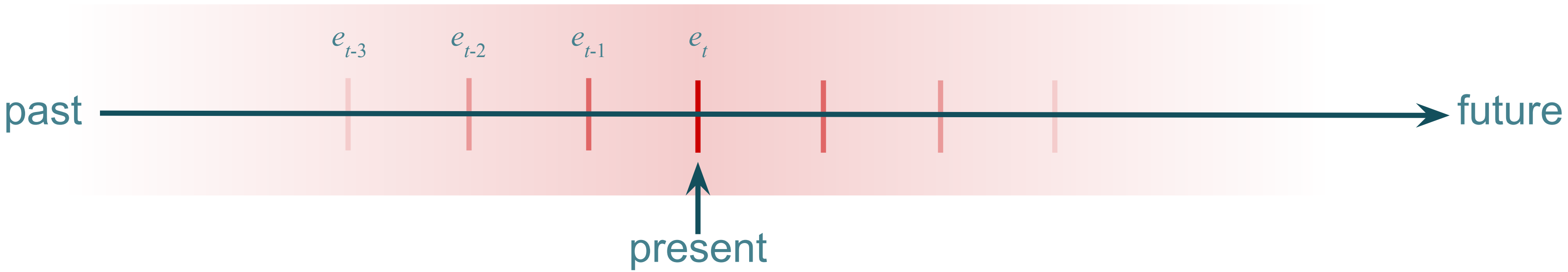
Streaming Deep Reinforcement Learning

March 3, 2025

Rupam Mahmood

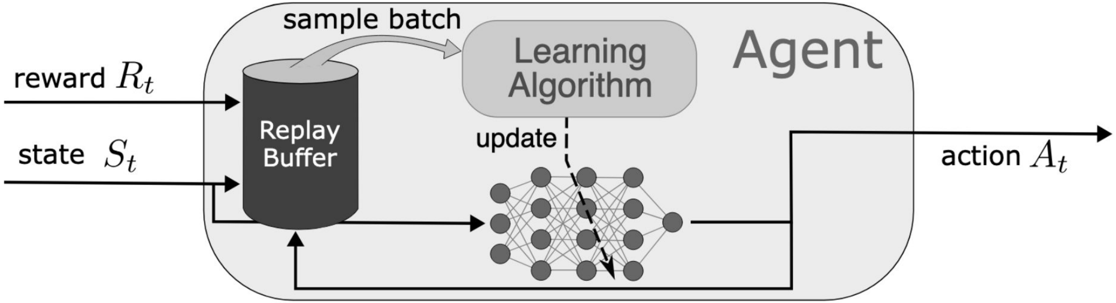
What is streaming learning?

- Learning from a stream of experience
  - as soon as they arrive
  - without storing experience in raw form



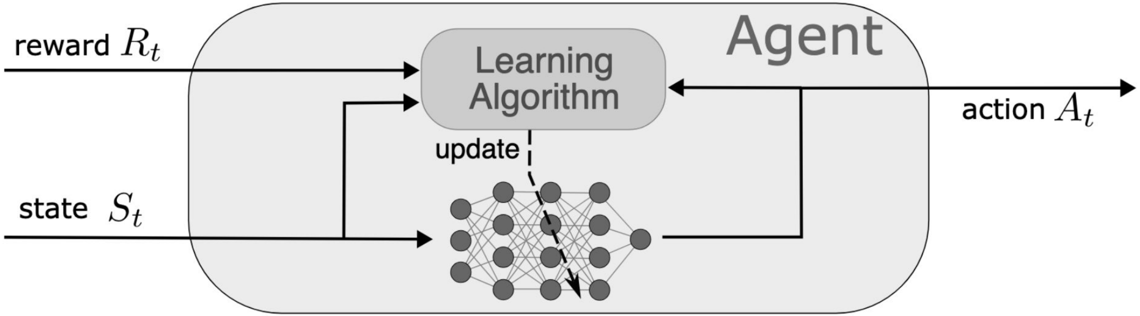
Deep RL has been batch learning instead of streaming

- Uses and learns a nonlinear function approximation
- Stores past experience in raw form
- Makes mini-batch updates



Classic RL has been streaming

- TD(λ), Q(λ), SARSA(λ), AC(λ)
- Uses linear function approximation
- Or learns last linear layer on top of fixed representation



Pseudocode of a typical Deep RL algorithm

Initialize an empty replay buffer: $\mathcal{D} \leftarrow \emptyset$

Initialize state S_0

For time $t : 0$ to T

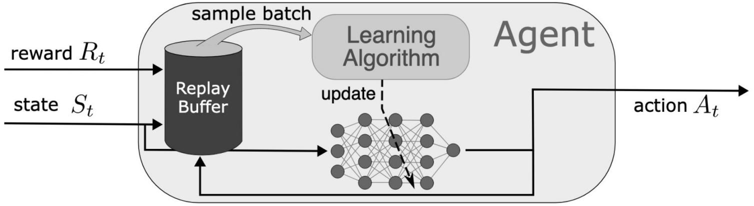
Policy inference: $A_t \sim \pi_w(\cdot|S_t)$

Observe reward and next state: R_{t+1}, S_{t+1}

Store sample: $\mathcal{D} \leftarrow \mathcal{D} \cup \{S_t, A_t, R_{t+1}, S_{t+1}\}$

Sample a mini-batch: $\mathcal{B} \sim \mathcal{D}$

Make batch update: $w \leftarrow w - \alpha \dfrac{\partial \mathcal{L}(w, \mathcal{B})}{\partial w}$



Why do we need streaming learning?

- Discussion

Why do we need streaming learning? (cont'd)

- To understand natural learning
- To adapt fast
- To learn continually in real time under resource constraints
    - In real-world deployed systems, on-demand learning compute is always scarce

Why do we need streaming learning? (cont'd)

Why hasn't there been streaming deep RL?

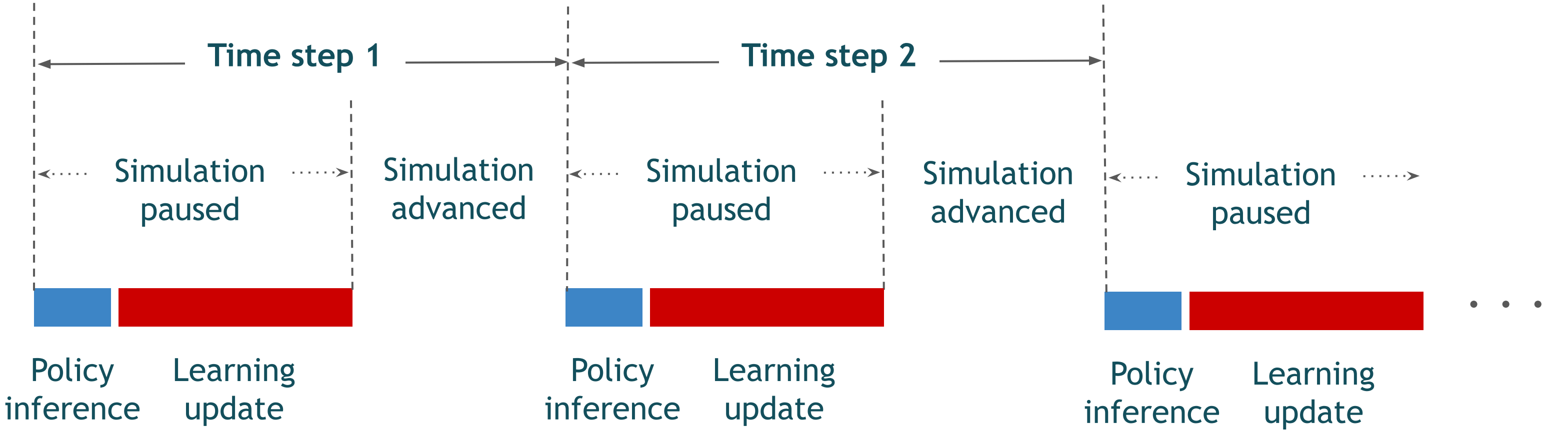
- Discussion

Why hasn't there been streaming deep RL? (cont'd)

This was done using deep RL but offline in simulations

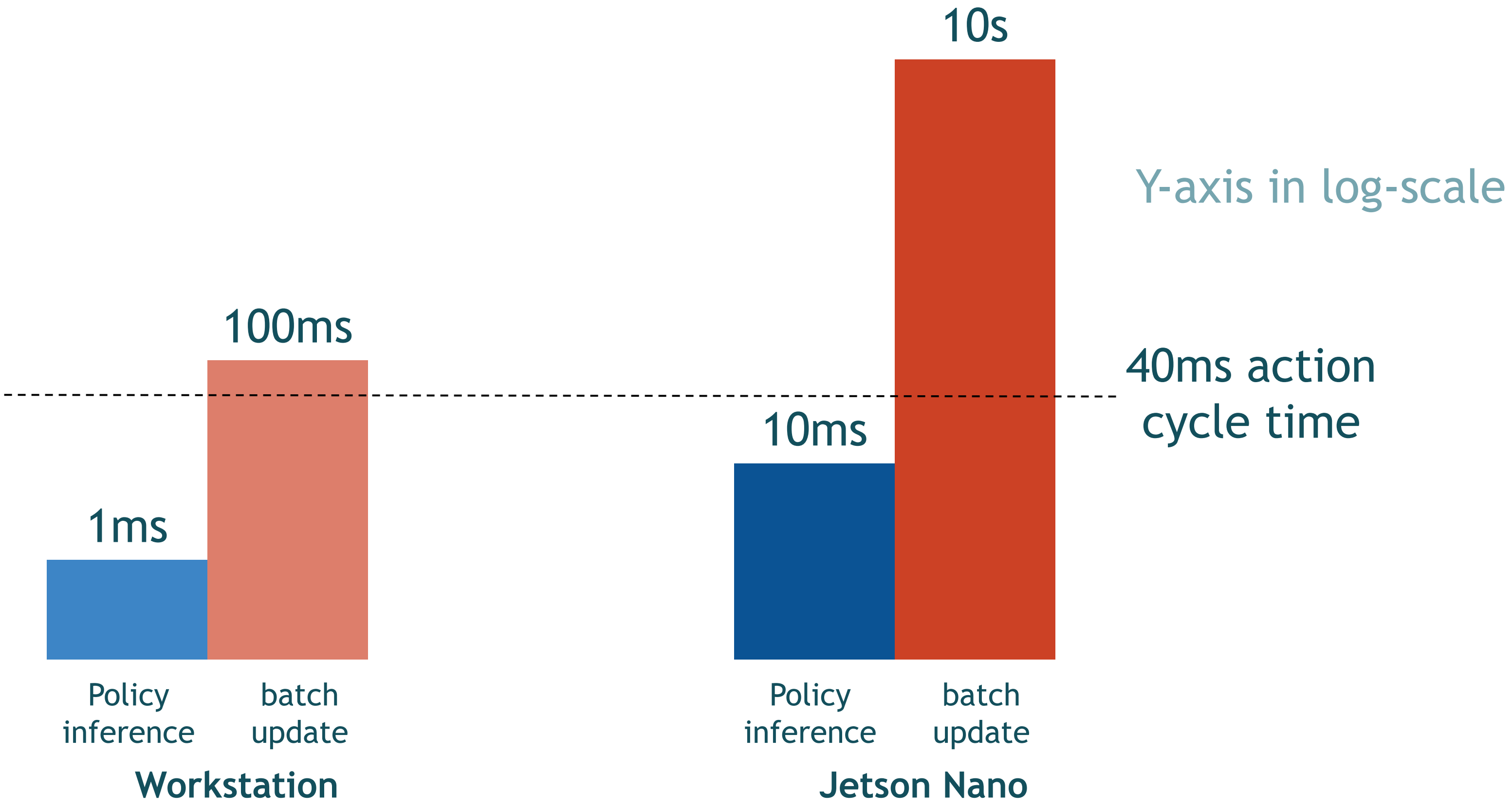
Kaufmann et al. (Nature 2023)

Simulations assume abundant computational power

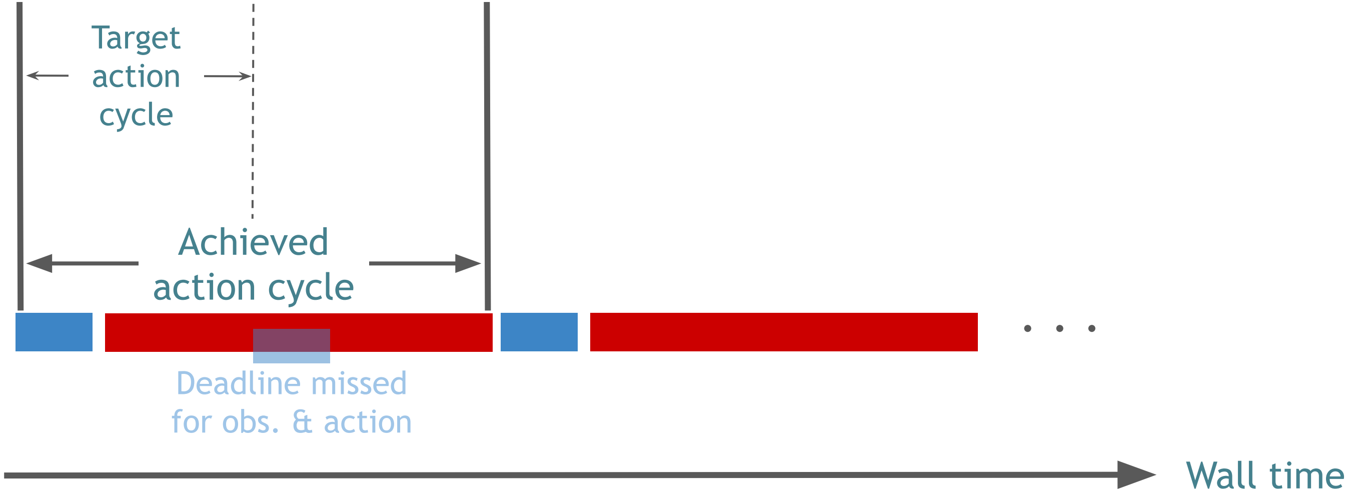


In simulations, we can always make one update per time step

Mini-batch updates are computation-wise expensive

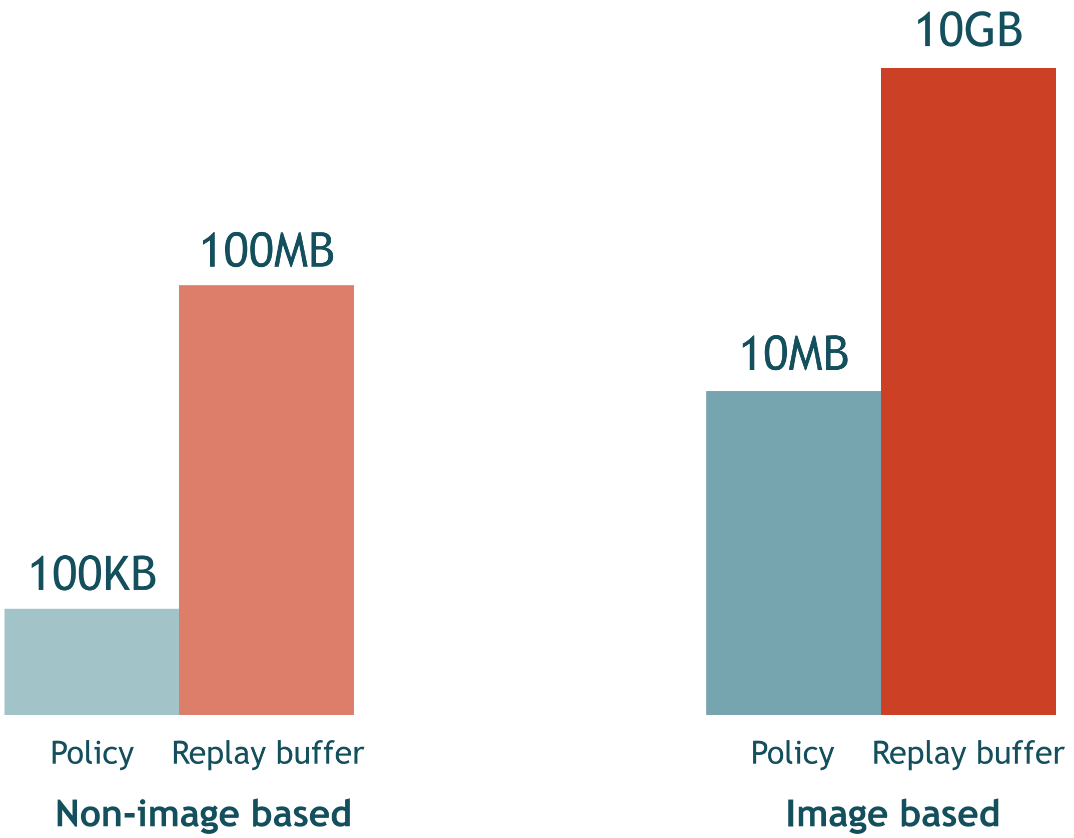


Mini-batch updates are too expensive for real-time learning



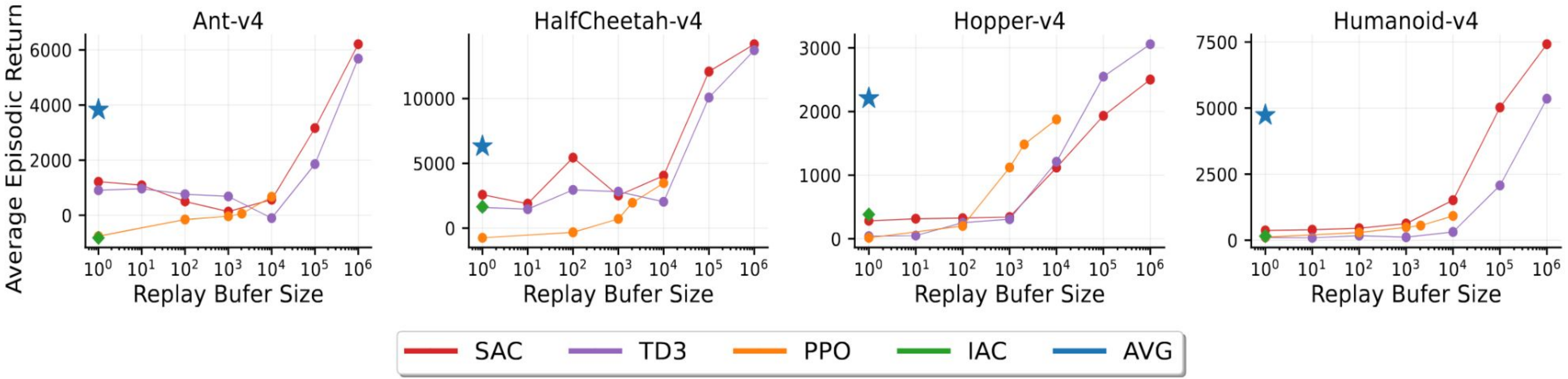
Yuan, Y., & Mahmood, A. R. (ICRA 2022)

Replay buffers are memory-wise expensive



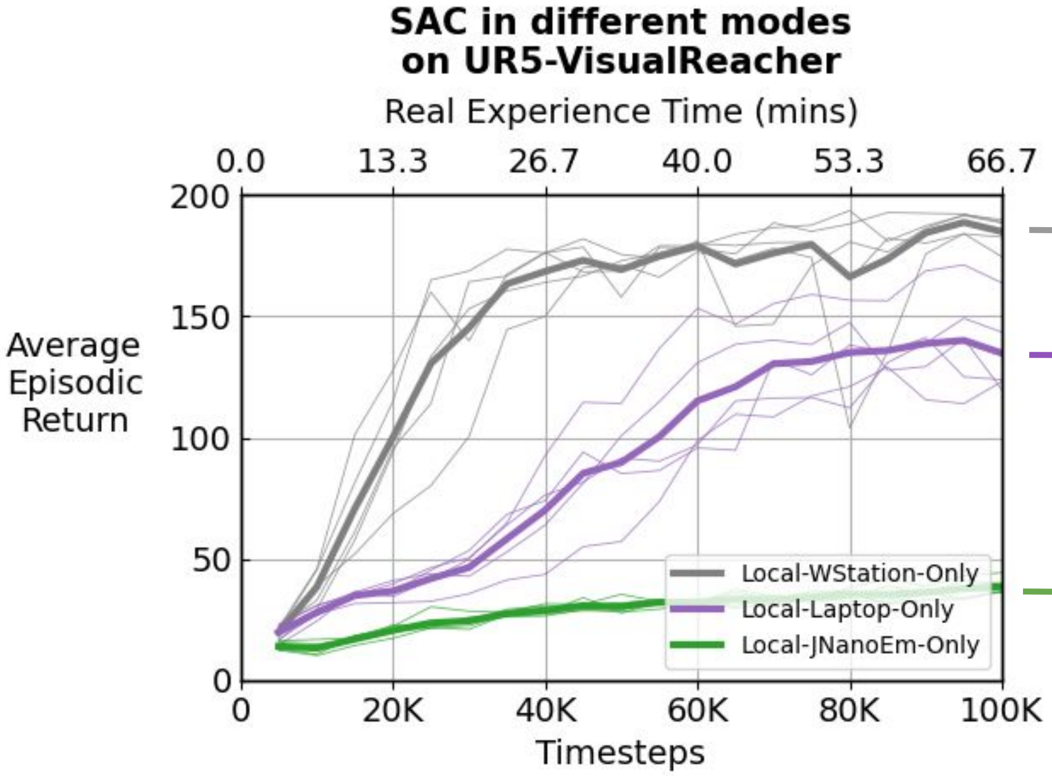
Y-axis in log-scale

Reducing the buffer size has a catastrophic effect



Vasan et al. (NeurIPS 2024)

Deep RL under resource constraint didn't work so far



Tethered to a workstation

Tethered to a laptop

Tethered to a Jetson Nano (emulated)

Wang*, Vasan*, & Mahmood (ICRA 2023)

Deep RL under resource constraint hasn't work so far

Continual Learning on Real Robots - Rupam Mahmood

Why has deep RL under resource constraint not work so far?

- Discussion

Why has deep RL under resource constraint not work so far? (cont'd)

- Learning without replay buffer causes failure

Why does learning without replay buffer cause failure?

- For that we can go to the source of replay/batch/non-sequential learning
- Experience replay in RL was first introduced by Lin (1992)
  - *"experience replay was quite effective in speeding up the credit assignment process"*
  - But a similar idea even earlier as *relaxation planning* in tabular Dyna-Q (Sutton 1990)
- But the idea of revisiting past samples was used earlier in NN
  - To address catastrophic forgetting in the original paper by McCloskey and Cohen (1989)
  - But even earlier by Hinto and Plaut (1987) to "deblur old memories" by "rehearsing"

What is catastrophic forgetting?

- *Catastrophic forgetting* (CF) is the phenomenon of the learner performing poorly on past tasks upon learning new tasks, i.e., in continual learning

- This problem was first observed in sequential learning
  - As can be seen, 80s were full of streaming learning and adjacent ideas

- Replaying/rehearsing/revisiting past samples was introduced as a remedy

Why is a pathology of CL relevant to single-task deep RL?

- Generally, learning dynamics face pathologies when learning …
  - under nonstationarity
  - using backpropagation/gradient descent
- Deep RL has both of these components even when learning a single task
- We contend that learning dynamics in deep RL face similar pathologies

What are some of the pathologies?

- Catastrophic forgetting caused by interfering signals from nonstationary data
    - Causes instability
- Instability can also be caused by exploding gradient
- There are also pathologies causing loss of plasticity/trainability
    - Feature dormancy/saturation, large weight, vanishing gradient, representation collapse, ill-conditioning, loss of curvature

A major remedy: Search

- Augment or go beyond backprop / gradient-based learning
- Search methods are known to avoid catastrophic forgetting
  - Like evolutionary methods, but they are expensive
- Representation can be searched in a streaming manner as well
  - Mahmood and Sutton (2013) introduced streaming search with feature re-initialization
  - It was combined with backprop for deep network by Dohare et al. (2024)

What are some of the pathologies and their remedies?

- The pathologies are often together referred to as stability-plasticity dilemma
- Many classic and modern ML/DL techniques provide recourse
  - Special weight initialization and re-initialization
  - Sparse representation and norm regularization
  - Normalization techniques
  - Skip connections, gradient clipping, and modern activations
  - Adaptive optimizers

In a recent work, we looked at backprop more closely

Forward pass: $a_{l+1,i} = \sum_{j=1}^{|\boldsymbol{h}_l|} W_{l+1,i,j} h_{l,j}$ $\quad \boldsymbol{h}_l = \boldsymbol{\sigma}(\boldsymbol{a}_l)$

Pre-activation | Weight | Input | Activation output | Activation function

Backpropagation chain rule:

$$\frac{\partial \mathcal{L}}{\partial W_{l,i,j}} = \frac{\partial \mathcal{L}}{\partial a_{l,i}} \frac{\partial a_{l,i}}{\partial W_{l,i,j}} = \frac{\partial \mathcal{L}}{\partial a_{l,i}} h_{l-1,j}$$

$$\frac{\partial \mathcal{L}}{\partial a_{l,i}} = \sigma'(a_{l,i}) \sum_{k=1}^{|\boldsymbol{a}_{l+1}|} \frac{\partial \mathcal{L}}{\partial a_{l+1,k}} W_{l+1,k,i}$$

"Steepest descent procedures preferentially change existing, already-useful features rather than make new ones from unused units." (Sutton 1986)

Elsayed, M., & Mahmood, A. R. (ICLR 2024)

We introduce Utility-based Perturbed Gradient Descent (UPGD)

- More useful weights have higher protection from change
- Less useful weights are repurposed for plasticity through perturbation

$$w_i \leftarrow w_i - \alpha \left(\underbrace{\frac{\partial \mathcal{L}}{\partial w_i}}_{\text{Gradient}} + \underbrace{\xi_i}_{\text{Perturbation}} \right) (1 - \underbrace{U_i}_{\text{Utility}})$$

[0, 1]

$$U_i = \mathcal{L}(\mathcal{W}_{\neg[i]}, Z) - \mathcal{L}(\mathcal{W}, Z)$$

Utility of a weight is defined as
the excess loss for not having the connection

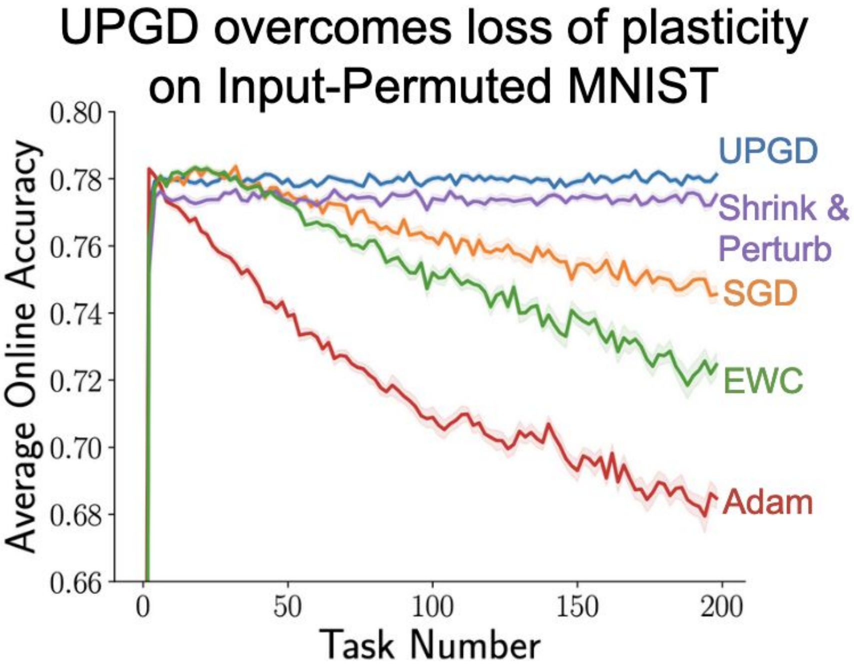
$$\approx -\frac{\partial \mathcal{L}}{\partial w_i} w_i + \frac{1}{2} \frac{\partial^2 \mathcal{L}}{\partial w_i^2} w_i^2$$

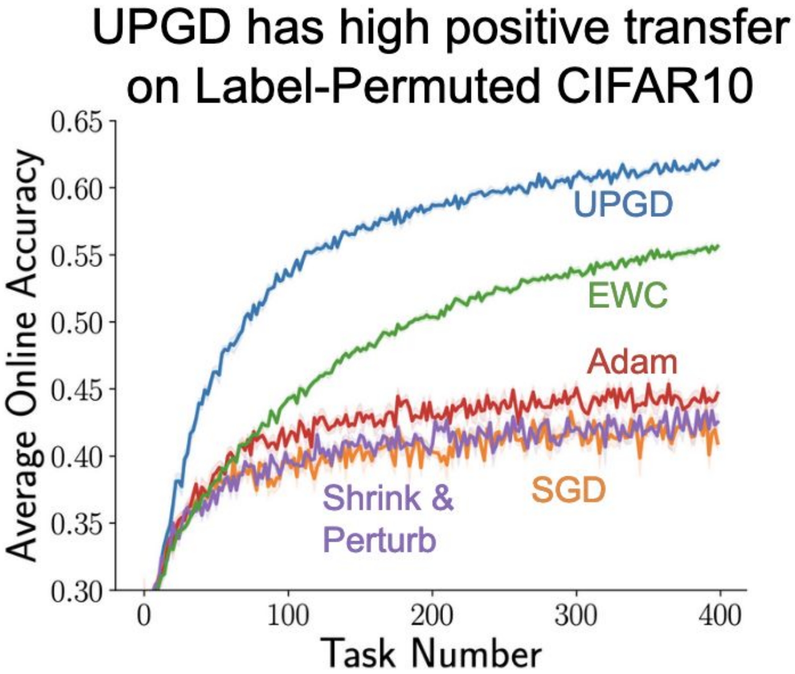
The diagonal of Hessian can be approximated well in $O(n)$ (ICML-2024)

Utility is estimated using Taylor approximation and scaled between [0, 1]

Elsayed, M., & Mahmood, A. R. (ICLR 2024)

UPGD addresses continual learning issues in the streaming learning setting fully incrementally



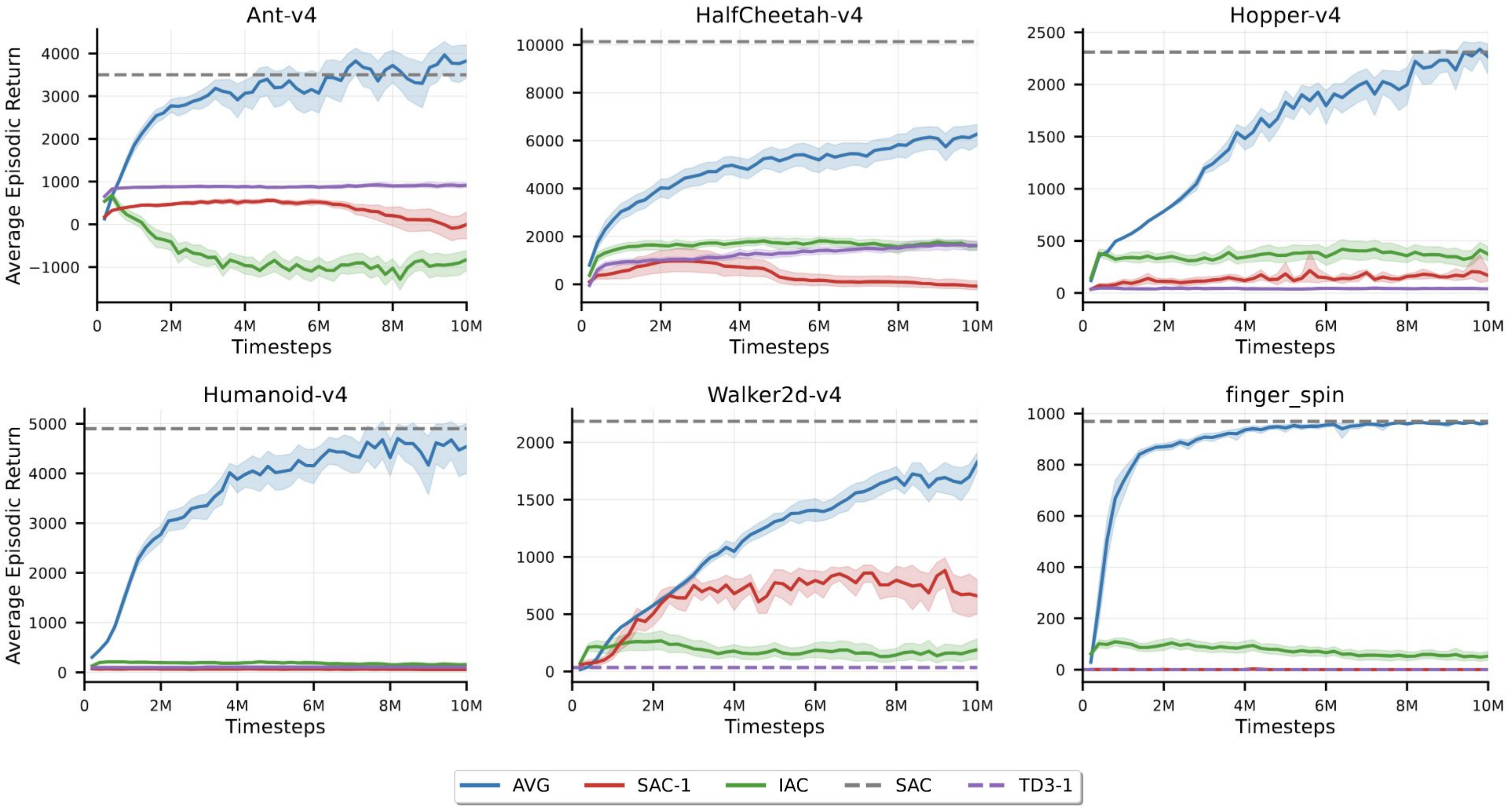


Elsayed, M., & Mahmood, A. R. (ICLR 2024)

Deep streaming learning algorithm: AVG

- We introduced the action-value gradient method that makes policy gradient updates by taking the gradient of the action-values wrt the (cont's) action
    - Think of SAC without replay buffers, target networks, and mini-batch updates
- We added the following techniques
    - Observation and TD error normalization
    - Penultimate Normalization

Deep streaming learning algorithm: AVG (cont'd)



Vasan et al. (NeurIPS 2024)

Class of deep streaming learning algorithms: Stream-X

- We introduced a number of techniques on top of classic RL algorithms with eligibility traces like Q(λ), SARSA(λ), AC(λ)
  - Sparse initialization
  - Layer normalization
  - Observation and reward normalization
  - Bounding step-size based on update size

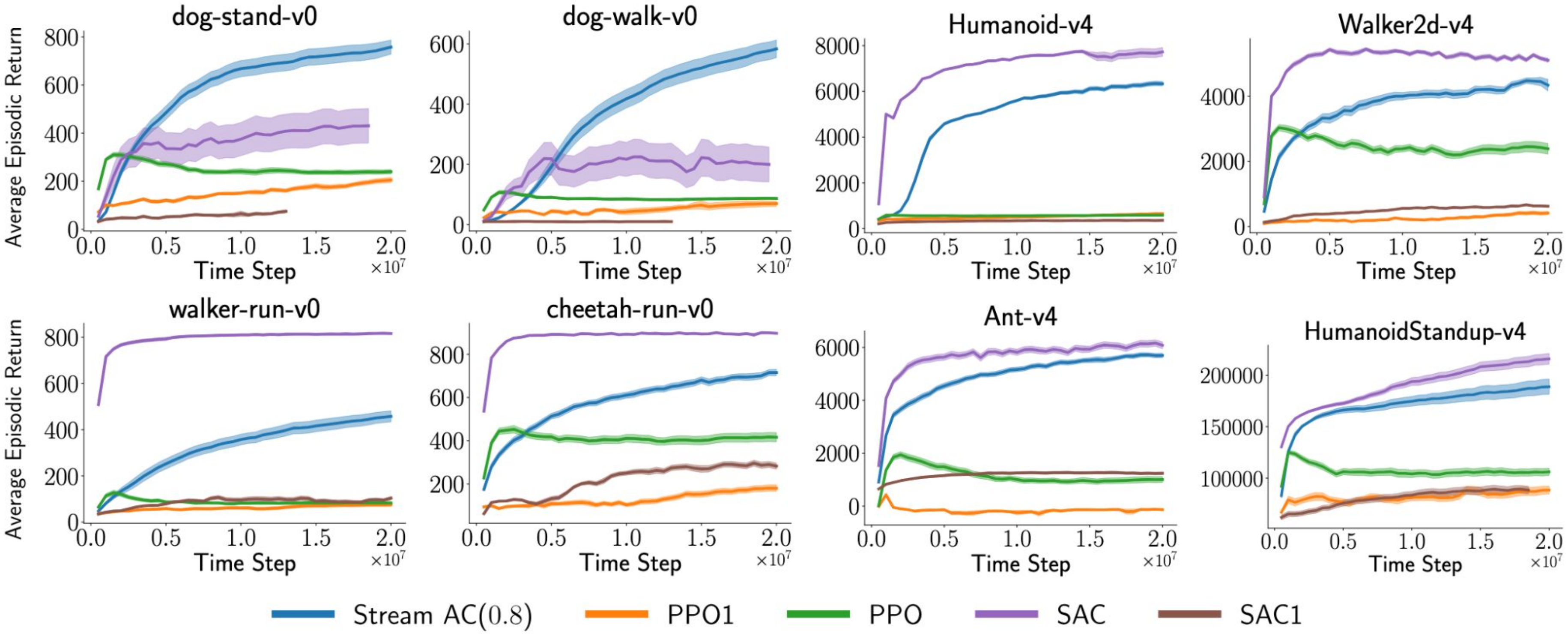
Quantifying update size

- One notion of a "too large" update is overshooting in the loss landscape
  - Used in batch learning or in optimization where the true loss function is known
- For single sample updates, overshooting can be quantified as
  - Opposing error sign: $\delta(w)\delta(w') < 0$
  - Negative inner product between gradients: $\nabla\delta^2(w)^\top\nabla\delta^2(w') < 0$
  - Effective step size being larger than 1: $(\ \delta(w) - \delta(w')\ )\ /\ \delta(w) > 1$
  - Introduced in Mahmood (2010) and Mahmood et al. (2012)

Bounding step size when the update too large

- When an update is deemed too large
  - Then bound the step size by a value to make the update small enough
  - Use the original step size, otherwise
- It mitigates having opposing error signs / gradient / interference
  - Potentially providing stability

Performance of Stream-AC

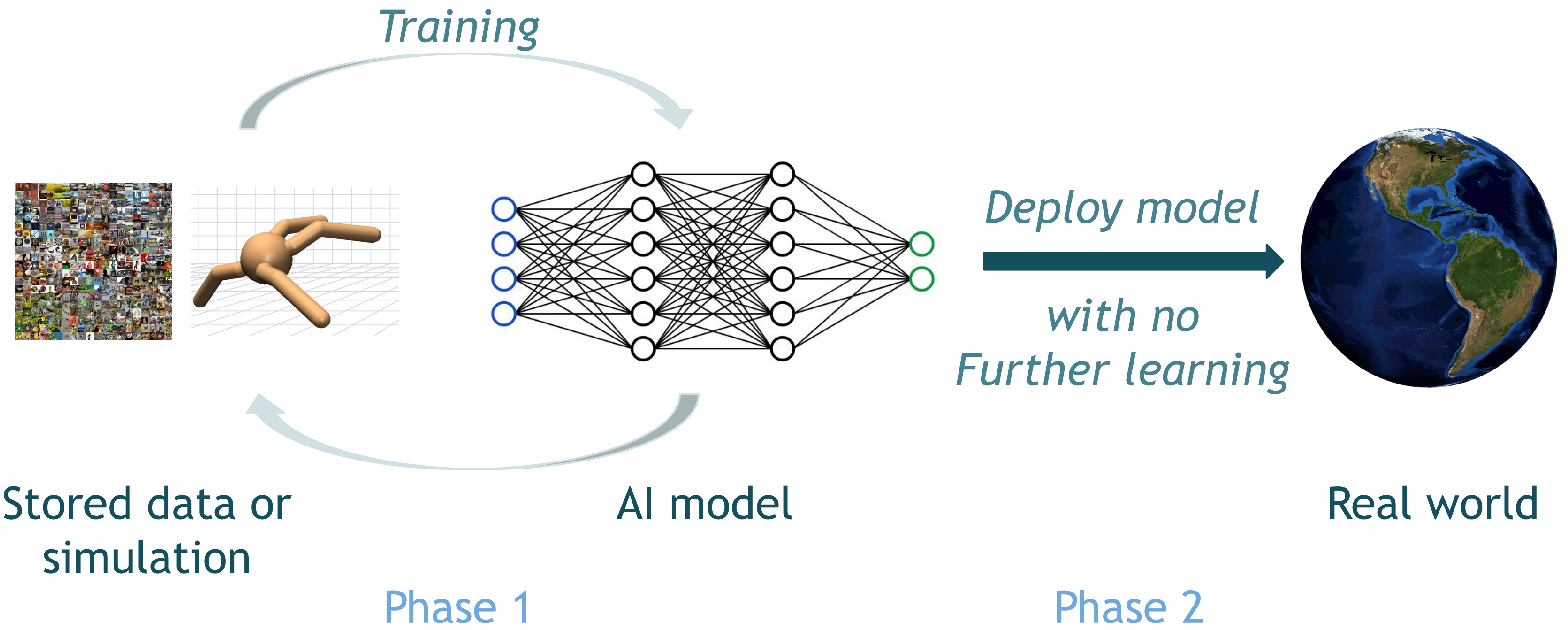


Elsayed, Vasan & Mahmood (arXiv 2024)

Conclusion

- Streaming deep RL is important for continual learning in deployment
- It shares the same issues continual learning has
- Overcoming some of the continual learning issues allowed us having successful streaming deep RL for the first time

Offline learning ends before deployment



Almost all AI we currently know are offline learned / train-once AI

Real-time learning interacts and learns while deployed