Evaluations of the Gap between Supervised and Reinforcement Lifelong Learning on Robotic Manipulation Tasks

Anonymous Author(s)
Affiliation
Address
email

Abstract: Overcoming catastrophic forgetting is of great importance for deep learning and robotics. Recent lifelong learning research has great advances in supervised learning. However, little work focus on reinforcement learning(RL). We propose a framework to modify supervised lifelong learning algorithms to be compatible with RL. We develop a manipulation benchmark task set to evaluate the performance of lifelong learning algorithms on robotic RL tasks. We evaluate the state-of-the-art lifelong learning algorithms from different categories on our benchmark and summarize the pros and cons for each category when applied in RL scenarios.

Keywords: Lifelong Learning, Reinforcement Learning, Manipulation

1 Introduction

Humans and animals continuously learn different skills and knowledge in their lifetime. It is natural for them to master a new skill without forgetting those learned before in most cases. However, neural networks suffer a lot from memorizing previously learned tasks. This phenomenon is called catastrophic forgetting [1, 2, 3]. Recent research has focused on this problem and proposed various solutions called lifelong learning or continual learning.

On the other hand, manipulation has been a challenging task in the robotics community. Recent research is focusing on enabling robots to have a more flexible and dexterous manipulation ability, e.g., multi-task learning, policy adaptation, and lifelong learning. Specifically, lifelong learning is of great importance for robotics. It provides possible solutions for developmental robotics [4]. Humans learn different skills as they grow up rather than completely rely on innate skills. It would also be difficult to pre-program all the required skills for advanced robots when setting up in the factory. Future artificial intelligence and robotics will have to continuously master different tasks as humans do in their lifetime. The ability of lifelong learning will make robots suit better in various applications.

In this work, we focus on the lifelong learning problem for robotics manipulation tasks in the reinforcement learning(RL) domain. Nearly all of the recent lifelong learning research focuses on supervised learning. However, lifelong learning for RL is much more challenging than supervised learning. Not only because the training of RL itself is challenging and unstable[5], but in most RL tasks, policies are sensitive to forgetting. Forgetting could be considered as an attack in the action space of the agent. Recent work has demonstrated the vulnerability of RL agent to attack on action space[6]. Moreover, in RL tasks, an action at one step will influence states and actions followed up, which could aggravate the influence of forgetting. For example, in grasping tasks, slight misplacement of a fingertip is likely to break the balance of a grasp and lead to its failure. Though recent lifelong learning research has promising performances on supervised learning, little evidence has proven that they could directly be applied in robotic RL tasks. Research on how the state-of-the-art lifelong learning algorithms perform on robotic reinforcement learning tasks is urgently needed in the research community, which is the major motivation of our work.

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

In this paper, we establish a framework to study the possibilities to extend the existing supervised lifelong learning method to the reinforcement scenarios, demonstrating their effectiveness on a carefully-designed multi-task manipulation scenario. Our contributions can be summarized into three-folds:

1. We propose a framework which could modify traditional lifelong learning algorithms in supervised learning into reinforcement learning.
2. We propose a manipulation benchmark for lifelong learning with ten different manipulation tasks. Policies trained on our benchmark could quickly converge.
3. We evaluated Soft-Actor-Critic with the state-of-the-art lifelong learning algorithms from four different categories using our benchmark. Forgetting performances and robustness to task orders are evaluated in experiments. We further analyze the potential of different lifelong learning algorithms applied in RL based on our experiments.

2 Related Works

2.1 Lifelong Learning

Recent lifelong learning algorithms can be classified into the following four categories:

2.1.1 Regularization-Based Methods

Regularization-based methods add an additional regularization term in the loss function to ensure the learned network in the new task will not forget important information for the old tasks. The most intuitive method would add an MSE loss for the parameter changes among the current task and old tasks. Instead, [1] proposed Elastic Weight Consolidation(EWC). It incorporates the Fisher information matrix to compute the importance of network parameters and tries to keep the important ones unchanged. The most important advantage of regularization-based methods is that the model size will not increase as the number of tasks increases. However, purely regularization-based methods usually will not have as good performances as other methods.

2.1.2 Memory-Based Methods

Memory-Based Methods would use a tiny replay memory to store several samples from previously learned tasks and replay when learning new tasks. The key idea of memory-based methods is trying to transfer lifelong learning problems into classic deep learning tasks. Episodic Replay(ER)[7] combines samples from previous tasks with current tasks when computing the loss for the current task. [8] is an updated version of [7], which adds weight to learn the importance of the current task and previous task.

2.1.3 Gradient-Constraint Methods

Gradient-constraint methods modify the gradient to ensure that the update for the current task would have minimal influence on previous tasks. Gradient Episodic Memory(GEM)[9] projects the gradient into spaces to ensure the update for the current task will not decrease the performance of previous tasks. [10] proposed A-GEM and optimize the computation efficiency of GEM. Orthogonal Gradient Descent(OGD)[11] believes moving orthogonally to the gradients of previous tasks would have the smallest influence on them. Therefore, it projects the gradients for the current tasks into the orthogonal space of previous tasks. Continual Learning in Orthogonal Subspaces(CLOS)[12] ensures that the gradients are orthogonal by nature. It combines gradient-constraint methods and memory-based methods for a better performance.

2.1.4 Expansion-Based Methods

Expansion-based methods add a new sub-module when learning a new task. Thus, the size of the model increases with the learning procedure. Additive Parameter Decomposition(APD)[13] includes a shared parameter space with task-specific sub-networks. Learning a new task would add a new sub-network and would also constrain the changes in the shared parameters.

2.2 Lifelong Learning in Reinforcement Learning

Though various lifelong learning algorithms are mentioned above, currently, most algorithms only focus on solving supervised learning tasks. Few could be found on dealing with reinforcement learning tasks. Even if there are some algorithms trying to solve RL tasks, they are applied in image-based video games instead of robotic tasks, e.g., [14] can only be used in video games due to its generative image-based methods. [1] and [15] evaluated their methods only on Atari video games instead of robotic tasks. Even there is some work on lifelong learning applied in the robotics domain, they choose primitive tasks and they only provide little insight on how the state-of-the-art lifelong learning algorithms perform in robotic tasks. For example, [16] is implemented to real robots, but their method can be only used in simple locomotion tasks with a limited control space and its specific task definition. [17]evaluated lifelong learning on reinforcement tasks for robotics. Though good performance, their method is too simple and straightforward, and they only have five tasks in the sequence.

In conclusion, the lifelong learning problem of robotic reinforcement learning is poorly studied. An overall investigation about how the state-of-the-art lifelong learning methods perform on robotic reinforcement learning tasks is needed, which leads to our work in modifying traditional lifelong learning algorithms into reinforcement ones and comparing their performance.

3 Preliminaries

3.1 Markov Decision Process

We formulate our the problem as a Markov Decision Process (MDP). We use $s_t^k \in \mathbb{R}^{d^s}$ to denote the state of the agent at time step t in task k. The action vector $a_t^k \in \mathbb{R}^{d^a}$ are sampled from the policy $\pi_\theta(a_t^k)$, where θ denotes the policy parameters. $p(s_{t+1}^k|s_t^k, a_t^k)$ denotes the probability of transforming from the state s_t^k to s_{t+1}^k by executing the action a_t^k. The agent will receives a reward $r(s_t^k, a_t^k, s_{t+1}^k)$ at each step. Our goal is to maximizing the sum of the discounted reward for all the tasks $\mathbb{E}_{\pi_\theta}[\sum_{k=1}^{K}\sum_{t=0}^{\infty}\gamma^t r(s_t^k, a_t^k, s_{t+1}^k)]$

3.2 Actor-Critic Framework

Actor-Critic-based algorithms compose a majority of RL algorithms. [18, 19, 20] all incorporate the framework. In traditional supervised learning problems, the loss for each trial can be computed by comparing the outputs with the ground-truth labels. However, in most reinforcement learning cases, such labels do not exist. Instead, in the Actor-Critic framework, a Critic is trained simultaneously with an Actor to evaluate the action the Actor chooses. The Critic loss is usually defined by the mean square Bellman error.

$$L(\phi) = \mathbb{E}[(Q_\phi(s_t^k, a_t^k) - (r_t^k + \gamma(1-d_t)\max_{a_{t+1}^k} Q_\phi(s_{t+1}^k, a_{t+1}^k)))^2], \tag{1}$$

where Q denotes the Critic network with parameters ϕ. Here we use a standard critic loss expression. However, it can also be extended to other forms, for example, the entropy term could also be added into the critic loss like [18] does.

The Actor loss is defined by:

$$L(\theta) = -Q_\phi(s_t^k, \pi_\theta(s_t^k)). \tag{2}$$

4 Lifelong Reinforcement Learning Framework

In this section, we proposed the framework to modify traditional supervised lifelong learning algorithms to reinforcement learning. Traditional lifelong learning algorithms cannot be directly applied to RL because of the following two reasons: (a) the loss in supervised learning is easily defined, which is usually mean square error or cross entropy error. However, it is not straightforward in RL. A critic network is needed for evaluation of the policy network. (b) It would be hard to indicate which task the robot is performing, while in supervised learning, e.g., image classification, the network usually does not need to know and process the task ID since the images themselves have sufficient information to indicate which task they are within.

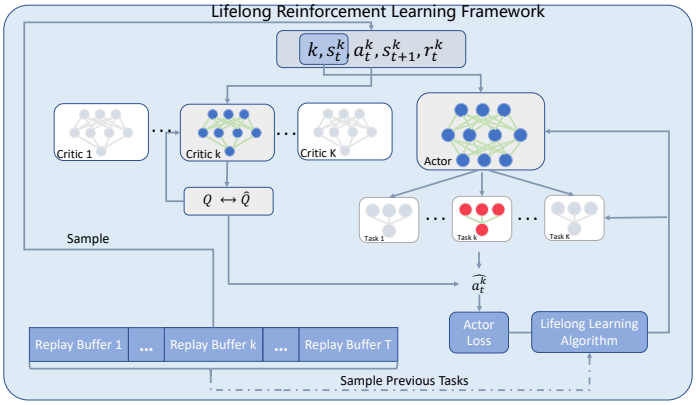


Figure 1: Lifelong Reinforcement Learning Framework. The framework combines an Actor-Critic algorithm with a lifelong learning algorithm. The actor is decomposed into shared parts and task-specific parts. Each task is assigned an individual critic. We keep the replay memory of previous tasks and sample data from them when needed in the lifelong learning process.

In order to solve these two problems, we modify the traditional lifelong learning algorithms in the following ways. The framework of our framework is shown in Fig.1

4.1 Actor-Critic Framework in the Lifelong Learning Algorithms

For most lifelong learning algorithms, it is different from traditional supervised learning because it modifies the loss or gradients, e.g., regularization-based methods add an additional term to the loss, gradient-constraint methods project gradients into sub-spaces. Thus, to enable the lifelong performance of RL algorithms, we adopt the Actor-Critic framework into our continual RL algorithms. we replace the loss in the lifelong learning algorithms with critic loss. Since in real-world applications, critics are usually discarded and only the Actor (policy network) is used, we only consider the continual learning properties of the actor while we use traditional deep RL algorithms to learn task-specific critics. However, it would also be straightforward to expand the lifelong learning algorithms for the critics.

4.2 Task-Specific Sub-Network

The observation spaces in RL tasks could be the same. In such cases, the policy network will have to distinguish different tasks by task IDs. It would be challenging if the task ID is directly input to the network for learning. Instead, we add task-specific layers into the policy network if the lifelong learning algorithms themselves do not include such task-specific parts. Formally, given k tasks, the parameters of the policy network π_θ can be decomposed into common parts and specific parts: $\theta = \{\theta_c\} \cup_{i=1}^{K} \{\theta_s^i\}$. For task i, the action a_t^i can be given by:

$$a_t^i = \pi_\theta(a_t^i) = \pi_{\theta_s^i}(\pi_{\theta_c}(s_t^i)). \tag{3}$$

Different sub-networks are chosen for different tasks according to task IDs. In our experiments, the task-specific sub-networks are only a single-layer affine transformation. If sub-networks with deep structures are implemented, the shared parts could be simply an identity transformation while the policy totally relies on the sub-network, which is basically training a policy network for each task.

The pseudo-code of our framework is shown in Algorithm1.

5 Lifelong Manipulation Benchmarks

In this section, we introduce the lifelong manipulation benchmarks. To the best knowledge of the authors, it is the first lifelong reinforcement learning benchmark. Our benchmarks adapt from ROBEL D'Claw[21], a mujoco-based robotic manipulation platform. We modify the tasks and get ten

Algorithm 1 Lifelong Reinforcement Learning Framework

INPUT K: the number of tasks needed to be learnt;
$D \longleftarrow \{\} * K$ # Initialize K empty replay buffer;
$C \longleftarrow \{c_1, c_2, \ldots c_K\}$ # Initialize the critic set;
for $k \in \{1, 2, \ldots K\}$ **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;
  $l'_a, g^{k'}_a \longleftarrow LifelongLearning(l_a, g^k_a)$ # Update the Actor Loss and gradients according to the lifelong learning algorithm
  $\theta_c \longleftarrow \theta_c - \alpha g^{k'}_a$ # 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

different manipulation tasks. The benchmarks consist of a robotic hand with three independent fingers. Each finger has three degrees of freedom. In our task, the robotic hand needs to turn the valve below 180 degrees. The observation space of the benchmarks consists of joint angles, joint velocities, and the valve angle. The action space of the robot consists of robot joint angles. The reward function is designed based on robot configurations and valve position to ensure a safe and ordinary manipulation. Each task only differs in the valve shape. Though each task seems to be similar, a policy trained on a single task cannot achieve high rewards on the other tasks. All the tasks are designed to have dense rewards for a faster learning procedure. Details of the reward function can be seen in [21].

One of the important challenges in lifelong reinforcement learning lies in the training time. Since reinforcement learning is usually unstable and simulation computations would usually rely heavily on CPU, training a policy could take a long time. This problem will be aggravated in the lifelong learning scenarios since each task should be trained one after another by definition. In our benchmark, each task is stable and easy to train. In our experiments, each policy can converge after 80,000 steps of updates using Soft-Actor-Critic(SAC). The details of the benchmark could be seen in Fig.2.

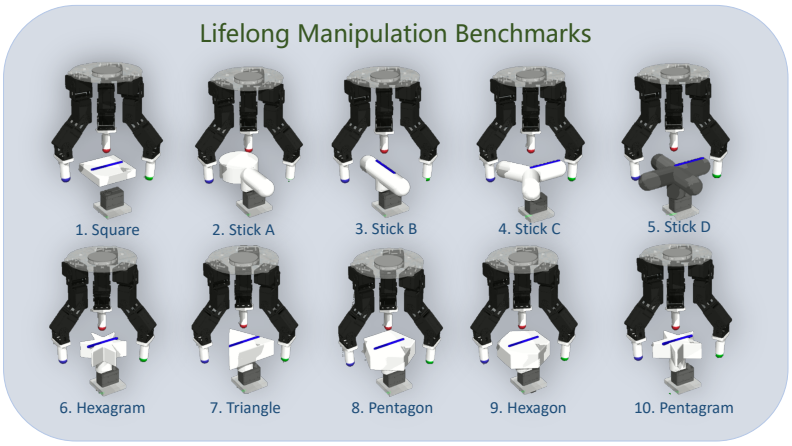


Figure 2: Ten manipulation tasks for lifelong manipulation benchmarks

6 Experiments

We will mainly focus on the forgetting of lifelong learning. We evaluated each algorithm using the average performance, which is slightly modified from the average accuracy, a standard evaluation metric used in [9][10] and [7].:

Average Performance: Let $p_{i,j}$ be the cumulative reward of the task j after the model finishes learning task i. Then the average performance P is defined as:

$$P = \frac{1}{K}\sum_{j=1}^{K} p_{K,j}, \tag{4}$$

where K denotes the total number of tasks. When evaluating, since we use the cumulative reward to evaluate policies, the time steps for each episode are set to be constant for the sake of fairness, no matter whether the task has already been done. Keeping in the successful state (in our benchmark, is keeping the valve at 180 degrees) will lead to a high reward at each step and finally a high cumulative reward.

In order to have an overall comparison of lifelong learning algorithms on RL, we choose one or two algorithms from the categories mentioned in Sec.2.1. **Oracle** means that each task is trained with an individual policy. Oracle does not consider the lifelong learning problem. It is used to indicate the best possible performance each algorithm can achieve. **Random** means using a random agent in the benchmark tasks. It is used to indicate the worst possible performance for each algorithm. **SAC**[18] means using the original Soft-Actor-Critic algorithm trained continuously from one task to another. However, for the fairness of comparison, we also modify the actor of SAC into shared parts and task-specific parts. It is designed to demonstrate the performance of the state-of-the-art RL algorithms without lifelong learning. **EWC**[1] and **L2** are representatives of regularization-based methods. L2 only introduces an MSE loss for the network differences between the current actor parameters and previous task parameters, as is mentioned in Sec.2.1.1. L2 is the most intuitive lifelong learning algorithm. **AGEM**[10] and **GEM**[9] and **CLOS**[12] are chosen as representatives for gradient-constraint methods. **ER**[7] is chosen for memory-based methods. **APD**[13] is chosen for expansion-based methods. In addition, **APD** and **CLOS** are the state-of-the-art ones to demonstrate the performance of the frontier research on lifelong learning.

We first show the performance of the algorithms on each task after finishing learning the last task. Each algorithm is trained and updated with nearly the same steps for a fair comparison. The environments for the previous tasks are not allowed to re-accessed after finishing the training. All the algorithms are trained with the same task order. The results are shown in Fig.3. The average performance is shown in Table1.

Oracle	Random	SAC	L2	EWC
1709.44	-810.432	-535.57	-618.87	-249.73
AGEM	GEM	ER	CLOS	APD
-286.44	549.52	867.21	1543.90	1379.65

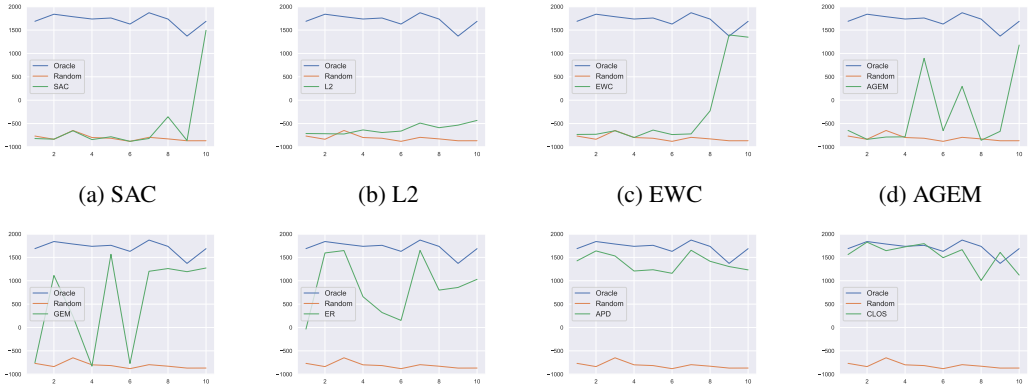
Table 1: Average performance of each lifelong learning algorithm

To better demonstrate the forgetting details of each algorithm, we provided figures of cumulative rewards along with the training procedures. The results are shown in Fig.4. The x-axis denotes training steps. The y-axis denotes cumulative rewards. Each curve shows performances of a policy for a specific task, from before training this task to after training the last task. Specifically, performances before training this task and after training the task are included in the figure to demonstrate the influence of training other tasks on the current task.

A reliable lifelong learning algorithm would not be influenced by the task orders. We train each algorithm in three different task orders and evaluate the policies on all the ten tasks after finishing training the last task. Results are shown in Fig.5

7 Discussion

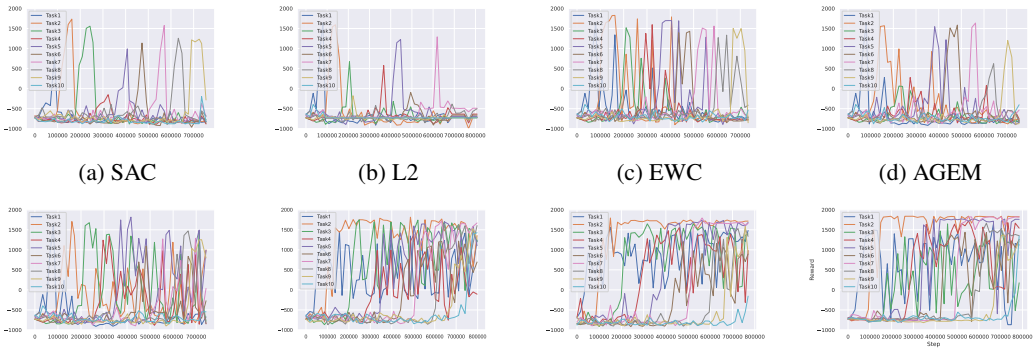
In this section, we analyze the results mentioned in Sec.6.



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

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

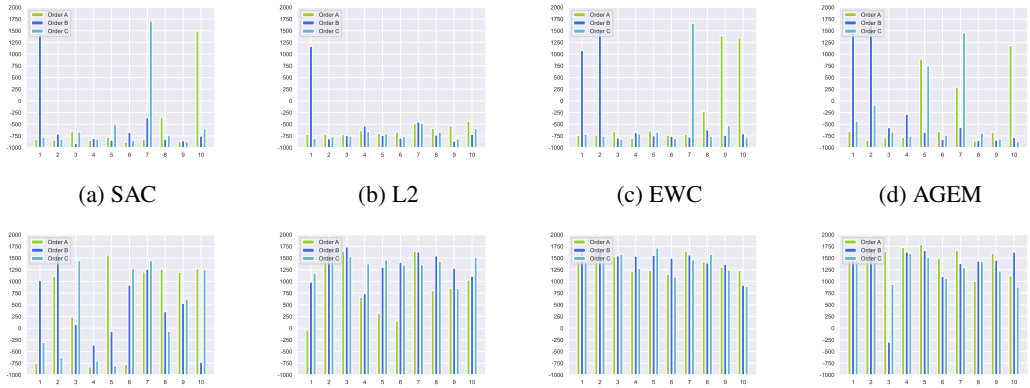
Figure 3: Performance of lifelong learning methods with the RL framework mentioned in Sec.4. The x-axis denotes task IDs. The y-axis denotes cumulative rewards. Each lifelong reinforcement learning algorithm consists of a lifelong learning algorithm and SAC as the RL algorithm. The figure shows the performance of the policy on each task after finishing learning the last one. Oracle and random agent is included in each figure for comparison.



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

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

Figure 4: Performance changes of lifelong learning methods along with the RL training procedure. Different lifelong learning algorithms are trained with the same task order. Each curve demonstrates the cumulative reward of the policy from before training on the task to finishing training the last task.



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

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

Figure 5: Performances of lifelong learning algorithms trained with different task orders.

In Fig.3, SAC only achieves a high reward on the last task. It indicates that traditional RL cannot solve the lifelong tasks designed in our benchmark tasks. Previous tasks still have a slightly higher performance than a random agent, which suggests each task is not totally independent from others. L2 achieves worse results than SAC because L2 is intuitive but a method with poor performance. Improper constraints of lifelong learning, e.g., regularization, is not only detrimental to memorizing previous tasks but will impede the learning of the current task. In comparison, EWC achieves better results, remembering previous tasks to some extent. However, it has a much worse forgetting performance compared to its applications in supervised learning. It could result from the sensitiveness of manipulation tasks mentioned in 1. GEM has better performances than AGEM. Compared to AGEM, GEM samples more data from previous tasks when updating. It could lead to a more conservative update, which is important in the sensitive RL environment. Though simple and intuitive, ER achieves satisfying results. It is worth noting that in RL, we do not need to design additional replay memories since RL algorithms will need to use replay memories themselves. CLOS includes memory-based methods into gradient-based methods by replaying previous tasks when training. It achieves better performances than gradient-based ones, which suggests that memory-based methods could help improve the performance of other lifelong algorithms. Though APD achieves a good result. However, in practice, expansion-based methods will add a sub-module when learning a new task. The size of the added sub-module could be as large as training a new policy from scratch, which degrades the effect of lifelong learning. Note that RL tasks would intensify the weakness of lifelong learning algorithms. Even some could have a good result on supervised learning, It could fail in RL.

In Fig.4, the performance for tasks in SAC only peaks when training it, which means it quickly forgets the policy on previous tasks. The learning curves for ER, APD, CLOS stands for learning curves for successful lifelong RL tasks. The rewards will keep in a high level after it has been trained. CLOS combines several methods together and has the best performance among the algorithms mentioned above. However, the training of CLOS is less stable than APD and ER. Specifically, the performance will vary a lot during the training procedure. One has to save several policies along with the training and search for the best one when using CLOS.

Task order experiments in Fig.5 demonstrate that the algorithms with satisfying performance in the previous experiments would usually also robust to order changes. Among all the algorithms, APD demonstrates the best robustness to task orders, which is reasonable since expanding the network makes it similar to training an independent policy for each task.

We also point out that, unlike supervised learning, even if the cumulative reward of a task could decrease greatly, it does not really mean the policy forgets greatly about the task. Since RL tasks could be highly sensitive and the policy might not be robust, a slight forgetting could lead to a complete failure of the task. Though in most scenarios for RL, we mainly care about the cumulative reward and it would be sufficient to use it to evaluate the performance of lifelong learning, a more comprehensive method of measuring forgetting is needed for RL tasks.

In conclusion, expansion-based methods and memory-based methods have a good performance on lifelong RL methods. Future work could consider incorporating these two methods as a part of its algorithm for a better performance. Regularization-based methods and gradient-based methods still suffer in RL tasks. But on the other hand, the model size is limited in these methods, which has great potential in real robot applications. However, for applications on real robots, it will not be possible for the robot to remember too many samples from previous tasks or will not allow a substantial increase in the model size. Existing lifelong learning methods still do not have a perfect solutions to it.

8 Conclusion

In this paper, we focus on the problem of how the state-of-the-art lifelong learning algorithms perform on RL manipulation tasks. We design a framework to modify traditional lifelong learning algorithms used in supervised learning into the ones in RL. We also develop a manipulation benchmark for evaluation. We test seven lifelong learning algorithms with three baselines on our benchmark. According to our experiments, we point out the sensitiveness challenge of lifelong RL tasks compared to lifelong supervised learning tasks. We also summarize the potentials of applying different kinds of lifelong learning tasks in RL.

References

[1] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[2] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[3] J. L. McClelland, B. L. McNaughton, and R. C. O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

[4] M. Lungarella, G. Metta, R. Pfeifer, and G. Sandini. Developmental robotics: a survey. *Connection science*, 15(4):151–190, 2003.

[5] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[6] X. Y. Lee, S. Ghadai, K. L. Tan, C. Hegde, and S. Sarkar. Spatiotemporally constrained action space attacks on deep reinforcement learning agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4577–4584, 2020.

[7] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

[8] Y. Guo, M. Liu, T. Yang, and T. Rosing. Improved schemes for episodic memory-based lifelong learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[9] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6470–6479, 2017.

[10] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations*, 2018.

[11] M. Farajtabar, N. Azizan, A. Mott, and A. Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR, 2020.

[12] A. Chaudhry, N. Khan, P. Dokania, and P. Torr. Continual learning in low-rank orthogonal subspaces. *Advances in Neural Information Processing Systems*, 33, 2020.

[13] J. Yoon, S. Kim, E. Yang, and S. J. Hwang. Scalable and order-robust continual learning with additive parameter decomposition. In *Eighth International Conference on Learning Representations, ICLR 2020*. ICLR, 2020.

[14] H. Caselles-Dupré, M. Garcia-Ortiz, and D. Filliat. S-trigger: Continual state representation learning via self-triggered generative replay. *arXiv preprint arXiv:1902.09434*, 2019.

[15] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pages 4528–4537. PMLR, 2018.

[16] R. Traoré, H. Caselles-Dupré, T. Lesort, T. Sun, N. Díaz-Rodríguez, and D. Filliat. Continual reinforcement learning deployed in real-life using policy distillation and sim2real transfer. *arXiv preprint arXiv:1906.04452*, 2019.

[17] D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, and G. Wayne. Experience replay for continual learning. *arXiv preprint arXiv:1811.11682*, 2018.

[18] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[19] 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.

[20] S. Fujimoto, H. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

[21] M. Ahn, H. Zhu, K. Hartikainen, H. Ponte, A. Gupta, S. Levine, and V. Kumar. Robel: Robotics benchmarks for learning with low-cost robots. In *Conference on Robot Learning*, pages 1300–1313. PMLR, 2020.