Deep Reinforcement Learning that Matters

Sungkwon On
13 – Mar – 2023

About the Paper

Deep Reinforcement Learning that Matters

Peter Henderson[1]*, Riashat Islam[1,2]*, Philip Bachman[2]
Joelle Pineau[1], Doina Precup[1], David Meger[1]
[1] McGill University, Montreal, Canada
[2] Microsoft Maluuba, Montreal, Canada
{peter.henderson, riashat.islam}@mail.mcgill.ca, phbachma@microsoft.com
{jpineau, dprecup}@cs.mcgill.ca, dmeger@cim.mcgill.ca

Motivation

Reproducing existing work and accurately judging the improvements offered by novel methods is vital.

However, reproducing the results is seldom straightforward. Reproductivity can be affected by:

1. Extrinsic factors:
    - Hyperparameters
    - Codebases
2. Intrinsic factors:
    - Random seeds
    - Environmental properties

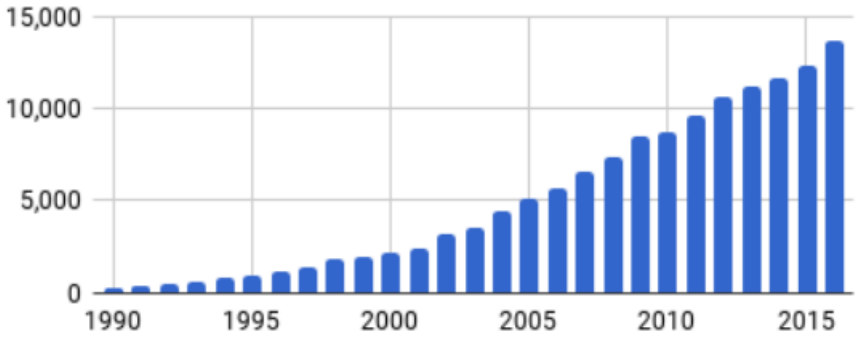


Figure 1: Growth of published reinforcement learning papers. Shown are the number of RL-related publications (y-axis) per year (x-axis) scraped from Google Scholar searches.

Experiment Protocol

Algorithms:

- PPO, TRPO, DDPG, ACKTR

Environments:

- Hopper-v1 & HalfCheetah-v1 MuJoCo from OpenAI Gym

Metrics:

- Hyperparameters:
  - Network Architecture
  - Reward Scale
  - Random Seeds and Trials
- Environments
- Codebases

Hyperparameters

What is the magnitude of the effect hyperparameter settings can have on baseline performance?

In the reported literature, there is no consistency in:

- Network Architecture
- Activation function
- Reward Scaling

Table 4: Evaluation Hyperparameters of baseline algorithms reported in related literature

Related Work (Algorithm)	Policy Network	Policy Network Activation	Value Network	Value Network Activation	Reward Scaling	Batch Size
DDPG	64x64	ReLU	64x64	ReLU	1.0	128
TRPO	64x64	TanH	64x64	TanH	-	5k
PPO	64x64	TanH	64x64	TanH	-	2048
ACKTR	64x64	TanH	64x64	ELU	-	2500
Q-Prop (DDPG)	100x50x25	TanH	100x100	ReLU	0.1	64
Q-Prop (TRPO)	100x50x25	TanH	100x100	ReLU	-	5k
IPG (TRPO)	100x50x25	TanH	100x100	ReLU	-	10k
Param Noise (DDPG)	64x64	ReLU	64x64	ReLU	-	128
Param Noise (TRPO)	64x64	TanH	64x64	TanH	-	5k
Benchmarking (DDPG)	400x300	ReLU	400x300	ReLU	0.1	64
Benchmarking (TRPO)	100x50x25	TanH	100x50x25	TanH	-	25k

Network Architecture

Comparison of:
MLP Network Architecture: (64, 64), (100, 50, 25), (400, 300)
Policy&Critic Network Activation: TanH, ReLU, Leaky ReLU

Results:
ReLU & Leaky ReLU works well overall
Network structure of (400, 300) shows poor performance → Requires tweaking other hyperparameters

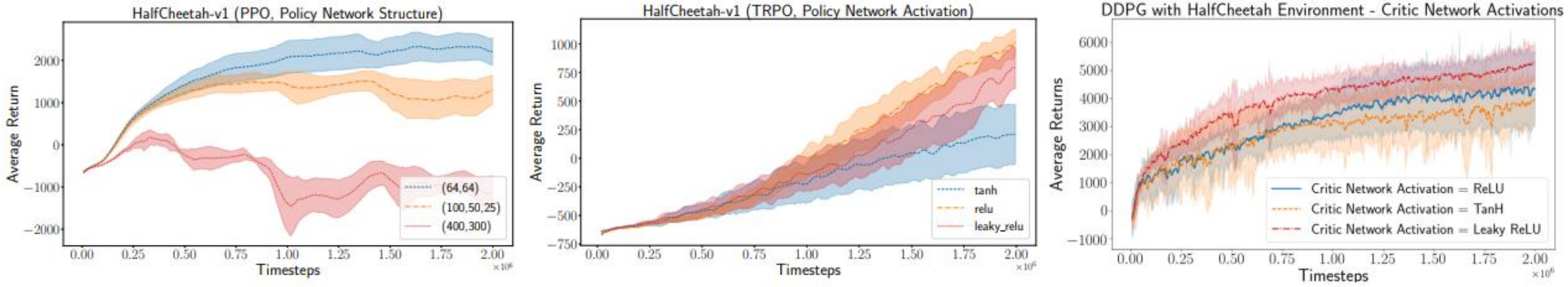


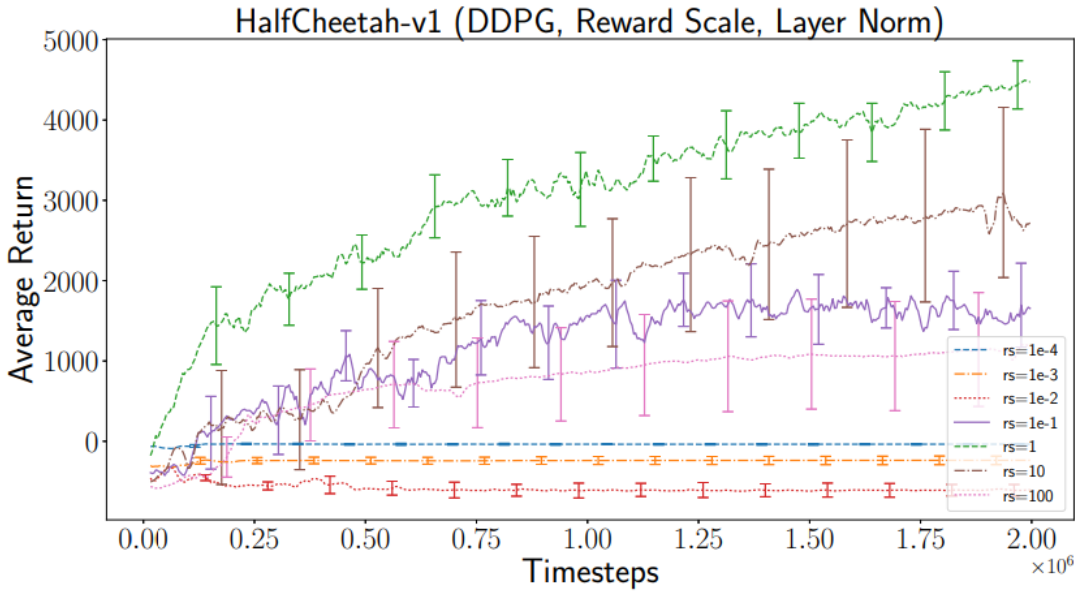
Figure 2: Significance of Policy Network Structure and Activation Functions PPO (left), TRPO (middle) and DDPG (right).

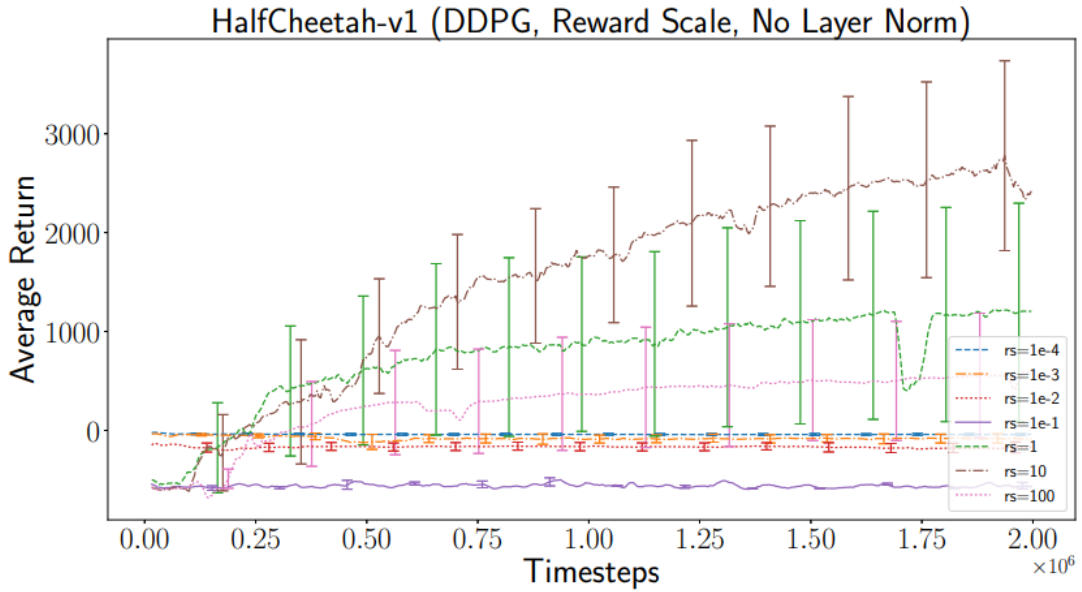
Reward Scale

Results vary by different scales and having normalized layers.

→ Authors infer the reason to be the use of deep networks and gradient based methods: value function may be tracking a moving target.

The Reward Scale has a potential to have a large impact!!





Random Seeds and Trials

- Performance of the exact same training differs by different seeds.
- Two different training curves trained and averaged on different 5 seeds show significant differences.
- Some recent reported results only report top-N trials among many or average over only small number of trials.
- No specific number of trials specified as a recommendation.

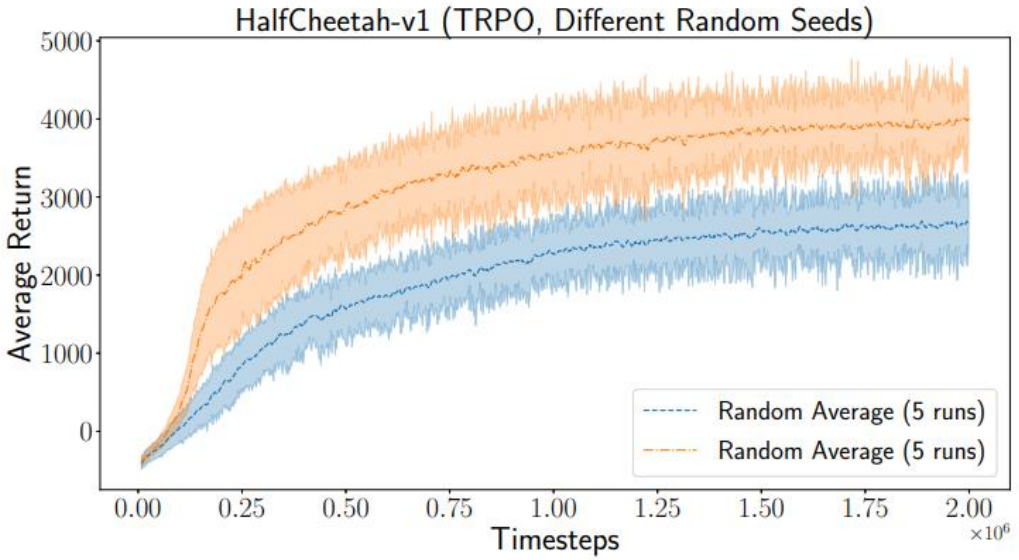
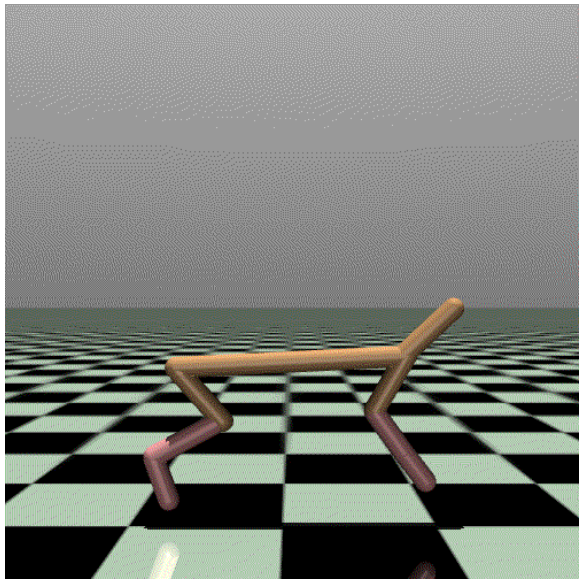


Figure 5: TRPO on HalfCheetah-v1 using the same hyperparameter configurations averaged over two sets of 5 different random seeds each. The average 2-sample t-test across entire training distribution resulted in $t = -9.0916$, $p = 0.0016$.

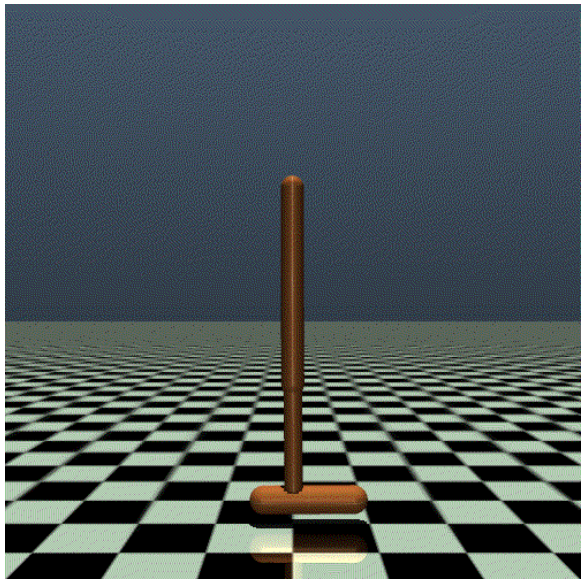
Environments

Half Cheetah



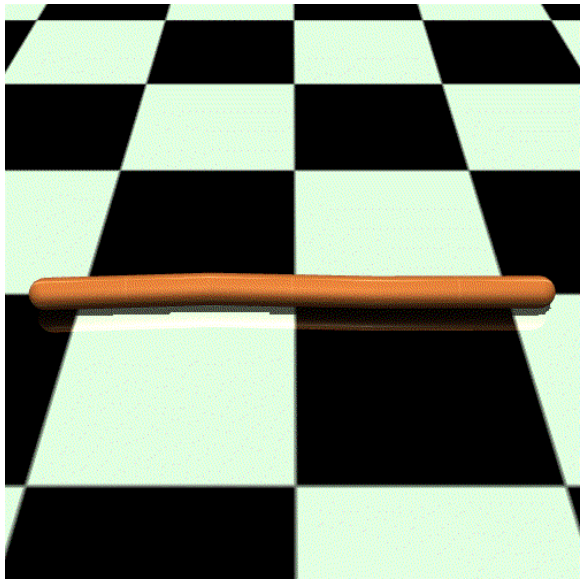
Stable Dynamics

Hopper



Unstable Dynamics

Swimmer



Easy to fall into local optimum

Environments

DDPG, an off-policy algorithm, performs well on envs with stable dynamics but not on envs with unstable dynamics.

TRPO seems to perform the best on Swimmer, but actually falls into local optimum (due to environmental factor) → environments are required to have a proper reward settings

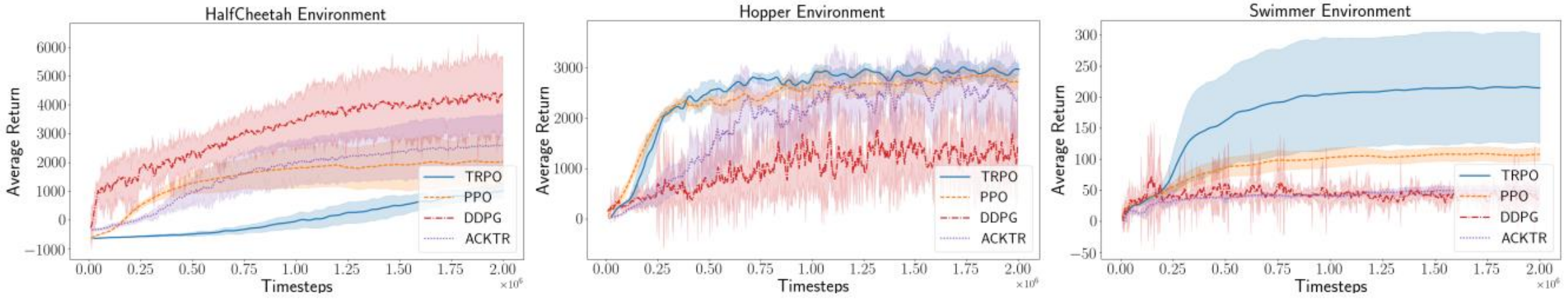


Figure 4: Performance of several policy gradient algorithms across benchmark MuJoCo environment suites

Codebases

Investigation of TRPO and DDPG on different codebases:

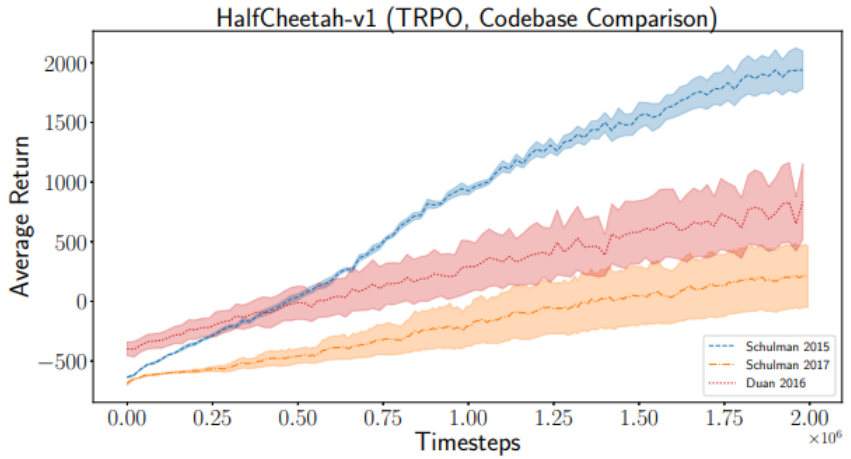
TRPO:

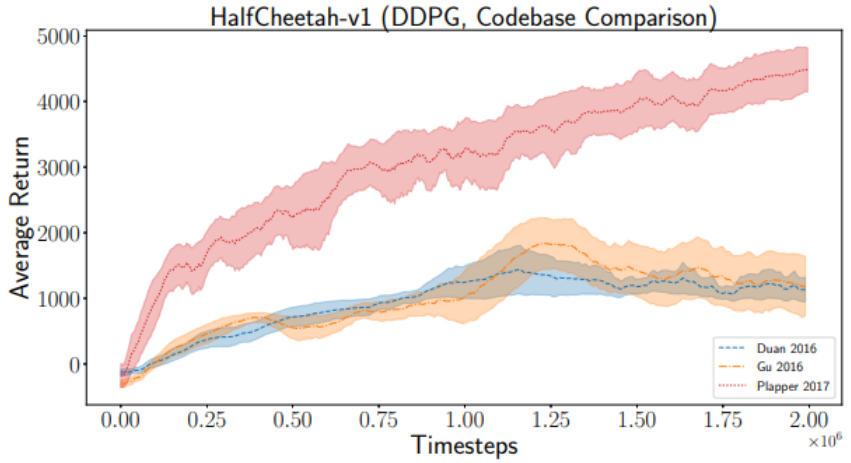
- OpenAI baselines (Schulman et al. 2017)
- Original (Schulman et al. 2015a)
- Rllab – Tensorflow (Duan et al. 2016)

DDPG:

- OpenAI baselines (Plappert et al. 2017)
- Rllab – Theano (Duan et al. 2016)
- Rllabplusplus (Gu et al. 2016)

The performances have high variances.
It is vital for the authors to publish with codebases and implementation details.





Reporting Evaluation Metrics

As in Supervised Learning, RL is in dire need of significance metrics to compare results.
Such as:

- K-fold, t-test
- Corrected resampled t-tests
- Kolmogorov-Smirnov test

-	DDPG	ACKTR	TRPO	PPO
DDPG	-	$t = -1.41, p = 0.196$ $KS = 0.60, p = 0.209$ -35.92 % (-85.62 %, -5.38 %)	$t = -2.58, p = 0.033$ $KS = 0.80, p = 0.036$ -44.96 % (-78.82 %, -20.29 %)	$t = -2.09, p = 0.070$ $KS = 0.80, p = 0.036$ -39.90 % (-77.12 %, -12.95 %)
ACKTR	$t = 1.41, p = 0.196$ $KS = 0.60, p = 0.209$ 56.05 % (-87.98 %, 123.15 %)	-	$t = -1.05, p = 0.326$ $KS = 0.60, p = 0.209$ -14.11 % (-37.17 %, 9.11 %)	$t = -0.42, p = 0.686$ $KS = 0.40, p = 0.697$ -6.22 % (-31.58 %, 18.98 %)
TRPO	$t = 2.58, p = 0.033$ $KS = 0.80, p = 0.036$ 81.68 % (-67.76 %, 151.64 %)	$t = 1.05, p = 0.326$ $KS = 0.60, p = 0.209$ 16.43 % (-27.92 %, 41.17 %)	-	$t = 2.57, p = 0.033$ $KS = 0.60, p = 0.209$ 9.19 % (2.37 %, 15.58 %)
PPO	$t = 2.09, p = 0.070$ $KS = 0.80, p = 0.036$ 66.39 % (-67.80 %, 130.16 %)	$t = 0.42, p = 0.686$ $KS = 0.40, p = 0.697$ 6.63 % (-33.54 %, 29.59 %)	$t = -2.57, p = 0.033$ $KS = 0.60, p = 0.209$ -8.42 % (-14.08 %, -2.97 %)	-

Table 10: Hopper Significance values and metrics for different algorithms. Rows in cells are: sorted 2-sample t-test, Kolmogorov-Smirnov test, bootstrap A/B comparison % difference with 95% confidence bounds.

Questions?