NEURAL INFORMATION PROCESSING SYSTEMS

UNIVERSITY OF ALBERTA

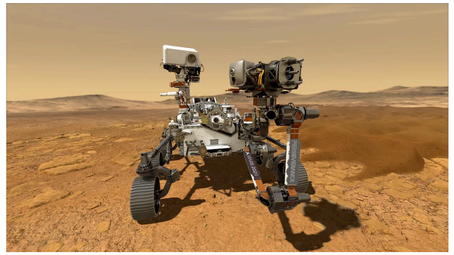
Deep Policy Gradient Methods Without Batch Updates, Target Networks, or Replay Buffers

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

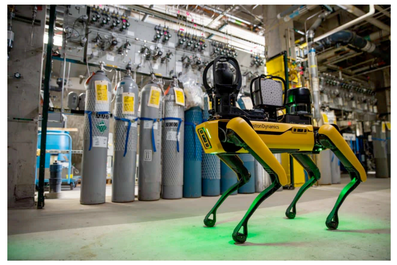
Canada NRC·CNRC CIFAR

amii

Real-Time Learning



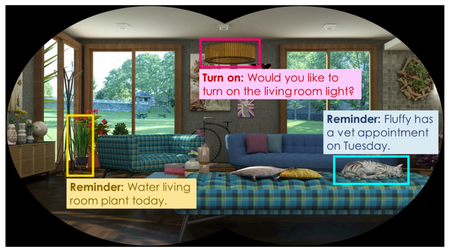
Mars Perseverance Rover



Industrial Inspection With Boston Dynamics Spot

Apple Watch

iRobot Roomba



Application of an AR Headset

Nest Thermostat

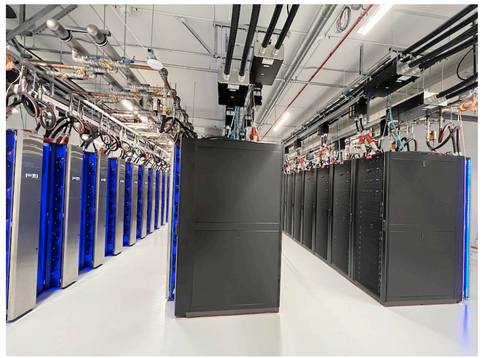
Tesla Autopilot

Loona Pet Robot

- 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

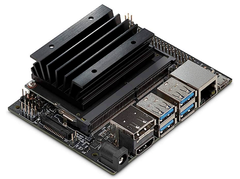
Real-Time Learning Constraints

Available during training



Compute Cluster

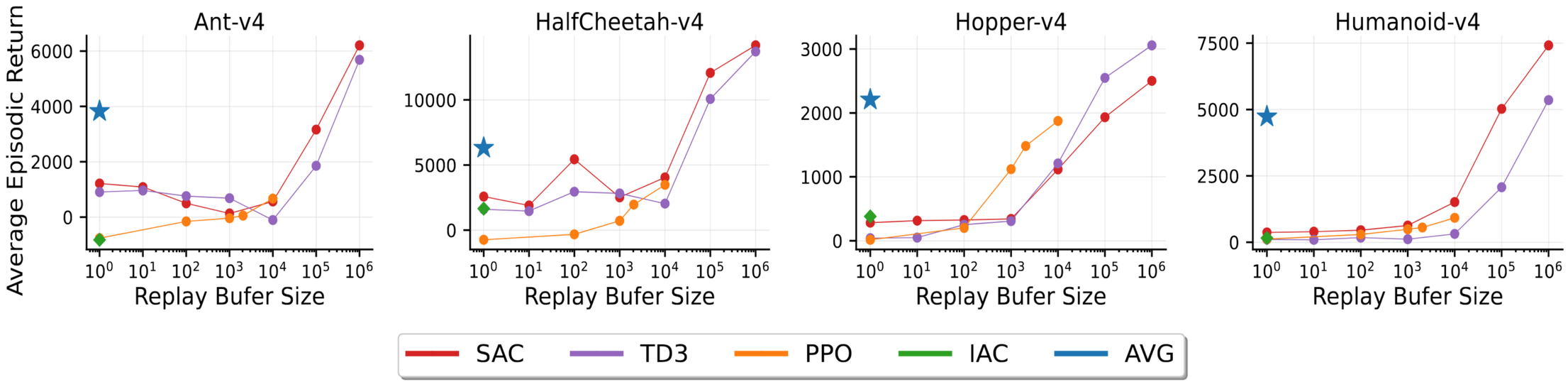
Available after deployment



Edge Device

- Deep reinforcement learning (RL) methods have steep resource requirements due to their use of large replay buffers, target networks and computationally intensive batch updates
    - They necessitate the use of large compute clusters for effective learning
- Current deep RL methods are ill-suited for on-device, real-time learning

What Happens When we use 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

Incremental Learning

- In general, it is computationally infeasible to store all robot experience throughout its life and iteratively learn over the entire replay buffer
- We need methods that learn on the fly, *incrementally,* where updates only use the most recent sample without batch updates or a replay buffer
- Incremental RL methods are more resource-friendly and amenable to real-time updates
- They support on-device learning, which is critical to preserving the privacy and security of the user
- However, incremental policy gradient methods are rarely used in deep RL applications because they struggle to learn effectively with deep neural networks.

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

Action Value Gradient (AVG)

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

$$\phi \leftarrow \phi + \alpha_Q \delta \, \nabla_\phi Q_\phi(S, a)\,|_{a=A_\theta} \qquad \text{// Critic Update}$$

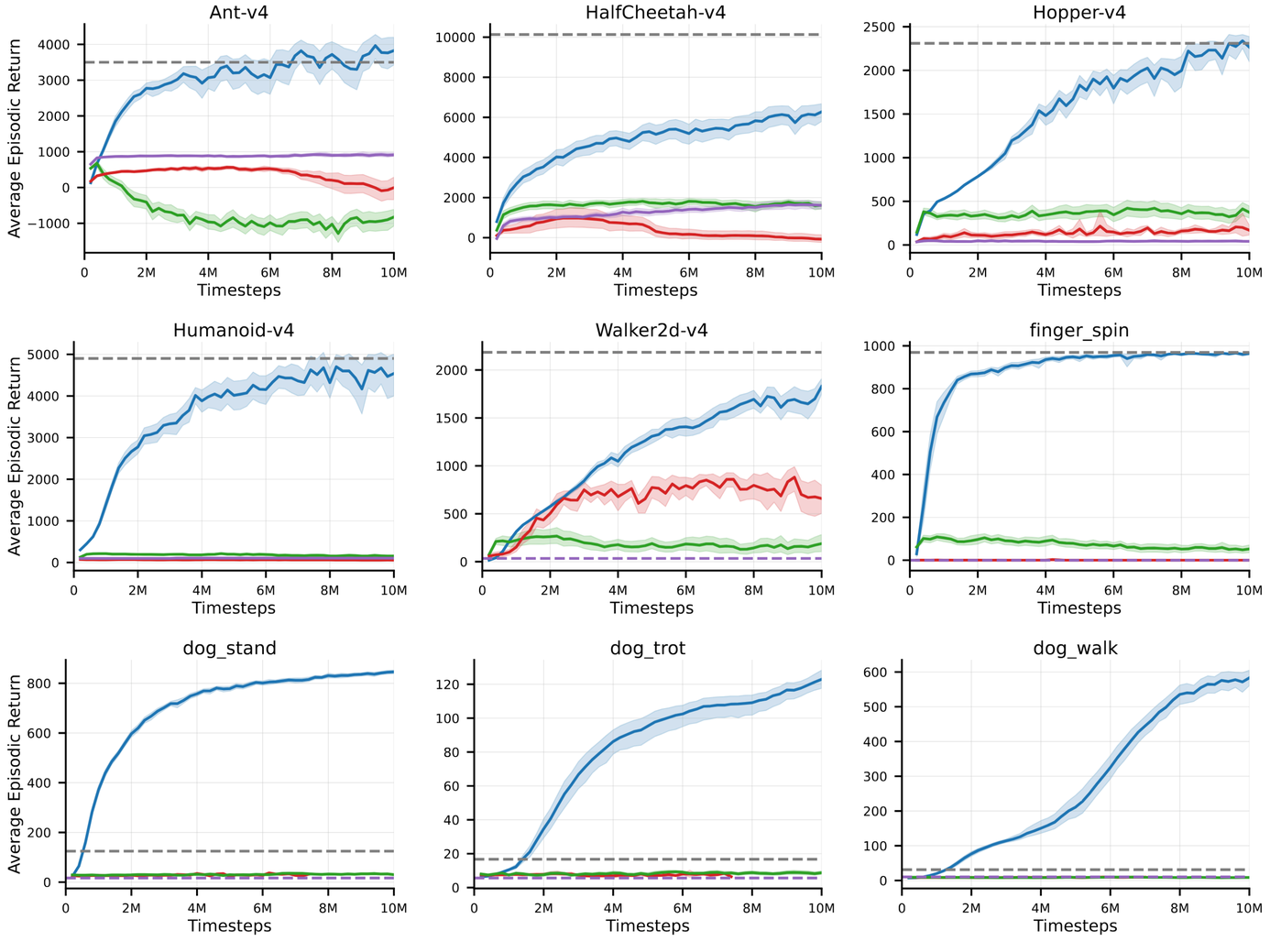
$$\theta \leftarrow \theta + \alpha_\pi \nabla_\theta (Q_\phi(S, A_\theta) - \eta \log \pi_\theta(A_\theta \mid S)) \qquad \text{// Actor Update}$$

- 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

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

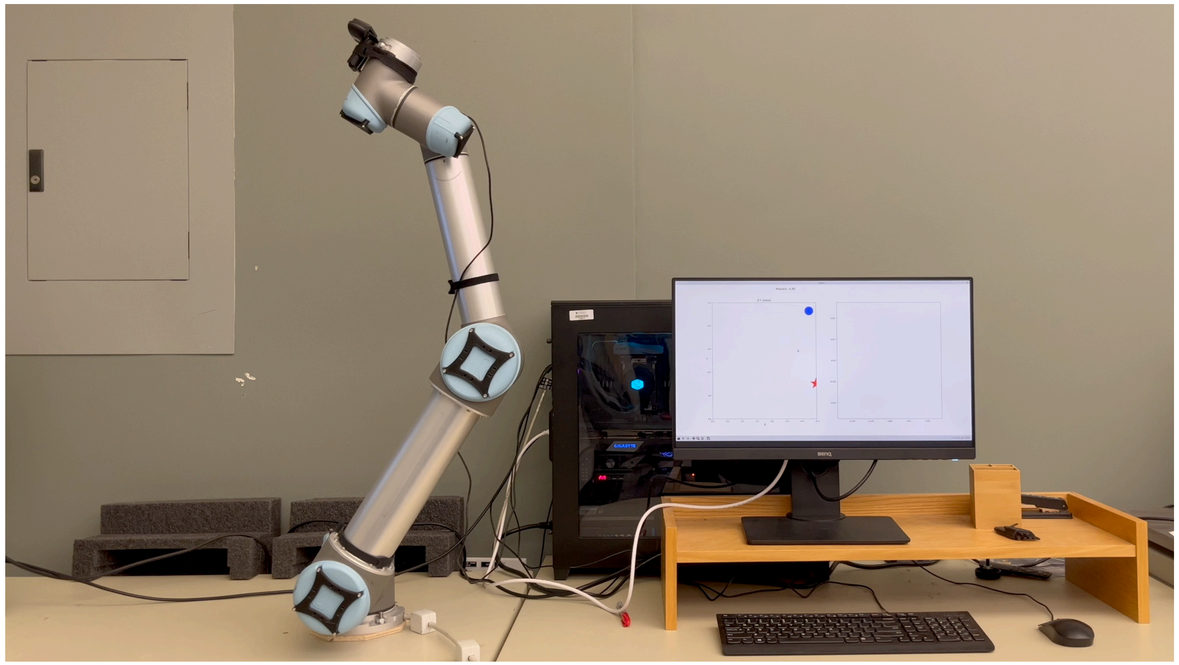


AVG enabled us to show for the first time effective deep RL with real robots using only incremental updates

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 :)

https://github.com/gauthamvasan/avg

https://gauthamvasan.github.io/

vasan@ualberta.ca