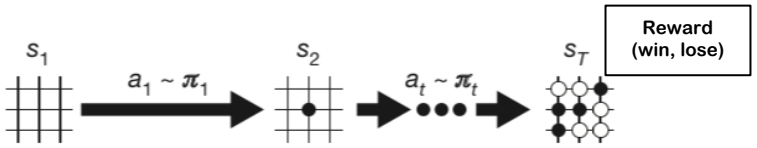
Learning Self-imitating Diverse Policies

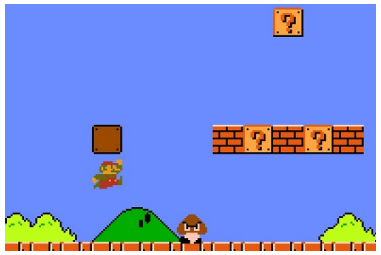
Tanmay Gangwani, Qiang Liu, Jian Peng
ICLR'19

Motivation

- Policy-gradients or Q-learning perform sub-optimally when rewards are delayed or episodic
- Temporal Credit Assignment is hard



GO: *Reward signal only at the end of the episode*



Super Mario: *Many actions don't yield any external reward signal*

Self-Imitation via Divergence-Minimization

Main idea → Efficiently exploit past *good* behaviors

- Maintain a buffer ***B*** of high return trajectories from the previous agent rollouts
  - We use a priority queue (min-heap). Trajectory return is the priority
  - Agent exploration could be guided by action- or parameter-space noise, curiosity etc.

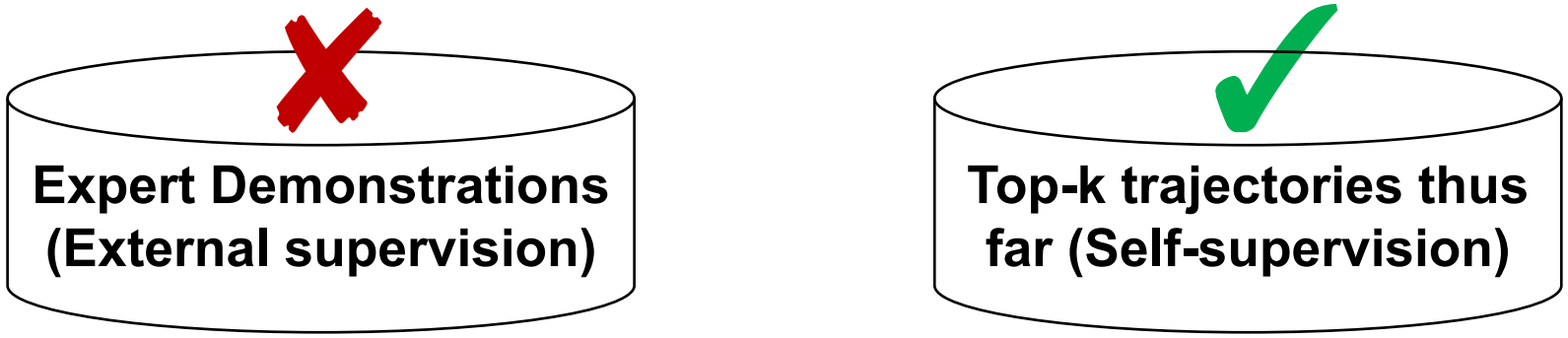
- Policy learning objective: minimize divergence *(in some distance metric)* between state-action distributions of policy and buffer

$$\min_{\pi} \mathbb{D}(\rho_{\pi}, \{s_i, a_i\}_{\mathcal{B}}) \qquad \mathbb{D} : \text{distance metric}$$

Self-Imitation via Divergence-Minimization

- Similar to GAIL[1], using Jenson-Shanon divergence as the distance metric gives a min-max objective

$$\min_{\pi} \max_{D} \mathbb{E}_{(s,a)\sim\rho_{\mathcal{B}}}[\log D(s,a)] + \mathbb{E}_{(s,a)\sim\rho_{\pi}}[\log(1 - D(s,a))]$$



1. Generative Adversarial Imitation Learning (Ho & Ermon, 2016)

Self-Imitation via Divergence-Minimization

- Similar to GAIL[1], using Jenson-Shanon divergence as the distance metric gives a min-max objective

$$\min_{\pi} \max_{D} \mathbb{E}_{(s,a)\sim\rho_{\mathcal{B}}} [\log D(s,a)] + \mathbb{E}_{(s,a)\sim\rho_{\pi}} [\log(1 - D(s,a))]$$

Gradient of this objective w.r.t policy parameters is the policy-gradient with rewards $r(s,a) = -\log(1 - D(s,a))$

1. Generative Adversarial Imitation Learning (Ho & Ermon, 2016)

Combining Reward Sources

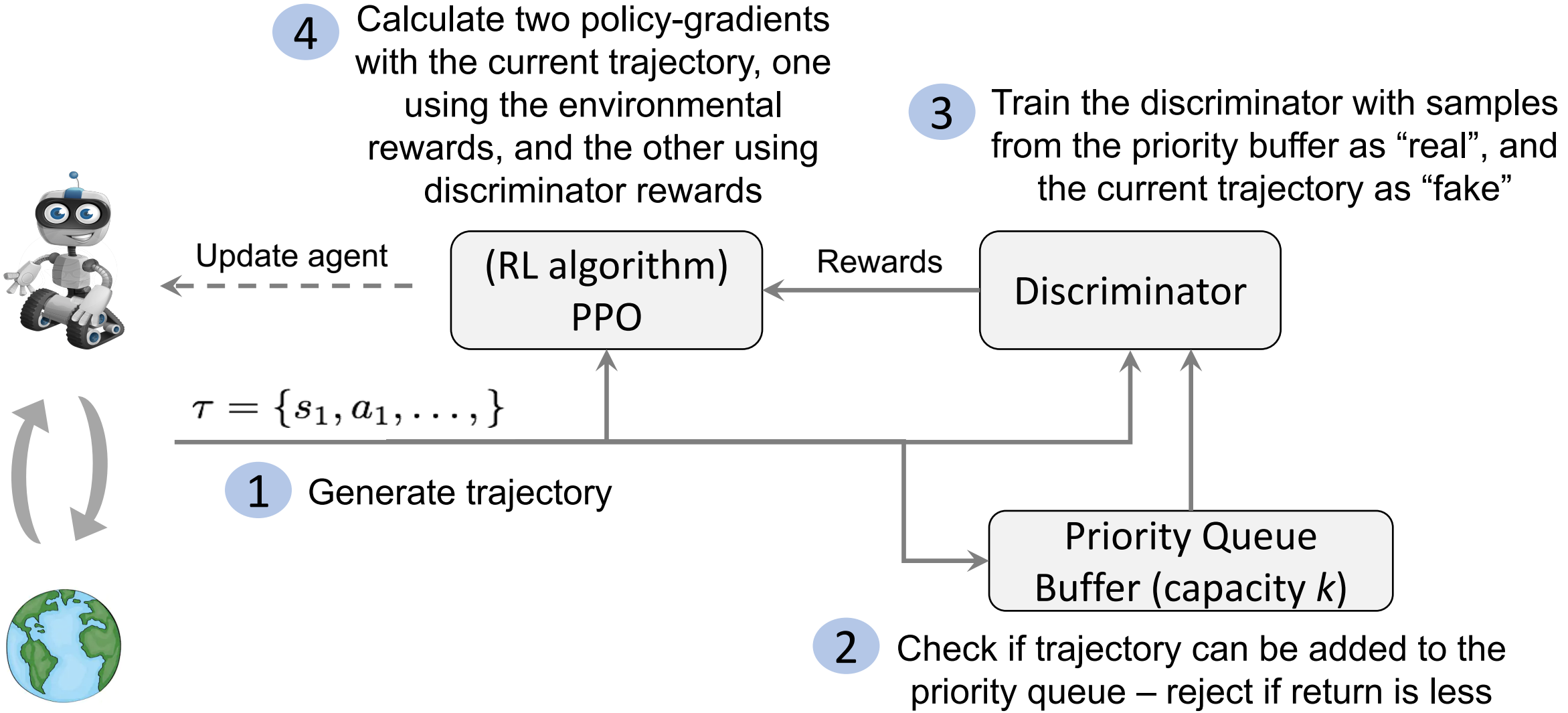
$$\nabla_\theta \eta(\pi_\theta) = \nabla_\theta \mathbb{E}_{(s,a)\sim\rho_\pi}[r^{\text{env}}(s,a)] + \beta \nabla_\theta \mathbb{E}_{(s,a)\sim\rho_\pi}[-\log(1 - D(s,a))]$$

Policy Gradient with real environmental rewards

Policy Gradient with synthetic rewards from a learned discriminator

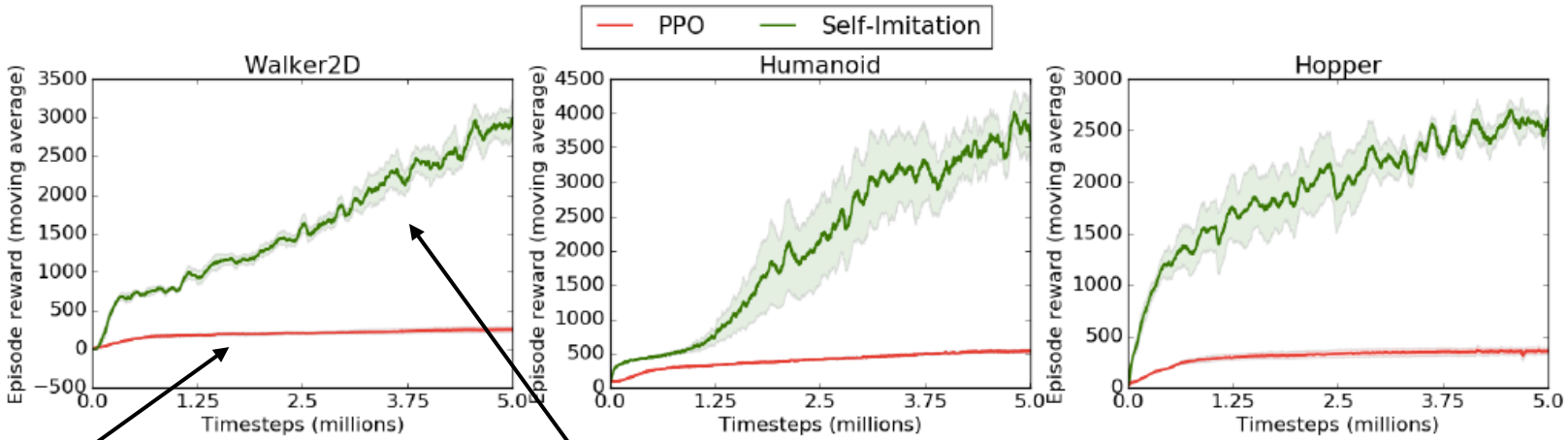
- Environmental rewards $r^{env}(s,a)$ can be sparse or delayed
- Synthetic rewards are **dense** for each (s,a), helping with temporal credit assignment

Algorithm Schema



Performance on Locomotion Tasks

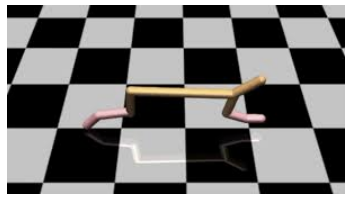
(with episodic rewards)

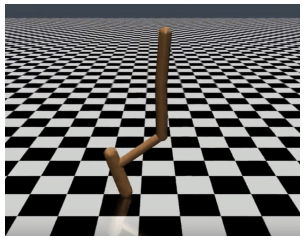


	SI	PPO
Walker	2996	252
Humanoid	3602	532
H-Standup ($\times 10^4$)	18.1	4.4
Hopper	2618	354
Swimmer	173	21
Invd.Pendulum	8668	344

Baseline PPO with only sparse environmental rewards is poor due to difficulty in temporal credit assignment

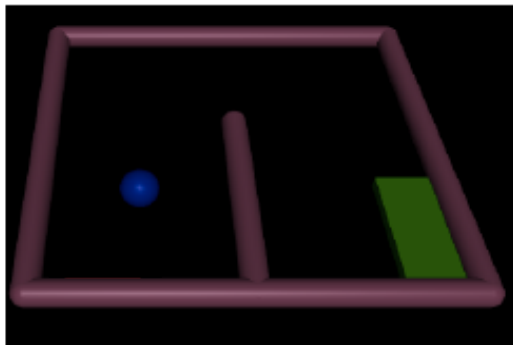
Adding dense discriminator rewards created from self-generated past good trajectories enables efficient learning



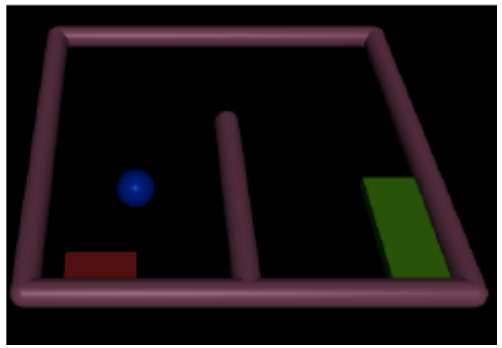


Limitations of Self-Imitation

🚫 The approach is reliant on the presence of high return (and correct) trajectories in the priority buffer. A few failure cases are presented



No reward signal in trajectory - Buffer has random trajectories, and SI is not useful until the reward is obtained at-least once



Deceptive Rewards - Buffer has undesirable trajectories, and SI hastens convergence to sub-optimal reward

Diverse Policy Ensemble

- **Proposed Solution** - Train an ensemble of interacting Self-imitation agents, with an explicit diversity enforcement
- Max-entropy objective over a policy distribution: $\max_q \mathbb{E}_{\theta\sim q}[\eta(\theta)] + \alpha H(q)$
- **Stein Variational Gradient Descent** as the Bayesian inference algorithm to sample from the resulting energy-based distribution#
  - It represents *q* with an ensemble of particles (policies) $\{\theta_i\}_{i=1}^n$ which are iteratively updated as:

$$\theta_i \leftarrow \theta_i + \epsilon\Delta\theta_i, \qquad \Delta\theta_i = \frac{1}{n}\sum_{j=1}^{n}\left[\nabla_{\theta_j}\eta(\pi_{\theta_j})k(\theta_j,\theta_i) + \alpha\nabla_{\theta_j}k(\theta_j,\theta_i)\right]$$

Policy Gradient

Kernel

 # Stein Variational Policy Gradient, Liu et al.

Diverse Policy Ensemble

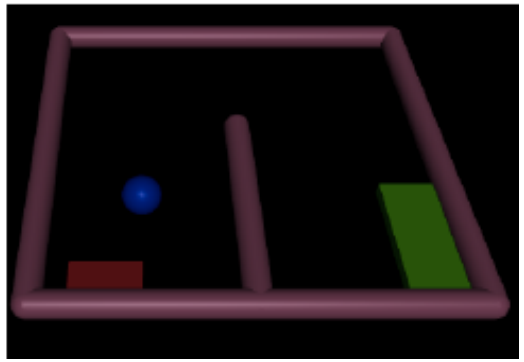
$$\theta_i \leftarrow \theta_i + \epsilon \Delta\theta_i, \qquad \Delta\theta_i = \frac{1}{n}\sum_{j=1}^{n}\left[\nabla_{\theta_j}\eta(\pi_{\theta_j})k(\theta_j,\theta_i) + \alpha\nabla_{\theta_j}k(\theta_j,\theta_i)\right]$$

Policy Gradient

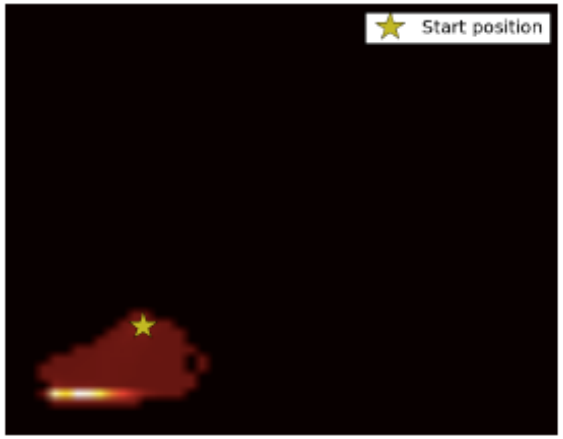
Kernel

- Kernel based on JS divergence: $k(\theta_j,\theta_i) = \exp(-D_{JS}(\rho_{\pi_{\theta_j}}, \rho_{\pi_{\theta_i}})/T)$
- $\Delta\theta_i$ then includes a term of the form $\nabla_{\theta_i} D_{JS}(\rho_{\pi_{\theta_j}}, \rho_{\pi_{\theta_i}})$ which is the **repulsion gradient**, pushing policies ***i*** and ***j*** apart in the state-action space
- Repulsion gradients are calculated using the policy-gradient theorem, with rewards obtained from trained discriminators

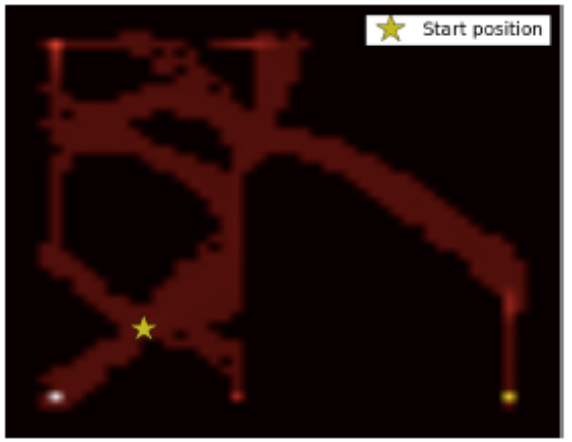
Diverse-SI Ensemble in 2D-Maze



Deceptive Rewards



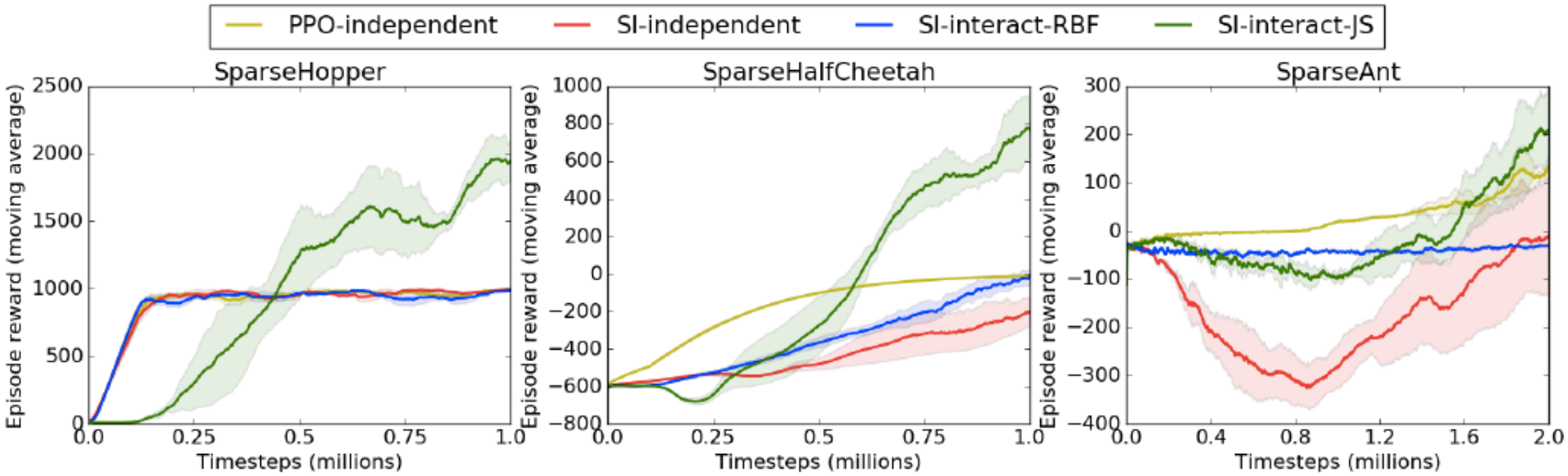
(a) SI-independent state-density



(b) SI-interact-JS state-density

- Baseline → 8 independent SI agents
- Ours → 8 interacting SI agents with D_{JS} repulsion

Diverse-SI Ensemble in Modified MuJoCo



- Forward velocity reward is only provided if the center-of-mass of bot is beyond a certain (pre-specified) threshold distance

- SI-interact-RBF baseline uses RBF kernel[1] for SVGD

1. Stein Variational Policy Gradient, Liu et al.

Key Takeaways

- ✓ **Diversification with SVGD helps in discovery of sparse rewards**
- ✓ **Self-imitation then helps to efficiently exploit the discovered rewards**