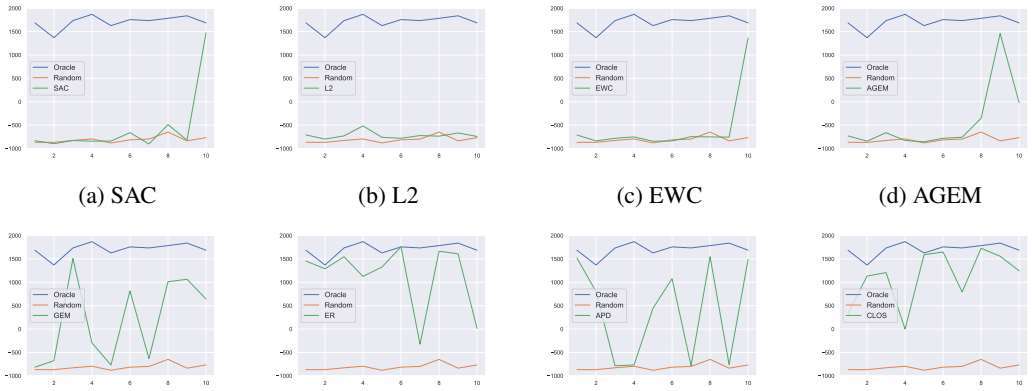
Appendix

1 DDPG Experiments

In addition to SAC, we also provide experiments on DDPG[1]. Instead of the probabilistic policy used in SAC, DDPG has a deterministic policy. The average performances are shown in 1. Though DDPG-based algorithms have worse performance than SAC, CLOS, ER, GEM and APD still have a satisfying result, which is consistent to that in SAC.



(a) SAC (b) L2 (c) EWC (d) AGEM

(e) GEM (f) ER (g) APD (h) CLOS

Figure 1: Performance of lifelong learning methods with the RL framework. The x-axis denotes task IDs. The y-axis denotes cumulative rewards. Each lifelong reinforcement learning algorithm consists of a lifelong learning algorithm and DDPG as the RL algorithm. The figure shows the performance of the policy on each task after finishing learning the last one. Oracle and random agent is included in each figure for comparison.

2 Reinforcement Lifelong Learning Framework

The pseudo code is shown in Algorithm. 1. The descriptive figure of it is shown in Fig. 2.

References

[1] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *ICLR (Poster)*, 2016.

Algorithm 1 Reinforcement Lifelong Learning Framework

INPUT K: the number of tasks needed to be learnt;
$D \longleftarrow \{\} * K$ # Initialize K empty replay buffer;
$C \longleftarrow \{c_1, c_2, \dots c_K\}$ # Initialize the critic set;
for $k \in \{1, 2, \dots K\}$ **do**
  $(s'_t, a'_t, s'_{t+1}, r'_t, d'_t) \sim RollOut(\theta, k)$;
  $D_k \longleftarrow D_k \cup (s^k_t, a^k_t, s^k_{t+1}, r_t)$;
  $(s^k_t, a^k_t, s^k_{t+1}, r^k_t, d^k_t) \sim D_k$ # Sample a data from the replay buffer;
  $l_c \longleftarrow (Q_{\phi^k}(s^k_t, a^k_t) - (r^k_t + \gamma(1 - d_t)Q_{\phi^k}(s^k_{t+1}, \pi_\theta(s^k_t)))^2$ #Compute the critic loss;
  $g^k_c \longleftarrow \nabla_{\phi^k} l_c$;
  $\phi^k \longleftarrow \phi^k - \beta g^k_c$; # Update the critic parameters;
  $l_a, g^k_a \longleftarrow -c_k(\pi_\theta(s^k_t), k), -\nabla c_k(\pi_\theta(s^k_t), k)$ # Compute the actor loss;
  $l'_a, g^{k\prime}_a \longleftarrow LifelongLearning(l_a, g^k_a)$ # Update the Actor Loss and gradients according to the lifelong learning algorithm
  $\theta_c \longleftarrow \theta_c - \alpha g^{k\prime}_a$ # Update the shared module in the actor
  $\theta_s \longleftarrow \theta_s - \alpha g^k_a$ # Update the task-specific module in the actor.
end for

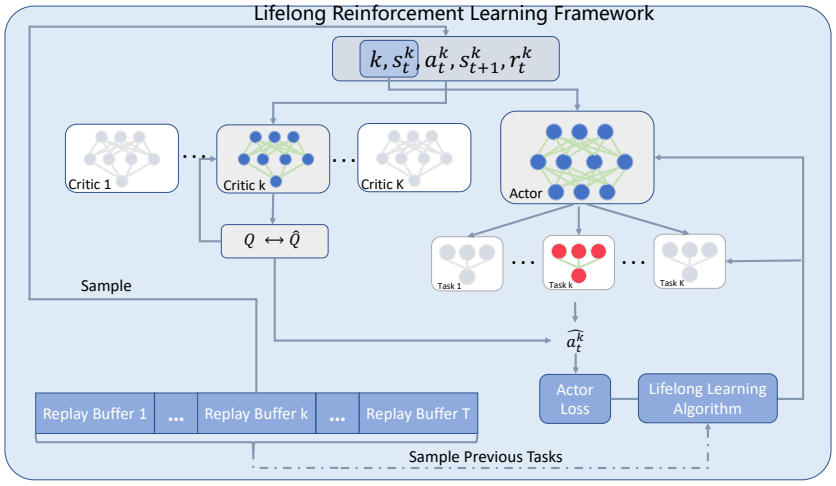


Figure 2: Reinforcement Lifelong Learning Framework. The framework combines an Actor-Critic algorithm with a lifelong learning algorithm. The actor is decomposed into shared parts and task-specific parts. Each task is assigned an individual critic. We keep the replay memory of previous tasks and sample data from them when needed in the lifelong learning process.