules (Couzin et al., 2002; Reynolds, 1987; Vicsek et al., 1995), with the ability to reproduce basic collective patterns, while leaving a gap in capturing the dynamics found in nature in all their complexity. With time, novel data-driven approaches to imitation learning (IL) enabled to derive complex movement patterns directly from observations. However, the existing frameworks are often limited to single-species groups (Heras et al., 2019; Yapei Wu et al., 2024), synthetic demonstrations (Ishiwaka et al., 2022; Yapei Wu et al., 2024; Yu et al., 2021) and without investigating the influence of survival pressure on emergence (Heras et al., 2019; Yapei Wu et al., 2024). Adversarial imitation learning between predator and prey agents based on real behavioral data remains largely unexplored. This thesis addresses this gap by developing a co-evolutionary imitation learning framework to model and reproduce the predator–prey dynamics observed in real fish observations. Real predator and prey recordings are processed using customized object detection and tracking. These trajectories are transformed into structured state–action representations, which serve as expert demonstrations for imitation. Within the proposed framework, separate predator and prey policies are trained adversarially against their own discriminators. Both policies co-adapt through alternating optimization using an evolutionary strategy. The learned policies are evaluated through comparisons with the expert demonstrations and generated trajectories from the Couzin model. Swarm metrics such as polarization and angular momentum are used to evaluate the behavior patterns achieved by the learned agents. Furthermore, the modular structure of the policies enables attention-based analyses of swarm composition, making the policies' decision-making transparent and providing insights into swarm leadership. Trajectory prediction provides further insights into learning success. In summary, this thesis contributes to the study of collective behavior in competitive systems by imitating predator–prey behavior directly from real biological data. While basic group motion patterns and single-species behavior can be reproduced, the model struggles to capture inter-

group dynamics such as sustained pursuit and coordinated avoidance. To address these limitations, the thesis discusses data- and model-related directions specifically aimed at recovering the missing inter-group dynamics. Beyond these limitations, it outlines key challenges of adversarial predator–prey imitation from video data and motivates promising approaches to overcome them. It is structured as follows: Chapter 2 reviews related literature and positions this study within the context of existing research. Chapter 3 describes the data collection and processing pipeline. Chapter 4 introduces the co-evolutionary GAIL framework and chapter 5 the training procedure. Chapter 6 presents the experimental results, including the leadership analysis and trajectory prediction. Chapter 7 highlights the work from an critical perspective. Finally, chapter 8 summarizes the findings and outlines potential future directions.

CHAPTER 2

Theoretical Background

Collective behavior is all around us. From a microscopic point of view, it can be observed in how interacting particles behave (Vicsek et al., 1995), to a more macroscopic perspective on the dangers of massive human crowds following uncontrolled collective motions (Gu et al., 2025). Even the scaling of large language models shows the phenomenon of emergent abilities (Wei et al., 2022), defining emergence as "when quantitative changes in a system result in qualitative changes in behavior" (Wei et al., 2022, p.2).

2.1 Collective Behavior in Nature

One of the most common association with collective behavior is its occurrence in nature, beautifully manifested in formations such as bird flocks, schooling fish, or trails of ant workers (Couzin et al., 2002; Reynolds, 1987; Vicsek et al., 1995). Even though the individuals act solely based on their own local perception, the swarm leads to a randomly arranged yet synchronized collective motion that resembles the flow of a fluid (Reynolds, 1987). Changes in direction and shape appear as the movement of a single coherent entity (Couzin et al., 2002). Following the notion of self-organized systems, this order emerges intrinsically through local interactions among its components (Di Marzo Serugendo et al., 2003). But the order is also influenced indirectly via the environment, which is continuously shaped by and, in turn, effects the system's dynamics. Foundational research demonstrated that such self-organized behavior can arise solely from three simple local rules: alignment, attraction, and repulsion (Couzin et al., 2002; Reynolds, 1987; Vicsek et al., 1995). Alignment refers to the tendency of individuals to match their direction with their neighbors, attraction describes the movement toward others to maintain group cohesion and repulsion ensures collision avoidance. Building upon these insights, Reynolds (1987) Boids model successfully simulated bird flocks, Couzin et al. (2002) extended it to model fish schooling and Vicsek et al. (1995) demonstrates particle movements. While these models successfully reproduce basic swarm patterns, their hand-crafted rules remain heuristic and static, unable to capture the adaptive complexity of real biological systems. However, animal collectives adapt dynamically to

changing environments. Li et al. (2023) showed that swarming behavior does not necessarily require explicit alignment rules, as survival pressure alone can drive the emergence of coordinated movement in predator–prey systems. This highlights the external influence of environmental factors in shaping collective behavior and further illustrates the challenge of accurately modeling swarm dynamics through simple interaction rules that neglect such external influences. Addressing these limitations, advances in imitation learning enable agents to learn collective motion patterns directly from real demonstrations (Song et al., 2018; Yifan Wu et al., 2025), thereby bridging the gap between biological observation and computational modeling.

2.2 Imitation Learning

The approach of IL builds upon the foundations of reinforcement learning (RL). Both learning frameworks can be formally described as a Markov Decision Processes (MDPs), where decision-making involves partially stochastic outcomes under the control of an agent (Pierson and Kao, 2025). Mathematically, an MDP is defined by a set of states S and actions A. A transition model $P(s'|s,a)$ describes the probability that an action a in state s leads to a subsequent state s', and an unknown reward function $R(s,a)$. The overall objective is to optimize this reward function in order to find the optimal policy π^*. To serve increasing environmental complexity and unstructured settings, manually specifying an agent's reward, as done in traditional RL, becomes extremely difficult (Zare et al., 2023). In such environments, it is necessary to flexibly adapt to changing situations, making it unfeasible to specify an optimal set of rules that address all possible cases. Even though experts who manually code the rules must have applicable knowledge about the experts behavior. The main advantage of IL lies in its ability to learn the expert's desired behavior from demonstrations rather than through explicitly defined reward functions. Following this approach, various algorithms have been developed that differ in their underlying assumptions and must therefore be distinguished in the context of this study's experimental design.

2.2.1 Behavioral Cloning

One of the most basic approaches in imitation learning is behavioral cloning (BC), which learns the expert's policy in a purely supervised manner (Heras et al., 2019; Schaal, 1996; Torabi et al., 2018). Given states and corresponding expert actions, the objective is to minimize the discrepancy between the expert's actions and those produced by the learned policy. However, despite its conceptual simplicity, BC requires a large amount of demonstration data and suffers from the problem of

compounding errors. BC implicitly assumes that state–action pairs are sampled independently and identically distributed (i.i.d.) (Lőrincz, 2019; Ross et al., 2011). In an MDP, state–action pairs are generated sequentially. Meaning that an action a_t taken in state s_t affects the next state s_{t+1}. As a result, consecutive samples are temporally correlated and therefore not independent, violating the independency assumption. From a practical perspective, small prediction errors accumulate over time, eventually driving the agent into states that were never visited by the expert and therefore never seen during training. Once the agent enters such unfamiliar states, no corrective signal is available, causing the errors to amplify further. This makes the algorithm applicable only to short-term scenarios, where the expert's demonstrations cover most of the state space and compounding errors do not result in critical failure. A well-known failure case of BC arises in autonomous driving (Lőrincz, 2019; Sun et al., 2024). Models trained primarily on expert lane-keeping demonstrations can accumulate small prediction errors, drift into unseen states, and eventually leave the road.

2.2.2 Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL) inverts the standard RL problem (Arora and Doshi, 2020). While RL aims to learn an optimal policy that maximizes a known reward function, IRL assumes that such a reward function is not explicit, but implicitly contained in the expert demonstrations. The goal of IRL is to recover this underlying reward function that best explains the observed behavior, while avoiding its manual specification. After the reward is recovered, an optimal policy is learned with respect to this function, yielding an imitation policy (Torabi et al., 2018). IRL can be further divided into model-given and model-free IRL. In a model-given setup, the environment is known, allowing the reward function to be recovered through iterative planning methods that rely on evaluating full trajectories (Lőrincz, 2019). Model-free approaches, on the other hand, are suited for more complex and partially unknown environments. Instead of relying on explicit knowledge of the environment dynamics, they learn the policy directly from interactions with the environment, updating it incrementally based on feedback from each step. For example, Naranjo-Campos et al. (2024) applied IRL to robotic arm manipulation using expert trajectories, aiming to develop efficient and adaptable techniques for solving complex tasks in continuous state spaces.

While IRL avoids the compounding error problem of purely supervised methods, it introduces challenges regarding reward inference and ambiguity. The assumption that expert demonstrations reflect an optimal policy does not imply that the observed behavior is a perfect representation of the desired behavior (Arora and Doshi, 2020). Multiple reward functions may explain the same demonstrated trajectories, making reward inference inherently underdetermined. In addition, the

computational complexity of IRL is high. Recovering the reward function typically requires repeatedly solving or approximating the underlying decision process, causing computation to grow disproportionately with the size and dimensionality of the environment. This makes IRL difficult to scale to complex systems, such as biological multi-agent environments. To address the ambiguity and computational overhead of explicitly recovering the reward function, adversarial imitation learning bypasses reward inference entirely.

2.2.3 Adversarial Imitation Learning

The adversarial imitation learning methods are a class of model-free imitation learning approaches based on generative adversarial networks (GANs). While GANs are primarily known from image generation, where the goal is to create synthetic data that resembles a target dataset, Ho and Ermon (2016) adapted this framework for the purpose of behavior imitation. Their method, Generative Adversarial Imitation Learning (GAIL), follows the same fundamental structure as the original GAN formulation by Goodfellow et al. (2014), consisting of two components: a generator and a discriminator. The adversarial aspect arises from the interplay between both. In GAIL, the policy π_θ acts as the generator and produces artificial trajectories by interacting with the environment. Its objective is to fool the discriminator D_β, which attempts to distinguish policy-generated state–action pairs from expert demonstrations. During training, the policy and discriminator are optimized alternately (Ho and Ermon, 2016). After each policy update, new trajectories are generated and used to train the discriminator, which outputs a probability indicating how likely a state–action pair originates from the expert $D_\beta(s, a) : \mathcal{S} \times \mathcal{A} \to [0, 1]$. In the idealized optimum, training converges when $D_\beta(s, a) = 0.5$, meaning the discriminator can no longer distinguish between expert and generated trajectories. The discriminator is optimized via the binary cross-entropy loss (Equation 1).

$$\min_\beta\{\mathcal{L}_\beta\} = \min_\beta \left\{\mathbb{E}_\pi\big[-\log(D_\beta(s,a))\big] + \mathbb{E}_{\pi_E}\big[-\log(1 - D_\beta(s,a))\big]\right\} \tag{1}$$

Following the objective of the policy to maximize the expected cumulative reward, the policy is optimized by treating the discriminator output as a learned surrogate reward (Ho and Ermon, 2016). Specifically, the policy receives the reward signal $r(s, a) = -\log\left(D_\beta(s, a)\right)$. This increases when the policy generates state–action pairs that the discriminator classifies as expert-like. The policy parameters θ are updated to maximize the expected return under this reward. GAIL offers several advantages over BC and IRL discussed above. Compounding errors are avoided because learning does not rely on supervised action labels. The method is also more sample-efficient and learns substantially faster than IRL, a difference that becomes

particularly relevant in multi-agent settings (Ho and Ermon, 2016). Moreover, GAIL does not require explicitly defining or recovering a reward function. GAIL directly matches the occupancy measures of the expert and the learned policy, reframing it as a distribution alignment problem, where successful learning is achieved once the agent reproduces the expert's state–action distribution (Section 4.3.2). Because this matching does not depend on the number of agents involved, the approach naturally scales to multi-agent systems, as demonstrated in recent work (Song et al., 2018; Yapei Wu et al., 2024; Yifan Wu et al., 2025).

2.3 Predator-Prey Systems

Multi-agent systems consist of multiple interacting agents whose behavior continuously influences, and is influenced by the actions of others (Yong and Miikkulainen, 2001). These interactions can range from fully cooperative to competitive or mixed settings. Predator–prey systems represent a specific form of multi-agent systems in which heterogeneous groups pursue contrasting objectives. In such pursuit–evasion problems, one or more predators attempt to capture one or more prey, while the prey attempt to avoid being caught. This interaction leads to opposing strategic objectives between the two groups. Within the prey group, individuals commonly exhibit cooperative behavior, such as coordinated motion or synchronized directional changes, which can confuse the predator and increase collective survival chances (Li et al., 2023). Predators, in contrast, may employ dispersion tactics, initially moving toward the center of the prey swarm to break cohesion, before shifting their focus toward individuals that become isolated during the dispersal. This illustrates why imitating such natural behavior is challenging. The resulting collective motion does not arise from purely harmonic interactions, but from an ongoing, survival-driven interplay between cooperative prey and actions of the predator.

Learning collective behavior in predator–prey systems differs fundamentally from imitation in the single-agent case. Multiple agents act simultaneously, and each agent's local observation depends on the positions and actions of others, making the environment non-stationary (Li et al., 2023; Yapei Wu et al., 2024). As a result, the objective of adversarial imitation is not to reproduce isolated state–action pairs, but to match the joint distribution of states and actions across all interacting agents. This role asymmetry requires the concurrent training of two policies: one homogeneous policy for the prey swarm and one heterogeneous policy for the predator. The influence of neighbors is not uniform. Interactions must be weighted, as closer agents and the predator exert stronger behavioral influence than distant individuals (Heras et al., 2019). Following the GAIL framework, each group is therefore assigned its own discriminator, allowing the model to capture role-specific behavioral patterns and to evaluate how well each policy reproduces the demonstrated

group behavior. However, training both policies and discriminators simultaneously introduces additional complexity. If a discriminator learns too quickly, the reward signal degenerates and the corresponding policy cannot adapt effectively. This issue is amplified in this predator-prey setting, where improvements in one policy directly alter the learning environment of the other. Furthermore, GAIL is susceptible to mode collapse (Ho and Ermon, 2016; Yifan Wu et al., 2025). In mode collapse, the generator focuses on producing only a limited subset of behavioral patterns that are favored by the discriminator, reducing behavioral diversity. As a result, the policy avoids exploring the full distribution of behaviors present in the expert demonstrations. Chapter 4 provides the algorithmic details of the co-evolutionary training procedure and explains how the above challenges are addressed.

2.4 Related Work

A variety of modeling approaches have been developed to understand the underlying concepts and rules governing collective behavior. While the study of collective behavior spans different fields, ranging from character animation in computer graphics (Gustafson et al., 2016), pedestrian movement in human crowds (Lee et al., 2018), to ensembles of interacting large language models (Wei et al., 2022), the scope here is restricted to the imitation of collective behavior in biological systems of swarming animals, particularly fish. Over time, advances in computational power and learning methods have shifted the focus from hand-crafted interaction rules towards data-driven models that infer group dynamics directly from real observations. One of the earliest rule-based models is the Boids model proposed by Reynolds (1987), which simulates flocking behavior based on three local interaction rules: separation to avoid collisions, alignment to match the heading of neighbors, and cohesion to maintain group structure. These simple behavioral rules were sufficient to generate emergent collective motion without requiring global coordination, leadership, or explicit communication. Building on this foundation, Couzin et al. (2002) introduced a model extension based on three spatial interaction zones: repulsion, orientation, and attraction. The rules are applied hierarchically. At the highest priority, in the short-range zone of repulsion, individuals maintain a minimum distance from others at all times. Outside the repulsion zone, individuals align their heading with neighbors in the zone of orientation. Finally, in the long-range zone of attraction, individuals move toward neighbors to maintain group cohesion. This hierarchical rule structure demonstrates that coordinated group patterns could also emerge as a direct consequence of local interaction ranges. Both the Boids model and the Couzin model are influential in the study of collective behavior, showing that coordinated group dynamics can emerge from simple local interaction rules. However, despite their explanatory power, these models remain idealized abstractions that do not capture the full complexity of real biological systems (Wang et al., 2024).

Alongside rule-based models, data-driven approaches have been developed that infer collective behavior directly from observed trajectories rather than defining interaction rules beforehand. Mining this data can be done in different ways. Yapei Wu et al. (2024) and Yu et al. (2021) generated synthetic trajectories using the Couzin and Vicsek model. From this synthetic demonstrations, Yapei Wu et al. (2024) used distances and relative velocities between individuals at each timestep as input for their neural network. In contrast, Calovi et al. (2015); Heras et al. (2019); Schafer et al. (2022); Yifan Wu et al. (2025); Yu et al. (2021) used video recordings of real fish as their primary data source. Calovi et al. (2015); Heras et al. (2019); Schafer et al. (2022); Yu et al. (2021) extracted individual trajectories using (multi-object) tracking methods to obtain positions, velocities, accelerations, relative distances, and headings. Meanwhile, Yifan Wu et al. (2025) processed raw videos using a masked video autoencoder to learn latent motion representations directly, without relying on explicit trajectory tracking. In a meta-analysis, Wang et al. (2024) reviewed data-driven policy learning methods derived from biological behavior and categorized them into three main types: IL, IRL, and Causal Policy Learning (CPL). In CPL, causal models are combined with policy learning methods to identify causal relationships between states and actions. All approaches share the common goal of learning behavioral strategies directly from data. Exemplary, Torabi et al. (2018) introduced their BC framework, which enables imitation from state-only demonstrations when the demonstrator's actions are not available. The method learns an inverse dynamics model through self-supervised exploration to reconstruct action labels, allowing standard BC to be applied afterward. Heras et al. (2019) trained two modular networks for pairwise interaction and aggregation of individuals in a swarm, using positions, relative velocities, accelerations, and heading angles extracted from tracked trajectories. Based on these features, the model predicts turning decisions of the focal individual in a supervised learning setup. Schafer et al. (2022) applied Bayesian IRL to infer the local behavioral rules governing a captive guppy population. By formulating collective behavior as a linearly-solvable Markov decision process, they recovered latent decision cost structures directly from observed trajectories. Similarly, Yu et al. (2021) introduced a parameter-sharing adversarial inverse reinforcement learning framework specifically adapted to the characteristics of biological swarm systems. Based on the assumption that all individuals are identical and interact only with local neighbors, the approach exploits parameter sharing to learn a single shared policy together with the underlying reward function from collective demonstrations.

Adversarial imitation learning approaches focus on reproducing the distribution of observed trajectories directly, without explicitly recovering a reward function. Yapei Wu et al. (2024) and Yifan Wu et al. (2025) adapted this approach to directly imitate the collective behavior of swarming fish from demonstrations. Yapei Wu et al. (2024) employed the deep attention network structure proposed by Heras et al. (2019) as the generator policy, modeling interactions between individuals through a pairwise interaction network and an attention-based aggregation mechanism. The

pairwise network models how a focal agent interacts with each neighbor, while the attention module adaptively weights these neighbor-specific influences based on spatial context. Using parameter sharing, the global policy is able to learn predefined synthetically generated fish behaviors such as schooling, milling, and swarming. Furthermore, the learned policy was successfully transferred to a real-world swarm robotics platform, demonstrating that their biologically inspired GAIL-framework can be applied beyond simulation. With their masked video autoencoder, Yifan Wu et al. (2025) directly imitated collective behavior from raw videos, without relying on explicit trajectory extraction. This avoids common issues in tracking-based pipelines, such as occlusions and overlapping individuals. In their GAIL framework, the latent space representation produced by the autoencoder serves as the target distribution for imitation. Acting as a compact descriptor of the observed swarm motion. To ensure stable and coherent group dynamics, bio-inspired reward-shaping was incorporated during training. Using this approach, Yifan Wu et al. (2025) were able to reproduce characteristic swarm patterns such as circling, alignment, aggregation, and cohesion. The framework also captures predation interactions and collective responses under survival pressure directly from video recordings using a multi-instance single policy.

Most of the methods reviewed so far focus on homogeneous collectives in which all agents follow a shared behavioral strategy. Predator–prey systems, however, introduce asymmetric roles in which pursuit and avoidance behaviors shape the resulting group dynamics. Here, collective patterns emerge not only from interactions within a group, but also from the opposing behavioral objectives between predators and prey. Li et al. (2023) investigated predator–prey interactions from an evolutionary perspective and showed that swarming behavior can emerge solely from survival pressure. Using a cooperative–competitive multi-agent RL framework, predators and prey co-adapted their policies based on opposing rewards, without any hand-crafted alignment or cohesion terms. The model successfully reproduced flocking and milling behaviors of prey, as well as dispersion tactics, confusion effects, and marginal predation strategies of predators. Looking specifically at the application of GAIL in predator–prey settings, Yifan Wu et al. (2025) demonstrated a unique example in which predation patterns can be imitated directly from videos, treating the imitation of predation more as a successful addition rather than the main focus of the study. Mentionable is the work of Song et al. (2018) who proposed a multi-agent GAIL framework for Markov decision games. Their approach introduces different discriminator–policy configurations to accommodate cooperative, mixed, and competitive interaction structures.

Summing up, imitating collective behavior is challenging and becomes particularly complex in predator–prey systems, where heterogeneous groups interact competitively and the observed dynamics emerge from survival-driven pursuit and avoidance. While data-driven approaches move beyond hand-crafted rules by learning collective motion directly from demonstrations, they still depend strongly on the

quality of the underlying observations and the coverage of behaviors. In adversarial imitation learning, training is further complicated by the coupled optimization of policies and discriminators and the co-evolutionary setup, where improvements in one group directly change the learning environment of the other. The next section describes the data collection and processing pipeline used to extract and structure predator–prey trajectories from video recordings.

CHAPTER 3

Data Collection and Processing

In this chapter, the experimental design to record the predator–prey aquarium is introduced. Following the data processing pipeline step-by-step, it is shown how the raw videos are processed into input tensors that serve as meaningful representations of expert behavior. First, object detection and tracking are used to extract trajectories. Then, feature engineering and transformations are applied to construct a tensor representation suitable for model training.

3.1 Data Collection

The recordings were provided by the research group Li at the Department of Collective Behaviour of the Max Planck Institute of Animal Behavior (Li, 2025). The experimental design and data collection were conducted independently and completed prior to the start of this thesis. Parts of the following section are therefore taken from the research group's description.

The prey species *Leucaspius delineatus* (sunbleak) is a freshwater fish that primarily lives in still or slowly moving waters which forms swarms near the surface (European Commission, 2026b; Leu et al., 2009). Sunbleaks are omnivorous and show flexible feeding behavior, ranging from algae to larvae and mosquitoes. Historically, their origin lies in Central and Eastern Europe. The predator species *Esox lucius* (northern pike) is mainly piscivorous, meaning it primarily feeds on fish (European Commission, 2026a). Pike can be found in Northern and Central Europe, as well as in parts of Asia and North America. It typically inhabits freshwater environments close to the shore or within reed beds. Since pike often hunts weak or sick fish, including individuals of its own species, it plays a relevant role in keeping the ecosystem healthy.

Wild sunbleaks and pike were captured and transported to a climate-controlled laboratory, where fish were housed in large holding tanks containing artificial plants and covered areas. Individuals were allowed to acclimatize to laboratory conditions for at least one week prior to experimentation. Across all experiments, the study

followed a common pipeline consisting of individual accommodation and acclimatization, collective behavioural trials, and video-based data extraction and analyses. The full procedure progressed from wild capture and laboratory acclimatization to collective behavioural trials manipulating group size and predation risk, followed by automated extraction of behavioural variables from recorded videos.

Trials were conducted in a 1.5×1.5 m featureless white tank, surrounded by black curtains to minimize visual disturbance, and placed in an isolated room to reduce noise from passersby (Figure 1). To further dampen vibrations, two layers of carpet were laid along the bottom of the tank. The arena was filled to a water depth of 6 cm, and water temperature, conductivity, and pH were maintained comparable to the species' home tanks (approximately 62° F, 700 S, and pH 7.1). Sunbleaks and pike were housed in separate rooms to prevent visual contact outside trials. To encourage predatory motivation, pike feeding was suspended for 24 h before the trial period.

For collective behavioural trials, groups of sunbleaks of varying size ($1, 2, 4, 8, 16, 32$, or 64 individuals) were netted from their home tanks and transferred in a covered bucket to the centre of the experimental arena. The experimenter then transferred a pike into a partitioned corner of the tank following the same handling precautions. This transfer procedure took approximately $3-5$ min. The lower part of the partition was opaque to restrict visual contact between predator and prey during the initial phase. Once all sunbleaks were introduced, curtains were closed around the tank, recording commenced, and the experimenter moved away from the arena. After 5 min, the partition was raised via a pulley to allow the pike access to the full arena. Ten minutes later, filming was stopped and a low-stress removal protocol was applied by turning off the main lights and switching on red light for 1 min before opening the curtains and slowly netting sunbleaks back to their home tanks, followed by the pike. Each group-size condition required approximately 1 h per trial, and experiments were limited to a maximum of 10 trials per 24 h.

Behavior was recorded using an overhead four-camera array positioned approximately 2 m above the tank, capturing video at $\sim 24 - 90$ $frames$ s^{-1} with 2048×2048-pixel resolution. Cameras were connected to an external recorder that saved data directly to a solid-state drive. Video data were subsequently processed using automated tracking based on computer-vision software previously developed within the research group Li to extract trajectories and quantify collective behavioural metrics. After completion of the experiments, fish were maintained temporarily in large social housing tanks and subsequently transferred to large mesocosms at the limnological institute for use in additional behavioural experiments documented elsewhere.

After cutting relevant scenes with predator presence, a total of 35 recordings were saved at a resolution of 2160×2160 pixels and 30 fps (the resolution was slightly altered during video cutting). The combined duration of 16:42:49 h resulted in 151.695

usable frames (Figure 1). Compared to the full recording duration, predator attacks are strongly underrepresented. In total, 32 predator attacks were recorded with a combined duration of 18.9 s. Based on time, only 0.12% of the total recordings show attack scenes in which the predator is actively hunting.

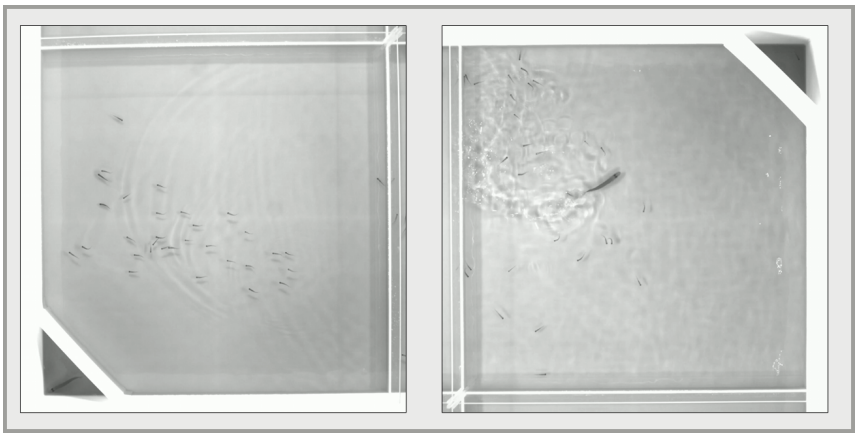


Figure 1: Example frame of the environment with and without the predator.

3.2 Data Processing

The goal of data processing is to transform raw video recordings into meaningful representations suitable for model training. The following section describes the data pipeline, from object detection and tracking to cleaning, feature engineering, and tensor construction.

3.2.1 Multi-Object Detection and Tracking

Object detection and tracking are central tasks in computer vision. A variety of algorithms exist with different methodological emphases and application domains. For real-time scenarios, the YOLO (You Only Look Once) family of models provides high accuracy and speed while keeping computational costs low (Khanam and Hussain, 2024; Ultralytics, 2026). The most recent version, YOLOv11, represents "a significant advancement in real-time object detection technology" (Khanam and Hussain, 2024, p. 4). YOLOv11 is based on a convolutional neural network (CNN) and follows a three-part architecture consisting of backbone, neck, and head (Khanam and Hussain, 2024). The backbone applies convolutions to extract multi-scale features from raw input images and produces feature maps. The neck aggregates these features across different scales and transmits them to the head, which generates the final

bounding boxes and class predictions. Compared to previous versions, YOLOv11 integrates attention mechanisms and additional architectural modules that contribute to faster processing without compromising detection accuracy.

For this study, YOLOv11 was pretrained and subsequently fine-tuned for the task of fish detection using *Label Studio* (Juras, 2025; Tkachenko et al., 2025). For training, 100 frames from the recordings were randomly sampled and manually annotated with bounding boxes. Three categories were distinguished: Prey, Predator, and Predator Head (Figure 2). For further analysis, only Prey and Predator Head were used, represented by the center of the corresponding bounding box. Due to the predator's size and its close proximity to prey during attack scenes, the Predator Head was used as the predator representation. In this setting, traditional classification metrics such as accuracy, precision, or recall are not suitable, since object detection involves a variable number of instances per frame rather than a fixed one-to-one correspondence. Detection errors caused by occlusions in dense fish groups, light reflections and shadows in the experimental setup led to deviations of the predicted counts, resulting in both under- and overestimation compared to the true values. To still provide a meaningful measure of detection quality, the mean absolute error (MAE) was used for evaluation. For predator detections, the model achieved high performance with a MAE of ± 0.03 relative to the ground truth (Appendix 3, Figure 33). Prey detections exhibited greater variance, with an MAE of ± 3.11 (Appendix A, Figure 34). It should be noted that this value is additionally affected by the predator's predation success, as the total number of prey decreases over the course of tracking. Overall, the results indicate stable detection performance when the surface was calm, with a systematic tendency toward undercounting. To mitigate disturbances, predator attacks were additionally hand-labeled to ensure full visibility of all individuals.

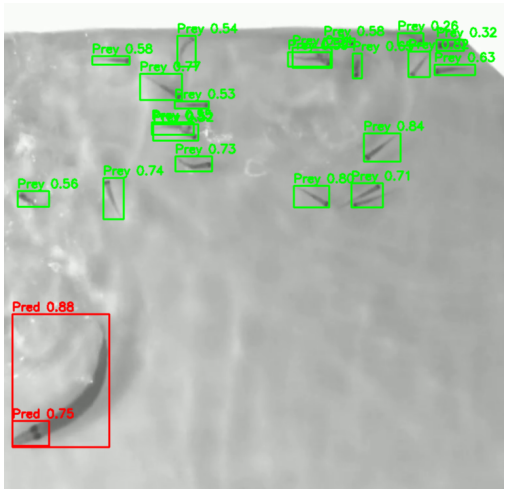


Figure 2: Sample frame with YOLO detections.

Detected fish were tracked across frames using DeepSORT (Simple Online and Realtime Tracking). The tracker is a "simple framework that performs Kalman filtering in image space and frame-by-frame data association using the Hungarian method with an association metric that measures bounding box overlap" (Wojke et al., 2017, p. 1). With its good tracking performances in terms of precision and accuracy, the algorithm is able to track objects through longer periods of occlusion while effectively reducing the number of identified switches. The Hungarian algorithm by (Kuhn, 1955) was developed to solve the assignment problem by finding minimum-

cost matchings between sets using a cost matrix, ensuring that each task is exactly assigned to one agent. The Kalman filter enables state estimation in cases where detections are missing, thereby smoothing trajectories and reducing fragmentation (Kalman, 1960). The parameter `max_age` determines how many consecutive frames a target may remain undetected before its trajectory is terminated. After analyzing different values, a threshold of 30 (1 sec) was selected to balance robustness against short-term occlusions while preserving trajectory continuity.

After detection and tracking, trajectories were cleaned and filtered. If multiple predator detections occurred in a single frame, only the bounding box with the highest confidence score was retained. Frames without predator detections were excluded. Valid trajectories were defined as sequences containing exactly one predator and 32 prey with consistent indices. The ratio of valid trajectories relative to all accessible frames is 7.79%. Yifan Wu et al. (2025) trained their model using sequence windows of length ten. Based on their setup, windows of the same length were extracted from all valid trajectories, resulting in 793 windows (Appendix A, Table 1). Most of these windows correspond to hand-labeled predator-attack scenarios. In comparison, Yapei Wu et al. (2024) trained on generated trajectories of 1,000 steps. The available dataset is therefore assumed to be sufficient for robust model training.

Quality assurance of the processed trajectories was carried out in several steps to ensure that the data met the requirements for subsequent model training. First, velocities were examined to detect unrealistic jumps in movement. For each tracked trajectory, the distribution of maximum speeds was plotted and extreme values analyzed (Appendix B, Figure 36 and Figure 37). Outliers at the extreme ends of the distribution, which could arise from faulty detections or sudden ID changes, were systematically clipped. This ensured that implausible accelerations did not propagate into the feature tensors. Second, the distribution of heading angles was analyzed (Appendix B, Table 2 and Figure 35). In principle, however, extreme steering manoeuvres are also conceivable, especially when the predator is approaching. Leading to no feature-engineering of the heading angle. Third, a visual validation was done. The processed tracking results were compared to the original video frames and manually inspected. Visual inspection also made it possible to identify typical tracking issues, such as ID switches when individuals overlapped or passed each other in dense groups. These switches were observed occasionally but remained localized and did not affect the overall integrity of the trajectories. Taken together, these steps ensured that the processed trajectories were both quantitatively and qualitatively reliable. With the exception of minor deviations, such as short-term ID changes during occlusions, no systematic weaknesses were found. The resulting trajectories can therefore be regarded as a consistent and high-quality foundation for model training.

3.2.2 Feature Engineering and Tensor Construction

Following Yapei Wu et al. (2024), the state representation was constructed from distances and relative velocities between individuals (Figure 3), consistent with evidence reported in related work by Heras et al. (2019). Distances were computed as pairwise differences in the x- and y-coordinates between neighbors and the focal individual and subsequently scaled into the interval $d_{x,i},\, d_{y,i} \in [-1, 1]$ (Equation 2).

$$\begin{aligned} x_i &= x_{\text{neighbor}} - x_{\text{focal}} \\ y_i &= y_{\text{neighbor}} - y_{\text{focal}} \end{aligned} \tag{2}$$

Velocities were derived from frame-to-frame displacements provided by the tracker. To avoid distortions from extreme values, the distribution of measured maximum speeds was analyzed. Based on the box plot inspection, a maximum value of 10 pixels per frame was set as an upper bound and used as the environment's speed constant for rollout collection (Appendix B, Figure 37). Values exceeding this threshold were clipped. Subsequently, relative velocities were computed in the focal frame by rotating the neighbor velocity vectors by the negative heading angle of the focal individual. Thus, the velocity components of neighbors are transformed into relative velocities with respect to the focal individual (Equation 3). The resulting values were scaled accordingly $v_{x,i},\, v_{y,i} \in [-1, 1]$.

$$\begin{bmatrix} v_{x,i} \\ v_{y,i} \end{bmatrix} = \begin{bmatrix} \cos(-\theta_{\text{focal}}) & -\sin(-\theta_{\text{focal}}) \\ \sin(-\theta_{\text{focal}}) & \cos(-\theta_{\text{focal}}) \end{bmatrix} \begin{bmatrix} v_{x,\text{neighbor}} \\ v_{y,\text{neighbor}} \end{bmatrix} \tag{3}$$

To account for the co-evolution of two interacting policies, an additional binary feature was introduced to indicate the role of each fish (Equation 4). This predator flag was incorporated exclusively into the prey tensor, enabling prey agents to explicitly distinguish between conspecifics and the predator. Like Yapei Wu et al. (2024), the focal speed is not included as an explicit input dimension. Instead, velocity information enters only through relative neighbor velocity components. This design choice reduces the input dimensionality and therefore the overall model complexity.

$$Predator : (d_x, d_y, v_x, v_y), \quad Prey : (flag, d_x, d_y, v_x, v_y) \tag{4}$$

The action is represented as $\Delta\theta$, which denotes the per-step heading change of the focal individual. First, the absolute heading angle θ_t is derived from the tracker motion vector. The action is then defined as the wrapped difference to avoid discontinuities at $\pm\,\pi$, and subsequently scaled to the interval $\Delta\theta \in [0, 1]$. The action is

only used for the purpose of pretraining via behavior cloning. During GAIL training, the model operates on transition features computed from states only, excluding the action channel (Section 4.2).

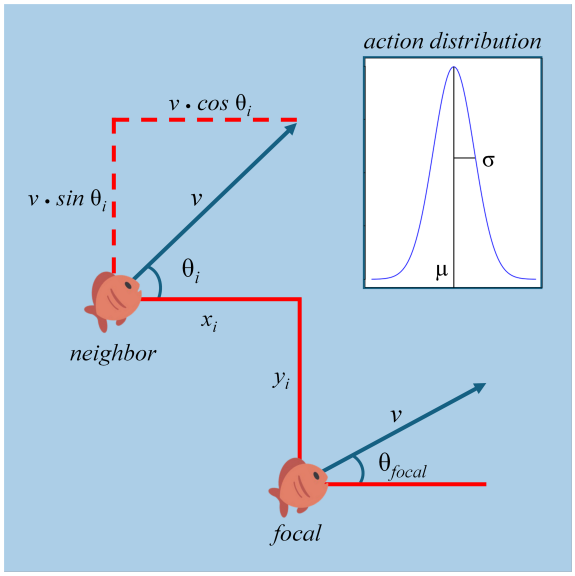


Figure 3: State definition for focal individual (Yapei Wu et al., 2024).

For convenience in data handling, the action value of $\Delta\theta$ was incorporated into the tensor representation. The overall tensor structure is defined as follows. The feature dimension (F) arises from the pairwise interactions between a focal fish and each of its neighbors. Accordingly, this yields 32 neighboring fish (N) for each focal fish (A). When multiple consecutive frames are processed, these tensors are stacked along the temporal dimension (T). Processing multiple sequences in parallel adds the batch dimension (B), resulting in a tensor of the form Tensor $\in \mathbb{R}^{B\times T\times A\times N\times F}$. The final tensors obtained for the two roles are given by the following expression 5.

$$\text{Predator Tensor} \in \mathbb{R}^{793\times 10\times 1\times 32\times 5}, \quad \text{Prey Tensor} \in \mathbb{R}^{793\times 10\times 32\times 32\times 6} \tag{5}$$

3.2.3 Summary of Data Pipeline

The trajectory processing pipeline transforms raw video recordings into structured expert tensors for model training (Figure 4). Fish were detected with a fine-tuned YOLOv11 model and tracked across frames using DeepSORT. Invalid or incomplete sequences were filtered, implausible movements clipped and critical predator–prey interactions manually corrected. From the validated trajectories, input features were computed as well as the predator flagged. Finally, the heading angles were derived from the motion vectors. The resulting representation yields consistent predator

and prey tensors that serve as the foundation for model training.

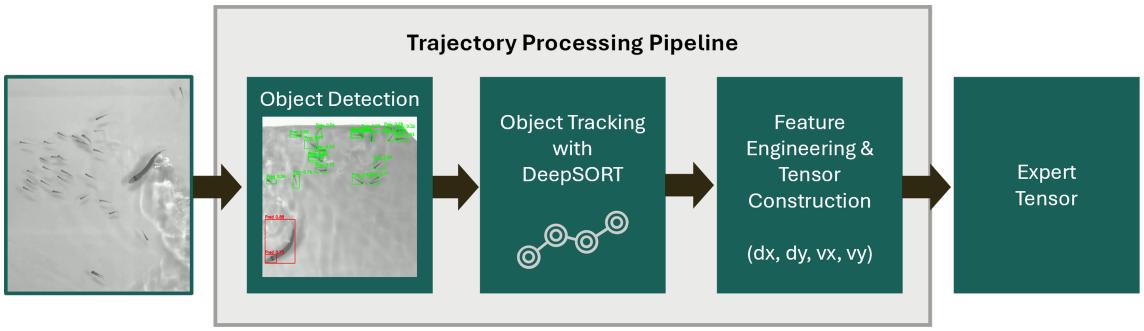


Figure 4: Overview of the trajectory preprocessing pipeline.

CHAPTER 4

Generative Adversarial Imitation Learning

Building upon the trajectories processed from the predator–prey recordings, the next step is to construct a framework capable of imitating this expert behavior (Figure 5). For this purpose, GAIL is applied to train individual policies for predator and prey. After introducing the main architectural components, the need for a transition encoder is explained, followed by the setup of the environment. Finally, the performance measures are introduced.

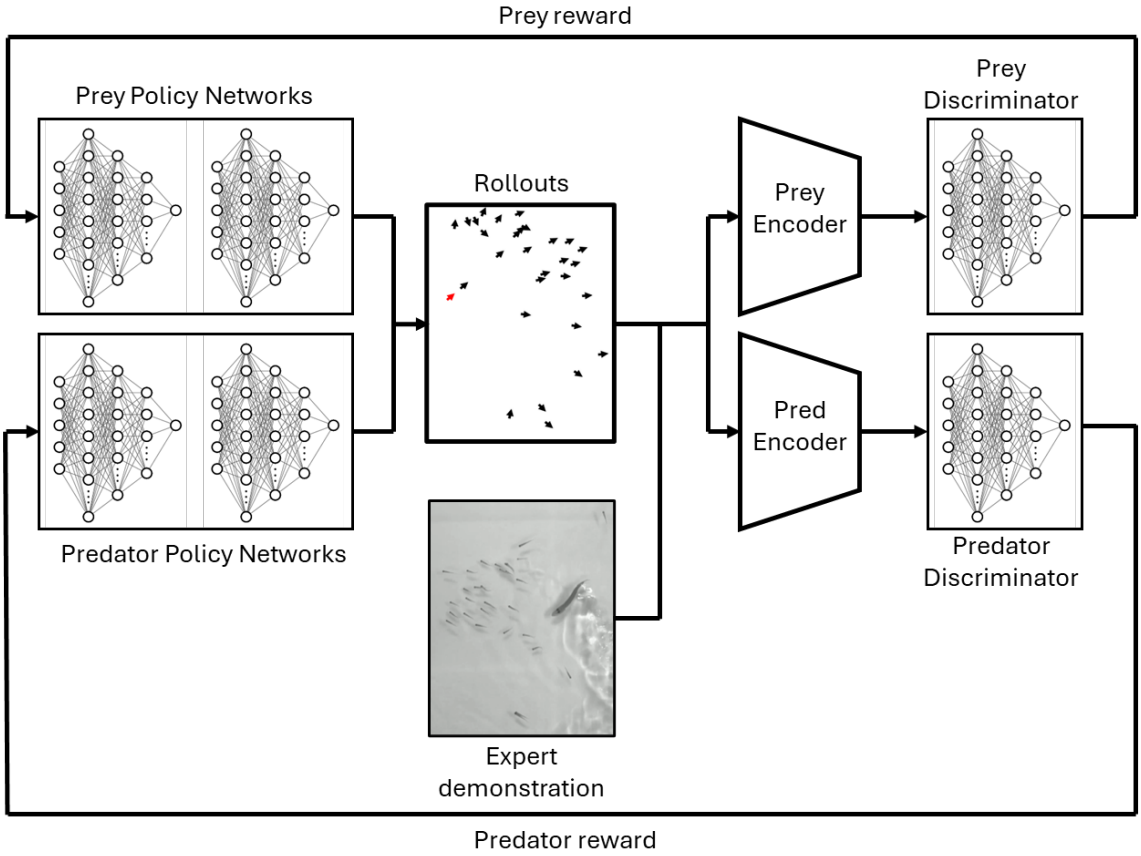


Figure 5: Overview of the GAIL architecture.

4.1 Generator and Policy

To model collective behavior with GAIL, a dedicated policy architecture is defined for both roles. The policy represents the decision-making mechanism of each agent, determining the next action based on the input features. To capture the behavioral asymmetry between predator and prey, the two roles are modeled with separate policies. The generator's primary task in the GAIL framework is to produce trajectories that resemble the expert demonstrations closely enough, to fool the discriminator. Following the research of Heras et al. (2019) and Yapei Wu et al. (2024), the policy consists of two key components: the pairwise interaction network (PIN) and the attention network (AN) (Figure 6). This decomposition increases interpretability by separating the interaction dynamics from the aggregation process. The PIN captures how a focal individual responds to a single neighbor, while the AN determines the strength of each neighbor's influence. The policies are denoted as $\pi_{P_{\phi,\varphi}}$ for the predator and $\pi_{Y_{\phi,\varphi}}$ for the prey, where ϕ refers to the parameters of the PIN and φ to the parameters of the AN. The corresponding expert demonstrations are denoted by $\tau_{E,P}$ for the predator and $\tau_{E,Y}$ for the prey.

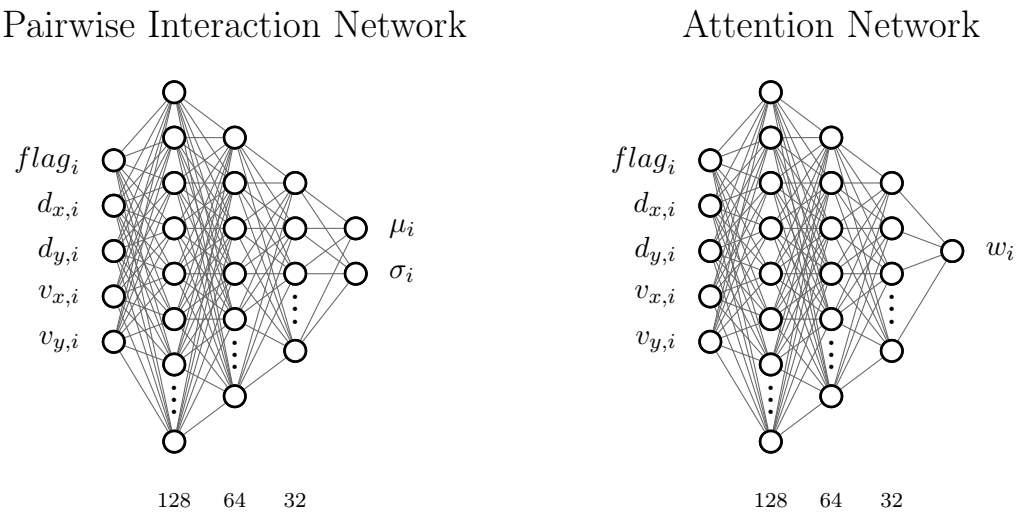


Figure 6: PIN and AN architectures with prey-specific input features.

4.1.1 Pairwise Interaction Network

The primary task of the PIN is to determine the action based on the focal interactions with its neighbors. More precisely, the network learns how the turning behavior

of an individual is influenced by a neighbor's distance and its velocity from focal perspective. The input layer processes a 4- or 5-dimensional feature vector. For the prey, an additional flag feature is included to explicitly indicate the predator. The features are then processed by three fully connected hidden layers with gradually decreasing widths, using ReLU activation functions. The output layer consists of two neurons, corresponding to the mean μ_i and standard deviation σ_i of a Gaussian distribution from which the action is sampled (Equation 6):

$$a_i \sim \mathcal{N}(\mu_i, \sigma_i) \tag{6}$$

The standard deviation is stabilized using a softplus transformation and a small numerical constant to ensure it remains positive and numerically robust. While the network architecture of Yapei Wu et al. (2024) served as an initial reference, a deeper structure was found to better capture the complexity of non-simulated behavior. Nevertheless, care was taken to avoid making the network too deep in order to keep training feasible. This stochastic formulation introduces non-determinism into the policy, increasing action variability instead of producing identical reactions in identical states. In turn, compounding errors that typically arise in pure BC frameworks are mitigated, and increased sample diversity reduces the risk of mode collapse.

4.1.2 Attention Network

The AN is responsible for assigning weights to the interactions modeled by the PIN. It learns the magnitude of influence that each neighbor i exerts on the focal individual, encoded in the attention weight w_i. This scaling of influence is akin to an attention mechanism, allowing the focal individual to selectively emphasize the most relevant neighbors rather than treating all neighbors equally (Yapei Wu et al., 2024). While research indicates that only the closest neighbors are most relevant for a focal's decision-making (Heras et al., 2019), the AN deliberately considers all neighbors. This gives the model the opportunity to learn interaction patterns autonomously, instead of restricting them beforehand. Furthermore, this formulation enables analyzing swarm composition and role-specific behavioral contributions of individual agents (Section 6). The AN follows the same architectural design as the PIN, following the evidence reported by Yapei Wu et al. (2024). Unlike the PIN, the output layer of the AN consists of a single neuron producing a raw attention score, which is subsequently normalized using a softmax operation. The resulting attention weights determine the contribution of each neighbor's interaction signal to the aggregated final action. The final policy output is obtained by aggregating the neighbor-wise actions (Yapei Wu et al., 2024). For each neighbor $i \in \mathcal{I}$, an action a_i

and a corresponding weight ω_i are computed using the PIN and AN. The resulting action is computed as the normalized weighted sum (Equation 7).

$$a = \sum_{i \in \mathcal{I}} a_i \frac{\omega_i}{\sum_{j \in \mathcal{I}} \omega_j} \tag{7}$$

Aggregating the prey actions represents a special case. In the tensor representation, the first neighbor entry always corresponds to the predator and is additionally marked by the predator flag. In the final implementation, the predator is not handled explicitly during aggregation beyond this flag encoding, allowing the policy to learn the separation and role-specific neighbor handling autonomously. This design aspect is discussed in Section 5.4. The final action is transformed to $[0, 1]$ using a sigmoid activation function.

4.1.3 Evolutionary Strategy Optimization

Instead of optimizing the predator and prey policies via backpropagation, this work applies a gradient-free optimization scheme based on evolutionary strategies (ES) (Salimans et al., 2017). This choice follows the approach of Yapei Wu et al. (2024), who trained their policies using ES in their GAIL setting. ES are inspired by natural evolution. "At every iteration ("generation"), a population of parameter vectors ("genotypes") is perturbed ("mutated") and their objective function value ("fitness") is evaluated" (Salimans et al., 2017, p.2). Mathematically, Salimans et al. (2017, p.7) represent the population as a probability distribution over policy parameters θ, written as $p_\phi(\theta)$, where ϕ defines the distribution parameters (Yapei Wu et al., 2024). The objective is to optimize ϕ such that the expected reward across the population is maximized. The fitness $f(\theta)$ is defined as the discriminator-based return obtained from environment rollouts (Equation 8).

$$\max \mathbb{E}_{\theta \sim p_\phi}[f(\theta)] \tag{8}$$

The algorithm proceeds in two phases (Salimans et al., 2017). First, stochastic perturbations are applied to the current parameter vector $\vec{\theta}_t$ by sampling Gaussian noise $\epsilon_i \sim \mathcal{N}(0, I)$ and scaling it by the exploration parameter σ. Here, I denotes the covariance matrix, meaning that noise is sampled independently for each parameter dimension. The perturbed policies $\pi_{\theta_t + \sigma\epsilon_i}$ and $\pi_{\theta_t - \sigma\epsilon_i}$ are evaluated by collecting rollouts in the environment. Second, the update is driven by the discriminator-based rewards (R^+, R^-), where perturbations leading to higher rewards contribute more strongly to the next parameter update. Following Yapei Wu et al. (2024) and Salimans et al. (2017), a dual-sampling scheme is used, in which each perturbation is mirrored, yielding symmetric perturbations $\theta_i + \epsilon_i$ and $\theta_i - \epsilon_i$. This reduces

estimator variance and ensures that the exploration remains unbiased around the mean parameter vector. The update rule for the policy parameters is given by equation 9.

$$\vec{\theta}_{t+1} \leftarrow \vec{\theta}_t + \lambda \frac{1}{2\sigma^2 n} \sum_{i=1}^{2n} \vec{\epsilon}_i \left(R_i^+ - R_i^- \right) \tag{9}$$

This update corresponds to a stochastic gradient-ascent step that approximates the true gradient of the expected reward in parameter space (Salimans et al., 2017). To further reduce variance and stabilize learning, reward differences are rank-normalized before gradient estimation (Salimans et al., 2017). This reduces the influence of outliers and lowers the risk of convergence to local optima early in training. Finally, both the learning rate λ and the exploration factor σ are decayed over generations using the decay factor γ, gradually reducing the perturbation magnitude as the optimization converges toward a high-performing region in parameter space (Figure 7).

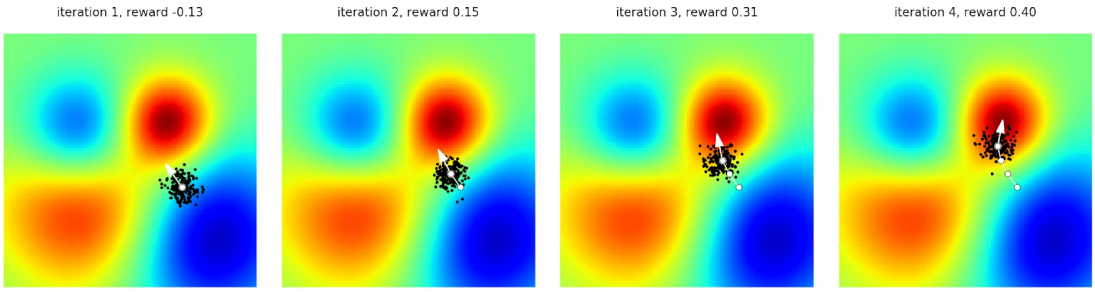


Figure 7: Overview of the ES optimization process (OpenAI, 2017).

ES provides a scalable and robust alternative to policy-gradient methods, as it directly optimizes the policy in parameter space without requiring value-function estimation or backpropagation through the policy (Salimans et al., 2017). Its high degree of parallelism enables efficient execution of multiple perturbed policies simultaneously, resulting in stable learning signals even when actions are sampled stochastically.

4.2 Transition Encoder

After generating rollouts using the modular policy, these have to be compared to the expert trajectories. For this purpose, a discriminator is trained on randomly

sampled input sequences and has to learn to differentiate between expert and generated behavior. However, processing sequences is challenging. Considering time dependencies, system dynamics, memory requirements, and the size of the input dimension, directly learning on raw sequences becomes computationally expensive and difficult to scale. For example, the discriminator initially received an agent-wise perspective, processing $1 \times 32 \times 5 = 160$ inputs at once. Changing the architecture to a frame-processing discriminator, the input size becomes $33 \times 32 \times 5 = 5,280$ inputs, while for sequences of length 10 it increases to $52,800$ inputs. Therefore, a self-supervised transition encoder is used to learn a latent representation of the states, reducing the input dimension while preserving system dynamics (further details see Section 5.4). The discriminator is trained on the transition representation rather than on sequences directly. The design choice was further motivated by the GAIL structure of Yifan Wu et al. (2025), which implemented a masked video autoencoder that processes short video clips while learning a latent representation of transitions between frames, preserving state transitions of collective motion. While the general idea was adapted, the structure of the encoder had to be changed completely in order to process the expert trajectory tensor rather than raw video frames. Again, two separate encoders for predator $E_{P,\psi}$ and prey $E_{Y,\psi}$ are trained, updating the encoder parameters ψ, respectively.

4.2.1 Architecture

The design and architecture of the encoder follows directly from the structure of the input tensor. The encoder operates on the state features, introduced in section 3.2.2. Its objective is to map the high-dimensional state representation to a compact latent space while preserving system dynamics. Following Yifan Wu et al. (2025), the latent space is constructed to represent transitions between consecutive states. For each agent, the encoder computes a single embedding that aggregates information across the neighbor and feature dimensions. Over a sequence of length $window$, this yields $window - 1$ transition embeddings per agent. Following this, the encoder consists of three levels:

$$\text{Input: } [batch, frames, agents, neighbors, features]$$

First, a neighbor-level aggregation compresses neighbor-wise features into a single representation. This initial level follows the same idea as the modular policy networks. Based on the states, an attention module computes unnormalized attention logits for each neighbor. In parallel, an multilayer perceptron (MLP) maps the neighbor features to embeddings. The attention weights are obtained via a softmax over the neighbor dimension and used to compute a weighted sum of these embeddings, resulting in a single embedding vector per agent.

Neighbor-Level: $[batch, frames, agents, embd\_dim]$

Second, the agent-level module maps this pooled representation to a latent agent state z_t using a MLP. This yields a compact per-agent latent state representation that serves as the basis for the transition features.

Agent-Level: $[batch, frames, agents, z]$

Third, the transition module constructs transition features between consecutive latent states. Given the per-agent latent states z_t, the encoder computes $\Delta z_t = z_{t+1} - z_t$ for each time step t and forms the transition representation by concatenating z_t and Δz_t. This yields the final tensor, which is used as input to the discriminator.

Transition-Level: $[batch, frames - 1, agents, 2z]$

4.2.2 Self-supervised Training

In the next step, the encoder is trained. A self-supervised learning approach is applied, where meaningful representations of the input structure are learned without requiring labeled data (Gui et al., 2024). The objective is to obtain a robust latent space that preserves the relevant system dynamics and provides a stable input for the discriminator. Initially, following Yifan Wu et al. (2025), a encoder–decoder architecture was trained. The encoder compresses expert tensors into a latent representation, while the decoder attempts to reconstruct the original input (Bengio et al., 2012, p.13). During training, the reconstruction loss, measuring the discrepancy between original input and reconstructed output, remained high and did not converge. This indicates that the autoencoder setup did not retain sufficient information for reliable reconstruction. Potential reasons include an overly restrictive latent space or insufficient model capacity. However, since the decoder is not required for the subsequent GAIL training, reconstruction quality alone is not a sufficient criterion to judge whether the learned embeddings are robust and informative. As argued by Basaran and Dressler (2025), minimizing reconstruction error alone can encourage overfitting to the training data, resulting in less robust and informative representations that generalize poorly to unseen data. For these reasons, reconstruction-based training was replaced by Variance–Invariance–Covariance Regularization (VICReg), which produced informative and stable embeddings and demonstrated valid performance in initial testing.

VICReg was originally designed to explicitly avoid the representation collapse problem in self-supervised encoders (Bardes et al., 2021). Representation collapse occurs when an encoder maps different inputs to nearly identical latent representations, yielding trivial embeddings that do not preserve meaningful structure. The joint-embeddings architecture works as follows (Figure 8): As input the model gets a batch of expert trajectories I. From this input two different augmented views (X, X') are created using transformations (t, t'), resulting in two different perspectives on the same trajectory. Both views are processed by the encoder using the same parameter vectors to build the latent space representations $Y = f_\theta(X)$ and $Y' = f_\theta(X')$. The projector h_ϕ is an additional MLP that maps the representations into an embedding space $Z = h_\phi(Y)$ and $Z' = h_\phi(Y')$ where the loss function is computed. The loss has two roles: On the one hand, to reduce the information by which the two representations differ (Invariance). On the other hand, it expands the dimension in a non-linear manner, enabling the decorrelation of embedding variables and the reduction of other dependencies between the variables of the representation vector (Covariance).

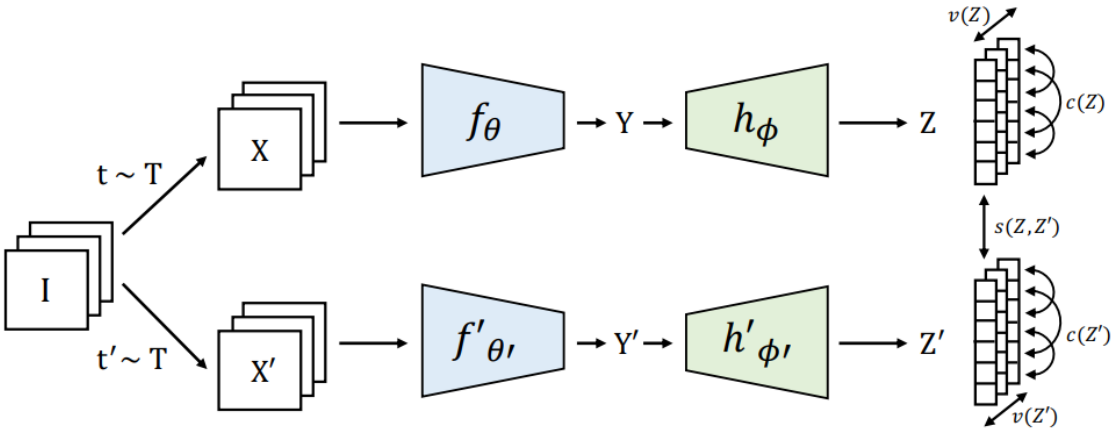


Figure 8: VICReg architectural overview (Bardes et al., 2021, p. 2).

The loss consists of three components, which pursue different objectives:

- **Invariance** is the mean squared distance between the embedding vectors. It aims to align both augmented views of the same trajectory by pulling their embeddings together, ensuring that the representation remains consistent under different perturbations.
- **Variance** forces the embedding vectors to preserve diversity across samples and protects against representation collapse. It is implemented as a hinge loss to maintain the standard deviation of each variable of the embedding above a given threshold. It therefore acts as a penalty if the standard deviation gets too small.

- **Covariance** decorrelates the variables of each embedding and reduces redundancy, preventing representations in which variables are highly correlated and vary together.

The final loss is computed as a weighted sum of the invariance, variance and covariance terms. The encoder is trained beforehand using the expert tensors from predator or prey. The projector is only used for the VICReg objective and is discarded afterwards. During GAIL training, the encoder is frozen and serves only as a dimensionality-reduction module for the discriminator.

4.2.2.1 Trajectory Augmentation

In literature, several ways of augmenting images exist, mainly by applying geometric (e.g. cropping, flipping, masking) or photometric augmentations (e.g. gray-scale, pixel-noise, inversion) (Kumar et al., 2023). Perturbating the trajectory tensors needs to be carefully planned, to preserve the system dependencies without destroying too much relevant information. Furthermore, augmentations such as flipping or rotation aren't useful, since relative distances and velocities remain unchanged under these transformations. Therefore, three augmentations are applied by adding noise to the states and by masking the neighbor and feature dimensions. Masking the neighbor dimension randomly removes neighbors from the tensor, which reduces the risk of heavily relying on single neighbors during decision making. The neighbor dropout rate was set to 10%, meaning that each neighbor has a 10% probability of being masked. Masking the feature dimension randomly removes single features for all neighbors, which reduces dependencies on individual input channels. Since this masking affects an entire feature dimension, the feature dropout rate was set to 5%. Additive noise is applied only to state features in order to preserve the action structure and the flag feature.

4.3 Discriminator

After generating artificial trajectories, these must be compared with the expert demonstrations. For this purpose, randomly sampled trajectory windows are first compressed by the transition encoder into latent transition features, on which the discriminator D_β then operates to return a Wasserstein score matrix. The matrix has shape $[batch, frames-1, agents]$ and provides one scalar discriminator feedback per agent and transition embedding $[z_t, \Delta z_t]$. In this setup, D_β acts as a critic rather than a hard classifier, assigning higher scores to transition patterns that resemble expert behavior and lower scores to transitions that deviate from the expert demonstrations. Analogous to the generators, separate discriminators are used for

predator and prey, since both roles follow different behavioral objectives and produce fundamentally distinct trajectory patterns. During optimization, the discriminator is trained using a Wasserstein objective with gradient penalty (GP).

4.3.1 Discriminator Architecture

As input, the discriminator receives randomly sampled trajectory windows from either expert or policy rollouts. Each batch is first processed by the frozen transition encoder, which maps the raw state sequences to latent transition features of shape $[batch, frames-1, agents, 2z]$. Concretely, for each agent and time step, the encoder provides a compact representation by concatenating the latent state and its temporal change $[z_t, \Delta z_t]$. The discriminator then evaluates these agent-wise transition features rather than the high-dimensional raw sequences directly. The transition features are flattened and passed through a MLP with three fully connected hidden layers and ReLU activations. A final linear output layer produces a scalar Wasserstein score for each agent-wise transition. The resulting scores are reshaped back into a matrix of shape $[batch, frames-1, agents]$. In addition, Gaussian noise can be added to the state inputs in early training stages with a linear decay over generations. This term helps to regularize the discriminator by injecting input noise, which can prevent it from becoming overly confident early on and thereby stabilize the balance between generator and discriminator updates. For optimization, Root Mean Square Propagation (RMSProp) was used. This choice follows Yapei Wu et al. (2024) and Arjovsky et al. (2017), who report stable performance of RMSprop in non-stationary adversarial training setups comparable to the one applied here. It is a gradient-based optimization algorithm that adapts the learning rate for each parameter individually by considering the magnitude of recent gradients (Tieleman and Hinton, 2012). This adaptive structure helps to handle non-stationary objectives and sparse gradients.

4.3.2 Wasserstein Loss with Gradient Penalty

The reward signal used to update the policies is derived from a discriminator that estimates how closely the transition features of generated trajectories (τ_π) resemble those of the expert demonstrations (τ_E). From a mathematical perspective, this corresponds to minimizing the divergence between the underlying expert $\tau_E \sim \mathbb{P}_E$ and policy $\tau_\pi \sim \mathbb{P}_\pi$ distributions (Arjovsky et al., 2017). The closer these distributions align, the more accurately the policy reproduces expert-like behavior. The original GAN and GAIL formulations (Goodfellow et al., 2014; Ho and Ermon, 2016), as well as the implementation by Yapei Wu et al. (2024), rely on binary cross-entropy

loss to minimize the Jensen–Shannon Divergence (JS) between expert and generated distributions, by measuring the overlap of two probability densities (Arjovsky et al., 2017). However, it becomes problematic when the two distributions have little or no overlap, which is common in the early stages of imitation learning. In this case, the discriminator can saturate, leading to vanishing gradients and an increasingly uninformative learning signal. In an initial stage of model development, this weak discriminator feedback was observed, motivating a change of the training objective to the Earth-Mover (Wasserstein-1) distance, which already has shown strong performance in adversarial imitation learning (Zhang et al., 2020) (Equation 10).

$$W(\mathbb{P}_E, \mathbb{P}_\pi) = \inf_{\gamma \in \Pi(\mathbb{P}_E, \mathbb{P}_\pi)} \mathbb{E}_{(\tau_E, \tau_\pi) \sim \gamma} \big[\|\tau_E - \tau_\pi\| \big] \tag{10}$$

"Intuitively, $\gamma(\tau_E, \tau_\pi)$ indicates how much "mass" must be transported from τ_E to τ_π in order to transform the distributions $\mathbb{P}_E$ into the distribution $\mathbb{P}_\pi$" (Arjovsky et al., 2017, p. 4). While the JS can yield vanishing gradients when the expert and generated distributions barely overlap, the Wasserstein-1 distance remains smooth and provides a non-vanishing learning signal even when the distributions are disjoint. In GAIL, the Wasserstein-1 distance corresponds to maximizing the difference in expected discriminator scores over the class of 1-Lipschitz functions (Arjovsky et al., 2017) (Equation 11).

$$W(\mathbb{P}_E, \mathbb{P}_\pi) = \sup_{\|D_\beta\|_L \leq 1} (\mathbb{E}_{\tau_E \sim \mathbb{P}_E}[D_\beta(\tau_E)] - \mathbb{E}_{\tau_\pi \sim \mathbb{P}_\pi}[D_\beta(\tau_\pi)]) \tag{11}$$

A central requirement in Wasserstein GAIL is that the discriminator fulfills a 1-Lipschitz constraint, which is necessary for estimating the Wasserstein distance and obtaining stable gradients (Gulrajani et al., 2017; Khromov and Singh, 2024; Yapei Wu et al., 2024). Mathematically, this constraint requires that the discriminator satisfies $|D(\tau_E) - D(\tau_\pi)| \leq \|\tau_E - \tau_\pi\|$ for all inputs τ_E, τ_π, meaning that the output of the discriminator cannot change arbitrarily fast with respect to its input (Lipschitz continuity). Following the recommendation of Gulrajani et al. (2017) and Yapei Wu et al. (2024), the Lipschitz constraint is enforced using gradient penalty regularization rather than constraining the discriminator parameters directly via weight clipping (Bhatia, 2021). The GP term softly penalizes deviations of the discriminator's gradient norm from 1, encouraging the gradient of D_β with respect to its input to remain close to unit norm (Equation 12). This maintains the required 1-Lipschitz continuity while preserving sufficient model capacity (Yapei Wu et al., 2024). The penalty works because a function is 1-Lipschitz whenever its gradient norm is bounded by 1. By enforcing $\|\nabla_\tau D_\beta(\tau)\|_2 \approx 1$ on interpolations between expert and policy samples, the discriminator is prevented from forming arbitrarily steep slopes. The strength of this regularization is controlled by the coefficient λ_{GP}.

As a weighting factor, λ_{GP} determines the magnitude of the gradient penalty and thus how strongly the Lipschitz constraint influences training.

$$\mathcal{L}_{GP} = \mathbb{E}_{\tau\sim\zeta}\left[(\|\nabla_\tau D_\beta(\tau)\| - 1)^2\right] \tag{12}$$

Following this, the discriminator is trained using the Wasserstein objective combined with the GP term (Gulrajani et al., 2017). The discriminator loss is defined in equation 13.

$$\mathcal{L}_D = \mathbb{E}_{\tau_E\sim\mathbb{P}_E}[D_\beta(\tau_E)] - \mathbb{E}_{\tau_\pi\sim\mathbb{P}_\pi}[D_\beta(\tau_\pi)] + \lambda\mathcal{L}_{\text{GP}}. \tag{13}$$

4.4 Environment

To generate trajectories, the policies are executed in an environment that produces rollouts for training. Two environment configurations are used (Haeri, 2026a). The first environment is designed for evaluation, collecting metrics for further analysis and visualization. The second environment is tightly coupled to ES and uses vectorized batch processing to update policy parameters efficiently and in parallel. Environment initialization is fully derived from expert positions stored in a dedicated buffer. Reusing these expert configurations instead of random initialization improves training efficiency, as rollouts start in realistic states rather than spending the first steps moving away from arbitrary starting conditions. For mirrored perturbations, it was ensured that positive and negative perturbations were evaluated from the same initial state. The environment simulates agents in a bounded 2D arena with configurable width and height. At each step, a policy action $a \in [0, 1]$ is mapped to a turn rate based on the expert-derived maximum turn rate. The heading is updated accordingly and the position is advanced using a fixed step size. Wall collisions are handled by clamping positions to the boundary and reflecting the heading to keep agents inside the arena. For training, rollout states are processed and stacked into the same tensor representation as the expert trajectories (Section 3.2.2), with special attention to preserving consistent agent indexing across timesteps. The environment can be configured dynamically for both prey-only and predator-prey scenarios.

4.5 Training Stabilization

The model was developed in an iterative manner, where complexity was increased only after the previous stage worked reliably. It progressed from component checks and supervised ES baselines, to policy training with an MSE objective, and finally to the discriminator-based reward using the original expert tensor structure. Throughout this process, several challenges such as generator-discriminator imbalance or exploding gradients needed to be addressed. Training stability therefore added the most complexity to the overall pipeline. While stabilizing adversarial training is a well-known problem in the literature (Ho and Ermon, 2016; Hui, 2018), it becomes even more critical in the present co-evolutionary predator-prey setting, where multiple policies and discriminators are optimized simultaneously. Here, the reward signal does not depend on a single generator-discriminator pair, but on the stability of all predator and prey networks trained in a shared loop. Even if the prey networks remain well-balanced, insufficient learning progress by the predator can affect overall convergence. To tackle these challenges, the following methods are applied:

- **ES-Clipping** is applied to prevent exploding updates during the ES step. The update is bounded by comparing the norm of the raw step $\|\Delta_{\text{raw}}\|$ to the current parameter norm $\|\theta\|$ and setting a maximum step size $\Delta_{\max} = \alpha\|\theta\|$, where α is the relative clipping factor. If $\|\Delta_{\text{raw}}\| > \Delta_{\max}$, the step is rescaled to satisfy $\|\Delta_{raw}\| \leq \Delta_{\max}$ before applying the update (Liu et al., 2019).
- **Exponential Moving Average (EMA)** smooths noisy updates by continuously averaging the current policy parameters over time, maintaining a slow-moving copy of the network weights (Morales-Brotons et al., 2024). Training was unstable with large jumps between generations, amplified by the stochasticity of the policy output. EMA stabilizes the learning and typically yields more consistent behavior and better generalization. The EMA parameters $\bar{\pi}_{\phi,\varphi}$ are initialized after the first ten generations to reduce bias from the initial training jumps. They are updated every fifth generation and used to generate the initial rollouts at the beginning of each generation.
- **Discriminator-Update-Factor** controls how often the discriminator is updated per generation before the policies are updated with ES. Updating the discriminator multiple times helps to compensate for balancing issues and keeps the reward signal stable.
- **Discriminator Noise** is used as an additional regularization to reduce early training imbalance between policy and discriminator. Gaussian noise is added only to the state channels of both expert and generated batches, while the action channel remains unchanged. The noise is applied only during the first half of training and is linearly decayed to zero over generations, which stabilizes

the reward signal in early stages without affecting late-stage convergence (Lee and Seok, 2020).

- **BC-pretraining** gives the policy parameters an initial direction instead of starting from random weights. During BC pretraining, the policies are trained on expert actions in a supervised manner. The objective is to minimize the MSE between the expert heading change and the policy's μ of the Gaussian action distribution, which yields a stable, deterministic pretraining target and avoids learning stochasticity in this stage. Pretraining with behavior cloning is also used in the original GAIL formulation by Ho and Ermon (2016).
- **Parameter Decay** is applied to both the learning rate λ and the exploration factor σ (Yapei Wu et al., 2024). The objective is to gradually reduce update magnitude and exploration over training, allowing large steps early on while making refinement more stable as the policies get closer to expert behavior.

As highlighted in Section 2.3, one of the most prominent failure modes in GAIL is mode collapse, where the policies converge to a limited subset of behavioral patterns that are consistently favored by the discriminator. Several implementation choices were aimed at mitigating this issue. First, policy stochasticity increases sample diversity, as the Gaussian action distribution can produce different actions in identical states. Second, especially in early training, the ES exploration factor σ further reduces the risk of collapsing into a tight local mode by encouraging broader parameter-space exploration. Third, discriminator regularization via gradient penalty and input noise stabilizes adversarial training and prevents the discriminator from becoming overly confident too early, which helps to maintain informative learning signals for the policies. Following Yifan Wu et al. (2025), feature weighting was considered as an additional countermeasure, but it was not included in the final implementation due to poor cluster separation in high-dimensional space (Section 5.4).

4.6 Performance Evaluation

To evaluate GAIL convergence, it is necessary to quantify how closely the policy-generated rollouts match the expert data distribution. For this purpose, in addition to the Wasserstein distance, Maximum Mean Discrepancy (MMD) and Sinkhorn approximation are used and logged throughout training to track the evolution of distribution matching. Since a single metric may be misleading, these metrics are used for evaluation, quantifying how closely the distributions align from different perspectives. To obtain an objective baseline, all metrics are additionally computed between two randomly sampled expert batches in a Monte-Carlo (MC) setup, allowing the definition of a mean objective and its variability as baseline. Policy-generated

behavior is considered expert-like when its distance to the expert data falls within the variability of the expert distribution.

Maximum Mean Discrepancy (MMD) is a kernel-based method used to quantify how similar two distributions are by measuring the distance between their mean embeddings (Mehraban and Pichler, 2025). In practice, the samples are compared through a radial basis function (RBF) kernel, which assigns higher similarity to pairs of points with smaller Euclidean distance (Beer, 2026).

Sinkhorn approximation and the Wasserstein distance are closely related and both arise from optimal transport theory (Luise et al., 2018). Both measure how much "effort" is needed to transform one distribution into another by optimally moving probability mass between samples. Given two data samples, Sinkhorn constructs a pairwise cost matrix that defines how expensive it is to move mass from one point to another. The main difference is that Sinkhorn introduces an entropy regularization term, resulting in a smoother transport plan and a computationally efficient iterative solution. As a consequence, Sinkhorn provides an approximation that can introduce additional bias compared to the unregularized Wasserstein distance.

In this study, the Wasserstein distance is approximated by the score gap between expert and policy discriminator outputs. Sinkhorn distance is computed on the encoder transition embeddings, while MMD is computed directly on sampled expert and policy batches, providing complementary perspectives on the distribution matching task. The metrics are computed after every evaluation step. The final model was chosen based on the best Sinkhorn distance, since it is computed on the transition embeddings and therefore directly compares the latent representation of the expert's system dynamics with the policy-generated ones.

4.7 Summary of Methodology

The GAIL framework imitates predator-prey dynamics by reproducing expert-like trajectories from video recordings. Separate modular policies are trained for both roles and used to generate rollouts in a 2D environment initialized from expert states. Expert and generated trajectories are compared by discriminators using a Wasserstein objective with gradient penalty. To enable sequence processing, a VICReg-pretrained transition encoder is frozen during GAIL training. Policy updates follow a ES scheme based on mirrored perturbations and rank-normalized rewards. To stabilize training, additional design choices are applied. Convergence is monitored with MMD and Sinkhorn distances, interpreted relative to a expert baseline. The following chapter presents training results across different policies and expert trajectories, illustrating how well the framework can imitate predator-prey systems.

CHAPTER 5

Training and Results

Following the approach of Yapei Wu et al. (2024), the training process is implemented as a cooperative co-evolutionary framework (Algorithm 1). Co-evolution refers to decomposing the high-dimensional problem into two modular subproblems that are optimized separately. Allowing each module to focus on its primary function, reducing training complexity, and enabling faster convergence. In the predator–prey setting, the predator's modular networks are updated first, followed by those of the prey. A full list of software dependencies, including version numbers, is provided in Appendix F, Table 10.

Algorithm 1 GAIL procedure to imitate predator-prey dynamics

Input: Policies $\pi_{P_{\phi,\varphi}}$ and $\pi_{Y_{\phi,\varphi}}$, Discriminators $D_{P,\beta}$ and $D_{Y,\beta}$, Encoders $E_{P,\psi}$ and $E_{Y,\psi}$, Expert trajectories $\tau_{\mathrm{E,P}}$ and $\tau_{\mathrm{E,Y}}$
Output: Best predator $\pi^{\star}_{P_{\phi,\varphi}}$ and prey policy $\pi^{\star}_{Y_{\phi,\varphi}}$

Pretrain $\pi_{P_{\phi,\varphi}}$ and $\pi_{Y_{\phi,\varphi}}$ with BC
Train and freeze $E_{P,\psi}$ and $E_{Y,\psi}$
Initialize EMA policies $\bar{\pi}_{P_{\phi,\varphi}}$ and $\bar{\pi}_{Y_{\phi,\varphi}}$
for $i = 0, 1, 2, \ldots, M$ generations **do**
    Execute $\bar{\pi}_{P_{\phi,\varphi}}$ and $\bar{\pi}_{Y_{\phi,\varphi}}$ to collect rollouts
    for $j = 1, \ldots, J$ $D_{P,\beta}$-update-ratio **do**
        update predator discriminator $D_{P,\beta}$
    end for
    for $j = 1, \ldots, J$ $D_{Y,\beta}$-update-ratio **do**
        update prey discriminator $D_{Y,\beta}$
    end for
    optimize predator PIN $\pi_{P_{\phi}}$ and AN $\pi_{P_{\varphi}}$ with ES
    optimize prey PIN $\pi_{Y_{\phi}}$ and AN $\pi_{Y_{\varphi}}$ with ES
    update EMA: $\bar{\pi}_{P_{\phi,\varphi}} \leftarrow \mathrm{EMA}(\pi_{P_{\phi,\varphi}})$, $\bar{\pi}_{Y_{\phi,\varphi}} \leftarrow \mathrm{EMA}(\pi_{Y_{\phi,\varphi}})$
    apply decay γ on learning rates λ_P, λ_Y and exploration factor σ_P, σ_Y
end for

Before training, the hyperparameter settings had to be defined. The initial values were taken from Yapei Wu et al. (2024) and were fine-tuned over time until

training stabilized and convergence was observed. For the generator training setup, `max_steps` determines the rollout length of the expert simulation. `num_generations` controls the total number of training generations, while `dis_balance_factor` specifies how many discriminator updates are performed per generation. The learning rates `lr_policy` and `lr_disc` define the update step sizes for the policy and discriminator, respectively. For the ES updates, `num_perturbations` sets the number of perturbations evaluated per submodule and generation, while the exploration factor `sigma` controls the magnitude of these perturbations. Discriminator training is regularized using `lambda_gp` as the gradient penalty weight and `noise` applied to the discriminator inputs. Finally, `update_mode` defines how the discriminator-based reward is computed, allowing the use of additional avoidance and attack rewards.

Following the model development process, complexity was increased step by step. First, a prey-only model was trained on simulated trajectories generated with the Couzin model, allowing the training to focus on a single policy while the expert trajectories remain clean and gap-free (Haeri, 2026a). Second, a predator-prey model was trained on simulated data and finally adapted to the processed trajectories from the video recordings.

5.1 Couzin Prey-Only Policy

The prey-only policy trained on Couzin-simulated expert data targets the same imitation objective as in Yapei Wu et al. (2024), who also trained a single-species model on Couzin-generated trajectories. While some implementation details differ, this experiment provides a comparable reference point before moving to the predator-prey models. The final hyperparameter configuration is reported in Appendix C, Table 3. The model was not pretrained with BC in order to isolate the performance of GAIL.

Before starting GAIL training, the prey transition encoder is pretrained on the expert data. As a brief recap of the VICReg loss components, invariance pulls both views together in embedding space, variance prevents representation collapse by enforcing a minimum standard deviation, and covariance reduces redundancy by decorrelating embedding dimensions. Figure 9 reports the three loss components over pretraining. Invariance quickly reaches a low plateau, indicating that the encoder learns consistent representations under the applied trajectory perturbations. At the same time, the variance term drops strongly, suggesting that collapse is avoided and the embedding maintains sufficient diversity across samples. Covariance stabilizes at a non-zero level, indicating remaining dependencies between embedding dimensions, but overall a stable representation is obtained.

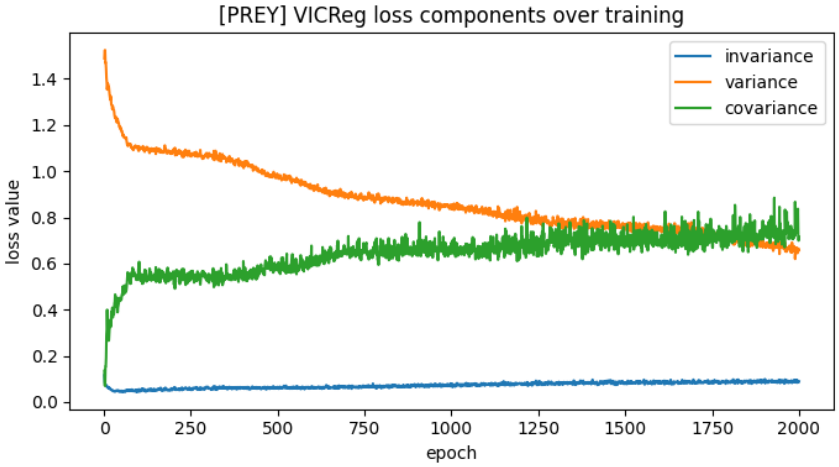


Figure 9: VICReg loss components over training for the Couzin prey-only model.

For the prey-only experiment, the training metrics are briefly explained once as a reference, while subsequent experiments only report the main trends. The left plot shows the absolute discriminator scores over time for expert and policy-generated trajectories (10). The right plot shows the magnitude of the discriminator score gap, serving as a proxy for the Wasserstein distance. The overall objective is that the discriminator cannot distinguish between expert and policy-generated behavior, which would be represented by a Wasserstein proxy close to zero in the perfect case. If the difference becomes large, the absolute score curves drift further apart, indicating that the discriminator learns to separate the behaviors. Throughout development, a standard pattern was observed: small differences at the beginning, where the discriminator still needs to learn how to differentiate, larger differences during mid-term training, and a decrease in difference towards the end near convergence.

Until 750 generations, expert and policy scores remain close, indicating that the discriminator struggles to separate both distributions (Figure 10). From 900 to 2300, the score gap increases and remains high but bounded, suggesting an informative reward signal and stable learning. Around 2400, the policy closes the gap again, making discrimination harder and indicating convergence towards expert-like trajectories.

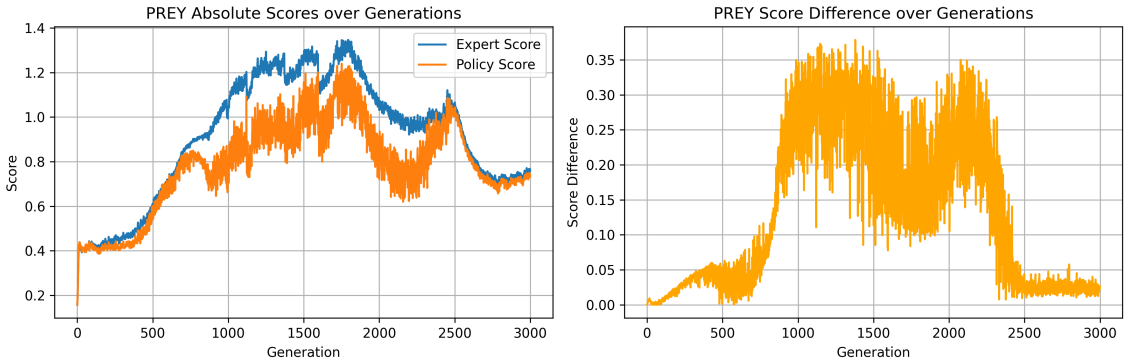


Figure 10: Expert and policy discriminator scores and their difference over generations for the Couzin prey-only model.

The ES step size became an important metric when dealing with exploding gradients and provides a complementary view on how strongly the policy parameters are changed from one generation to the next. It is regularized by the update parameter α, which constrains the magnitude of each parameter update and thus enforces smaller changes of the parameter vector per generation, acting similarly to a learning-rate mechanism in gradient-based optimization. The logged value corresponds to the norm of the applied parameter update, reported separately for PIN and AN. A stable and steadily rising step size indicates smooth training without extreme jumps and stable gradients. The reward-difference standard deviation describes how variable the discriminator-based reward signal is across perturbations: higher variability indicates noisier and more exploratory updates, whereas lower variability suggests a more consistent and target-driven signal. When approaching convergence, this standard deviation should also diminish.

Across training, the ES step size remains bounded while gradually increasing (Figure 11). The occasional jumps towards the end are likely late-stage fluctuations around a (local) optimum or plateau. Reward-difference variability peaks in mid training ($\sim$ 1200), reflecting noisier and more exploratory updates, and decreases clearly towards the end, consistent with a more stable learning signal and convergence.

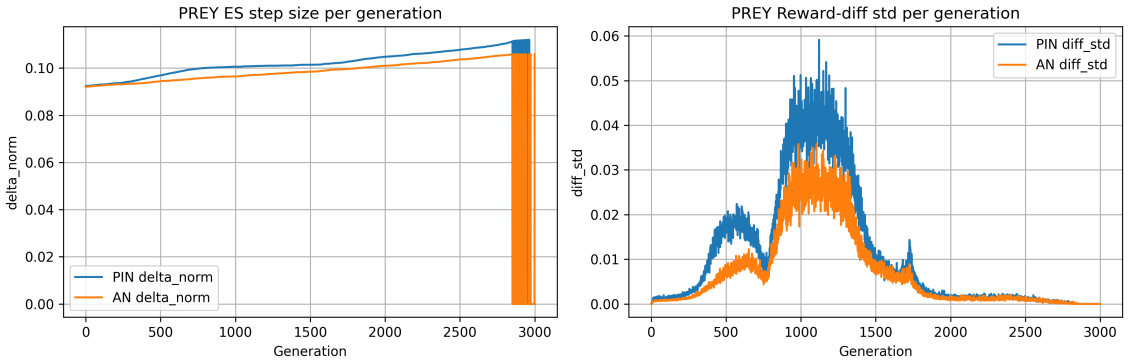


Figure 11: ES update step size and reward difference over generations.

As already introduced in Section 4.6, MMD and Sinkhorn distance serve as performance evaluation metrics, measuring the distance between expert and policy-generated data in different ways. The red baseline serves as the training objective. If the MMD or Sinkhorn distance reaches this line, the policy-generated trajectories closely match the expert distribution under the respective metric.

Both MMD and Sinkhorn show a clear convergence signal by reaching the expert-related baseline (Figure 12). After 1700 generations, both distances drop sharply and remain close to zero, suggesting a strong alignment between generated and expert distributions.

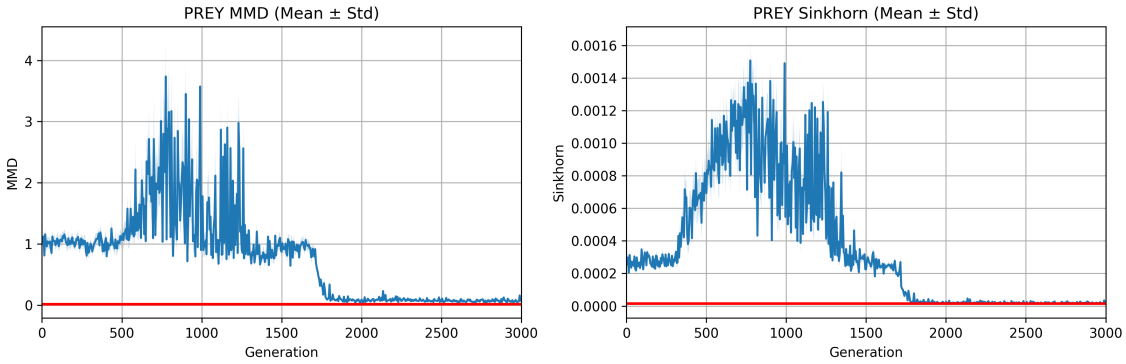


Figure 12: MMD and Sinkhorn distances over generations.

After successfully training the prey-only model on simulated data, which was also visually confirmed by running rollouts with the trained policy, the general model architecture can be considered a working foundation for multi-agent swarm imitation. In the next step, complexity is increased to serve the predator–prey imitation setting.

5.2 Couzin Predator-Prey Policy

As in the prey-only case, hyperparameters were initially set to the same baseline values and fine-tuned across runs as training stability improved. The final hyperparameter settings are listed in Appendix C, Table 4. Again, the model was not pretrained with BC.

For the prey encoder, invariance again converges rapidly to a low plateau, but the variance term drops more strongly over training, indicating better avoidance of collapse and higher embedding diversity. Covariance increases and stabilizes at a non-zero level, implying residual correlations, but overall the representation remains stable. Figure 13 shows stable VICReg pretraining for the predator encoder. The invariance term quickly reaches a low plateau, while the variance term decreases only

moderately and remains comparatively high. Covariance stabilizes with noticeable fluctuation, indicating remaining dependencies between embedding dimensions.

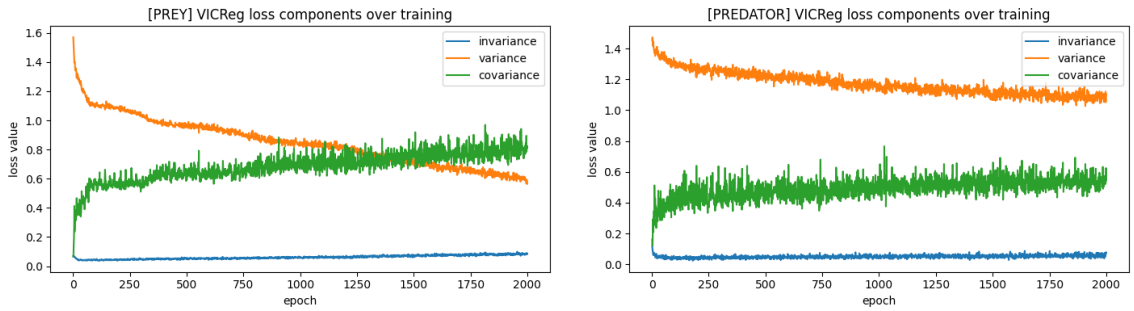


Figure 13: VICReg loss components over training for the Couzin predator–prey model.

Figure 14 highlights a clear role asymmetry. Prey shows a distinct mid-training separation followed by a sharp drop of the Wasserstein proxy around 1400 where the closest gap occurs, before drifting apart again towards the end. Predator score differences stay considerably larger and more volatile throughout training, showing no comparably sharp convergence phase.

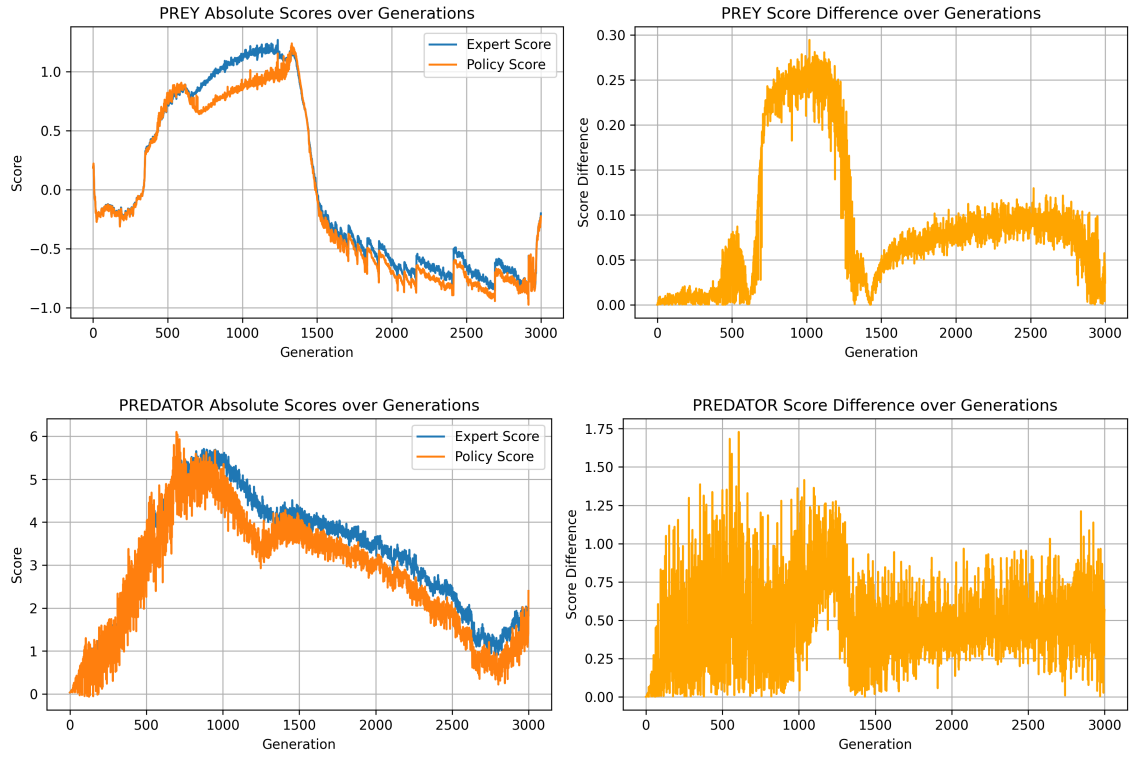


Figure 14: Expert and policy discriminator scores and their difference over generations for the Couzin predator-prey model.

For both roles, the ES step size increases consistently over generations and remains stable, indicating solid parameter updates (Figure 15). Reward-difference variability is elevated early on and reaches its minimum for the prey around generation 1400, matching the observed minimum in score distance. For the predator, AN variability stays relatively consistent across training, whereas PIN variability decreases after mid-term training and stabilizes. Towards the end, prey variability increases again.

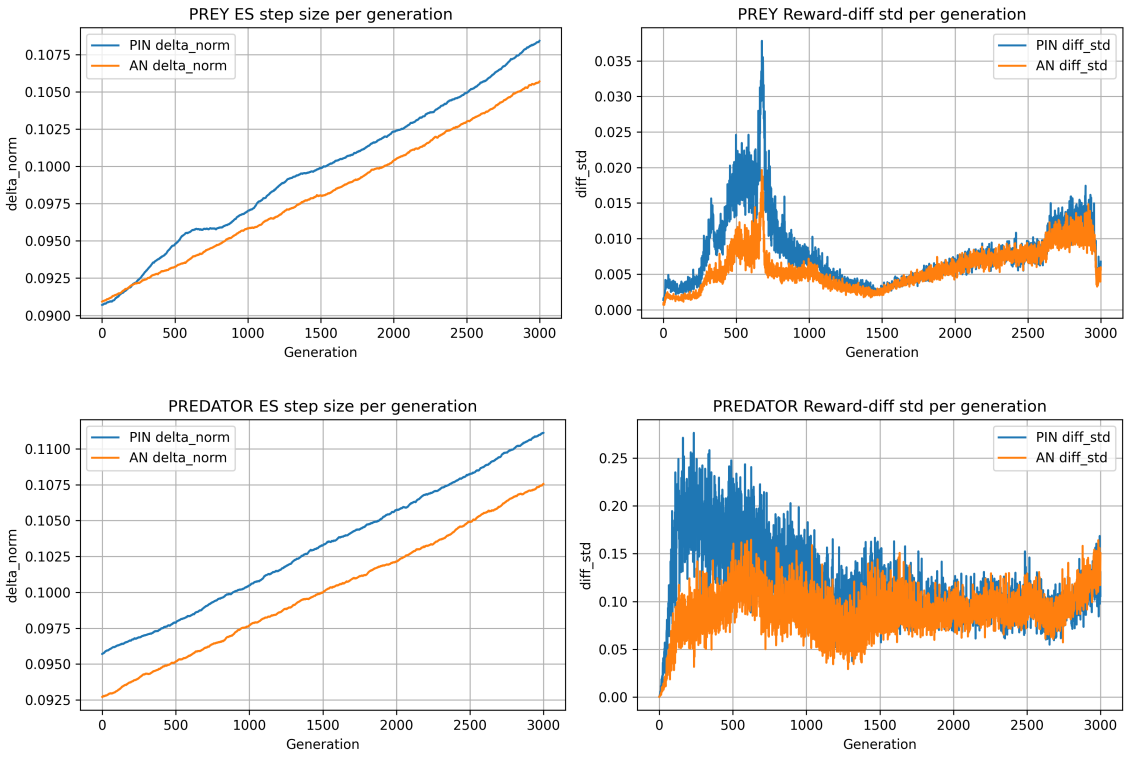


Figure 15: ES update step size and reward difference over generations for predator and prey.

For prey, both MMD and Sinkhorn peak early and drop afterwards. Sinkhorn briefly reaches the expert baseline around 1100–1400, whereas MMD remains above it (Figure 16). After a long stable phase, both distances increase again towards the end. For the predator, MMD stays noisy and above the baseline, while Sinkhorn remains close to the baseline mainly during mid training before rising again near the end. Overall, the Couzin predator-prey training shows only a short convergence phase for the prey, while the predator remains volatile, so a stable joint convergence is not achieved.

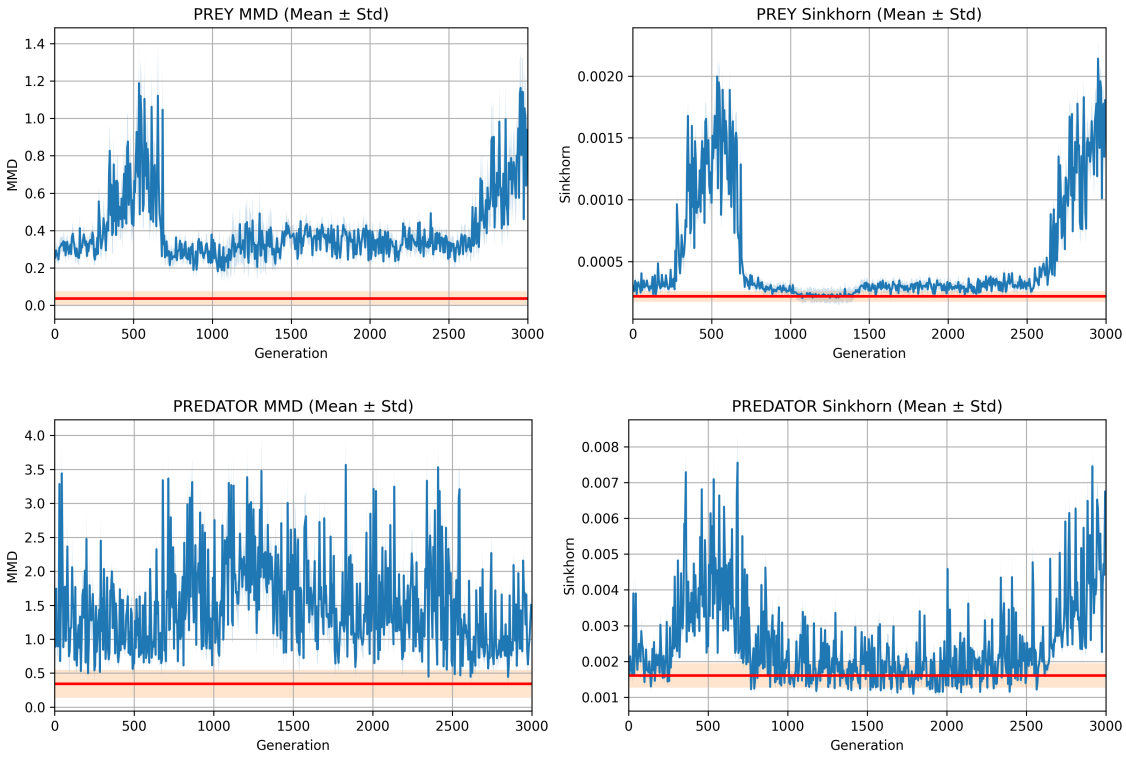


Figure 16: MMD and Sinkhorn distances over generations for predator and prey.

5.3 Video Predator-Prey Policy

The last complexity step was to train the predator–prey GAIL on the extracted video trajectories. To match the scaling of the expert data and the generated data, the environment needs to be adapted to the video settings accordingly. The area width and height were set based on the video resolution to 2160×2160 pixels. Analyzing the maximum per-frame speeds extracted by the DeepSORT tracker showed that, after cleaning outliers, the speeds are around 10 pixels per frame. Therefore, both predator and prey max speeds were set to 10, with a step size of 1 (Appendix B, Figure 36 and Figure 37). The maximum turn rate was derived from the action $\Delta\theta$ by analyzing its quantiles (Appendix B, Table 2 and Figure 35). Within the 99% quantile range, action values lie within $\pm 18°$, meaning that 99% of the observed turning behavior is captured by this range. The hyperparameter configuration is reported in Appendix C, Table 5.

The Video GAIL policies were pretrained with BC on the expert data. Figure 17 shows the training progress with a stable and steady decrease of the MSE loss over epochs for both predator and prey. Early stopping terminated training after no

improvement for 10 consecutive epochs, which allows initializing the policy parameters with a meaningful direction instead of random weights, while still preserving generalization rather than perfectly memorizing the training trajectories. For the prey policy, early stopping triggered after 185 epochs and for the predator after 473 epochs.

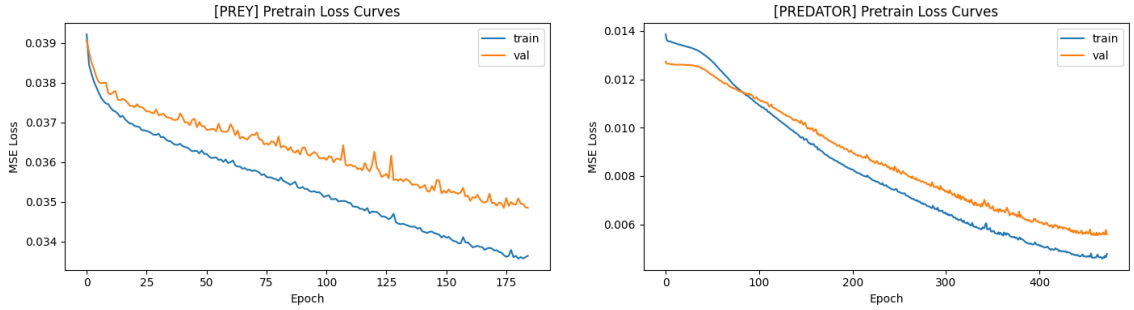


Figure 17: BC training curves with early stopping for predator and prey.

For the prey encoder, invariance stays low but increases slightly over time, meaning that the embeddings of both augmented views become less aligned (Figure 18). This is tightly coupled with the strong reduction of the variance term, as the embedding maintains sufficient spreads, the mean squared distance between both views becomes harder to further squeeze close to zero. At the same time, the variance term drops strongly and approaches a low value, indicating robust prevention of collapse and high embedding diversity across samples. Covariance increases quickly and stabilizes at a non-zero plateau with occasional spikes, implying remaining dependencies between embedding dimensions, but overall stable representations. For the predator encoder, training reaches a stable plateau. Invariance stays low throughout training, while the variance term steadily decreases but remains clearly above zero. Covariance rises early and then stabilizes with persistent fluctuations.

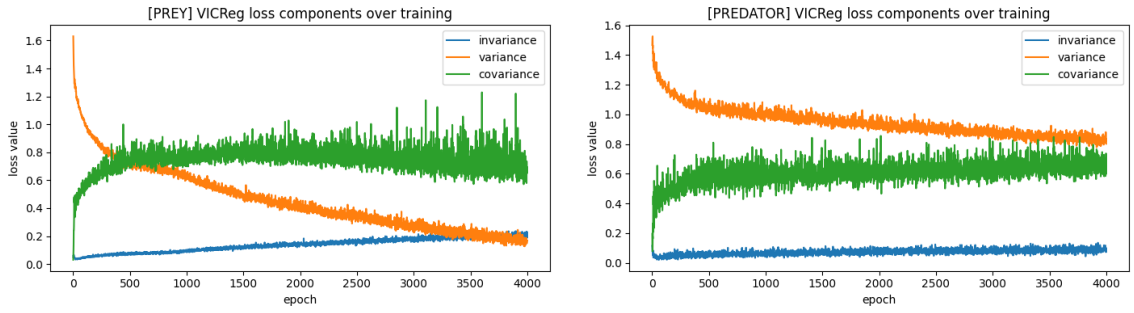


Figure 18: VICReg loss components over training for the video predator-prey model.

Having a look at the prey discriminator scores, it can be observed that the discriminator starts to separate expert and policy trajectories after around 500 generations (Figure 19). Afterwards, the gap increases slowly and shows its strongest volatility

around 2300–2800. Towards the end, both the Wasserstein proxy and its variance decrease again, indicating a more stable but still non-zero separation. The predator logs show a different signal. The score difference increases early on, briefly drops around 900–1200, indicating a short phase of weaker separation, before rising again with much larger variance after 1500. Overall, the predator score gap stays high and volatile, indicating that the discriminator keeps separating expert and policy trajectories.

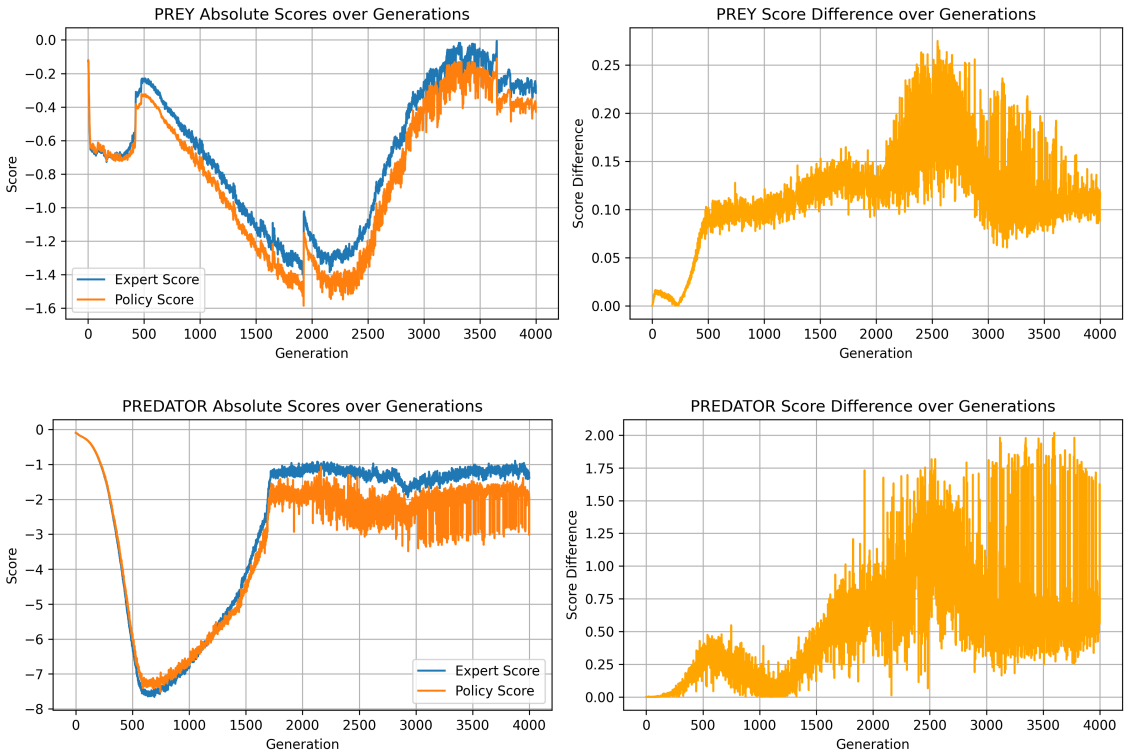


Figure 19: Expert and policy discriminator scores and their difference over generations for the video predator-prey model.

The ES step size for both predator and prey shows a strong increase until around 700–800 generations, followed by a clear bend and a slower but steady increase afterwards (Figure 20). At the same time, the reward-diff standard deviation drops after this early phase, but rises again during mid-to-late training, with a much larger magnitude for the predator. Towards the end, variability decreases again, indicating more stable updates around generation 3700.

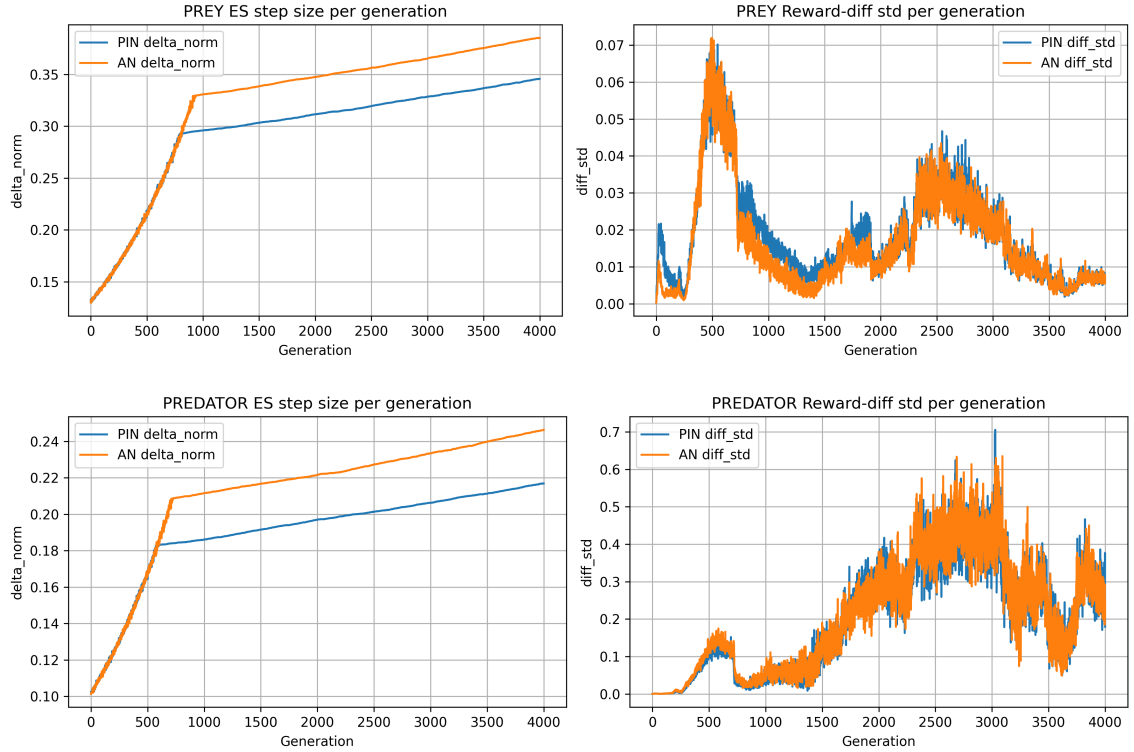


Figure 20: ES update step size and reward difference over generations for predator and prey.

For the prey, MMD stays high and relatively flat during early training and drops strongly only after around 2300 generations but still showing high volatility, without touching the objective line (Figure 21). Sinkhorn shows a similar behavior, it remains clearly above the baseline for most of training, then briefly reaches it around 2300–2600, before increasing again with higher deviation towards the end. For the predator, both metrics show a similar pattern, remaining highly noisy and not showing sustained convergence.

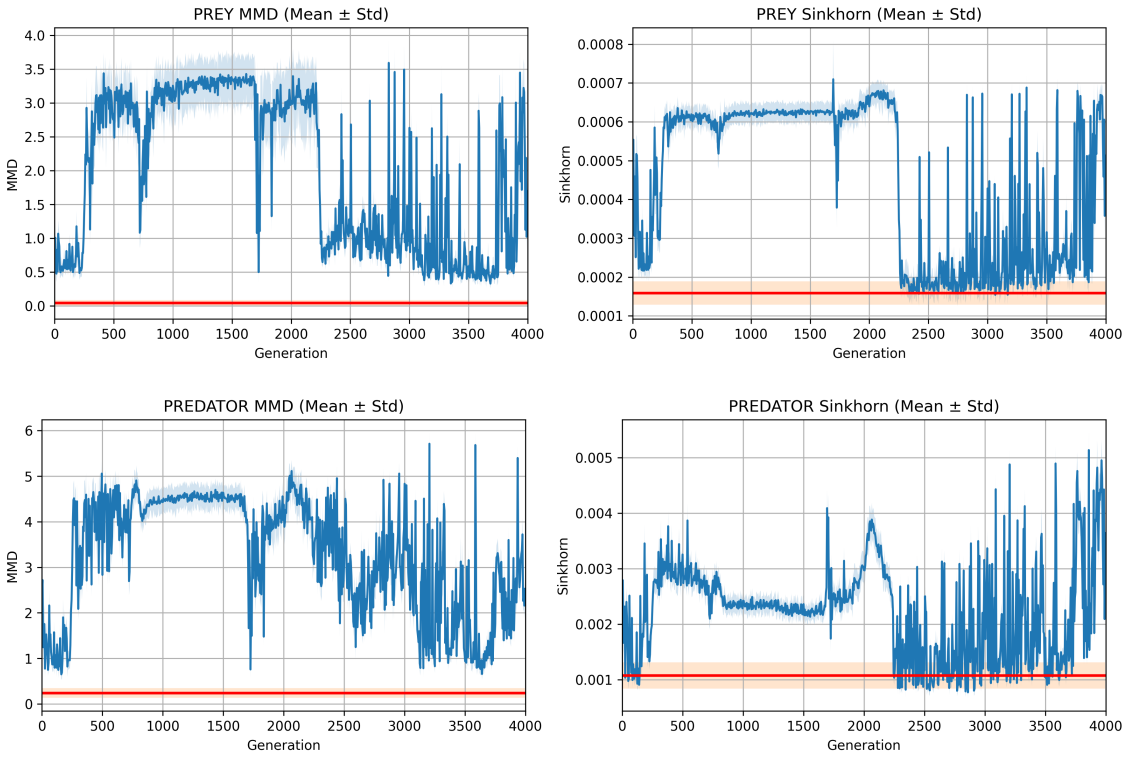


Figure 21: MMD and Sinkhorn distances over generations for predator and prey.

Since the training success cannot be fully assessed from the reported metrics alone for both predator–prey GAILs, a visual inspection of the learned policy behavior is required. The rollouts show that the learned policies are partially able to imitate the expert behavior, but still lack a clear and persistent predator–prey interaction. For example, prey behavior remains similar to the prey-only policy, while the predator steers towards the prey swarm but often bypasses individuals without triggering consistent avoidance or attack dynamics. Overall, the inter-group dynamics of clear prey evasion and predator pursuit are not reliably reproduced, which can also be observed in Figure 22. The right image shows the expert behavior. The distance to the predator is larger, the prey are cornered, and escape alignment tendencies become visible. The left image, in contrast, shows policy-generated behavior, where avoidance and attack dynamics are largely missing.

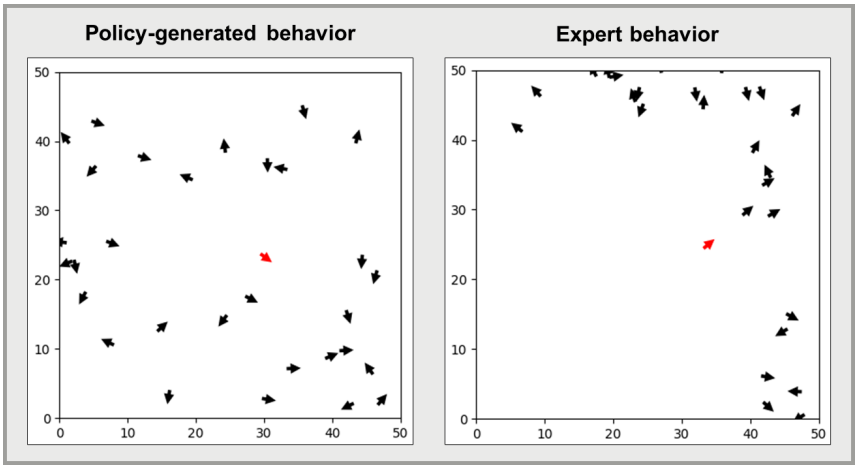


Figure 22: Comparison of policy-generated with expert avoidance behavior.

5.4 Avoidance and Attack Behavior

To address the missing inter-group dynamics, several approaches were implemented and are summarized in the following. Some of them were already introduced as part of the architecture in Section 4. Initially, the predator was only represented implicitly by its fixed position in the prey tensor at the first index. Later, an additional flag feature was added to explicitly mark the predator as different.

As already noted in Section 3, the ratio of attack to non-attack scenes in the input data is extremely underrepresented, with 0.12% of the total video duration. Furthermore, reflections and waves caused by the predator reduced tracking quality, making the pipeline insufficient for relevant attack scenes. Therefore, attack scenes were hand-labeled to ensure consistent trajectories and heavily oversampled during training. Even training on attack scenes only did not lead to the desired avoidance and attack behavior. To exclude data quality as the main limitation, the expert data source was switched to Couzin-simulated trajectories (Haeri, 2026b), providing clean and gap-free predator–prey rollouts showing permanent predation behavior.

Next, the action aggregation mechanism of the modular policy was adjusted. As described in Section 4.1.2, the final action is computed as a normalized weighted sum of neighbor-wise actions a_i and the corresponding weights ω_i. Multiple variants were tested to enforce a stronger predator influence. For example, aggregation was applied only to prey actions and then coupled with the predator-related action using a mixing ratio. This ratio was defined by distance to the predator, by predator–prey attention weights, and as a fixed hard-coded survival-pressure. In addition, an extra attention head was implemented to derive only predator–prey weights in a one-to-one relationship while ignoring prey–prey interactions. None of these variants produced reliable escape or pursuit behavior.

Since neither data quality nor action aggregation resolved the issue, the learning signal was analyzed next, focusing on the discriminator. Initially, the discriminator was applied in an agent-wise view, meaning it received as input the perspective of one single agent with all its neighbors and features without the corresponding action $[agent = 1, neighbors = 32, features = 4]$. Rethinking this approach, the model may discard too much contextual information in the process, enabling it to learn an agent-wise perspective while neglecting the global view on the whole system. This motivated the use of the transition encoder derived from the work of Yifan Wu et al. (2025), training the discriminator on a representation of the original input data. Furthermore, Yifan Wu et al. (2025) applied feature clustering in the latent transition space of the encoder output to identify clusters reflecting behavioral patterns and to weight the discriminator reward accordingly. This idea was adapted to to find clusters representing pursuit-related patterns. PCA was used for dimensionality reduction and both KMeans and Gaussian Mixture Models were tested. However, cluster evaluation indicated no clear separation and strong overlap, so the approach was not used further (Appendix D, Figure 38 and Figure 39).

Finally, discriminator reward shaping was applied. Instead of using only the mean discriminator output, two additional rewards were added: an avoidance reward for prey that increases with predator distance, and an attack reward for the predator that minimizes the distance to the nearest prey. The influence was controlled by λ_{avoid} and λ_{attack}. Multiple settings were tested, including training with avoid/attack rewards only and different magnitudes. The additional rewards did not produce pursuit behavior and often reduced training stability, behaving more like an additional noise term than a fine-tuning objective.

Getting closer to the submission deadline, further improving the model's ability to successfully imitate pursuit behavior was not feasible. While both predator–prey models show tendencies towards the desired behavior and partial imitation, they still struggle to capture the full system dynamics. Further approaches on how predator–prey behavior might still be imitated successfully are discussed in Section 7.

5.5 Summary of Training

In summary, the proposed GAIL pipeline is able to reproduce collective motion from expert demonstrations, as shown by the prey-only model trained on Couzin-generated trajectories, which is consistent with the findings of Yapei Wu et al. (2024). Extending the setup to predator–prey dynamics increases the model complexity. While training on Couzin data yields only a short and unstable convergence phase, training on video-derived trajectories does not lead to sustained convergence across roles. Different approaches were explored to recover missing inter-group dynamics, but none produced reliable pursuit and avoidance behavior.

CHAPTER 6

Experiments

After training the model on the real video recordings, different experiments are conducted to provide deeper insights into the training process. First, the learned policies are compared against the expert data and the Couzin model using a set of swarm metrics. Second, the modular policies are evaluated in detail to gain insights into the artificial black box, using heat maps to increase explainability. In the next step, prediction errors are calculated per simulation step, making deviations from the expert trajectories transparent. Finally, the composition of the swarm is analyzed using attention weights, allowing the identification of swarm leadership and providing further insights into swarm decision-making. All analyses are performed on the Video GAIL model to evaluate the final pipeline, even though training did not fully converge.

6.1 Model Comparison on Swarm Metrics

To assess imitation quality on swarm level, the trained model is compared to the expert data and to Couzin-model simulations, which serve as a plain rule-based baseline. Figure 23 reports polarization, angular momentum, and the degree of sparsity over time. Polarization expresses how strongly the agents align their heading (Li et al., 2023; Yapei Wu et al., 2024). A perfect score of 1 indicates that all fish swim in the same direction, while low values close to 0 indicate unordered motion. Angular momentum expresses how strongly the swarm shows collective rotation around its center (Yapei Wu et al., 2024). High values indicate milling behavior, while low values close to 0 indicate no rotation. Since velocities are normalized $\|v_{\text{norm}}\| = 1$, the scale of the metric is bounded by the maximum achievable radius within the arena. For an environment of $2160 \times 2160\ px$, this yields an upper bound of $r_{\max} \cdot v_{\text{norm}} = 2160/2 \cdot 1 = 1080\ px$. The degree of sparsity describes the spatial composition of the swarm, defining how spread out the group is (Ishiwaka et al., 2022; Li et al., 2023). The value is calculated as the mean distance to the center of mass and can therefore be directly interpreted as a distance measure in pixels. A value of 0 corresponds to all fish perfectly overlapping, whereas the highest value is equivalent to the largest possible distance from the center of mass. In theory, the

upper bound is given by half the maximum possible distance in the environment, corresponding to half the diagonal ($\sim 764\ px$). Low values indicate a compact, dense swarm, while high values indicate larger separation.

The average polarization across all expert data shows a mean value of 0.68 with a large standard deviation of ± 0.28 (Figure 23). The polarization scores of the trained GAIL remain continuously below the expert mean and only reach the lower bound of the expert band for a short period of time. Overall, GAIL shows a volatile behavior, ranging roughly from $0.05 - 0.5$. The Couzin model, on the other hand, is less volatile and primarily stays in the range of $0.1 - 0.4$, while towards the end it also approaches the expert variability. Both models capture moderate alignment as observed in the real fish, but neither consistently reaches the expert mean level.

Interpreting the angular momentum allows a different view on the models. The expert values are comparatively small, with a mean momentum of 51.38 and a standard deviation of ± 64.14, indicating that milling behavior is not dominant in the expert data. The Couzin model shows similar results, as its angular momentum stays most of the time within the expert variability. In contrast, the trained GAIL model shows strong differences with high volatility and values that are primarily above the expert mean, including several peaks that exceed the expert range. However, angular momentum has to be interpreted with caution, as the overall goal is to imitate pursuit behavior rather than producing milling motion.

Looking at the degree of sparsity, both the Couzin and GAIL model start with a high mean distance to the center of mass of around 200 px and decrease over time towards a more stable range around 140 px, requiring a few steps until the system reaches a more ordered state. However, both models still differ from the expert distribution, which shows a mean sparsity of 93.91 px with a standard deviation of $\pm 13.88\ px$. This indicates consistently dense swarms in the expert recordings, with comparatively low distances to the swarm center and only small variation across sequences.

Overall, both models fail to capture the expert behavior when comparing them on the basis of different swarm metrics. GAIL fails to show similar behavior across all three metrics, while Couzin mainly suffers in polarization and sparsity. Of course, it should not be ignored that the training was incomplete, showing certain gaps between expert data and trained behavior due to the lack of inter-group interaction. However, based on the metrics, it is clear how large this difference still is.

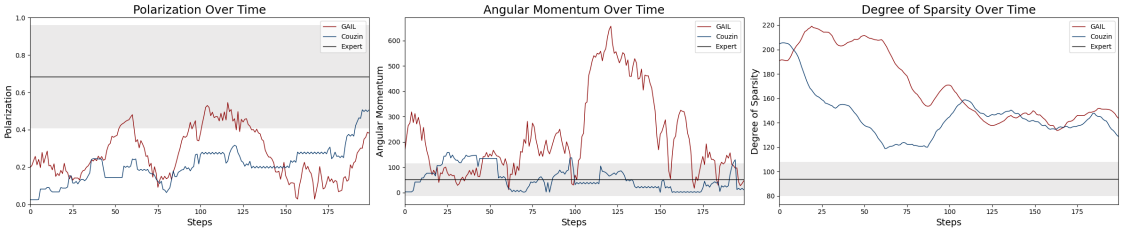


Figure 23: Model comparison on swarm metrics.

To also capture the presence of the predator in the system, additional predator-related metrics are calculated (Figure 24). Two distance measures are analyzed: the mean distance of the prey swarm to the predator and the predator's distance to its nearest prey. The plots are scaled to the original video resolution of 2160×2160 px, thereby reporting distances in true pixel units. As a third metric, escape alignment is calculated. It is similar to the polarization score and describes the prey alignment during escape from the predator (Bartashevich et al., 2024). The metric ranges from $+1$, showing perfect heading alignment away from the predator, to -1, indicating alignment toward the predator.

Interpreting the distance measures, the GAIL model and the expert data show strong alignment when focusing on the global view between the predator and the prey swarm (Figure 24). The GAIL average distance stays within the expert standard deviation of ± 416.04 px and close to the expert mean of 1015.34 px. The Couzin model, in comparison, stays mostly below the expert mean, indicating smaller average distances. But again, this metric has to be interpreted with caution, as the mean distance can be influenced by the spatial distribution of the prey and may therefore appear plausible even without realistic predator-prey interaction. For this reason, the predator distance to the nearest prey was added as an additional evaluation criterion. For both models, the distance to the nearest prey from the predator perspective is considerably smaller than the expert mean of 750.51 ± 379.63 px, with GAIL approaching the lower bound of the expert variability band. As explained in Section 3.2, predator attack scenes are extremely underrepresented in the recordings. The comparatively large expert distance could therefore be driven by the predator being present in the aquarium without consistently approaching prey. The Couzin model shows extremely small nearest-prey distances, which can be explained by the simplified pursuit dynamics. Since predator and prey operate at the same speed, once prey enters the predator's attack range, the predator can follow it persistently while the prey flees, keeping the same distance continuously.

To further understand how good the models imitate avoidance behavior, escape alignment gives clearer insights. The expert mean is negative (-0.36) and only sometimes becomes positive when considering the variability given by the standard deviation of ± 0.41. Meaning, for most of the time, prey headings are aligned towards the predator rather than away from it, which contradicts the initial assumption

that predator presence necessarily induces consistent escape alignment. A plausible explanation is that when the distance to the predator is large, as it is in most frames, the immediate attack risk remains low and prey stay in a cautious state, observing the predator from distance and therefore keeping their headings oriented towards it. Comparing the expert with the models, both stay closer to the neutral state and show no strong consistent alignment towards or away from the predator. Couzin shows low variability and slightly positive values, resulting in a weak escape-related tendency rather than the inverse. The GAIL model shows higher variability, indicating escape-related behavior in earlier phases, but also drops strongly at the end, suggesting phases where prey align towards the predator.

Summing up, both models differ compared to the expert metrics. Even if the global average distance to the prey swarm shows some consistency, nearest-prey distances reveal clear interaction-level mismatches. Escape alignment further underlines that neither model reproduces the expert avoidance behavior reliably. Comparing GAIL and the Couzin model directly, both show similarly small nearest-prey distances on average, and their escape-alignment curves largely overlap, suggesting comparable trends despite differing from the expert.

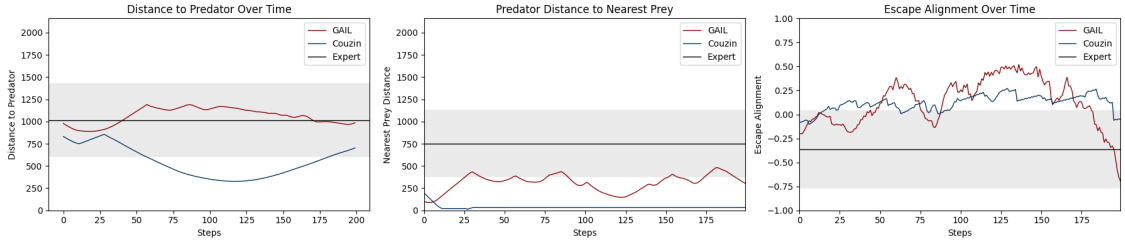


Figure 24: Model comparison on predator-related metrics.

6.2 Analysis of Policy Networks

The use of deep learning and neural networks is often perceived as working with black boxes. The desired outcome can be observed, but it is difficult to gain insights into how decisions are made, especially as networks get deeper. The modular structure of the policy networks allows further explanation and a deeper look into the decision process (Heras et al., 2019; Yapei Wu et al., 2024). For this purpose, the input-output mappings of the PIN and AN are analyzed separately. While Yapei Wu et al. (2024) analyzed policy networks trained on Couzin-simulated data, this analysis provides insights based on real fish recordings. The PIN map provides insights into how the focal's action is influenced by the neighbor's position, showing the focal response to neighbors at different locations. The AN map shows the mapping between neighbor positions and the attention weights. The maps are created by evaluating PIN and AN on a grid of relative neighbor positions and averaging the

outputs over uniformly sampled relative velocity directions. The resulting maps are visualized over the 2D position space. The colors indicate the magnitude of the respective outputs: for PIN, they reflect directional changes of the focal agent by updating the heading angle $[-180°, 180°]$, and for AN, they reflect the neighbor weights ranging from low importance (0) to high importance (1).

Interpreting the prey's PIN map, some expected behavioral tendencies can be observed (Figure 25). If the nearest neighbor is positioned closely to the right of the focal, it steers toward the right neighbor, and similarly for left-positioned neighbors. These nearby regions receive stronger weights, with the left weighting being stronger than the right. In addition, the region behind the prey on the left is also heavily involved in the action aggregation, which leads to strong left steering. If neighbors are positioned directly behind the focal, it keeps steering relatively straight. However, these regions still receive high attention, indicating that they influence the decision primarily through weighting rather than inducing strong actions. When conspecifics are further away, the behavior becomes more distorted and extreme. In these far-field regions, especially in the front-view left and right, it matters less where the neighbor is positioned, as both can lead to strong left responses. However, as indicated by the AN map, most of these extreme responses are typically low-weighted and therefore have less influence on the final aggregated action.

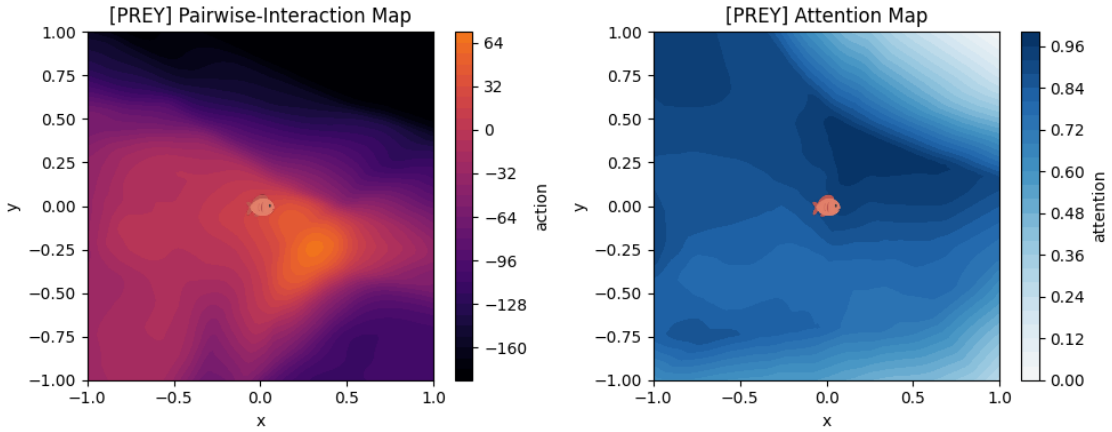


Figure 25: Prey PIN and AN maps over relative neighbor positions.

Interpreting both maps of the predator, similar tendencies can be observed (Figure 26). Close left located prey lead to left steering towards it, whereas right located prey lead to a right steering. From the predator perspective, most relevance is given to prey located in front, showing clear and relatively unnoisy attention pattern, while prey behind receives noticeably less attention. Again, in regions of low attention more extreme behavior is indicated by the PIN network, suggesting that these far behind positioned prey can produce strong steering outputs but typically have less influence on the final aggregated action. Overall, the region with the highest attention aligns with the intuition that the predator primarily focusses on his front view,

supporting a forward-oriented motion towards prey rather than reacting strongly to what is located in its back.

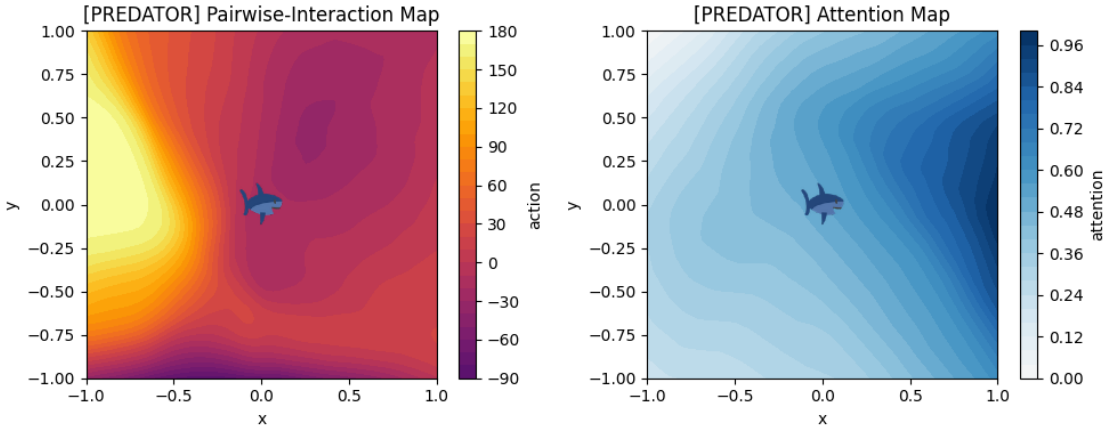


Figure 26: Predator PIN and AN maps over relative neighbor positions.

Furthermore, the policy networks allow to focus on the prey–predator relationship only (Figure 27). While the AN map shows reasonable results, with the prey focusing most strongly on what is directly in its front view, the PIN map appears strongly biased. Showing nearly the same strong left-turn response for most predator positions, independent of where the predator is approaching from.

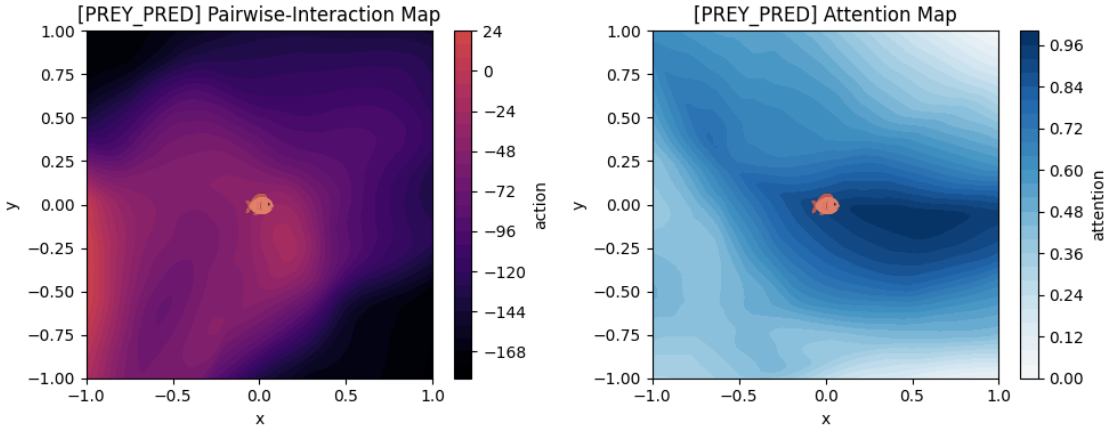


Figure 27: Prey-Predator relationship PIN and AN maps over relative neighbor positions.

Comparing the insights from the policy maps with the previous results, they seem to contradict. On the one hand, both policies are not able to successfully imitate inter-group dynamics and deviations in the swarm-metric comparisons are observed. On the other hand, the PIN and AN maps show reasonable tendencies on a single-neighbor level. Even though a lot of noise is visible in the maps, these regions are mainly linked to lower attention scores and therefore likely have less influence on the

final aggregated action. This suggests that the remaining lack of pursuit behavior is less about single-neighbor interactions, but possibly a more global aggregation problem when taking all neighbors into account over longer sequences.

6.3 Trajectory Prediction

To quantify the predictive accuracy of the learned policies, an evaluation is performed to measure how closely policy-generated rollouts match the expert trajectories when initialized from the same starting position. Even though this analysis provides a useful visual representation of the policies' decision-making, it needs to be interpreted with caution. The overall objective is distribution matching, not learning trajectories in a supervised manner. Prediction errors therefore do not necessarily indicate poor performance, as the goal is to resemble expert behavior from a global system perspective rather than matching individual trajectories. Nevertheless, the analysis is included, as it still provides additional insights into the learned decision-making.

To conduct the experiment, the first state of a given expert clip is used as the starting point. Due to the stochasticity of the policy output, 500 MC samples are obtained by repeatedly running the simulation, resulting in 500 distinct trajectories. For each timestep, prediction errors are computed and reported as mean and standard deviation across MC rollouts. Errors are computed for position offsets and heading. The position error is defined as the per-agent Euclidean distance to the corresponding expert position, reported in pixels. The heading offset refers to the absolute angular difference and is mapped to $[-180°, 180°]$. Furthermore, MC simulations allow the analysis of uncertainty in the model, expressed through the standard deviation across rollouts. The policy-generated rollouts are shown in blue, while the expert trajectory is shown in red (overlapping with the x-axis). The evaluation is performed over 10 steps to match the training window length. MC samples are collected in an environment of size 2160×2160 px with step size 1 and fixed speeds for predator and prey of 10. In a worst-case scenario, if the policy-generated trajectory moves in the exact opposite direction of the expert, the distance can increase by up to 2×10 per step, resulting in a maximum offset of 20 px per step. The maximum turn rate of the environment is set to 0.314 radians, corresponding to roughly 18° per step (Section 5.3). For comparison, Appendix E shows how fully random initialized policies behave on the same task.

Figure 28 shows the prediction errors for prey over time (Appendix E, Table 6). The position error increases approximately linearly and exhibits low uncertainty, with a standard deviation below one pixel (± 0.80 at step 9). Even though the cumulative error grows by an average increase of 7.95 pixels per step, which is far below the worst-case maximum of 20 px per step, it is only slightly better than the

random policies with a mean increase of 8.09 (Appendix E, Table 8 and Figure 40). This indicates that, when interpreting the position error only within a fixed window length of 10, the learned policy is close to a random initialization. The gap to the random baseline becomes even larger when looking at the heading error. While the θ-error increases on average by 9.70° per step, its uncertainty increases heavily after the second timestep, rising from a standard deviation of ±1.09° at step 2 to ±29.83° at step 9. This strong turning behavior can also be observed in Figure 30, where the trajectories scatter widely after a short time. Such extreme turning behavior can not be observed for the random policies, where the average accumulation is only 1.30° per step.

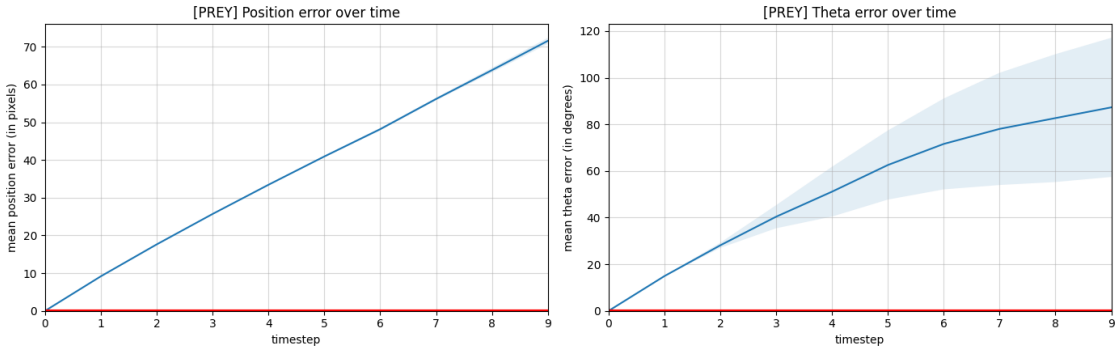


Figure 28: Prey position and heading error over time.

For the predator (Figure 29), the position error accumulation behaves similar to prey (7.67 *px* per step), with a slightly increasing uncertainty after step 4 (Appendix E, Table 7). Again, the predator policy has a slightly lower position error compared to the random policy (7.89 *px* per step) (Appendix E, Table 9 and Figure 41). The θ-error has overall higher uncertainty, which stays relatively stable in the mid range (steps 3–6) and then increases further towards the end. However, the trained policies again exhibit extreme steering motion, resulting in high heading errors and increasing uncertainty after a short period of time (Figure 30). In contrast, the heading difference of the random policies stays comparatively small, with an average accumulation of only 0.32° per step, even though the standard deviation reaches ±2.15° at step 9.

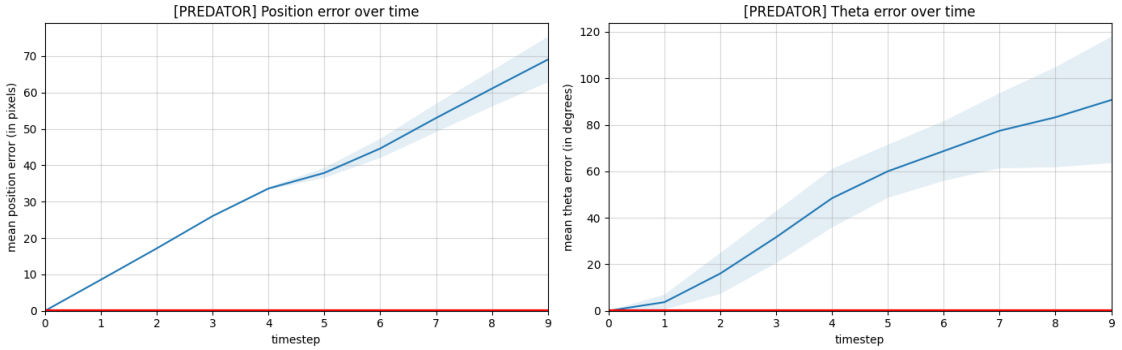


Figure 29: Predator position and heading error over time.

In addition to the prediction errors over all agents, the trajectory distributions for individuals can be analyzed by plotting all MC rollouts (gray) together with the expert trajectory (red). To make the trajectories comparable across different initial headings, trajectories are rotated such that the initial heading points upwards from the starting point (Figure 30). As described before, both policies exhibit extreme heading changes. While in the prey case the policy-generated rollouts sometimes hit the direction of the expert trajectory, the predator policy does so only rarely. The random policy shows less extreme manoeuvres (Appendix E, Figure 42).

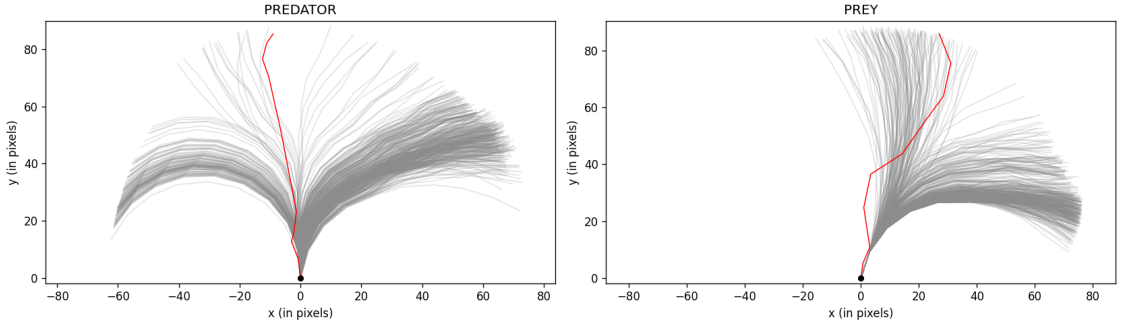


Figure 30: Expert vs. policy-generated rollouts from the same initial state.

In summary, the trajectory prediction analysis provides a complementary view of the learned policies when rolling out from identical initial states. While position offsets remain comparable to the random baseline, larger deviations are primarily observed in the heading component, reflecting strong steering variability across rollouts. Importantly, these deviations do not directly translate to overall imitation quality, since the training objective targets distribution matching at the system level rather than supervised trajectory reproduction.

6.4 Analysis of Swarm Leadership

To the current standpoint, collective behavior is primarily highlighted from a physics- and dynamics-related view, trying to quantify behavioral responses through trajectory analysis or swarm metrics. But swarms also represent a social system, where individuals obtain different roles, show group functioning, and reflect underlying internal and external influences. For example, Jolles et al. (2017) study schooling fish in a social-science framework by linking individual differences to social roles in groups. They report that individuals with a lower social proximity tendency swim faster, and are typically found more at the front and at the periphery of the group. Moreover, these individuals show higher influence in leader–follower dynamics, as directional changes propagate through the school, influencing the collective motion. In addition, they highlight that such patterns can emerge naturally in simulations of self-organized groups. Múgica et al. (2022) describe how this collective turning spreads through the group as behavioral cascades ("avalanches"), where a small subset of fish can initiate large group-level events. In their leadership analyses, they define a leadership probability based on how often an individual is already active in the first frame of an avalanche, relative to all avalanches the individual participates in. In this work, the identification of leadership in the swarm composition can be approached using the AN of the modular policies as an influence indicator. Similar to the maps in Section 6.2, neighbor importance can be analyzed based on the input–output mappings, allowing the computation of importance scores for each individual. For this, all incoming attention weights of each focal fish are summed into one score, similar to an importance voting of each fish based on all conspecifics. For visualization, the aggregated incoming weights are additionally power-scaled and re-normalized to reduce weight imbalance and obtain a clearer contrast in the transparency encoding. The weights are indicated by the transparency of each fish, and the leader is additionally marked.

Interpreting the GAIL attention graph using the connections of all fish, there is no clear order observable in this single frame (Figure 31). Importance appears rather scattered across agents, without a dominant local structure. Furthermore, the fish itself are rather scattered in space, with no obvious global alignment or density. The leader fish in the top right corner of the swarm shows a heading in a different direction compared to the preys around. But as the cascading information flow suggests, heading changes of conspecifics can occur with a time delay and are therefore not necessarily directly observable within one frame. The same pattern is observed when focusing on the predator's attention weights only. Instead of mainly attending the closest prey in this frame, the most important fish is comparatively far away, while the closest fish receives only low attention.

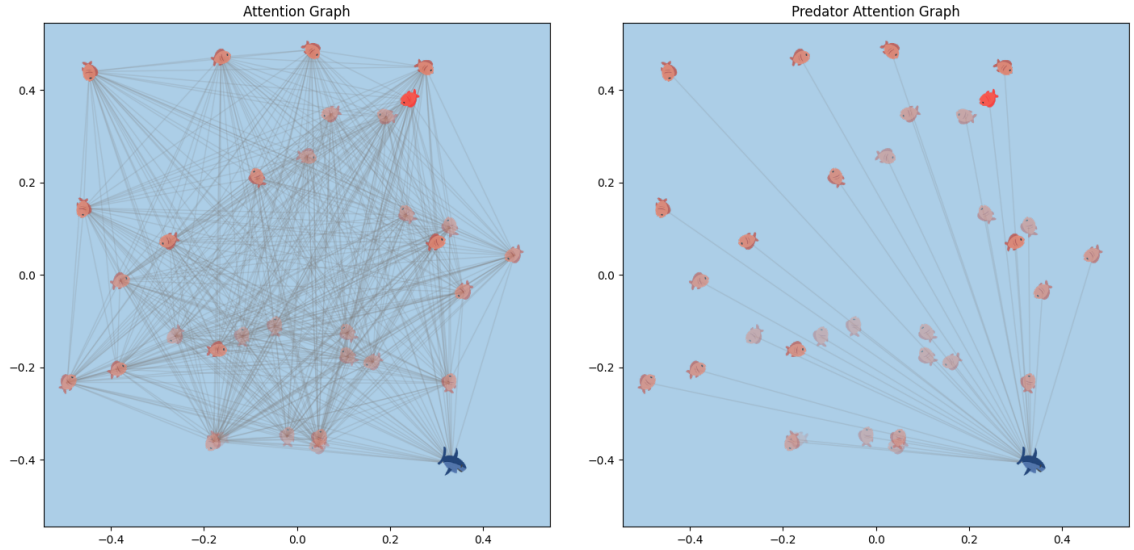


Figure 31: GAIL attention graphs. Transparency indicates attention-based importance.

Accidentally, the same analysis was also done for the pretrained policies using BC. Although the results are meant to refer exclusively to GAIL, these BC results are still reported to avoid hiding relevant observations. After pretraining with BC, the policies AN already shows a decent attention distribution, where attention is more concentrated on one side of the swarm rather than randomly scattered. In this frame, prey close to the predator receive most of the importance from the group (Figure 32). Similarly, the predator attention graph shows reasonable behavior where closer prey receive higher attention, with the predator currently steering towards the most important prey. The results also provide further insights into the training of the Video GAIL (Figure 21), as it is the only model where the policies are not randomly initialized but pretrained with BC. Directly after starting the training, both MMD and Sinkhorn distance are relatively close to the expert baseline, indicating that BC pretraining already learned sufficient behavior. This initial direction is then partially destroyed by the GAIL updates, and the model needs nearly 2000 generations to return to a comparable state. However, as suggested by the GAIL attention graph, MMD and Sinkhorn distance can indicate comparable results, while the AN functionality appears to be more destroyed than improved.

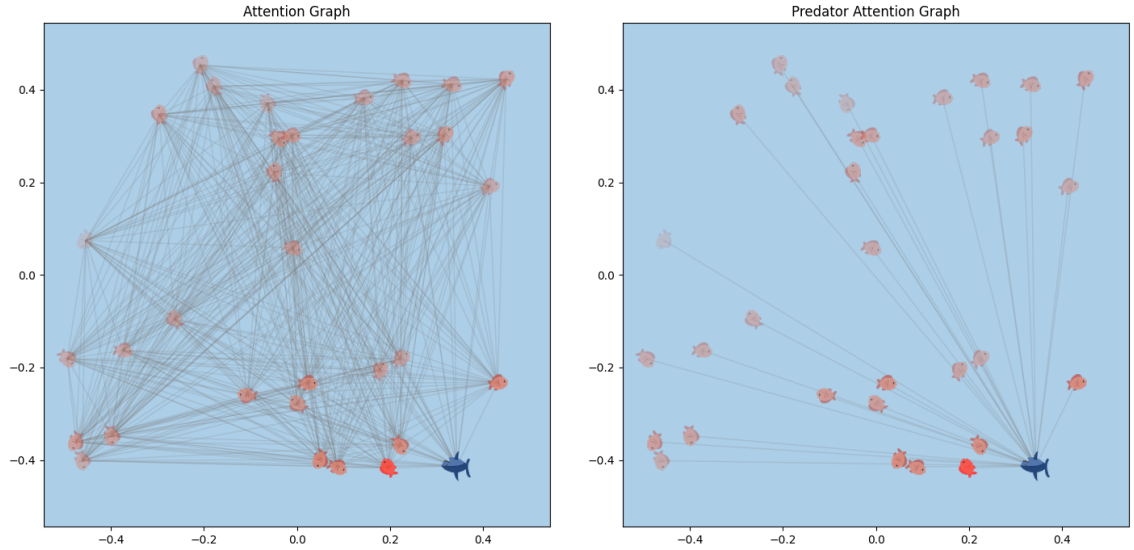


Figure 32: BC attention graphs. Transparency indicates attention-based importance.

For further investigation and to better capture potential cascading effects over time, a video of the attention graph over consecutive frames was created for the BC-pretrained policies. During the recording, the most important prey switches continuously across frames. Most of the time, this individual is located at the outer edge of the swarm, which is consistent with findings by Jolles et al. (2017) that individuals in more front positions can exert higher influence in leader–follower dynamics. Across frames, nearby conspecifics often show aligned headings. Furthermore, directional changes of the currently most important individual is often followed by changes in neighboring fish, which follows the idea of turning cascades reported in Múgica et al. (2022). However, similar to the visual interpretation of the GAIL models, the BC policy still fails to capture inter-group dynamics of successful pursuit imitation, with limited attacking behavior of the predator and insufficient avoidance responses of the prey. This suggests that the imitation itself is not necessarily the main issue, but rather that the information shared in this competitive training setup is too uninformative or gets suppressed by the aggregated information from all other prey.

6.5 Summary of Experiments

In summary, the different experiments provided different views on the Video GAIL's training success. The comparison on swarm and predator-related metrics indicates remaining gaps in reproducing the expert demonstrations reliably. While the policy maps show reasonable results, especially for the predator, the one-to-one relationship provides detailed insights into how the prey–predator relationship is represented by

the policy, resulting in noisy maps with extreme steering manoeuvres. The trajectory prediction analysis highlights these steering responses, especially in comparison to a random baseline, and shows how the error quickly accumulates over time. The leadership analysis further shows that attention-based importance is scattered for the final GAIL policies, whereas BC-pretrained policies exhibit more structured attention patterns and cascade-like behavior across frames. Overall, the results consistently indicate remaining discrepancies between policy-generated and expert behavior, while still capturing several meaningful patterns, most notably in the policy heat maps. Suggesting that the policy learned the correct tendencies while fine-grained dynamics are not yet reproduced reliably.

CHAPTER 7
Discussion

The thesis follows the overall objective of imitating predator–prey interactions directly from real aquarium recordings without relying on hand-crafted behavioral rules. It addresses key challenges such as multi-agent imitation, co-evolution, and stabilizing adversarial training with stochastic policies. It contributes a complete data pipeline for learning predator–prey behavior from videos, including trajectory extraction, tensor construction, co-evolutionary GAIL training, and an extensive evaluation that covers both a physics-based and a social perspective on swarm modeling.

During model development, the same model architecture was applied to three different expert demonstrations: Couzin-generated prey-only data, Couzin-generated predator–prey data, and the real aquarium recordings (Section 5). The architecture successfully imitated the Couzin prey-only data, showing convergence and yielding visually plausible rollouts. This provides an important sanity check. The proposed architecture and training setup can learn multi-agent collective motion when the expert demonstrations are clean and the objective is single-species imitation. Extending the complexity to predator–prey dynamics revealed a more nuanced form of success. For both predator–prey models, training exhibited only short and unstable windows in which the expert baseline of the Sinkhorn distance was reached, but not for MMD. Nevertheless, the learned behavior is not random. The predator tends to steer towards individual prey, while the prey maintain basic motion that resembles the prey-only case. However, when the predator approaches or bypasses prey, this does not trigger coordinated avoidance or attack behavior.

To tackle this problem, several approaches were implemented but ultimately did not enable the model to capture the missing inter-group dynamics (Section 5.4). First, the data pipeline was treated as the primary bottleneck, with a detailed analysis of data imbalance and trajectory quality. This included the low number of predator attacks and object detection issues caused by occlusions and light reflections. A predator flag was added to explicitly mark the predator as distinct. To rule out data quality as the main limitation, training was conducted on Couzin-simulated expert data, which provides clean and gap-free trajectories with permanent predation. Second, the policy's action aggregation was modified by reinforcing the predator contribution in the overall aggregation across all prey. Multiple variants were tested,

including shaping the predator influence based on distance to the predator, predator attention weighting, and a hard-coded mixing ratio. However, these adjustments did not lead to the expected imitation. Even with an aggregation ratio where the predator's influence was close to its maximum, prey still tended to perceive the predator as not dangerous, but rather as a conspecific. Third, inspired by Yifan Wu et al. (2025), the discriminator input was changed from agent-wise discrimination to latent transition representations, aiming to provide a more global view of the system by learning denser representations of sequences without losing too much contextual information. The prey-only model was successfully trained after the implementation of the encoder, even though this did not change the addressed inter-group dynamics in both predator–prey models targeted by this adaptation. In addition, a feature-weighting analysis was conducted to emphasize features that could reflect the desired behavior, but no clear separation of transition features emerged in the high-dimensional space. Fourth and finally, reward shaping was applied by adding additional attack and avoidance signals to the discriminator reward to further nudge the model towards the desired behavior. In practice, this mainly increased training instability rather than fine-tuning motion to capture pursuit dynamics.

Even if the policies occasionally reach the expert baseline in Sinkhorn distance, this does not necessarily imply proper imitation. A similar trend is visible in the additional analyses used for training evaluation. When interpreting the swarm-level and predator-related metrics, the Video GAIL model performs poorly across all measures and, in many cases, does not even fall within the variability indicated by the expert standard deviation. While the PIN and AN maps show reasonable patterns, especially from the predator perspective, they remain noisy in regions that correspond to low attention scores. The one-to-one maps between prey and predator provide further insight into why avoidance behavior does not emerge reliably. Although the AN map appears plausible and assigns higher weight to the predator when it is directly within the prey's field of view, the corresponding PIN map is heavily distorted and largely predicts extreme left turns. Such strong steering responses are also reflected in the trajectory prediction analysis. Compared to a purely random model, the trained policies show larger accumulated heading offsets over a short horizon, while position errors accumulate at a similar rate. However, these short-horizon trajectory mainly reflect steering variability in rollouts initialized from identical states and should not be over-interpreted as a direct measure of global imitation quality. The analysis of social roles provides an additional perspective on the models' learning dynamics. After GAIL training, attention weights are largely scattered across the swarm, whereas the BC-pretrained policies already show more plausible patterns. This suggests that the expert trajectories extracted from the recordings contain a meaningful learning signal, but that the GAIL struggled to preserve it and can even degrade a good initialization provided by BC. At the same time, BC alone also fails to learn consistent pursuit patterns, indicating that imitation itself may not be the only limiting factor. Instead, the limitation may lie in the architectural framework and, in particular, in how information is shared (or not shared) in this competitive

training setup. For example, Yifan Wu et al. (2025) successfully imitated predation behavior in their GAIL framework using a multi-instance single policy shared across both fish species. This makes their setup closer to a prey-only imitation setting, allowing a single discriminator to provide a unified reward signal with access to the full system state. Future work could start from this point. In the present thesis, even though predator and prey policies are trained competitively, information is only shared through the input tensors. While the predator input contains all accessible prey information, in the prey case the predator is effectively just one agent among many neighbors, which weakens predator-related cues. Beyond the tensors, no additional information is exchanged between roles, although such information may be required to capture inter-group dynamics. For this purpose, a joint discriminator could be implemented, receiving information from both predator and prey. Inspired by Yifan Wu et al. (2025), who successfully imitated predation behavior using a single multi-instance policy and discriminator, this could be implemented in a two-stage training setup, where both policies are trained separately first (as done currently) and, in a second stage, fine-tuned using a joint discriminator.

Next to the primary limitation of missing inter-group dynamics, this study is constrained in several additional aspects. While the data processing pipeline constructs tensors that contain useful information, as indicated by BC pretraining and the policy maps, part of this information is still lost due to tracking gaps and missing detections. This directly limits the feasibility of training on longer temporal contexts beyond the fixed window length of 10. Future work should address this by improving the tracking pipeline to preserve more complete trajectories and by enabling variable time horizons during training to capture longer and more informative sequences. As discussed, predator attack scenes are extremely underrepresented in the recordings. Although the hand-labeled attack clips were annotated with the best possible accuracy, they can still introduce additional noise. Moreover, by observing the recordings, avoidance behavior seems to be mainly triggered when the predator shows suddenly fast and hectic movements, rather than by predator proximity alone. At the same time, there are also clips where the predator stays peaceful and prey swim close nearby without showing a clear avoidance response. This can further bias the input data in predator-related scenes and makes the pursuit signal harder to learn consistently from the trajectories. Another limitation is the restricted experimental setting. The model was trained only on one predator and 32 prey. The Li group also recorded aquariums with varying group sizes (1, 2, 4, 8, 16, 32, and 64 individuals). Future work could therefore investigate training on smaller schools, which may reduce tracking errors, improve trajectory completeness, and potentially increase the relative influence of predator cues due to a more balanced predator–prey ratio. Finally, the simulation currently assumes constant speed, which reduces complexity but introduces a simplification compared to the observed system. Future work should include learnable speed and acceleration to better match real trajectories. For example, Heras et al. (2019) used a similar modular policy structure but extended the feature representation by including acceleration. In addition, rethinking

the model architecture can be beneficial. The discriminator MLP is not well suited to processing longer sequences, as it does not explicitly model temporal dependencies. If longer sequences are used, the discriminator architecture should therefore be revisited, for example by incorporating recurrent neural networks (RNNs) or long short-term memory (LSTM) networks.

In addition to the technical perspective, the proposed GAIL framework can serve as a foundation for application-oriented research. For example, Yapei Wu et al. (2024) transferred their learned policies to swarm robotics, illustrating how imitation-learned interaction rules can be deployed in real systems. Research Group Li at the Max Planck Institute of Animal Behavior could adapt this notion by transferring both the GAIL and BC policies to robotic fish. Even though the current policies only partially imitate the expert behavior and mainly reproduce basic swarm motion without sustained pursuit, such a deployment could still provide useful insights for swarm robotics. Furthermore, the model architecture could be adapted to other domains to imitate emergent collective behavior, for example pedestrian dynamics in the social sciences or active-matter particle systems in physics.

CHAPTER 8
Conclusion

The thesis followed the overall objective of imitating predator-prey interactions directly from real aquarium recordings using GAIL. Instead of relying on hand-crafted rules and reward engineering, the problem is reframed as a distribution-matching task, enabling imitation directly from observations. On a methodological level, the proposed framework transforms raw recordings into structured tensors and trains separate modular policies for predator and prey co-evolutionarily against role-specific discriminators. Beyond imitation, the pipeline provides a set of evaluation tools to interpret learned behavior, including swarm-level metrics, policy maps for modular networks, MC trajectory prediction, and an attention-based analysis of prey leadership. The architecture successfully imitates prey-only collective behavior on clean Couzin-generated trajectories, indicating that the training setup successfully learns multi-agent dynamics when the objective is single-species imitation. Extending the complexity to predator-prey dynamics revealed a more nuanced outcome. Across both Couzin-simulated predator–prey data and real recordings, the learned policies show only partial imitation, as they reproduce basic motion patterns only. The predator tends to steer towards individual prey, while the prey behave similarly to the prey-only case. However, survival-driven inter-group dynamics remain largely missing, including coordinated avoidance and consistent pursuit behavior. Overall, the results suggest that the problem is not necessarily imitation itself, but rather the learning signal available in the competitive setup and how information is shared (or not shared) between roles. From the predator perspective, the input explicitly contains the full prey group, whereas from the prey perspective, the predator is effectively one agent among many neighbors, weakens predator-related cues. In addition, the discriminators are decoupled, providing role-specific information without a coupled feedback that represents a view on the whole concatenated system. Future work should start here by enabling more joint information in training. For example, this could be achieved by introducing a joint discriminator or a two-stage setup that first trains both roles separately and then fine-tunes them with a single discriminator accessing the full system state. Further, training on longer temporal contexts and improving trajectory completeness would allow the discriminator to gain more information from its inputs. Finally, extending the simulator towards learnable speed and acceleration, and leveraging recordings with varying group sizes, could reduce modeling bias and strengthen predator-prey signals. In summary, this

thesis demonstrates that adversarial imitation from real predator-prey recordings is feasible at the level of basic collective motion, while robustly capturing inter-group dynamics remains an open challenge that is tightly linked to information availability and training design.

Bibliography

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein gan.

Arora, S. and Doshi, P. (2020). A survey of inverse reinforcement learning: Challenges, methods and progress.

Bardes, A., Ponce, J., and LeCun, Y. (2021). Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *CoRR*, abs/2105.04906.

Bartashevich, P., Herbert-Read, J., Hansen, M., Dhellemmes, F., Domenici, P., Krause, J., and Romanczuk, P. (2024). Collective anti-predator escape manoeuvres through optimal attack and avoidance strategies. *Communications Biology*, 7.

Basaran, O. T. and Dressler, F. (2025). Xainomaly: Explainable and interpretable deep contractive autoencoder for o-ran traffic anomaly detection. *Computer Networks*, 261:111145.

Beer, Y. (2026). Mmd loss in pytorch. GitHub repository. Accessed: 2026-01-23.

Bengio, Y., Courville, A. C., and Vincent, P. (2012). Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538.

Bhatia, L. (2021). Demystified: Wasserstein gan with gradient penalty. Accessed 2026-02-05.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Calovi, D. S., Lopez, U., Schuhmacher, P., Chaté, H., Sire, C., and Theraulaz, G. (2015). Collective response to perturbations in a data-driven fish school model. *Journal of The Royal Society Interface*, 12(104):20141362.

Clark, A. (2015). Pillow documentation.

Couzin, I. D., Krause, J., James, R., Ruxton, G. D., and Franks, N. R. (2002). Collective memory and spatial sorting in animal groups. *Journal of Theoretical Biology*, 218(1):1–11.

da Costa-Luis, C. (2019). tqdm: A fast, extensible progress meter for python and cli. *Journal of Open Source Software*, 4:1277.

Di Marzo Serugendo, G., Foukia, N., Hassas, S., Anthony, K., Mostéfaoui, S., Rana, O., Ulieru, M., Valckenaers, P., and van Aart, C. (2003). Self-organisation: Paradigms and applications. volume 2977, pages 1–19.

European Commission (2026a). Commercial designations and scientific name of *esox lucius*. `https://fish-commercial-names.ec.europa.eu/fish-names/species/esox-lucius_de`. Accessed: 2026-02-16.

European Commission (2026b). Commercial designations and species information for *leucaspius delineatus*. `https://fish-commercial-names.ec.europa.eu/fish-names/species/leucaspius-delineatus_en`. Accessed: 2026-02-16.

Feydy, J., Séjourné, T., Vialard, F.-X., Amari, S.-i., Trouve, A., and Peyré, G. (2019). Interpolating between optimal transport and mmd using sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2681–2690.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks.

Gu, F., Guiselin, B., Bain, N., Zuriguel, I., and Bartolo, D. (2025). Emergence of collective oscillations in massive human crowds. *Nature*, 638(8049):112–119.

Gui, J., Chen, T., Zhang, J., Cao, Q., Sun, Z., Luo, H., and Tao, D. (2024). A survey on self-supervised learning: Algorithms, applications, and future trends.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. (2017). Improved training of wasserstein gans.

Gustafson, S., Arumugam, H., Kanyuk, P., and Lorenzen, M. (2016). Mure: fast agent based crowd simulation for vfx and animation. New York, NY, USA. Association for Computing Machinery.

Haeri, H. (2026a). Couzin swarm model: Predator–prey simulation script. File: swarm_pray_predator.py; accessed 2026-02-05.

Haeri, H. (2026b). couzin_swarm_model: swarm_pray_predator.py. `https://github.com/hossein-haeri/couzin_swarm_model/blob/master/swarm_pray_predator.py`. Accessed: 2026-01-25.

Hagberg, A. A., Schult, D. A., and Swart, P. J. (2008). Exploring network structure, dynamics, and function using networkx. In Varoquaux, G., Vaught, T., and Millman, J., editors, *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA, USA.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W.,

Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

Heras, F. J. H., Romero-Ferrero, F., Hinz, R. C., and de Polavieja, G. G. (2019). Deep attention networks reveal the rules of collective motion in zebrafish. *PLOS Computational Biology*, 15(9):1–23.

Ho, J. and Ermon, S. (2016). Generative adversarial imitation learning.

Hui, J. (2018). Gan — wasserstein gan & wgan-gp. `https://jonathan-hui.medium.com/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490`. Medium article.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

Ishiwaka, Y., Zeng, X., Ogawa, S., Westwater, D., Tone, T., and Nakada, M. (2022). Deepfoids: Adaptive bio-inspired fish simulation with deep reinforcement learning. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 18377–18389. Curran Associates, Inc.

Jocher, G. and Contributors (2025). Ultralytics yolo. Python package version 8.3.164.

Jolles, J. W., Boogert, N. J., Sridhar, V. H., Couzin, I. D., and Manica, A. (2017). Consistent individual differences drive collective behavior and group functioning of schooling fish. *Current Biology*, 27(18):2862–2868.e7.

Juras, E. (2025). Train-and-deploy-yolo-models. Tutorials and examples showing how to train and deploy Ultralytics YOLO models.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45.

Khanam, R. and Hussain, M. (2024). Yolov11: An overview of the key architectural enhancements.

Khromov, G. and Singh, S. P. (2024). Some fundamental aspects about lipschitz continuity of neural networks.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97.

Kumar, T., Mileo, A., Brennan, R., and Bendechache, M. (2023). Image data augmentation approaches: A comprehensive survey and future directions.

Lee, J., Won, J., and Lee, J. (2018). Crowd simulation by deep reinforcement learning. New York, NY, USA. Association for Computing Machinery.

Lee, M. and Seok, J. (2020). Regularization methods for generative adversarial networks: An overview of recent studies.

Leu, T., Lüscher, B., Zumbach, S., and Schmidt, B. R. (2009). Small fish (leucaspius delineatus) that are often released into garden ponds and amphibian breeding sites prey on eggs and tadpoles of the common frog (rana temporaria). *Amphibia-Reptilia*, 30(2):290–293.

Li, J., Li, L., and Zhao, S. (2023). Predator–prey survival pressure is sufficient to evolve swarming behaviors. *New Journal of Physics*, 25(9):092001.

Li, L. (2025). Behavioral fish trajectory dataset. Data access provided for this study.

Liu, G., Zhao, L., Yang, F., Bian, J., Qin, T., Yu, N., and Liu, T.-Y. (2019). Trust region evolution strategies. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:4352–4359.

Luise, G., Rudi, A., Pontil, M., and Ciliberto, C. (2018). Differential properties of sinkhorn approximation for learning with wasserstein distance.

Lőrincz, Z. (2019). A brief overview of imitation learning. Accessed: 2025-10-10.

Mehraban, Z. and Pichler, A. (2025). Quantization of probability measures in maximum mean discrepancy distance.

Morales-Brotons, D., Vogels, T., and Hendrikx, H. (2024). Exponential moving average of weights in deep learning: Dynamics and benefits.

Múgica, J., Torrents, J., Cristin, J., Puy, A., Miguel, M. C., and Pastor-Satorras, R. (2022). Scale-free behavioral cascades and effective leadership in schooling fish. *Scientific Reports*, 12:10783.

Naranjo-Campos, F. J., Victores, J. G., and Balaguer, C. (2024). Expert-trajectory-based features for apprenticeship learning via inverse reinforcement learning for robotic manipulation. *Applied Sciences*, 14(23).

OpenAI (2017). Evolution strategies as a scalable alternative to reinforcement learning. `https://openai.com/index/evolution-strategies/`. accessed: 2025-10-14.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Pierson, E. and Kao, P. (2025). Markov decision processes.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, page 25–34, New York, NY, USA. Association for Computing Machinery.

Ross, S., Gordon, G. J., and Bagnell, J. A. (2011). A reduction of imitation learning and structured prediction to no-regret online learning.

Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. (2017). Evolution strategies as a scalable alternative to reinforcement learning.

Schaal, S. (1996). Learning from demonstration. In *Proceedings of the 10th International Conference on Neural Information Processing Systems*, NIPS'96, page 1040–1046, Cambridge, MA, USA. MIT Press.

Schafer, T. L. J., Wikle, C. K., and Hooten, M. B. (2022). Bayesian inverse reinforcement learning for collective animal movement.

Song, J., Ren, H., Sadigh, D., and Ermon, S. (2018). Multi-agent generative adversarial imitation learning.

Sun, X., Yang, S., Zhou, M., Liu, K., and Mangharam, R. (2024). Mega-dagger: Imitation learning with multiple imperfect experts.

Team, T. P. D. (2020). pandas-dev/pandas: Pandas.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude. Lecture slides (often listed as Lecture 6e/6.5 in the course materials).

Tkachenko, M., Malyuk, M., Holmanyuk, A., and Liubimov, N. (2020-2025). Label Studio: Data labeling software. Open source software available from https://github.com/HumanSignal/label-studio.

Torabi, F., Warnell, G., and Stone, P. (2018). Behavioral cloning from observation.

Ultralytics (2026). Model comparisons and interactive performance benchmarks. `https://docs.ultralytics.com/compare/#interactive-performance-benchmarks`. Official documentation for Ultralytics YOLO model comparisons.

Vicsek, T., Czirók, A., Ben-Jacob, E., Cohen, I., and Shochet, O. (1995). Novel type of phase transition in a system of self-driven particles. *Physical Review Letters*, 75(6):1226–1229.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris,

C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

Wang, P. (2025). ema-pytorch. Version 0.7.9.

Wang, Y., Hayashibe, M., and Owaki, D. (2024). Data-driven policy learning methods from biological behavior: A systematic review. *Applied Sciences*, 14(10).

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. (2022). Emergent abilities of large language models.

Wojke, N. (2023). deep-sort-realtime. https://pypi.org/project/deep-sort-realtime/. Version 1.3.2.

Wojke, N., Bewley, A., and Paulus, D. (2017). Simple online and realtime tracking with a deep association metric.

Yapei Wu, Wang, T., Liu, T., Zheng, Z., Xu, D., and Peng, X. (2024). Adversarial imitation learning with deep attention network for swarm systems. *Complex & Intelligent Systems*, 11(1):26.

Yifan Wu, Dou, Z., Ishiwaka, Y., Ogawa, S., Lou, Y., Wang, W., Liu, L., and Komura, T. (2025). Cbil: Collective behavior imitation learning for fish from real videos. *ACM Transactions on Graphics*, 43(6):1–17.

Yong, C. H. and Miikkulainen, Y. (2001). Cooperative coevolution of multi-agent systems.

Yu, X., Wu, W., Feng, P., and Tian, Y. (2021). Swarm inverse reinforcement learning for biological systems. In *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 274–279.

Zare, M., Kebria, P. M., Khosravi, A., and Nahavandi, S. (2023). A survey of imitation learning: Algorithms, recent developments, and challenges.

Zhang, M., Wang, Y., Ma, X., Xia, L., Yang, J., Li, Z., and Li, X. (2020). Wasserstein distance guided adversarial imitation learning with reward shape exploration. In *2020 IEEE 9th Data Driven Control and Learning Systems Conference (DDCLS)*, pages 1165–1170.

Zulko and Contributors (2020). Moviepy. Version 1.0.3.

Appendix

A Data Collection and Processing

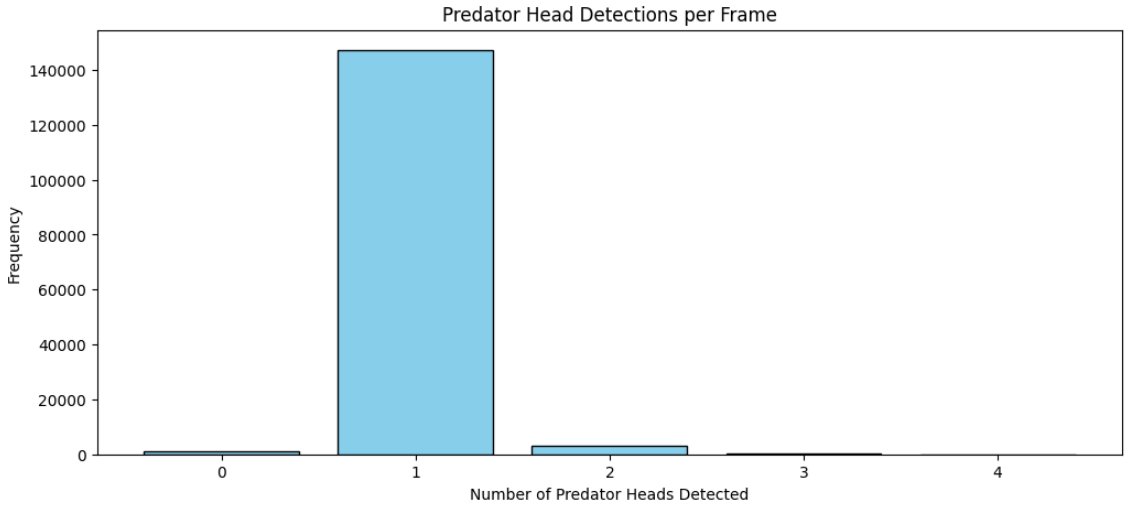


Figure 33: Distribution of predator head detections per frame.

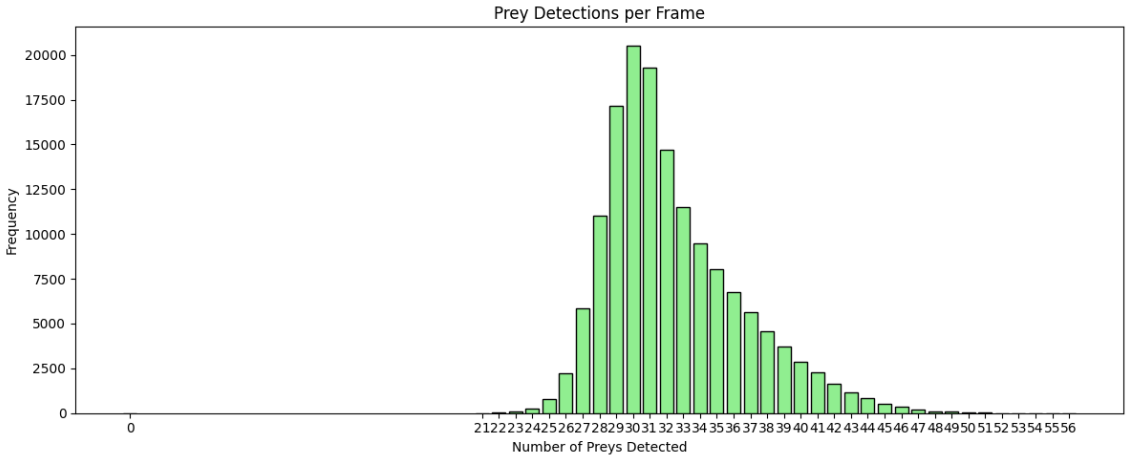


Figure 34: Distribution of prey detections per frame.

Table 1: Number of available windows for each window length.

Window length	**Total windows**
1	11981
2	3201
3	1280
4	628
5	345
6	195
7	110
8	68
9	41
10	24
11	13
12	6
13	2
14	1
15	0

B Video Environment Setup

Table 2: Heading-change quantiles in radians and degrees.

Quantile	Radians	Degrees
95%	0.123	7.406
99%	0.314	18.003
99.9%	2.095	120.040

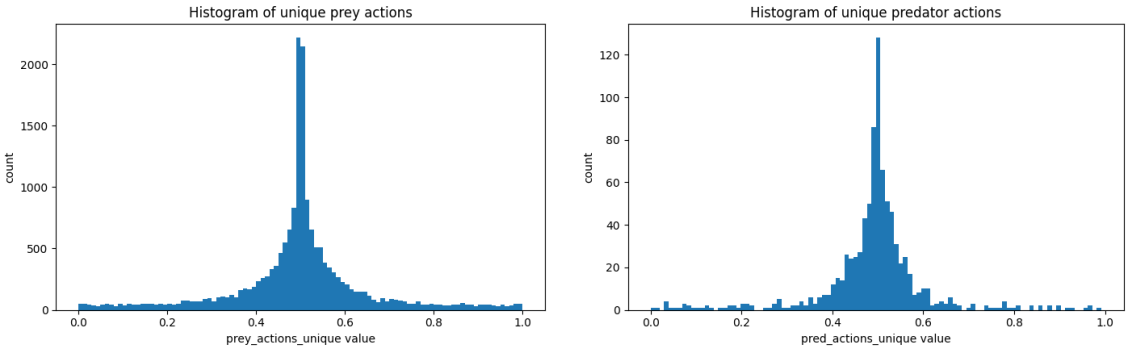


Figure 35: Histograms of unique action values for predator and prey.

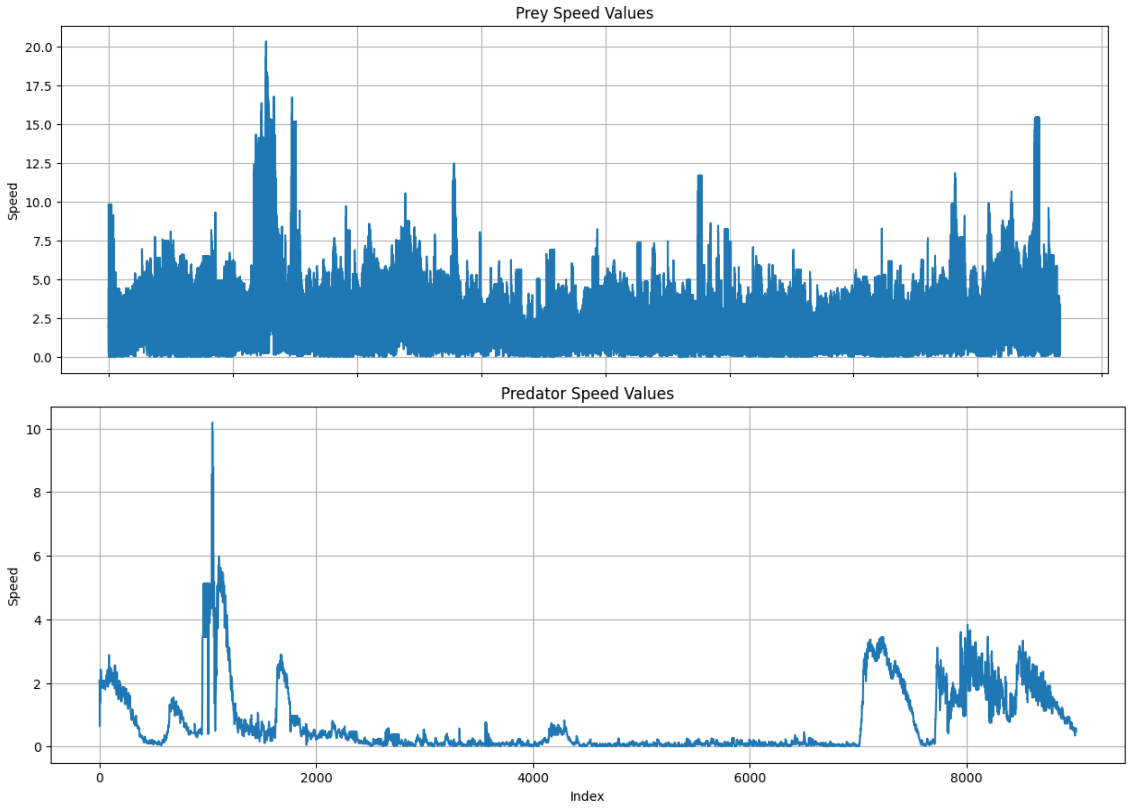


Figure 36: Maximum tracked speed values for predator and prey.

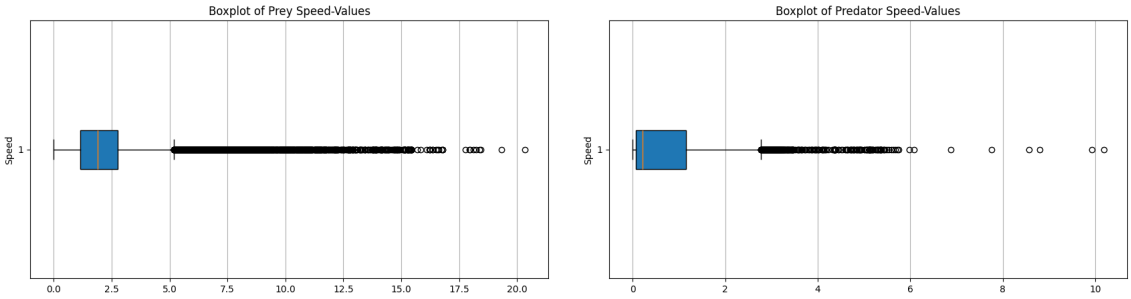


Figure 37: Boxplots of maximum tracked speed values for predator and prey.

C Hyperparameters

Table 3: Overview of hyperparameters used for the Couzin prey-only model.

Parameter	Value
Expert Data Generation	
Max steps per episode (`max_steps`)	500
Training Setup	
Number of generations (`num_generations`)	3000
Decay factor (`gamma`)	0.999
Pretraining (`pretrain`)	False
Performance evaluation interval (`performance_eval`)	5
Policy	
Policy learning rate (`lr_policy`)	5×10^{-4}
Number of perturbations (`num_perturbations`)	64
Exploration noise (`sigma`)	0.1
Discriminator	
Discriminator-policy ratio (`dis_balance_factor`)	2
Input noise (`noise`)	0.005
Discriminator learning rate (`lr_disc`)	5×10^{-4}
Gradient penalty (`lambda_gp`)	5
Update mode (`prey_update_mode`)	mean
Environment Settings	
Arena height (`height`)	50
Arena width (`width`)	50
Prey speed (`prey_speed`)	5
Predator speed (`pred_speed`)	5
Step size (`step_size`)	0.5
Max turn per step (`max_turn`)	0.25
Rollout Configuration	
Perturbation steps (`pert_steps`)	100
Initial steps (`init_steps`)	500

Table 4: Overview of hyperparameters used for the Couzin predator-prey model.

Parameter	Value
Expert Data Generation	
Max steps per episode (`max_steps`)	500
Training Setup	
Number of generations (`num_generations`)	3000
Decay factor (`gamma`)	0.999
Pretraining (`pretrain`)	False
Performance evaluation interval (`performance_eval`)	5
Prey Policy	
Prey policy learning rate (`lr_prey_policy`)	2×10^{-4}
Number of perturbations (`num_perturbations`)	64
Prey exploration noise (`sigma_prey`)	0.1
Prey Discriminator	
Prey discriminator-policy ratio (`prey_dis_balance_factor`)	2
Prey input noise (`prey_noise`)	0.005
Prey discriminator learning rate (`lr_prey_disc`)	5×10^{-4}
Prey gradient penalty (`lambda_gp_prey`)	5
Prey update mode (`prey_update_mode`)	mean ($\lambda = None$)
Predator Policy	
Predator policy learning rate (`lr_pred_policy`)	1×10^{-4}
Predator exploration noise (`sigma_pred`)	0.08
Predator Discriminator	
Predator discriminator-policy ratio (`pred_dis_balance_factor`)	2
Predator input noise (`pred_noise`)	0.005
Predator discriminator learning rate (`lr_pred_disc`)	2×10^{-4}
Predator gradient penalty (`lambda_gp_pred`)	10
Predator update mode (`pred_update_mode`)	mean ($\lambda = None$)
Environment Settings	
Arena height (`height`)	50
Arena width (`width`)	50
Prey speed (`prey_speed`)	5
Predator speed (`pred_speed`)	5
Step size (`step_size`)	0.5
Max turn per step (`max_turn`)	0.25
Rollout Configuration	
Perturbation steps (`pert_steps`)	100
Initial steps (`init_steps`)	500

Table 5: Overview of hyperparameters used for the video predator-prey model.

Parameter	Value
Training Setup	
Number of generations (`num_generations`)	4000
Decay factor (`gamma`)	0.999
Pretraining (`pretrain`)	True
Performance evaluation interval (`performance_eval`)	5
Prey Policy	
Prey policy learning rate (`lr_prey_policy`)	2×10^{-4}
Number of perturbations (`num_perturbations`)	64
Prey exploration noise (`sigma_prey`)	0.1
Prey Discriminator	
Prey discriminator-policy ratio (`prey_dis_balance_factor`)	2
Prey input noise (`prey_noise`)	0.005
Prey discriminator learning rate (`lr_prey_disc`)	5×10^{-4}
Prey gradient penalty (`lambda_gp_prey`)	5
Prey update mode (`prey_update_mode`)	mean ($\lambda =$ `None`)
Predator Policy	
Predator policy learning rate (`lr_pred_policy`)	1×10^{-4}
Predator exploration noise (`sigma_pred`)	0.08
Predator Discriminator	
Predator discriminator-policy ratio (`pred_dis_balance_factor`)	2
Predator input noise (`pred_noise`)	0.005
Predator discriminator learning rate (`lr_pred_disc`)	1×10^{-4}
Predator gradient penalty (`lambda_gp_pred`)	10
Predator update mode (`pred_update_mode`)	mean ($\lambda =$ `None`)
Environment Settings	
Arena height (`height`)	2160
Arena width (`width`)	2160
Prey speed (`prey_speed`)	10
Predator speed (`pred_speed`)	10
Step size (`step_size`)	1.0
Max turn per step (`max_turn`)	0.314
Rollout Configuration	
Perturbation steps (`pert_steps`)	100
Initial steps (`init_steps`)	500

D Feature Clustering & Weighting

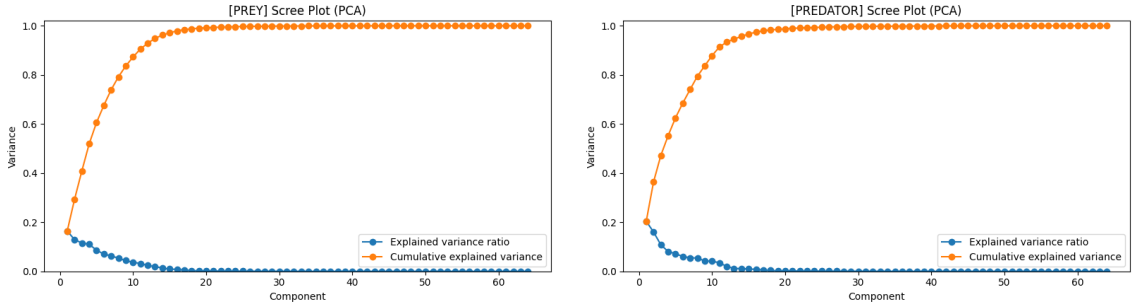


Figure 38: PCA scree plots (explained variance by component).

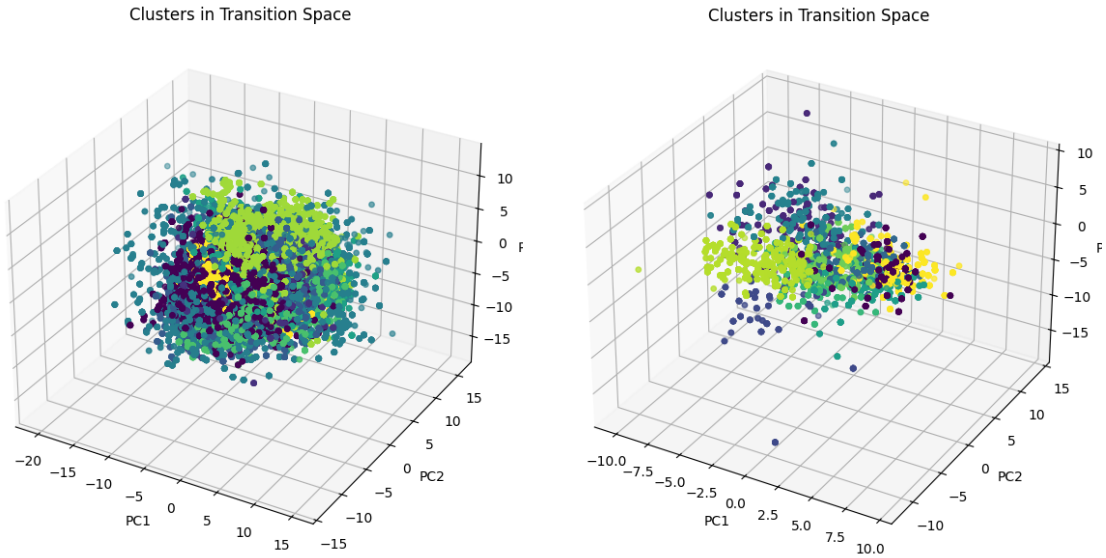


Figure 39: PCA transition cluster visualizations for prey (left) and predator (right).

E Experiments

Table 6: Prey trajectory prediction errors over time (mean ± std across MC rollouts).

Step	Position Error (in pixel)	Theta Error (in degree)
0	0.0000 ± 0.0000	0.00 ± 0.00
1	9.1741 ± 0.0005	14.92 ± 0.15
2	17.6304 ± 0.0068	28.13 ± 1.09
3	25.6891 ± 0.0534	40.41 ± 4.95
4	33.4266 ± 0.1627	51.14 ± 10.67
5	40.9260 ± 0.2093	62.58 ± 14.81
6	48.1575 ± 0.3265	71.62 ± 19.48
7	56.1581 ± 0.3535	78.08 ± 24.06
8	63.7996 ± 0.6015	82.71 ± 27.42
9	71.5674 ± 0.8040	87.31 ± 29.83

Table 7: Predator trajectory prediction errors over time (mean ± std across MC rollouts).

Step	Position Error (in pixel)	Theta Error (in degree)
0	0.0000 ± 0.0000	0.00 ± 0.00
1	8.5368 ± 0.0232	3.70 ± 3.15
2	17.1490 ± 0.1011	16.03 ± 8.79
3	26.0300 ± 0.1584	31.65 ± 11.12
4	33.6087 ± 0.3400	48.41 ± 12.56
5	37.8968 ± 1.3586	60.02 ± 11.42
6	44.6024 ± 2.6221	68.71 ± 12.80
7	52.9816 ± 3.8686	77.44 ± 16.12
8	61.0695 ± 4.9021	83.23 ± 21.49
9	69.0401 ± 6.2269	90.69 ± 27.12

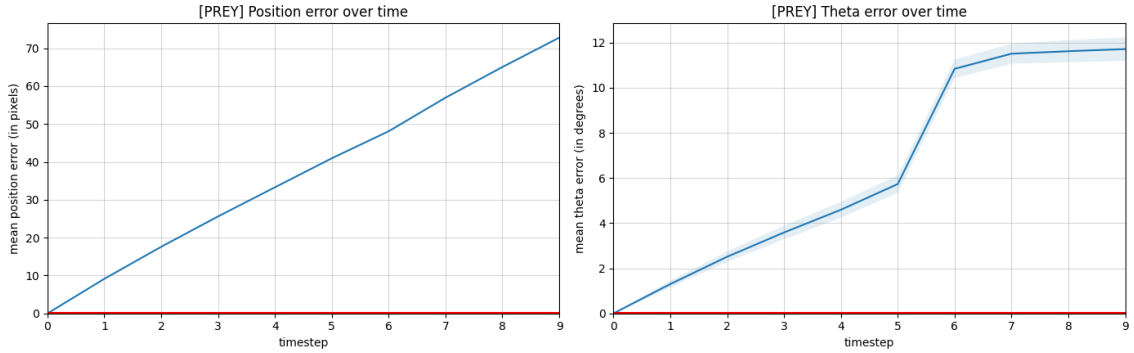


Figure 40: Prey position and heading error over time for a random policy.

Table 8: Random prey trajectory prediction errors over time (mean ± std across MC rollouts).

Step	Position Error (in pixel)	Theta Error (in degree)
0	0.0000 ± 0.0000	0.00 ± 0.00
1	9.1397 ± 0.0012	1.30 ± 0.15
2	17.5980 ± 0.0036	2.51 ± 0.23
3	25.6122 ± 0.0074	3.59 ± 0.31
4	33.2997 ± 0.0140	4.59 ± 0.35
5	40.9574 ± 0.0198	5.73 ± 0.39
6	48.0955 ± 0.0509	10.84 ± 0.40
7	56.9625 ± 0.0703	11.51 ± 0.44
8	65.0515 ± 0.0874	11.62 ± 0.49
9	72.8116 ± 0.1001	11.71 ± 0.52

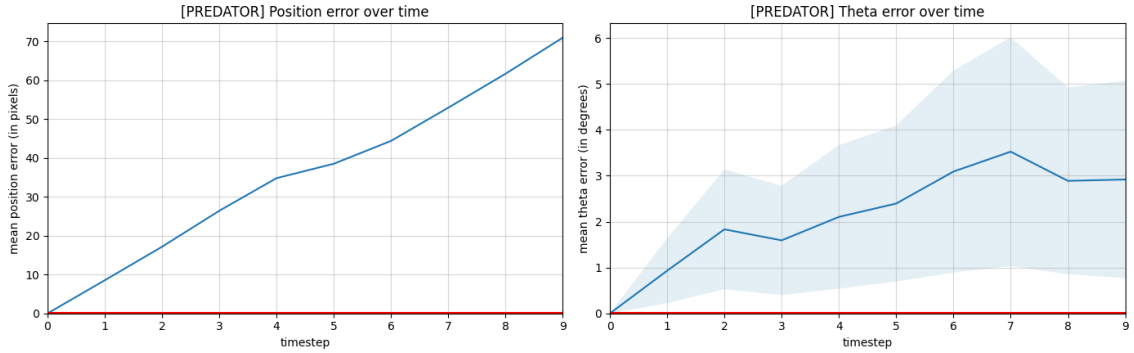


Figure 41: Predator position and heading error over time for a random policy.

Table 9: Random predator trajectory prediction errors over time (mean ± std across MC rollouts).

Step	Position Error (in pixel)	Theta Error (in degree)
0	0.0000 ± 0.0000	0.00 ± 0.00
1	8.5279 ± 0.0036	0.93 ± 0.70
2	17.1532 ± 0.0169	1.83 ± 1.30
3	26.4107 ± 0.0136	1.59 ± 1.19
4	34.8087 ± 0.0111	2.10 ± 1.56
5	38.5092 ± 0.0520	2.39 ± 1.70
6	44.3931 ± 0.0901	3.09 ± 2.20
7	52.9417 ± 0.1138	3.52 ± 2.49
8	61.6769 ± 0.1053	2.89 ± 2.03
9	70.9685 ± 0.0917	2.92 ± 2.15

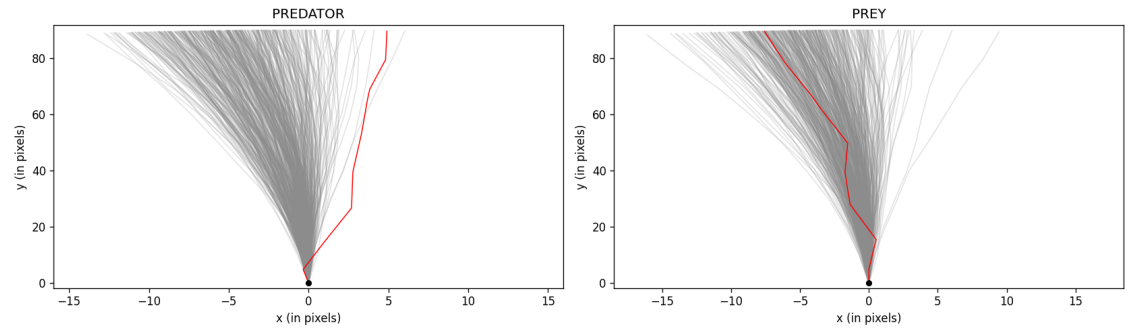


Figure 42: Expert vs. random policy-generated rollouts from the same initial state.

F Software Usage

Table 10: Python libraries used in this thesis.

Package	Version	Reference
`deep-sort-realtime`	1.3.2	(Wojke, 2023)
`ema-pytorch`	0.7.9	(Wang, 2025)
`geomloss`	0.2.6	(Feydy et al., 2019)
`matplotlib`	3.10.3	(Hunter, 2007)
`moviepy`	1.0.3	(Zulko and Contributors, 2020)
`networkx`	3.4.2	(Hagberg et al., 2008)
`numpy`	1.26.4	(Harris et al., 2020)
`opencv-python`	4.12.0.88	(Bradski, 2000)
`pandas`	2.3.1	(Team, 2020)
`pillow`	11.3.0	(Clark, 2015)
`scikit-learn`	1.7.1	(Pedregosa et al., 2011)
`scipy`	1.15.3	(Virtanen et al., 2020)
`torch`	2.6.0+cu124	(Paszke et al., 2019)
`tqdm`	4.67.1	(da Costa-Luis, 2019)
`ultralytics`	8.3.164	(Jocher and Contributors, 2025)