Off-Policy Learning (Part 2)

Espeholt, Lasse et al. “IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures.” *ICML* (2018). http://arxiv.org/abs/1802.01561.

Haarnoja, Tuomas et al. “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor.” *ICML* (2018). http://arxiv.org/abs/1801.01290.

Yugdeep Bangar
April 9, 2019

Leveraging Off-Policy methods

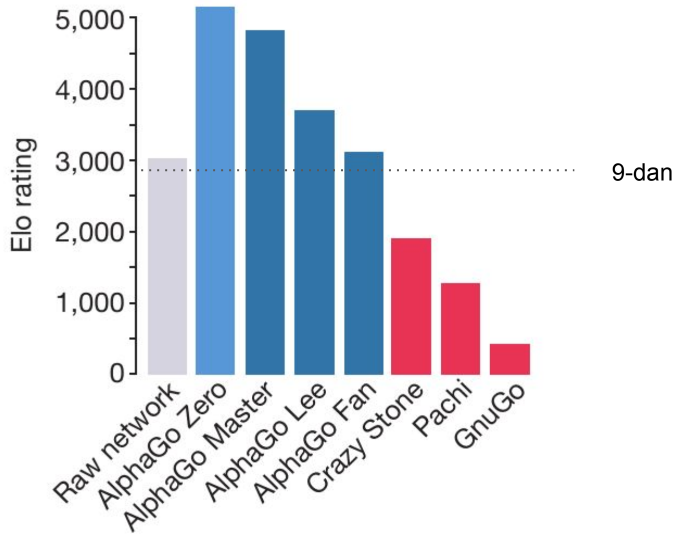
IMPALA - Distributed architecture for greater utilization of GPUs

Soft-Actor Critic - Practical RL for non-simulation environments

IMPALA

(Importance Weighted Actor-Learner Architectures)

Progress in Deep RL for expert agents



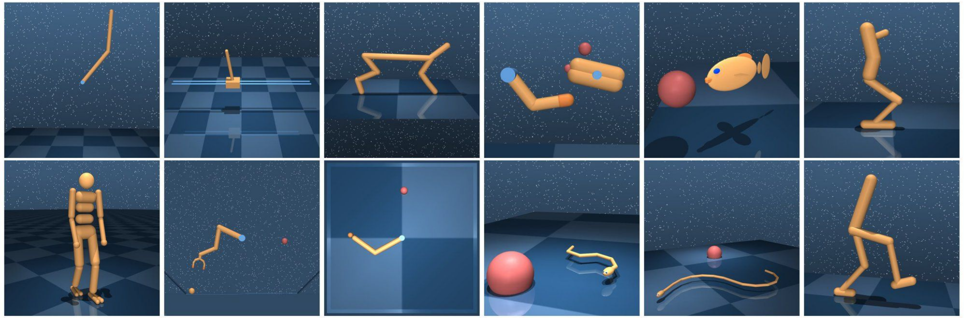
Silver, Schrittwieser, simonyan et al. (2017)

General Agents

Atari

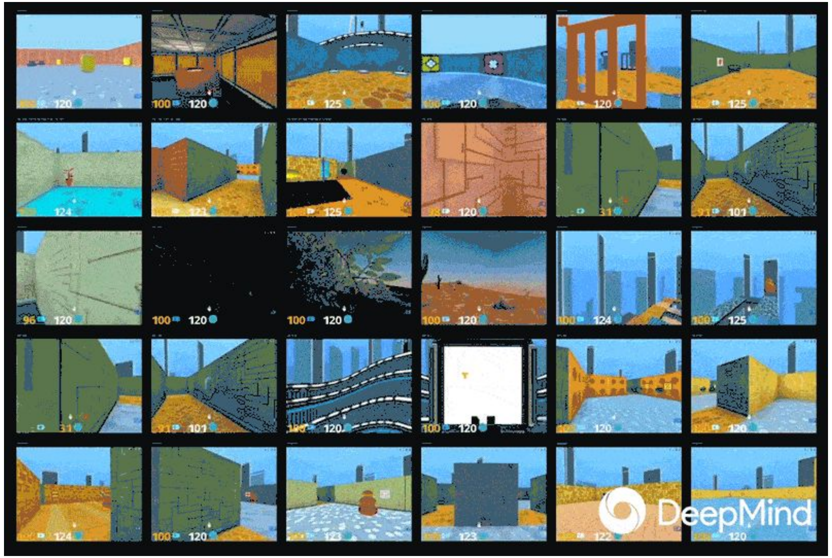
General Agents

dm_contol



General Agents

DMLab-30



Objective

Solve large collection of tasks (e.g. DMLab-30), with

a single reinforcement learning agent (network), and

a single set of parameters

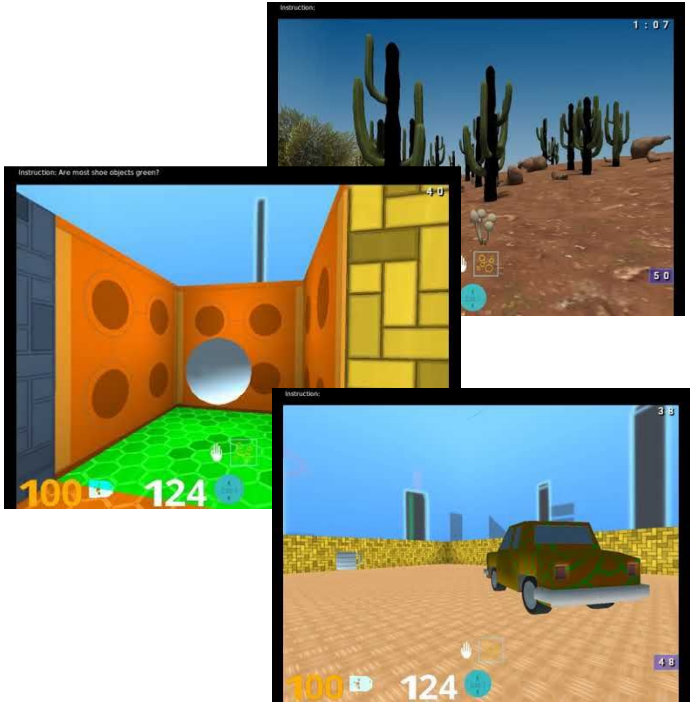
Challenges for a general agent

Data efficiency - (number of tasks) * (hundreds of millions of frames for each task) ?

Stability - multiple hyperparameters?

Scale - bigger networks?

Task interference - interference or positive transfer?



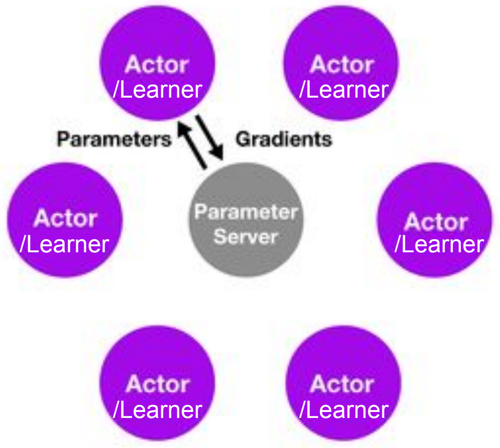
Approach

A scalable distributed agent (IMPALA)

Off-policy correction method V-trace

Asynchronous Advantage Actor-Critic (A3C)

- Agent learns a policy and a state value function
- Uses bootstrapped n-step return to reduce variance over REINFORCE with a baseline
- Distributed experience collection



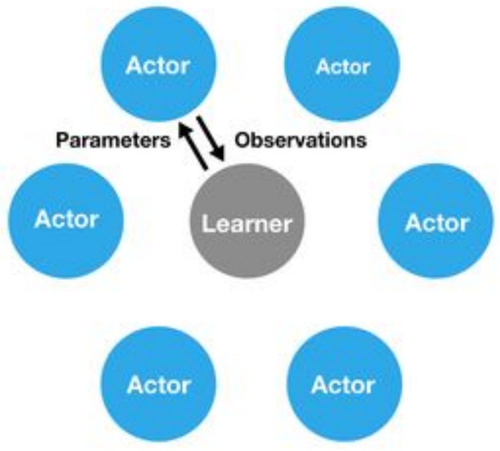
- Adding more actor/learners leads to stale gradients
- Not GPU friendly

IMPALA - Single Learner

Centralized learner(s) and distributed actors

Actors receive parameters but send observations

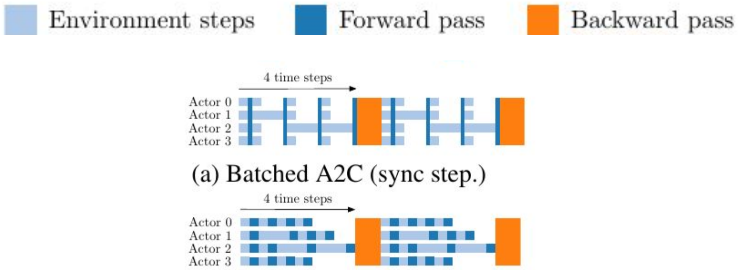
Centralized learner can parallelize as much of the forward and backward passes as possible



Update timeline

Batched A2C

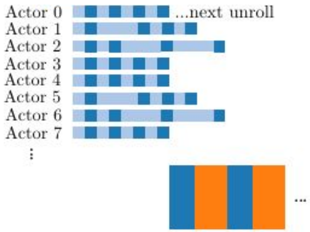
- Rendering time variance
- Low GPU utilisation



(b) Batched A2C (sync traj.)

IMPALA - decoupled backward pass

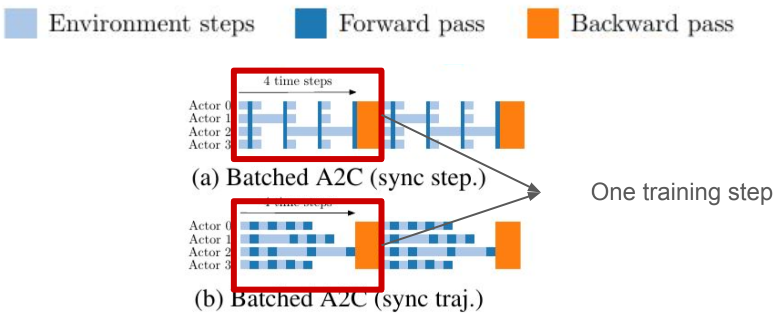
- Acting decoupled from learning
- Actor parameters can lag by several updates



Update timeline

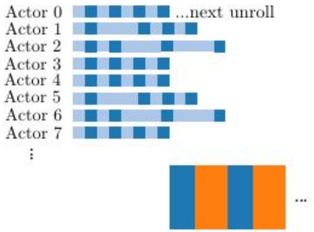
Batched A2C

- Rendering time variance
- Low GPU utilisation



IMPALA - decoupled backward pass

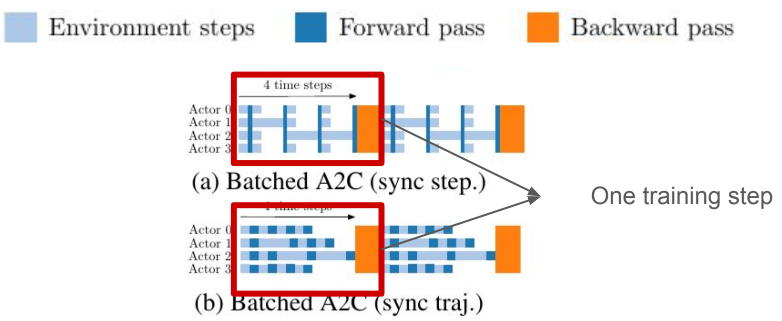
- Acting decoupled from learning
- Actor parameters can lag by several updates



Update timeline

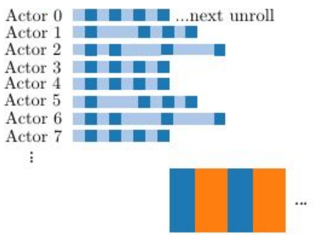
Batched A2C

- Rendering time variance
- Low GPU utilisation



IMPALA - decoupled backward pass

- Acting decoupled from learning
- Actor parameters can lag by several updates



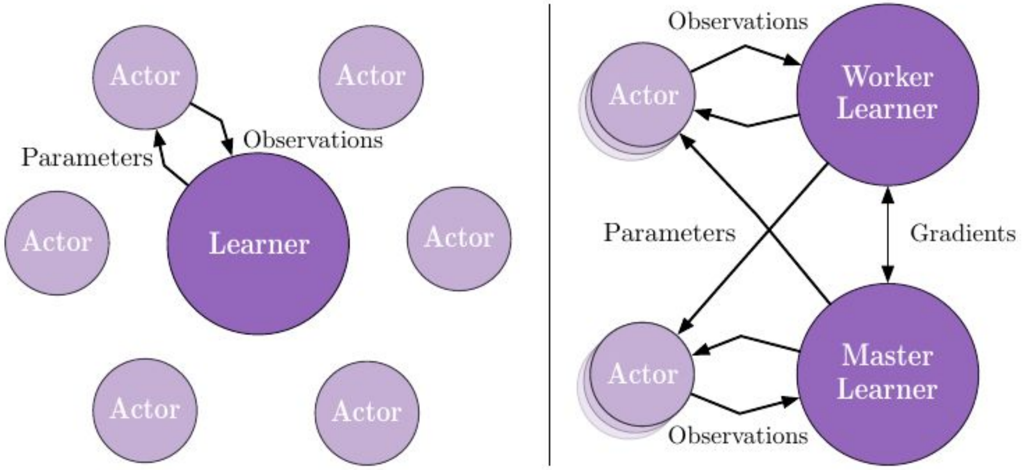
Easier to deal with stale experiences (using off-policy learning) than stale gradients

Actor-critic setup

- Learner off-policy π
- Baseline function V^{π}
- Local policy μ

GPU-accelerated learner

Distributed actors



Problem: policy-lag between the actors and learner

V-trace

Principled off-policy advantage actor critic called V-Trace

$$v_s \stackrel{\text{def}}{=} V(x_s) + \sum_{t=s}^{s+n-1} \gamma^{t-s} \Big(\prod_{i=s}^{t-1} c_i \Big) \underbrace{\rho_t \big(r_t + \gamma V(x_{t+1}) - V(x_t) \big)}_{\delta_t V}$$

where $\rho_i \stackrel{\text{def}}{=} \min\left(\bar{\rho}, \frac{\pi(a_i|x_i)}{\mu(a_i|x_i)}\right)$ and $c_i \stackrel{\text{def}}{=} \min\left(\bar{c}, \frac{\pi(a_i|x_i)}{\mu(a_i|x_i)}\right)$

ρ_i : which value function μ or π

c_i : speed of convergence

Weights are truncated (at most 1) to reduce variance

V-trace

Principled off-policy advantage actor critic called V-Trace

$$v_s \stackrel{\text{def}}{=} V(x_s) + \sum_{t=s}^{s+n-1} \gamma^{t-s} \Big(\prod_{i=s}^{t-1} c_i \Big) \underbrace{\rho_t \big(r_t + \gamma V(x_{t+1}) - V(x_t) \big)}_{\delta_t V}$$

where $\rho_i \stackrel{\text{def}}{=} \min \big(\bar{\rho}, \boxed{\frac{\pi(a_i|x_i)}{\mu(a_i|x_i)}} \big)$ and $c_i \stackrel{\text{def}}{=} \min \big(\bar{c}, \frac{\pi(a_i|x_i)}{\mu(a_i|x_i)} \big)$

ρ_i : which value function μ or π

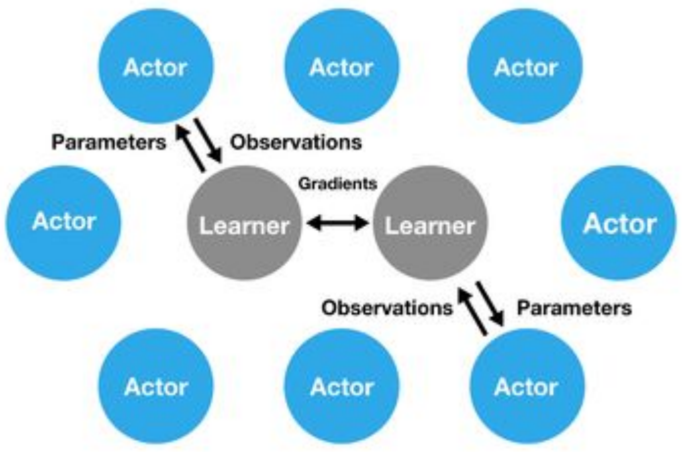
c_i : speed of convergence

Weights are truncated (at most 1) to reduce variance

IMPALA - multiple learners

One or more GPU (or TPU) learners

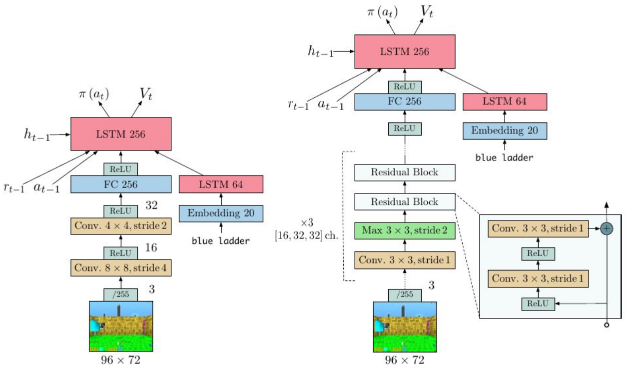
Many CPU actors



Evaluation

Two networks

- Small CNN-LSTM
- Deep ResNet CNN-LSTM



Two test suites

- Atari-57
- DMLab-30 - language, memory, foraging, navigation

Throughput

Architecture	CPUs	GPUs[1]	FPS[2]	
Single-Machine			Task 1	Task 2
A3C 32 workers	64	0	6.5K	9K
Batched A2C (sync step)	48	0	9K	5K
Batched A2C (sync step)	48	1	13K	5.5K
Batched A2C (sync traj.)	48	0	16K	17.5K
Batched A2C (dyn. batch)	48	1	16K	13K
IMPALA 48 actors	48	0	17K	20.5K
IMPALA (dyn. batch) 48 actors[3]	48	1	21K	24K
Distributed				
A3C	200	0	46K	50K
IMPALA	150	1	80K	
IMPALA (optimised)	375	1	200K	
IMPALA (optimised) batch 128	500	1	250K	

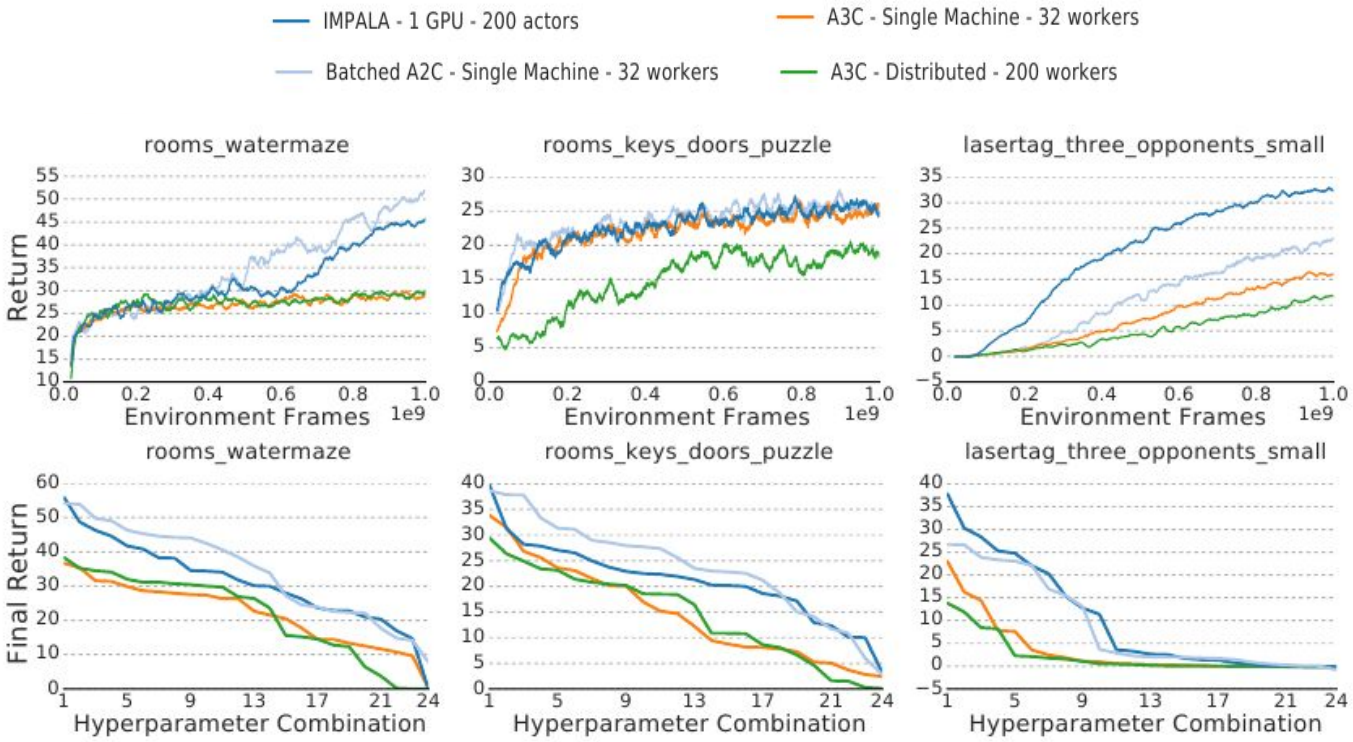
[1] Nvidia P100 [2] In frames/sec (4 times the agent steps due to action repeat). [3] Limited by amount of rendering possible on a single machine.

Throughput

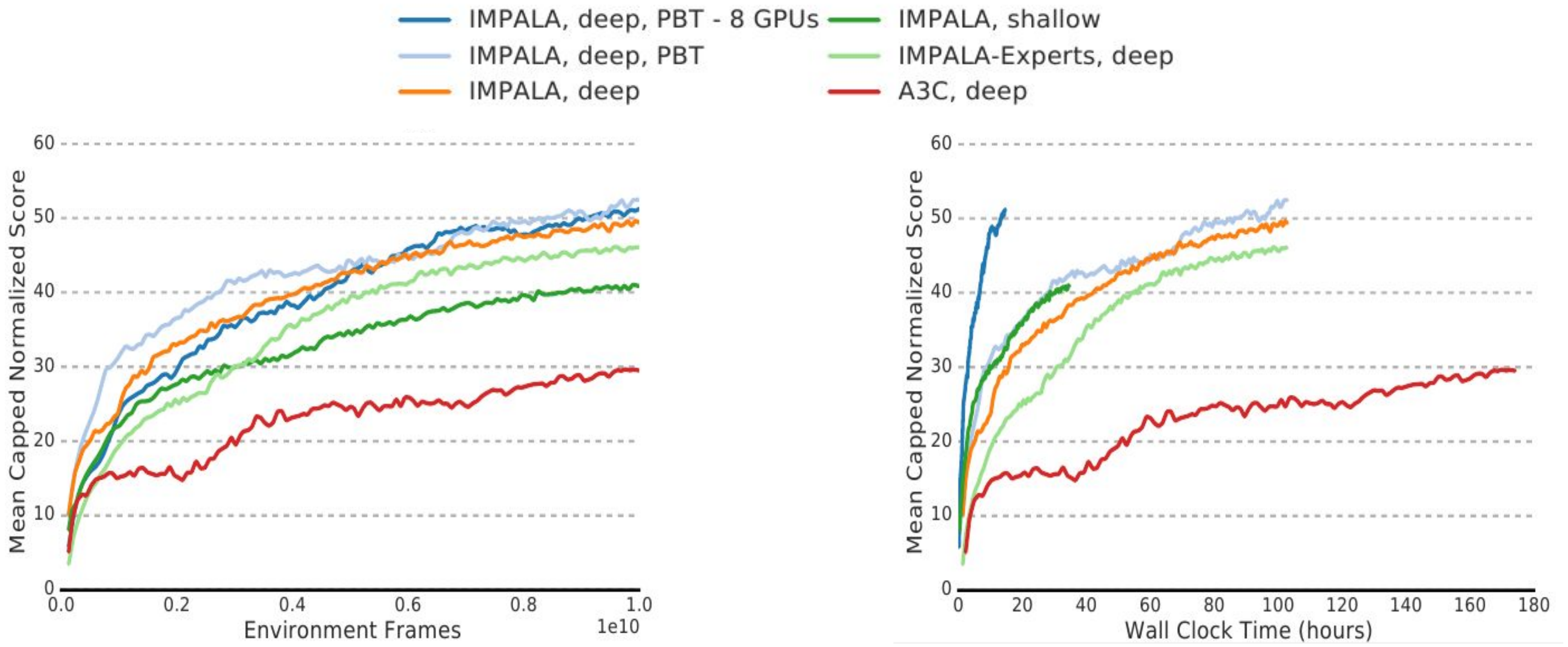
Architecture	CPUs	GPUs[1]	FPS[2]	
Single-Machine			Task 1	Task 2
A3C 32 workers	64	0	6.5K	9K
Batched A2C (sync step)	48	0	9K	5K
Batched A2C (sync step)	48	1	13K	5.5K
Batched A2C (sync traj.)	48	0	16K	17.5K
Batched A2C (dyn. batch)	48	1	16K	13K
IMPALA 48 actors	48	0	17K	20.5K
IMPALA (dyn. batch) 48 actors[3]	48	1	21K	24K
Distributed				
A3C	200	0	46K	50K
IMPALA	150	1	80K	
IMPALA (optimised)	375	1	200K	
IMPALA (optimised) batch 128	500	1	250K	

[1] Nvidia P100 [2] In frames/sec (4 times the agent steps due to action repeat). [3] Limited by amount of rendering possible on a single machine.

Performance and stability

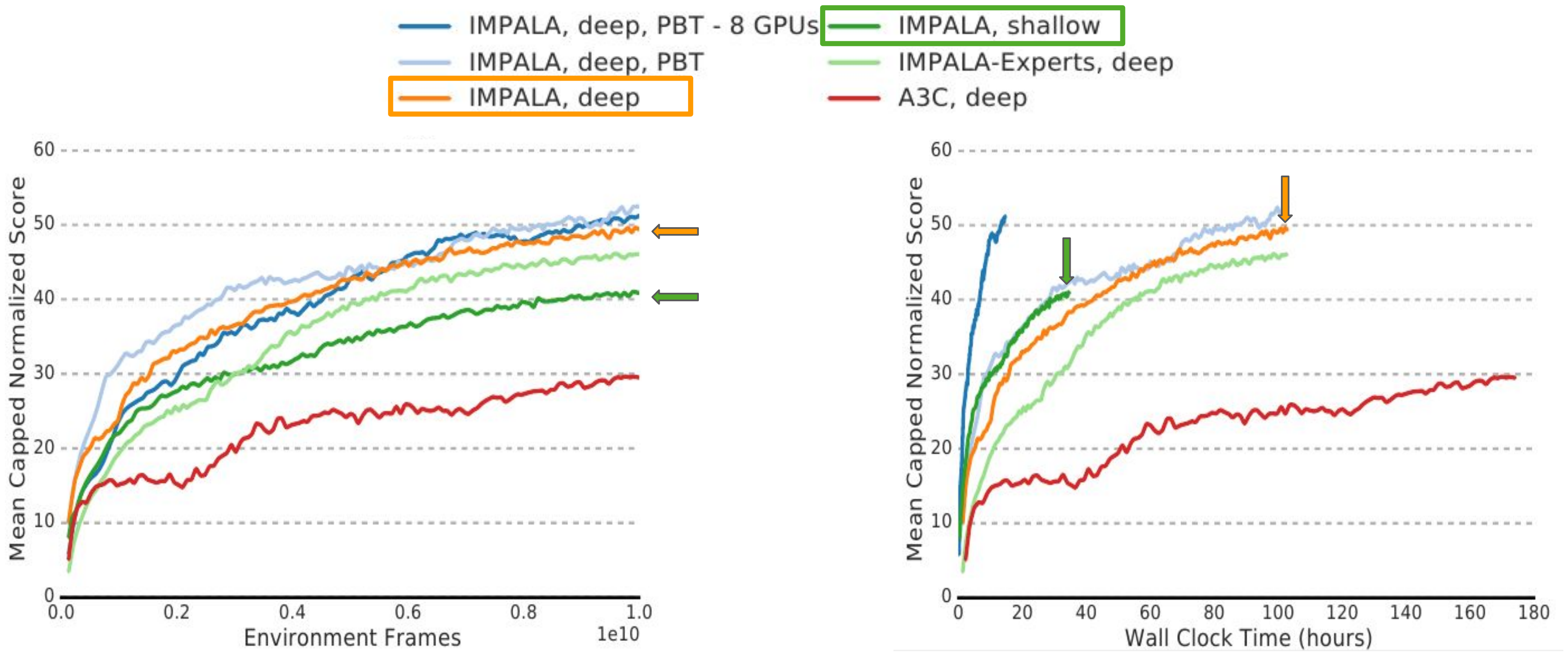


DMLab-30

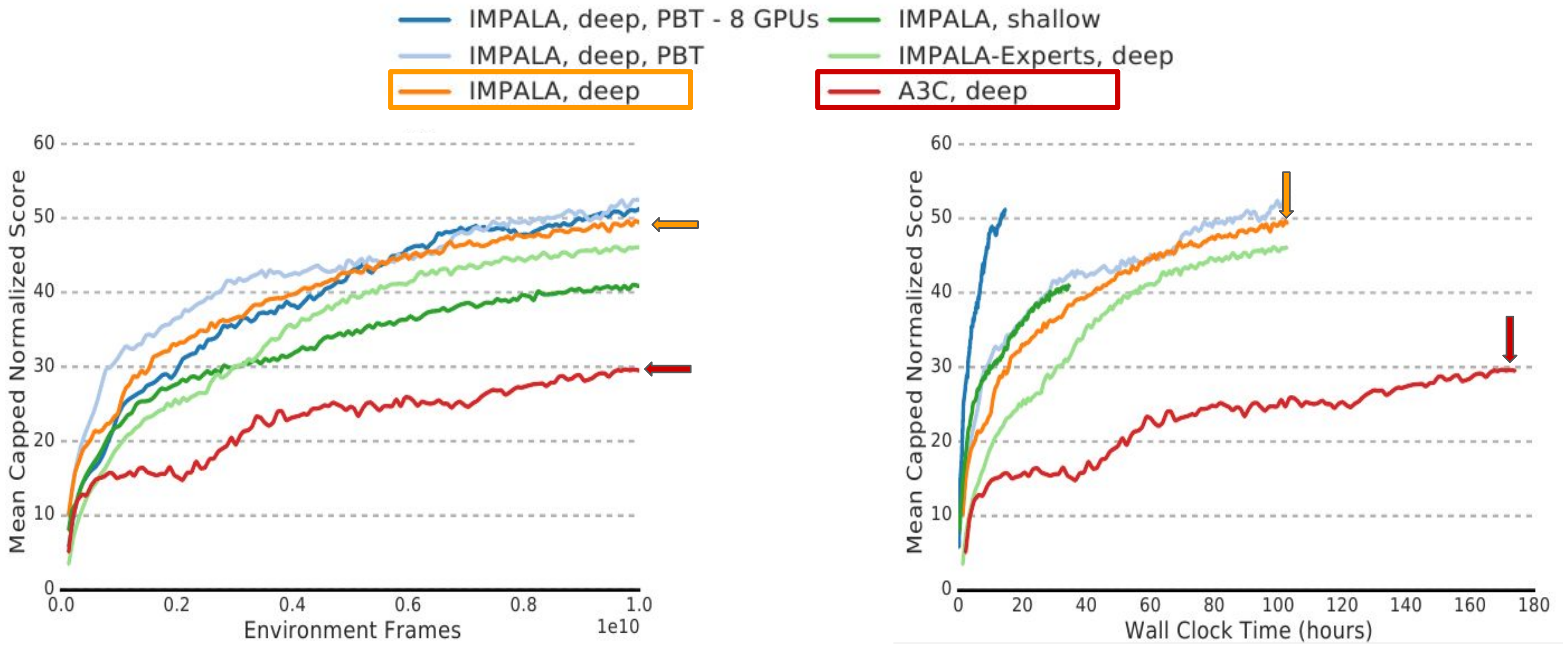


IMPALA, deep, PBT - 8 GPUs
IMPALA, deep, PBT
IMPALA, deep
IMPALA, shallow
IMPALA-Experts, deep
A3C, deep
Mean Capped Normalized Score
Environment Frames
1e10
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
50
60
Mean Capped Normalized Score
Wall Clock Time (hours)
0
20
40
60
80
100
120
140
160
180

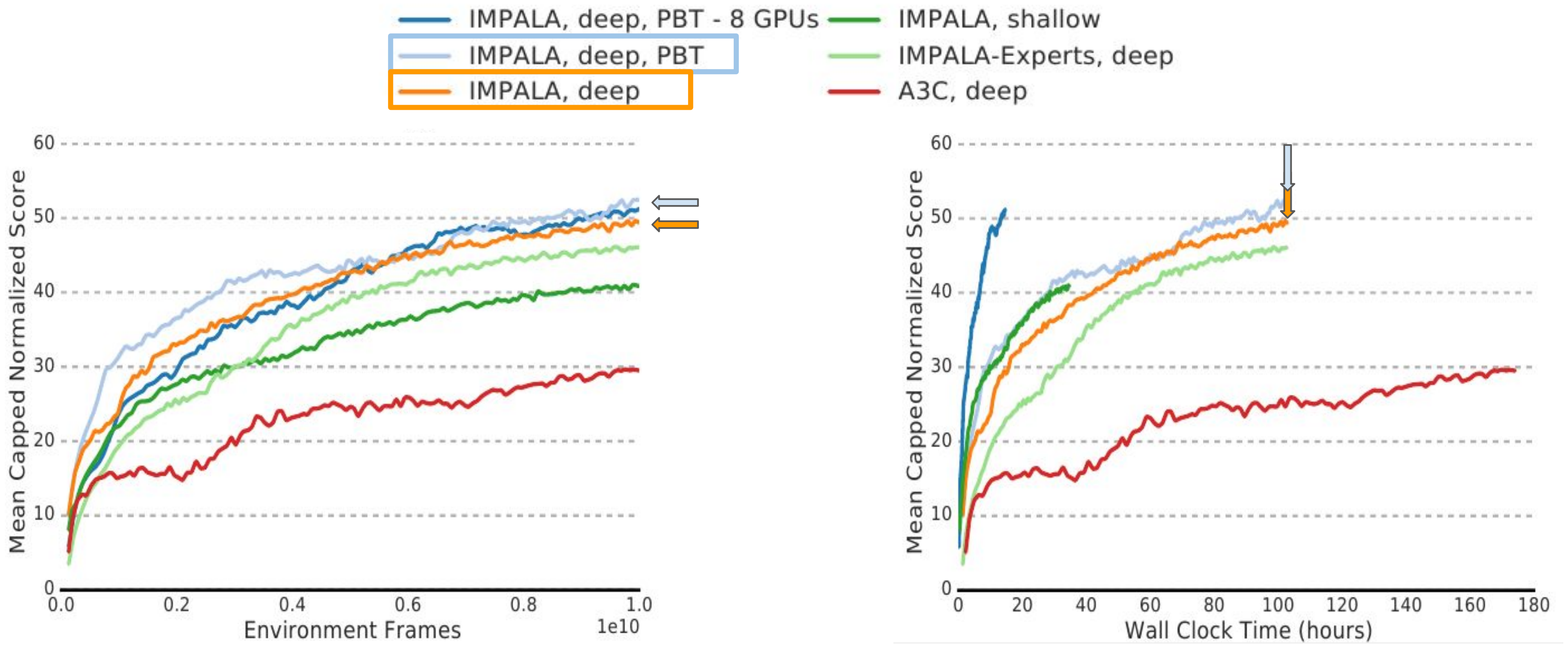
Shallow vs Deep networks



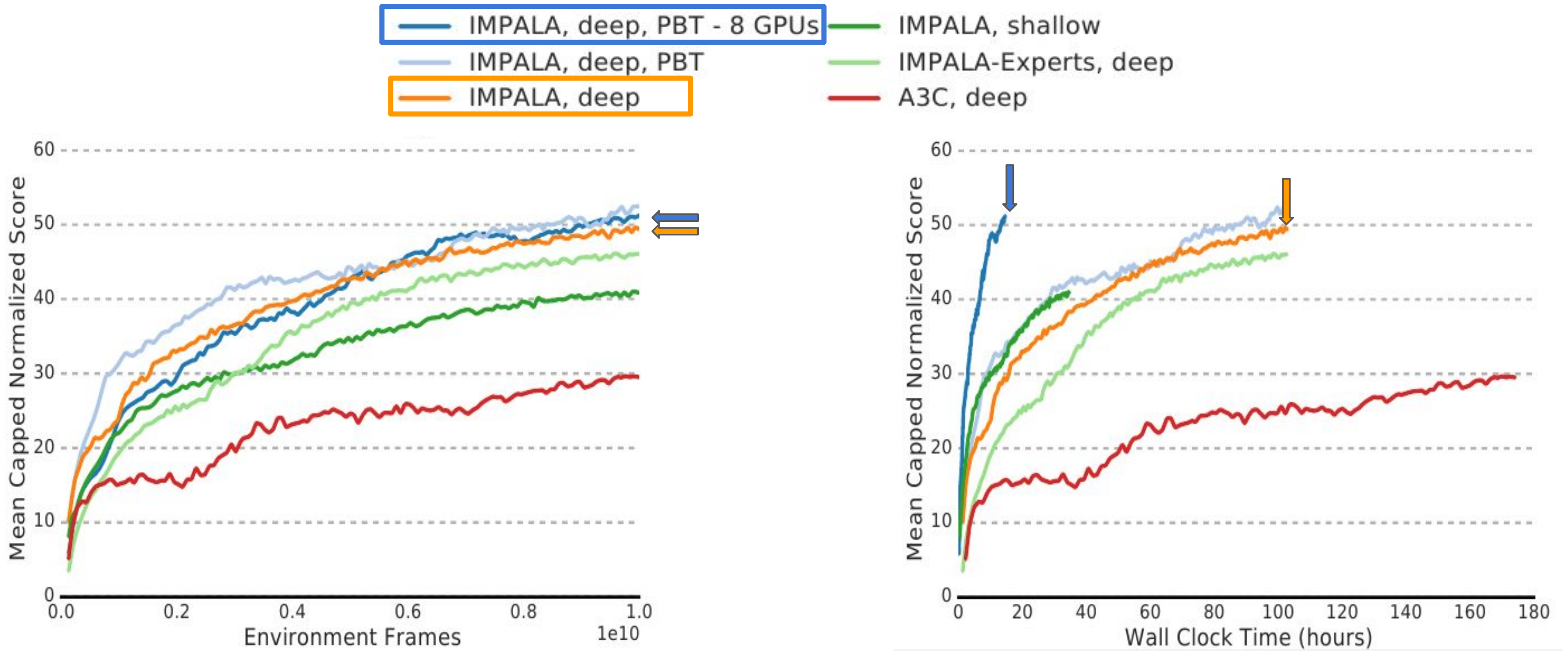
IMPALA vs A3C



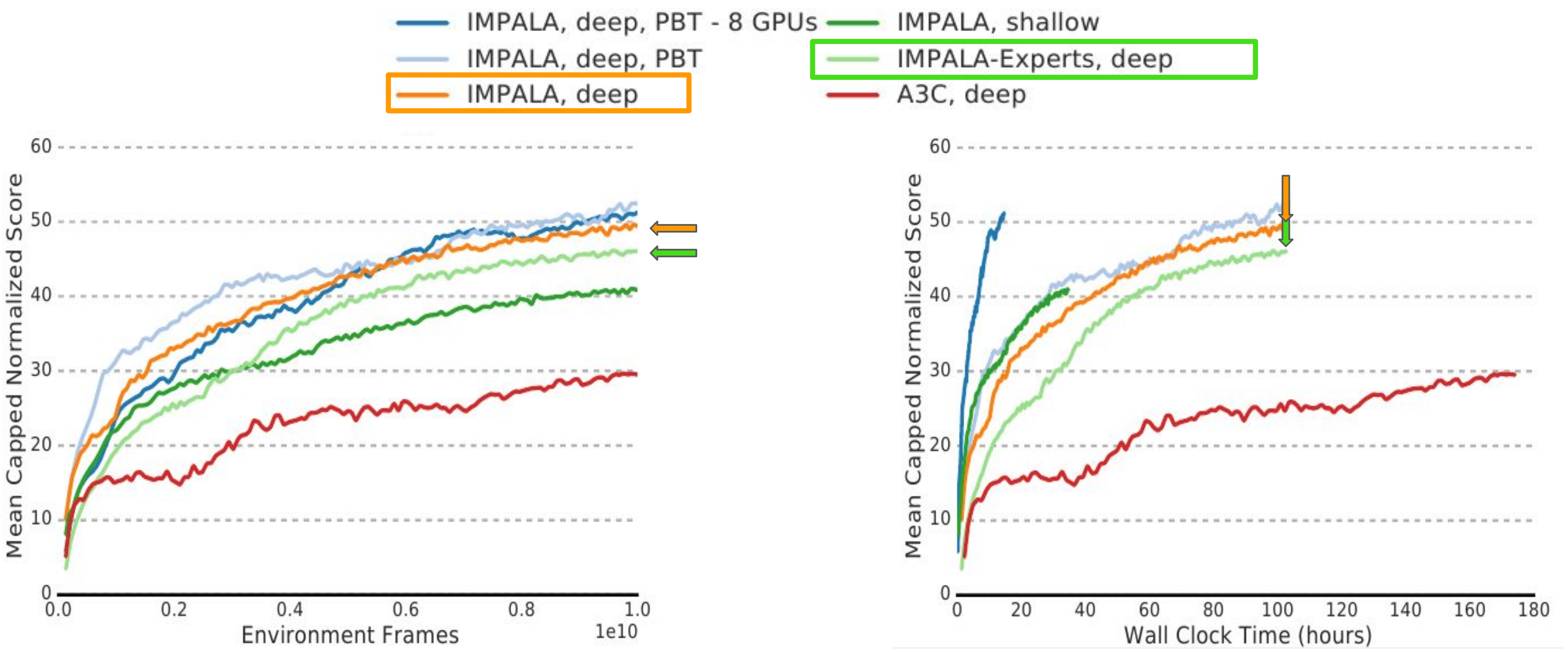
Integrating PBT



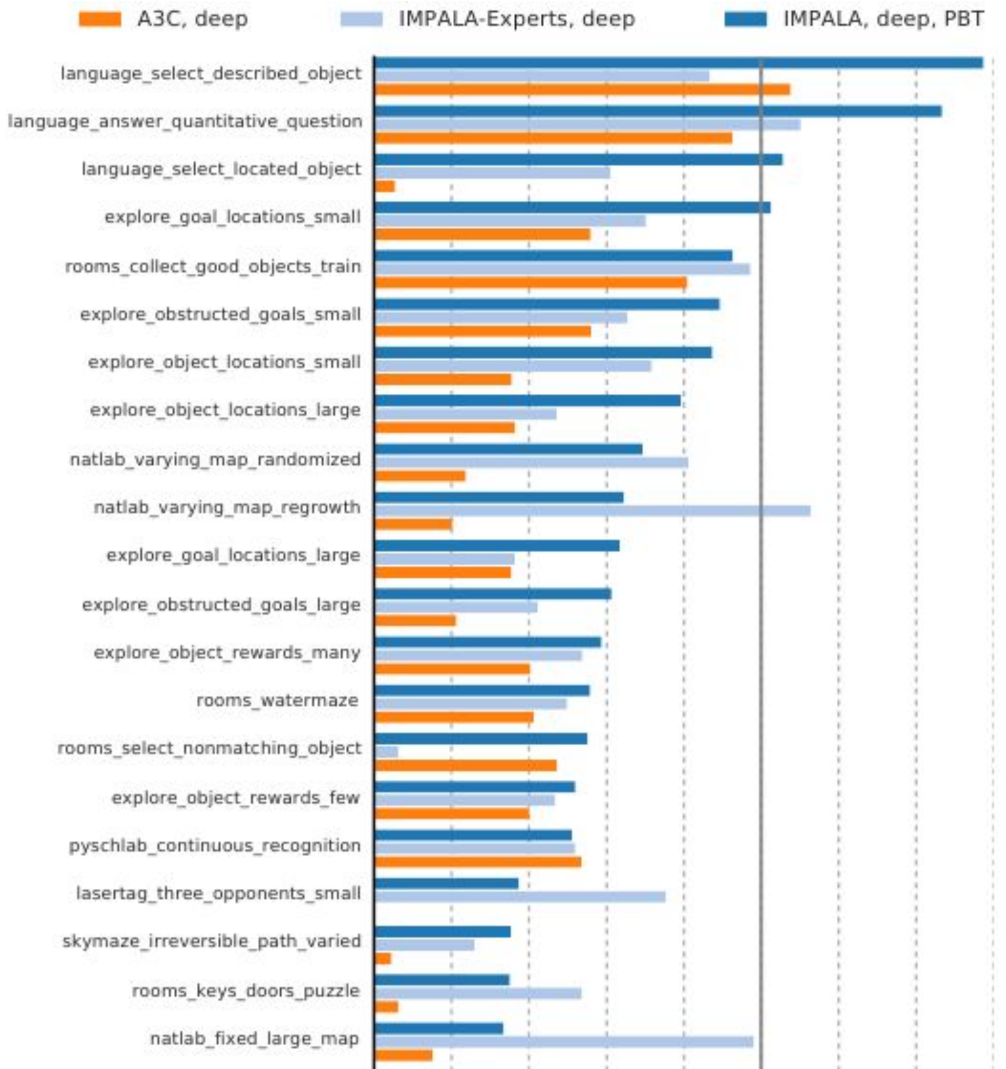
Scalability



Multi-agent vs experts

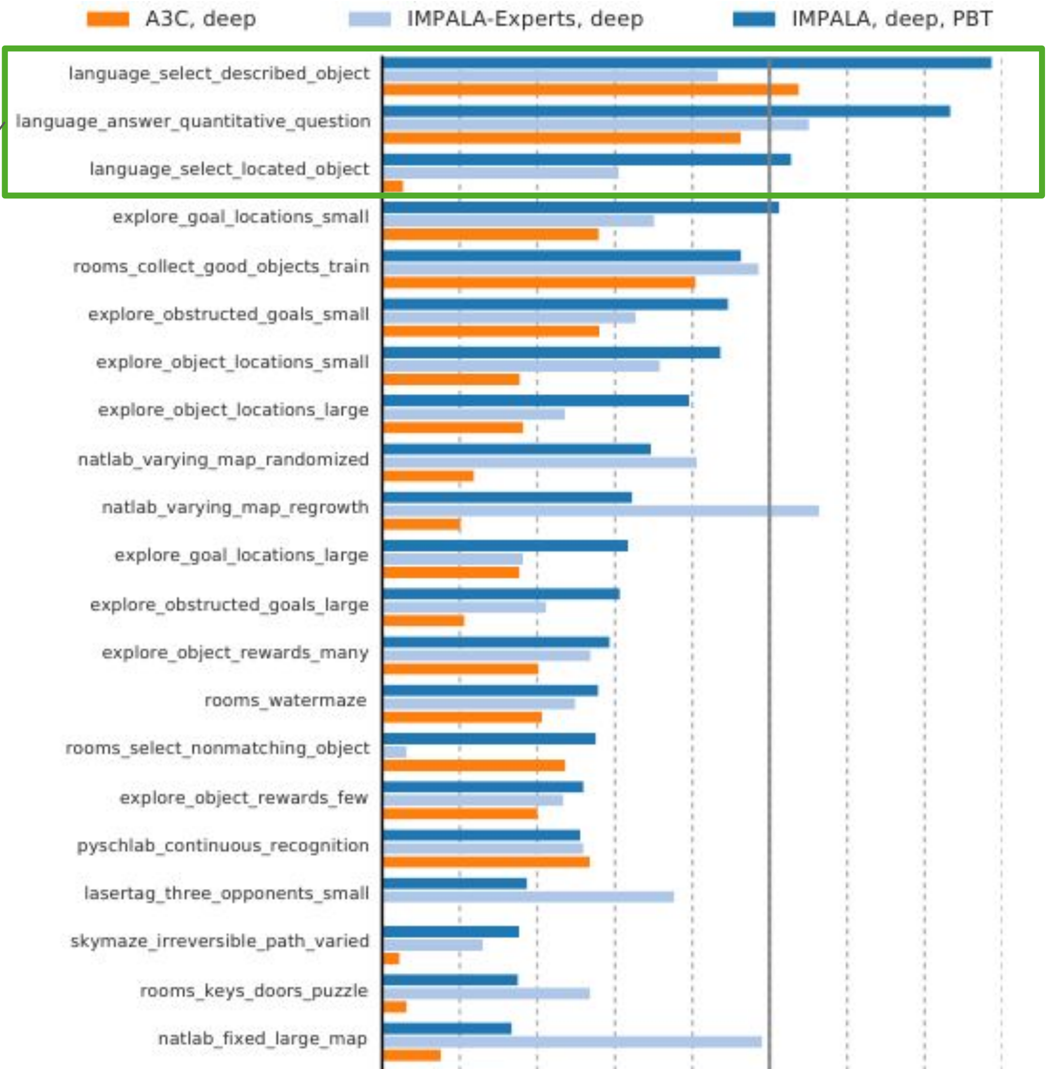


Level Breakdown



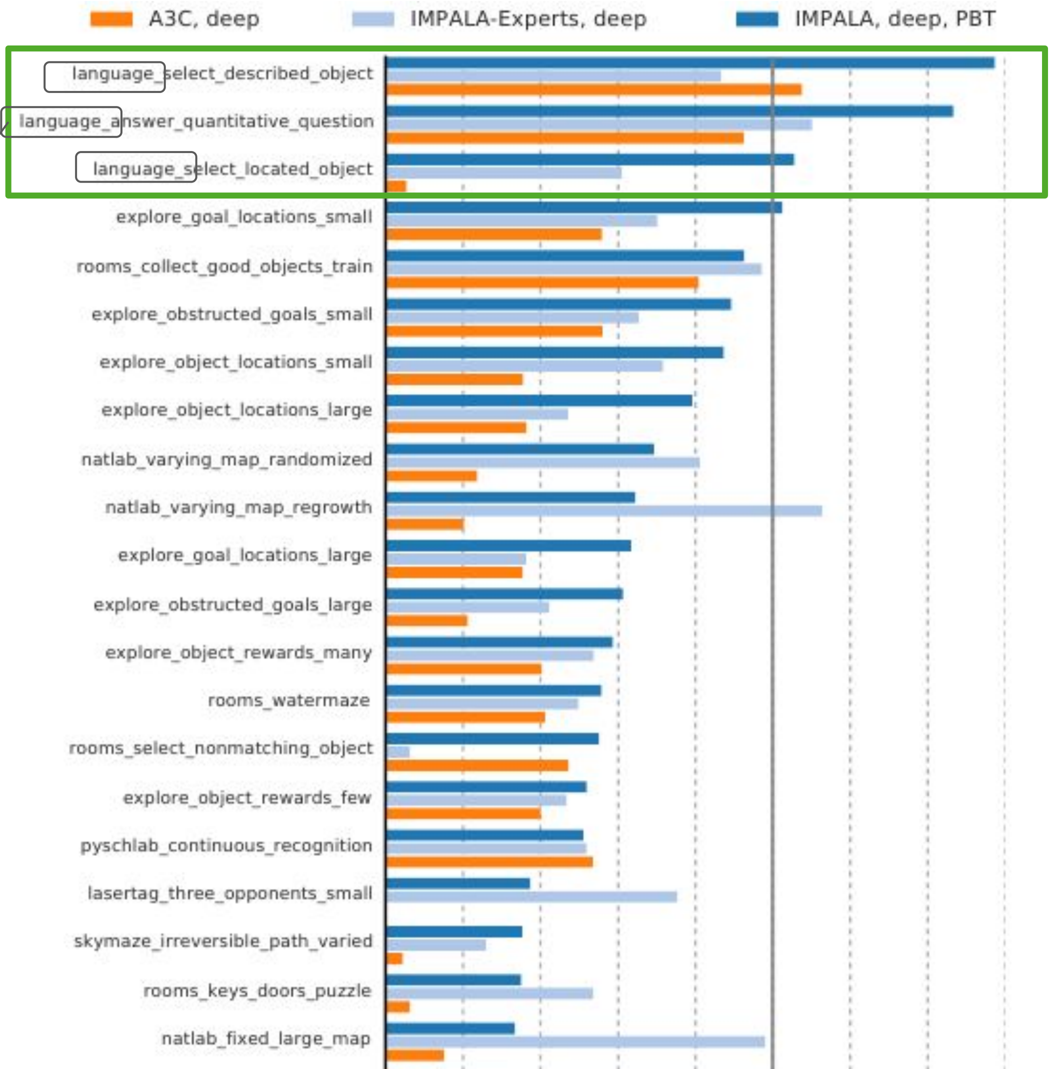
Level Breakdown

Positive transfer



Level Breakdown

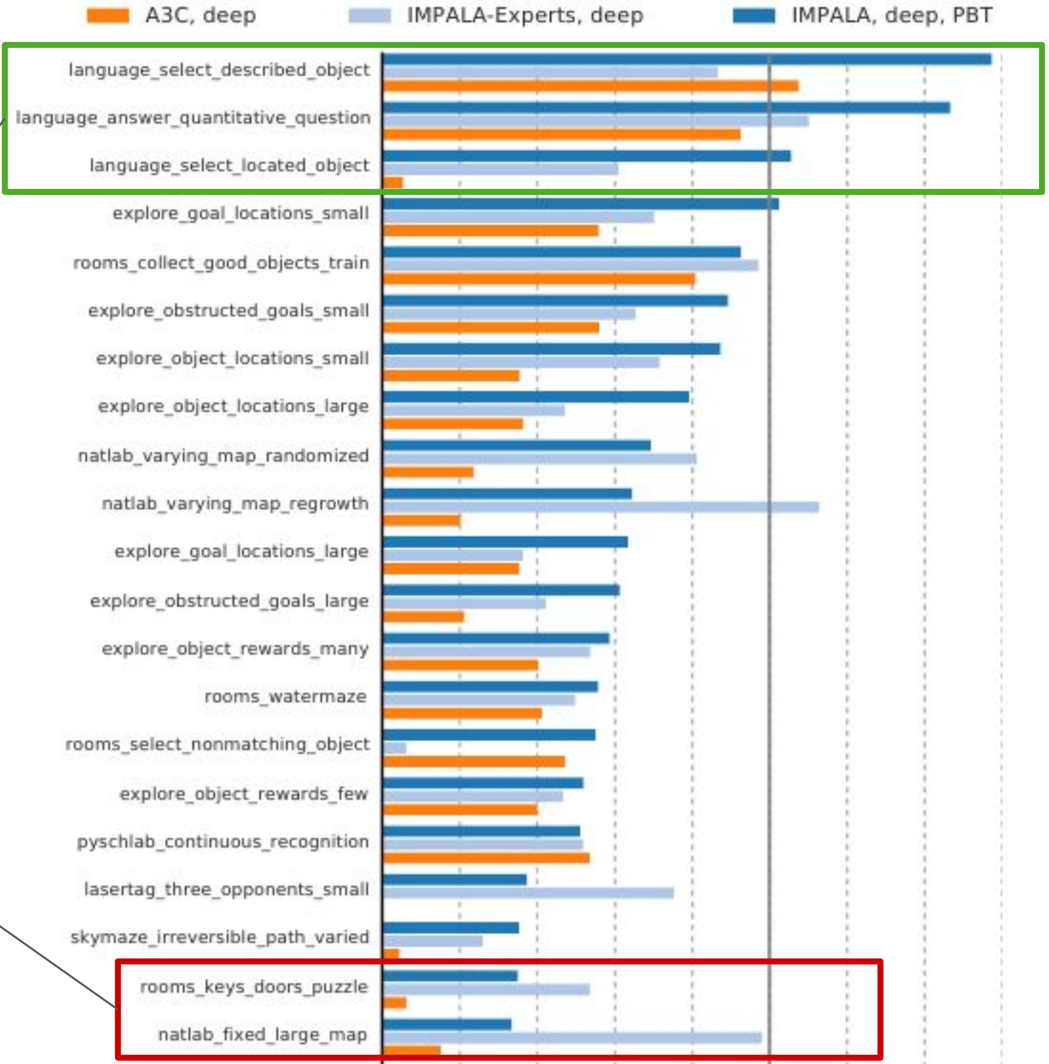
Positive transfer



Level Breakdown

Positive transfer

Not so much!



Takeaways

- **Efficient** and **scalable** Deep-RL agent
  - Efficient on single machine
  - Scales to 1000s of machines
- New off-policy correction (**V-trace**)
- New level-suite **DMLab-30**
- **Strong multi-task performance** with **some positive transfer**
- **Deeper networks perform better**
- **Effective across wide range of RL problems**

Resources

TensorFlow Implementation: https://github.com/deepmind/scalable_agent ★ Star 616

DeepMind Lab: https://github.com/deepmind/lab ★ Star 5,728

DeepMind blog: https://deepmind.com/blog/impala-scalable-distributed-deeprl-dmlab-30/

Paper: https://arxiv.org/abs/1802.01561

Lectures

- ICLR 2018 (Koray Kavukcuoglu): https://www.youtube.com/watch?v=N5oZIO8pE40
- ICML 2018 (Lasse Espeholt): https://www.facebook.com/icml.imls/videos/session-1-reinforcement-learning/432150780632776/
- Fields Institute 2018 (Volodymyr Mnih): https://www.fields.utoronto.ca/video-archive/static/2018/01/2509-18003/mergedvideo.ogv

Soft Actor Critic

Desired features for real world applications

Sample efficiency

No sensitive hyperparameters

Off-policy learning

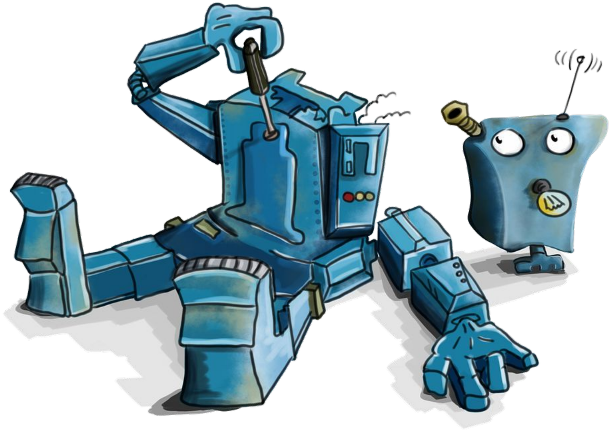
www.arl.army.mil

Desired features for real world experimentation

Asynchronous sampling

Stop/resume training

Action smoothing



Simultaneously maximizing reward and entropy (MaxEnt)

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} \left[r(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathcal{H}(\pi(\,\cdot\,|\mathbf{s}_t)) \right].$$

Simultaneously maximizing reward and entropy (MaxEnt)

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} \left[r(\mathbf{s}_t, \mathbf{a}_t) + \boxed{\alpha \mathcal{H}(\pi(\,\cdot\,|\mathbf{s}_t))} \right].$$

Simultaneously maximizing reward and entropy

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} \left[r(\mathbf{s}_t, \mathbf{a}_t) + \boxed{\alpha \mathcal{H}(\pi(\cdot|\mathbf{s}_t))} \right].$$

How to optimize it?

Optimizing the MaxEnt Objective

Soft Q-learning?

Optimizing the MaxEnt Objective

Soft Q-learning methods?

- Intractable in continuous domains
- Continuous solutions rely on biased approximations

Optimizing the MaxEnt Objective

Soft Q-learning (SQL)?

- Intractable in continuous domains
- Continuous solutions rely on biased approximations

Proposed solution: Soft actor-critic (SAC)

Optimizing the MaxEnt Objective

Soft Q-learning (SQL)?

- Intractable in continuous domains
- Continuous solutions rely on biased approximations

Proposed solution: Soft actor-critic (SAC)

- Learns the soft Q-function of policy and the policy jointly.
- Similar to DDPG, but with a stochastic policy
- Easy to implement, sample efficient, and stable

Soft Policy Iteration

1. Soft policy evaluation: Fix policy, apply soft Bellman backup until converges:

$$\mathcal{T}^\pi Q(\mathbf{s}_t, \mathbf{a}_t) \triangleq r(\mathbf{s}_t, \mathbf{a}_t) + \gamma\, \mathbb{E}_{\mathbf{s}_{t+1} \sim p} \left[V(\mathbf{s}_{t+1}) \right],$$

$$V(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi} \left[Q(\mathbf{s}_t, \mathbf{a}_t) - \boxed{\log \pi(\mathbf{a}_t | \mathbf{s}_t)]}\right.$$

This converges to Q^π.

2. Soft policy improvement: Update the policy through information projection

$$\pi_{\text{new}} = \arg\min_{\boxed{\pi' \in \Pi}} \mathrm{D}_{\mathrm{KL}} \left(\pi'(\,\cdot\,|\mathbf{s}_t) \,\middle\|\, \frac{\exp\left(Q^{\pi_{\text{old}}}(\mathbf{s}_t, \,\cdot\,)\right)}{Z^{\pi_{\text{old}}}(\mathbf{s}_t)} \right)$$

From the new policy, we have $Q^{\pi_{\text{new}}}(\mathbf{s}_t, \mathbf{a}_t) \geq Q^{\pi_{\text{old}}}(\mathbf{s}_t, \mathbf{a}_t)$

Soft Policy Iteration to Soft Actor-Critic

Use function approximators

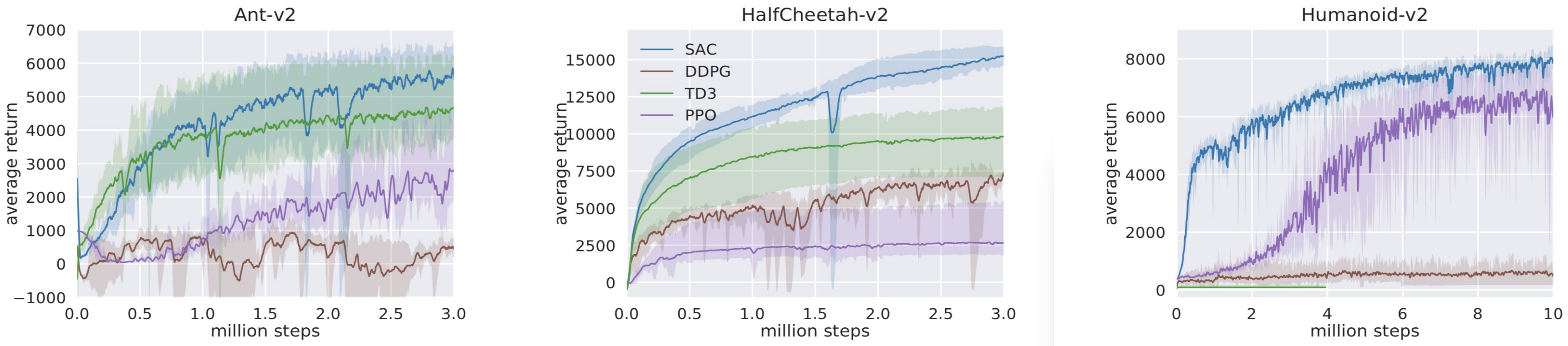
Alternate optimization between Q-function (V parameterized by ψ) and policy network (π parameterized by ϕ) with SGD

Additional network - Soft Q-function (Q parameterized by θ)

Soft Actor Critic

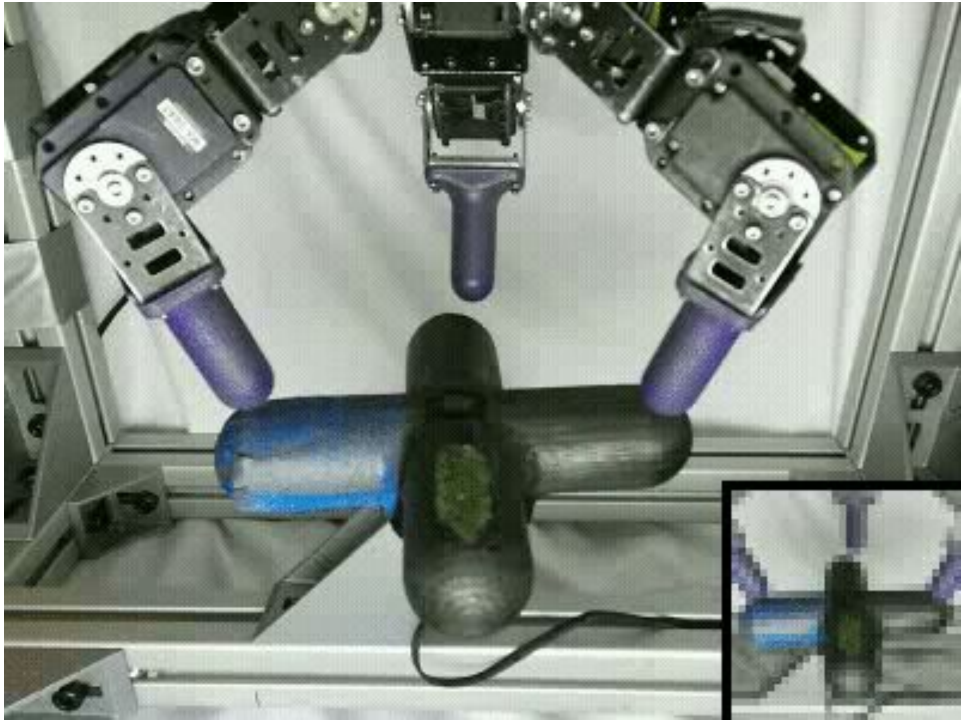
Initialize parameter vectors ψ, $\bar{\psi}$, θ, ϕ.
for each iteration **do**
  for each environment step **do**
    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$
    $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$
  end for
  for each gradient step **do**
    $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
    $\bar{\psi} \leftarrow \tau\psi + (1 - \tau)\bar{\psi}$
  end for
end for

Simulated benchmarks



Real World experiments

Real World experiments



Resources

Implementations

- https://github.com/rail-berkeley/softlearning (by authors) ★ Star 261
- https://github.com/vitchyr/rlkit ★ Star 690
- https://github.com/openai/spinningup ★ Star 2,707
- https://github.com/higgsfield/RL-Adventure-2 ★ Star 1,663

Blogs/Tutorials

- https://spinningup.openai.com/en/latest/algorithms/sac.html
- https://bair.berkeley.edu/blog/2018/12/14/sac/

Talk - Tuomas Haarnoja (NIPS 2017) https://vimeo.com/252185258

Questions?