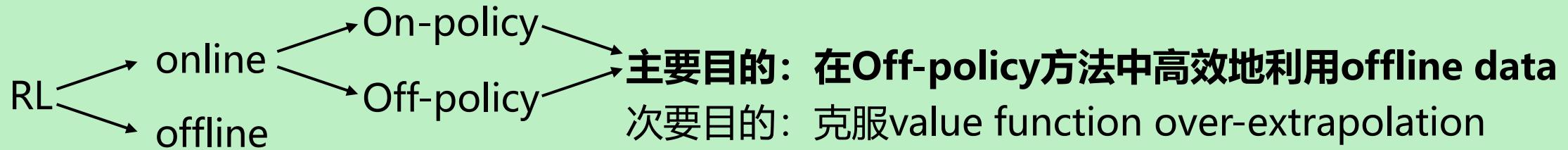




# Abstract



**title:** Efficient Online Reinforcement Learning with Offline Data  
**contribution:** <sup>1</sup>University of Oxford <sup>2</sup>UC Berkeley



## 过去的方法

Offline RL Pre-training  
+  
Online fine-tuning

## 缺陷

超参数、额外的计算

利用先验数据限制  
探索过程中的动作

磨灭了RL的本质

## 本文方法 基于SAC算法

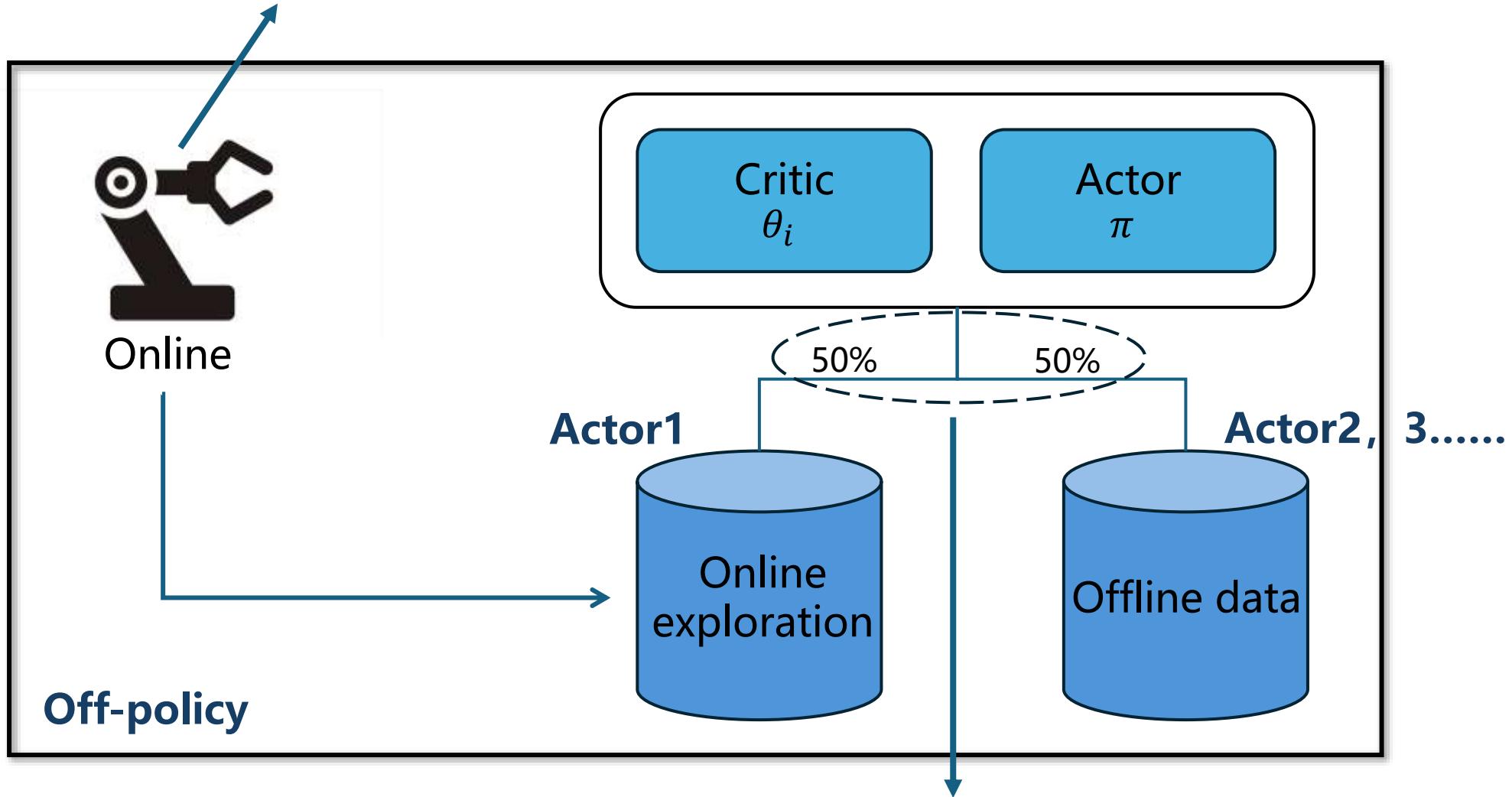
1. 将offline data与replay buffer融合
2. LayerNormalization防止过估计
3. 高UTD ratio + random ensemble distillation抑制过拟合

1. Double Q learning
2. 加入熵项
3. 2~3层的网络设置



# ① Incorporate Offline Data

这里的采样策略与训练的目标策略一致，是一直更新的



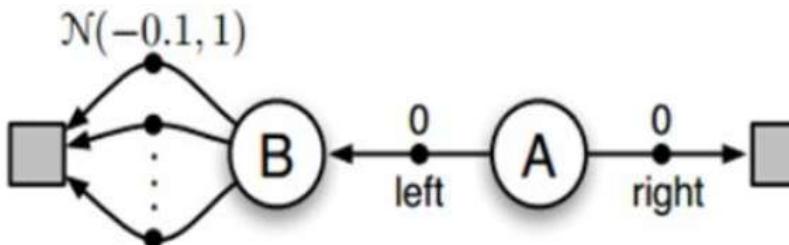
数据采样的actor不同，但环境动力学 $P(s'|s, a)$ 和reward与policy无关

# ②Layer Normalization

## Critic divergence

造成发散的三种机制

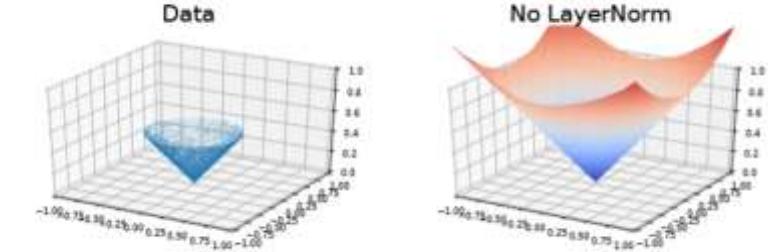
max



bootstrapping

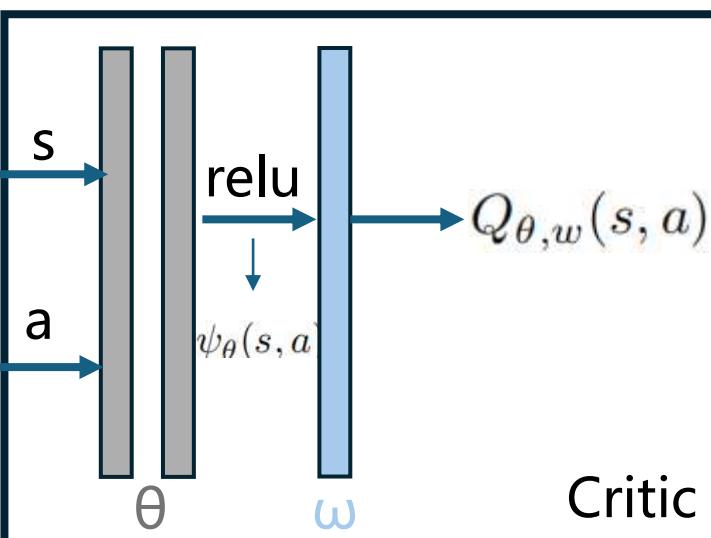


外推



主要原因

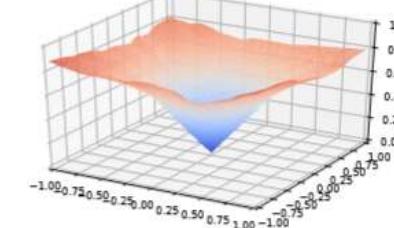
目标：需要找到一个不限制探索范围，又能防止Q值外推的方法



relu( $\psi_\theta(s, a)$ ) 是归一化的中间变量

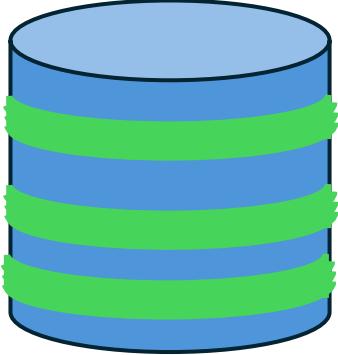
$$\begin{aligned}\|Q_{\theta, w}(s, a)\| &= \|w^T \text{relu}(\psi_\theta(s, a))\| \\ &\leq \|w\| \|\text{relu}(\psi_\theta(s, a))\| \leq \|w\| \|\psi(s, a)\| \\ &\leq \|w\|\end{aligned}$$

With LayerNorm





## ③Sample Efficient RL——REDQ



UTD ratio : 环境智能体进行更新的轮数和实际与环境交互轮数的比值

**从一个大总量里随机抽取一部分，允许交叉**

**目标：使UTD ratio在合理区间内尽可能大**

小的话样本利用率低、大的话容易**过拟合**

### 解决过拟合的已有研究

**1.** L2 normalization      降低输出的幅度，减小每次更新的影响

**2.** Dropout      每次计算随机使一部分神经元失灵，遗忘机制

**3.** Random ensemble distillation      设定N个不同的critic函数，每次随即在里面取两个critic取较小值



# 伪代码流程解析

**加入熵项以平衡探索性**

$$\max_{\pi} \mathbb{E}_{s \sim \rho^{\pi}, a \sim \pi} \left[ \sum_{t=0}^{\infty} \gamma^t (r_t + \alpha \mathcal{H}(\pi(a|s))) \right]$$

**网络设置为2~3层**

---

## Algorithm 1 Online RL with Offline Data (RLPD)

---

```

1: Select LayerNorm, Large Ensemble Size  $E$ , Gradient
   Steps  $G$ , and architecture.
2: Randomly initialize Critic  $\theta_i$  (set targets  $\theta'_i = \theta_i$ ) for
    $i = 1, 2, \dots, E$  and Actor  $\phi$  parameters. Select dis-
   count  $\gamma$ , temperature  $\alpha$  and critic EMA weight  $\rho$ .
3: Determine number of Critic targets to subset  $Z \in \{1, 2\}$ 
4: Initialize empty replay buffer  $\mathcal{R}$ 
5: Initialize buffer  $\mathcal{D}$  with offline data
6: while True do
7:   Receive initial observation state  $s_0$ 
8:   for  $t = 0, T$  do
9:     Take action  $a_t \sim \pi_{\phi}(\cdot|s_t)$ 
10:    Store transition  $(s_t, a_t, r_t, s_{t+1})$  in  $\mathcal{R}$ 
11:    for  $g = 1, G$  do
12:      Sample minibatch  $b_R$  of  $\frac{N}{2}$  from  $\mathcal{R}$ 

```

13:  
14:  
15:  
16:

Sample minibatch  $b_D$  of  $\frac{N}{2}$  from  $\mathcal{D}$   
Combine  $b_R$  and  $b_D$  to form batch  $b$  of size  $N$   
Sample set  $Z$  of  $Z$  indices from  $\{1, 2, \dots, E\}$   
With  $b$ , set

$$y = r + \gamma \left( \min_{i \in Z} Q_{\theta'_i}(s', \tilde{a}') \right), \quad \tilde{a}' \sim \pi_{\phi}(\cdot|s')$$

Add entropy term  $y = y + \gamma \alpha \log \pi_{\phi}(\tilde{a}'|s')$   
**for**  $i = 1, E$  **do**  
    Update  $\theta_i$  minimizing loss:

$$L = \frac{1}{N} \sum_i (y - Q_{\theta_i}(s, a))^2$$

**end for**  
    Update target networks  $\theta'_i \leftarrow \rho \theta'_i + (1 - \rho) \theta_i$

**end for**  
    With  $b$ , update  $\phi$  maximizing objective:

$$\frac{1}{E} \sum_{i=1}^E Q_{\theta_i}(s, \tilde{a}) - \alpha \log \pi_{\phi}(\tilde{a}|s), \quad \tilde{a} \sim \pi_{\phi}(\cdot|s)$$

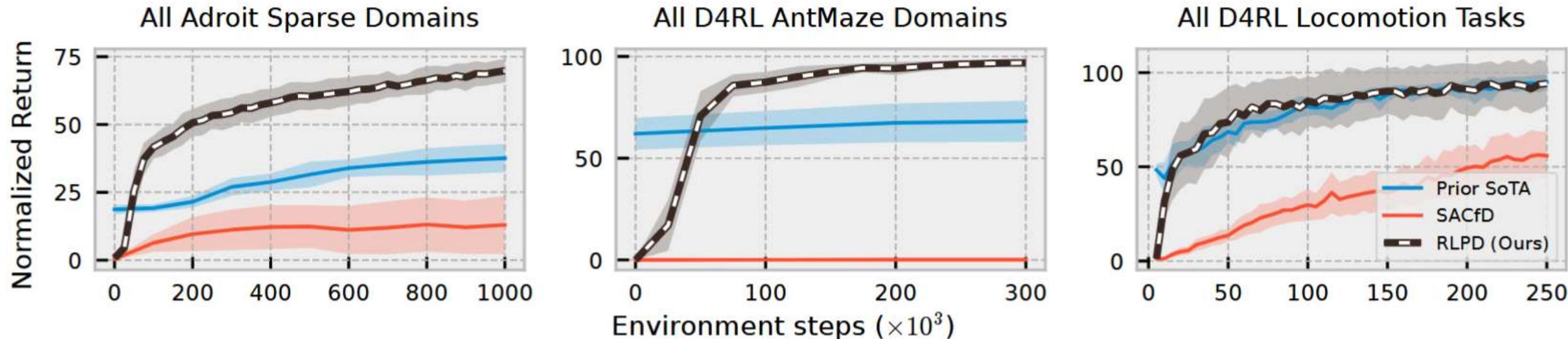
24: **end for**  
25: **end while**

**缓更新目标网络**

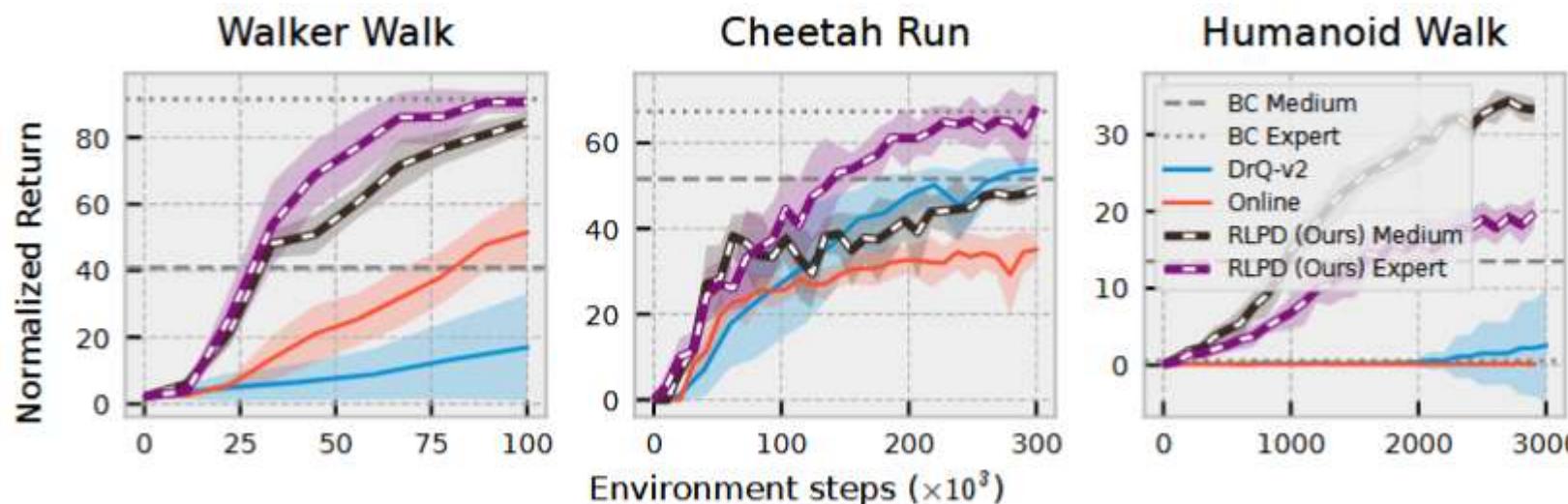


# Experiment

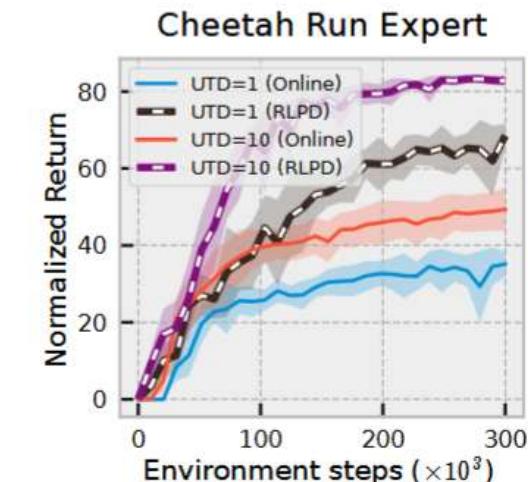
## 与SOTA在操作数据集对比



## 与SOTA在纯视觉数据集对比



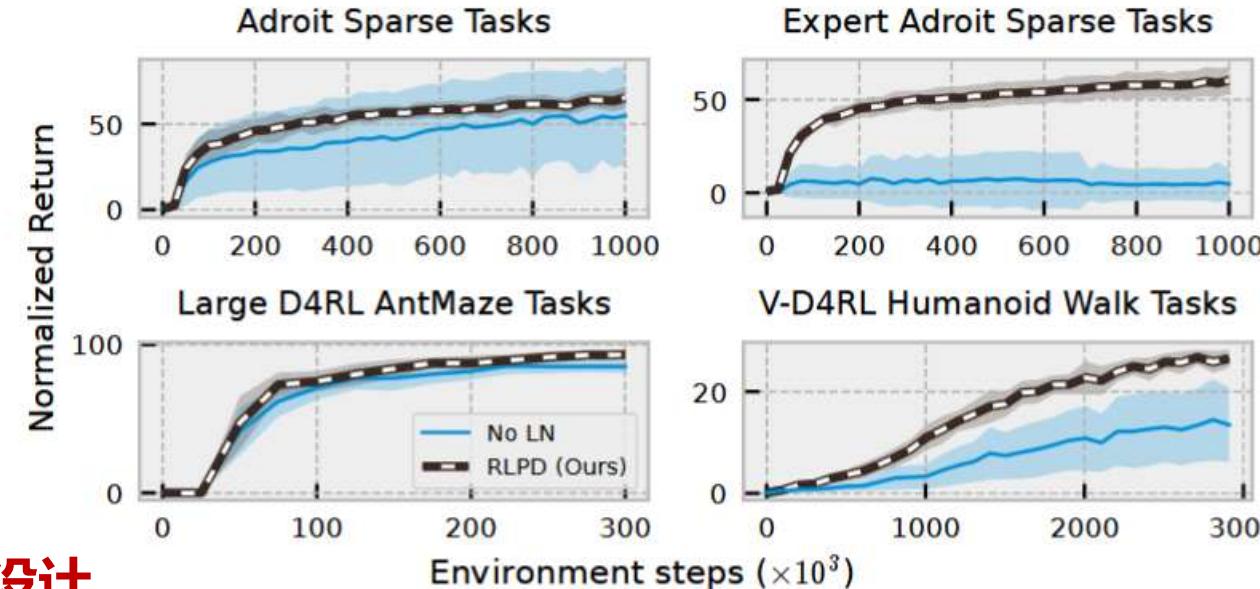
## UTD ratio不同的对比



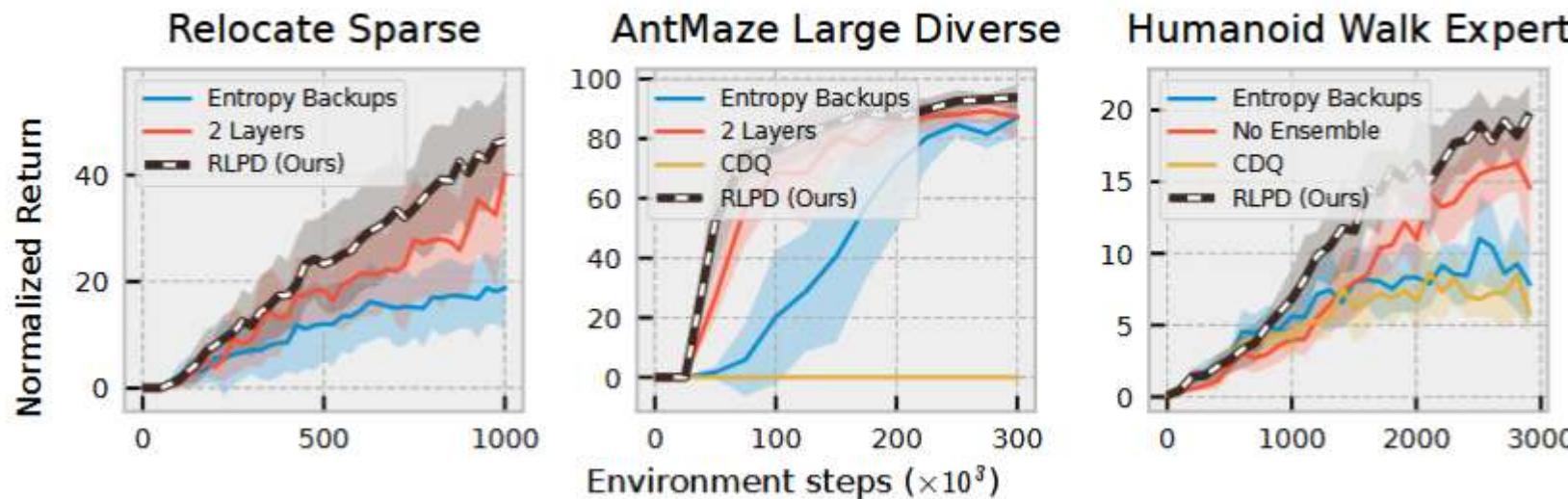


# Ablation

## 有无LayerNormalization



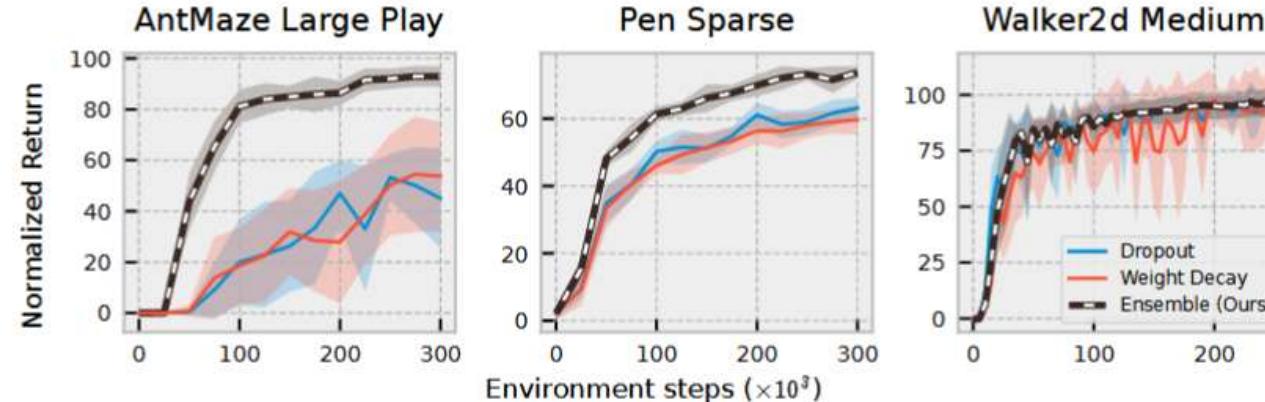
## 一些针对任务的设计



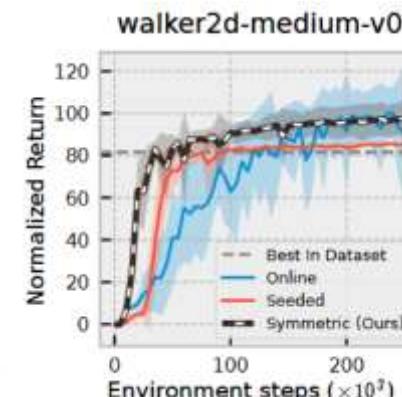
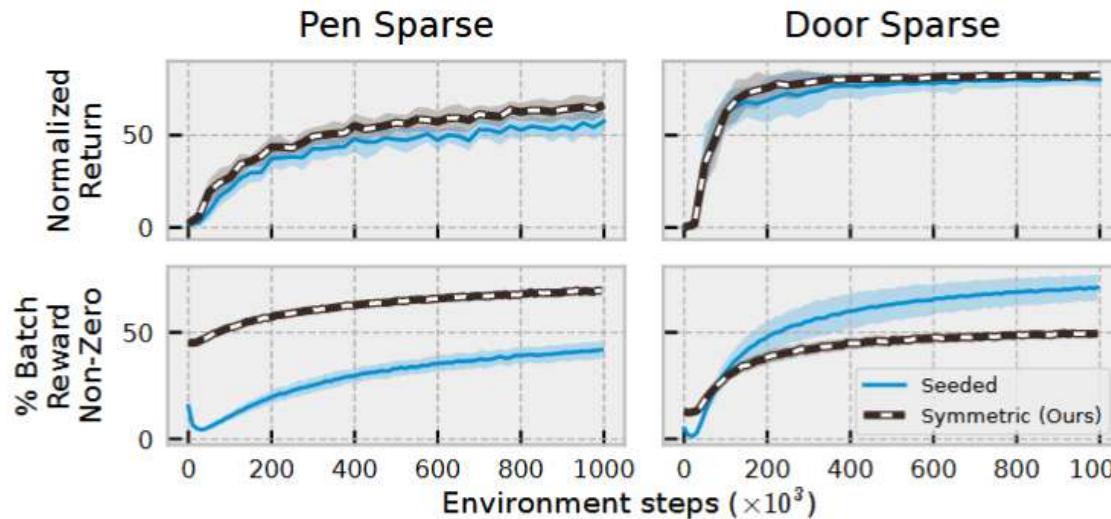
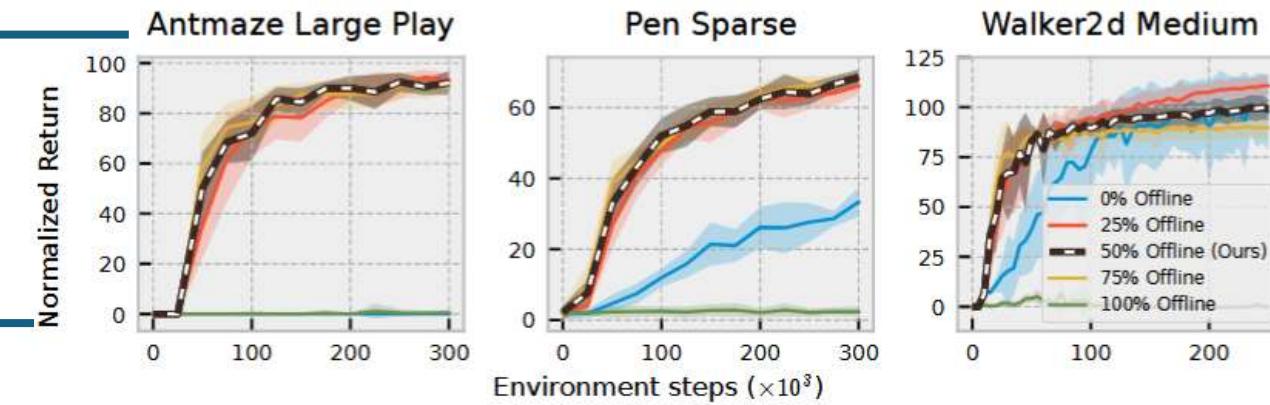


# Ablation

## REDQ的作用



## 数据比例的分析



## 离线数据的加入方式对比