ME5424 Course Project Report: Phase-Based Reinforcement Learning for the Eagle–Hen Chain Adversarial Task

Contributors: Project Team

Advisor: —

Affiliation: —

Submission Date: 2025 年 11 月 21 日

Table of Contents

Abstract

This report addresses the cold-start and convergence challenges of multi-agent adversarial reinforcement learning in physics-rich environments. We propose a curriculum-based, phase-wise training scheme: Stage 1 trains hens against a heuristic eagle, and Stage 2 freezes the hen policy while training the eagle. Under Box2D physics and Gymnasium interfaces, we adopt Stable-Baselines3 PPO with parallel sampling and TensorBoard logging to build a reproducible engineering pipeline. Our contributions include environment modeling (world, chain and joints, boundary and bounce), observation and reward design, parallel training with evaluation callbacks, and behavior visualization with metric analysis. Results show improved hen defense in Stage 1 (mean return increases, explained variance approaches 1, entropy loss decreases) and effective eagle approach/capture behavior in Stage 2 (positive potential-based rewards, sparse positive returns upon capture), with overall smooth training curves. Limitations include reward weighting and physical parameters requiring empirical tuning. Future work will consider joint fine-tuning, self-play, and extensions to more complex scenarios.

Keywords: Multi-agent adversarial learning; Curriculum learning; PPO; Gymnasium; Box2D

1 Introduction

This project focuses on multi-agent adversarial reinforcement learning under realistic physical constraints. We construct a two-dimensional task where hens guard a chain of chicks while an eagle attacks the tail, to assess stable learning under sparse objectives and strongly coupled dynamics. Early weak policies and sparse returns impede adversarial shaping; chain actions and inertia propagate via distance joints causing delay and oscillation, and boundary constraints with bounce further increase optimization difficulty. Direct joint training tends to oscillate or prematurely converge. We adopt a phase-based curriculum: Stage 1 fixes the eagle to a heuristic tail-chasing policy and trains hens to learn blocking and guarding; Stage 2 freezes the hen policy and trains the eagle to learn approach and capture against a non-trivial opponent. Implemented with Gymnasium and Box2D, and combined with viewpoint-specific observation normalization, potential-based reward shaping, and unified boundary/bounce modeling, this scheme mitigates distribution instability and exploration failures.

Regarding roles, the eagle forms a local state each step from positions of uncaptured chicks and observations of guardians/teammates, with rewards emphasizing rapid capture, boundary adherence, and penalizing repetitive circling, including penalties when

repelled by hens. Hens maintain observations around a safety radius, prioritize protection of chicks within that radius, and gain higher rewards by blocking along the eagle–tail line; any capture penalizes the group to drive tight defense. Chicks continually sense the nearest hen and eagle; their policy favors moving toward protectors and away from threats, with rewards encouraging staying within the guarding radius and penalizing capture, yielding coordinated evasion. This tri-party design aligns objectives, physical constraints, and interpretable rewards, facilitating analysis and evaluation. Our goal is to validate phase-based training for sparse-reward MARL using realistic chain dynamics and interpretable rewards, and to deliver a reproducible, evaluable pipeline with teaching value. Behaviorally, hens remain centered under chain inertia, defend the tail, and form an effective blocking line between the eagle and tail; the eagle learns side/rear detours and accelerated approach. Metric-wise, Stage 1 shows steady mean return growth, explained variance near 1, and decreasing entropy loss; Stage 2 shows increased capture success, positive approach potentials, and smooth overall loss curves. We emphasize repeatability and verifiability via fixed seeds, standardized evaluation callbacks, and visualizations, providing instructional and research value.

2 Literature Review and Research Status

2.1 Current Status and Challenges

Multi-agent reinforcement learning (MARL) for adversarial tasks often uses joint training or self-play. In sparse-reward, physics-coupled scenarios, training can oscillate, advantage estimates exhibit high variance, and mutual adaptation induces distribution shift. Mainstream approaches include:

- Using PPO, SAC and other on-/off-policy methods with parallel sampling and entropy regularization to mitigate exploration deficiency [4, 5]
- Employing curriculum learning to gradually increase task difficulty [6]
- Introducing potential-based rewards, shaping, or imitation learning to stabilize training [1]

Limitations:

- Early joint training yields ineffective adversaries with near-zero returns and noisy gradients
- Single-view observations lack consistent normalization in each agent's body frame

- Physics collisions and boundary constraints are not properly encoded in rewards, leading to pathological behaviors such as boundary stalling

2.2 Multi-Agent RL and Self-Play

Adversarial multi-agent tasks are common in robotics cooperation and game-theoretic environments. Self-play and opponent modeling are typical routes to improved robustness, but early training often suffers from sparse rewards and unstable distributions. Mutual adaptation causes distribution shift, increasing optimization difficulty.

A systematic review summarizes challenges and progress in communication, cooperation, and adversarial MARL, highlighting non-stationarity, credit assignment, and insufficient exploration as key bottlenecks requiring stable algorithms and principled task design [1]. In communication, differentiable end-to-end mechanisms can improve cooperation/adversarial performance under partial observability but introduce optimization instabilities, which must be controlled via regularization and curriculum strategies [2].

2.3 Proximal Policy Optimization (PPO)

PPO constrains policy update magnitude via a clipped objective, combines advantage estimation and entropy regularization, and balances stability with sample efficiency, making it a mainstream baseline for continuous control. We use SB3 with parallel environments and large batches to reduce temporal correlation, along with evaluation callbacks and TensorBoard monitoring.

Compared to other continuous control methods, DDPG trains deterministically with target networks, suiting low-dimensional action spaces but being sensitive to exploration and hyperparameters [3]; SAC, based on maximum entropy, encourages balanced exploration and often exhibits improved robustness in complex tasks [4]. PPO's simple updates, mature implementations, and relative insensitivity to hyperparameters make it a preferred choice in engineering contexts, with reliable open-source support for reproducibility [5].

2.4 Curriculum Learning

Curriculum learning alleviates cold-start and exploration challenges by progressively increasing task difficulty. Here, Stage 1 fixes the eagle to a heuristic policy so hens can learn blocking/defense against a controlled opponent; Stage 2 freezes the hen policy and trains the eagle to learn approach/capture against a non-trivial opponent. This avoids near-zero gradients from ineffective early adversaries.

Classically, curriculum learning organizes samples/objectives in a sequence from easy to hard, improving generalization and stability [6]. In adversarial MARL, phase-wise or semi-static training alleviates oscillation and maladaptation from simultaneous bilateral learning without altering the final task, and has shown effectiveness in group adversarial settings [7].

2.5 Reward Shaping and Potential Functions

Reward shaping introduces potential functions or geometric scores to convert sparse objectives into dense, optimizable signals. We use: (1) a geometric blocking score measuring whether hens effectively position between the eagle and tail; (2) an approach potential capturing reductions in eagle–tail distance; and (3) stretch, boundary, and bounce terms to enforce physical plausibility and encourage active exploration.

Under the maximum entropy framework, entropy regularization encourages diverse policies and maintains sufficient exploration near local optima; combined with potential-based rewards, it can further improve stability and sample efficiency [4]. In MARL, rewards depend on both individual and team states, so shaping must align with physical constraints to avoid unreasonable strategic biases [1].

2.6 Physics Simulation and Interpretability

Box2D distance joints and rigid-body collisions provide interpretable dynamics for the chain and agents. Unified boundary constraints and bounce rules yield learned strategies that match intuition in visualizations (e.g., hens spreading “wings”to form a blocking band, eagle rear detours and accelerated approach), enhancing physical consistency and instructional readability.

For standardized evaluation in control tasks, the DeepMind Control Suite provides reproducible task sets and metrics, forming a basis for comparison and replication [8]. We follow similar reproducibility principles: fixed seeds, standardized logging and evaluation callbacks, and unified physical parameters and interface definitions [5].

2.7 Parallel Sampling and Stable Training

Parallel environments (`SubprocVecEnv` with `n_envs`=16) reduce single-environment temporal correlation and increase sampling throughput. Environments step asynchronously with batched rollouts aggregated to global updates; fixed seeds ensure reproducibility. An `EvalCallback` runs at a fixed frequency (e.g., every K updates), saving the best-performing checkpoints to enable phase-wise review and consistent comparisons.

Engineering practice shows that stable implementations, clear logging and evaluation protocols, and consistency with community baselines are critical to reproducible experiments [5]. We align interfaces and settings with mainstream baselines to minimize bias from implementation differences.

3 Project Design and Implementation

3.1 Requirements Analysis

The system must support stable adversarial interactions within a 2D world boundary:

- Hens defend the tail and maintain stable chain motion
- The eagle efficiently approaches and captures within a limited number of steps
- Observations are normalized in each agent's body frame
- Rewards balance goal attainment with physical plausibility
- Training supports parallel sampling and reproducibility, with evaluation and visualization

3.2 Solution Design (Technical Approach and Rationale)

Our pipeline centers on reproducibility: two derived training stages on a unified physical environment, parallel sampling and evaluation callbacks to ensure sampling independence and observation stability, and consistent designs for observation normalization, reward shaping, and boundary bounce to avoid redundant theory and unstable implementations. Specific algorithmic and simulation choices are detailed in subsequent subsections.

3.3 Implementation (Workflow and Key Steps)

Stage 1

- Initialize world and chain
- Eagle uses a heuristic tail-chasing strategy
- Hens learn defense and position control with PPO
- Evaluation callback saves the best model

Stage 2

- Load and freeze the hen policy
- Train the eagle with PPO to learn approach and capture
- Use the evaluation callback to save the best model
- Provide visualization scripts for behavior inspection

3.4 Physical Environment Implementation

Physical parameters are managed via a unified configuration, including `world_size`, `dt`, `max_steps`, `chain_links`, `chain_spacing`, `hen_max_speed`, `eagle_max_speed`, `catch_radius`, and `block_margin`. World construction and chain/joint setup use dedicated builders, and forces with velocity clipping are applied during action to maintain simulation stability.

3.5 Observation and Action Spaces

Observations are normalized in each agent's body frame; actions are continuous 2D accelerations:

- Eagle observations include relative tail position/velocity, relative bearing and distance to hens/teammates, and boundary distance/normal; action $a \in \mathbb{R}^2$ is an acceleration vector with subsequent velocity clipping and boundary bounce.
- Hen observations include geometric relations to tail/eagle (line angle, perpendicular distance), approximate chain tension indicators, and boundary distance; actions are 2D accelerations for centering and blocking control.
- Chick observations include relative position/velocity to the nearest hen and eagle; actions tend to move toward protectors and away from threats to reduce stretch penalty and capture risk.

3.6 Physical Parameters and Configuration

The parameter choices balance physical plausibility and training stability: `world_size` bounds exploration; `dt` sets integration fidelity; `chain_links` and `chain_spacing` control flexibility and oscillation; `hen_max_speed` and `eagle_max_speed` differentiate maneuverability to avoid stalemates; `catch_radius` reflects capture locality; `block_margin` encodes a practical blocking width. Values were validated by small grid searches to minimize excessive stretch and boundary collisions while maintaining task difficulty.

Physical Parameter	Value
World half-size `world_size`	20.0
Time step `dt`	$\frac{1}{30}$ s
Max steps per episode `max_steps`	600
Chain links `chain_links`	7
Chain spacing `chain_spacing`	1.0
Hen max speed `hen_max_speed`	9.0
Eagle max speed `eagle_max_speed`	36.0
Capture radius `catch_radius`	0.7
Blocking margin `block_margin`	2.0

Table 1: Overview of physical environment parameters (design and implementation)

3.7 Training Hyperparameters and Configuration

Training Hyperparameter	Value
Parallel environments `n_envs`	16
Steps per environment `n_steps`	256
Batch size `batch_size`	512
Learning rate `learning_rate`	3×10^{-4}
Discount factor `gamma`	0.995
Total steps `total_steps`	1,000,000
Device `device`	auto/cpu

Table 2: Overview of PPO training hyperparameters (design and implementation)

Hyperparameters were tuned for stability and throughput: `n_envs`=16 with `SubprocVecEnv` reduces temporal correlation; `n_steps`=256 balances return estimation and memory; `batch_size`=512 and learning rate 3×10^{-4} follow SB3 baselines; $\gamma = 0.995$ preserves long-horizon rewards. Pilot runs monitored explained variance and policy entropy to avoid under- or over-regularization.

3.8 Chain Dynamics and Team Coordination

Chicks are connected via distance joints to form a flexible chain. Joint target length, frequency, and damping determine chain stiffness and oscillation amplitude. Chain structure directly impacts team coordination:

- **Coupling and delay**: Acceleration or turning of any chick propagates through joints to neighbors, creating temporal lag and phase differences; hens must guide stably during defense to avoid excessive stretch and sway.
- **Shape and formation**: Motion naturally forms head–middle–tail speed/inertia gradients; the tail is easiest for the eagle to approach, so defense prioritizes tail protection and building a blocking line between eagle and tail.
- **Constraints and coordination**: Chain length and tension constrain team maneuverability; chick policies tend to stay within the hen's safety radius and maintain spacing from teammates, reducing stretch penalties and increasing stability.
- **Reward coupling**: Stretch, boundary, and bounce terms simultaneously affect multiple segments'rewards, coupling individual behavior with team outcomes and promoting coordinated evasion/guarding.

3.9 Rewards and Collision Mechanics

The Stage 1 hen total reward R_H is a weighted sum of:

Blocking score	$r_{\text{block}} = \max\big(0,\ 1 - \frac{d(H,\overline{ET})}{a}\big)$	(1)
Avoidance score	$r_{\text{avoid}} = \mathbb{I}[\text{not captured}]$	(2)
Stretch penalty	$r_{\text{stretch}} = -\lambda_s \sum_i \big\lvert \lVert x_i - x_{i-1} \rVert - \ell \big\rvert$	(3)
Tail distance penalty	$r_{\text{tail}} = -\lambda_t \lVert x_T - x_H \rVert$	(4)
Boundary penalty	$r_{\text{bound}} = -\lambda_b \mathbb{I}[\text{out of bounds}]$	(5)
Survival reward	$r_{\text{survive}} = \alpha$	(6)

Define $R_H = \alpha_1 r_{\text{block}} + \alpha_2 r_{\text{avoid}} + \alpha_3 r_{\text{survive}} + r_{\text{stretch}} + r_{\text{tail}} + r_{\text{bound}}$.

Example coefficients used in experiments: R_H combines $0.7\, r_{\text{block}} + 0.3\, r_{\text{avoid}} - 0.05\,\text{stretch} - 0.1\,\text{tail distance} - 0.2\,\text{boundary} + \text{survival bonus}$ with a small time-scaled survival bonus and a -200) penalty upon capture.

The Stage 2 eagle total reward R_E includes:

Capture reward	$r_{\text{catch}} = \kappa\, \mathbb{I}[\text{captured}]$	(7)
Approach potential	$r_{\text{pot}} = \beta\,(\|x_T^{t-1} - x_E^{t-1}\| - \|x_T^t - x_E^t\|)$	(8)
Stretch suppression	$r_{\text{stretch}} = +\eta \sum_i \big(\ell - \|x_i - x_{i-1}\|\big)_+$	(9)
Boundary penalty	$r_{\text{bound}} = -\lambda_b\, \mathbb{I}[\text{out of bounds}]$	(10)
Pacing and speed	$r_{\text{pace}} = \rho\, \|v_E\| - \rho'\, \mathbb{I}[\text{repetitive circling}]$	(11)
Time pressure	$r_{\text{time}} = -\zeta$	(12)

Define $R_E = r_{\text{catch}} + r_{\text{pot}} + r_{\text{stretch}} + r_{\text{bound}} + r_{\text{pace}} + r_{\text{time}}$.

Example coefficients used in experiments: capture +100; potential term coefficient $\beta \approx 2.0$ per unit distance reduction; stretch bonus ≈ 0.5 beyond a baseline; bounce penalty ≈ -2.0; border penalty ≈ -5.0; flanking and speed bonuses ≈ 0.5 and 0.1 scaled; distance pressure ≈ -0.01 distance; time pressure and low-activity penalties increase gradually over the episode.

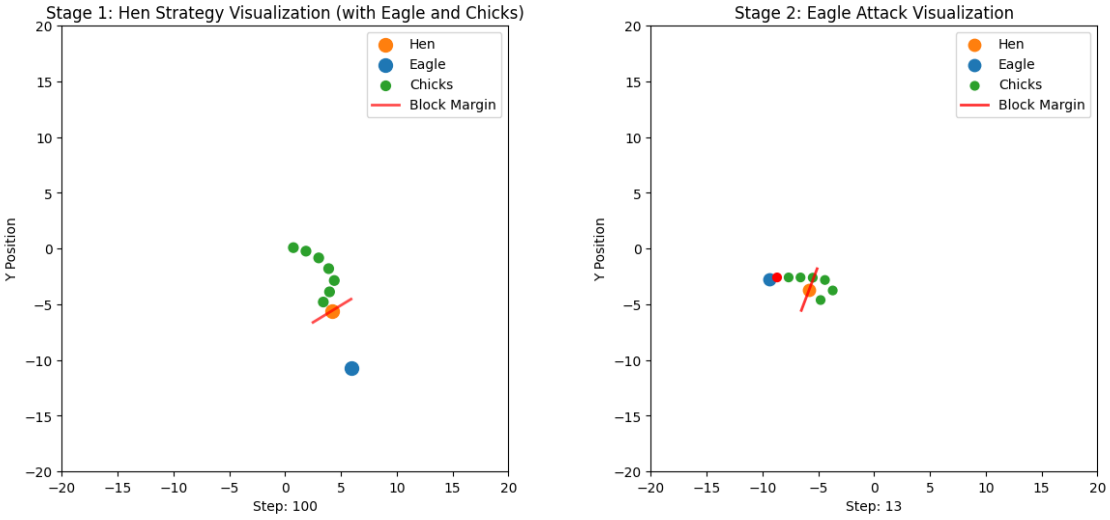
Collisions and bounce are handled uniformly: upon agent–boundary or "wing band" collisions, velocity is decomposed into normal and tangential components; the normal component is reflected with energy loss, and position is offset along the normal to avoid penetration, ensuring stable simulation and interpretability. In implementation, normal reflection uses a restitution of 0.5 and tangential scaling of 0.9; a minimum offset of 0.5 m (or 0.15 times the blocking span) prevents re-penetration and enforces separation.

3.10 Training and Evaluation

Stage 1 uses parallel environments (e.g., $n\_envs = 16$, $n\_steps = 256$ per env, batch size 512, learning rate 3×10^{-4}, discount $\gamma = 0.995$). Stage 2 trains the eagle under a frozen hen policy with the same settings. The evaluation callback periodically saves the best model to the output directory.

3.11 Visualization Window

Agent behaviors are rendered with Matplotlib visualization scripts. Role observations and rewards are summarized in the introduction.



(a) Stage 1 behavior visualization snapshot (hen defense)

(b) Stage 2 behavior visualization snapshot (eagle approach)

Figure 1: Behavior visualization examples across stages

4 Results and Analysis

4.1 Game Rules

Within a bounded 2D world and `max_steps` per episode: the eagle aims to capture the tail; hens guard the chain and prioritize tail protection; blocking along the eagle–tail line is rewarded; out-of-bounds causes bounce and penalties; repetitive circling is penalized. Initial states are randomized within ranges to ensure non-trivial adversaries. Rules are consistent with the introduction and design sections to avoid redundancy.

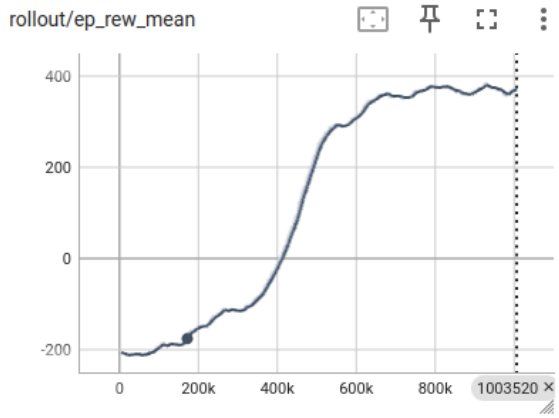
4.2 Result Data Processing

During training we log `ep_rew_mean`, losses and entropy, explained variance, etc. To improve stability and readability, we apply:

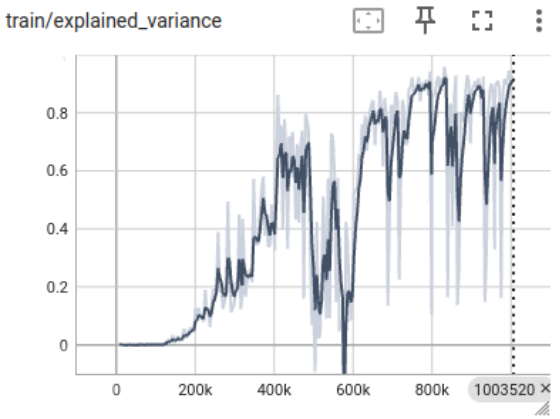
- Metric smoothing: exponential moving average $m_t = (1 - \beta)m_{t-1} + \beta x_t$ (e.g., $\beta = 0.1$) to reduce high-variance fluctuations
- Step alignment: resampling/interpolation to align events from parallel environments to a unified time step for comparable curves
- Warm-start removal: exclude early non-stationary samples (e.g., first N evaluations) to avoid initialization bias

- Metric normalization: zero-mean unit-variance or interval scaling for mixed-dimension indicators to enable joint visualization

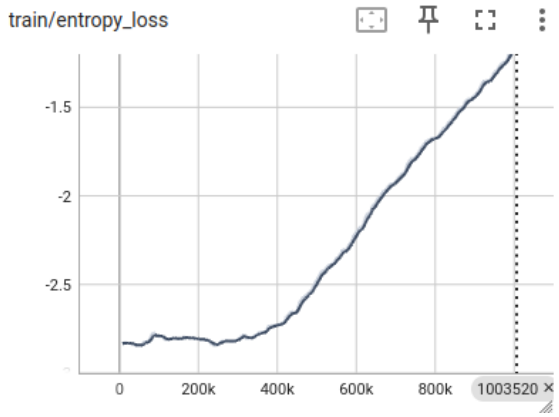
4.3 Training Curves



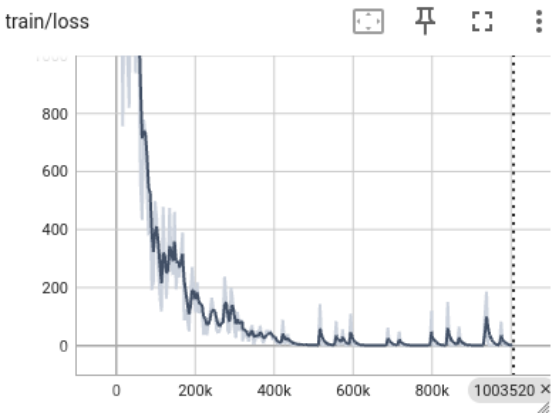
(a) Stage 1 mean episode return vs. timesteps (up to 1e6)



(b) Stage 1 value function explained variance vs. timesteps



(c) Stage 1 policy entropy vs. timesteps

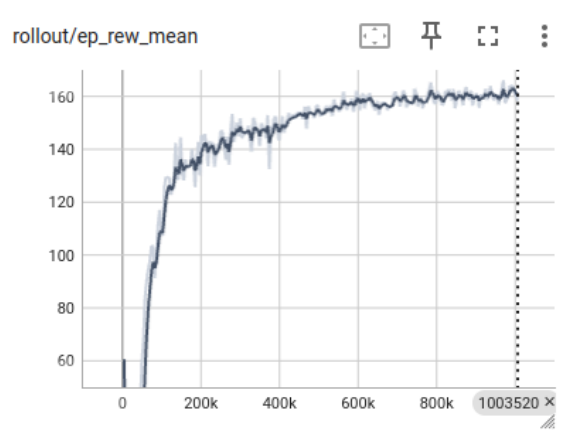


(d) Stage 1 PPO overall loss vs. timesteps

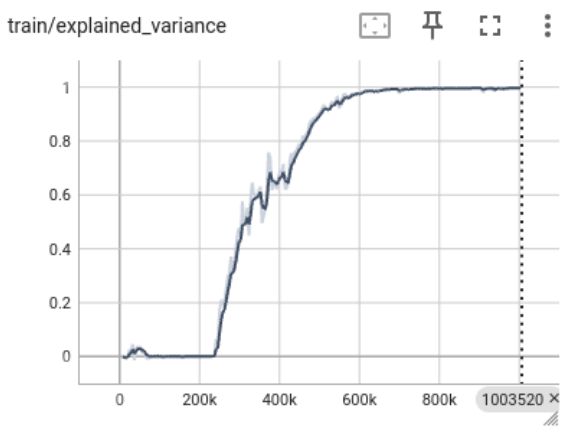
Figure 2: Stage 1 training metrics summary (mean return, explained variance, policy entropy, PPO loss)

4.4 Results Analysis

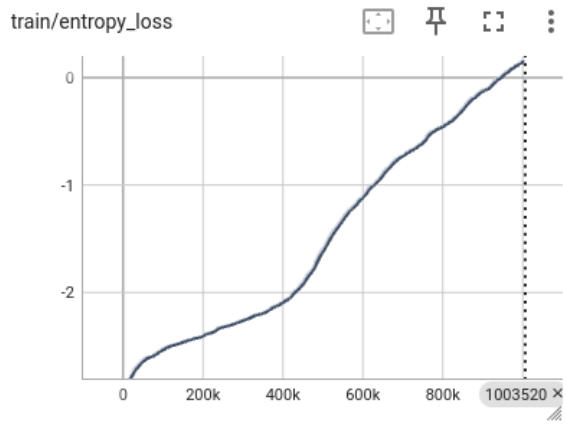
In Stage 1, mean return steadily increases, explained variance approaches 1, and entropy loss decreases, indicating a transition from exploration to more deterministic behavior and a well-fit value function. In Stage 2, the approach potential drives continuous reductions in eagle–tail distance, capture events yield sparse positive returns, and overall loss curves remain smooth. Visualizations show hens staying centered under chain inertia



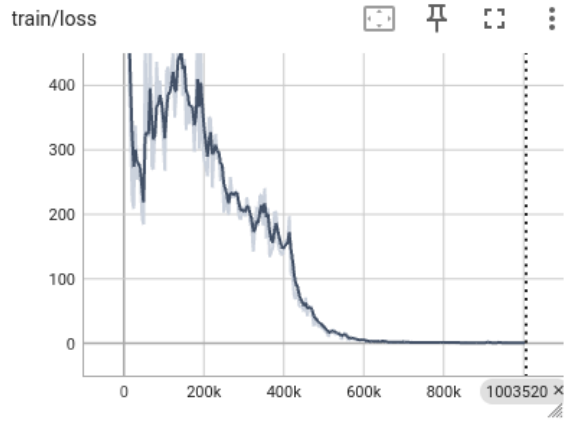
(a) Stage 2 mean episode return vs. timesteps (up to 1e6)



(b) Stage 2 value function explained variance vs. timesteps



(c) Stage 2 policy entropy vs. timesteps



(d) Stage 2 PPO overall loss vs. timesteps

Figure 3: Stage 2 training metrics summary (mean return, explained variance, policy entropy, PPO loss)

and blocking the eagle, while the eagle learns side/rear detours and accelerated approach under a frozen hen policy. Both stages achieve their expected objectives.

5 Conclusion

5.1 Conclusion

This project decomposes the cold-start challenge in adversarial MARL via curriculum learning. With viewpoint normalization, physical bounce and geometric scoring, parallel sampling, and evaluation callbacks, we achieve stable training and visual verification, delivering a reproducible pipeline and model artifacts. Innovations include frozen-opponent inference within the curriculum, geometric blocking scores, and unified bounce mechanics. Limitations include manual tuning of reward weights and physical parameters, and

potential overfitting to boundaries under extreme settings. Future work: introduce joint fine-tuning and self-play, explore richer potentials or inverse RL for better generalization, extend to more complex obstacles/maps, and systematize evaluation metrics with statistical significance testing.

6 Appendix: Running and Reproducibility

Dependencies are installed via `pip` and `conda`. Training scripts are `python src/train_hen.py` and `python src/train_eagle.py`; visualization scripts are `python src/visualize_hen_stage1.py` and `python src/visualize_eagle_stage2.py`. Logs are viewed with `tensorboard --logdir results/curriculum`. Source locations are indicated throughout to ensure visibility of reproduction paths and experiment repeatability.

Entry points and structure: physical environment and curriculum logic are in `src/curriculum_env` training scripts in `src/train_hen.py` and `src/train_eagle.py`; visualization tools in `src/visualize_hen_stage1.py` and `src/visualize_eagle_stage2.py`.

References

[1] Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE Transactions on Cybernetics*, 50(9):3826–3839, 2020.

[2] Jakob Foerster, Yannis M Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2137–2145, 2016.

[3] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016.

[4] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, pages 1861–1870. PMLR, 2018.

[5] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 23(268):1–8, 2022.

[6] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, pages 41–48, 2009.

[7] Hongliang Cai, Yugang Luo, Hongli Gao, et al. A multiphase semistatic training method for swarm confrontation using multiagent deep reinforcement learning. *Computational Intelligence and Neuroscience*, 2023:2955442, 2023.

[8] Yuval Tassa, Yotam Doron, Alistair Muldal, Joel Z Leibo, Tom Erez, and Yuval Li. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

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