Improving PPO Sample Efficiency on Sparse Reward Environments

LUH|AI

Leibniz Universität Hannover

Johny Tarbouch & Qasim Sefaldeen

Poster Presentations in context of Reinforcement Learning Lecture

1 Motivation

PPO
explore with fresh data

off-policy
Buffer Store successful episodes

RND
Intrinsic Reward

2 Problem Setting

PPO: Collect → Update → Discard

Off-Policy: Collect → Update → Reuse

Sparse Rewards → Long Exploration → Sample Inefficiency

Trade-off

Less Stable

3 Approach

SIL Loop

- Collect rollout τ using πθ store in PPO buffer
- Update πθ, Vϕ using PPO on τ

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

$$L(\theta) = L^{CLIP}(\theta) + c_1 L^{VF}(\theta) + c_2 S[\pi_\theta](s_t)$$

- Store positive episodes τ in SIL buffer

→ Is Episodic Return > threshold

Decrease threshold if successful episodes > 5

recent 20 episode max > current threshold

→ Sampling

$$p_i = \max(return_i, \epsilon)^{\alpha}$$

Prioritization

$$P(i) = \frac{p_i}{\sum_k p_k}$$

Sampling

40%
25%
15%

Higher return → More sampling

- if |BSIL| ≥ batch_size and i % update_freq = 0 then

$$\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \qquad w_t = \left(\frac{1}{N \cdot P(i)}\right)^{\beta}$$

remove bias

$$L^{SIL}_{policy} = -\mathbb{E}_{(s,a,R)\sim B_{SIL}}\left[\min\left(r_t(\theta) \cdot A_t^+, clip(r_t(\theta), 1-\epsilon, 1+\epsilon) \cdot A_t^+\right) \cdot w_t\right]$$

$$(R - V(s))^+ > 0$$

$$L^{SIL}_{value}(\phi) = \mathbb{E}_t\left[(V_\phi(s_t) - R_t)^2 \cdot w_t\right]$$

PPO RND

1 Initialization
- Networks: π_θ, V_φ, f (fixed), f (trainable)
- Normalizers: N_{obs}, N_{rew}

2 Environment Rollout
- Collect T-step trajectories using π_θ
- Store normalized experiences in buffer $\mathcal{B}$

3 RND Intrinsic Reward
- Compute prediction error: $\varepsilon_{RND} = ||f(s'_{t+1}) - f(s'_{t+1})||^2$
- Normalize and scale: $r^i_t = N_{rew}(\varepsilon_{RND})$
- Total reward: $R_t = r^e_t + \eta \cdot r^i_t$

4 PPO & RND Updates
- Compute GAE advantages $\hat{A}_t$
- Update π_θ and V_φ using PPO loss
- Train RND predictor f to minimize: L_{RND} = MSE(f(s'), f(s'))

Environment Interaction
Agent explores using π_θ

↓

RND Curiosity
Novel states = high intrinsic reward

↓

Combined Reward
$R = r^{extrinsic} + \eta \cdot r^{intrinsic}$

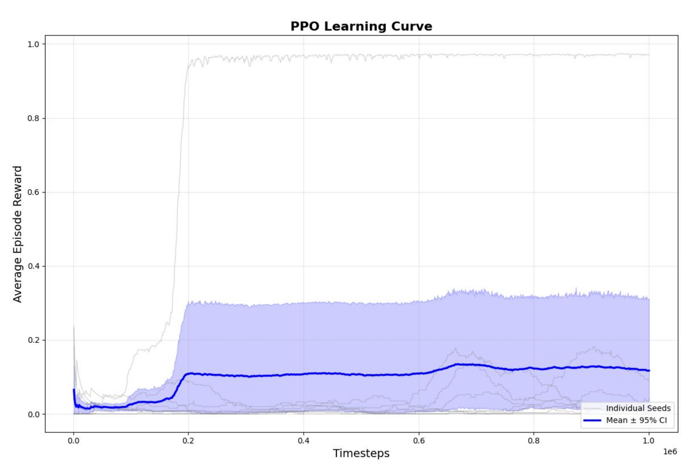
↓

Policy Learning
PPO optimizes exploration

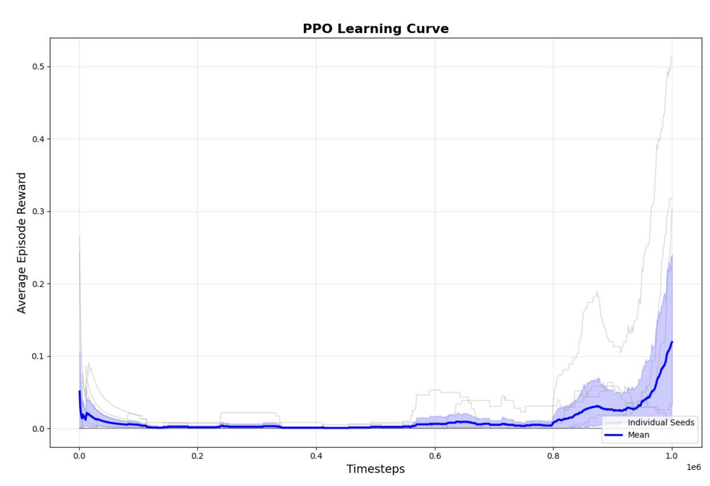
4 Experiment & Results

PPO Parameters		SIL Parameters		RND Parameters	
Learning rate	1×10^{-4}	SIL learning rate	1×10^{-4}	RND learning rate	1×10^{-4}
Discount factor (γ)	0.99	Buffer size	10,000	Intrinsic coef. (η)	0.05
GAE lambda (λ)	0.95	Batch size	32	Embedding dim	128
Clip ratio	0.14	Update ratio	4	Update epochs	1
Entropy coefficient	0.01	**Update frequency**	[3, 5]	Obs normalization	True
Value coefficient	0.5	**Success threshold**	[0.7, 0.8, 0.9]	Reward normalization	True
Rollout size	1024	Warmup episodes	50		
Batch size	64	**Priority exponent** (α)	[0.4, 0.6, 0.8]		
Hidden dimension	256	Importance sampling (β)	0.4		
Training Configuration					
Total timesteps: 1,000,000			Environment: MiniGrid		
Random seeds: 10 [0-9]					

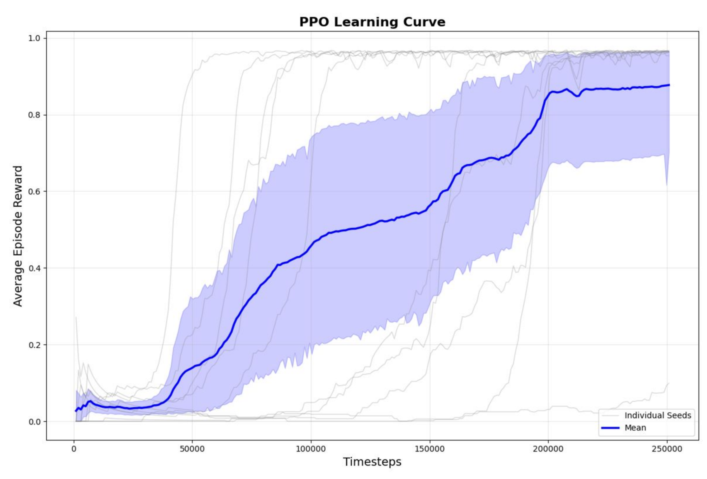
RND Results



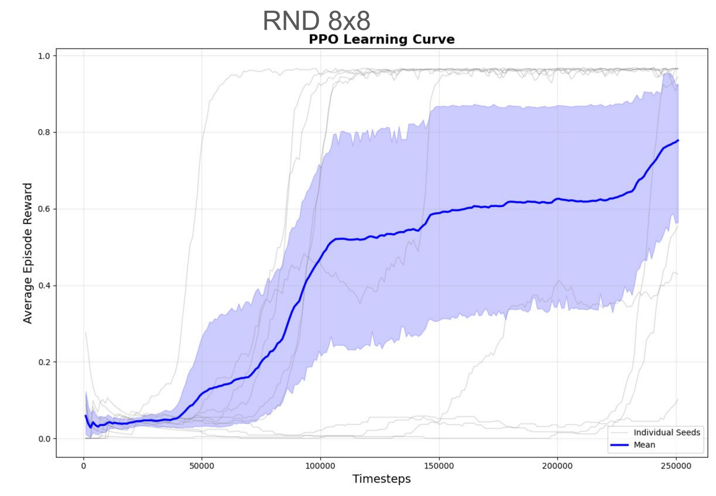
baseline 8x8



RND 8x8



baseline 6x6



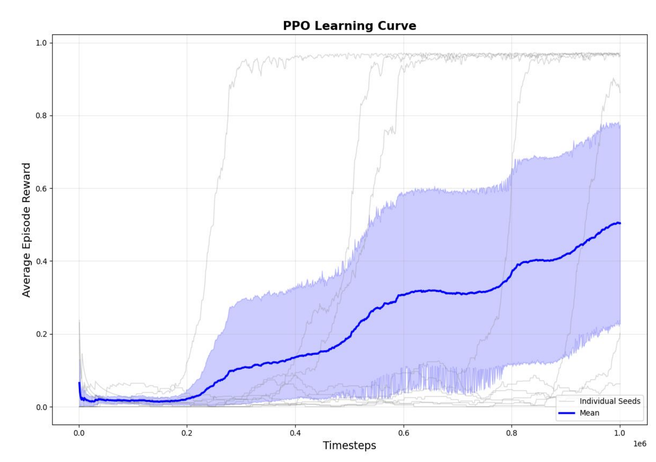
RND 6x6

→ Main Finding:
- ◆ RND's effectiveness depends on task complexity

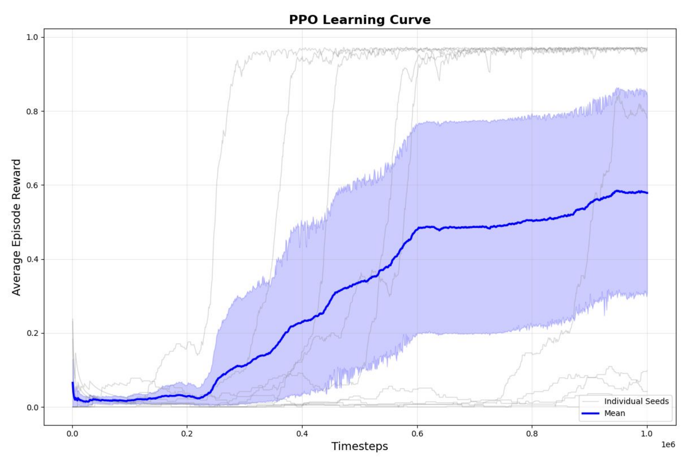
→ What Worked
- ❖ RND works on simple task like Doorkey 6x6.
- ❖ Vanilla PPO worked better than PPO+RND setup.

SIL Results

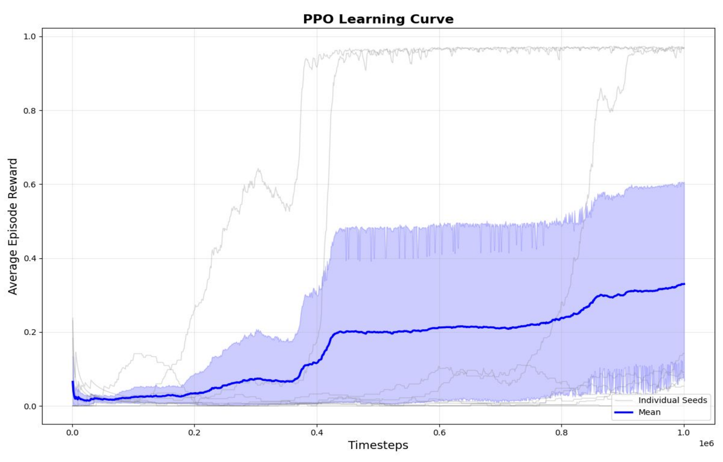
- Update Frequency: 5 -> Optimal Frequency 3 -> Too slow, hinder the performance
- Success Threshold:



0.7: Diverse but suboptimal

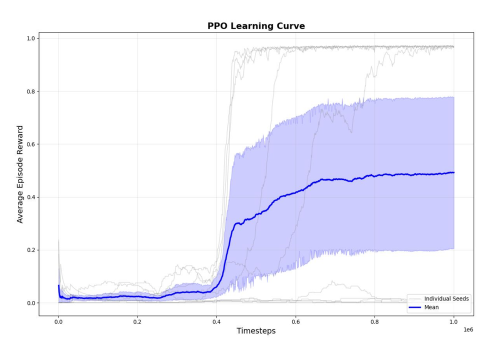


0.8: Best balance

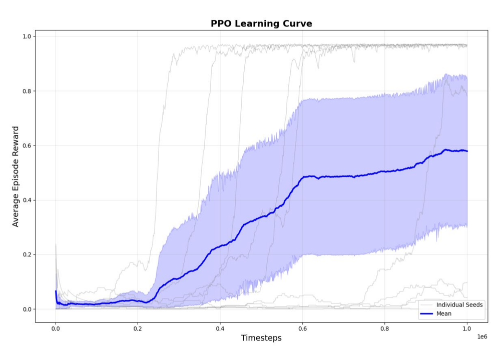


0.9: Too restrictive

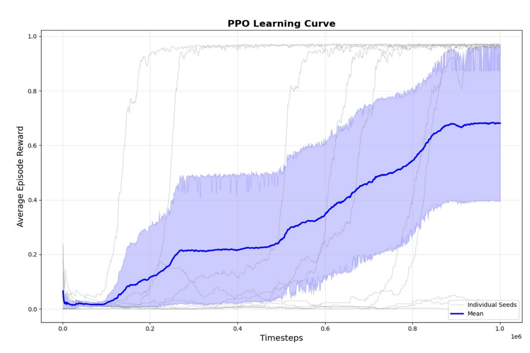
- Alpha



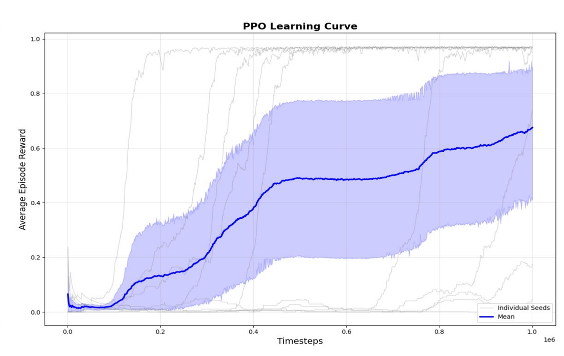
0.4: More exploration



0.6: Better exploitation



0.8:Higher alpha wins



1.0: Pure sampling

→ Main Finding:
- ◆ PPO+SIL improves sparse reward learning
- ◆ SIL prefer learning from only successful episodes
- ◆ PPO prefere exploitation on sparse reward environments

→ What Worked:
- ◆ Adaptive threshold prevents low-quality buffer
- ◆ Prioritized replay focuses learning
- ◆ Maintains PPO stability
- ◆ Improving sample efficiency

5 Future Works

- Extensions include adaptive prioritization scheme
- integration with other exploration driven methods
- Further investigate performance and improve when successful experiences become repetitive.
- Additional hyperparameter tuning Entropy coef
- SIL with Entropy loss

6 References

[1] Oh, Junhyuk, et al. "Self-imitation learning." *International conference on machine learning*. PMLR, 2018.
[2] Eimer, Theresa, Marius Lindauer, and Roberta Raileanu. "Hyperparameters in reinforcement learning and how to tune them." *International conference on machine learning*. PMLR, 2023.
[3] Yuri Burda and Harrison Edwards and Amos J. Storkey and Oleg Klimov "Exploration by Random Network Distillation" CoRR.2018.