Reinforcement Unlearning

Dayong Ye[1], Tianqing Zhu[2], Congcong Zhu[2], Derui Wang[3], Kun Gao[1], Zewei Shi[3], Sheng Shen[4], Wanlei Zhou[2], and Minhui (Jason) Xue[3]

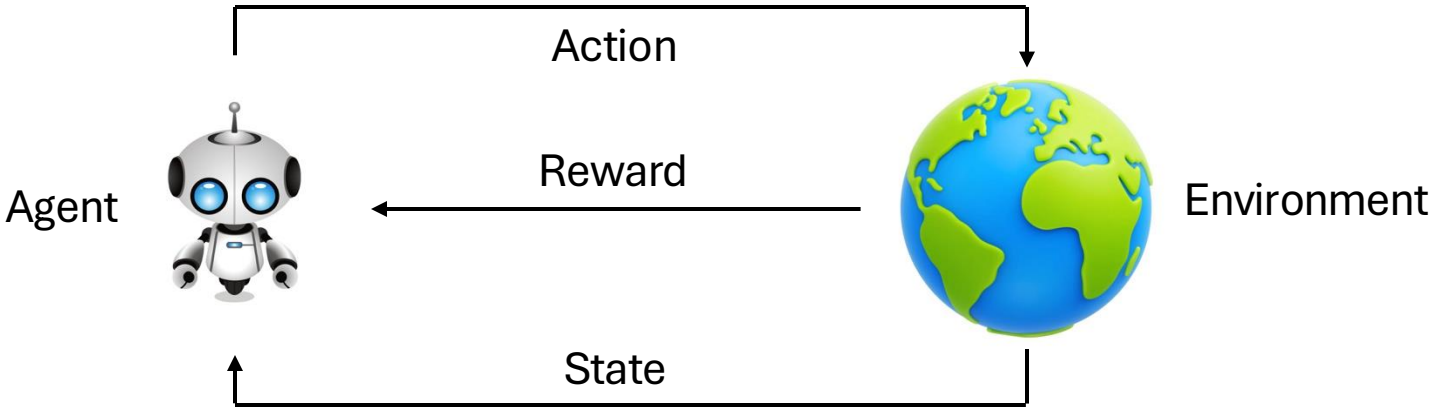
University of Technology Sydney (UTS)

Background

- **Machine Unlearning** is the process of selectively removing the influence of specific data from a trained machine learning model.

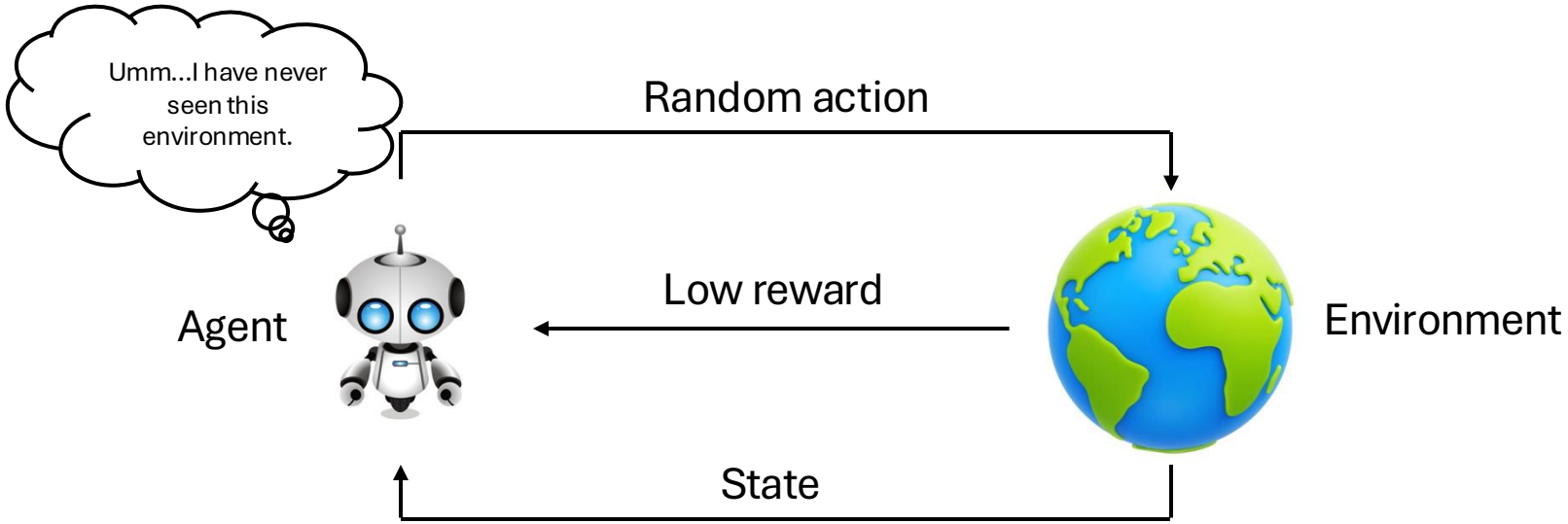
Background

- **Reinforcement Learning** is a type of machine learning, where an agent learns to make decisions by interacting with an environment to maximize cumulative rewards.



Background

- Machine Unlearning+Reinforcement Learning = **Reinforcement Unlearning**.



Background

- Why **reinforcement unlearning** is important?
  - **Privacy and Compliance**: Privacy regulations, like GDPR, enable users to request their data, or other information property, to be deleted from systems.
  - **Safety-Critical Systems**: In scenarios like autonomous driving or robotics, where outdated or incorrect knowledge could lead to unsafe behavior.
  - **Dynamic Environments**: To adapt to changes where some learned information becomes irrelevant or sensitive.

Problem Definition

- A learning environment is formulated by the tuple $M = \langle S, A, T, r \rangle$. S and A denote the state and action sets, respectively, while T represents the transition function, and r is the reward function.
- Suppose there is a set of n environments: $\{M_1, \dots, M_n\}$, and the target environment to be unlearned is $M_u = \langle S_u, A_u, T_u, r \rangle$.
- Given a learned policy π, **the goal** is to update π to π' such that the accumulated reward received in M_u is minimized:

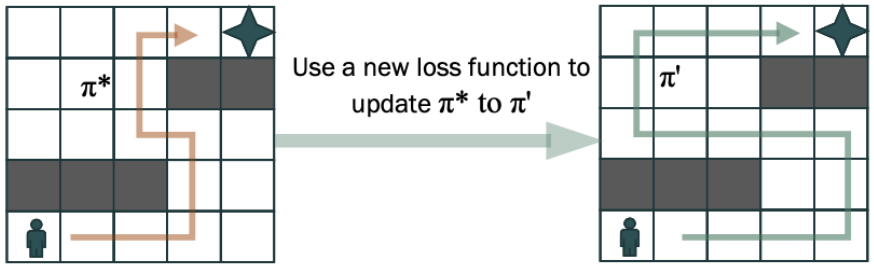
$$min_{\pi'} ||Q_{\pi'}(s)||_\infty,$$

where $s \in S_u$, while the accumulated reward received in the remaining environments remains the same:

$$min_{\pi'} ||Q_{\pi'}(s) - Q_\pi(s)||_\infty.$$

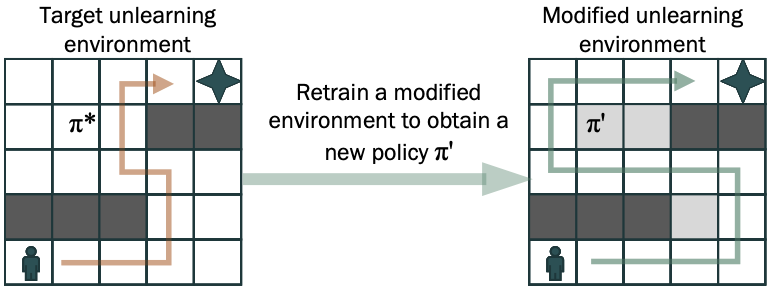
Methodology

- Method 1: Decremental RL-based Approach, which consists of two steps.
  - Step 1: The agent explores the unlearning environment M_u, collecting experience samples in it: $(s_1, a_1, r_1, s_2), \ldots, (s_m, a_m, r_m, s_{m+1})$.
  - Step 2: Fine-tune the agent using the collected experience samples with loss function: $L_u = \mathbb{E}_{s\sim S_u}[\|Q_{\pi'}(s)\|_\infty] + \mathbb{E}_{s\nsim S_u}[\|Q_{\pi'}(s) - Q_\pi(s)\|_\infty]$



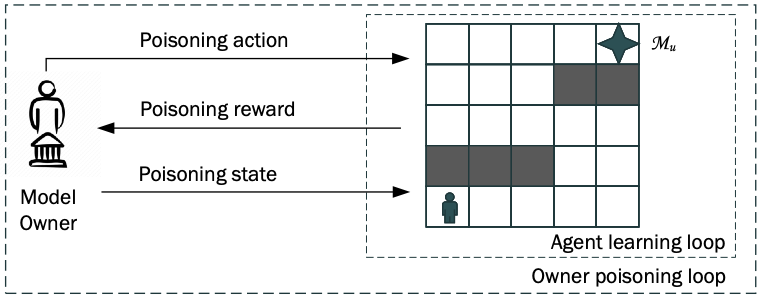
Methodology

- Method 2: Poisoning-based Approach, which consists of three steps.
  - Step 1: We apply a random poisoning strategy to modify the transition function of the unlearning environment M_u.
  - Step 2: The agent learns a new policy in this modified environment.
  - Step 3: Based on the agent's learned policy, we update the poisoning strategy and re-poison the unlearning environment.
  - These three steps are iteratively repeated until a predetermined number of poisoning epochs is reached.



Methodology

- Method 2: Poisoning-based Approach, which consists of three steps.
  - Step 1: To modify the transition function $T_u(s'|s,a)$, we introduce a poisoned transition function $\hat{T}_u(\hat{s}'|s,a)$. This means that after the agent takes action a in state s, the agent observes new state $\hat{s}'$ instead of s'.
  - Step 2: Train the agent in this modified environment: $\widehat{M}_u = \langle S_u, A_u, \hat{T}_u, r \rangle$.
  - Step 3: Based on the learned policy in $\widehat{M}_u$, we update the poisoning strategy using the reward loss function: $R_i := \lambda_1 \Delta(\pi_i(s_i)||\pi'(s_i)) + \lambda_2 \sum_{s \nsim S_u} \sum_a \pi_i(s,a) r(s,a)$.

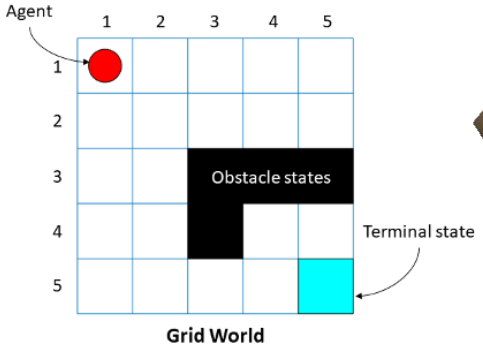


Methodology

- Comparison of the decremental RL-based and poisoning-based approaches:
  - The decremental RL-based approach directly adjusts the agent's policy by minimizing its rewards in the unlearning environment. It is computationally efficient, as it relies on fine-tuning the agent's policy without altering environmental dynamics.
  - The poisoning-based approach modifies the unlearning environment itself by altering transition functions. It is computationally intensive but is more effective than the decremental RL-based approach, because it directly targets the environment's underlying structure, ensuring a thorough unlearning process.

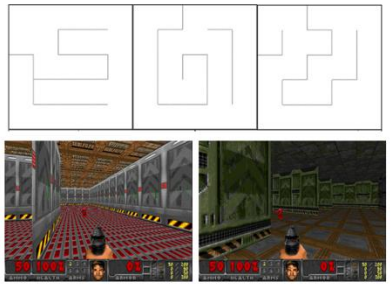
Experiments

- Experimental settings:
  - Grid world: The objective for the agent in this task is to navigate towards the predetermined destination.
  - Virtual home: It is a multi-agent platform designed to simulate various activities within a household setting.
  - Maze explorer: It is a customizable 3D platform. The objective is to guide an agent through a procedurally generated maze to collect a predetermined number of keys.



Grid World

Virtual Home



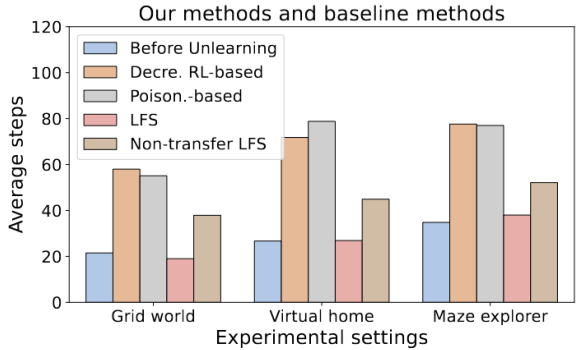
Maze Explorer

Experiments

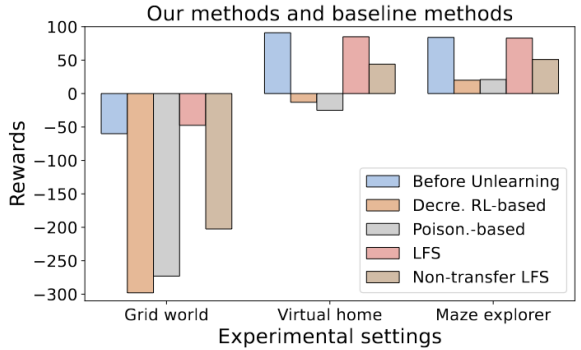
- Baseline methods:
  - Learning from scratch (LFS): This method removes the unlearning environment and then retrains the agent from scratch using the remaining environments.
  - Non-transfer learning from scratch (Non-transfer LFS): This method retrains the agent in all the environments. When retraining in the unlearning environments, an inverse loss function is used to minimize the agent's cumulative reward. When retraining in remaining environments, a standard loss function is used to maximize the agent's overall reward.

Experiments

- General results: After unlearning, the agent's performance in the unlearning environment reduces when using both the decremental RL-based and poisoning-based methods, where the agent requires more steps to complete tasks and receives lower reward.



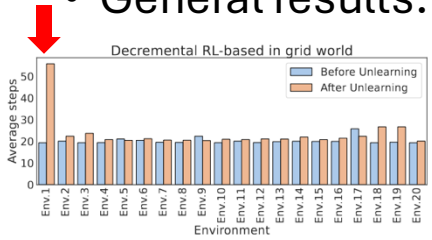
(a) Average steps of the four methods

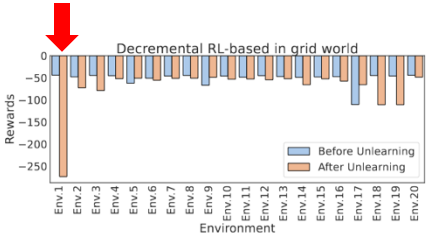


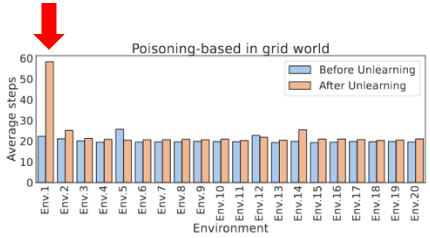
(b) Rewards of the four methods

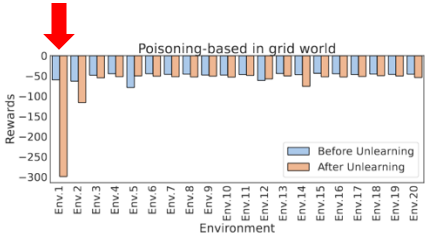
Experiments

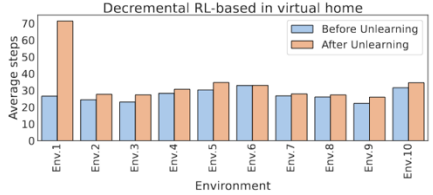
- General results:

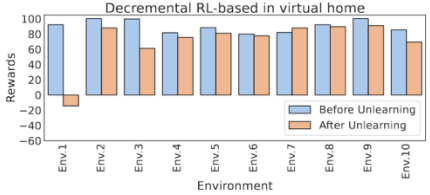


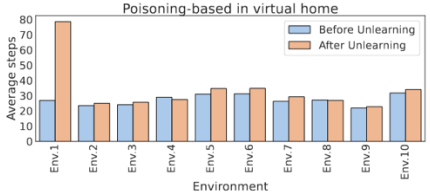


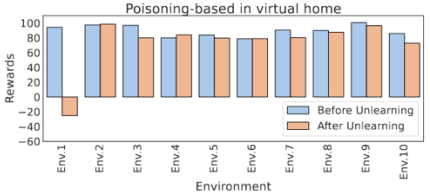


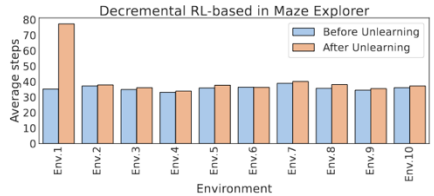


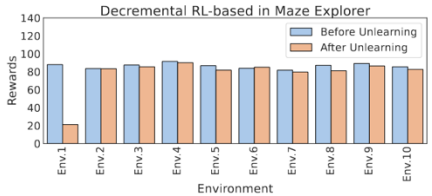


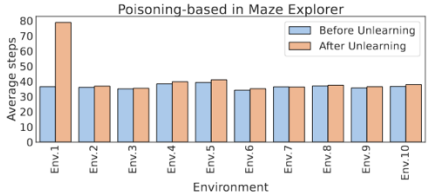


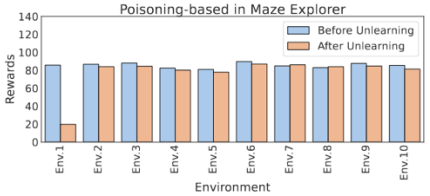








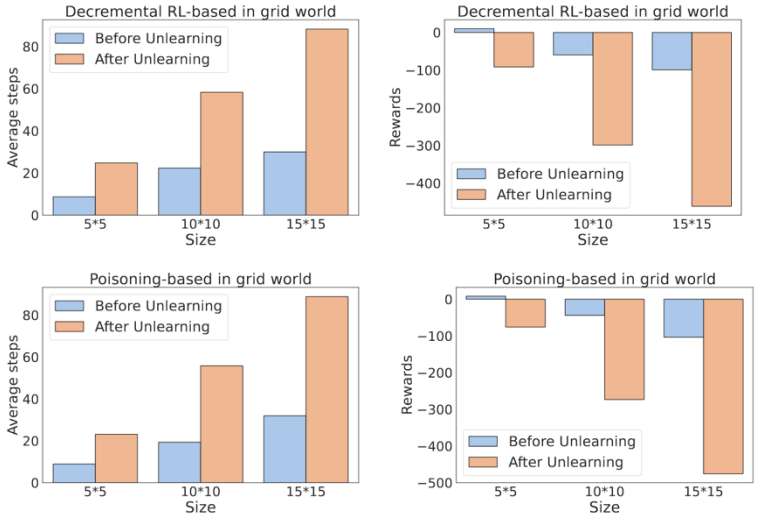




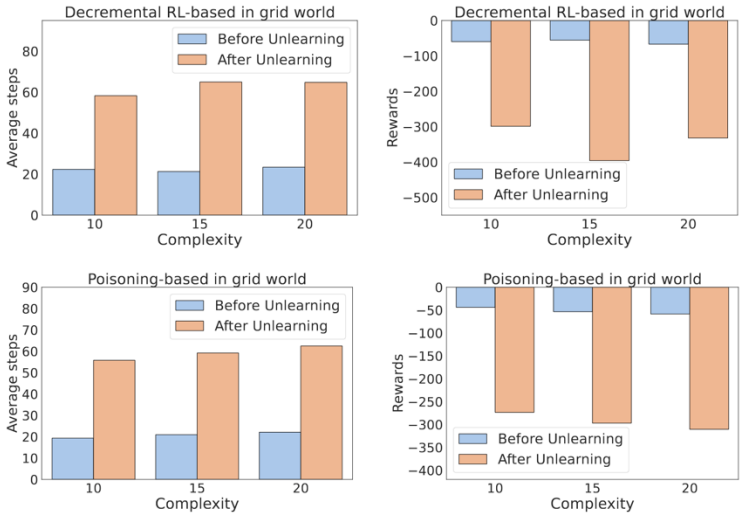
Experiments

- Adaptability study:

Impact of Environment Size



Impact of Environment Complexity



Experiments

- Computation overhead:

TABLE II. COMPUTATION OVERHEAD OF THE FOUR METHODS IN GRID WORLD (SECONDS)

Methods	Computation time of unlearning 1 environment	Computation time of unlearning 10 environments
Decremental RL	$18.80s$	$64.83s$
Poisoning	$20.17s$	$73.56s$
LFS	$198.62s$	$746.25s$
Non-transfer LFS	$196.97s$	$742.51s$

Experiments

- Privacy study: This study is conducted using recommendation systems, where the key indicator is recommendation accuracy.

TABLE III. COMPARISON OF RECOMMENDATION ACCURACY AND REWARDS BEFORE AND AFTER UNLEARNING

Methods	Performance for the unlearned user		Average performance for the remaining users	
	Accuracy	Reward	Accuracy	Reward
Before Unlearning	92.07%	41.42	91.52%	39.9
Decremental RL	68.63%	20.03	90.89%	38.82
Poisoning	64.41%	18.25	91.43%	37.17

Conclusion

- This work proposed a new concept: reinforcement unlearning.
- To achieve reinforcement unlearning, we designed two novel unlearning methods: decremental RL-based and poisoning-based.
- We conducted extensive experiments to evaluate the effectiveness of both unlearning methods in ensuring forgetting quality within the unlearning environments while maintaining performance in the remaining environments.

Questions?