Accelerating Reinforcement Learning with Learned Skill Priors

Feng Xu

2021/1/19

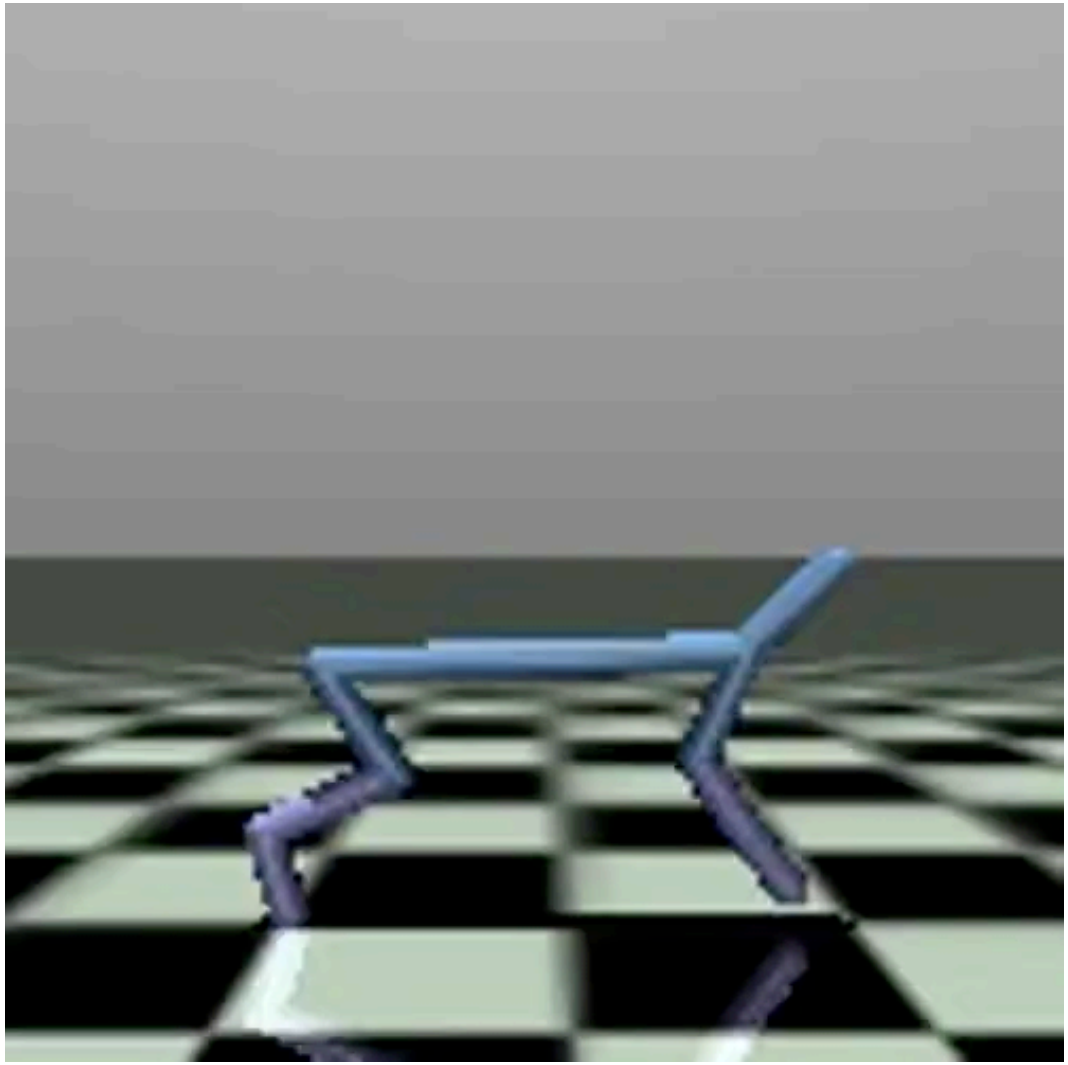
Outline

- Background
- Algorithm
- Experiment

Meta-RL

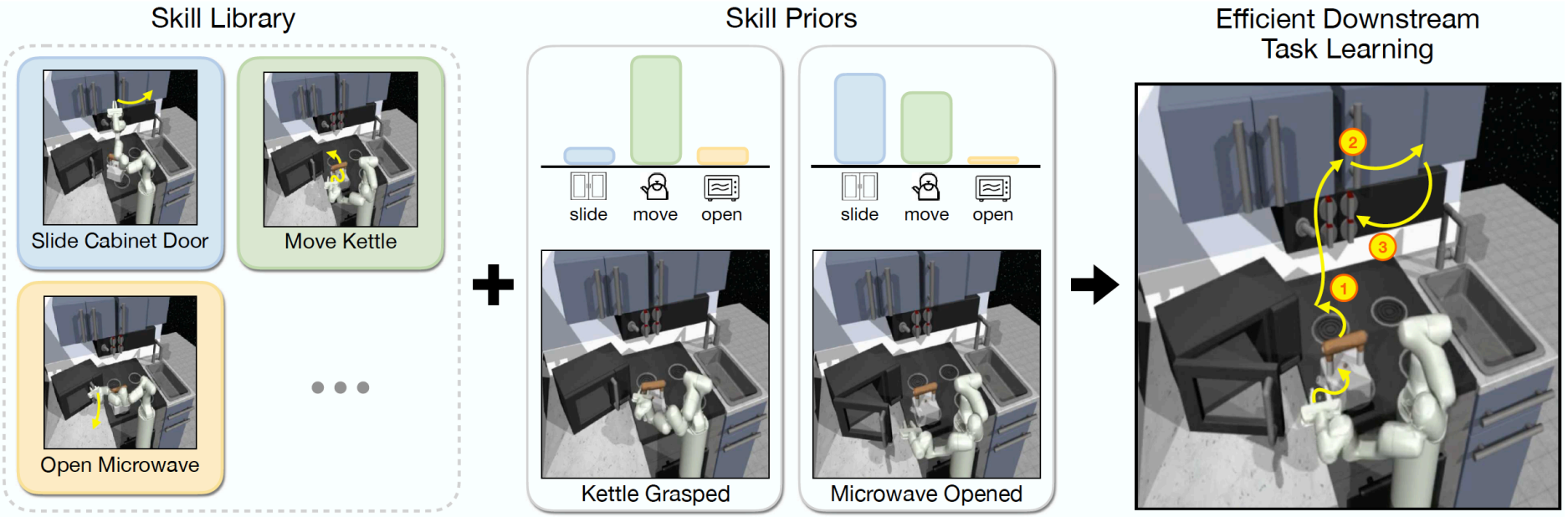
- Learning to Learn
  - Leverage prior experience
- Previous methods:
  - Black Box based: Pearl, RL^2
  - Optimization based: MAML and its variants
- Instant adaptation

What's wrong with conventional RL algorithms



Motivation

- Leverage prior experience to accelerate the learning process of downstream tasks



Prior-Guided Skill Transfer

- Extract skill embedding and prior from offline data
  - Learning continuous skill embedding and skill prior
- Prior guided learning of downstream tasks with a hierarchical policy
  - Skill prior regularized reinforcement learning

Formulation

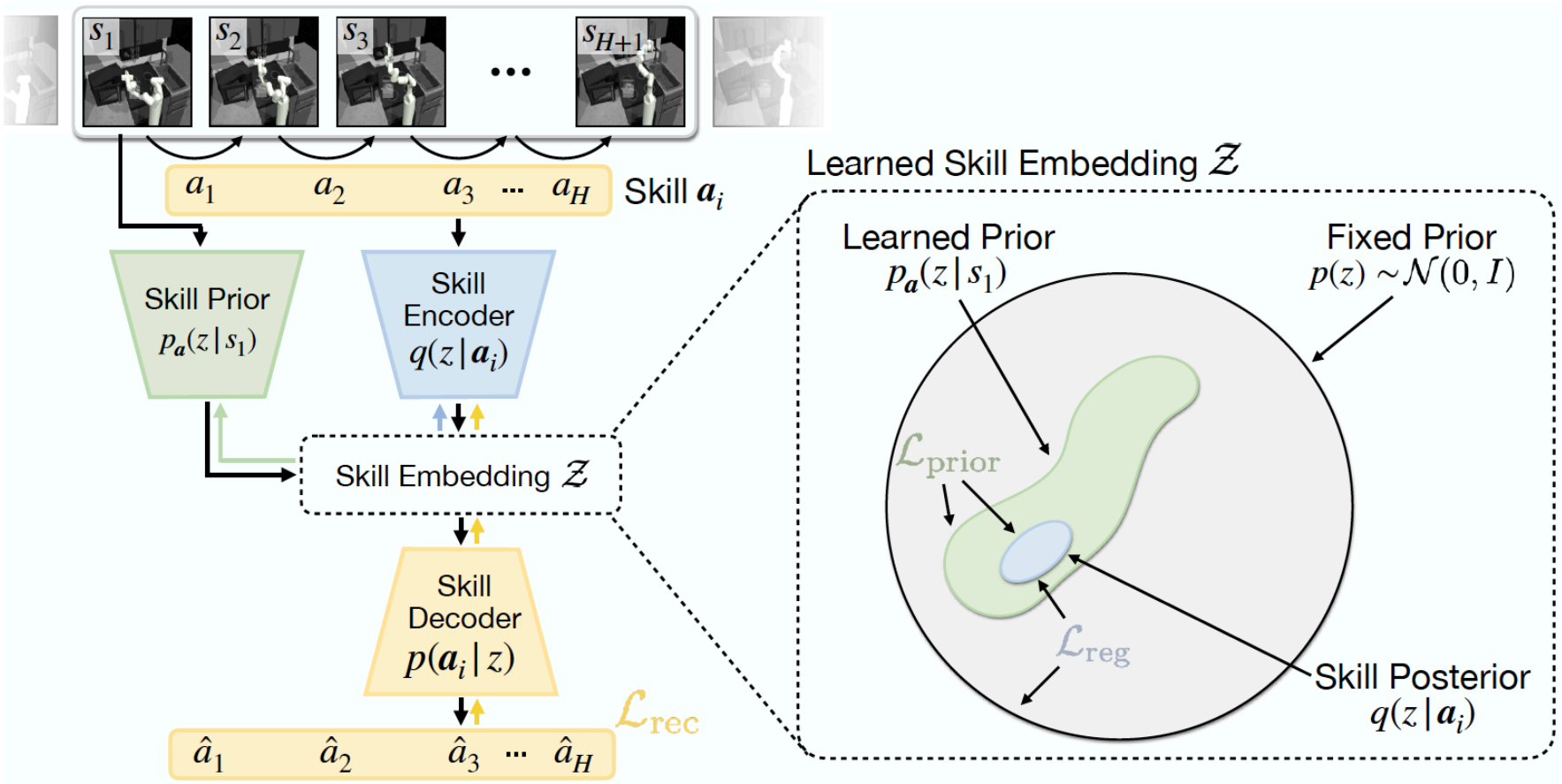
- MDP $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, R, \rho, \gamma\}$
- Pre-recorded experience of trajectories

$$\tau_i = \{(s_0, a_0), \ldots, (s_{T_i}, a_{T_i})\}.$$

- Learn a policy that maximize expected return of T horizon

$$J(\theta) = \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]$$

Overall Structure



- A skill is defined as a sequence of actions with fixed horizon H $\{a_t^i, \ldots, a_{t+H-1}^i\}$
- Optimization objection(ELBO)

$$\log p(\boldsymbol{a}_i) \geq \mathbb{E}_q \Big[\underbrace{\log p(\boldsymbol{a}_i|z)}_{\text{reconstruction}} - \beta \big(\underbrace{\log q(z|\boldsymbol{a}_i) - \log p(z)}_{\text{regularization}} \big) \Big]$$

- Output: skill encoder $q(z|\boldsymbol{a}_i)$ and decoder $p(\boldsymbol{a}_i|z)$

- To adjust to environment and task -> skill prior

$$p_{\boldsymbol{a}}(z|\cdot) \qquad p_{\boldsymbol{a}}(z|s_t)$$

- Optimization objective:
  - max $\mathbb{E}_{(s,\boldsymbol{a}_i)\sim\mathcal{D}} D_{\mathrm{KL}}\left(q(z|\boldsymbol{a}_i), p_{\boldsymbol{a}}(z|s_t)\right)$

Skill Prior Regularized Reinforcement Learning

- A hierarchival policy learning scheme:
  - Skill embedding space as action space
  - Policy $\pi_\theta(z|s_t)$ -> $\{a_t^i, \ldots, a_{t+H-1}^i\} \sim p(\boldsymbol{a}_i|z)$
- Equivalent to solving a MDP: $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, R, \rho, \gamma\}$
  - Replace the action space with skill space
  - H-step rewards
  - H-step transitions

Guided Learninng: Incorporate with Maximum Entropy RL

- Naïve implementation

$$J(\theta) = \mathbb{E}_{\pi}\left[\sum_{t=1}^{T} \gamma^t r(s_t, a_t) + \alpha \mathcal{H}\big(\pi(a_t|s_t)\big)\right]$$

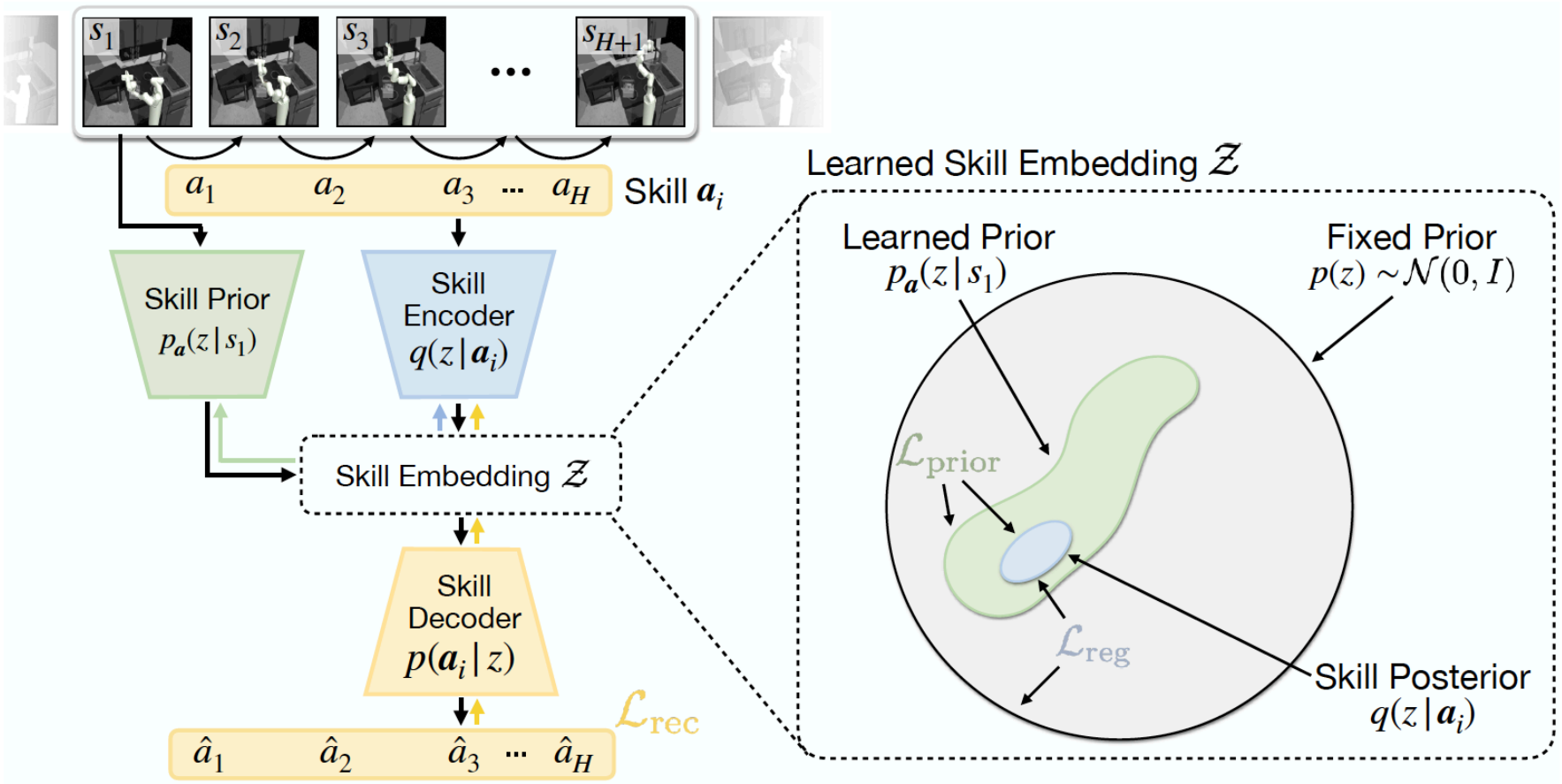
- Equivalent to the negated KL divergence between the policy and a uniform action prior:

$$U(a_t)\text{: } \mathcal{H}(\pi(a_t|s_t)) = -\mathbb{E}_{\pi}\big[\log \pi(a_t|s_t)\big] \propto -D_{\mathrm{KL}}(\pi(a_t|s_t), U(a_t))$$

- Replace the entropy term with negated KL divergence from prior

$$J(\theta) = \mathbb{E}_{\pi}\left[\sum_{t=1}^{T} \tilde{r}(s_t, z_t) - \alpha D_{\mathrm{KL}}\big(\pi(z_t|s_t), p_{\boldsymbol{a}}(z_t|s_t)\big)\right]$$

Overall Structure

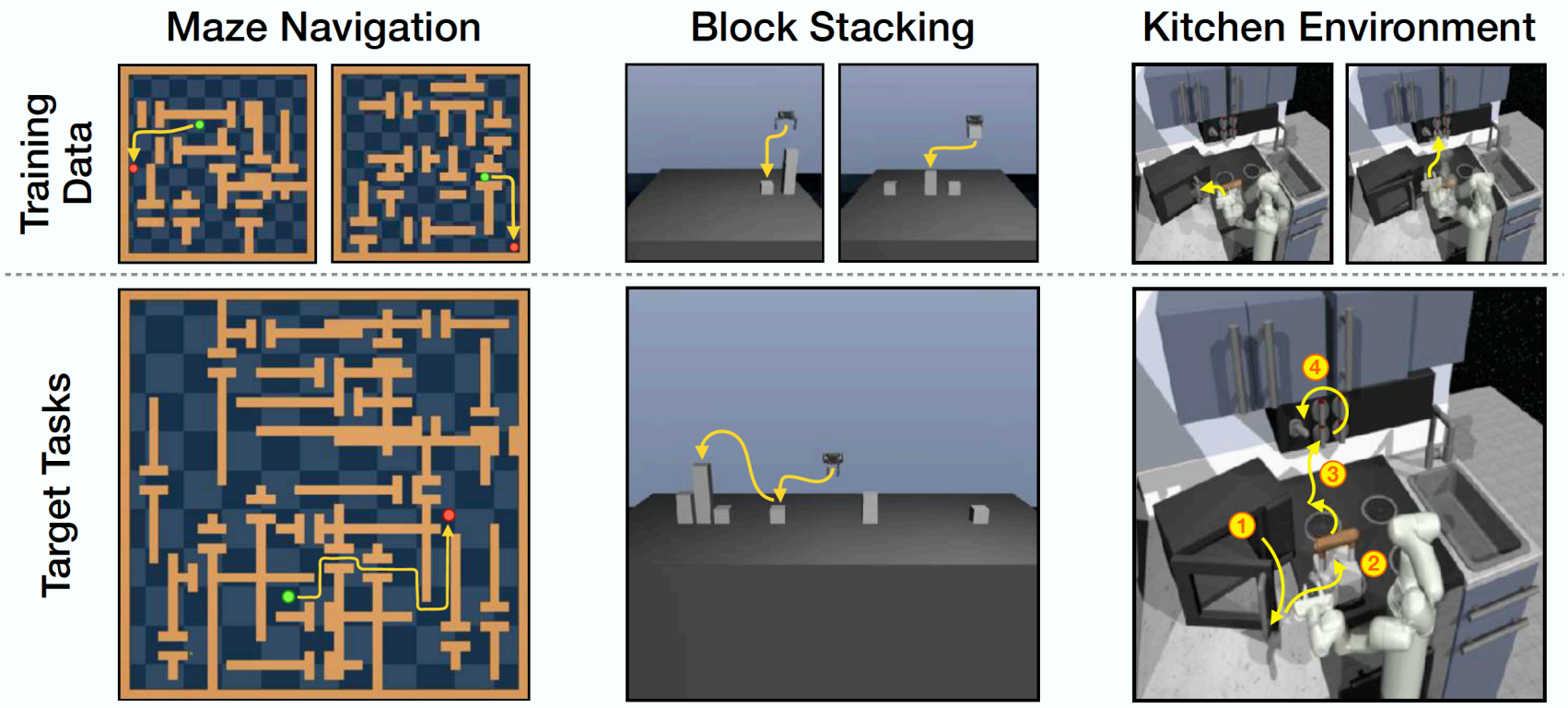


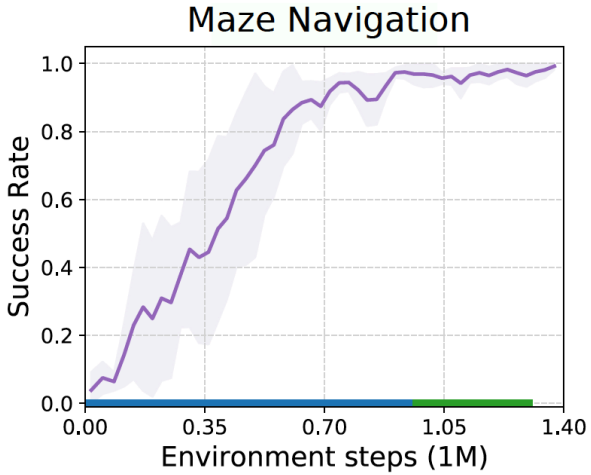
Overall Training Process

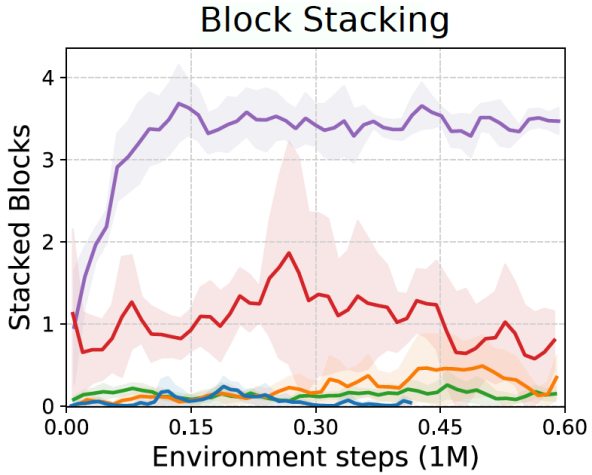
Algorithm 1 SPiRL: Skill-Prior RL

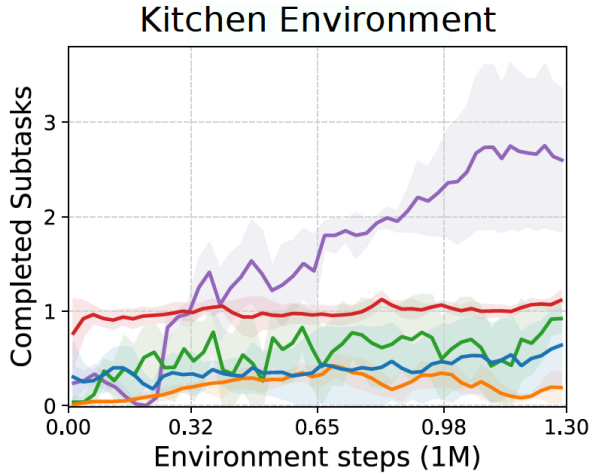
Inputs: H-step reward function $\tilde{r}(s_t, z_t)$, discount γ, target divergence δ, learning rates $\lambda_\pi, \lambda_Q, \lambda_\alpha$, target update rate τ.
Initialize replay buffer $\mathcal{D}$, high-level policy $\pi_\theta(z_t|s_t)$, critic $Q_\phi(s_t, z_t)$, target network $Q_{\bar{\phi}}(s_t, z_t)$
for each iteration **do**
  for every H environment steps **do**
    $z_t \sim \pi(z_t|s_t)$ ▷ sample skill from policy
    $s_{t'} \sim p(s_{t+H}|s_t, z_t)$ ▷ execute skill in environment
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{s_t, z_t, \tilde{r}(s_t, z_t), s_{t'}\}$ ▷ store transition in replay buffer
  for each gradient step **do**
    $\bar{Q} = \tilde{r}(s_t, z_t) + \gamma \left[Q_{\bar{\phi}}(s_{t'}, \pi_\theta(z_{t'}|s_{t'})) - \alpha D_{\text{KL}}\left(\pi_\theta(z_{t'}|s_{t'}), p_{\boldsymbol{a}}(z_{t'}|s_{t'})\right)\right]$ ▷ compute Q-target
    $\theta \leftarrow \theta - \lambda_\pi \nabla_\theta \left[Q_\phi(s_t, \pi_\theta(z_t|s_t)) - \alpha D_{\text{KL}}(\pi_\theta(z_t|s_t), p_{\boldsymbol{a}}(z_t|s_t))\right]$ ▷ update policy weights
    $\phi \leftarrow \phi - \lambda_Q \nabla_\phi \left[\frac{1}{2}\left(Q_\phi(s_t, z_t) - \bar{Q}\right)^2\right]$ ▷ update critic weights
    $\alpha \leftarrow \alpha - \lambda_\alpha \nabla_\alpha \left[\alpha \cdot (D_{\text{KL}}(\pi_\theta(z_t|s_t), p_{\boldsymbol{a}}(z_t|s_t)) - \delta)\right]$ ▷ update alpha
    $\bar{\phi} \leftarrow \tau\phi + (1-\tau)\bar{\phi}$ ▷ update target network weights
return trained policy $\pi_\theta(z_t|s_t)$

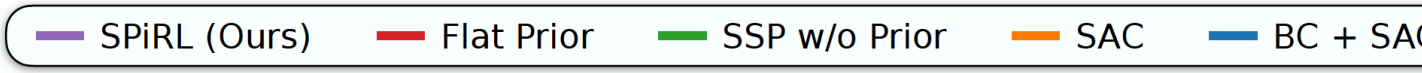
Experiment

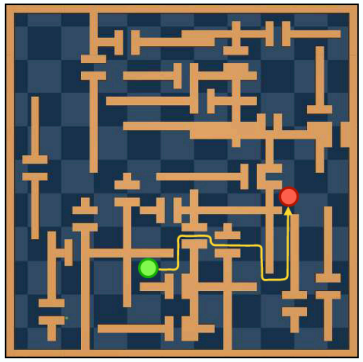

Maze Navigation
Block Stacking
Kitchen Environment
Training Data
Target Tasks
1
2
3
4

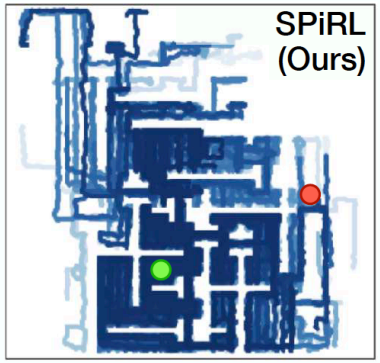

Maze Navigation
Success Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.35
0.70
1.05
1.40
Environment steps (1M)

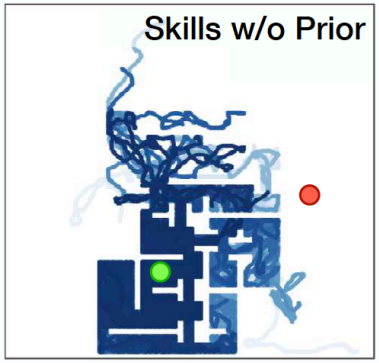

Block Stacking
Stacked Blocks
4
3
2
1
0
0.00
0.15
0.30
0.45
0.60
Environment steps (1M)

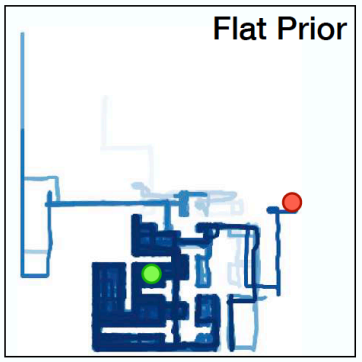

Kitchen Environment
Completed Subtasks
3
2
1
0
0.00
0.32
0.65
0.98
1.30
Environment steps (1M)

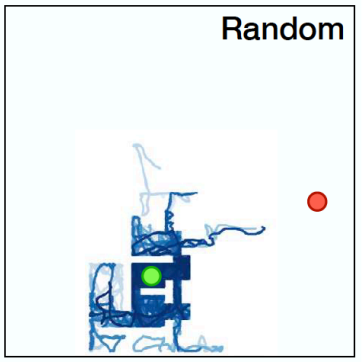

SPiRL (Ours)
Flat Prior
SSP w/o Prior
SAC
BC + SAC




SPiRL
(Ours)


Skills w/o Prior


Flat Prior


Random

http://lamda.nju.edu.cn

Thanks !