Competitive Racing with Proximal Policy Optimization

Introduction

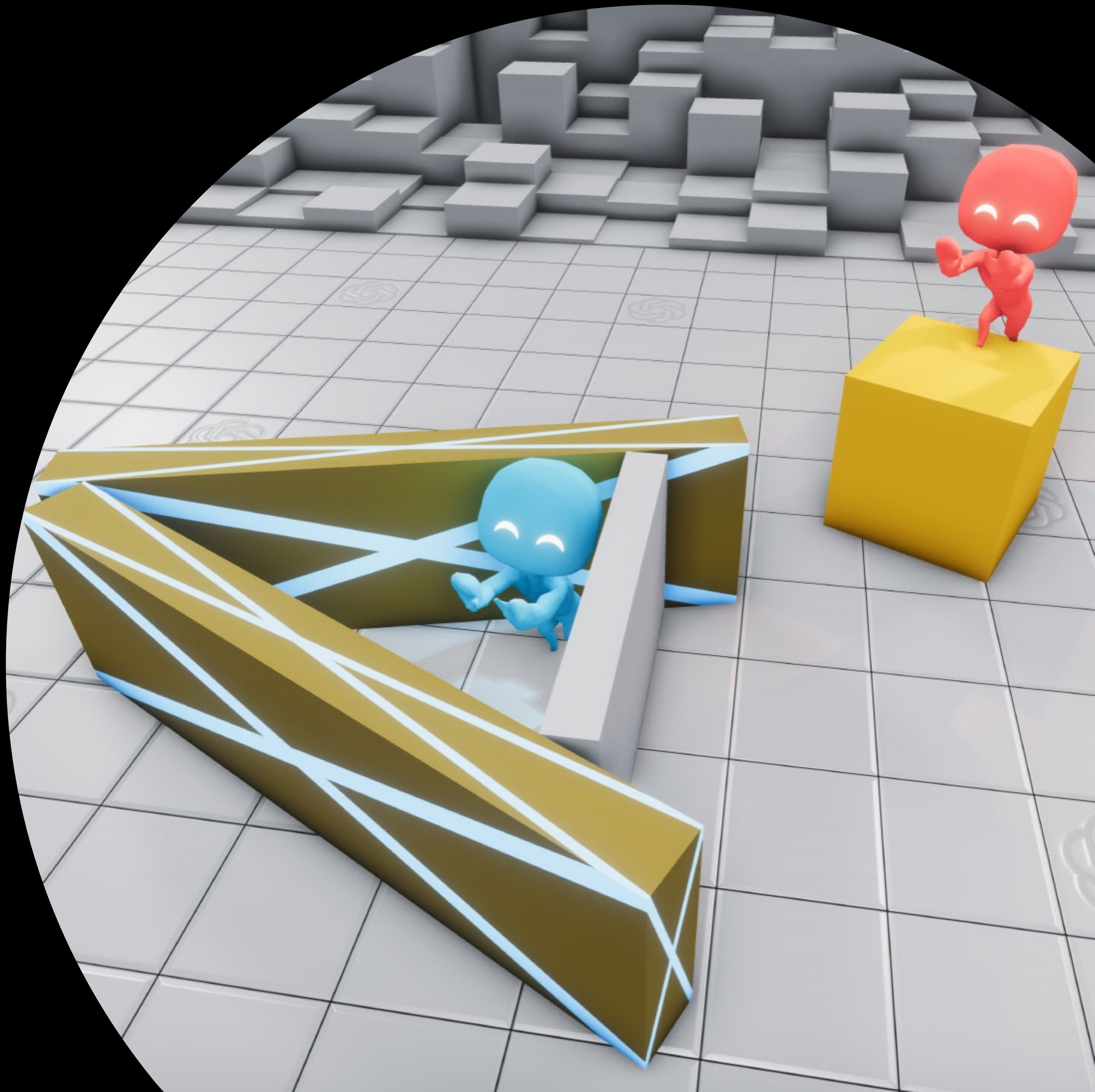
- Motivation
- Problem Statement
- Related Work and Background
- Approach
- Evaluation Procedure
- Results
- Conclusions and Future Work

Motivation

- Racing sports often involve strategies for reducing wind resistance
    - “Slipstream” Effect
    - Reduced Energy Expenditure
    - Improved Efficiency
- Competitive/cooperative strategies emerge
    - When should you lead/draft?
    - Should you work alone or as a team?

Problem Statement

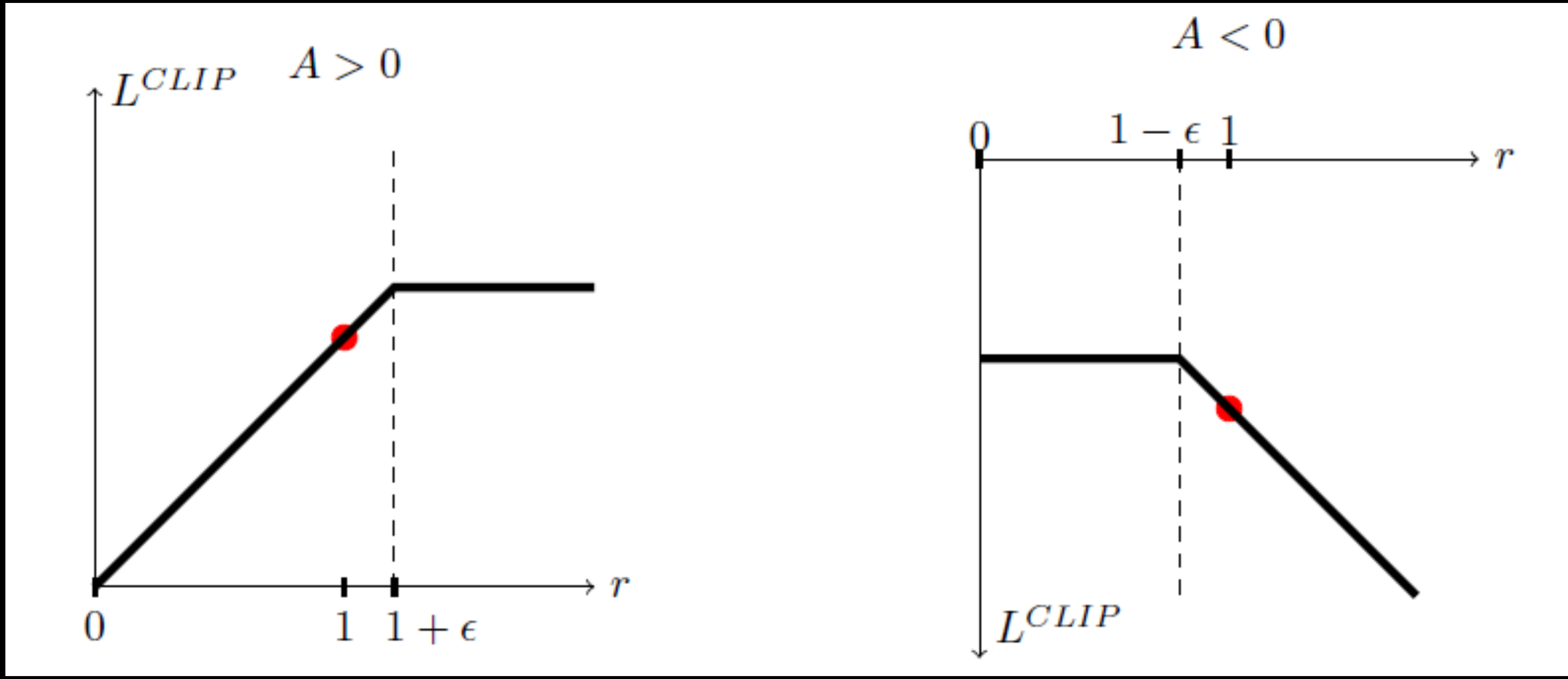
- Can agents learn to compete in a racing game?
  - Simulate the effects of drafting in a racing environment
  - Train agents to race with other agents (MARL)
  - Encourage competitive/cooperative racing
  - Evaluate the emergent strategies



Proximal Policy Optimization (PPO)

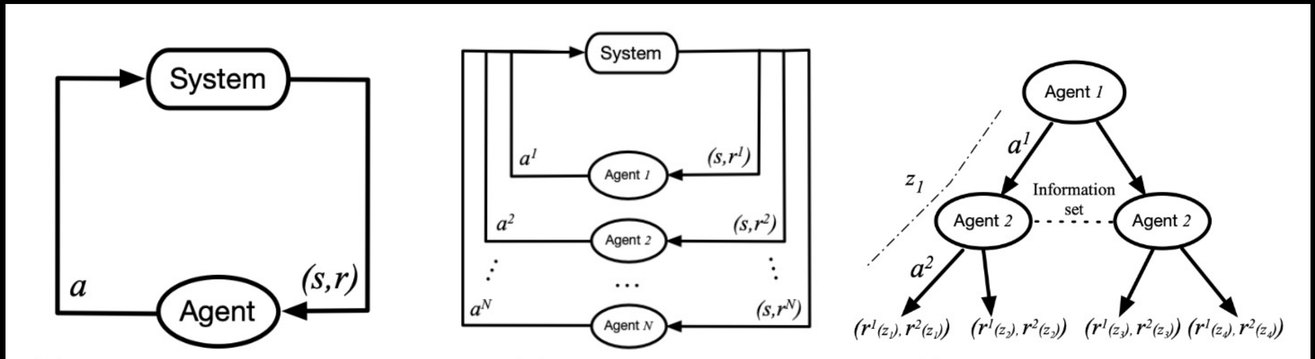
- Limit policy updates by clipping changes in the policy

$$L_t^{CLIP+VF+S}(\theta) = \hat{E}_t[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)]$$

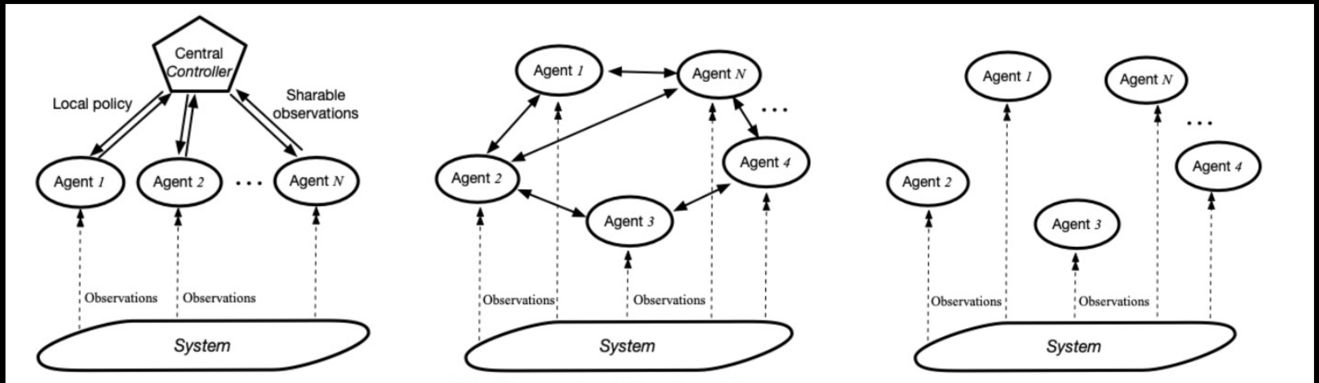


Multi-Agent Reinforcement Learning (MARL)

- Markov/Stochastic game
- Fully decentralized setting
- Partially observed model
- Homogeneous Agents



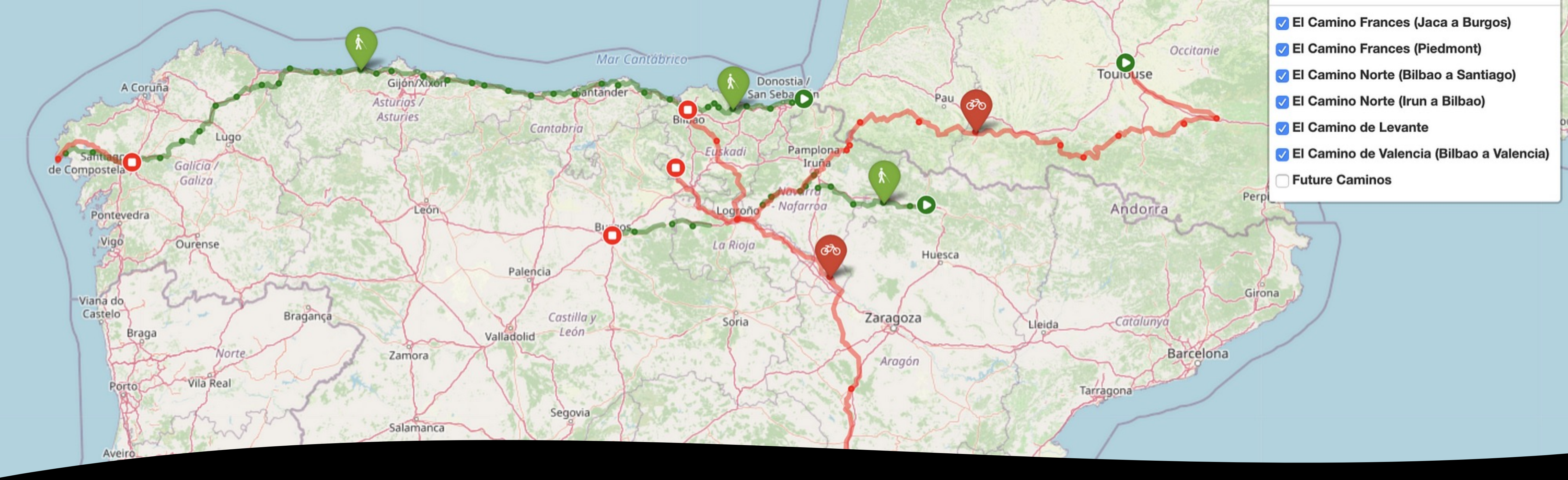
(a) Markov decision process (b) Markov game (c) Extensive-form game



(a) Centralized setting (b) Decentralized setting with networked agents (c) Fully decentralized setting

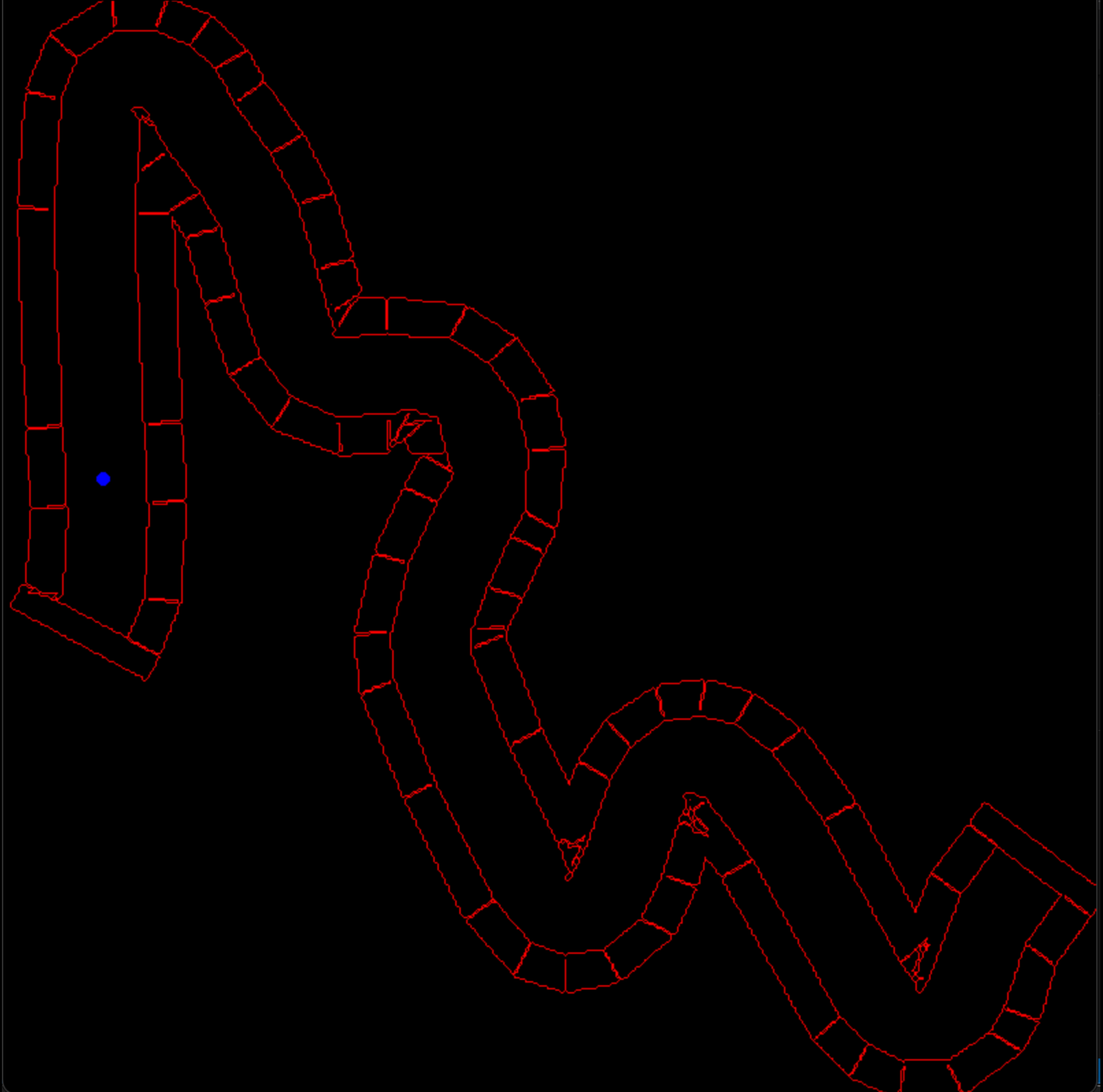
Approach


Map Generation
Environment Creation
PPO Implementation
Single Agent
Multiple Agents (No Drafting)
Multiple Agents (Drafting)



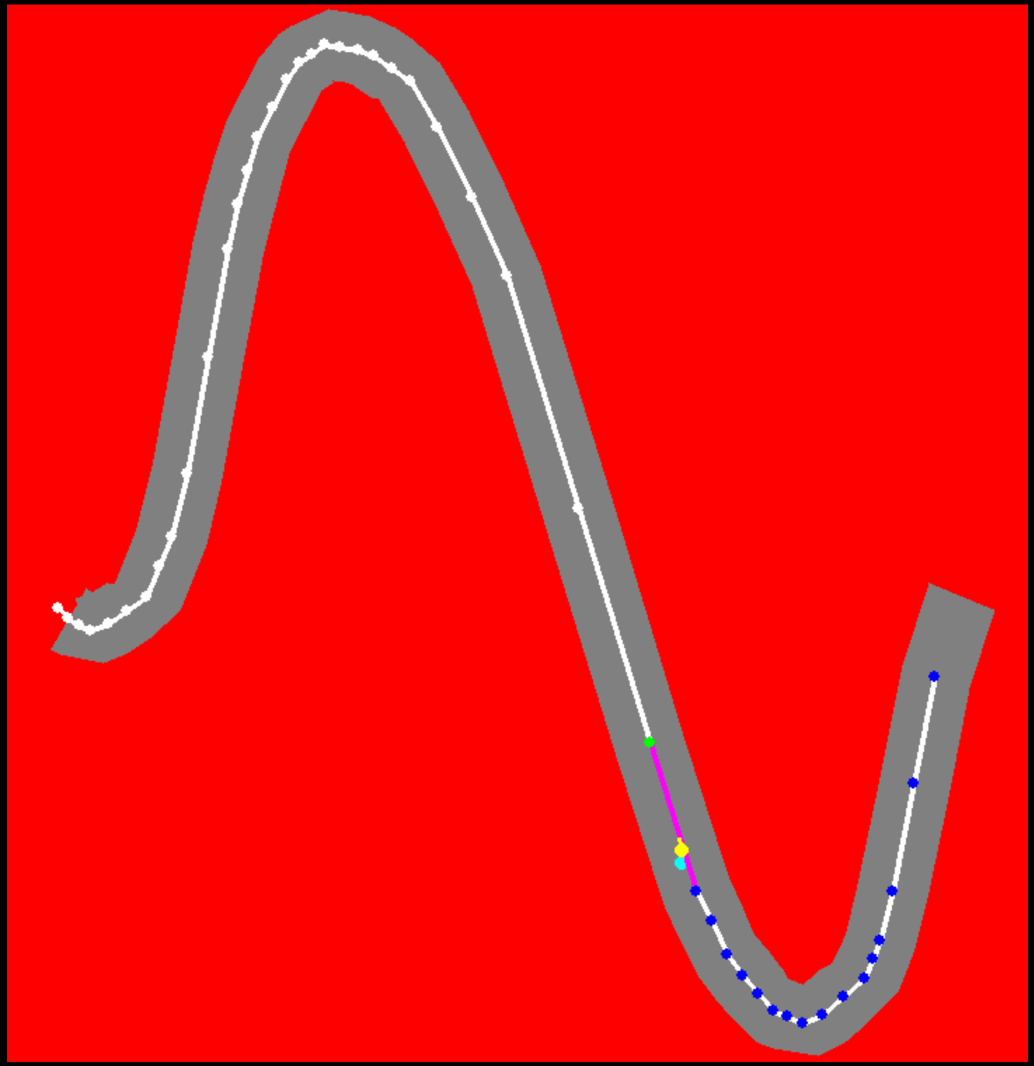
Map Generation

- Where are the agents going to race?
- Idea: Use GPS data in the form of .gpx files
  - Establish "waypoints" based on longitude/latitude
  - Define boundaries of the road as polygons for detecting collisions



Environment Definition

- Map State:
  - 800x800 grid
  - Walls, Waypoint Segments
- Agent State:
  - Position (x, y)
  - Speed $[0, 200]$
  - Heading $[-\pi, +\pi]$
  - Waypoint
- Actions:
  - Steer
  - Throttle
- Agent Observation:
  - 64x64x3 image centered on agent
  - Frame-stacking 3 frames
  - Ego Perspective



PPO Implementation

Algorithm 1 PPO-Clip

1: Input: initial policy parameters θ_0, initial value function parameters ϕ_0

2: **for** $k = 0, 1, 2, ...$ **do**

3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.

4: Compute rewards-to-go $\hat{R}_t$.

5: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.

6: Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \quad g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

typically via stochastic gradient ascent with Adam.

7: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.

8: **end for**

“Nature” CNN

Input: 64x64x3

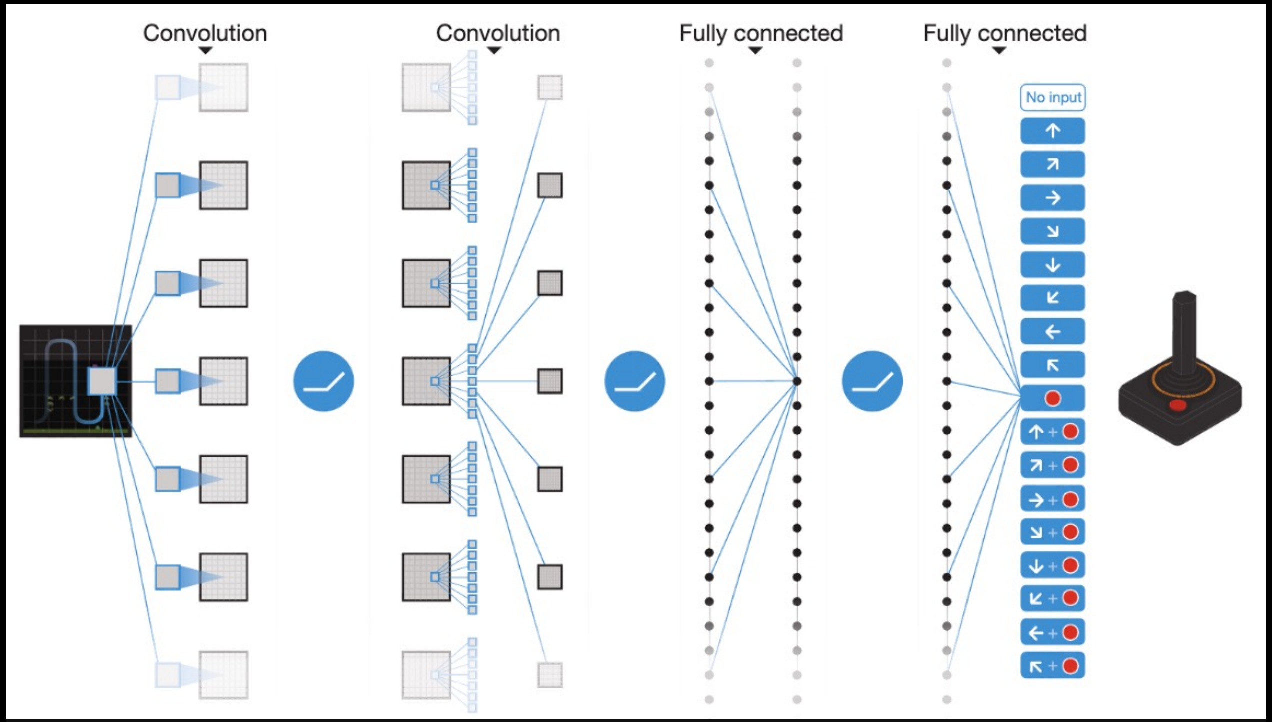
Hidden Layer 1
- 32 filters (8x8), stride = 4

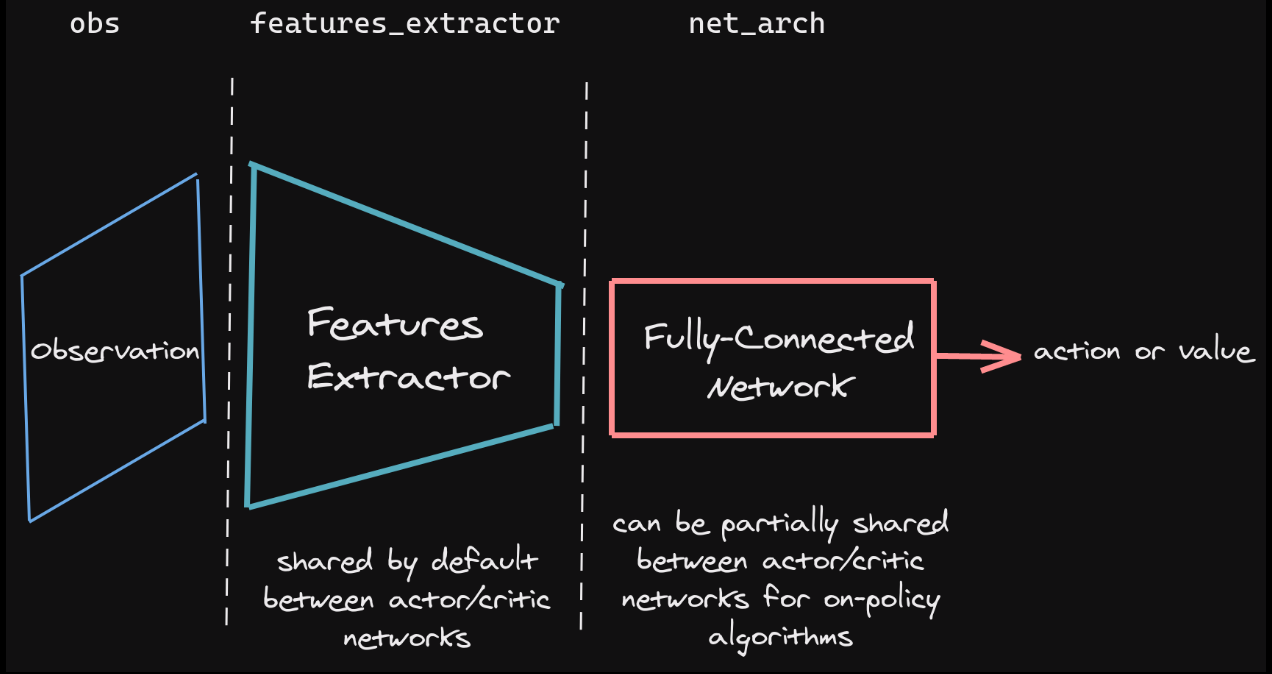
Hidden Layer 2
- 64 filters (4x4), stride = 2

Hidden Layer 3
- 64 filters (3x3), stride = 1

Hidden Layer 4
- Fully Connected
- 512 units

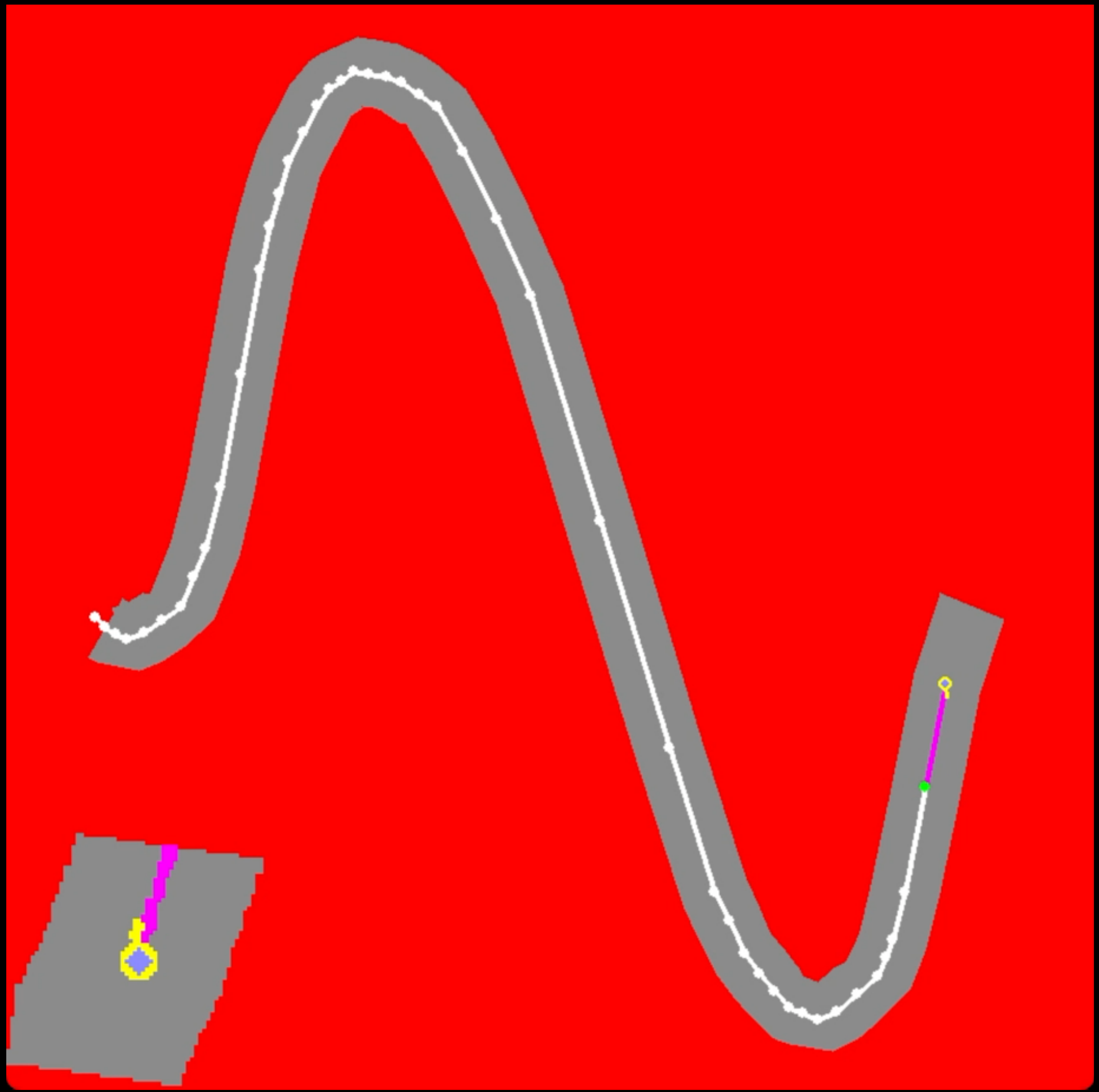
Output Layer:
- Gaussian Distribution (mean, std. dev) of action values
- Steering & Throttle





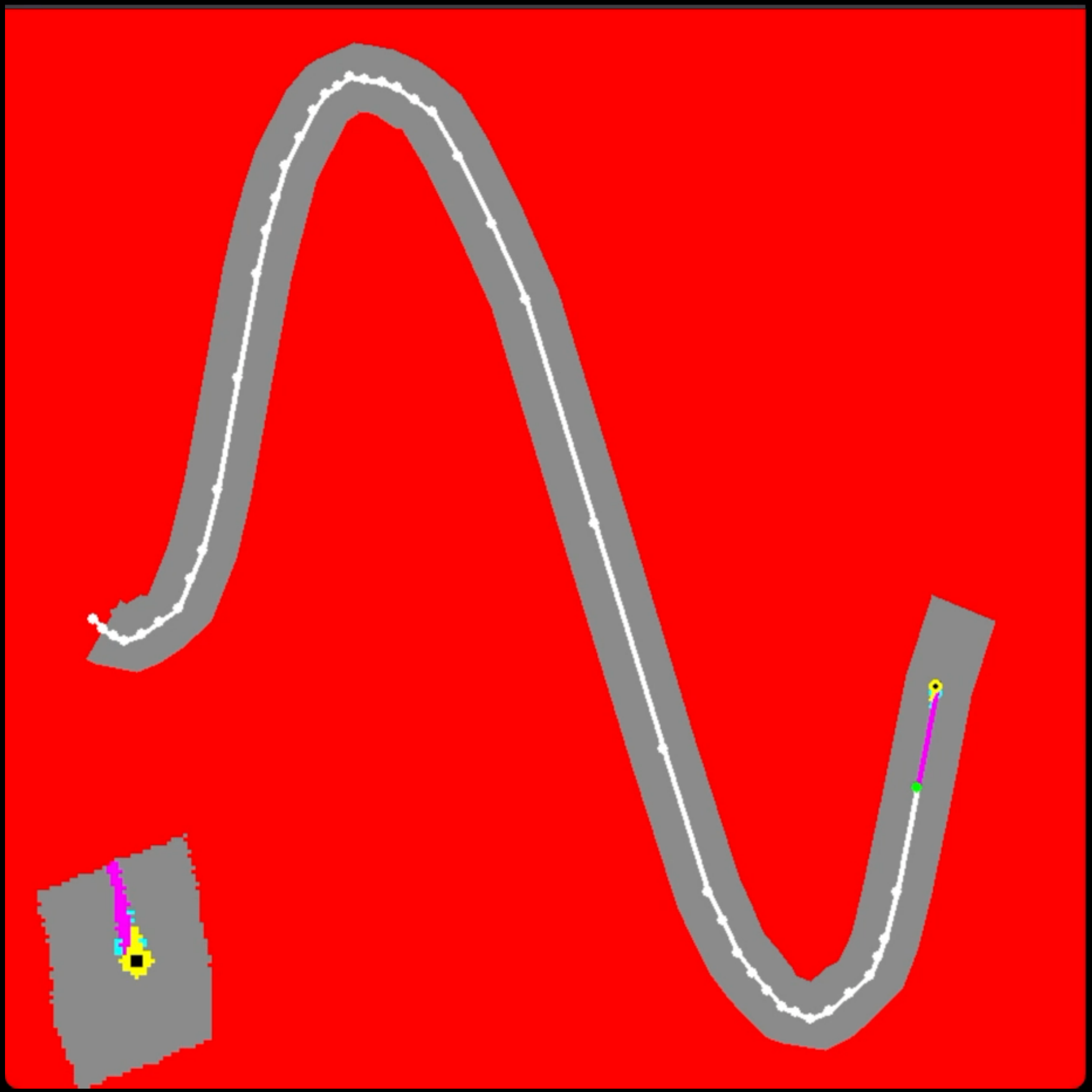
Single Agent

- Checking for “signs of life”
- Can the agent learn to navigate a map?
- Rewards:
  - -0.01 for every timestep
  - +5 for each waypoint
  - +100 for reaching last waypoint
  - -1 for colliding with wall
  - -50 for out of bounds (edge case)



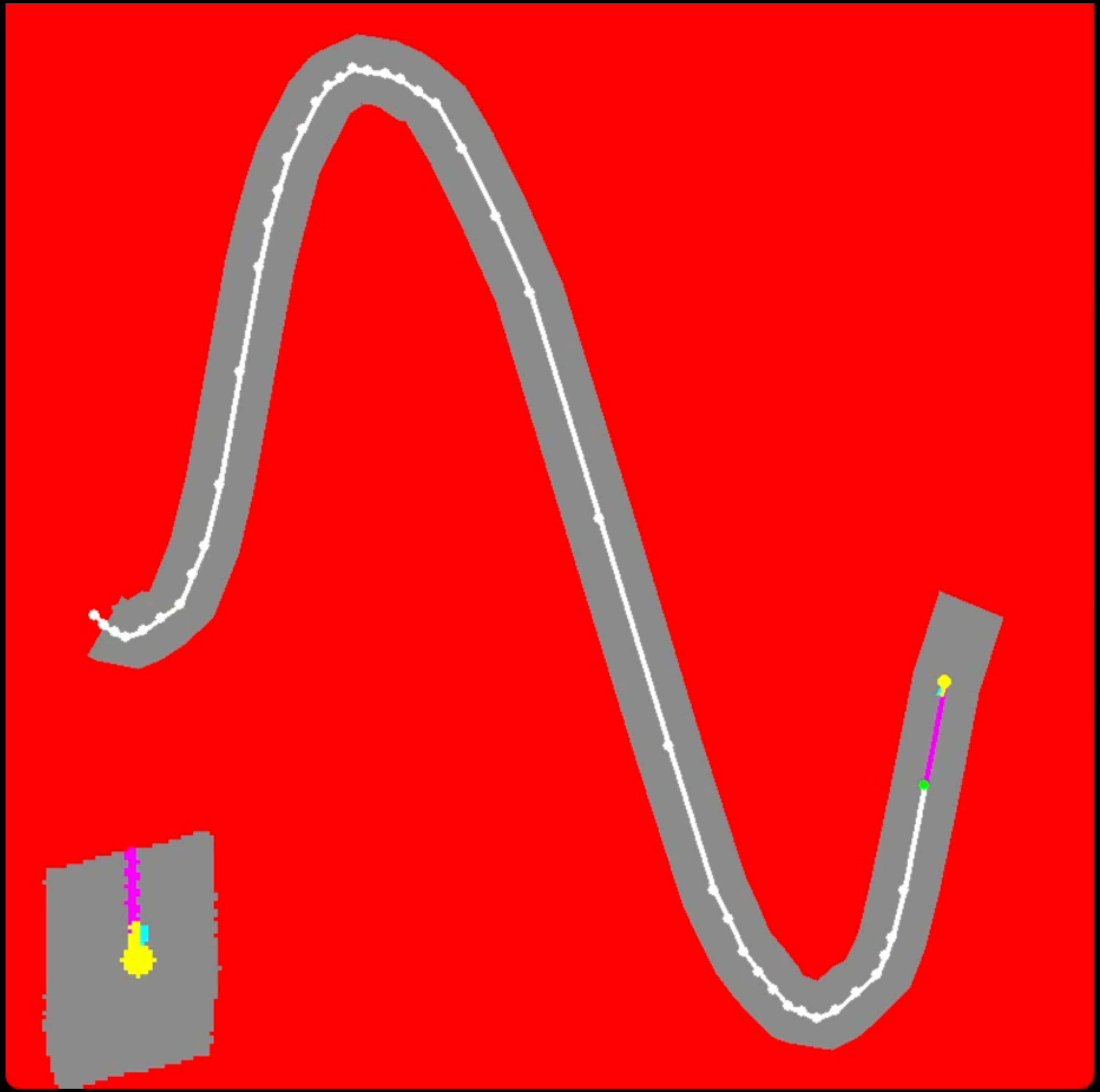
Multi-Agent (No Drafting)

- What happens when two agents are put in the same map?
- Double the reward for being "first"
- Agents can collide with each other
- Race against yourself for ~10,000 timesteps
  - Improve as much as possible
  - Update the opponent for next training session
- Reward Changes:
  - +5 for each waypoint while first
  - +2.5 for each waypoint while behind



Multi-Agent (Drafting)

- How can agents utilize drafting to compete or cooperate?
- Drafting effect when behind another agent
  - Must be within 30 units
  - Velocity increase by a factor of 1.1

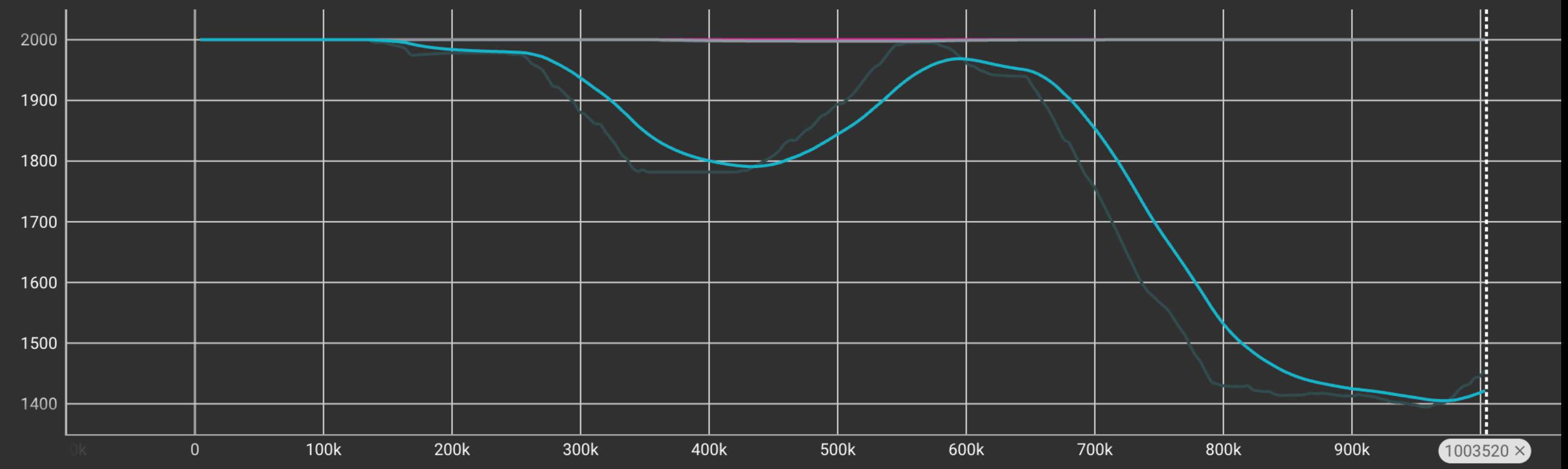


Evaluation Procedure

- Run PPO for each scenario
  - TensorBoard Logging
- Metrics
  - Mean Reward & Episode Length
  - Loss Function Value (Training Only)
- Training (Rollout)
  - Running Average over last 100 episodes
- Evaluation
  - Every 10,000 timesteps
  - Save model and evaluate over 20 episodes

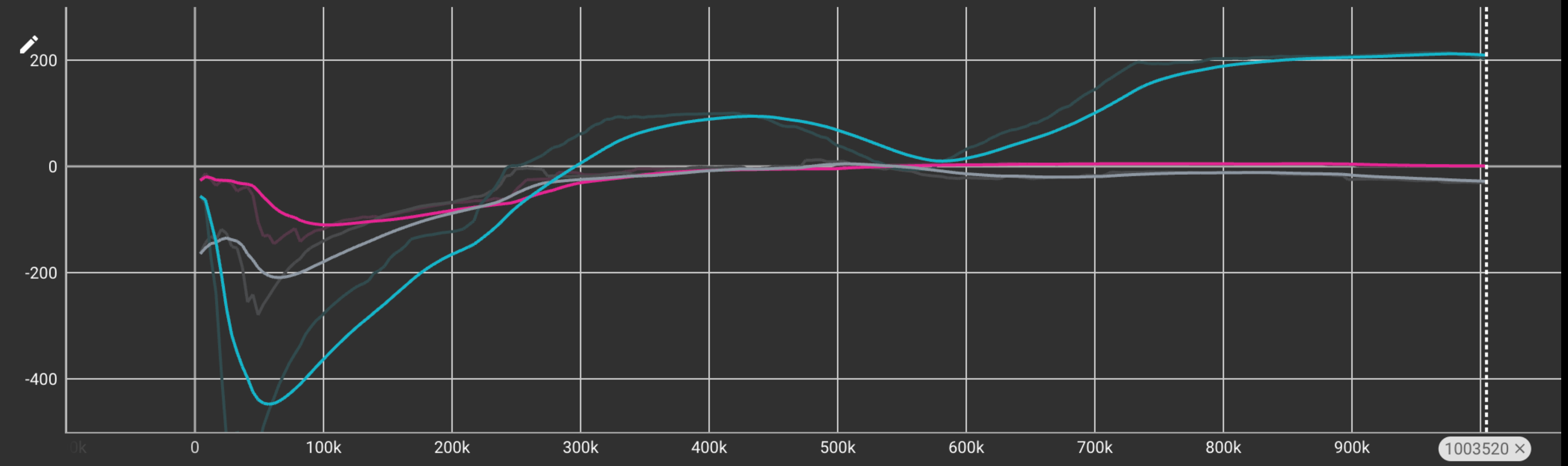
Rollout Results

rollout/ep_len_mean



Run ↑	Smoothed	Value	Step	Relative
multi_draft	1,421.718	1,455.2	1,003,520	1.885 hr
multi_no_draft_no_checkpoints	2,000	2,000	1,003,520	2.017 hr
single_agent	2,000	2,000	1,003,520	1.249 hr

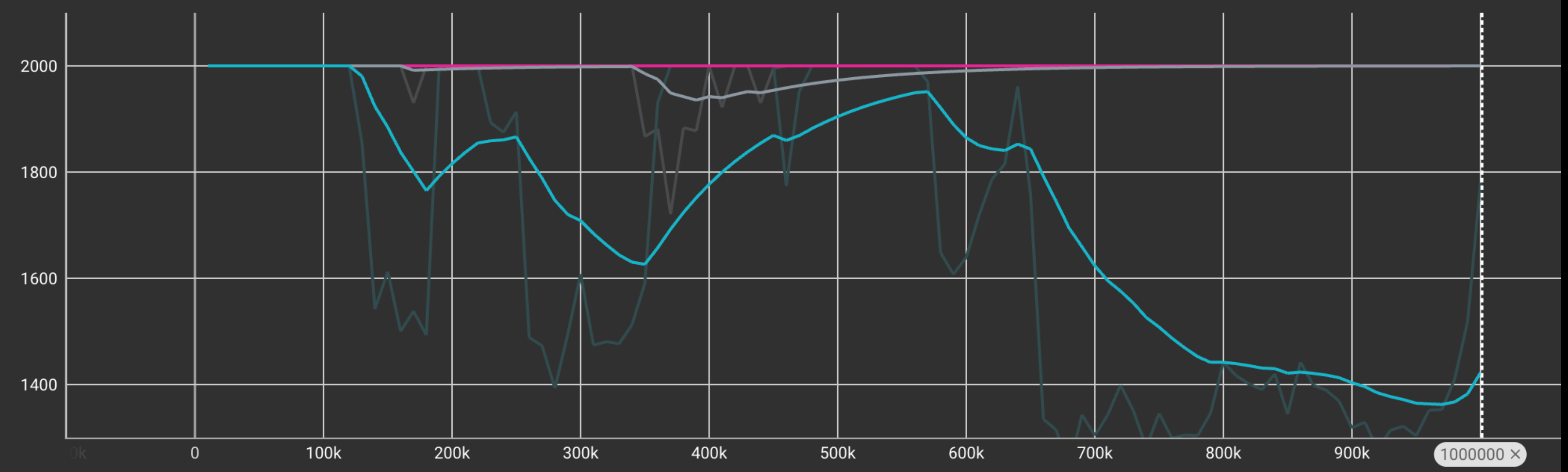
rollout/ep_rew_mean



Run ↑	Smoothed	Value	Step	Relative
multi_draft	209.3615	201.473	1,003,520	1.885 hr
multi_no_draft_no_checkpoints	-27.8224	-30.735	1,003,520	2.017 hr
single_agent	0.934	0.65	1,003,520	1.249 hr

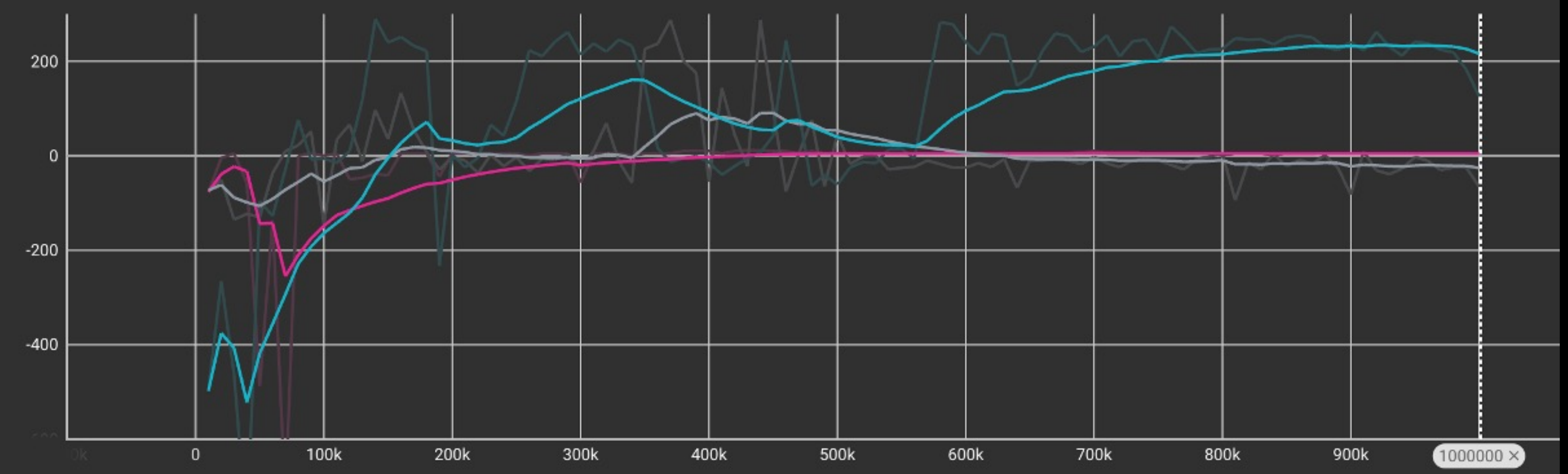
Evaluation Results

eval/mean_ep_length



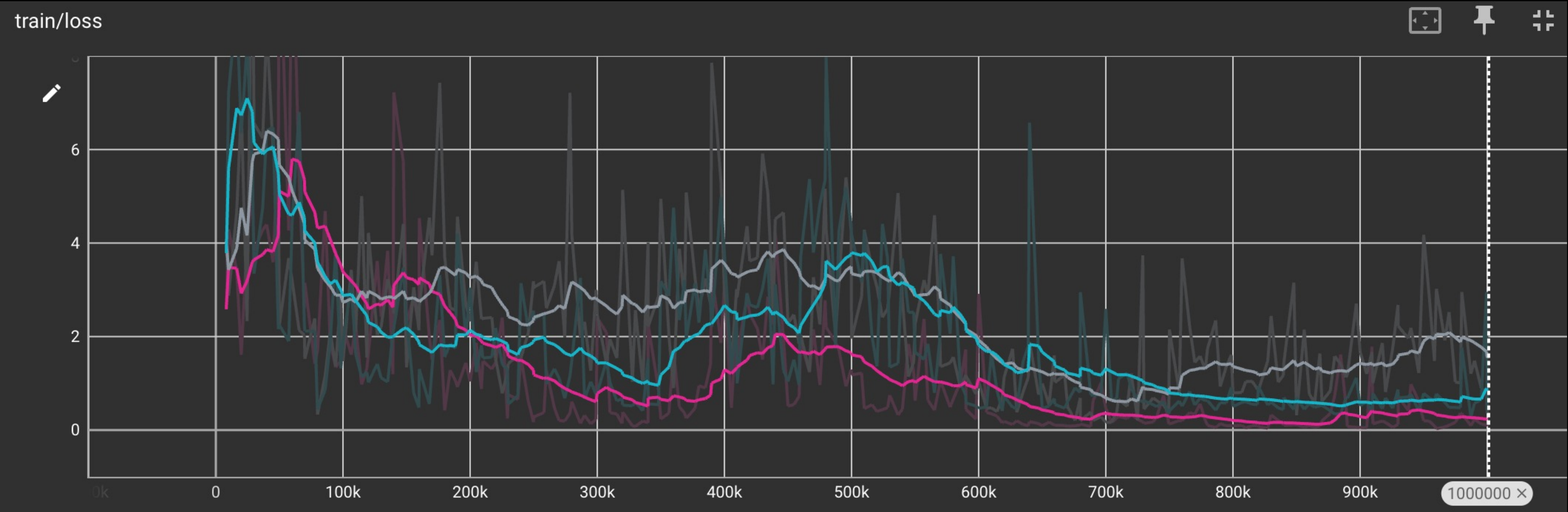
Run ↑	Smoothed	Value	Step	Relative
multi_draft	1,422.1037	1,780	1,000,000	1.87 hr
multi_no_draft_no_checkpoints	1,999.8614	2,000	1,000,000	1.999 hr
single_agent	2,000	2,000	1,000,000	1.237 hr

eval/mean_reward



Run ↑	Smoothed	Value	Step	Relative
multi_draft	216.1528	125.7	1,000,000	1.87 hr
multi_no_draft_no_checkpoints	-25.8688	-65.4	1,000,000	1.999 hr
single_agent	4.9192	5	1,000,000	1.237 hr

Loss Results



	Run ↑	Smoothed	Value	Step	Relative
	multi_draft	0.8422	0.3962	1,000,000	1.878 hr
	multi_no_draft_no_checkpoints	1.5451	0.3608	1,000,000	2.008 hr
	single_agent	0.2194	0.0785	1,000,000	1.244 hr

Conclusions

- PPO can train agents to navigate, race, and cooperate (sort of)
- MARL & MDRL is complex
  - Self-play is too simple
  - Independent and decentralized agents are limited
- Reward shaping is important
  - Sparse/delayed rewards makes learning difficult
- Training collapses are difficult to avoid with on-policy methods
  - Catastrophic forgetting

Future Work

Complex State Representation

Hyperparameter Tuning

Multi-Agent Teams

Information Sharing Network

Fictitious Self-play

Algorithm Comparison

Appendix - Hyperparameters

Hyperparameter	Value
Learning Rate	0.0003
Number of Steps Per Update (Batch Size)	4096
Minibatch Size	1024
Gamma	0.99
Clipping Range (ε)	0.2
Value Function Loss Coefficient	0.5
Entropy Loss Coefficient	0.01

$$L_t^{CLIP+VF+S}(\theta) = \hat{E}_t[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)]$$

Appendix – 10 Agents Demo

https://www.youtube.com/watch?v=u048982E9OE