On-Policy Algorithms for Continual Reinforcement Learning

Supplementary material

Anonymous submission

CRL Methods

In both PPO and SAC we regularize only the actor network with CL methods.

Fine-tuning Fine-tuning serves as the baseline approach. With fine-tuning, the PPO agent learns each task sequentially without resetting the neural network. After completing a task, the network continues training on the next task. No specific continual learning techniques are applied in this baseline approach, providing a simple framework for assessing how well the agent adapts to new tasks while retaining previously learned information.

L2 regularization L2 regularization is used to mitigate catastrophic forgetting by encouraging the network to retain previously learned knowledge. Specifically, it adds L_2 penalty to the loss function, which regularizes the model by discouraging large deviations from the weights learned on previous tasks. This constraint ensures that as the PPO agent learns new tasks and does not stray too far from the parameters optimized for earlier tasks.

EWC Elastic weight consolidation (EWC) (Kirkpatrick et al. 2017) is implemented to address catastrophic forgetting and forward transfer. It introduces a penalty to the training process that prevents significant changes to important weights associated with previous tasks. It uses the Fisher Information Matrix to identify which weights are crucial for earlier tasks and keeps these weights stable while training on new tasks.

PackNet PackNet (Mallya and Lazebnik 2018) is a technique to handle catastrophic forgetting and forward transfer. It works by pruning, or removing, parts of the neural network after each task, which makes the network more efficient for the current task. It then retrains the remaining parts for new tasks while keeping the old weights unchanged. This method allows the network to handle multiple tasks without overwriting what it has already learned.

CRL Metrics

Average performance Let $p_i(t)$ denote the success rate of the model on task i at time t, which is defined as the average number of times the agent successfully reaches the goal of task i during the evaluation phase with stochastic policy at time t. The average performance is then defined as the mean success rate across all N tasks at a final evaluation time T:

$$\text{AP} = \frac{1}{N}\sum_{i=1}^{N} p_i(T).$$

Forward transfer We measure the forward transfer of a task as a normalized area between the training curve of a particular task (trained in a sequence) and the training curve of the reference (a single-task experiment). Let $p_i^b(t) \in [0, 1]$ be the success rate of the model on the reference of task i and $p_i(t)$ be the success rate of the model on task i at time t. The forward transfer for the task i, denoted by FT_i, is formally defined as:

$$FT_i := \frac{\text{AUC}_i - \text{AUC}_i^b}{1 - \text{AUC}_i^b},$$

where:

$$\text{AUC}_i := \frac{1}{\Delta}\int_{(i-1)\cdot\Delta}^{i\cdot\Delta} p_i(t)\,dt,$$

$$\text{AUC}_i^b := \frac{1}{\Delta}\int_{0}^{\Delta} p_i^b(t)\,dt.$$

The average forward transfer for the entire sequence of tasks then defined as follows:

$$FT = \frac{1}{N}\sum_{i=1}^{N} FT_i.$$

Forgetting To quantify forgetting for task i, we measure the difference between the success rate on the task at the end of its training and the success rate on that task at the end of whole learning process, i.e.:

$$F_i = p_i(i \cdot \Delta) - p_i(T).$$

The average forgetting for the entire sequence of tasks is then defined as follows:

$$F = \frac{1}{N}\sum_{i=1}^{N} F_i.$$

Parameter	Value range	Selected value
optimizer	–	Adam
clipping ratio	–	0.2
target KL divergence	–	0.01
learning rate	{5e-5, 1e-4}	1e-4
number of policy updates	{80, 256}	80
number of value function updates	{80, 256}	80
(L2) regularization coefficient	$\{10^{-1}, 1, 10, 10^2, 10^3, 10^4, 10^5, 10^6\}$	10^4
(EWC) regularization coefficient	$\{10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3, 10^4, 10^5\}$	1
(PackNet) number of policy updates during retraining	$\{0, 10, 10^2, 10^3\}$	10
(PackNet) number of value updates during retraining	$\{0, 10, 10^2, 10^3\}$	10

Table 1: Hyperparameters for PPO algorithm and their selected values

Parameter	Value range	Selected value
optimizer	–	Adam
replay buffer size	–	10^6
learning rate	{3e-5, 1e-4, 3e-4, 1e-3}	3e-4
batch size	{128, 256}	256
(L2) regularization coefficient	$\{10^{-1}, 1, 10, 10^2, 10^3, 10^4, 10^5, 10^6\}$	10^2
(EWC) regularization coefficient	$\{10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3, 10^4, 10^5\}$	10^2
(PackNet) number of updates during retraining	$\{10, 10^2, 10^3, 10^4, 10^5, 10^6\}$	10^3

Table 2: Hyperparameters for SAC algorithm and their selected values

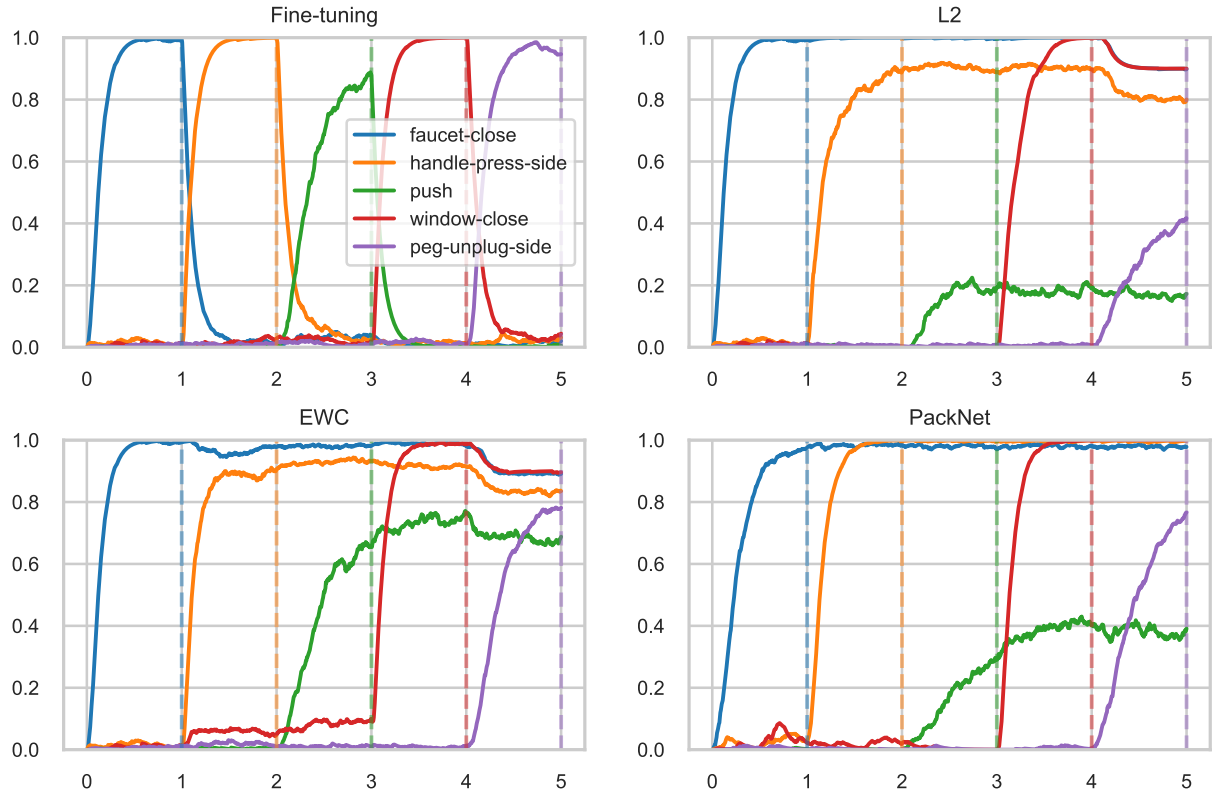

Figure 1: Comparison of a success rate obtained by SAC across fine-tuning and three different CL methods (L2 regularization, EWC, and PackNet) on a sequence of $N = 5$ tasks.

Experimental details

Tasks To evaluate our methods we use Continual World (see (Wołczyk et al. 2021)) benchmark specifically designed to evaluate RL agents on challenges that resemble properties of continual learning. It is constructed around a sequences of realistic robotic manipulation tasks from Meta-World (Yu et al. 2020). In our experiments, we use a sequence of five following tasks:

1. `faucet-close-v2`: the agent learns to close a faucet,
2. `handle-press-side-v2`: the agent must press a handle down sideways,
3. `push-v2`: the agent is tasked with pushing a puck to the goal location,
4. `window-close-v2`: the agent needs to push close a window,
5. `peg-unplug-side-v2`: the agent must unplug a peg sideways.

Hyperparameters For both PPO and SAC algorithms the hyperparameter search was performed in two stages. In the first stage, for fine-tuning, the success rate of each task is evaluated at the end of its training, and the mean success rate is then calculated across all tasks. Then, in the second stage, for each CL method, we tuned the method-specific hyperparameters for the average performance metric, while keeping the hyperparameters selected in the first stage fixed (in practice we took last 10 evaluation checkpoints for stability). The hyperparameter search space as well as optimal parameters for both algorithms are presented in Tab. 1 and Tab. 2.

Network architecture For PPO we use a network with 2 linear layers, each consisting of 256 neurons. After the first layer we apply the tanh activation. For SAC we use a network with 4 linear layers, each consisting of 256 neurons. After the first layer apply the tanh activation and after next layers we use leaky ReLU activations (with $\alpha = 0.2$). The architectures are the same for both actor and critic in both algorithms. Each network has five heads, each one corresponding to different task of the sequence.

Evaluation For both algorithms, we evaluate the policy every 5000 steps, using the mean of 10 stochastic trajectories. All metrics were evaluated on 20 different random seeds.

SAC In all SAC experiments, we reset the replay buffer on task changes. Success rates for all methods are presented in Fig. 1.

References

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

Mallya, A.; and Lazebnik, S. 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 7765–7773.

Wołczyk, M.; Zajac, M.; Pascanu, R.; Łukasz Kuciński; and Miłoś, P. 2021. Continual World: A Robotic Benchmark For Continual Reinforcement Learning. arXiv:2105.10919.

Yu, T.; Quillen, D.; He, Z.; Julian, R.; Hausman, K.; Finn, C.; and Levine, S. 2020. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, 1094–1100. PMLR.