Catastrophic Interference in RL

Donghu Kim

Catastrophic Interference

= Catastrophic Forgetting

= Networks trained on new task “forget” how to complete previously solved tasks.

McCloskey & Cohen (1989)

1. Train a neural network to recognize pattern [A-B].
2. Train a neural network to recognize pattern [A-C].
3. Accuracy on pattern [A-B] drops to near 0.
4. Neurons were activated similarly to [A-C], even when given [A-B] inputs.
5. When trained on [A-B] and [A-C] simultaneously, neural network was able to learn both.

[1] https://en.wikipedia.org/wiki/Catastrophic_interference#The_Sequential_Learning_Problem:_McCloskey_and_Cohen_(1989)

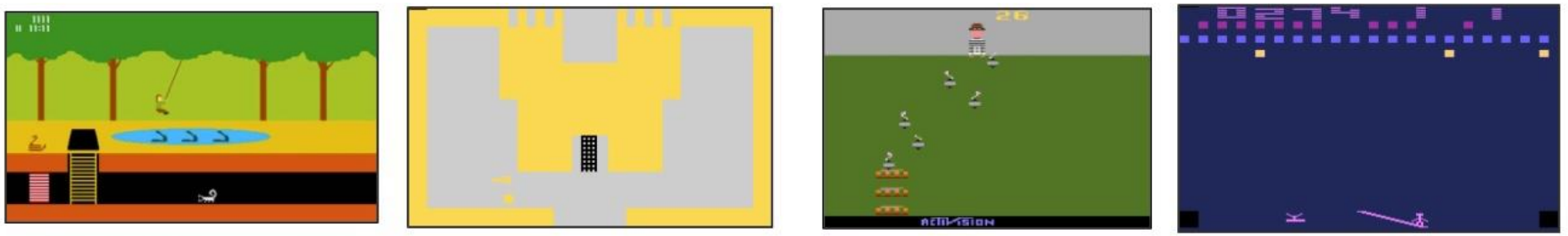
Catastrophic Interference

= One end of plasticity-stability dilemma.

Catastrophic Interference

In RL, often investigated in a sequence of environments/tasks.

Train sequentially…



Evaluate on all games.

Catastrophic Interference: Case Study

Montezuma's Revenge

Extremely challenging exploration + sparse reward.

Each room often has a completely different layout.

Rainbow (+ intrinsic reward (CTS)) plateaus at a certain point… why?

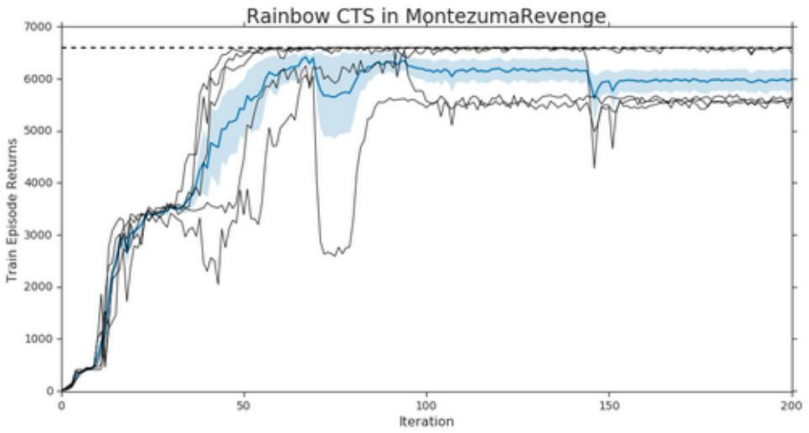
Is it the lack of exploration? → No.

Is it the loss of plasticity? → No.

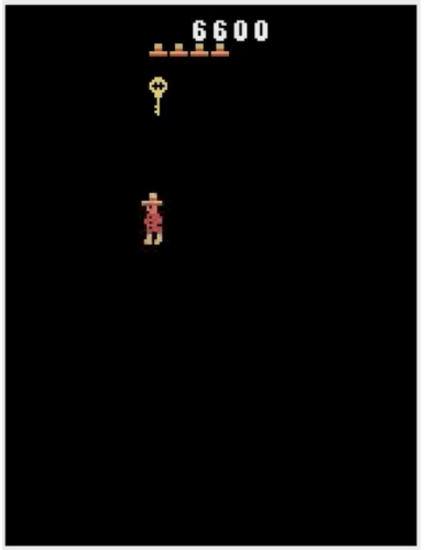
Is it the lack of model capacity? → No.

Is it the lack of training time? → No.

Is it the algorithm's fault? → Probably not.



(a) Baseline Rainbow-CTS agent achieves a maximum achieved score of 6600 (5 seeds).



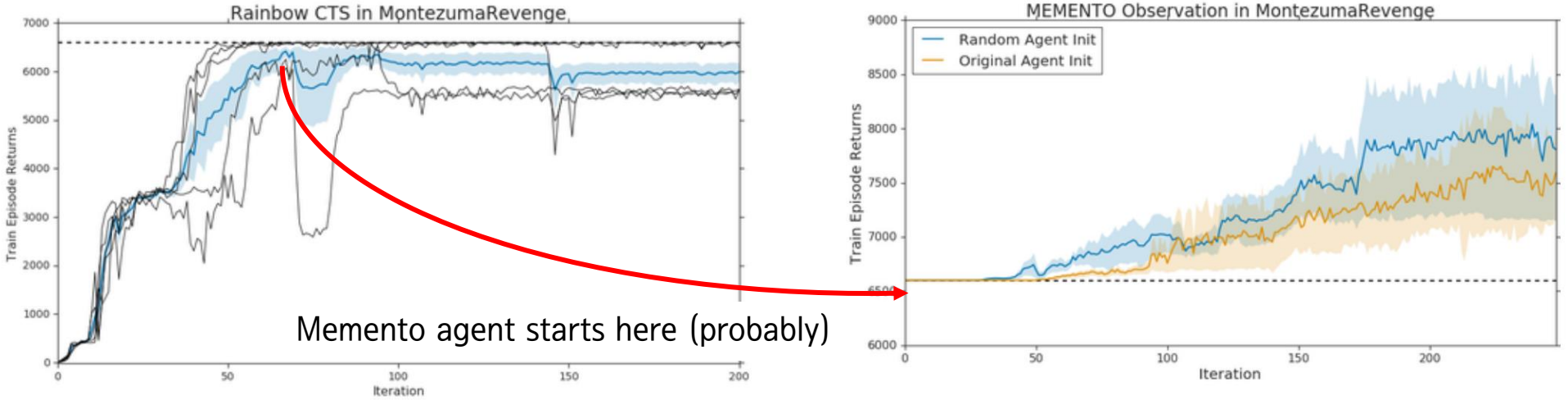
Catastrophic Interference: Case Study

Memento experiment

Start training a new agent where the old agent has converged.

Even when the new agent is a copy of the old agent, performance greatly improves.

→ Old agent was unable to overcome new rooms, because learning it *interfered* with earlier rooms!



(a) Baseline Rainbow-CTS agent achieves a maximum achieved score of 6600 (5 seeds).

(b) A *non-interfering* agent, whether randomly initialized (blue) or initialized with weights of the original model (orange) make further progress.

Catastrophic Interference in a Single Environment

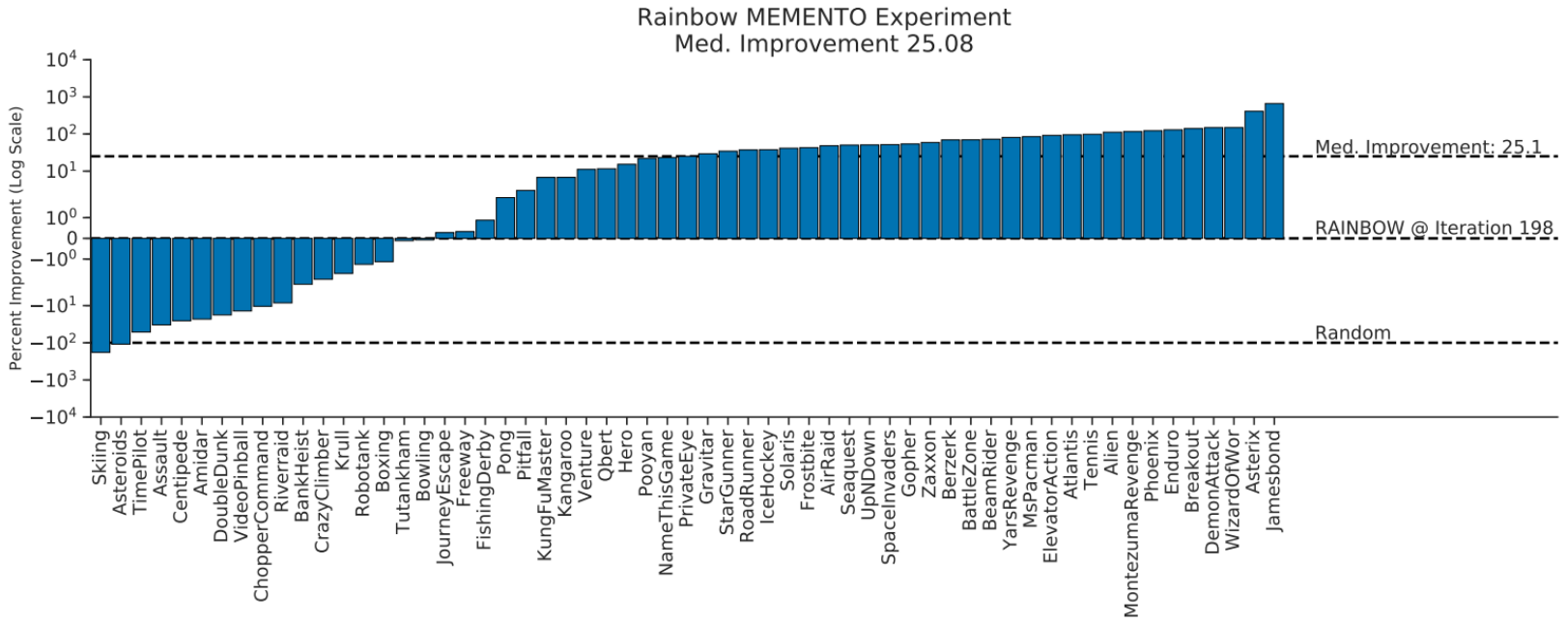
Was the same process effective in other Atari games?

Yes, most of them.

Effective on both Rainbow (w/o CTS) and DQN.

Even seemingly monotonous games (e.g., Breakout) had catastrophic interference.

Note: Memento agent's starting point is chosen heuristically, so performance could be improved.

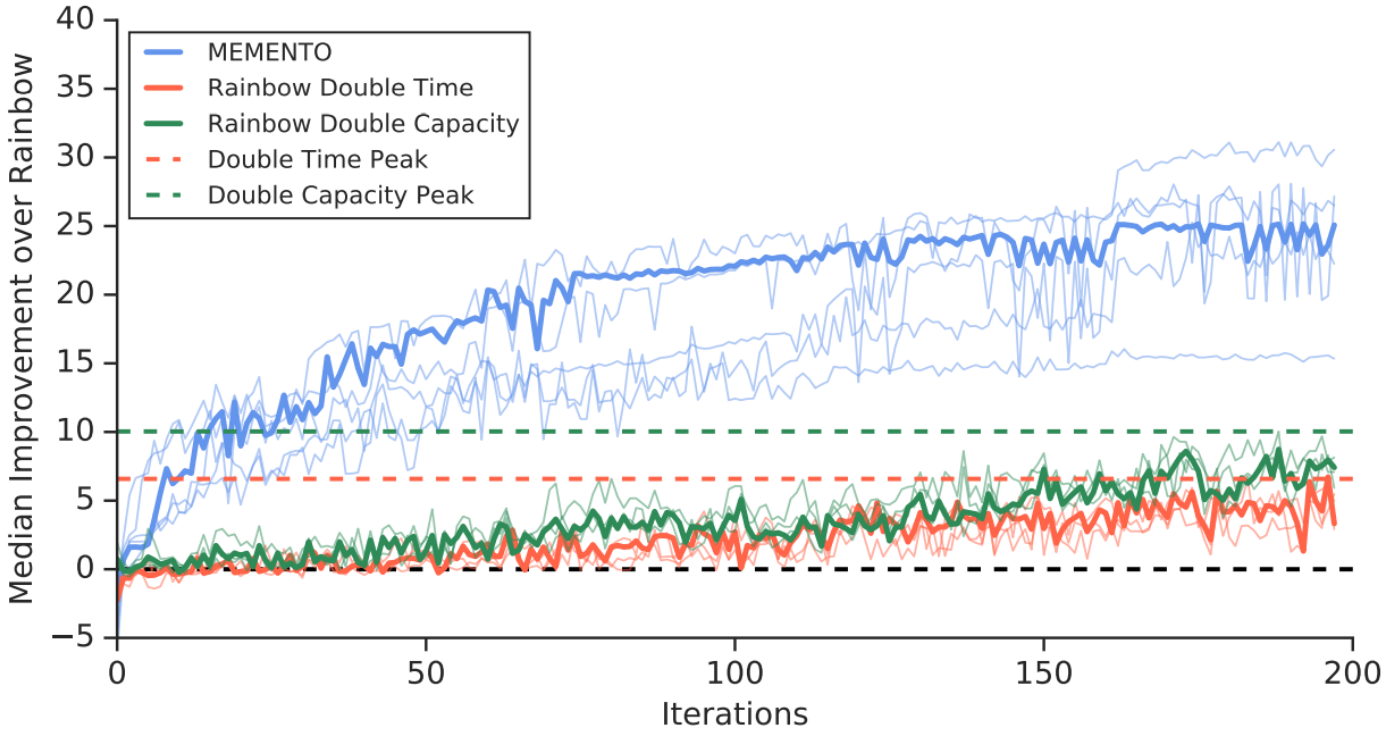


Catastrophic Interference in a Single Environment

Was the same process effective in other Atari games?

Yes, most of them.

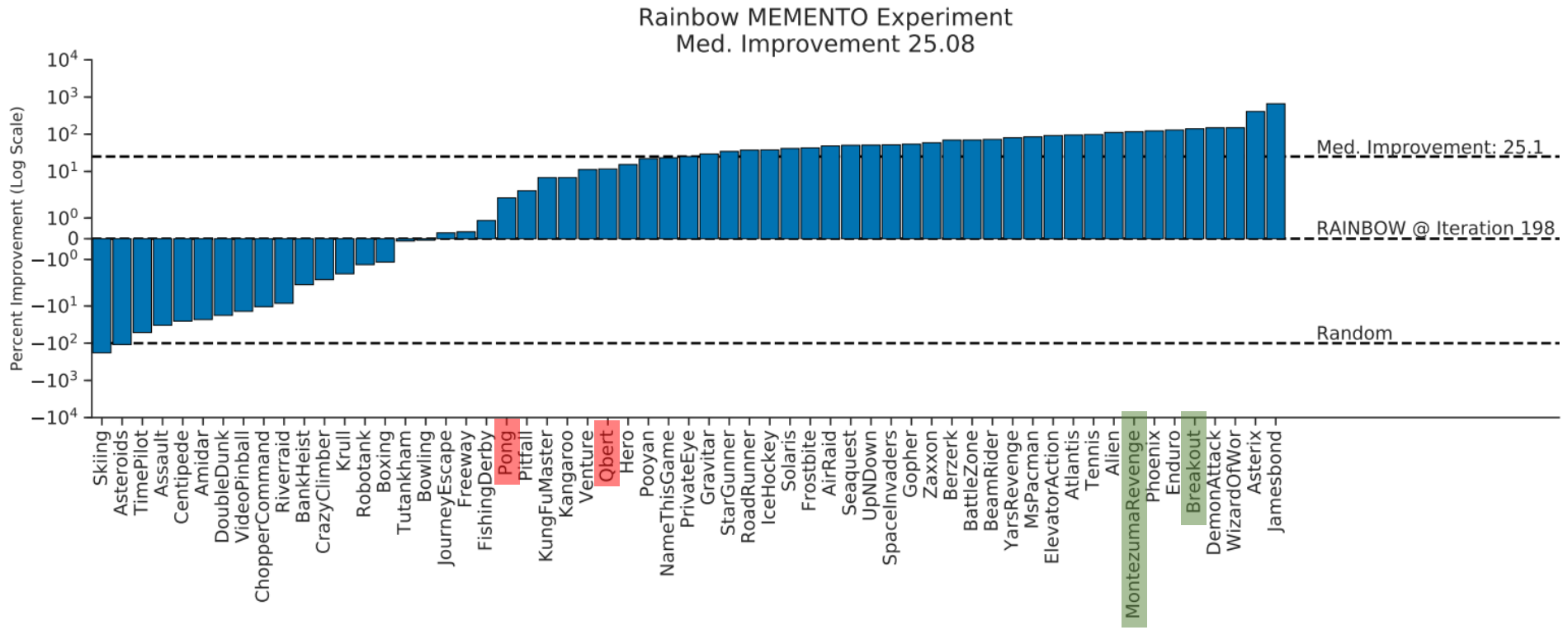
And they were not a model capacity or training time problem.



High Interference vs Low Interference

Games that benefit from Memento vs Games that don't.

Montozuma's Revegence & Breakout vs Pong & Qbert.

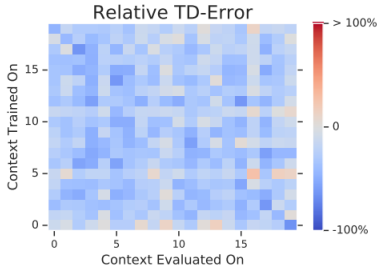


High Interference vs Low Interference

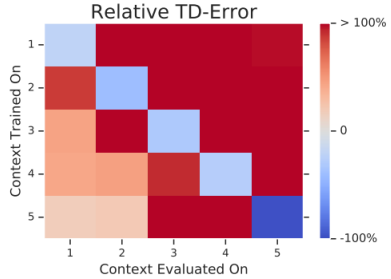
Split the game into segments, based on score range (e.g., 0~1000 / 1000~2000 / ⋯)

Not perfect, but each segment will contain different state/task distribution.

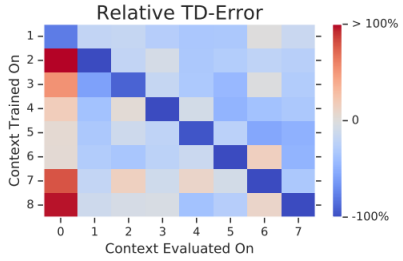
Interference can be measured by training on data from one segment, and seeing how much the TD error in other segments increase.



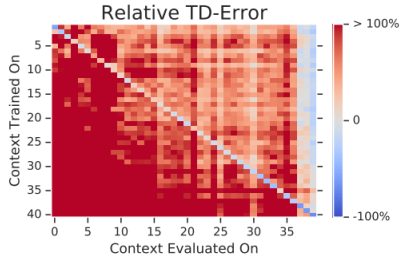
(a) PONG



(b) MONTEZUMA'S REVENGE



(c) QBERT.



(d) BREAKOUT.

Training on one segment generalizes to others

Training on one segment causes forgetting

How Can We Combat Catastrophic Interference?

How Can We Combat Catastrophic ~~Interference~~? Forgetting

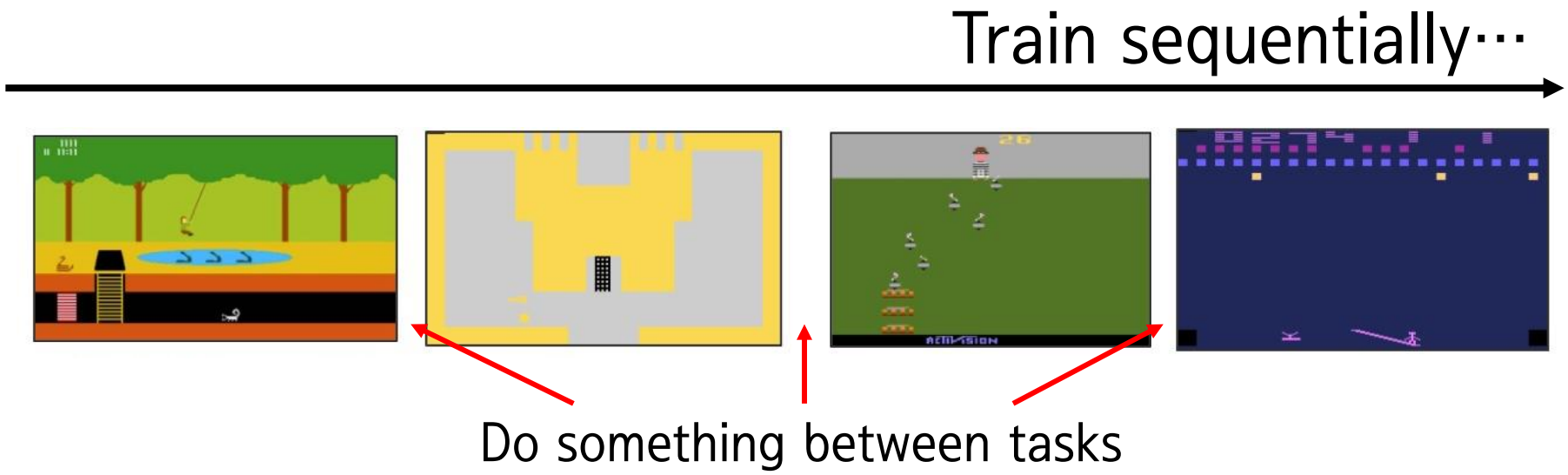
같다고는 했지만 굳이 구분하면…
Interference = 두 task를 sequential 하게 학습할 수 없는 상태
Forgetting = 나중 task를 억지로 배움으로써 기존 task를 잊어버리는 현상

Combatting Catastrophic Forgetting

Majority of works are focused on sequential task setting.

Often, the agent is given exactly what task it's in and when it changes.

Most of these are not directly applicable to single environment, but at least we can get some inspirations.



Combatting Catastrophic Forgetting

A Comprehensive Survey of Forgetting in Deep Learning Beyond Continual Learning

Zhenyi Wang, Enneng Yang, Li Shen, Heng Huang

Continual RL approach. The existing continual RL methods can be categorized into four main groups: (1) regularization-based methods. These approaches employ techniques such as knowledge distillation to alleviate forgetting [202], (2) rehearsal-based methods. These methods utilize rehearsal or experience replay to mitigate forgetting [203], (3) architecture-based methods. These approaches focus on learning a shared structure, such as network modularity or composition, to facilitate continual learning [204] and (4) meta-learning-based methods [179].

[1] A Comprehensive Survey of Forgetting in Deep Learning Beyond Continual Learning., Wang et al., CVPR 2023.

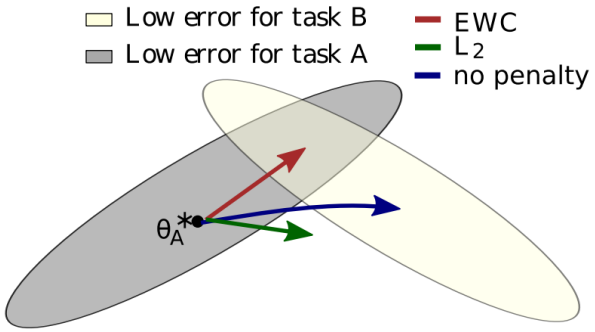
Regularization Based Method

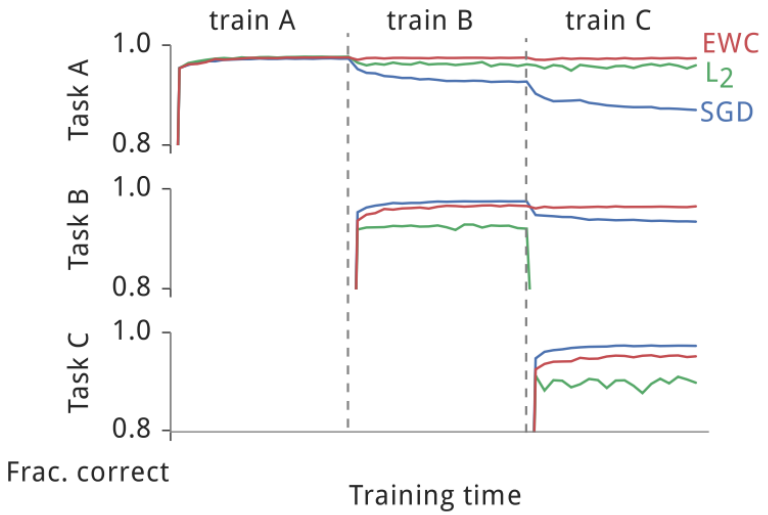
Regularize future training on past parameters to prevent forgetting.

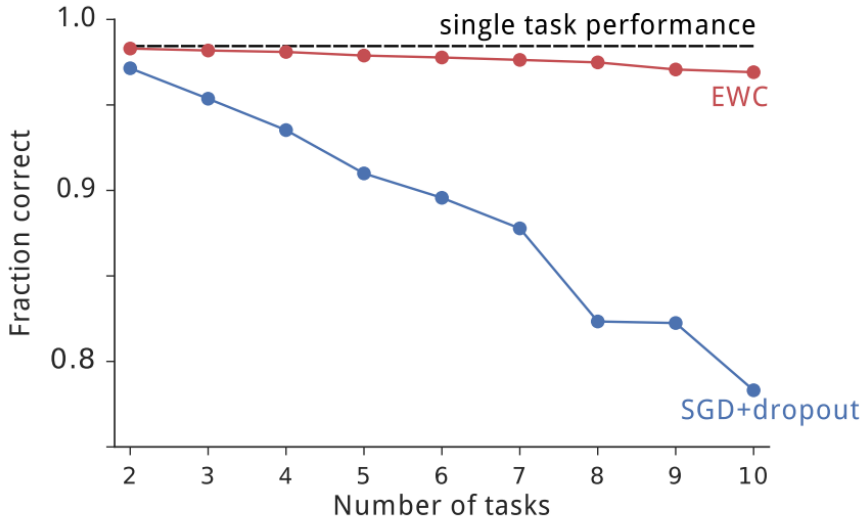
e.g., Elastic Weight Consolidation (EWC) → L2 distance towards old parameters.

$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i})^2$$

Parameters from previous task







[1] Overcoming catastrophic forgetting in neural networks., Kirkpatrick et al., PNAS 2017.

Regularization Based Method

Regularize future training on past parameters to prevent forgetting.

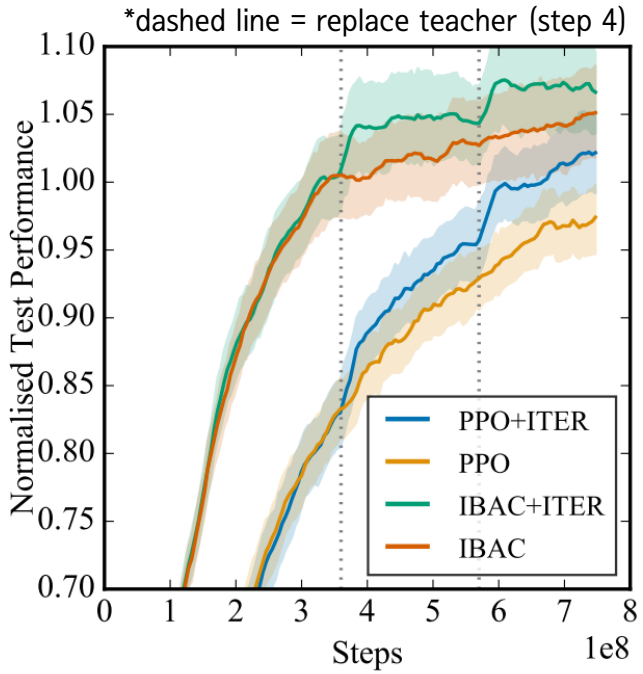
e.g., Iterative Relearning (ITER) → Distillation based. Use old parameter as teacher, distill to new one.

$$\mathcal{L}_\pi(\theta_{k+1}) = \mathbb{E}_{s\sim d_{\pi^{(k)}}}\left[D_{\mathrm{KL}}\left[\pi^{(k)}(\cdot|s) \,\|\, \pi^{(k+1)}(\cdot|s)\right]\right],$$

$$\mathcal{L}_{\mathrm{V}}(\theta_{k+1}) = \mathbb{E}_{s\sim d_{\pi^{(k)}}}\left[\left(V^{(k)}(s) - V^{(k+1)}(s)\right)^2\right].$$

1. Use the base RL algorithm to train $\pi^{(k)}$ and $V^{(k)}$.
2. Initialise new *student* networks for $\pi^{(k+1)}$ and $V^{(k+1)}$. We refer to the current policy $\pi^{(k)}$ and value function $V^{(k)}$ as the *teacher*.
3. Distill the teacher into the student. This phase is discussed in more detail in section 4.1.
4. The student replaces the teacher: $\pi^{(k)}$ and $V^{(k)}$ can be discarded.
5. Increment k and return to step 1. Repeat as many times as needed.

ProcGen performance

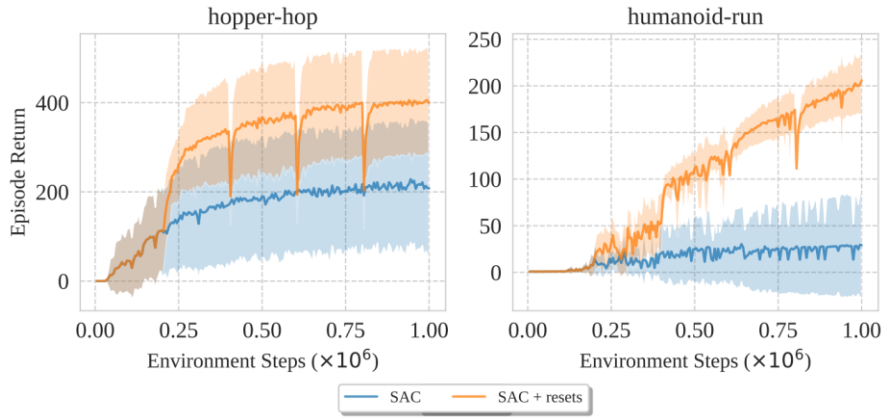


[1] Transient Non-Stationarity and Generalisation in Deep Reinforcement Learning., Igl et al., ICLR 2021.

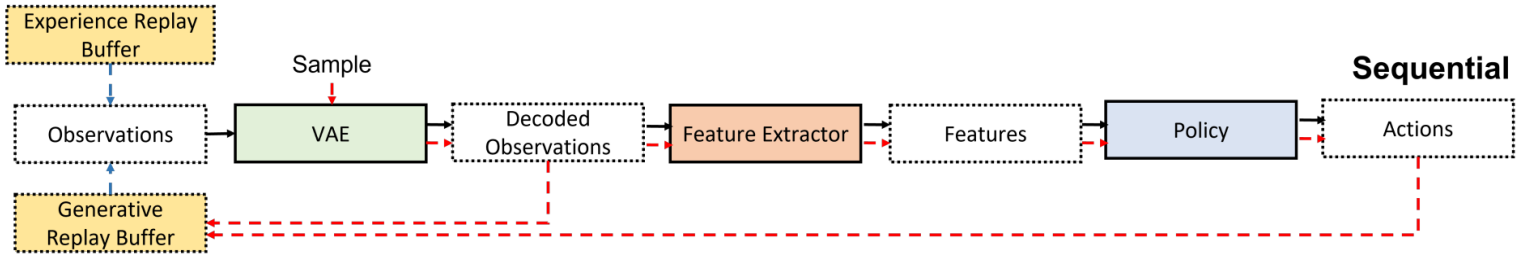
Rehearsal Based Method

Save previous tasks' data or generate your own data, and keep training on them.

e.g., Replay buffer + Reset (in some sense)



e.g., Model-Free Generative Replay (using VAE)



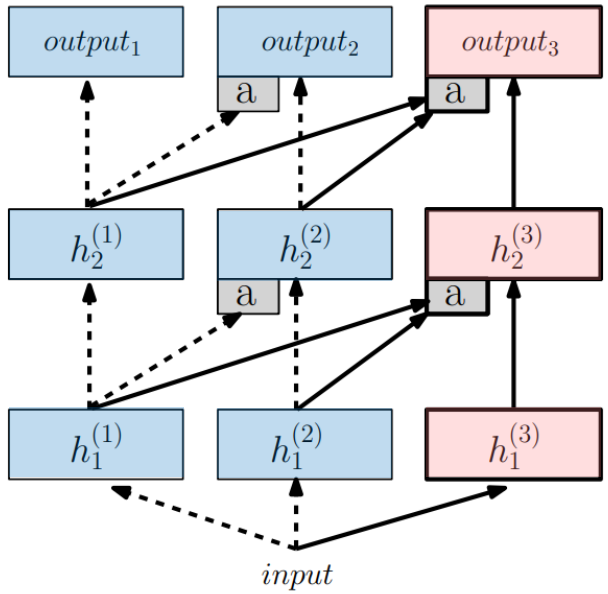
[1] The Primacy Bias in Deep Reinforcement Learning., Nikishin et al., ICML 2022.
[2] Model-Free Generative Replay for Lifelong Reinforcement Learning: Application to Starcraft-2., Daniels et al., CoLLAs 2022.

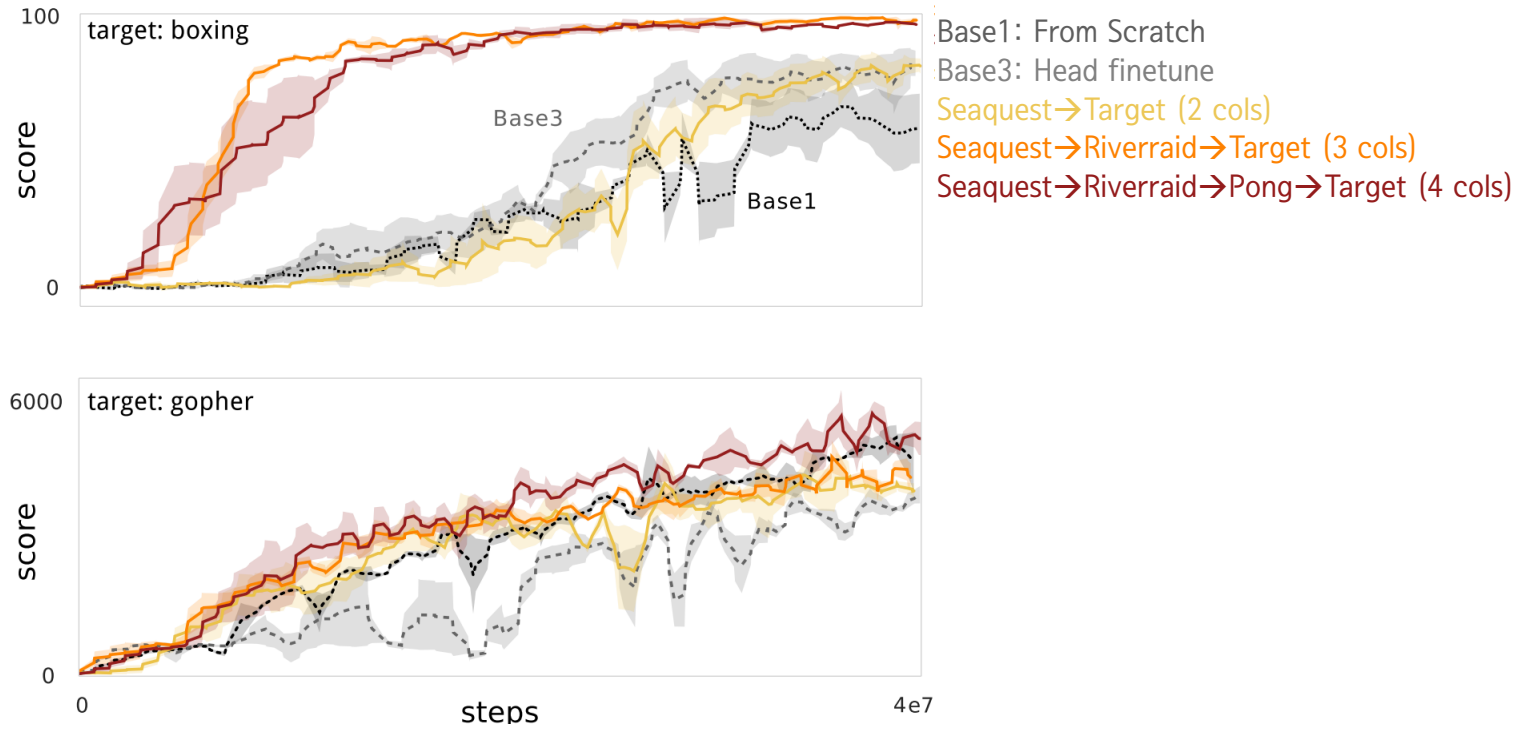
Architecture Based Method

Add a new set of parameters between each task

e.g., Memento

e.g., Progressive Neural Network → Allows forward transfer while preventing interference.



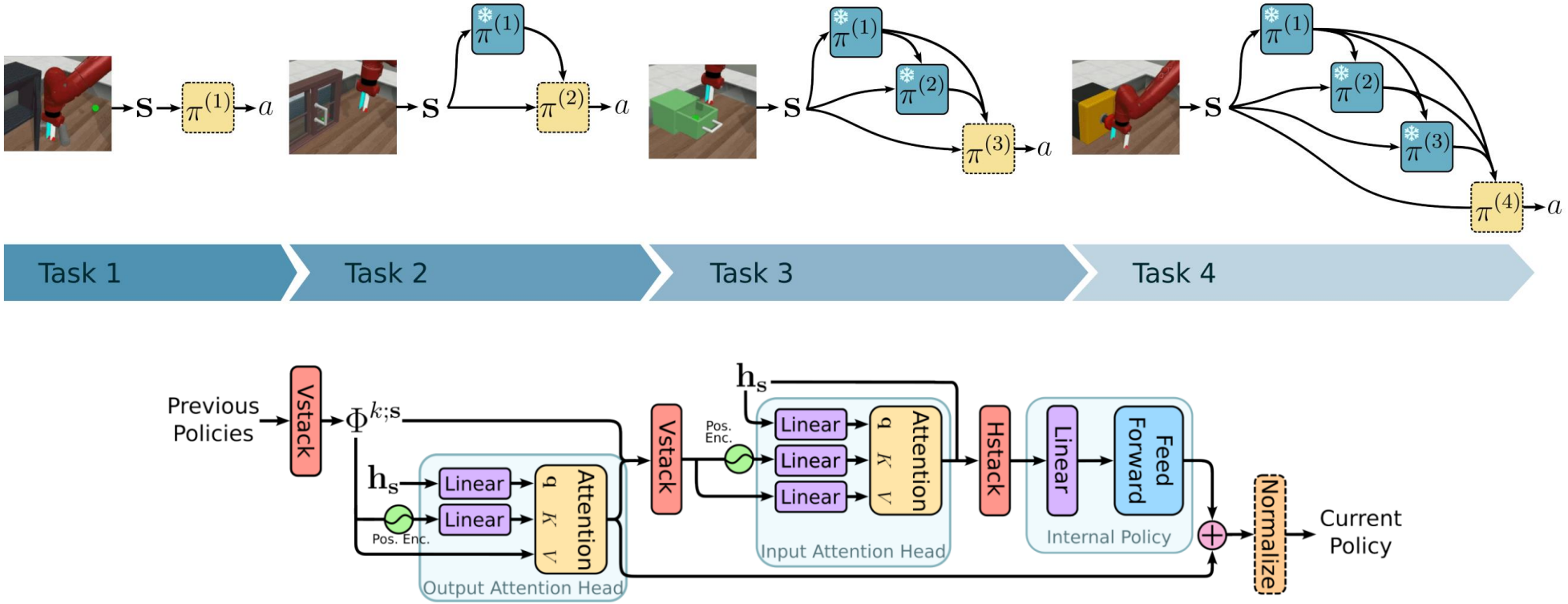


[1] Progressive Neural Networks., Rusu et al., arXiv 2016.

Architecture Based Method

Add a new set of parameters between each task

e.g., CompoNet (ICML 2024 oral)



[1] Self-Composing Policies for Scalable Continual Reinforcement Learning., Malagon et al., ICML 2024.

Architecture Based Method

Grow the subspace of parameter space.

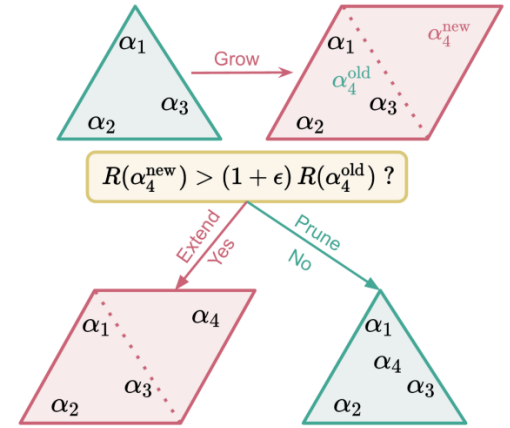
e.g., Continual Subspace of Policies (CSP)

Learn anchors in parameter space. Treat policy π as a linear combination of those anchors.

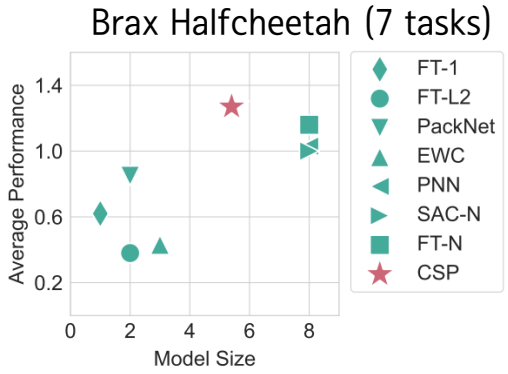
i.e., Learn anchors $\{\theta_1, \ldots \theta_k\}$, define a policy with $\pi(a|s, \sum \alpha_i \theta_i)$, where $\left|\left|\alpha\right|\right|_1 = 1$

This defines a simplex (triangle, tetrahedron, ...) where policy parameters can reside.

At each new task, decide whether to add another anchor (i.e., whether to expand the policy space).



(a) Continual Subspace of Policies (CSP)



(b) Performance-Size Trade-Off

[1] Building a subspace of policies for scalable continual learning., Gaya et al., ICLR 2023.

Meta-Learning Based Method

Meta-Learning algorithm is literally born to solve multi-task problems.

Interestingly, there are attempts to mitigate single-env interference with meta-learning.

e.g., Online-Aware Meta Learning

Measure interference as the degradation of value prediction accuracy.

Accuracy Change. At the most fine-grained, we can ask if an update, going from $\boldsymbol{\theta}_t$ to $\boldsymbol{\theta}_{t+1}$, resulted in interference for a specific point (s, a). The change in accuracy at s, a after an update is

$$\text{Accuracy Change}((s,a), \boldsymbol{\theta}_t, \boldsymbol{\theta}_{t+1}) := \mathbb{E}[\delta(\boldsymbol{\theta}_{t+1})|S=s, A=a]^2 - \mathbb{E}[\delta(\boldsymbol{\theta}_t)|S=s, A=a]^2$$

We want to minimize interference with current param θ_t and the parameter after n-updates θ_{t+n}

This is achieved simply by minimizing TD loss of θ_{t+n}, or $\delta(\theta_{t+n})$.

In other words, find θ_t such that after n-updates, the resulting θ_{t+n} has the smallest TD loss.

$\boldsymbol{\theta}_{t,0} \leftarrow \boldsymbol{\theta}_t$
for $i \leftarrow 1, 2, ...n$ **do**
  Sample a set of transitions B_{i-1} from the buffer
  $\boldsymbol{\theta}_{t,i} \leftarrow U_{B_{i-1}}(\boldsymbol{\theta}_{t,i-1})$ *# Inner update*
end for
Meta update by second-order MAML method:
  Sample a set of transitions B from the buffer
  $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \frac{\alpha}{|B|} \sum_j \delta_j(\boldsymbol{\theta}_{t,n}) \nabla_{\boldsymbol{\theta}_t} Q_{\boldsymbol{\theta}_{t,n}}(s_j, a_j)$

Use MAML!

while not done **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  for all $\mathcal{T}_i$ **do**
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
    Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
  end for
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
end while

[1] Measuring and Mitigating Interference in Reinforcement Learning., Liu et al., CoLLAs 2023.

Reference

The Primacy Bias in Deep Reinforcement Learning., Nikishin et al., ICML 2022. [MILA]
Deep Reinforcement Learning with Plasticity Injection., Nikishin et al., NeurIPS 2023. [Deepmind]
A Comprehensive Survey of Forgetting in Deep Learning Beyond Continual Learning., Wang et al., CVPR 2023. [Maryland]
Transient Non-Stationarity and Generalisation in Deep Reinforcement Learning., Igl et al., ICLR 2021. [Oxford, Deepmind]
Progressive Neural Networks., Rusu et al., arXiv 2016. [Deepmind]
Overcoming catastrophic forgetting in neural networks., Kirkpatrick et al., PNAS 2017 [Deepmind]
Continual Unsupervised Representation Learning., Rao et al., NeurIPS 2019. [Deepmind]
Measuring and Mitigating Interference in Reinforcement Learning., Liu et al., CoLLAs 2023. [Alberta]
Self-Composing Policies for Scalable Continual Reinforcement Learning., Malagon et al., ICML 2024. [Basque]
Model-Free Generative Replay for Lifelong Reinforcement Learning: Application to Starcraft-2., Daniels et al., CoLLAs 2022. [SRI International]
Neurogenesis Deep Learning., Draelos et al., IJCNN 2017 [Sandia National Laboratories]
- For autoencoders. Adding neurons + self-replay stabilization.
UNCLEAR: A Straightforward Method for Continual Reinforcement Learning., Kessler et al., ICML 2024 Workshop [Oxford]
- EWC on all parameters from each preceding tasks. Interestingly, their setting gives task index only at training time.
Building a subspace of policies for scalable continual learning., Gaya et al., ICLR 2023. [Meta, Mcgill]
- Learn anchors in the parameter space. Treat each task's policy as a linear combination of those anchors.
- New anchor = Expansion of the subspace where policies can reside.
Lifelong Learning with Dynamically Expandable Networks., Yoon et al., ICLR 2018. [KAIST]
- A bunch of something. enforce sparsity + selective retrain + selective expansion + duplicating neurons
The Forget-me-not Process., Milan et al., NeurIPS 2016. [Deepmind, Alberta]
- Bayesian something something. Actually couldn't understand :/