Rewiring Neurons in Non-Stationary Environments

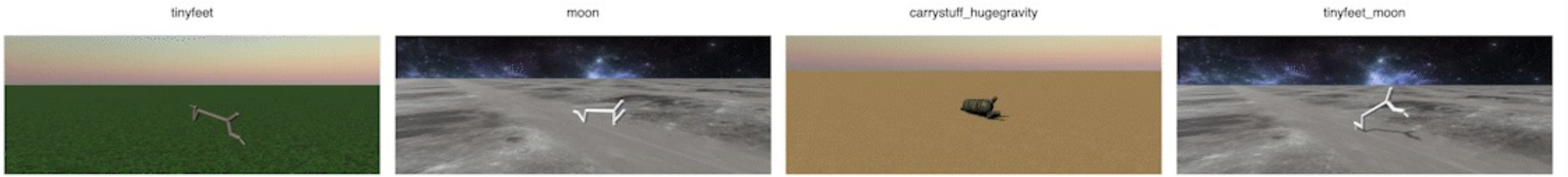
Zhicheng Sun, Yadong Mu

Peking University, Beijing, China

Introduction

Problem description

- Continual reinforcement learning[1] concerns learning over non-stationary environments
- It requires our policy network to quickly adapt to environmental changes[2] while not catastrophically forgetting[3] the learned policy

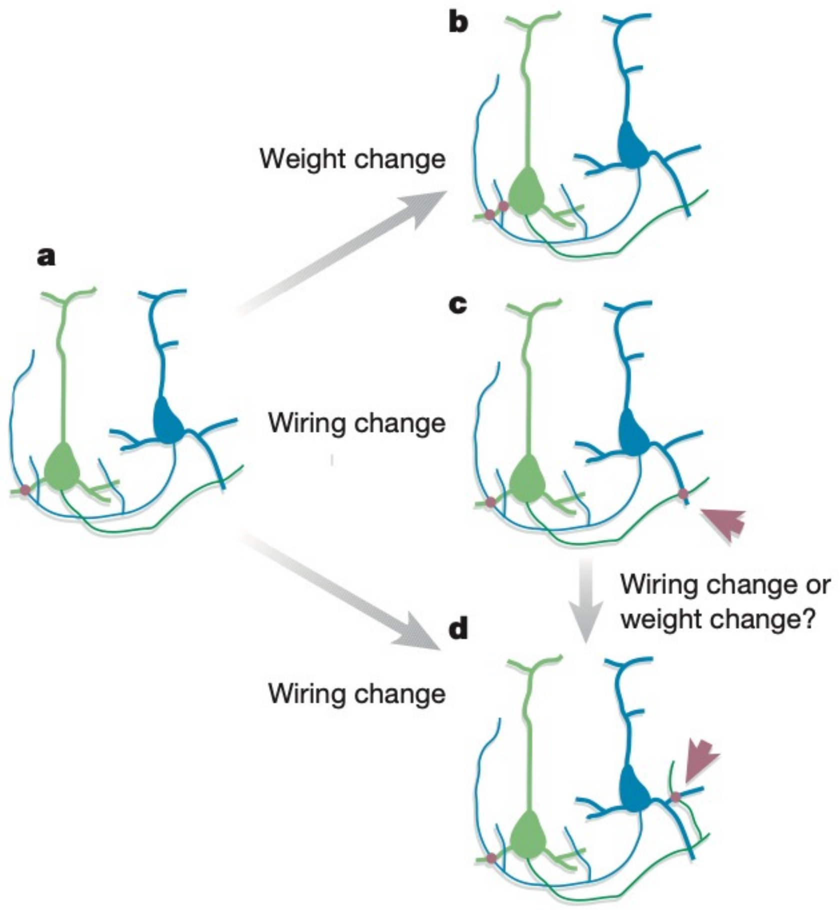


[1] Mark B Ring. "Continual learning in reinforcement environments". PhD thesis, University of Texas at Austin, 1994.
[2] Khimya Khetarpal, Matthew Riemer, Irina Rish et al. "Towards Continual Reinforcement Learning: A Review and Perspectives". JAIR, 2022, 75: 1401–1476.
[3] Michael McCloskey and Neal J Cohen. "Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem". Psychology of Learning and Motivation, 1989, 24: 109–165.

Introduction

Motivation

- Continual reinforcement learning requires the policy network to quickly adapt to new environments[1]
- We are inspired by the brain's remarkable adaptivity by rewiring itself[2] and seek to incorporate a similar process into the policy network

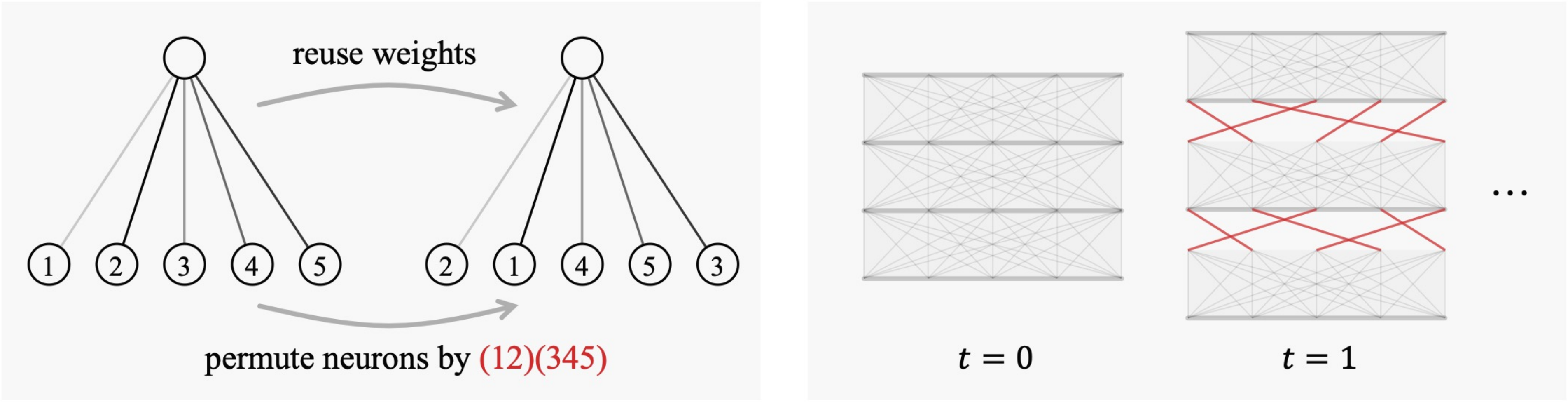


[1] Khimya Khetarpal, Matthew Riemer, Irina Rish et al. "Towards Continual Reinforcement Learning: A Review and Perspectives". JAIR, 2022, 75: 1401–1476.
[2] Dmitri B Chklovskii, BW Mel and K Svoboda. "Cortical Rewiring and Information Storage". Nature, 2004, 431(7010): 782–788.

Method

Rewiring via permutation

- By exploiting the layered structure of the network, it fully reuses existing synapses to achieve structural plasticity in continual learning



(a) Synaptic level

(b) Network level

Method

Rewiring via permutation

- Rewire between layers by permuting hidden neurons

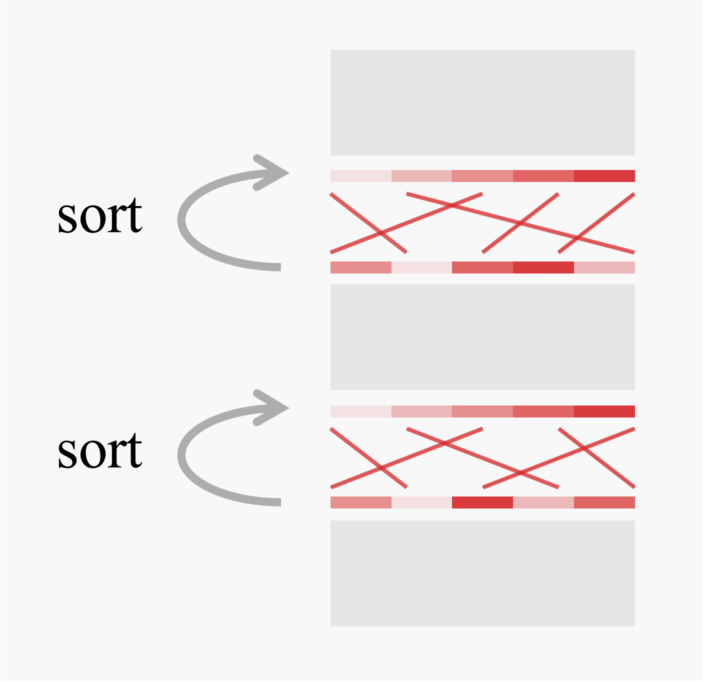
$$Y = W_L \circ \sigma \circ P_{L-1} W_{L-1} \circ \ldots \circ \sigma \circ P_1 W_1 X.$$

end-to-end learnable via differentiable sorting[1]

$$P_l = I[z_l, :], \quad z_l = \mathrm{argsort}(v_l),$$

$$\hat{P}_l = \mathrm{softmax}\left(\frac{-d(\mathrm{sort}(v_l)\mathbf{1}^\top, \mathbf{1}v_l^\top)}{\tau}\right),$$

Advantages: highly parameter-efficient, exploit numerous structural variations



[1] Sebastian Prillo and Julian Eisenschlos. “SoftSort: A Continuous Relaxation for the argsort Operator”. In: ICML. 2020: 7793–7802.

Method

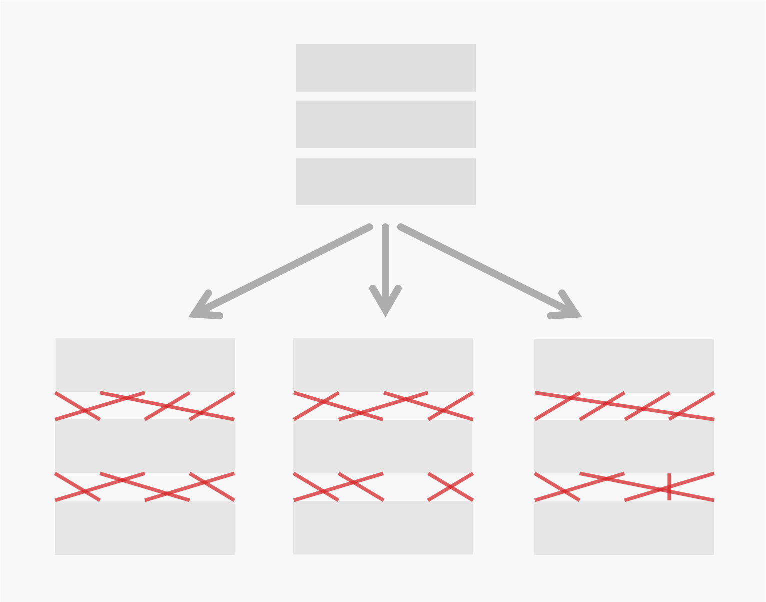
Rewiring for exploration

- Maintain a set of wirings and randomly sample from these wirings at each step to generate diverse policies

$$P_l \in \{P_{l,1}, P_{l,2}, \ldots, P_{l,K}\}.$$

- Distill knowledge[1] across wirings for knowledge sharing

$$L_{\mathrm{KL}}(W, P) = \mathbb{E}_{k' \neq k}\left[D_{\mathrm{KL}}\left(\pi_{k'}(\cdot|s) \,\|\, \pi_k(\cdot|s)\right)\right],$$



[1] Geoffrey Hinton, Oriol Vinyals and Jeff Dean. "Distilling the Knowledge in a Neural Network". arXiv preprint arXiv:1503.02531, 2015.

Method

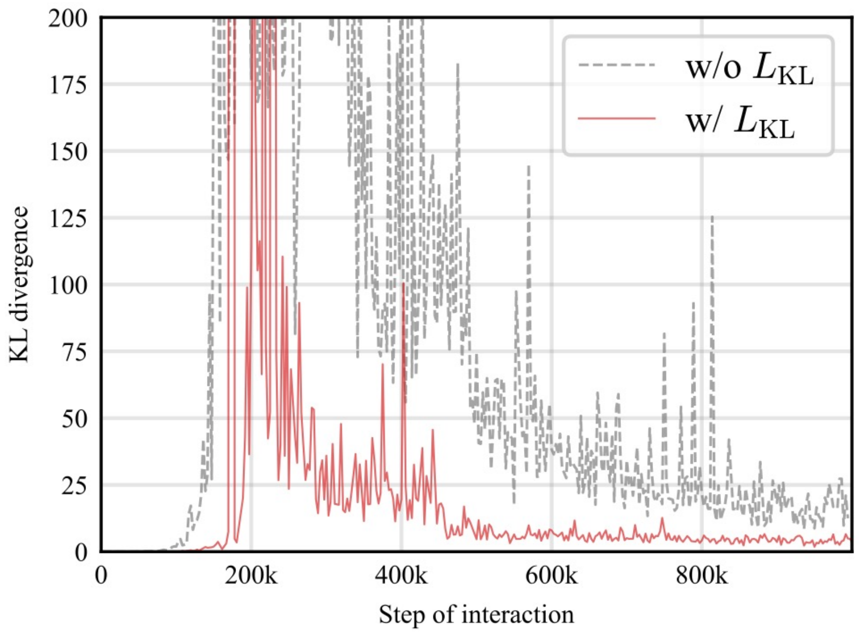
Rewiring for exploration

- Maintain a set of wirings and randomly sample from these wirings at each step to generate diverse policies

$$P_l \in \{P_{l,1}, P_{l,2}, \ldots, P_{l,K}\}.$$

- Distill knowledge[1] across wirings for knowledge sharing

$$L_{\mathrm{KL}}(W, P) = \mathbb{E}_{k' \neq k}\left[D_{\mathrm{KL}}\left(\pi_{k'}(\cdot|s) \,\|\, \pi_k(\cdot|s)\right)\right],$$



[1] Geoffrey Hinton, Oriol Vinyals and Jeff Dean. “Distilling the Knowledge in a Neural Network”. arXiv preprint arXiv:1503.02531, 2015.

Method

Rewiring for stability-plasticity

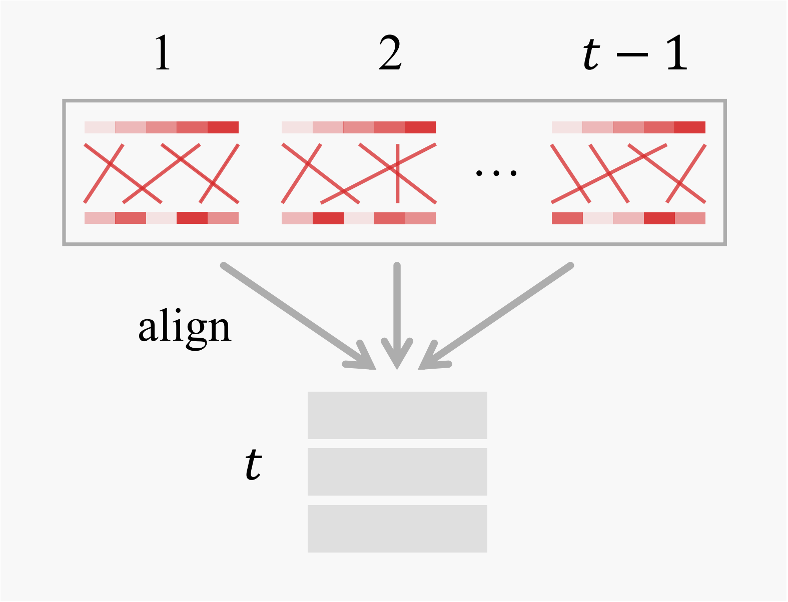
- Cache each learned wiring while regularizing the weight changes

$$L_{\text{reg}}(W^t) = \sum_{l=1}^{L} \|W_l^t - W_l^{t-1}\|^2.$$

- Jointly refine the wiring and the weights to align with each other.

$$Y = \ldots \circ \sigma \circ \underbrace{P_l' P_l^{t-1} P_l''^{\top}}_{\text{adapters on } P_l^{t-1}} W_l^t \circ \ldots X.$$

$$L_{\text{SP}}(W^t, P', P'') = \sum_{l=1}^{L} \|W_l^t - P_l'' W_l^{t-1} P_{l-1}'^{\top}\|^2.$$



Experiments

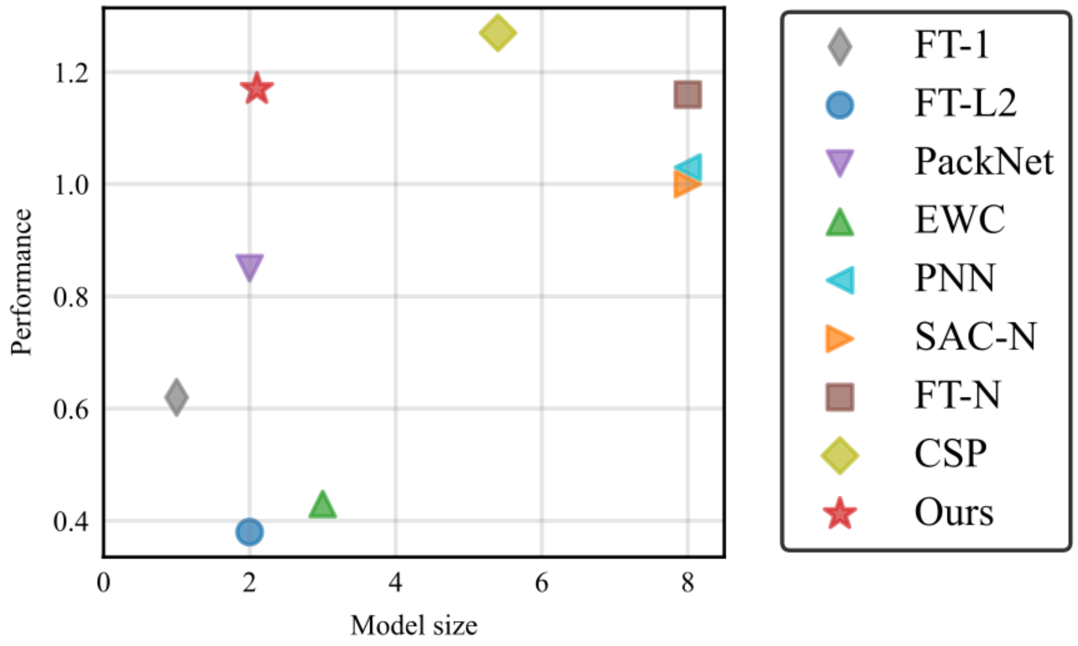
- Average performance (↑) and model size (↓) on Brax scenarios[1,2]

Method	HalfCheetah		Ant		Humanoid	
	Performance	Model size	Performance	Model size	Performance	Model size
FT-1	0.62 ± 0.29	**1.0**	0.52 ± 0.26	**1.0**	0.71 ± 0.07	**1.0**
FT-L2	0.38 ± 0.15	2.0	0.78 ± 0.20	2.0	0.68 ± 0.28	2.0
PackNet [41]	0.85 ± 0.14	2.0	1.08 ± 0.21	2.0	0.96 ± 0.21	2.0
EWC [33]	0.43 ± 0.24	3.0	0.55 ± 0.24	3.0	0.94 ± 0.01	3.0
PNN [54]	1.03 ± 0.14	8.0	0.98 ± 0.31	8.0	0.98 ± 0.26	4.0
SAC-N	1.00 ± 0.15	8.0	1.00 ± 0.38	8.0	1.00 ± 0.29	4.0
FT-N	1.16 ± 0.20	8.0	0.97 ± 0.20	8.0	0.65 ± 0.46	4.0
CSP [20]	$\mathbf{1.27} \pm \mathbf{0.27}$	5.4	1.11 ± 0.17	3.9	1.76 ± 0.19	3.4
Ours	1.17 ± 0.15	2.1	$\mathbf{1.22} \pm \mathbf{0.11}$	2.1	$\mathbf{1.78} \pm \mathbf{0.22}$	2.0

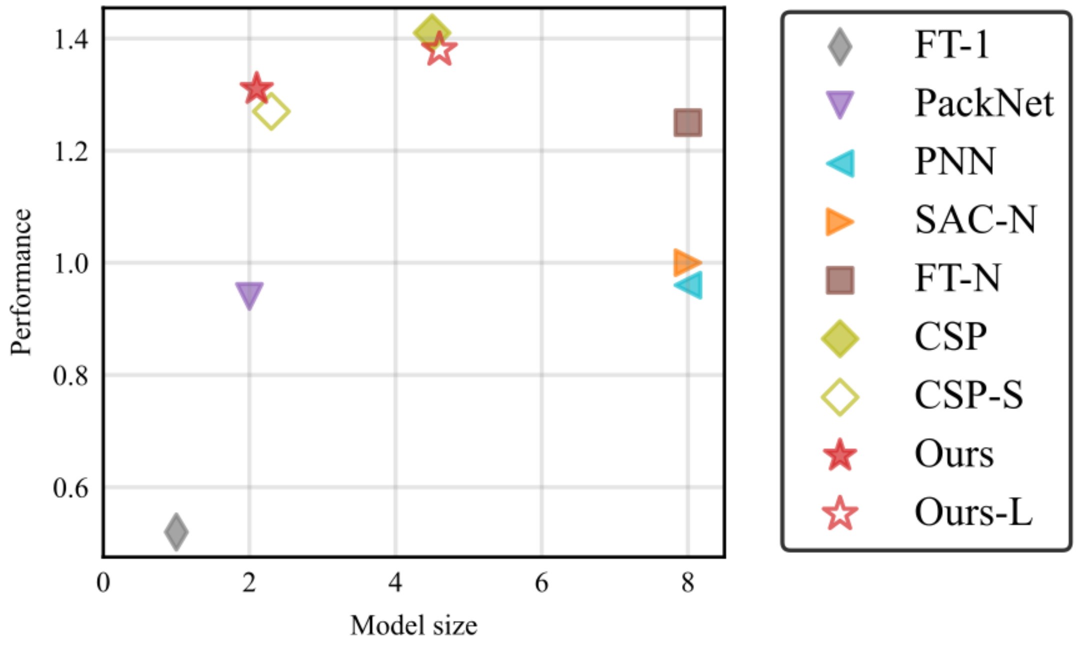
[1] Jean-Baptiste Gaya, Thang Doan, Lucas Caccia et al. "Building a Subspace of Policies for Scalable Continual Learning". In: ICLR. 2023.
[2] C Daniel Freeman, Erik Frey, Anton Raichuk et al. "Brax–A Differentiable Physics Engine for Large Scale Rigid Body Simulation". arXiv preprint arXiv:2106.13281, 2021.

Experiments

- Performance-size tradeoffs on the HalfCheetah scenarios



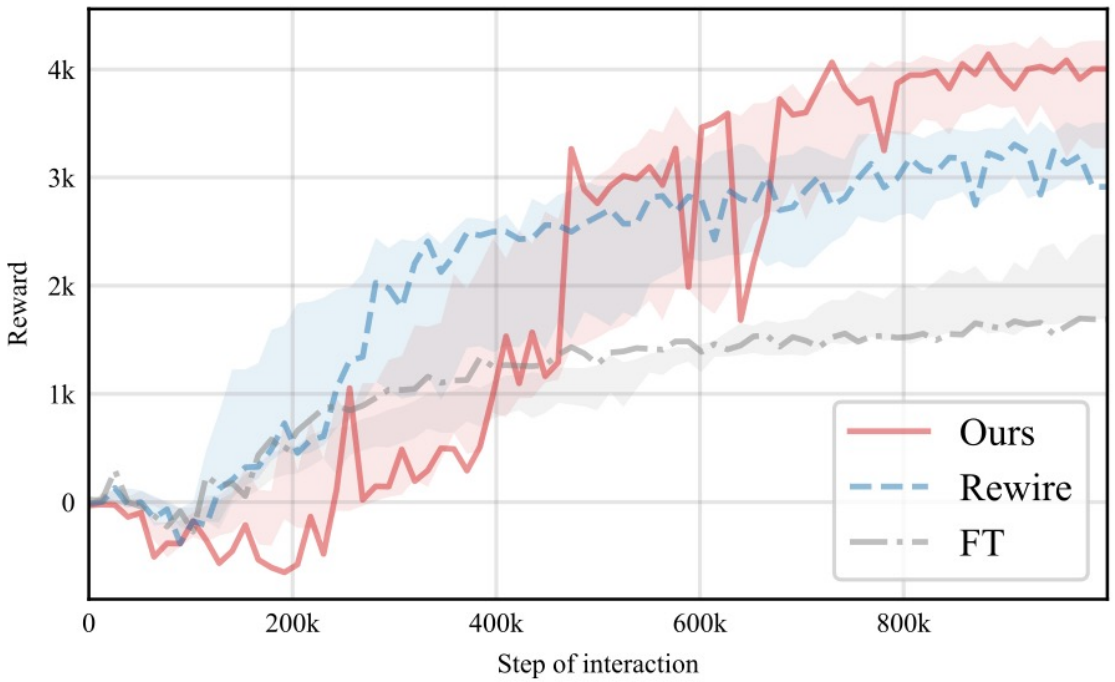
(a) HalfCheetah scenarios



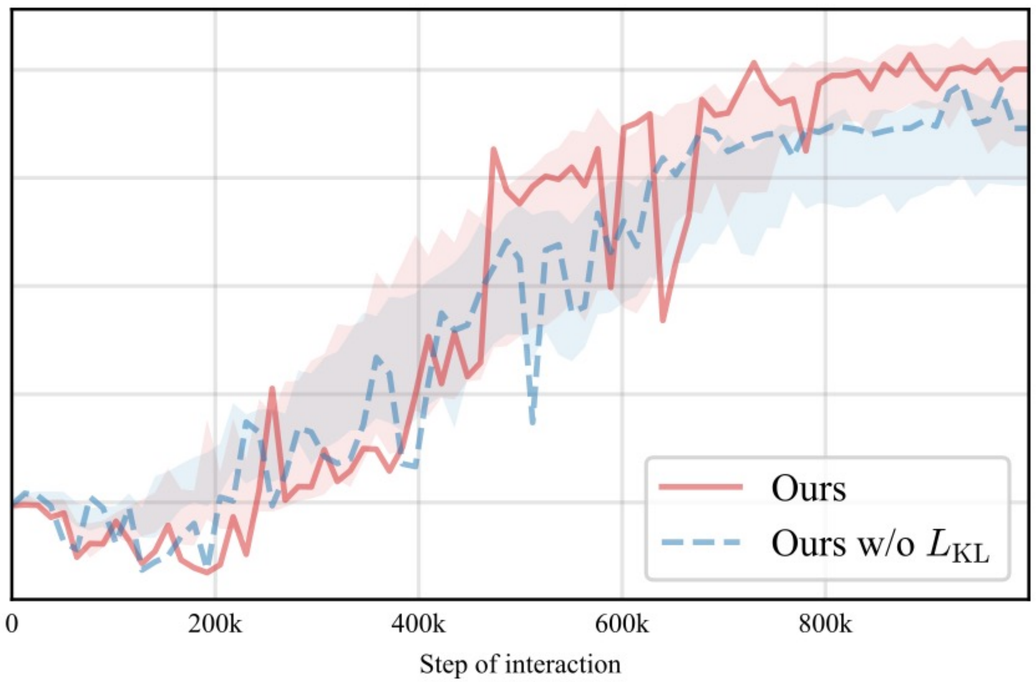
(b) HalfCheetah/forgetting scenario

Experiments

- Evolution of performance in the first stage of HalfCheetah/forgetting scenario



(a) Effectiveness of rewiring and multi-mode



(b) Effectiveness of the distillation loss L_{KL}

Thanks for listening

Code is available at https://github.com/feifeiobama/RewireNeuron