On Learning Intrinsic Rewards for Policy Gradient Methods

Zeyu Zheng, Junhyuk Oh, Satinder Singh

NeurIPS 2018

Daejin Jo

Contents

- Introduction
- Background
- LIRPG
- Implementation
- Experiments
- Discussion

Introduction

- Designing *reward functions* that help an RL agent efficiently learn behavior is challenging.

- In most cases there may not be any clear of reward function.
  1. What is the reward for human-satisfaction in human-interaction systems (e.g. dialog system)?
  2. What is the reward when the tasks contains multiple criteria such as minimizing energy consumption & maximizing throughput?

Introduction

- In many complex real-world tasks an RL agent is simply not going to learn an optimal policy due to various limitations on the agent-environment interaction.

- Existing solutions
  1. shaping reward functions that are less sparse than an original reward function
  2. exploration bonuses (e.g. count-based reward bonuses)

  → These methods can sometimes lead to unexpected and undesired behaviors (Reward hacking).

Introduction

- In this paper, the derivation of a new policy gradient based method for learning parametric intrinsic rewards that optimizes w.r.t task-specifying(extrinsic) reward function is provided.

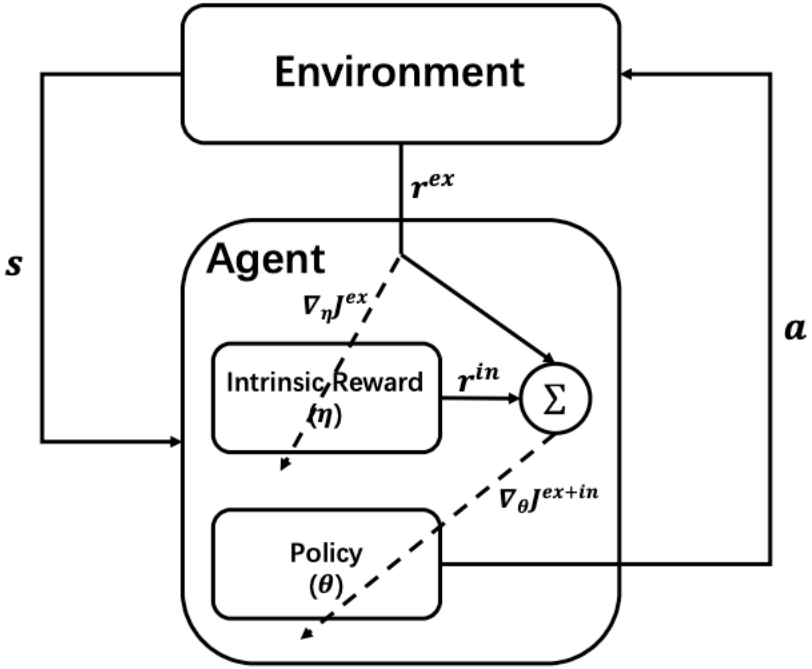
Background: Policy Gradient based RL

- Objective: $\text{maximize}\, J(\theta) = E_{s_t \sim T(s_{t-1}, a_{t-1}),\, a_t \sim \pi_\theta(s_t)}[\sum_{t=0}^{\infty} \gamma^t r_t]$
  - T: transition dynamics

- Policy Gradient
  - $\nabla_\theta J(\theta) = E_\theta[G(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t)]$
  - $G(s_t, a_t) = \sum_{i=t}^{\infty} \gamma^{i-t} r_i$

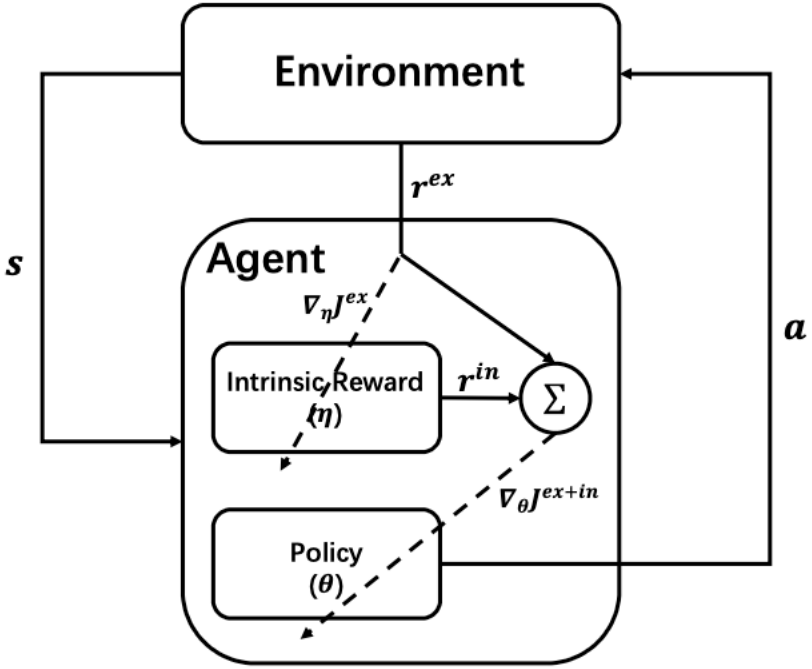
LIRPG: Learning Intrinsic Rewards for Policy Gradient

- Overview



LIRPG: Learning Intrinsic Rewards for Policy Gradient

- Notation



- θ: policy parameters
- η: intrinsic reward parameters
- r^{ex}: extrinsic reward from the environment
- $r_\eta^{in} = r_\eta^{in}(s, a)$: intrinsic reward estimated by η
- $G^{ex}(s_t, a_t) = \sum_{i=t}^{\infty} \gamma^{i-t} r_i^{ex}$
- $G^{in}(s_t, a_t) = \sum_{i=t}^{\infty} \gamma^{t-i} r_\eta^{in}(s_i, a_i)$
- $G^{ex+in}(s_t, a_t) = \sum_{i=t}^{\infty} \gamma^{i-t} (r_i^{ex} + \lambda r_\eta^{in}(s_i, a_i))$
- $J^{ex} = E_\theta[\sum_{t=0}^{\infty} \gamma^t r_t^{ex}]$
- $J^{in} = E_\theta[\sum_{t=0}^{\infty} \gamma^t r_\eta^{in}(s_t, a_t)]$
- $J^{ex+in} = E_\theta[\sum_{t=0}^{\infty} \gamma^t (r_t^{ex} + \lambda r_\eta^{in}(s_t, a_t)]$
- λ: relative weight of intrinsic reward.

LIRPG: Learning Intrinsic Rewards for Policy Gradient

- Updating Policy Parameters θ

$$\begin{aligned}\theta' &= \theta + \alpha \nabla_\theta J^{ex+in}(\theta) \\ &\approx \theta + \alpha G^{ex+in}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t | s_t).\end{aligned}$$

- Updating Intrinsic Reward Parameters η

$$\nabla_\eta J^{ex} = \nabla_{\theta'} J^{ex} \nabla_\eta \theta'$$

LIRPG: Learning Intrinsic Rewards for Policy Gradient

$$\nabla_\eta J^{ex} = \nabla_{\theta'} J^{ex} \nabla_\eta \theta'$$

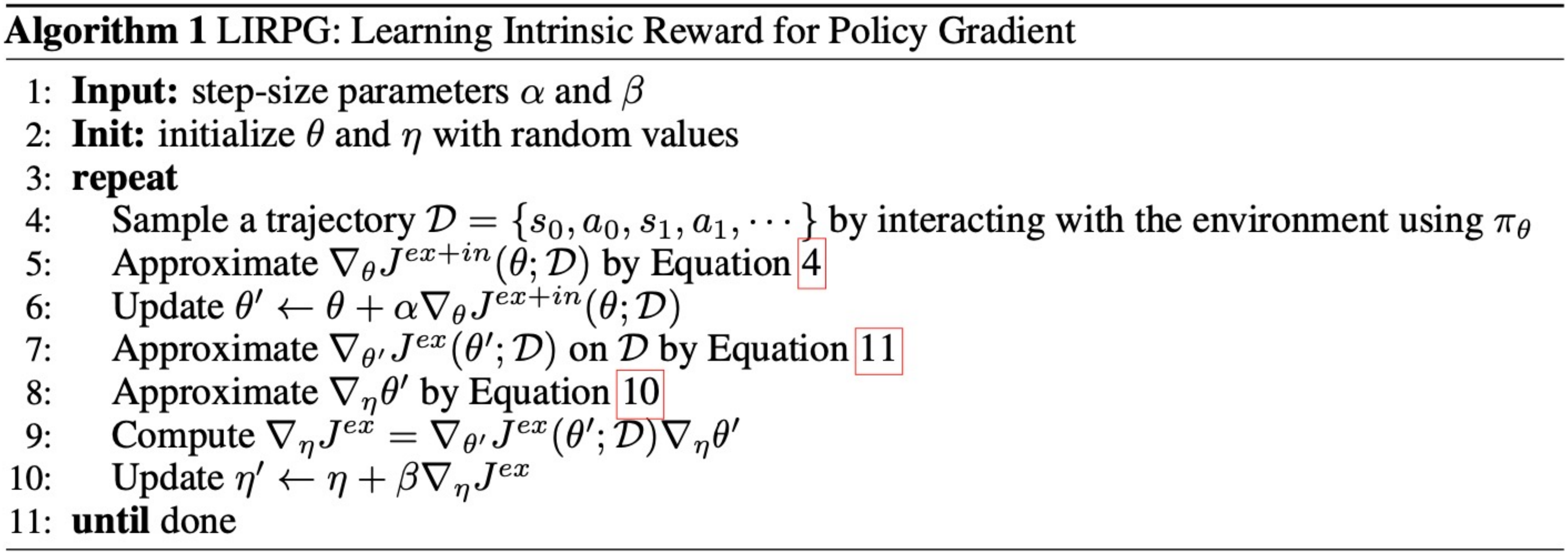
- $\nabla_{\theta'} J^{ex} \approx G^{ex}(s_t, a_t) \nabla_{\theta'} \log \pi_{\theta'}(a_t | s_t)$
- $\nabla_\eta \theta' = \nabla_\eta (\alpha \lambda G^{in}(s_t, a_t) \nabla_\theta \log \pi_\theta (a_t | s_t))$

LIRPG: Learning Intrinsic Rewards for Policy Gradient

$$\nabla_\eta J^{ex} = \nabla_{\theta'} J^{ex} \nabla_\eta \theta'$$

- $\nabla_{\theta'} J^{ex} \approx G^{ex}(s_t, a_t) \nabla_{\theta'} \log \pi_{\theta'}(a_t | s_t)$
- $\nabla_\eta \theta' = \nabla_\eta \big(\alpha \lambda G^{in}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t | s_t) \big)$

- Full derivation of $\begin{aligned}\nabla_\eta \theta' &= \nabla_\eta \left(\theta + \alpha G^{ex+in}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t | s_t) \right) \\ &= \nabla_\eta \left(\alpha G^{ex+in}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t | s_t) \right) \\ &= \nabla_\eta \left(\alpha \lambda G^{in}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t | s_t) \right) \\ &= \alpha \lambda \sum_{i=t}^{\infty} \gamma^{i-t} \nabla_\eta r_\eta^{in}(s_i, a_i) \nabla_\theta \log \pi_\theta(a_t | s_t).\end{aligned}$



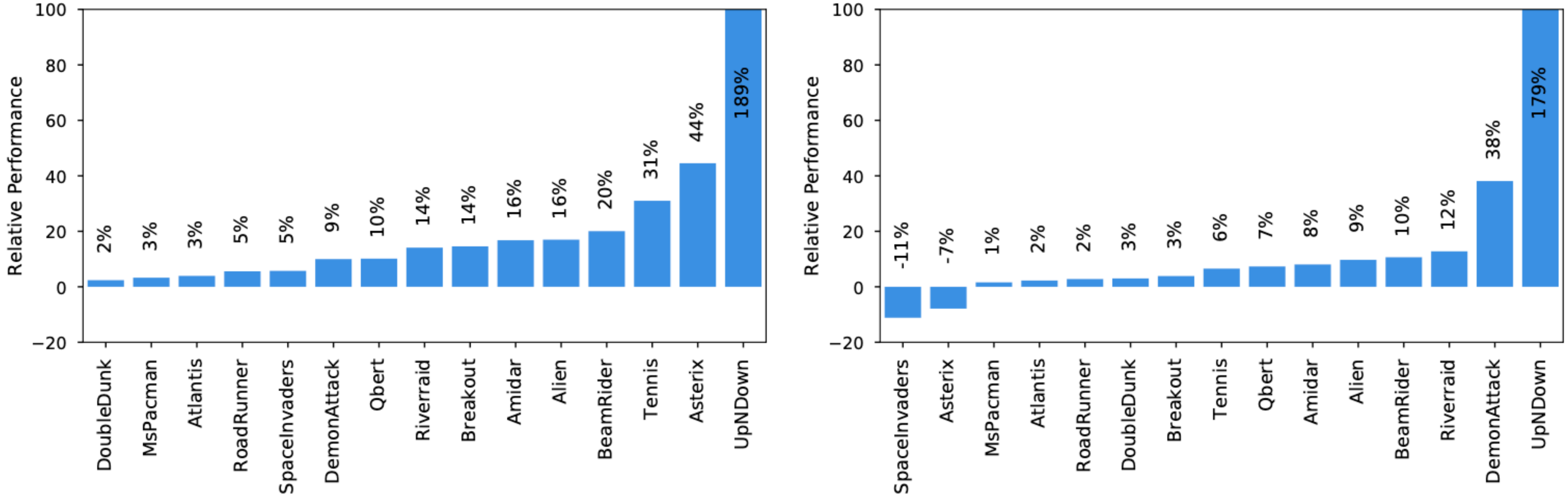
Algorithm 1 LIRPG: Learning Intrinsic Reward for Policy Gradient

```
**Input:** step-size parameters $\alpha$ and $\beta$
**Init:** initialize $\theta$ and $\eta$ with random values
**repeat**
    Sample a trajectory $\mathcal{D} = \{s_0, a_0, s_1, a_1, \cdots\}$ by interacting with the environment using $\pi_\theta$
    Approximate $\nabla_\theta J^{ex+in}(\theta; \mathcal{D})$ by Equation 4
    Update $\theta' \leftarrow \theta + \alpha \nabla_\theta J^{ex+in}(\theta; \mathcal{D})$
    Approximate $\nabla_{\theta'} J^{ex}(\theta'; \mathcal{D})$ on $\mathcal{D}$ by Equation 11
    Approximate $\nabla_\eta \theta'$ by Equation 10
    Compute $\nabla_\eta J^{ex} = \nabla_{\theta'} J^{ex}(\theta'; \mathcal{D}) \nabla_\eta \theta'$
    Update $\eta' \leftarrow \eta + \beta \nabla_\eta J^{ex}$
**until** done
```

Implementation of LIRPG

https://github.com/Hwhitetooth/lirpg/blob/master/baselines/a2c/a2c.py

Experiments on Atari Games

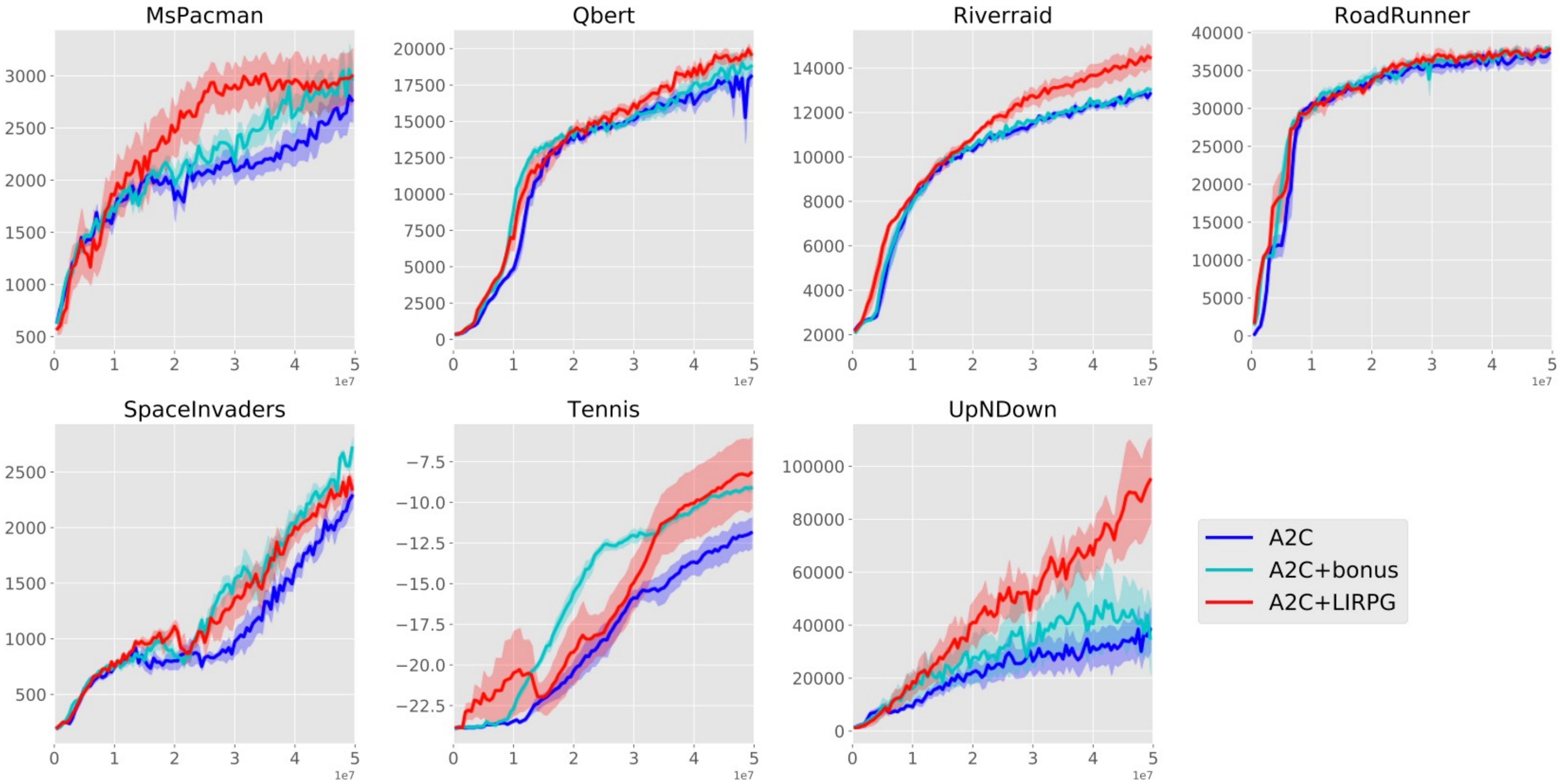


(a)

(b)

Figure 2: (a) Improvements of LIRPG augmented agents over A2C baseline agents. (b) Improvements of LIRPG augmented agents over live-bonus augmented A2C baseline agents. In both figures, the columns correspond to different games labeled on the x-axes and the y-axes show human score normalized improvements.

Experiments on Atari Games



Experiments on Delayed Mujoco

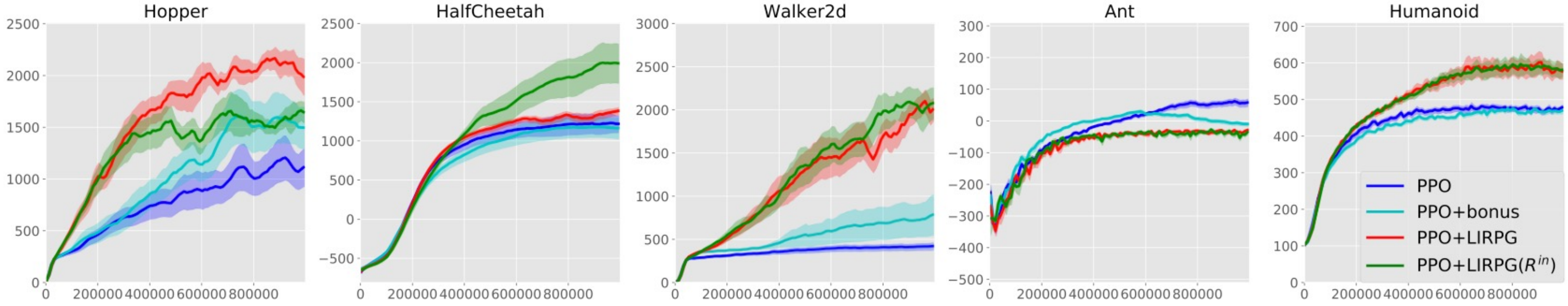


Figure 4: The x-axis is time steps during learning. The y-axis is the average reward over the last 100 training episodes. The deep blue curves are for the baseline PPO architecture. The light blue curves are for the PPO-bonus baseline. The red curves are for our LIRPG based augmented architecture. The green curves are for our LIRPG architecture in which the policy module was trained with only intrinsic rewards. The dark curves are the average of 10 runs with different random seeds. The shaded area shows the standard errors of 10 runs.

Discussion

- LIRPG can be helpful for learning task-relevant motivation on practical environment.
- However, the task-relevant motivation may be hard to learn in hard exploration problems.
- Somewhat difficult bi-level optimization at scale