Reward Shaping, Advanced Exploration, and Entropy Regularization

Alpha Zero: Actor Critic with MCTS

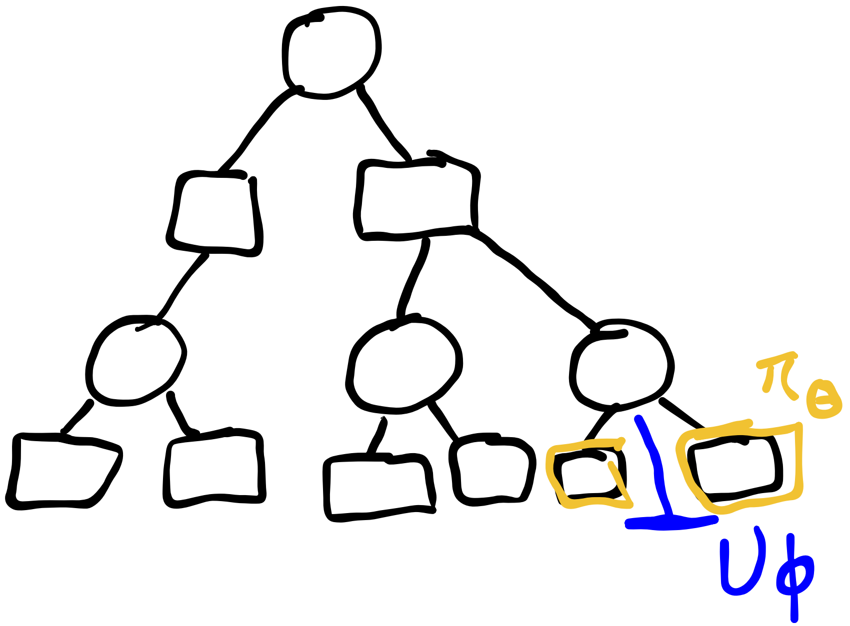
1. Use π_θ and U_ϕ in MCTS
2. Learn π_θ and U_ϕ from tree

$$\ell(\theta) = -\mathbb{E}_s\left[\sum_a \pi_{\mathrm{MCTS}}(a \mid s) \log \pi_\theta(a \mid s)\right]$$

$$\pi_{\mathrm{MCTS}}(a \mid s) \propto N(s,a)^\eta$$

$$\ell(\phi) = \frac{1}{2}\mathbb{E}_s\left[\left(U_\phi(s) - U_{\mathrm{MCTS}}(s)\right)^2\right]$$

$$U_{\mathrm{MCTS}}(s) = \max_a Q(s,a)$$



$$a = \arg\max_a Q(s,a) + c\pi_\theta(a \mid s)\frac{\sqrt{N(s)}}{1+N(s,a)}$$

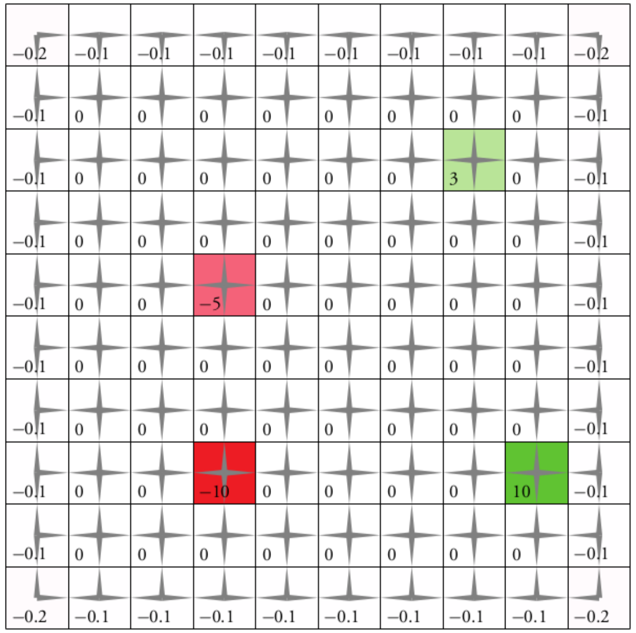
https://www.youtube.com/embed/tlOIHko8ySg?enablejsapi=1

https://www.youtube.com/watch?v=tlOIHko8ySg

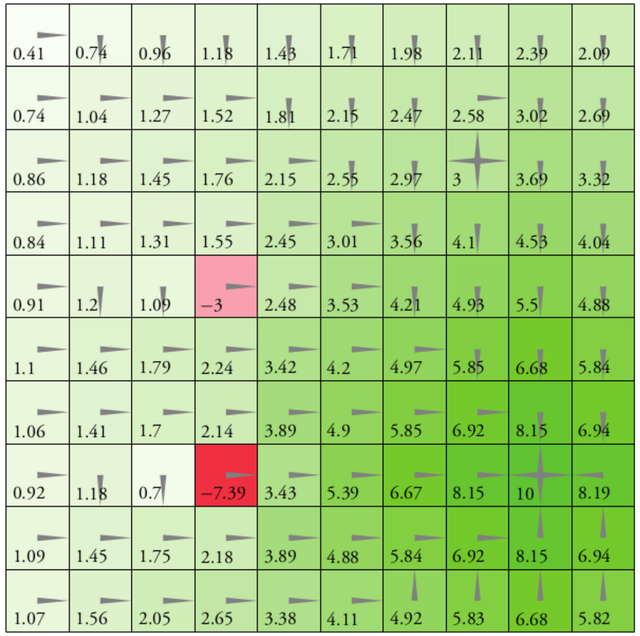
Reward Shaping

"As a general rule, it is better to design performance measures according to what one actually wants in the environment, rather than according to how one thinks the agent should behave." - Stuart Russell

Reward



Value



Reward Shaping

- $R(s, a, s')+ = F(s) - \gamma F(s')$
- any other transformation may yield sub optimal policies unless further assumptions are made about the underlying MDP

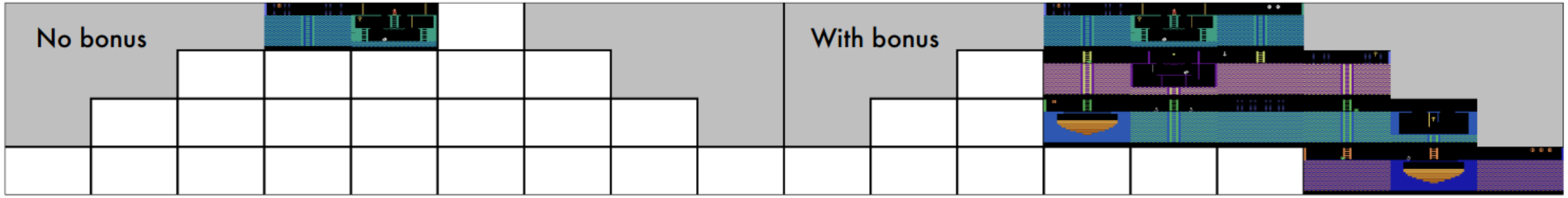
Is Exploration Important?
Montezuma's Revenge

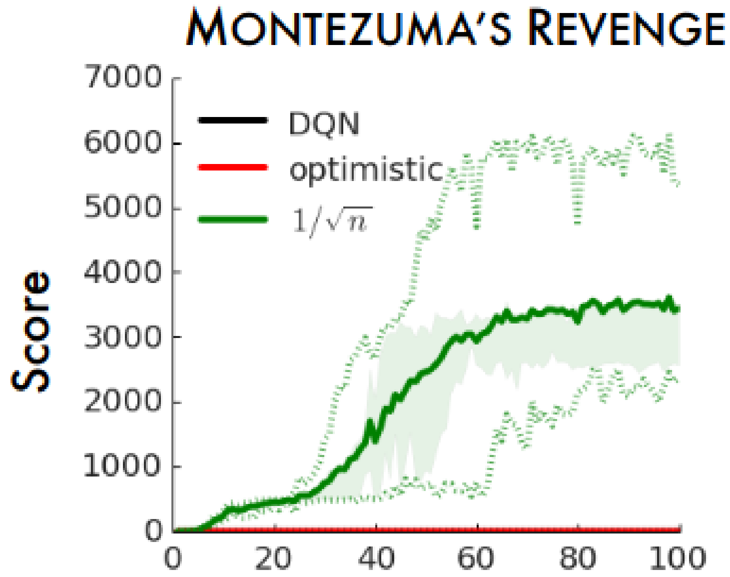
Is Exploration Important?
Theory

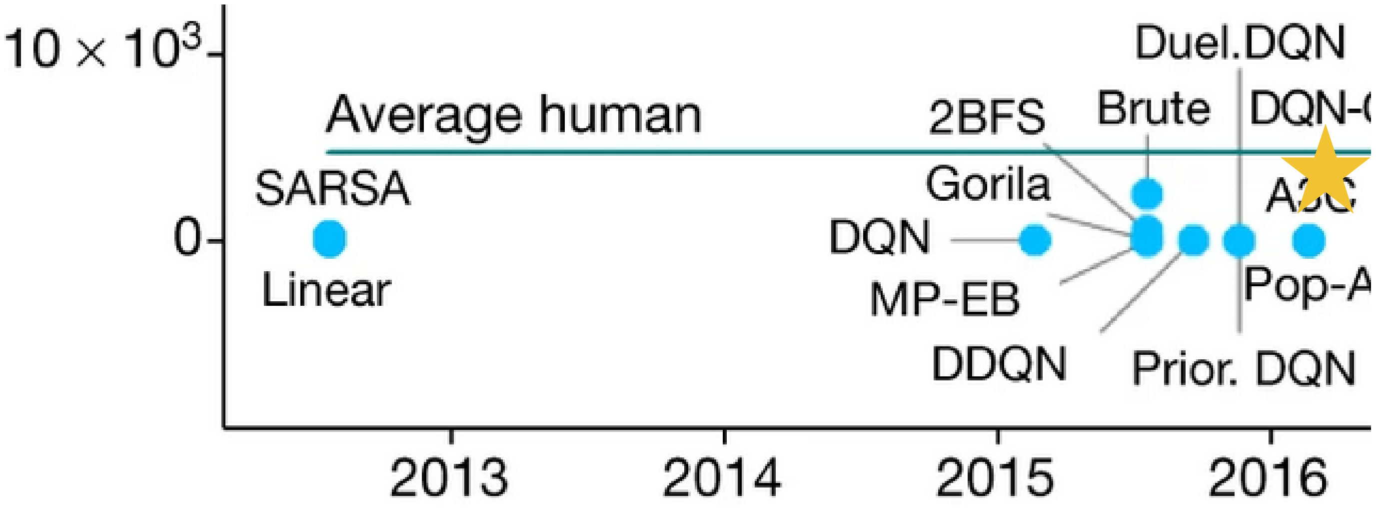
Exploration Bonus

Example 1: Learn Pseudocount

$B(s,a) \approx \frac{1}{\sqrt{\hat{N}(s)}}$ where $\hat{N}(s)$ is a learned function approximation





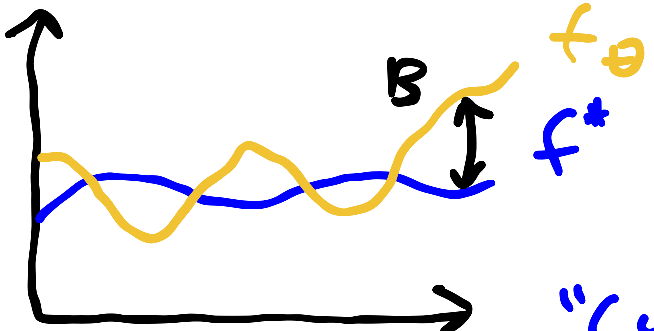


Bellemare, et al. 2016 "Unifying Count-Based Exploration..."

Exploration Bonus

Example 2: Learn a function of the state and action

$$B(s,a) = \|\hat{f}_\theta(s,a) - f^*(s,a)\|^2$$

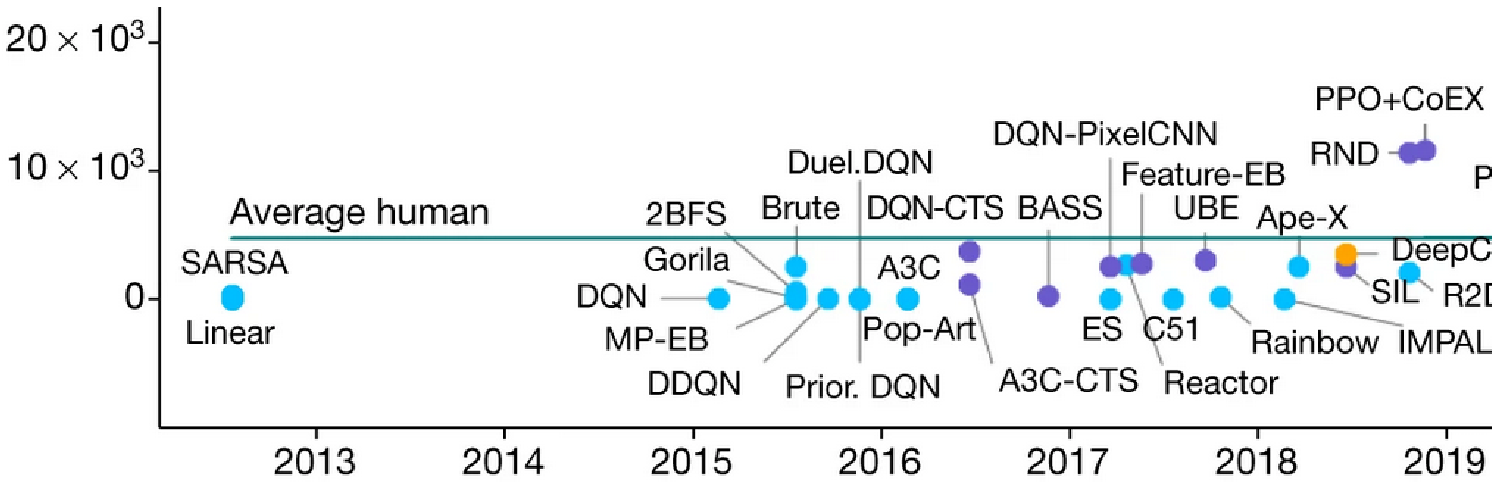


What should f^* be?

- $f^*(s,a) = s'$ (Next state prediction)
- $f^*(s,a) = f_\phi(s,a)$ where f_ϕ is a random neural network.

"Curiosity"

"Random Network Distillation"



Exploration Bonus

Example 3: Thompson Sampling

1. Maintain a distribution over Q ← Hard
2. Sample Q
3. Act according to Q

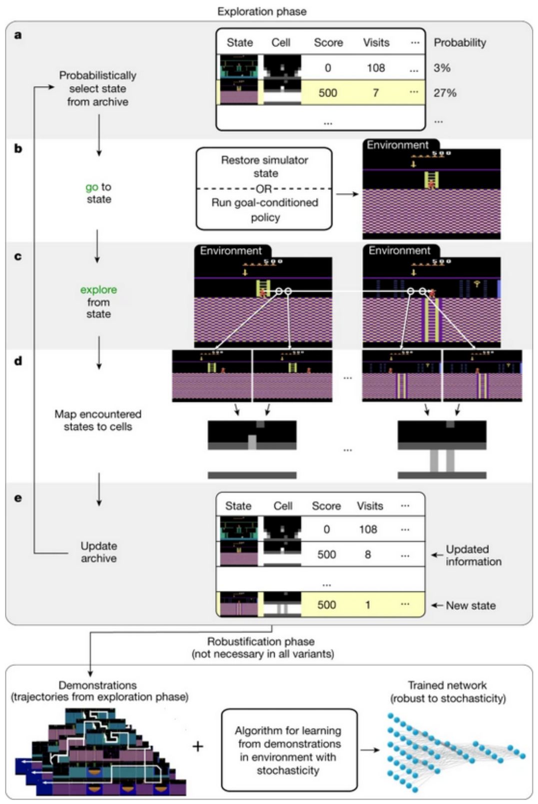
- Bootstrapping with multiple Q networks
- Dropout

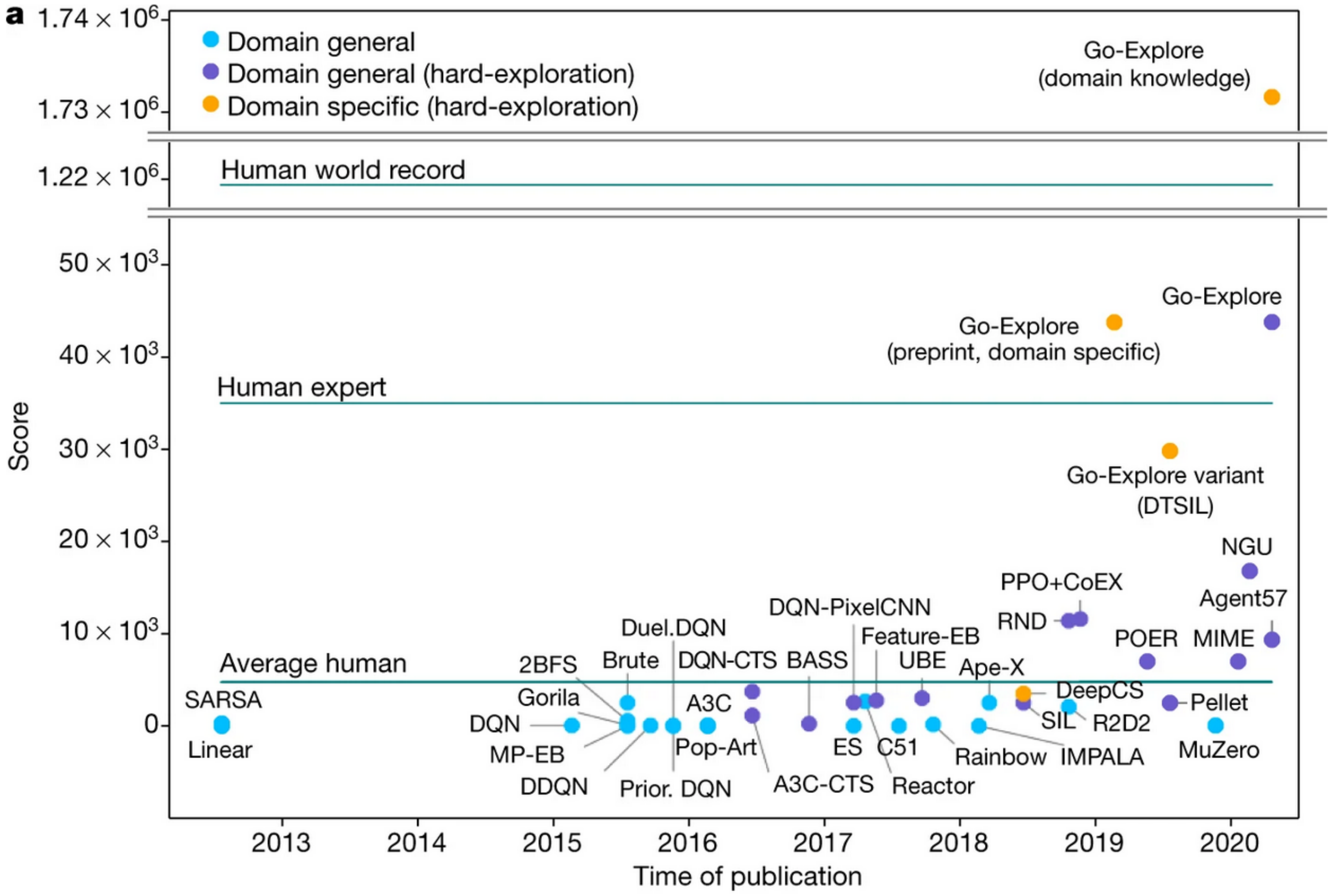
Exploration Bonus

Example 4: Go-Explore

"First return, then explore"

Fig. 1: Overview of Go-Explore.





(Uber AI Labs)

Soft Actor Critic: Entropy Regularization

$$U(\pi) = E\left[\sum_{t=0}^{\infty} \gamma^t \left(r_t + \alpha \mathcal{H}(\pi(\cdot \mid s_t))\right)\right]$$

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

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

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

Soft Actor Critic

Algorithm 1 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

$$J_V(\psi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}}\left[\tfrac{1}{2}\left(V_\psi(\mathbf{s}_t) - \mathbb{E}_{\mathbf{a}_t \sim \pi_\phi}\left[Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)\right]\right)^2\right]$$

$$J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \hat{Q}(\mathbf{s}_t, \mathbf{a}_t)\right)^2\right]$$

$$\hat{Q}(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma\, \mathbb{E}_{\mathbf{s}_{t+1} \sim p}\left[V_{\bar{\psi}}(\mathbf{s}_{t+1})\right]$$

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi_\phi(\,\cdot\,|\mathbf{s}_t) \,\middle\|\, \frac{\exp\left(Q_\theta(\mathbf{s}_t, \cdot)\right)}{Z_\theta(\mathbf{s}_t)}\right)\right]$$

Soft Actor Critic

Advantages:

- St
- Le
- Ex
- Ins
- Of

Disadvantages

py

Algorithm 1 Soft Actor-Critic

Input: θ_1, θ_2, ϕ ▷ Initial parameters

$\bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2$ ▷ Initialize target network weights

$\mathcal{D} \leftarrow \emptyset$ ▷ Initialize an empty replay pool

for each iteration **do**

  for each environment step **do**

    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$ ▷ Sample action from the policy

    $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ ▷ Sample transition from the environment

    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$ ▷ Store the transition in the replay pool

  end for

  for each gradient step **do**

    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$ ▷ Update the Q-function parameters

    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$ ▷ Update policy weights

    $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$ ▷ Adjust temperature

    $\bar{\theta}_i \leftarrow \tau\theta_i + (1 - \tau)\bar{\theta}_i$ for $i \in \{1, 2\}$ ▷ Update target network weights

  end for

end for

Output: θ_1, θ_2, ϕ ▷ Optimized parameters

Wisdom

Deep RL: The Dream

openai / gym

Watch 995

Code | Issues 112 | Pull requests 13 | Actions | Projects | Wiki | Security | Insights

Leaderboard

Aman Arora edited this page 3 days ago · 352 revisions

This page tracks the performance of user algorithms for various tasks in gym. Previously, users could submit their scores directly to gym.openai.com/envs, but it has been decided that a simpler wiki might do this task more efficiently.

This wiki page is a community driven page. Anyone can edit this page and add to it. We encourage you to contribute and modify this page and add your scores and links to your write-ups and code to reproduce your results. We also encourage you to add new tasks with the gym interface, but not in the core gym library (such as roboschool) to this page as well.

Links to videos are optional, but encouraged. Videos can be youtube, instagram, a tweet, or other public links. Write-ups should explain how to reproduce the result, and can be in the form of a simple gist link, blog post, or github repo.

We have begun to copy over the previous performance scores and write-up links over from the previous page. This is an ongoing effort, and we can use some help.

Environments

Classic control

CartPole-v0

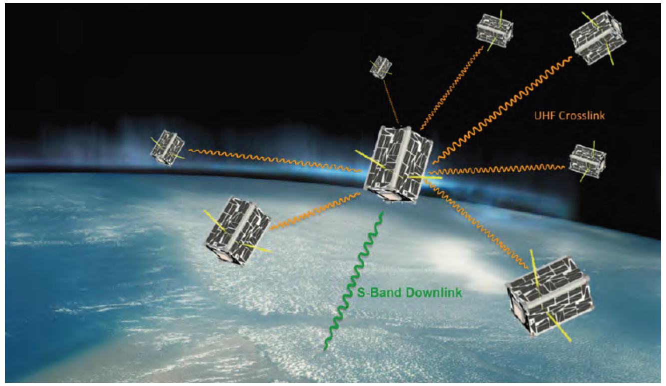
A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of +1 or -1 to the cart. The pendulum starts upright, and the goal is to prevent it from falling over. A reward of +1 is provided for every timestep that the pole remains upright. The episode ends when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center.

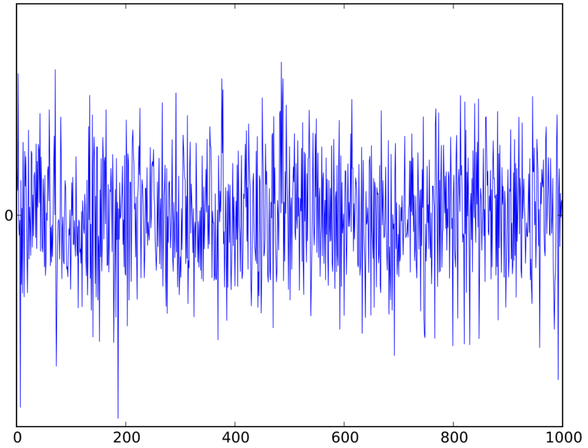
- Environment Details
- *CartPole-v0 defines "solving" as getting average reward of 195.0 over 100 consecutive trials.*
- *This environment corresponds to the version of the cart-pole problem described by Barto, Sutton, and Anderson [Barto83].*

User	Episodes before solve	Write-up	Video
Zhiqing Xiao	0 (use close-form preset policy)	writeup	
Hengjian Jia	0 (use close-form PID policy)	code/writeup	
Keavnn	0	writeup	
Shakti Kumar	0	writeup	Video
Nextgrid.ai 🔥	0	writeup	Video
iRyanBell	2	writeup	

Using Deep RL for your problem

1. Some interesting problem (smallsat swarm)
2. Spend weeks theorizing about the exact-right cost function and dynamics
3. Decide RL can solve all of your problems
4. Fire up open-ai baselines
5. Does it work??



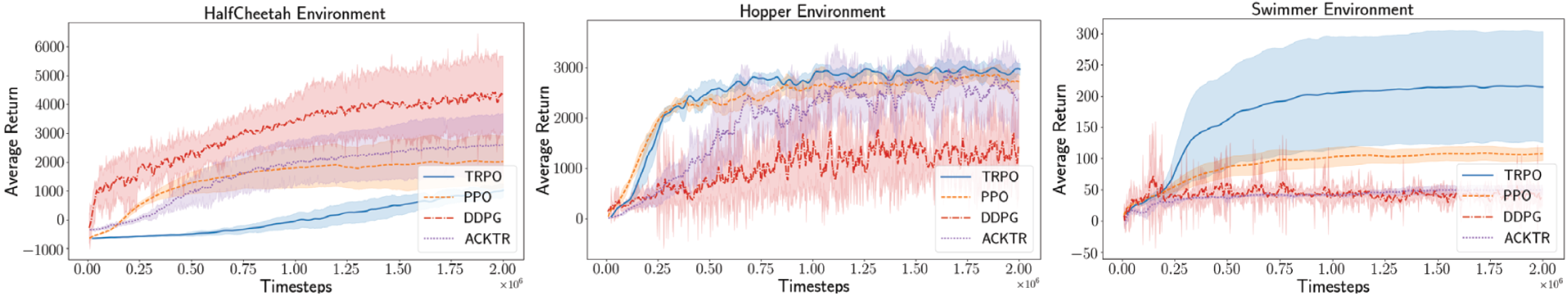


openai / baselines

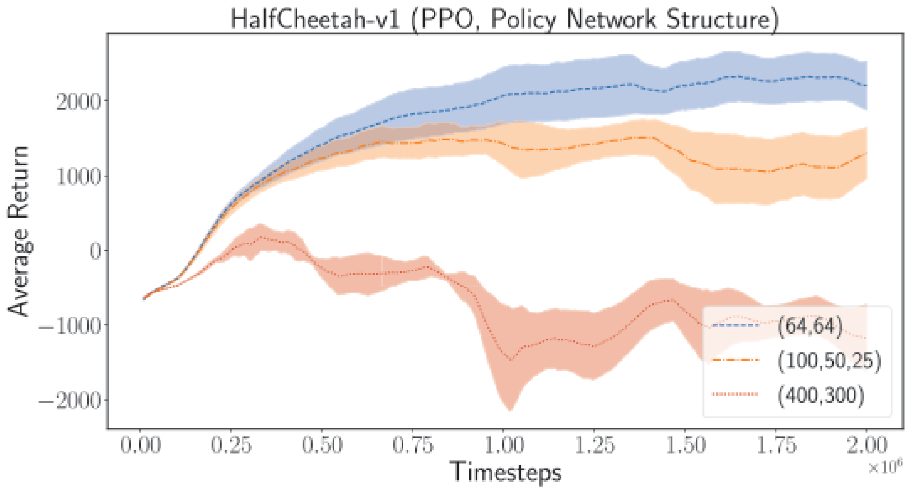
Why not?

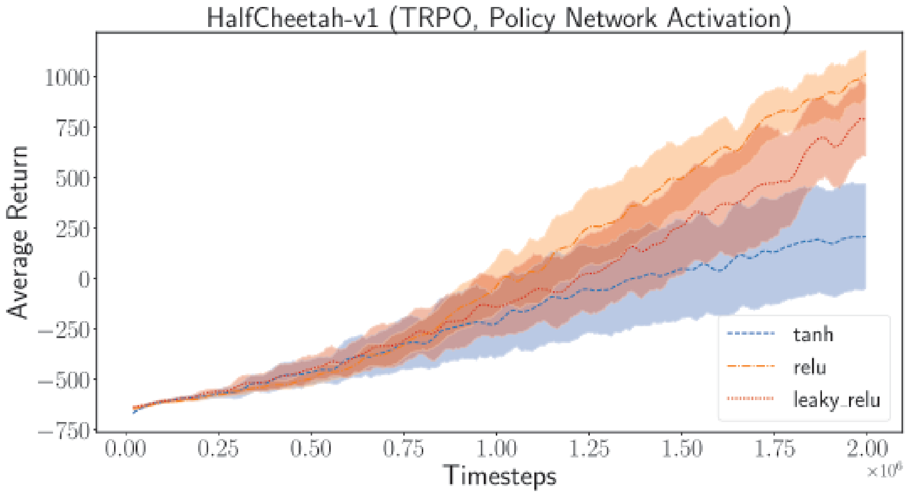
- Hyperparameters?
- Reward scaling?
- Not enough training time????

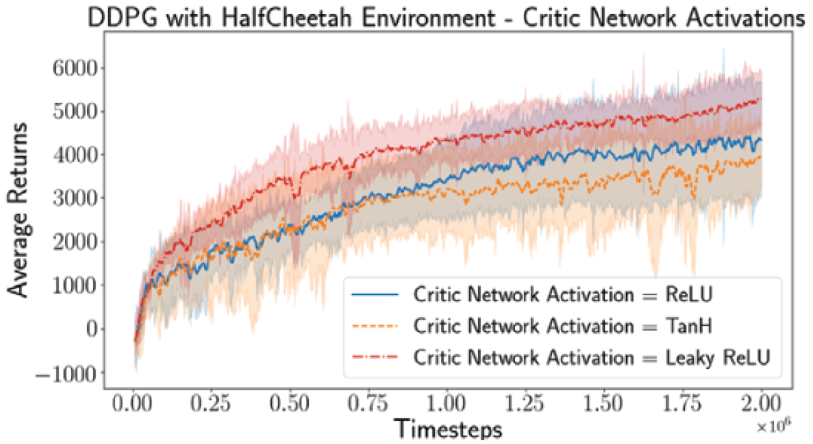
Algorithms


HalfCheetah Environment
Hopper Environment
Swimmer Environment
Average Return
Timesteps
$\times 10^6$
TRPO
PPO
DDPG
ACKTR

Policy Network Architecture

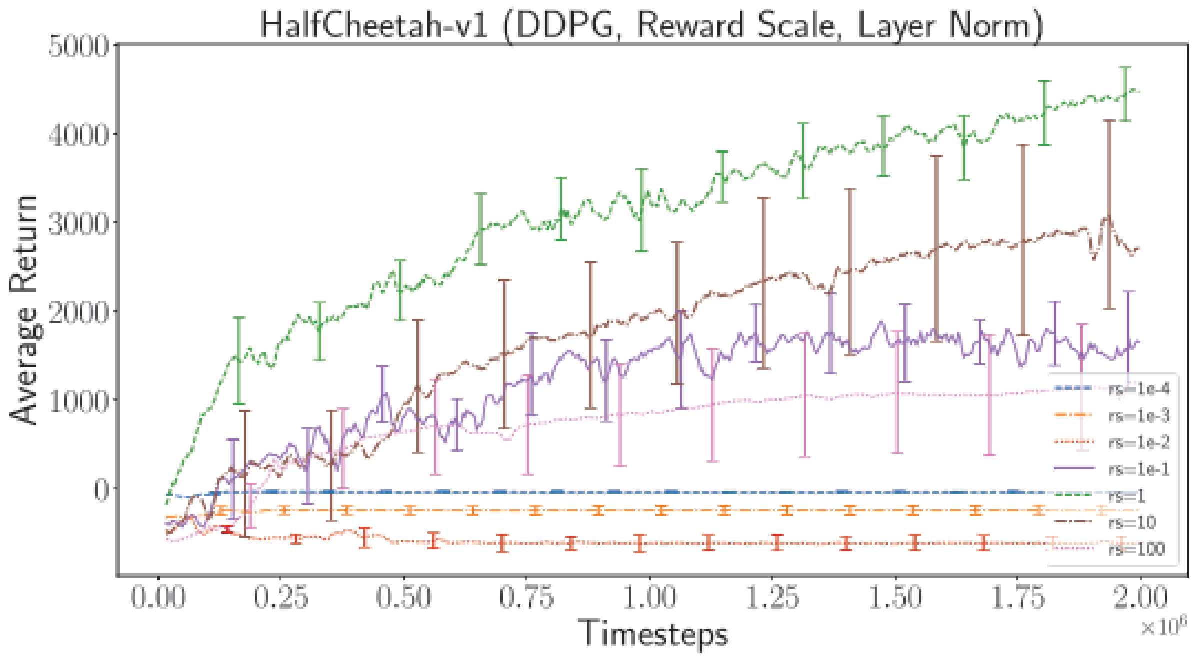


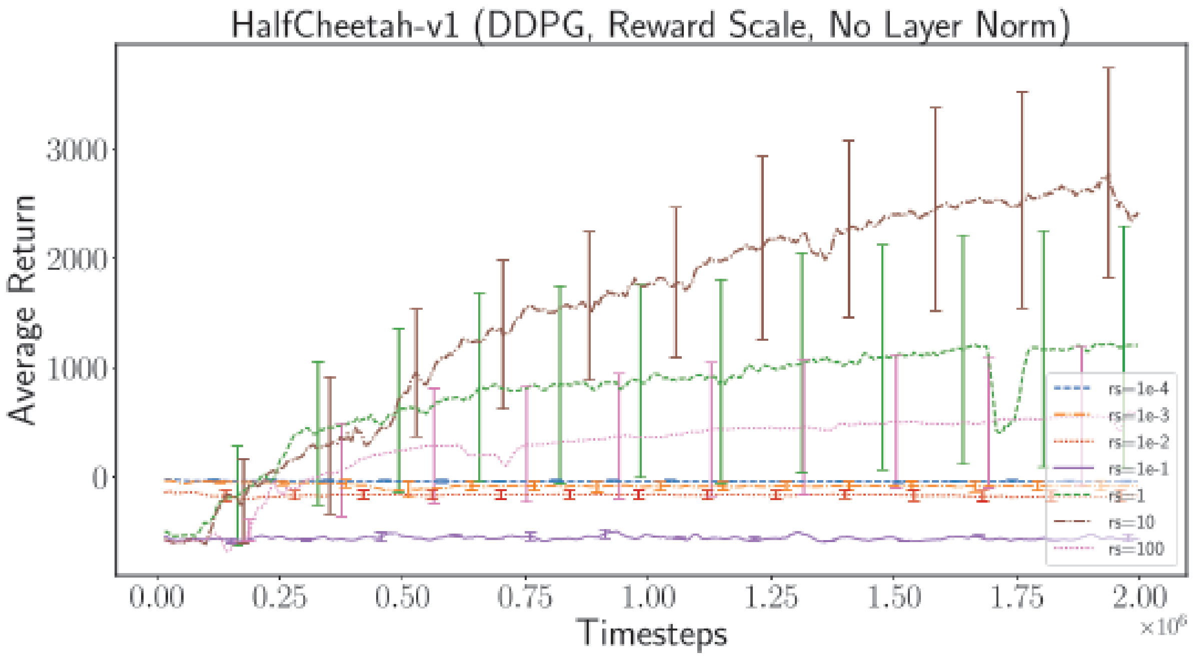




Reward Rescaling

"simply multiplying the rewards generated from an environment by some scalar"





Statistical Significance

"Unfortunately, in recent reported results, it is not uncommon for the top-N trials to be selected from among several trials (Wu et al. 2017; Mnih et al. 2016)"

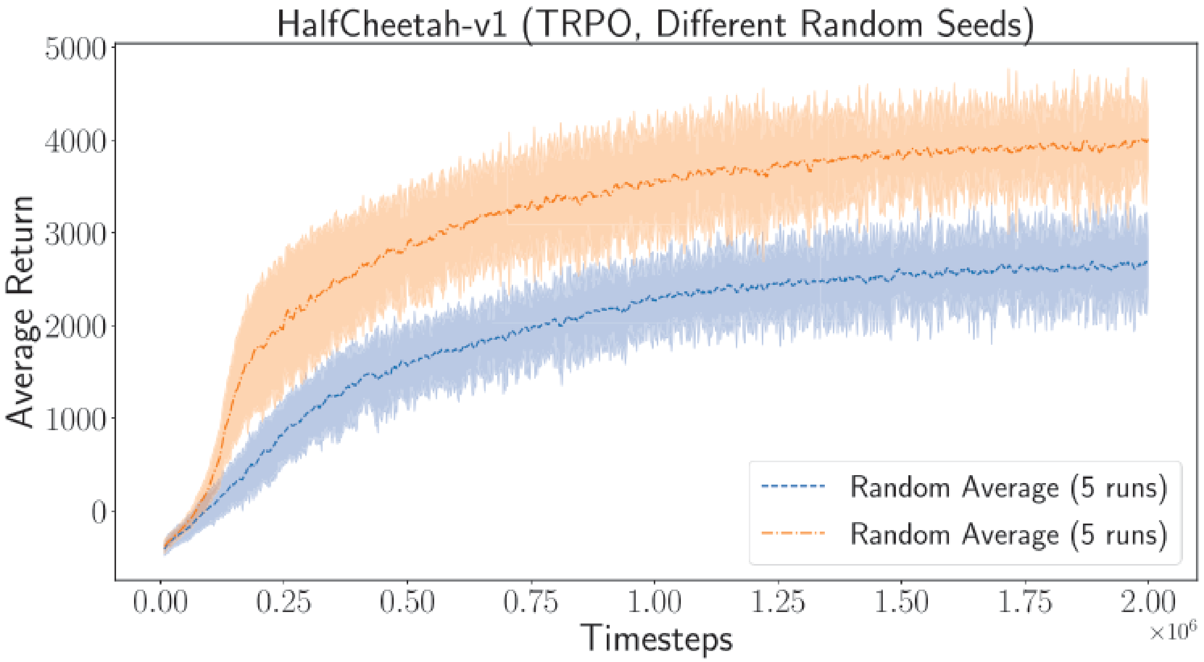
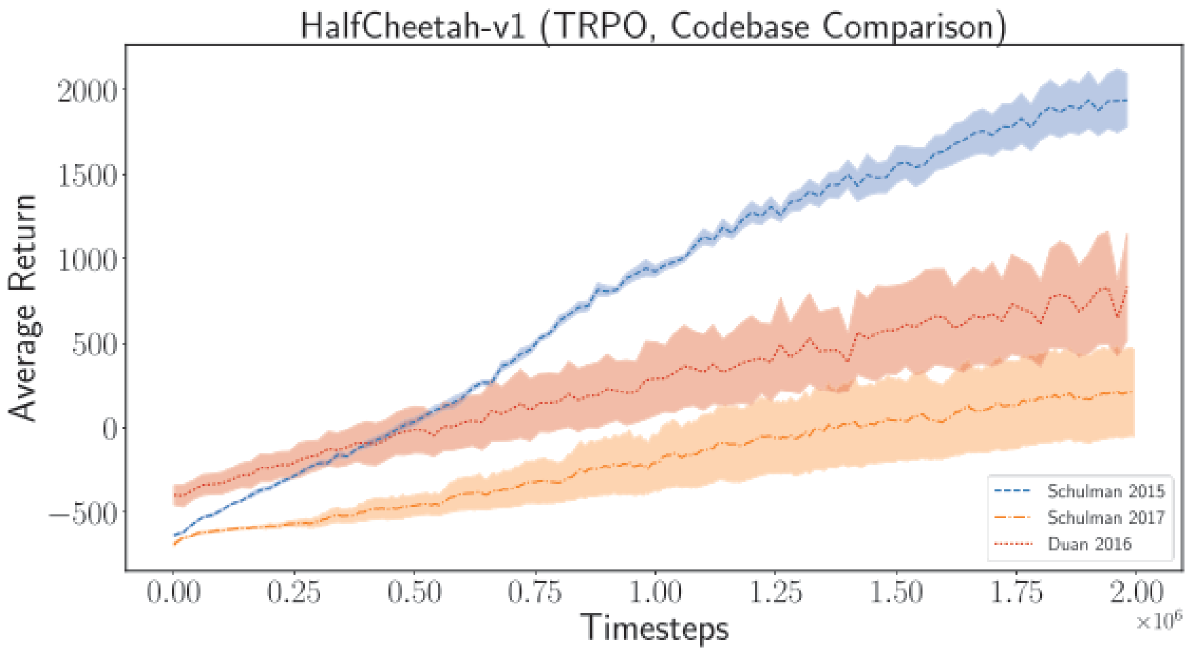
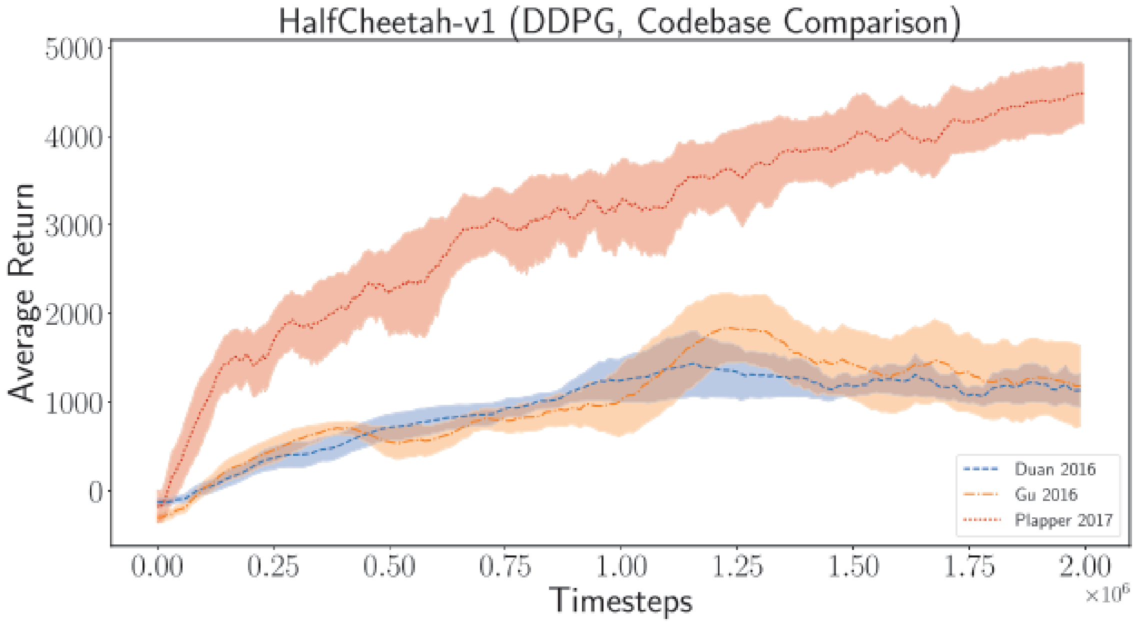


Figure 5: TRPO on HalfCheetah-v1 using the same hyperparameter configurations averaged over two sets of 5 different random seeds each. The average 2-sample t-test across entire training distribution resulted in $t = -9.0916$, $p = 0.0016$.

Codebases



HalfCheetah-v1 (TRPO, Codebase Comparison)
Average Return
Timesteps
$\times 10^6$
Schulman 2015
Schulman 2017
Duan 2016



HalfCheetah-v1 (DDPG, Codebase Comparison)
Average Return
Timesteps
$\times 10^6$
Duan 2016
Gu 2016
Plapper 2017

How to choose an RL Algorithm

(According to Sergey Levine)

Sample Efficiency

Ease of Use / Stability

← Fewer Samples

More Stable (?) →

Model-Based RL

Model-Based Deep RL

Off-Policy Q-Learning

Actor Critic

On Policy Policy Gradient

Evolutionary/ Gradient Free

(Most people use SAC or PPO)

How to be successful with RL

- Always start with a small problem that works and scale up (keep verifying that it works with every change)
- Plot everything that you can think of (TensorBoard)
  - *Losses*
  - Policies
  - Value functions
  - Trajectories
  - (Average return) Learning curve
- Keep calm and lower your learning rate