Deep Policy Gradient Methods Without Batch Updates, Target Networks, or Replay Buffers

Gautham Vasan, Mohamed Elsayed, Alireza Azimi, Jiamin He, Fahim Shahriar,
Colin Bellinger, Martha White & A. Rupam Mahmood

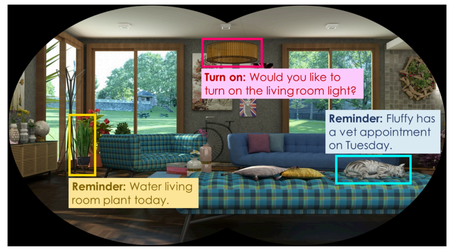
Deep Learning: Classics and Trends, ML Collective
Feb 7th 2025

I’m interested in agents that can learn on the fly,
in real-time, by interacting with their environment

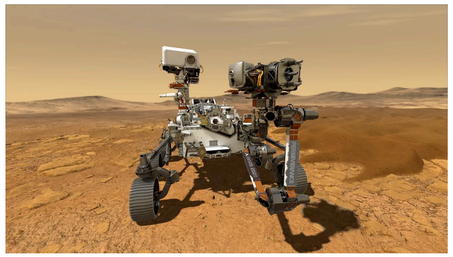
Fun with Squirrels 🐿️

- Real-time learning
- Learn through trial-and-error interactions with the environment
- React in milliseconds to achieve their goal

Real-Time Adaptation



Application of an AR Headset



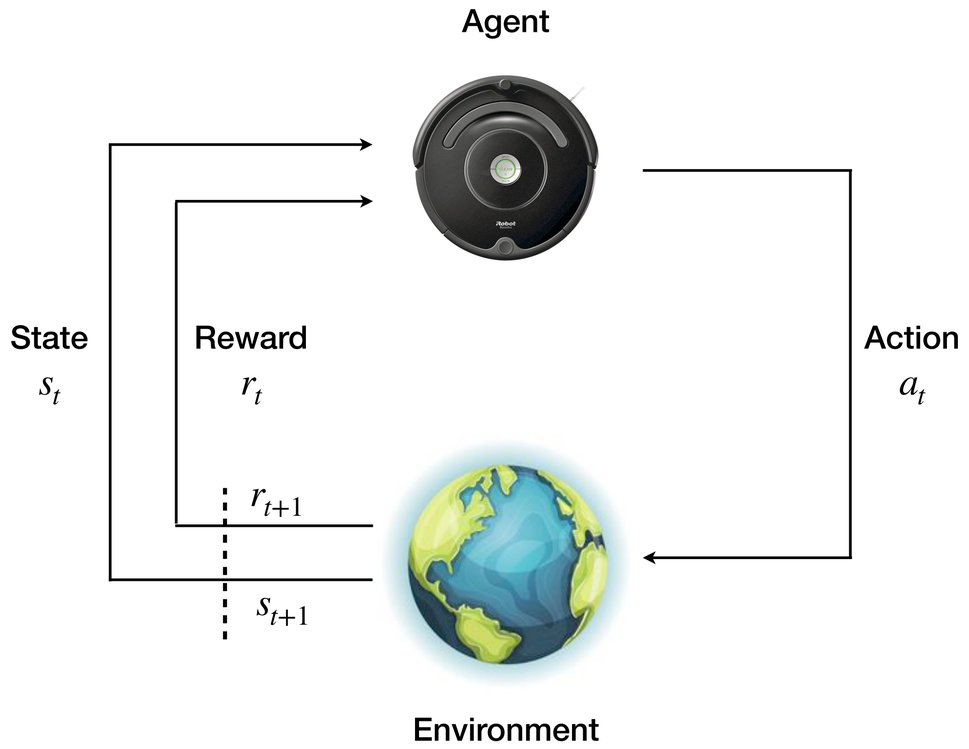
Mars Perseverance Rover

Apple Watch

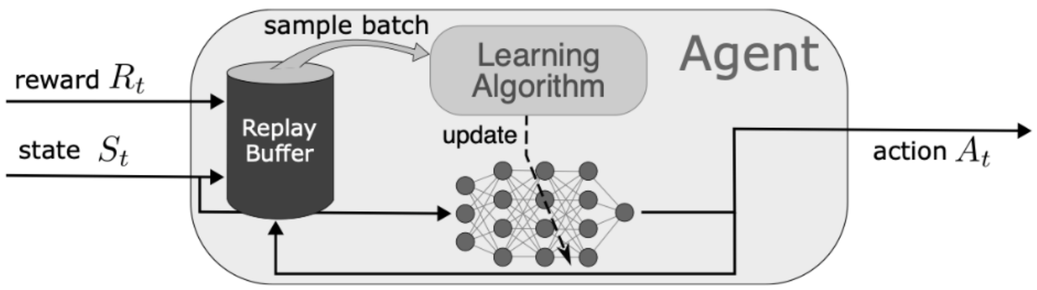
iRobot Roomba

- In current AI systems, training and deployment are distinct phases, with the system frozen during deployment
- Learning online, in real-time, can enable such systems to adapt to unforeseen changes in dynamic environments
    - Learning must occur on-device!

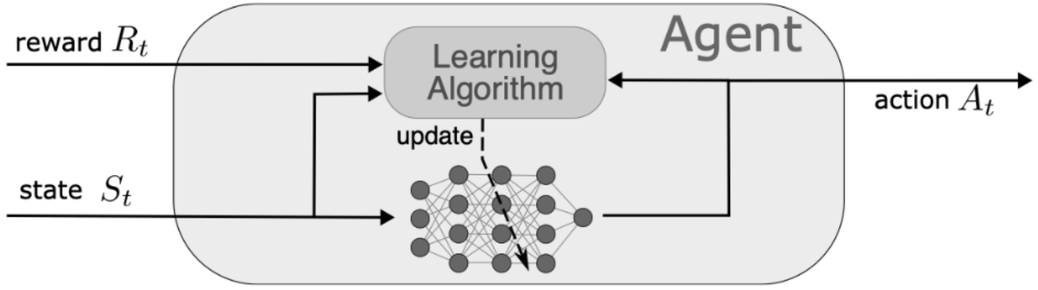
Agent-Environment Interaction in Reinforcement Learning (RL)



Batch RL vs Streaming RL



(a) Batch RL



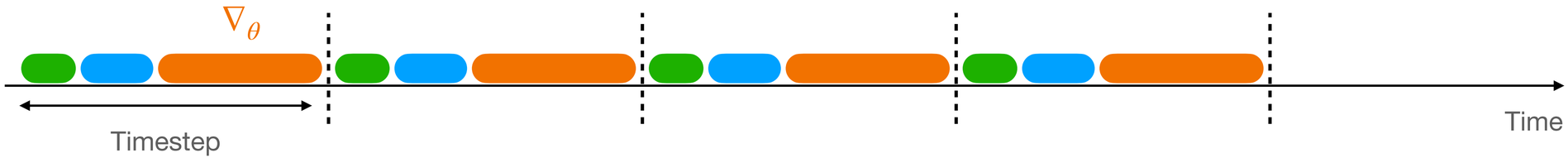
(b) Streaming RL

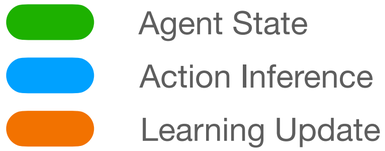
- Batch RL methods sample a batch of data from the replay buffer and use it to make a learning update
    - For example, Soft Actor Critic (SAC), Proximal Policy Optimization (PPO) and Twin Delayed DDPG (TD3)
- In contrast, streaming RL methods learn incrementally by processing one sample at a time, without storing any samples
    - For example, Actor Critic, Q(λ) and Sarsa(λ)
- Streaming RL methods
    - are computationally cheap and amenable to real-time updates
    - support on-device learning, which is critical to preserving the privacy and security of the user

Note:
I'm using incremental and streaming interchangeably in this talk

Elsayed, M., Vasan, G., & Mahmood, A. R. (2024). Streaming Deep Reinforcement Learning Finally Works. *arXiv preprint arXiv:2410.14606*.

Learning in Simulation

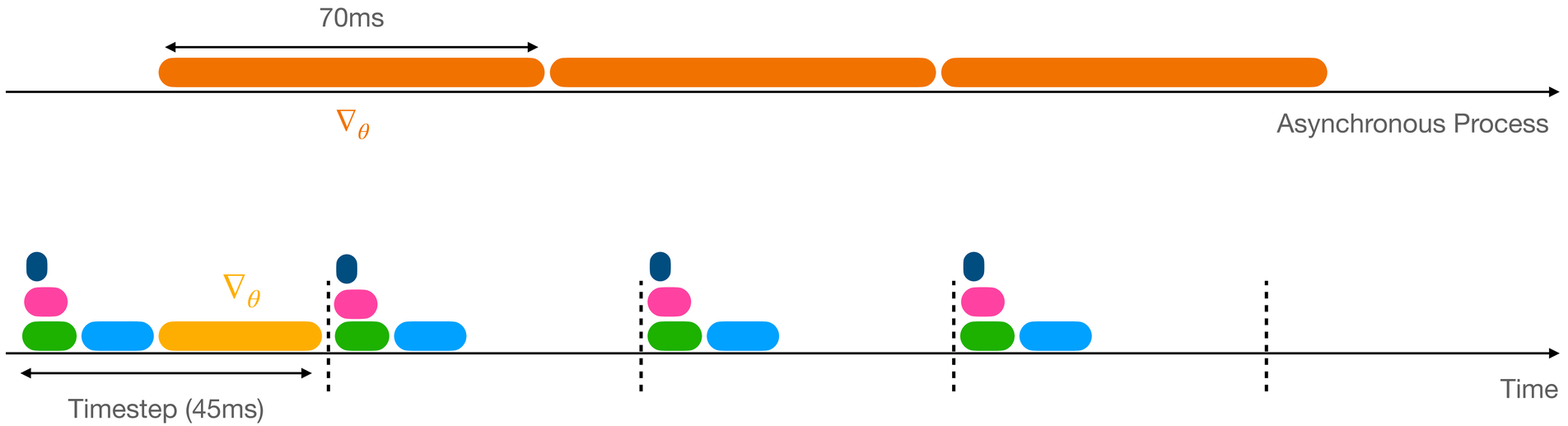




“The real world does not *pause* while the agent computes actions or makes learning updates”

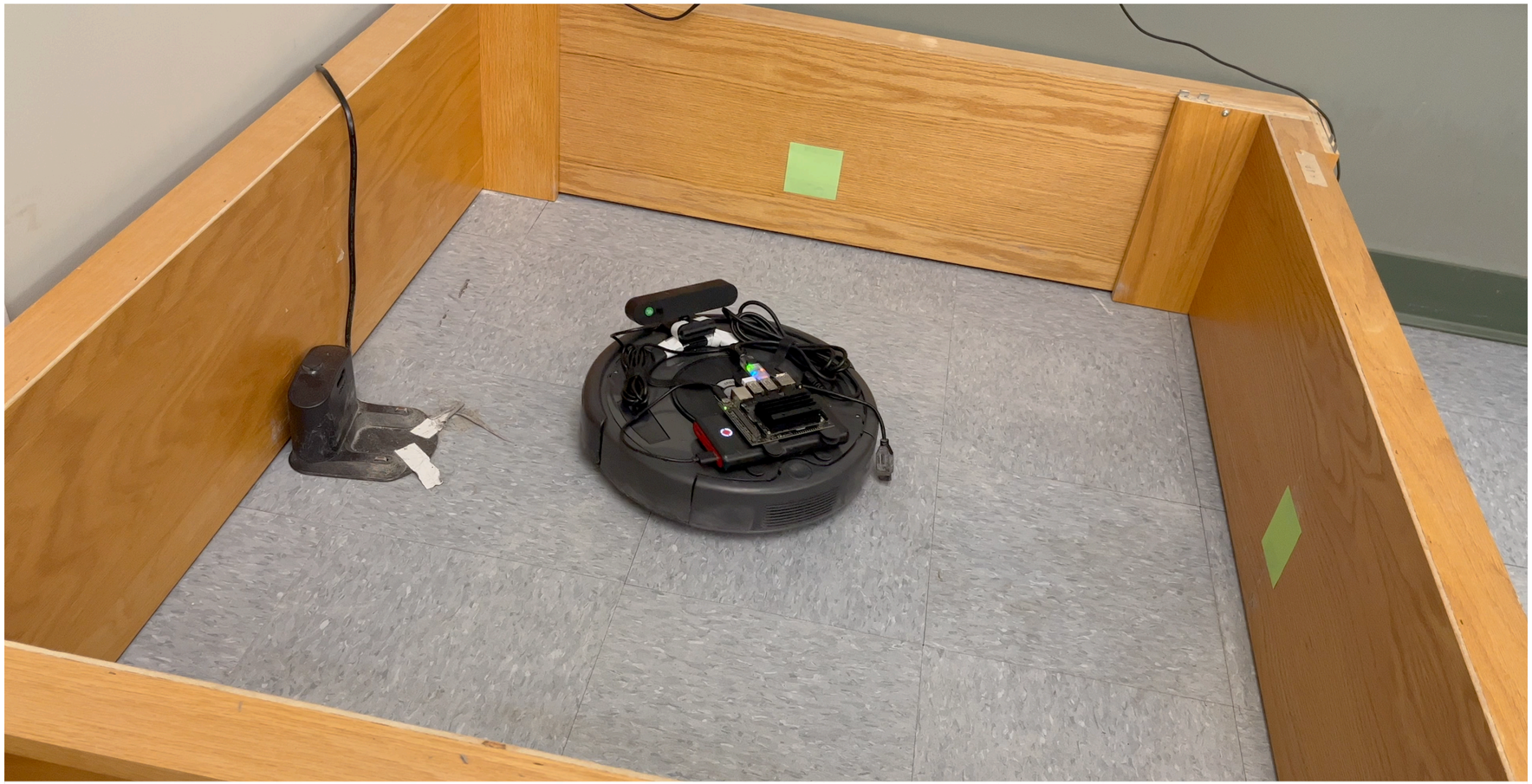
Wang, Y., Vasan, G., & Mahmood, A. R. (2023, May). Real-time reinforcement learning for vision-based robotics utilizing local and remote computers. In *2023 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 9435-9441). IEEE.

Real-Time Learning



Example: Vision-based Real-Time RL

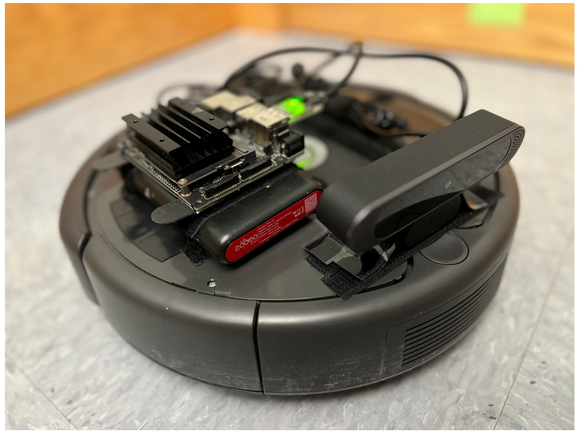
Create-Reacher



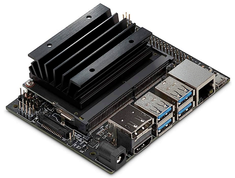
Random Policy (1x)

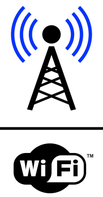
Wang, Y., Vasan, G., & Mahmood, A. R. (2023, May). Real-time reinforcement learning for vision-based robotics utilizing local and remote computers. In *2023 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 9435-9441). IEEE.

Remote-Local Distributed Learning



Jetson Nano 4GB





Workstation
(128GB RAM,
12GB Nvidia 3090 GPU)

- Local device
- Policy inference
- Store only Actor π

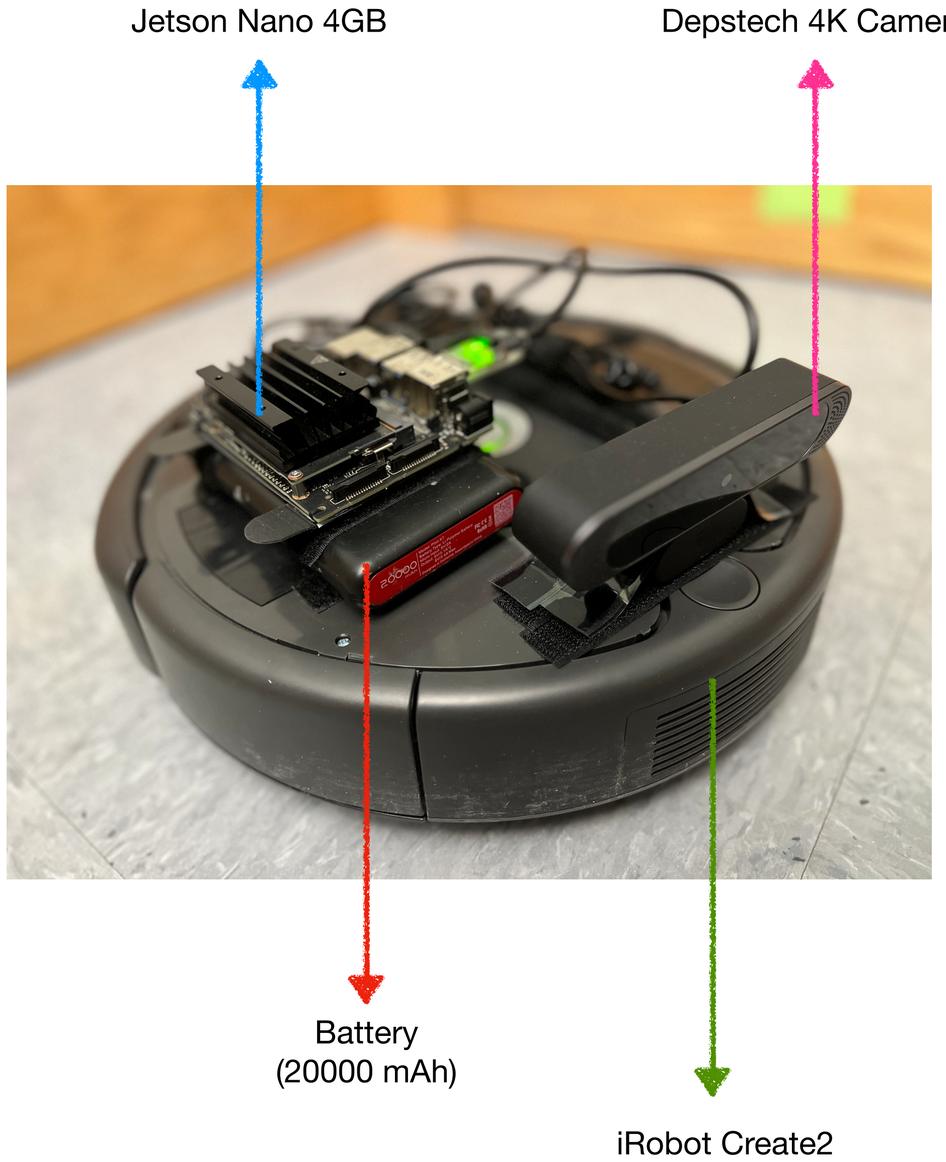
- Remote device
- Policy learning
- Store: Actor π, Two Q networks, Target networks and Replay Buffer

Asynchronous
Batch RL

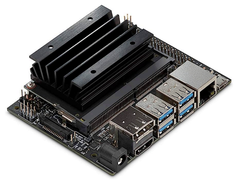
Wang, Y., Vasan, G., & Mahmood, A. R. (2023, May). Real-time reinforcement learning for vision-based robotics utilizing local and remote computers. In *2023 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 9435-9441). IEEE.

Compute requirements for on-device learning

- Send an action every 45 milliseconds
- Replay buffer with 1M samples: ~40GB RAM
    - Stacked RGB images (160 x 90 x 3)
    - Proprioception
- Policy inference time:
    - ~17ms on Jetson Nano
- Time per learning update (Batch size 256):
    - Nvidia 3090Ti GPU: ~60-70ms
    - Jetson Nano: >2s



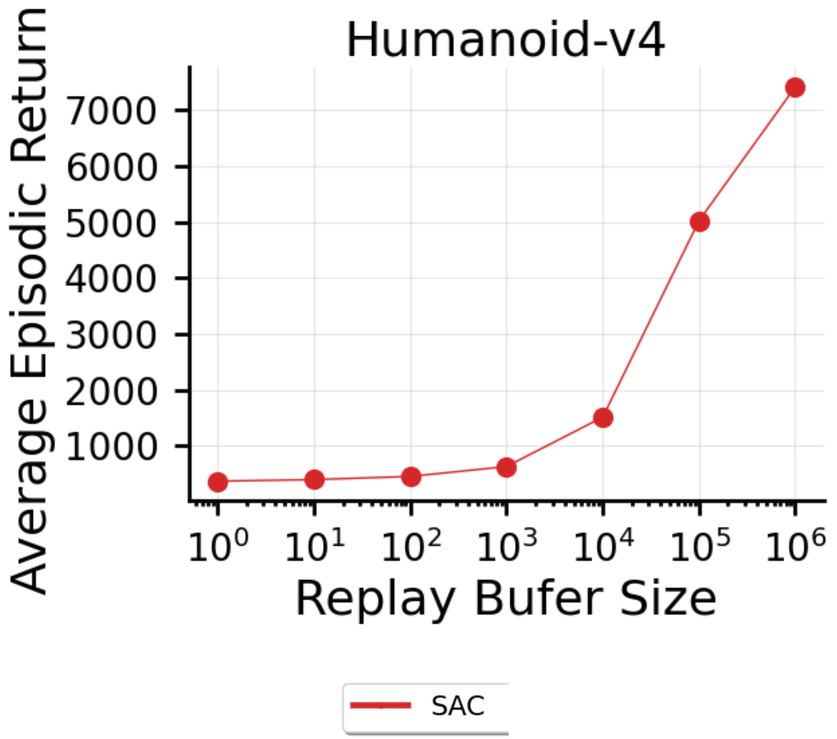
Batch deep RL methods like SAC are ill-suited for real-time on-device learning



Jetson Nano 4GB

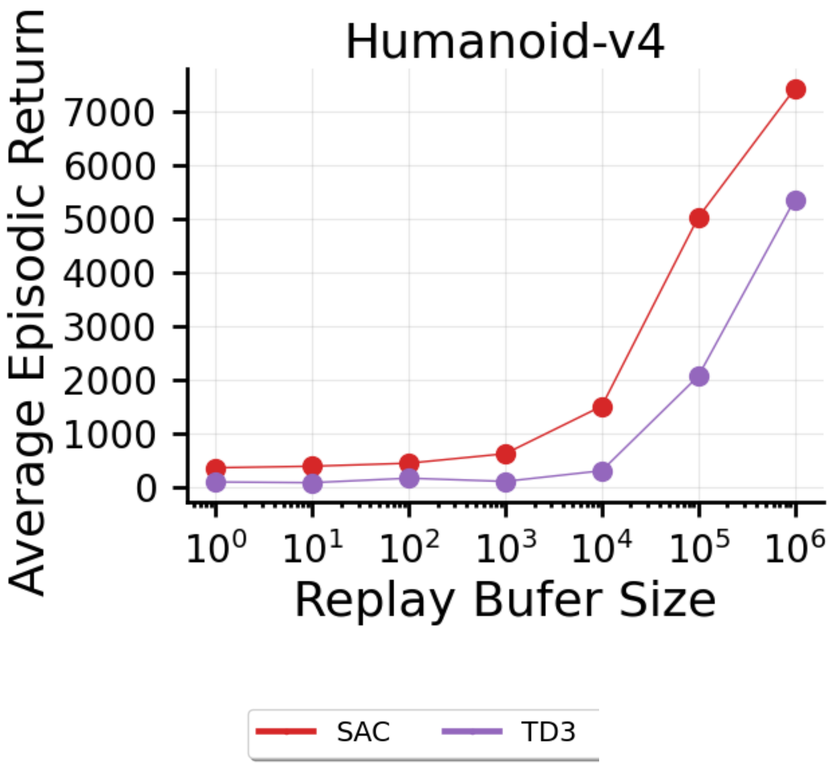
Idea #1: Reduce the replay buffer size and use smaller batch updates that meet the resource constraints

Deep RL Methods Under Real-Time Learning Constraints

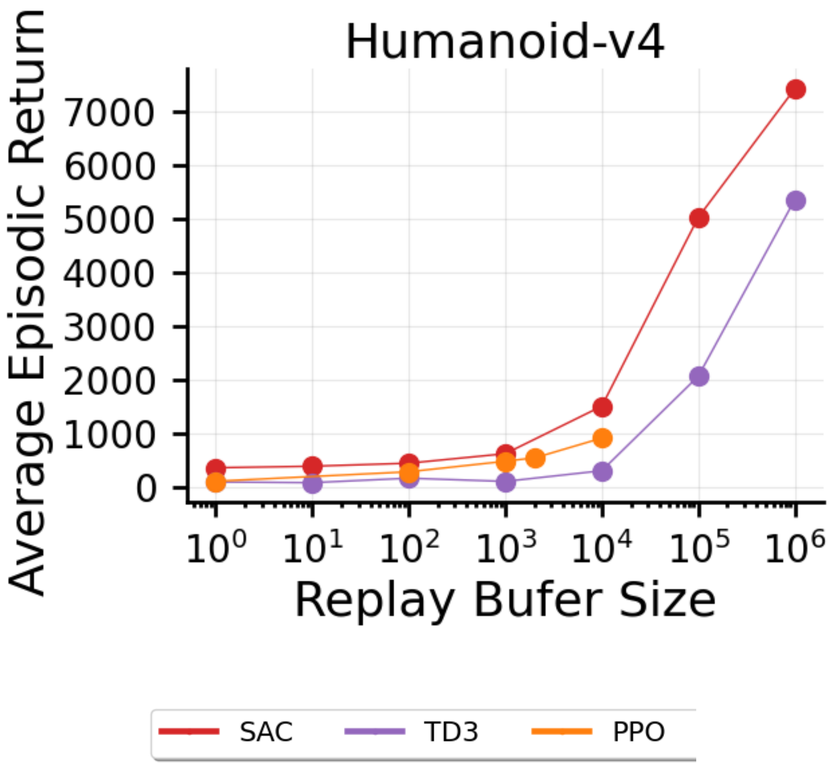


- 10M training steps
- Y-axis reports performance in the last 10K steps, averaged over 30 seeds
- X-axis is in logarithmic scale

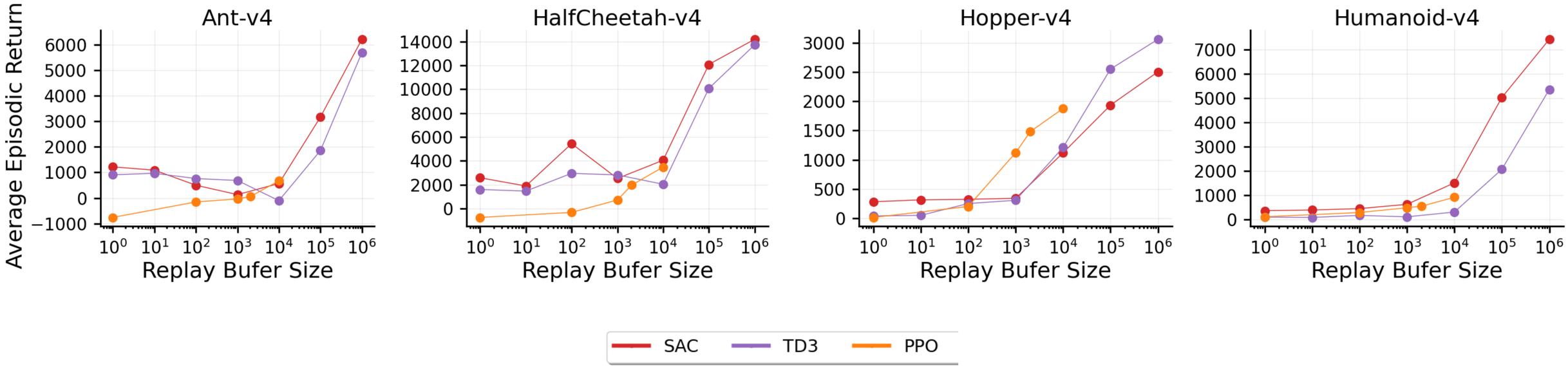
Deep RL Methods Under Real-Time Learning Constraints



Deep RL Methods Under Real-Time Learning Constraints



Deep RL Methods Under Real-Time Learning Constraints



- The learning performance of batch policy gradient methods degrades substantially when the replay buffer size is reduced from their large default values
- PPO, TD3 and SAC fail catastrophically when buffer size is reduced to 1

Idea #2: Use an existing incremental learning method

Incremental One-Step Actor Critic (IAC)

- IAC learns from a continuous stream of data, one sample at a time, without storing any samples
- It uses the likelihood-ratio gradient (LG) estimator
- It is computationally cheap
- Examples of real-time learning with linear function approximation

// TD Error

$$\delta \leftarrow R + \gamma V_\phi(S') - V_\phi(S)$$

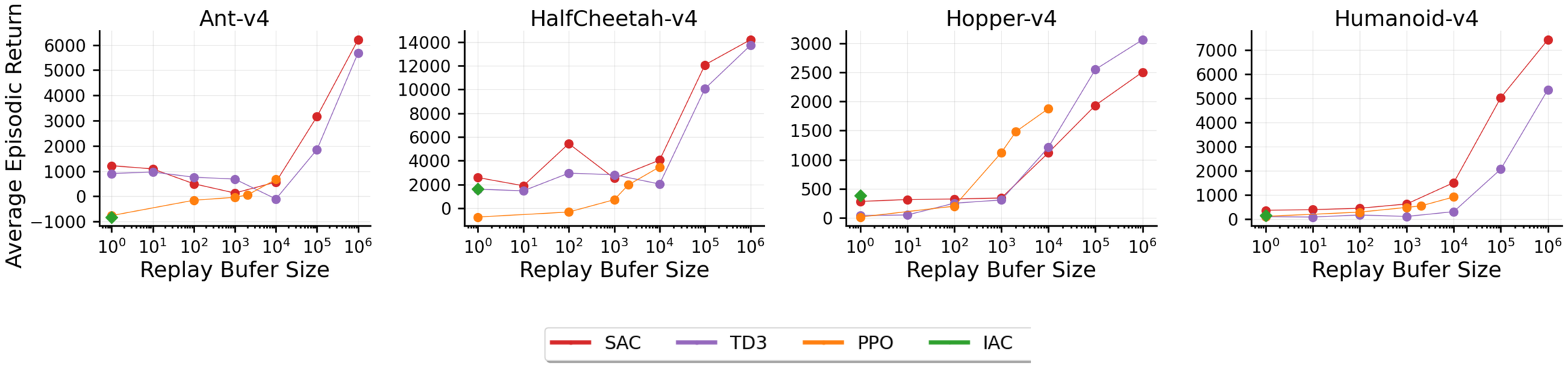
// Critic Update

$$\phi \leftarrow \phi + \alpha_V \, \delta \, \nabla_\phi V_\phi(S)$$

// Actor Update

$$\theta \leftarrow \theta + \alpha_\pi \, \delta \, \nabla_\theta \log \pi_\theta(A \mid S) + \eta \, \nabla_\theta H(\pi_\theta(\cdot \mid S))$$

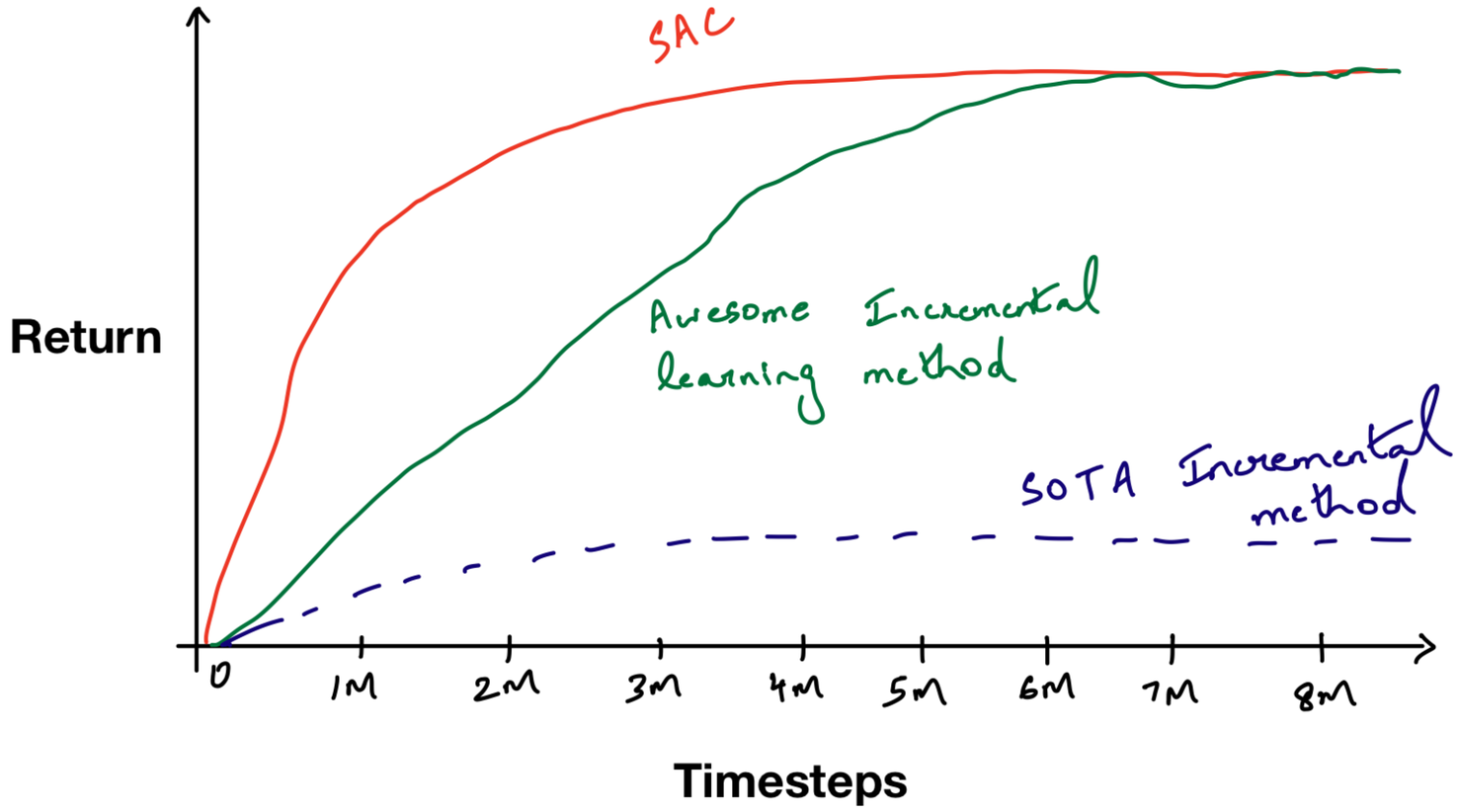
Performance of IAC with Deep Neural Networks



- IAC fails to learn a good policy on these Mujoco environments

A robust incremental method that can leverage deep neural networks for learning in real-time remains an important open challenge

Goal

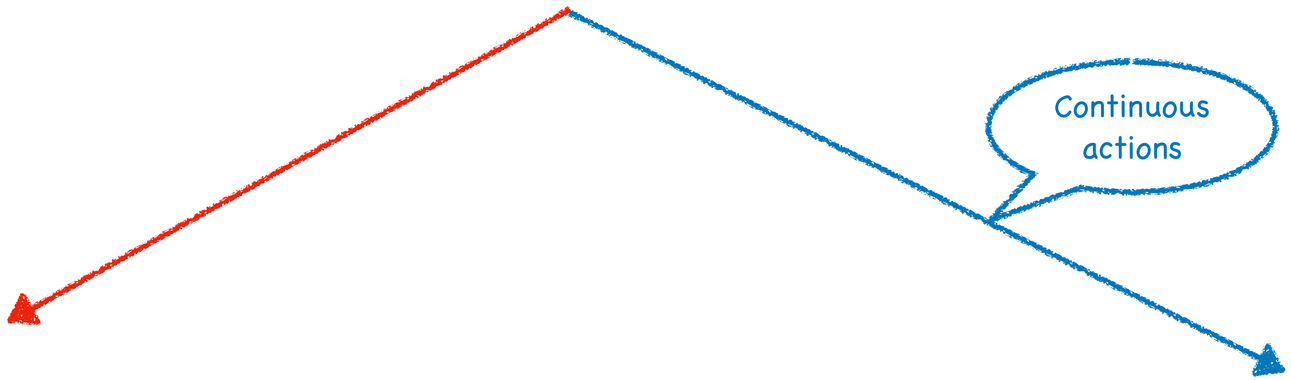


Our incremental algorithm may take longer to learn, but given enough time, it should achieve the final performance of a sample-efficient batch method

We propose a novel incremental deep policy gradient method — Action Value Gradient (AVG)— which does not require batch updates, target networks or a replay buffer

Policy gradients

$$\nabla_\theta J(\theta) \propto \nabla_\theta \int d_{\pi,\gamma}(s)\ \pi_\theta(a \mid s) q_{\pi_\theta}(s,a)\ da\ ds$$



$$\mathbb{E}_{S\sim d_{\pi,\gamma},\ A\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(A \mid S)\ q_{\pi_\theta}(S,A)\right]$$

Likelihood Ratio Gradient
(Sutton et al., 1999)

$$\mathbb{E}_{S\sim d_{\pi,\gamma},\ A\sim\pi_\theta}\left[\nabla_\theta f_\theta(\epsilon; S)\,|_{\epsilon=g_\theta(A;S)}\ \nabla_A q_{\pi_\theta}(S,A;\phi)\right]$$

Reparametrization Gradient
(Parmas et al., 2021, Lan et al. 2021)

	Incremental	Batch
Likelihood Ratio Gradient	REINFORCE Actor Critic	PPO TRPO ACKTR
Reparametrization Gradient	??	DDPG SAC TD3

Action Value Gradient (AVG)

$$A_\theta = f_\theta(\epsilon; S) \quad \epsilon \sim \mathcal{N}(0,1)$$ // Action Sampling

$$\delta \leftarrow R + \gamma(Q_\phi(S', A') - \eta \log \pi_\theta(A' \mid S')) - Q_\phi(S, A_\theta)$$ // TD Error

$$\phi \leftarrow \phi + \alpha_Q \; \delta \; \nabla_\phi Q_\phi(S, A_\theta)$$ // Critic Update

$$\theta \leftarrow \theta + \alpha_\pi \nabla_\theta (Q_\phi(S, A_\theta) - \eta \log \pi_\theta(A_\theta \mid S))$$ // Actor Update

- Notably, AVG is the only existing incremental policy gradient method that uses the reparametrization gradient estimator

Normalization and Scaling

- We use three normalization and scaling techniques in AVG
    - Observation Normalization
    - Normalization of the Penultimate Layer Feature Activations
    - Scaling the Temporal Difference (TD) Error
- These techniques are essential for maintaining good learning dynamics, reducing instability, improving plasticity, and resolving issues related to the scale of large bootstrapped targets

Step 1: Normalize Observations

- $$obs_{norm} = \frac{obs - \mu_{obs}}{\sigma_{obs}}$$
- Estimate sample mean and std for observations
- Commonly used with PPO

Welford, B. (1962). Note on a method for calculating corrected sums of squares and products. Technometrics, 4(3):419–420.

Step 2: Penultimate Norm

- Simply setting penultimate features to unit norm (*penultimate normalization*) can avoid exploding activations
  - $\hat{\psi}_\theta(S) = \dfrac{\psi_\theta(S)}{\|\psi_\theta(S)\|_2}$
- Penultimate features $\psi_\theta(S)$: Post-activations of last hidden layer
- Use for both Actor and Critic

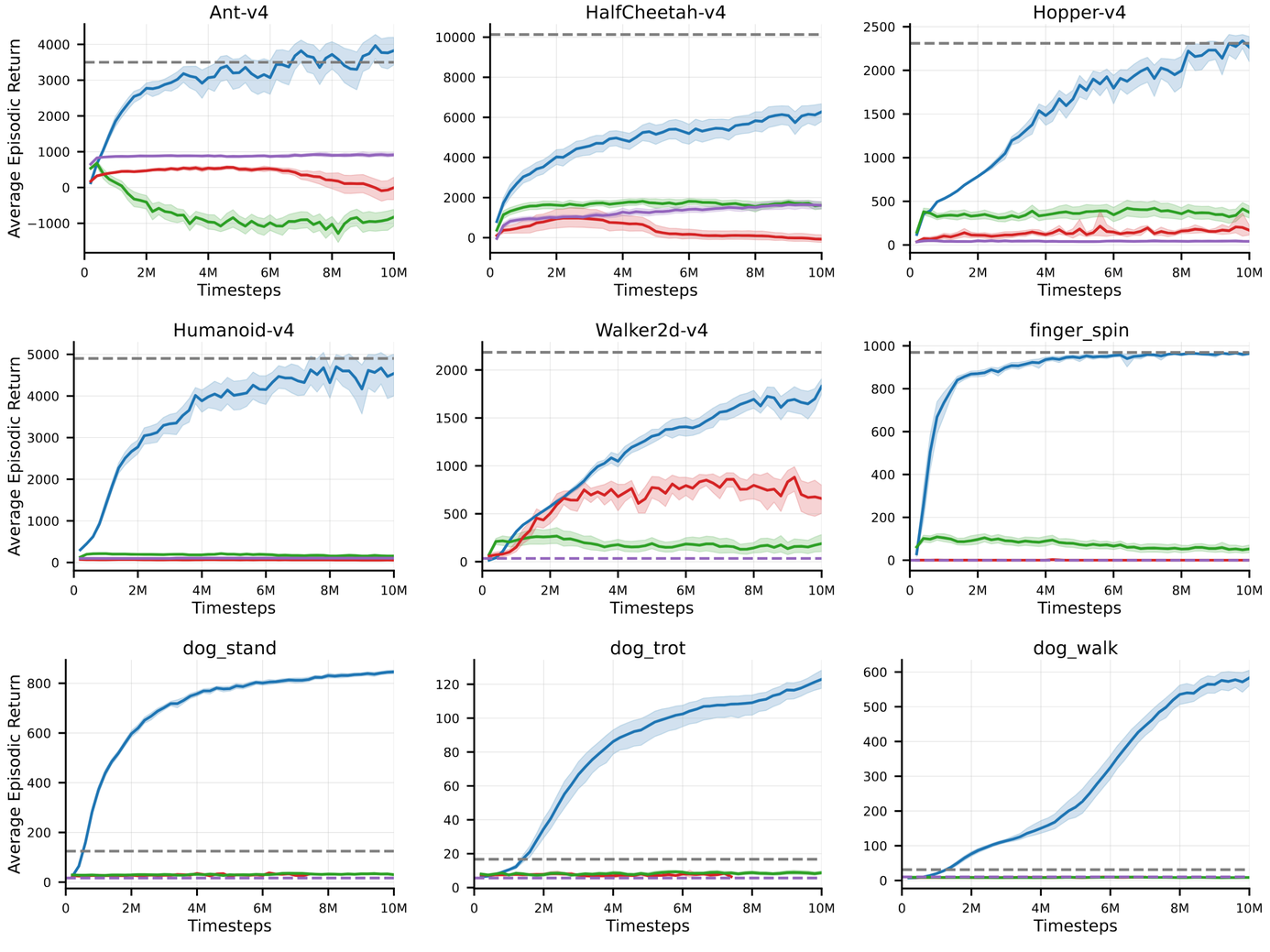
Bjorck, J., Gomes, C. P., & Weinberger, K. Q. (2022). Is high variance unavoidable in rl? a case study in continuous control. International Conference on Learning Representations.

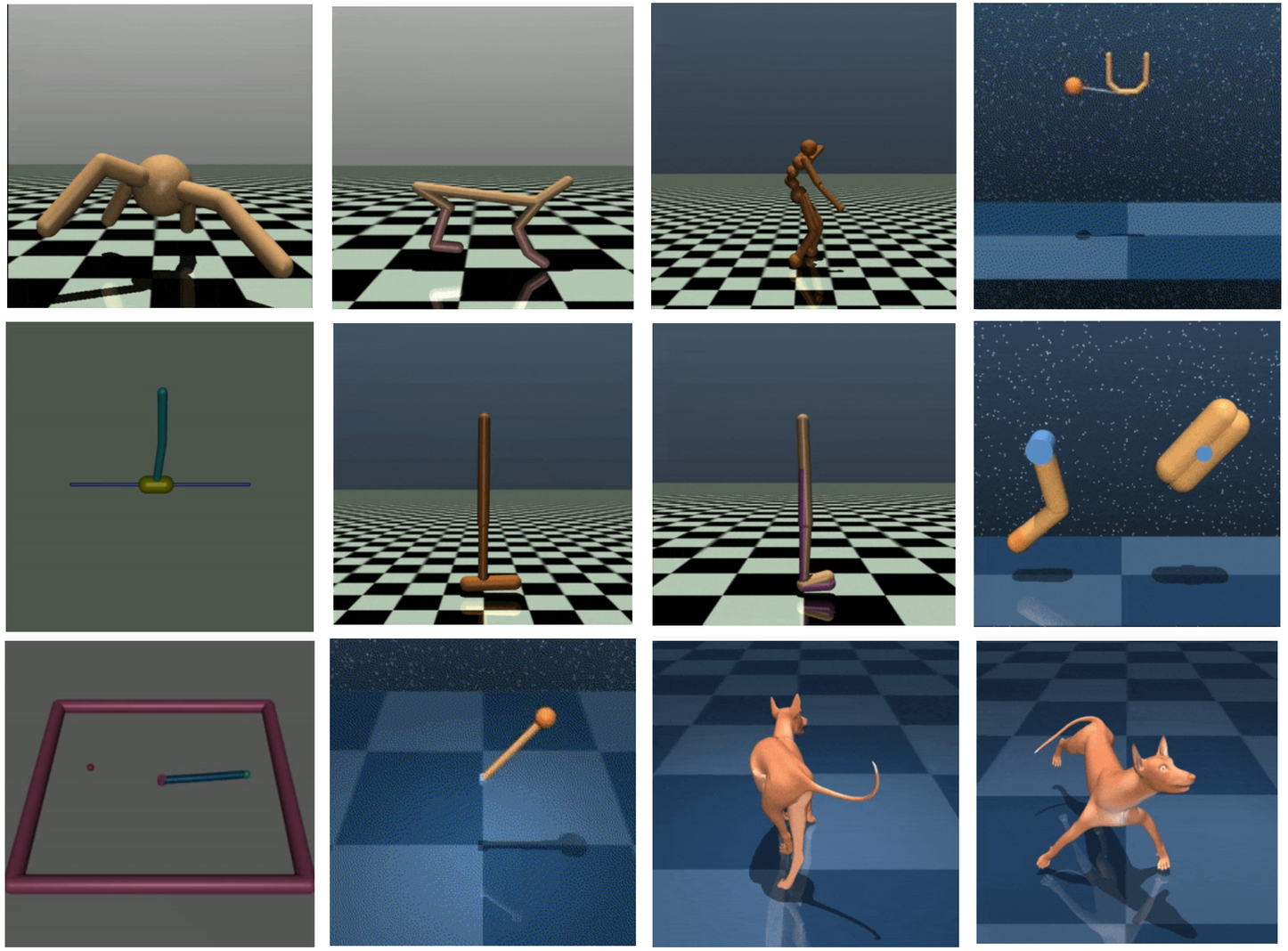
Step 3: Return Scaling

- $\sigma_\delta^2 = \mathbb{V}[R] + \mathbb{V}[\gamma]\mathbb{E}[G^2]$
- Scale TD error: $\delta = \dfrac{\delta}{\sigma_\delta}$
- Maintain a sample mean and std estimate for reward R, discount factor γ and return G to estimate σ_δ

Schaul, T., Ostrovski, G., Kemaev, I., & Borsa, D. (2021). Return-based scaling: Yet another normalisation trick for deep RL. arXiv preprint arXiv:2105.05347.

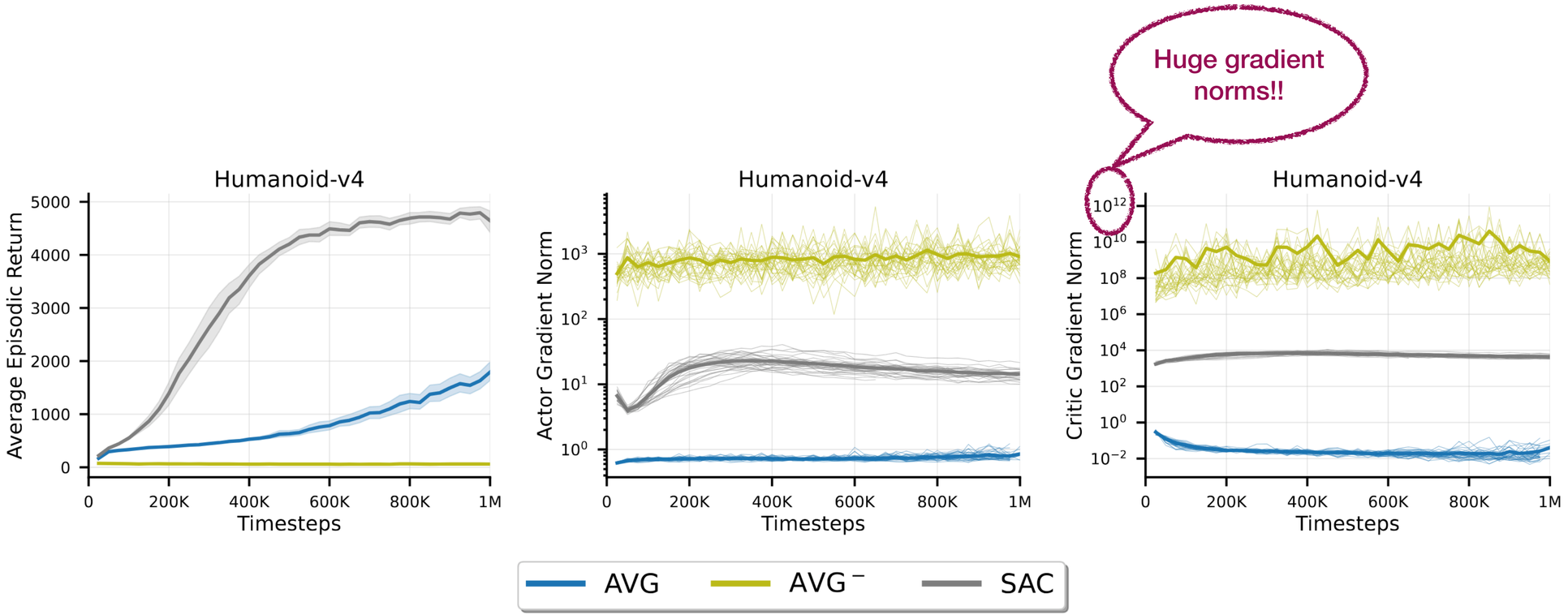
AVG is the only incremental method that learns effectively, often achieving final performance comparable to batch policy gradient methods





A Tale of Large and Noisy Gradients

Instability Without Normalization and Scaling



Hypothesis: Stable learning in streaming methods can be achieved by balancing update magnitudes across time steps and episodes, reducing the influence of outlier experience

Impact of normalization and scaling on policy gradient methods

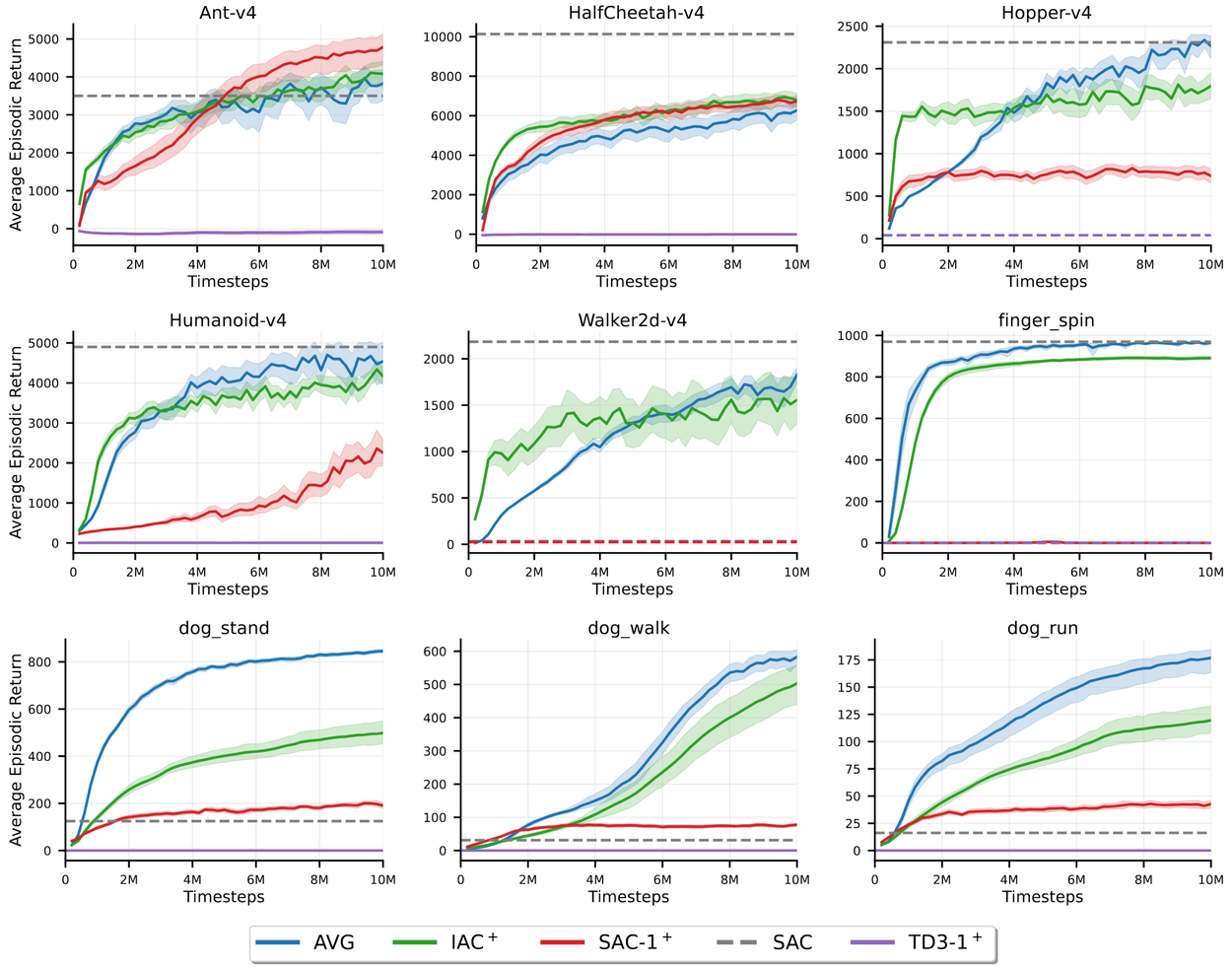


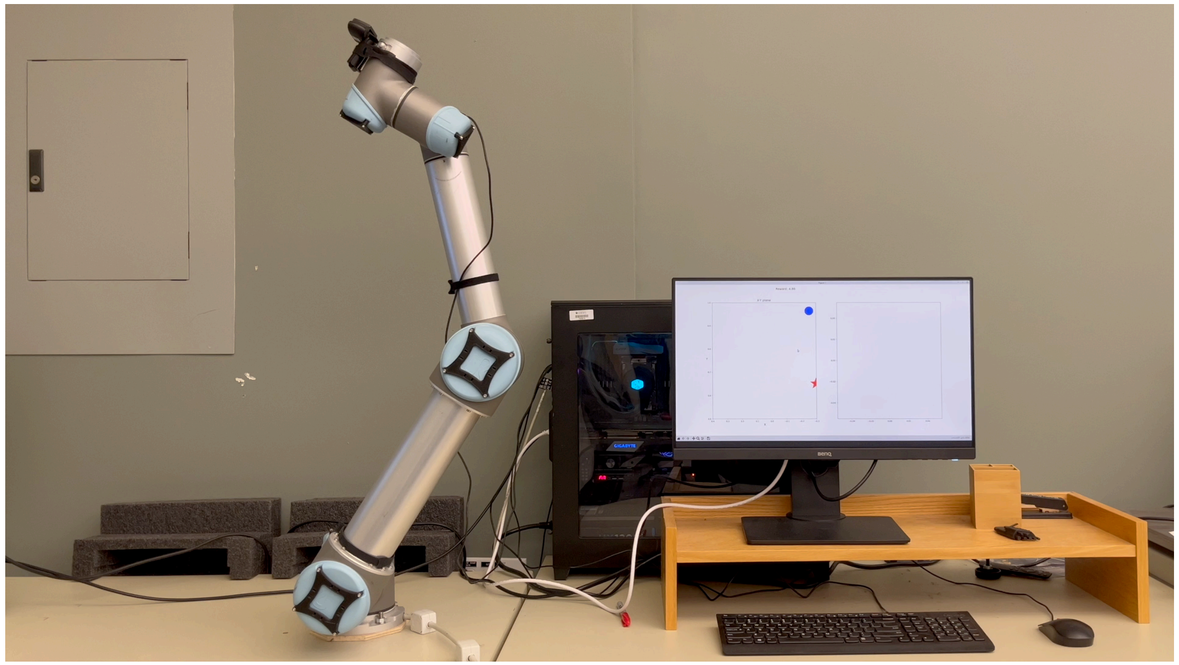
Figure 7: Impact of normalization and scaling on IAC, SAC-1 and TD3-1. Suffix "+" denotes each algorithm plus normalization and scaling.

AVG enabled us to show for the first time effective deep RL with real robots using only incremental updates

Create-Mover Policy After 100K Steps (1x)

Reward: Moving Forward

Learned Solely on Onboard Computer (Jetson Nano)



UR-Reacher-2 Policy After 400K Steps (1x)

Reward: Distance Penalty + Precision Bonus

Thank You :)

github.com/gauthamvasan
https://gauthamvasan.github.io/
vasan@ualberta.ca

Discussion

- Why is incremental deep reinforcement learning important? What real life problems and applications can it help us solve? (e.g., space)
- How can we be more sample efficient with incremental deep RL agents?
- How does it fit with continual learning? How can we have methods that experience minimal forgetting?
- How can we maintain/encourage exploration in incremental methods?

Ablation Study of Normalization & Scaling Techniques

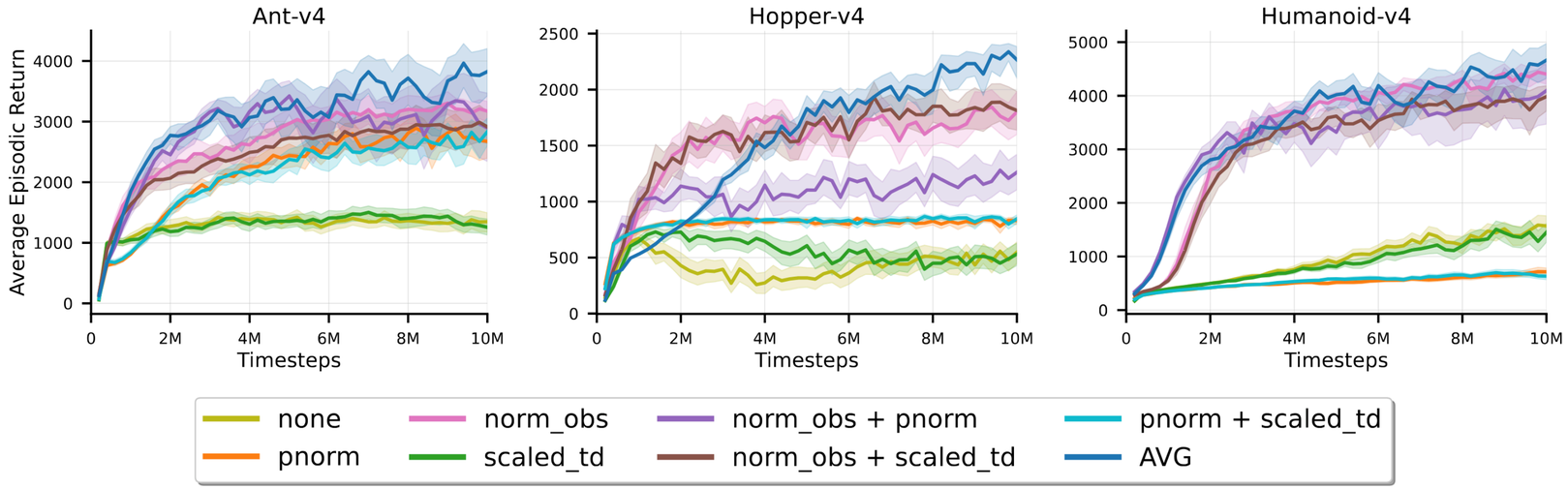


Figure 6: Ablation study of normalization and scaling techniques used with AVG. We plot the learning curves of the best hyperparameter configurations for each task variant. Each solid learning curve is an average of 30 independent runs. The shaded regions represent a 95% confidence interval.

Does AVG benefit from target networks?

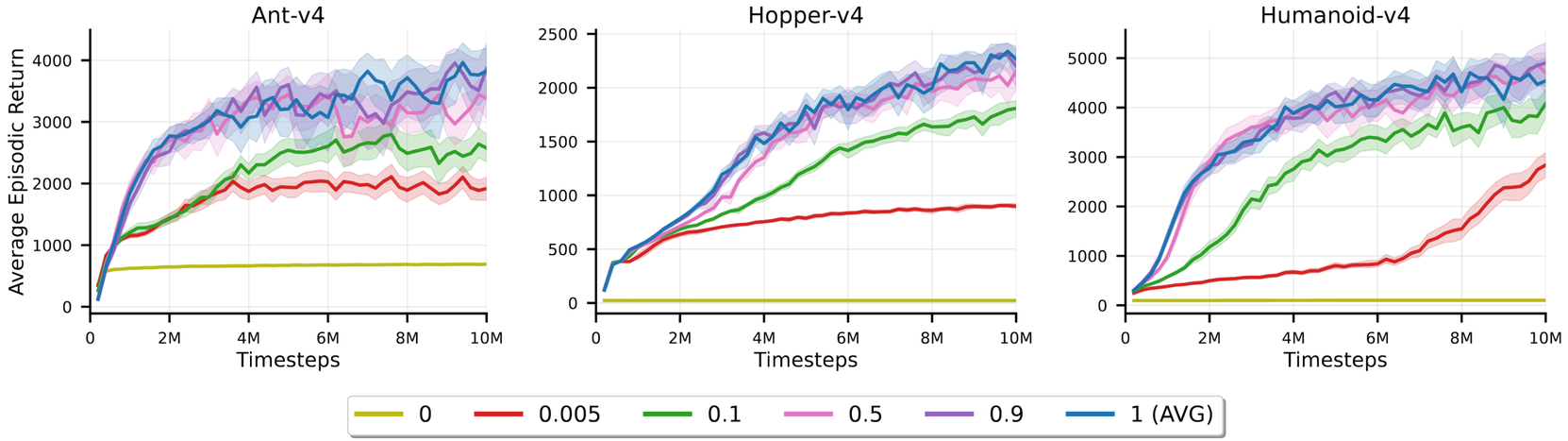


Figure 8: Impact of target Q network on AVG for different values of τ, which represents the Polyak averaging coefficient. Here, $\tau = 0$ corresponds to a fixed target network, and $\tau = 1$ indicates that the current Q-network and the target network are identical, that is, not using a target network.

Alternatives to Penultimate Norm

B.2 The Effect of Other Network Normalization Techniques

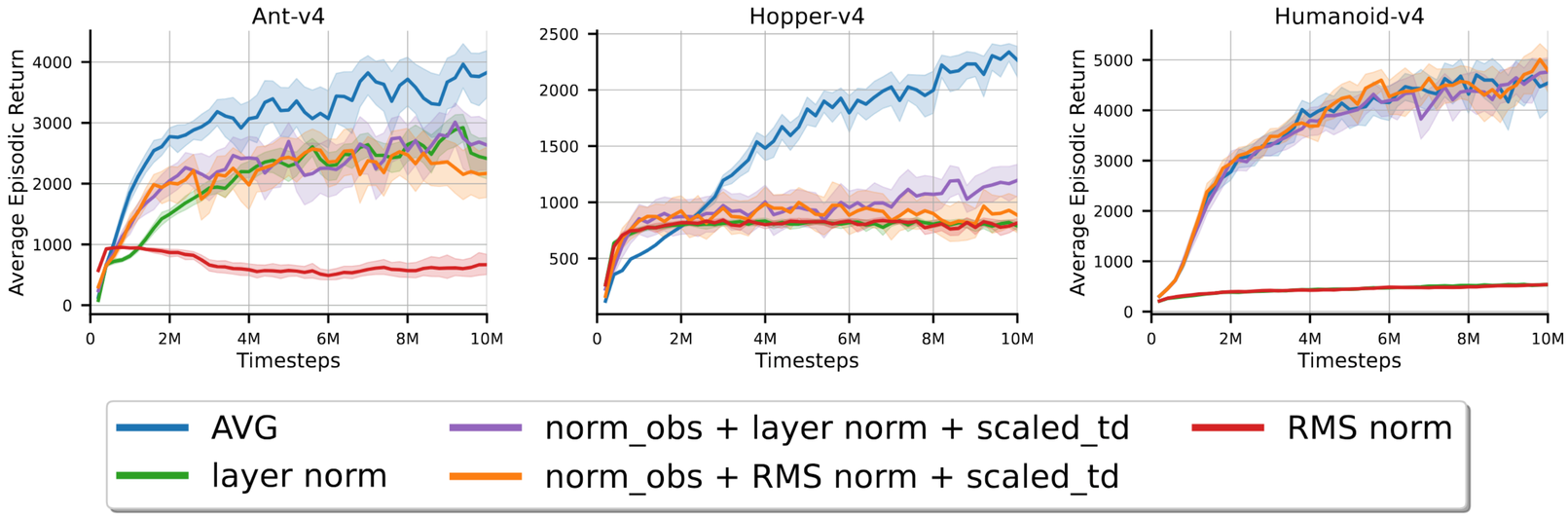


Figure 13: Comparing different neural network feature normalizations — penultimate normalization (pnorm) against layer normalization (layer norm) and RMS normalization (RMS norm)

Robot Experiments

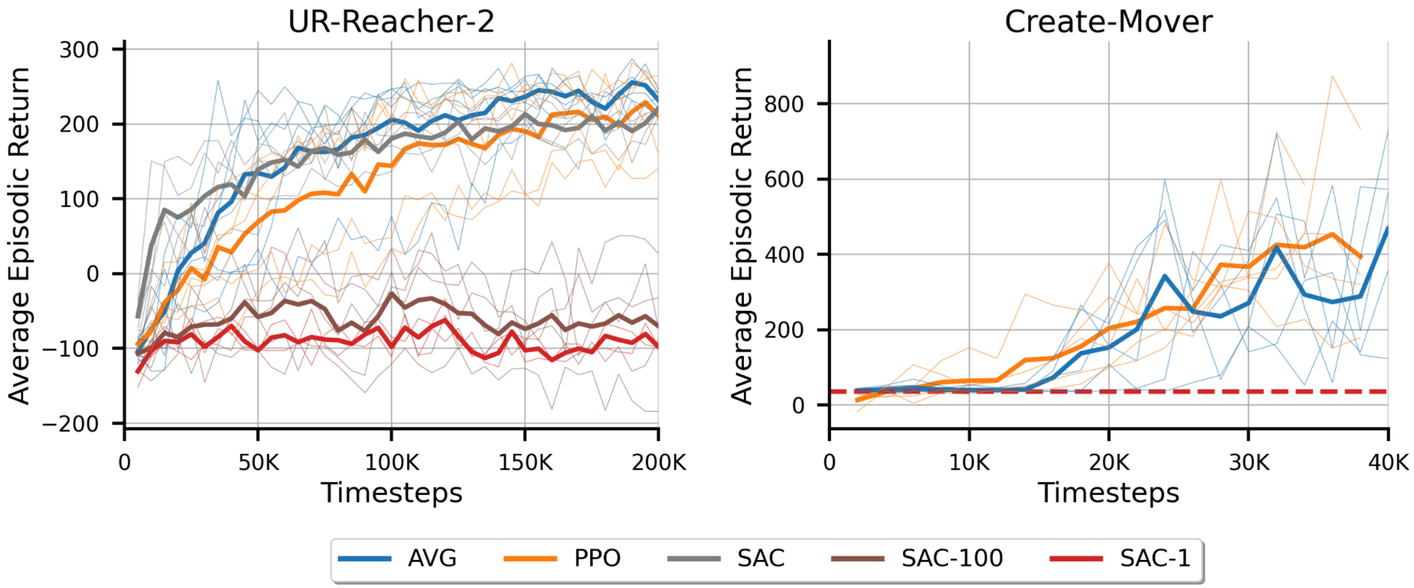


Figure 21: Learning curves on Robots. Comparison of AVG with full PPO & SAC. Note that running SAC and SAC-100 onboard for the Create-Mover task is computationally infeasible.

Algorithm 1 Action Value Gradient (AVG)

Initialize γ, η, α_π, α_Q
θ, ϕ with penultimate normalization
$n \leftarrow 0, \mu \leftarrow 0, \overline{\mu} \leftarrow 0$
$\boldsymbol{n}_\delta \leftarrow [0,0,0], \boldsymbol{\mu}_\delta \leftarrow [0,0,0], \overline{\boldsymbol{\mu}}_\delta \leftarrow [0,0,0]$
for however many episodes **do**
  Initialize S (first state of the episode)
  $S, n, \mu, \overline{\mu}, \_ \leftarrow \texttt{Normalize}(S, n, \mu, \overline{\mu})$
  $G \leftarrow 0$
  while S is not terminal **do**
    $A_\theta = f_\theta(\epsilon; S)$ where $\epsilon \sim \mathcal{N}(0,1)$
    Take action A_θ, observe S', R
    $S', n, \mu, \overline{\mu}, \_ \leftarrow \texttt{Normalize}(S', n, \mu, \overline{\mu})$
    $\sigma_\delta, \boldsymbol{n}_\delta, \boldsymbol{\mu}_\delta, \overline{\boldsymbol{\mu}}_\delta \leftarrow$
      $\texttt{ScaleTDError}(R, \gamma, \emptyset, \boldsymbol{n}_\delta, \boldsymbol{\mu}_\delta, \overline{\boldsymbol{\mu}}_\delta)$
    $G \leftarrow G + R$
    $A' \sim \pi_\theta(\cdot|S')$
    $\delta \leftarrow R + \gamma(Q_\phi(S', A') - \eta \log \pi_\theta(A'|S'))$
      $-Q_\phi(S, A_\theta)$
    $\delta \leftarrow \delta / \sigma_\delta$
    $\phi \leftarrow \phi + \alpha_Q \delta \, \nabla_\phi \, Q_\phi(S, a)|_{a=A_\theta}$
    $\theta \leftarrow \theta + \alpha_\pi \nabla_\theta (Q_\phi(S, A_\theta) - \eta \log \pi_\theta(A_\theta|S))$
    $S \leftarrow S'$
  end while
  $\sigma_\delta, \boldsymbol{n}_\delta, \boldsymbol{\mu}_\delta, \overline{\boldsymbol{\mu}}_\delta \leftarrow$
    $\texttt{ScaleTDError}(R, 0, G, \boldsymbol{n}_\delta, \boldsymbol{\mu}_\delta, \overline{\boldsymbol{\mu}}_\delta)$
end for