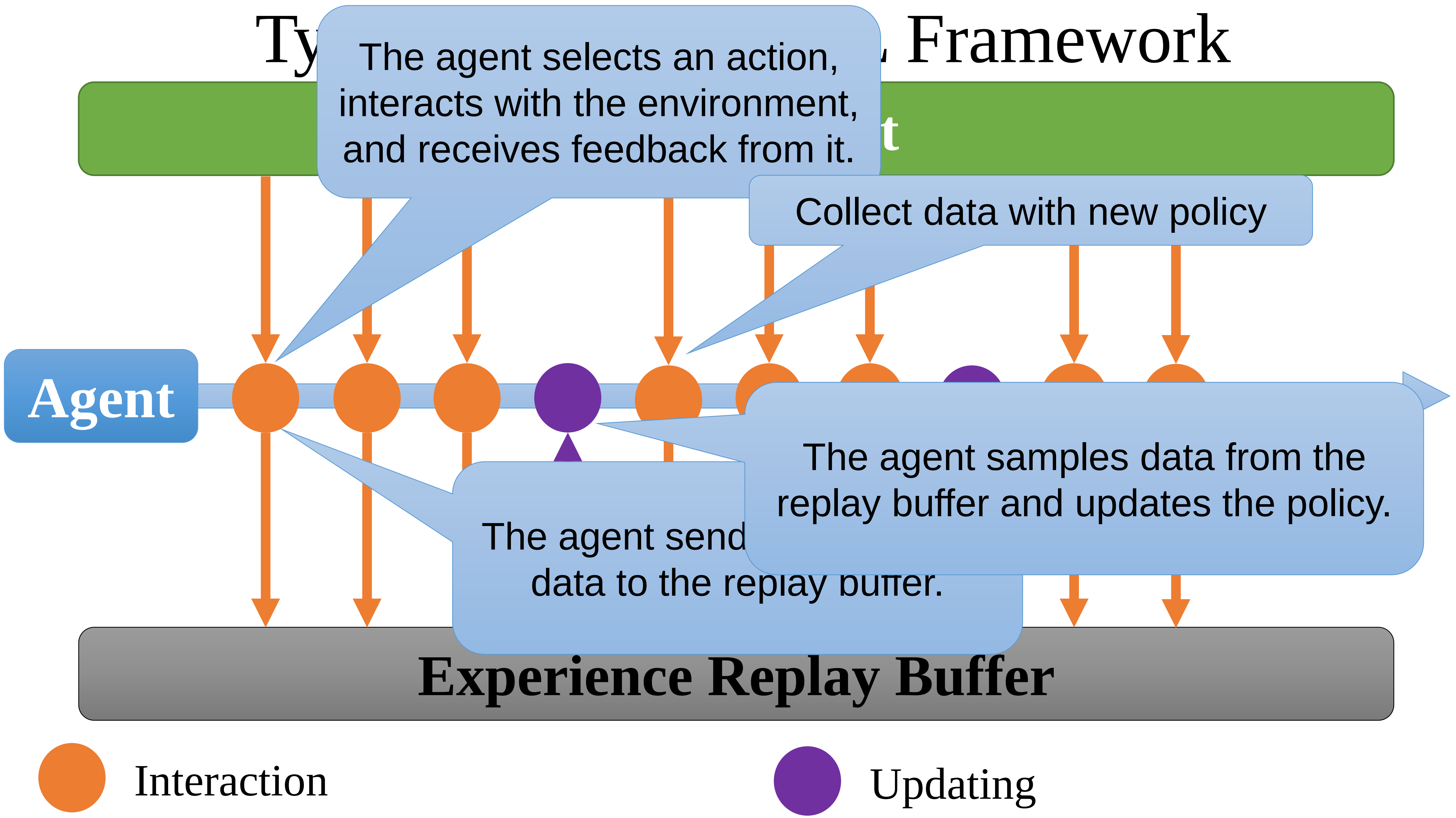
Low-Switching-Cost Reinforcement Learning on Robot Control Environments

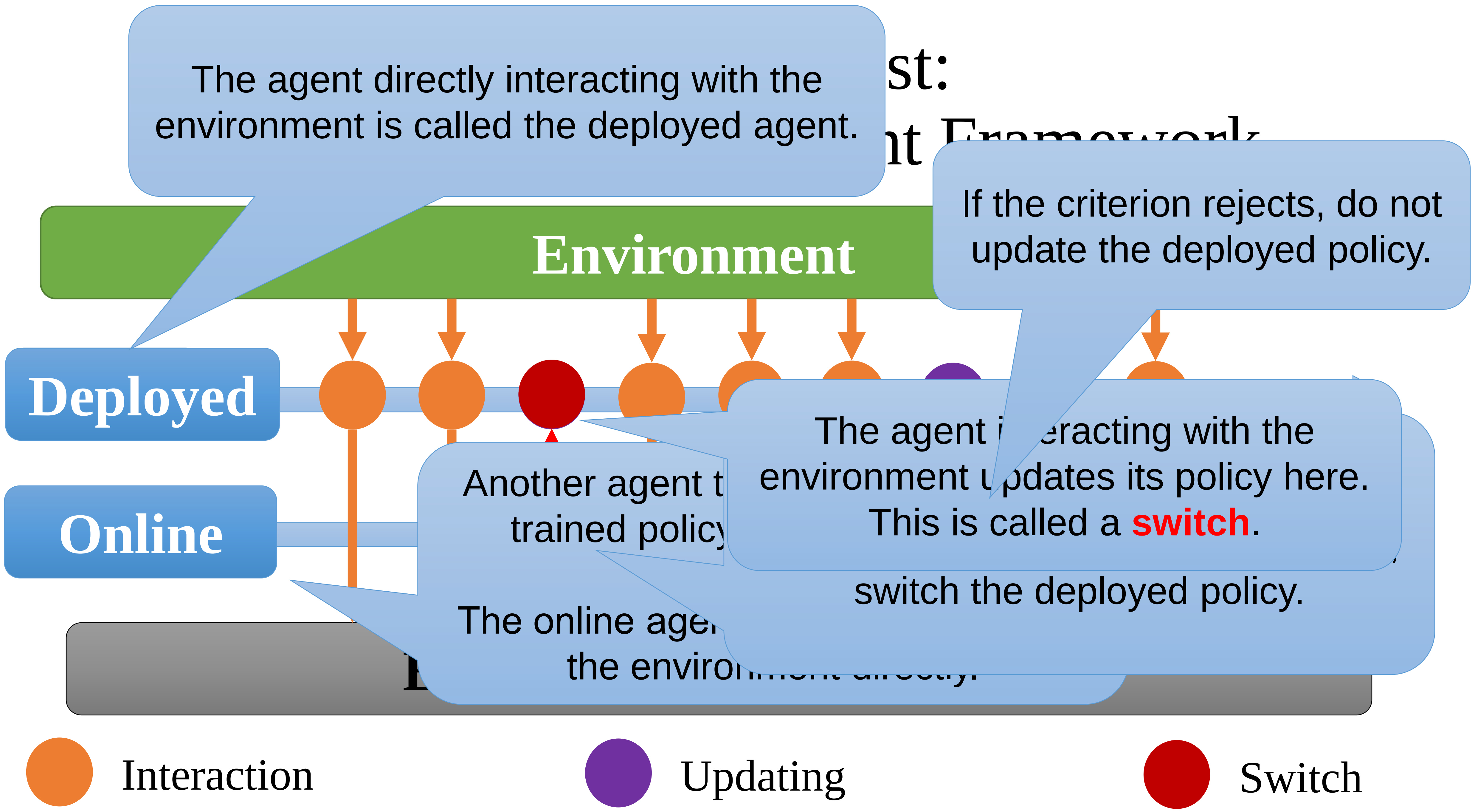
Yancheng Liang

Yushuo Chen

Part of the paper

“Beyond Information Gain: An Empirical Benchmark for Low-Switching-Cost Reinforcement Learning”


Ty
Framework
The agent selects an action, interacts with the environment, and receives feedback from it.
t
Collect data with new policy
Agent
The agent samples data from the replay buffer and updates the policy.
The agent send
data to the replay buffer.
Experience Replay Buffer
Interaction
Updating


st:
nt Framework
The agent directly interacting with the environment is called the deployed agent.
Environment
If the criterion rejects, do not update the deployed policy.
Deployed
Online
The agent i eracting with the environment updates its policy here. This is called a **switch**.
Another agent t
trained policy
switch the deployed policy.
The online age
the environ
Interaction
Updating
Switch

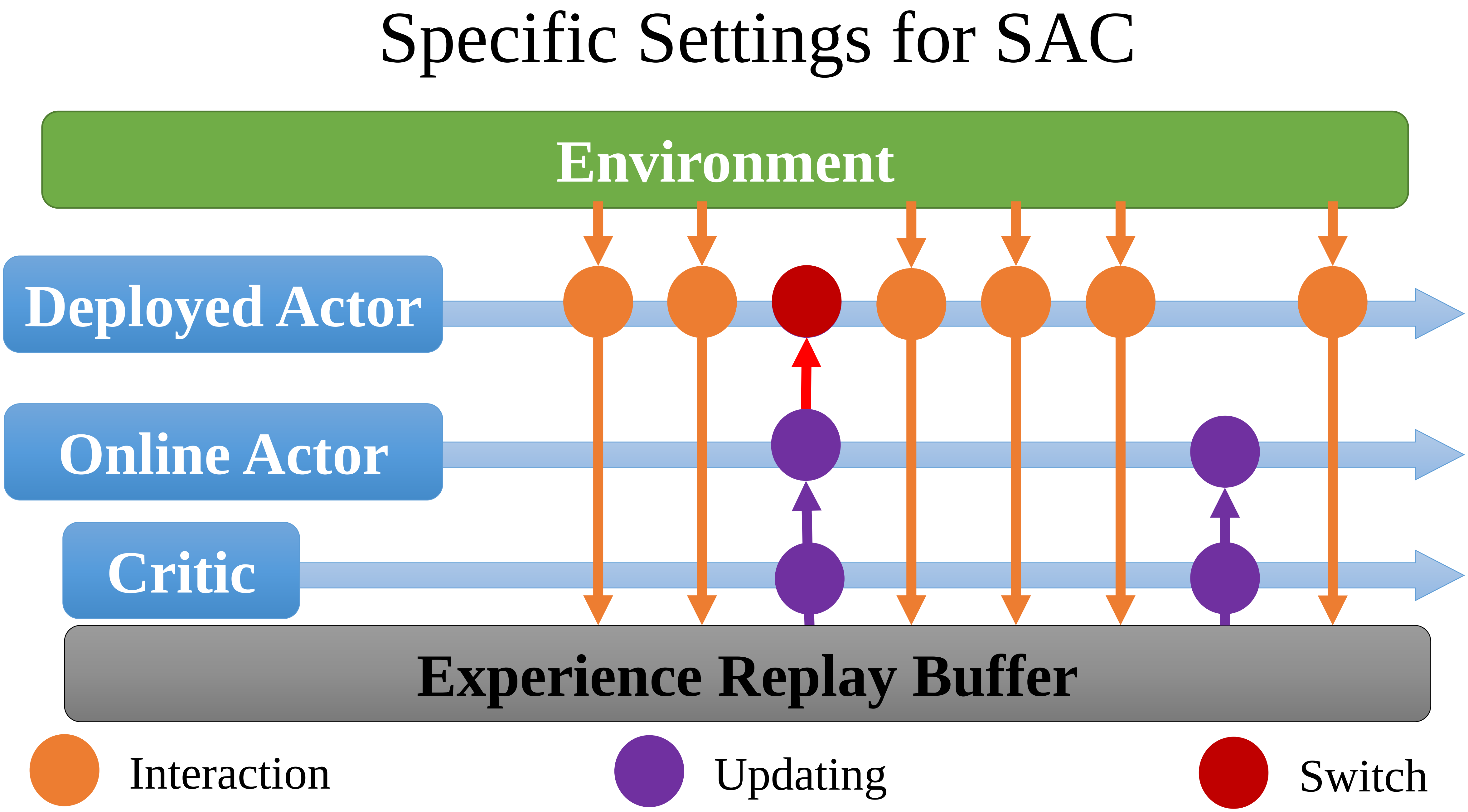
Goal: switch rarely, and maintain the performance

Why Low-Switch-Cost?

- Cost: high cost to deploy the newest policy
- Risk: in medical, robotics …

MuJoCo: Environments for Robot Control Tasks

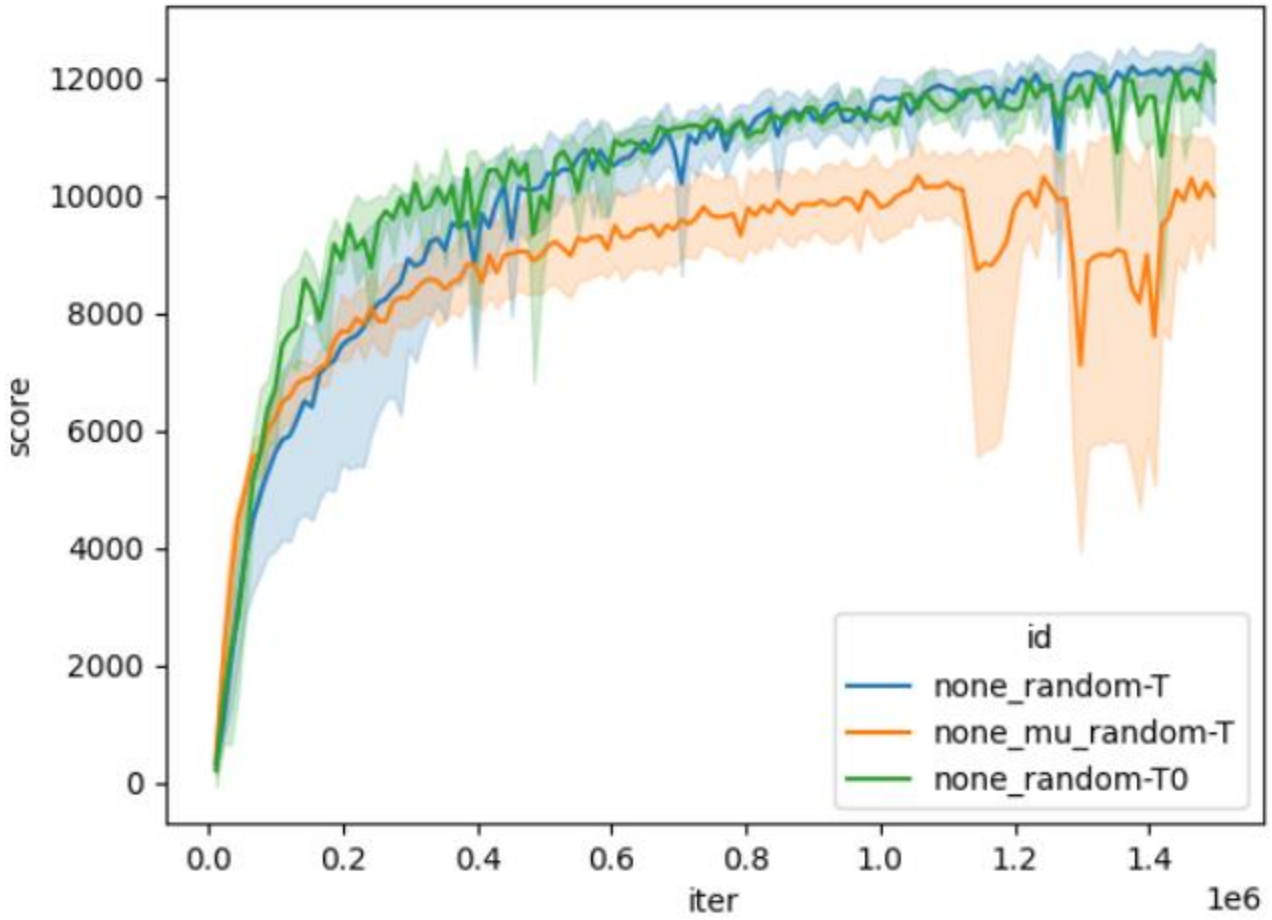
- High dimensional states
- Continuous action space
    - Infinite number of possible states
- SOTA: soft actor-critic[1] (SAC)
    - Actor: actual policy
    - Critic: an auxiliary model to help the learning of the actor


Specific Settings for SAC
Environment
Deployed Actor
Online Actor
Critic
Experience Replay Buffer
Interaction
Updating
Switch

For SAC and MuJoCo

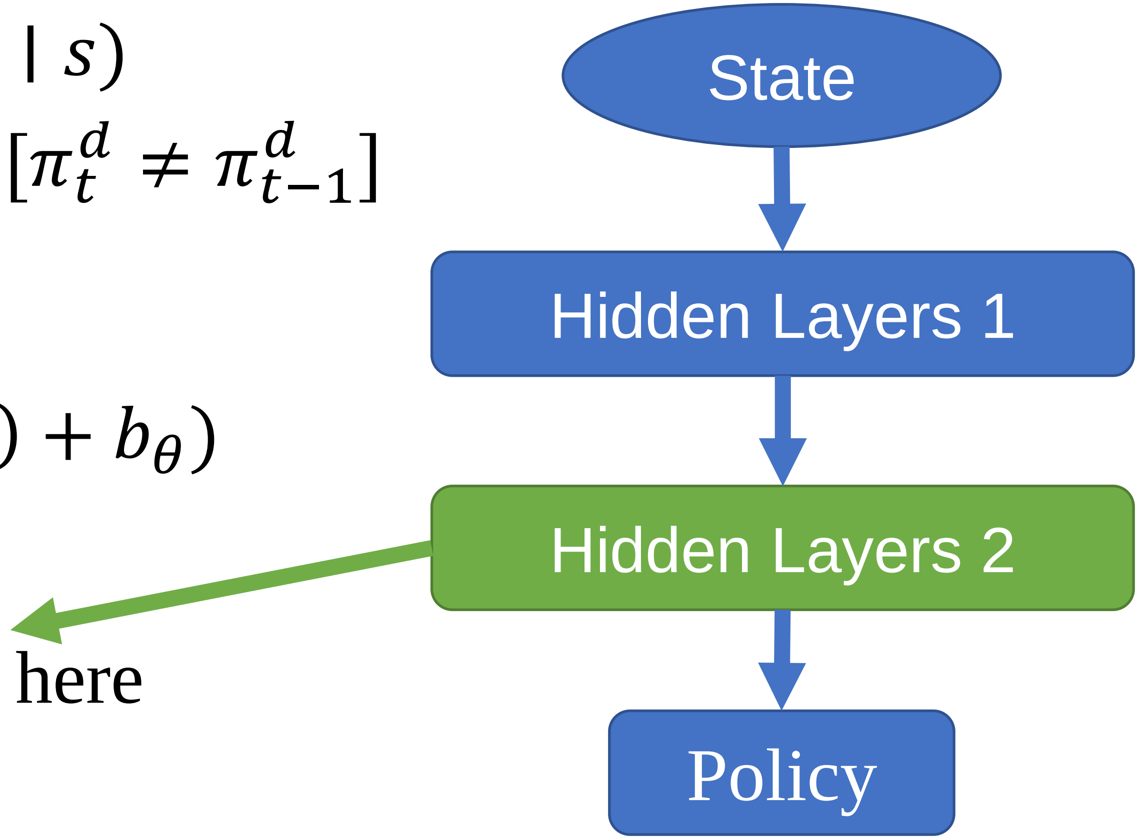
- Typically $a \sim N(\mu(s;\varphi), \sigma^2(s;\varphi))$

- Should not use deterministic exploration (deployed policy) $a = \mu(s;\varphi)$! As it significantly undermines the performance.
  - It does not help even with count-based reward bonus $r \mathrel{+}= O\left(\frac{1}{\sqrt{N(s,a)}}\right)$

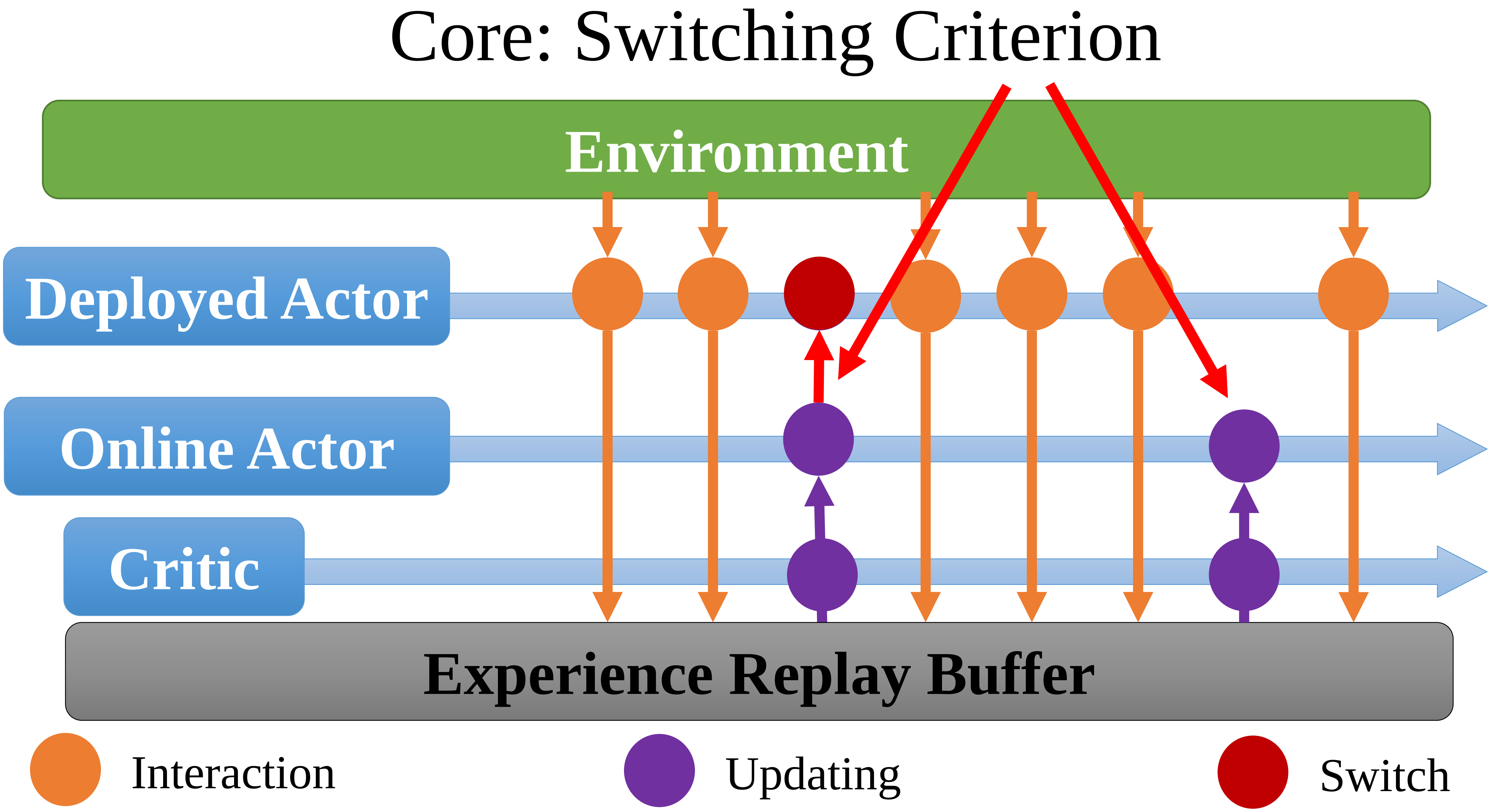


Specific Settings for SAC

- Policy of the deployed actor at step t: $\pi_t^d(a \mid s)$
- Policy of the online actor: $\pi_t^o(a \mid s)$
- Switching cost: $N_{switch} = \sum_{t=1}^{T} [\pi_t^d \neq \pi_{t-1}^d]$



- $\pi(\cdot \mid s) = softmax(W_\theta f(s;\phi) + b_\theta)$
    - Extract feature vector f here
    - Denoted as $f(s)$



Core: Switching Criterion
Environment
Deployed Actor
Online Actor
Critic
Experience Replay Buffer
Interaction
Updating
Switch

Intuition from the Theory: Information Gain as Switching Criteria

- UCB2[2] for Multi-Arm Bandit:
  - UCB: let $\tilde{r}_j = \bar{r}_j + O\left(\sqrt{\frac{\log T}{N(j)}}\right)$, choose $a = \arg\max \tilde{r}_j$
  - UCB2 (low switch cost): re-compute a only when $N(a) = (1+\eta)^k$

Works poorly:
Too many switches!
(UCB assume all arms are independent)

- Generalization[3] for MuJoCo
  - Count $N(s, a)$, only switch when it doubles
  - Use LSH: $\phi: R^{d_s + d_a} \to \{-1,1\}^{d_h}$ and count $N(\phi(s, a))$

Visitation

- Information Matrix for Linear Stochastic Bandit [4][5]:
  - $\left|\tilde{\theta} - \theta^*\right| \le O(\sqrt{\log \det(\bar{V})})$ where $\bar{V}$ is the information matrix $\bar{V} = \sum_{t=1}^{T} X_t^\top X_t$. Switch only when $\det(\bar{V})$ doubles
  - Still use $\phi: R^{d_s + d_a} \to \{-1,1\}^{d_h}$ and $\bar{V} = \sum_{t=1}^{T} \phi(s,a)^\top \phi(s,a)$,

Info

Naïve Switching Criteria

- Fix_n: switch after a fixed n number of step
  - Works fairly well but we need to tune n for different tasks

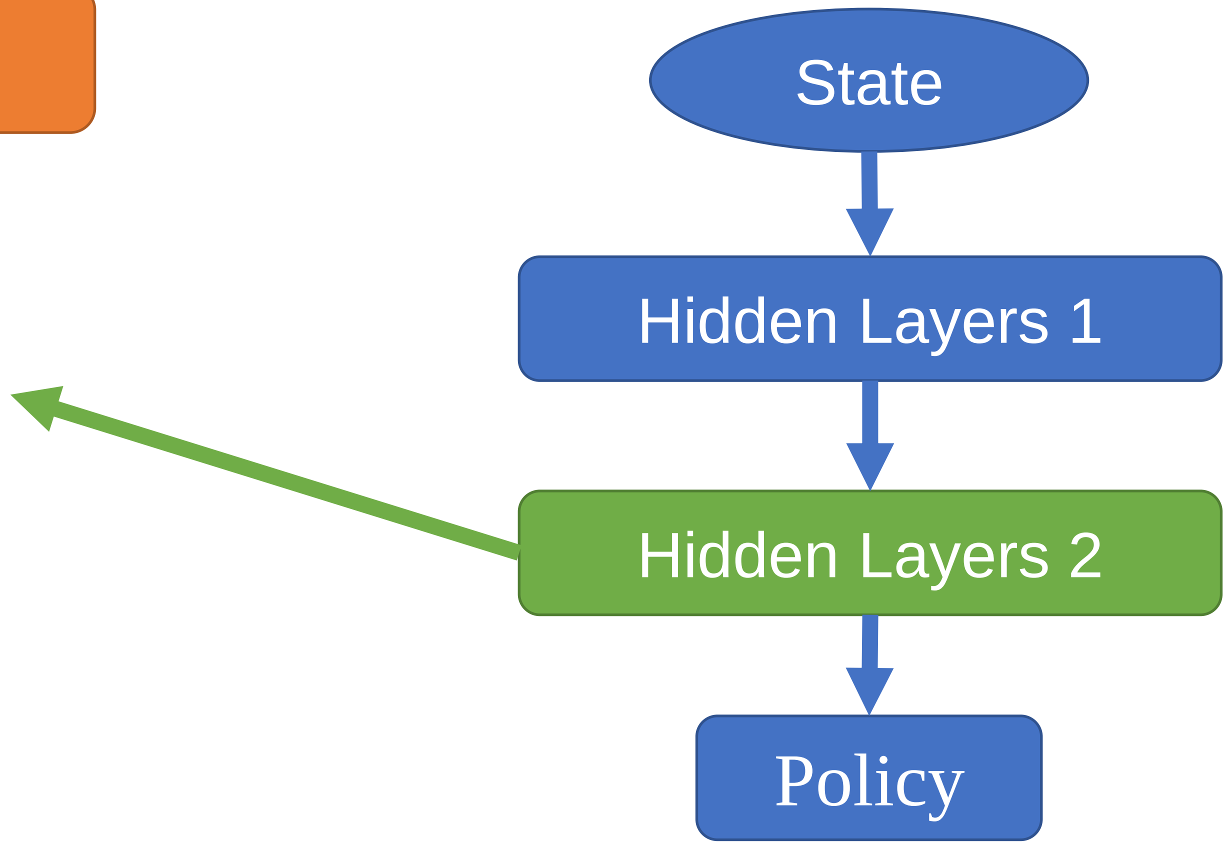
- KL Divergence: switch if $\mathbb{E}_s\left[KL(\pi^o(\cdot|\ s\ )||\pi^d(\cdot|\ s\ ))\right]$ is larger than a threshold
  - Switch even more frequently as the agent becomes stronger
  - $KL \approx \frac{\Delta\mu^2}{2\sigma^2}$ and $\sigma \to 0$ as the agent learns more

KL Divergence

Feature-Based Switching Criteria

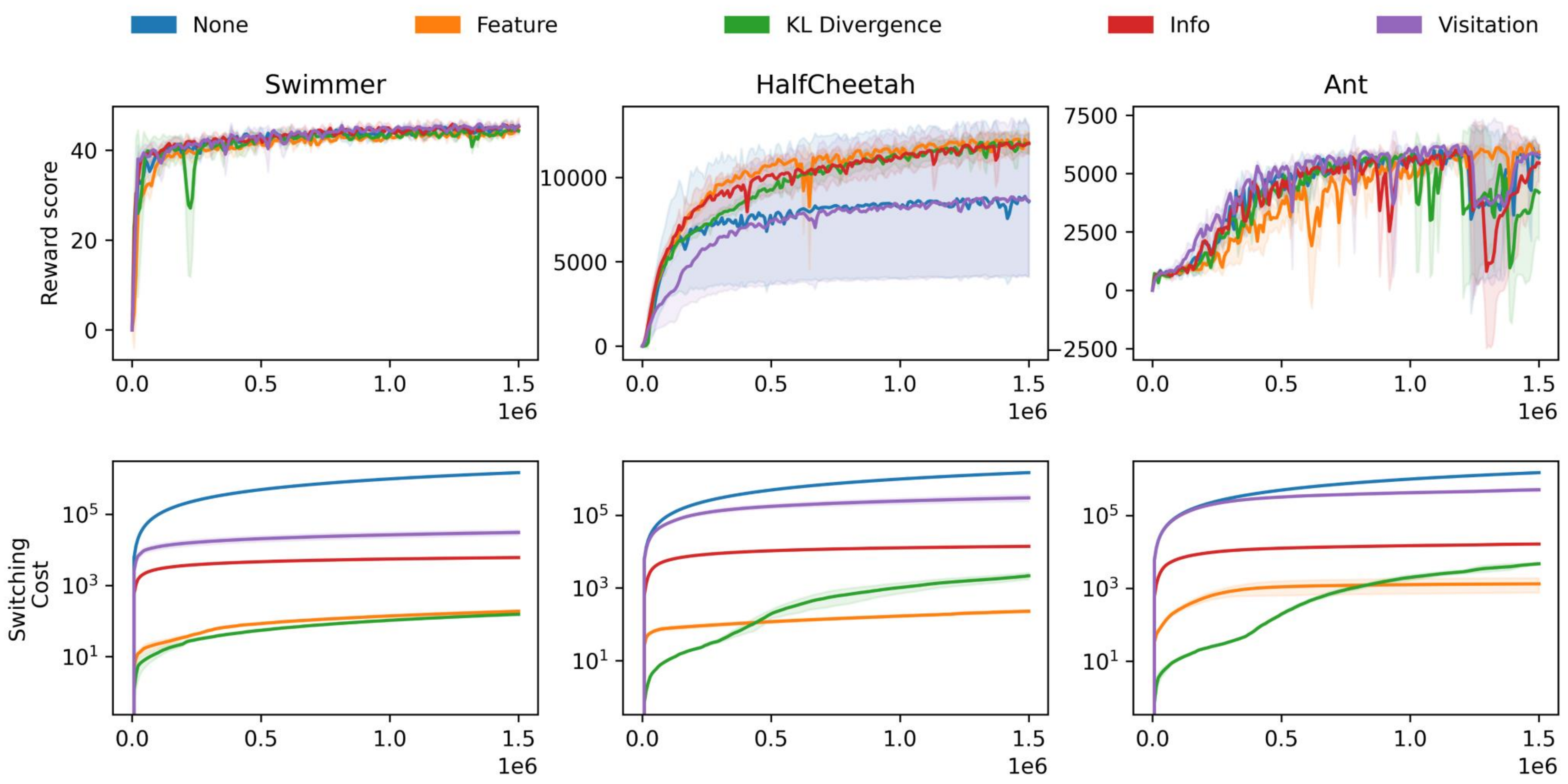
- Switch based on cos-similarity of the features $\mathbb{E}_s\left[\left\langle \widehat{f^d(s)}, \widehat{f^o(s)} \right\rangle\right]$
  - $\hat{n} = \frac{n}{||n||_2}$

Feature

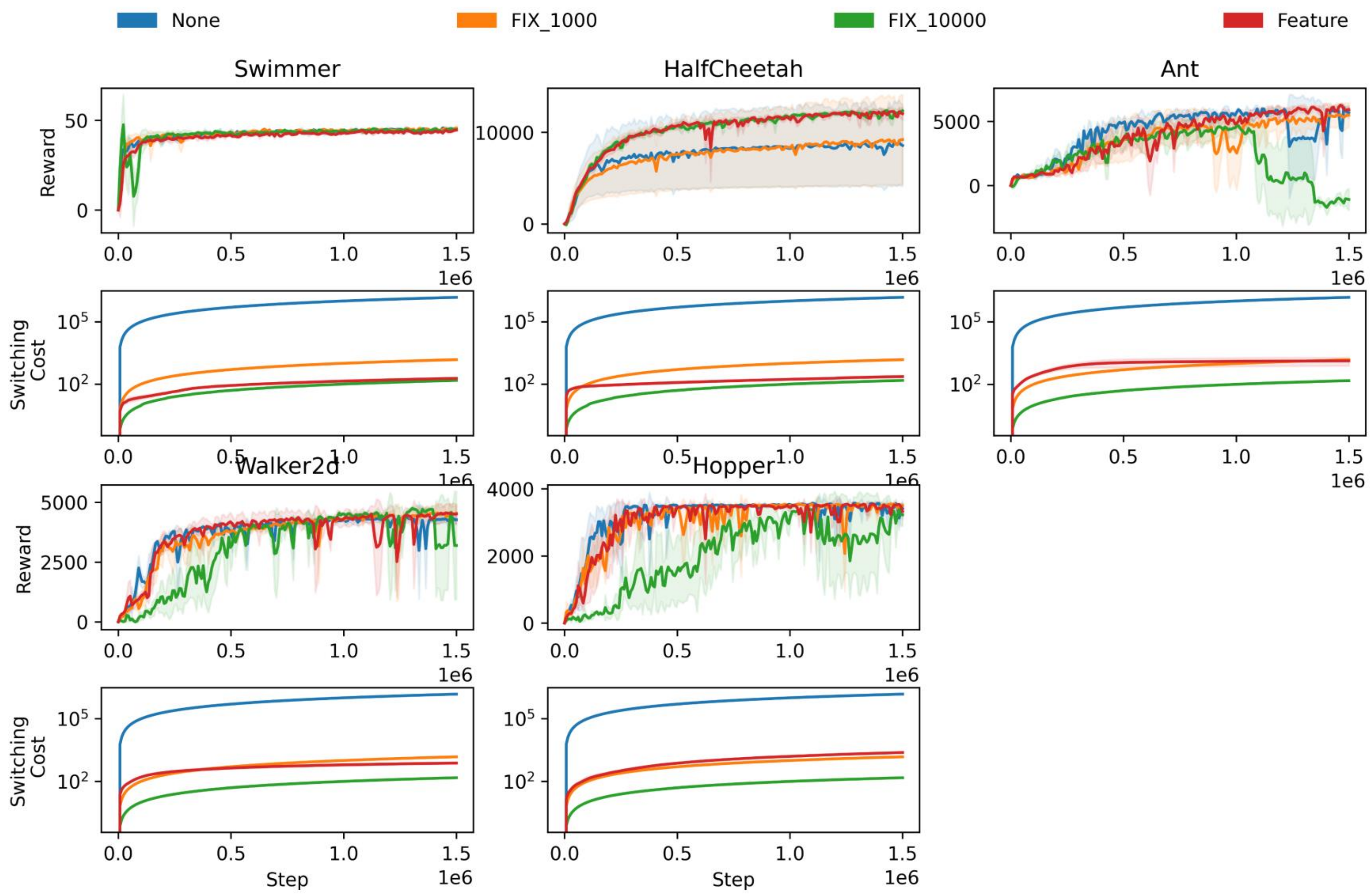


- Extract feature vector f here
- Denoted as $f(s)$

- “Feature”: best performance, lowest switching cost
- Information Gain: high switching cost
- “KL”: switch more as learning more



- "Feature" automatically adjusts the switching frequency.



Future Work

- Best switching cost for fix_n of different environments

- More on better switching criteria

Thank You for Listening!

Q & A

Acknowledgement

- Work with Prof. Wu, Prof.Du and our TAs, Shusheng Xu and Yunfei Li.

References

- [1] Haarnoja, Tuomas, et al. "Soft actor-critic algorithms and applications." arXiv preprint arXiv:1812.05905 (2018).
- [2] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. Machine learning, 47(2-3):235–256, (2002).
- [3] Bai, Yu, et al. "Provably efficient q-learning with low switching cost." arXiv preprint arXiv:1905.12849 (2019).
- [4] Yasin Abbasi-yadkori, D´avid P´al, and Csaba Szepesv´ari. Improved algorithms for linear stochastic bandits. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems, volume 24, pages 2312–2320. Curran Associates, Inc., (2011).
- [5] Ruan, Yufei, Jiaqi Yang, and Yuan Zhou. "Linear bandits with limited adaptivity and learning distributional optimal design." arXiv preprint arXiv:2007.01980 (2020).