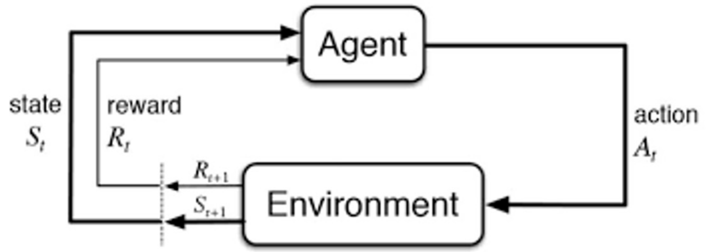
Soft Actor-Critic Algorithms and Applications

By: Tuomas Haarnoja et al. 2018

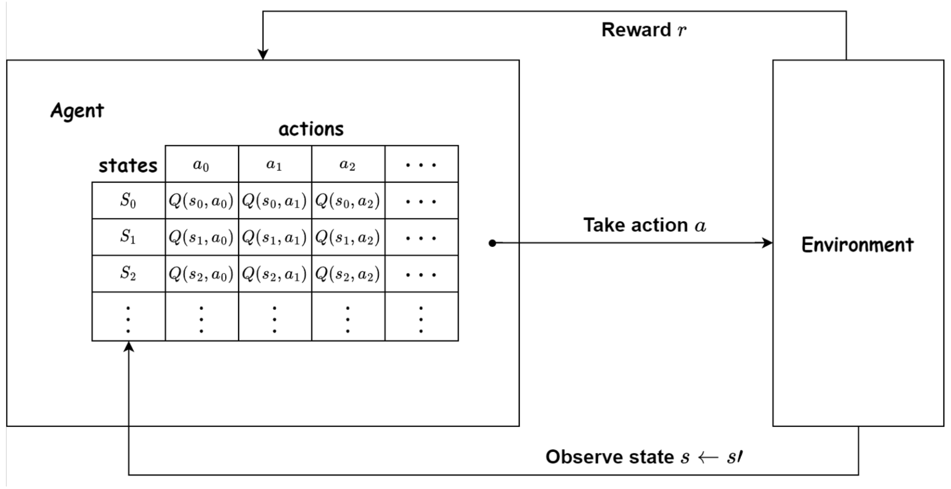
Presenter: Jacob Rivera
09/26/2023

Background

Reinforcement Learning:



Q-Learning:



states	a_0	a_1	a_2	$\cdots$
S_0	$Q(s_0,a_0)$	$Q(s_0,a_1)$	$Q(s_0,a_2)$	$\cdots$
S_1	$Q(s_1,a_0)$	$Q(s_1,a_1)$	$Q(s_1,a_2)$	$\cdots$
S_2	$Q(s_2,a_0)$	$Q(s_2,a_1)$	$Q(s_2,a_2)$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Motivation

- Model-free deep reinforcement learning is expensive in terms of sample complexity, want to make it real-world applicable
    - Millions of steps of data collection
    - Complex behavior with high dimensionality
- Model-free deep RL is extremely brittle with respect to hyperparameters
    - Learning rates, constants and other settings must be set specifically per task
- Handle Continuous Action Spaces
- Exploration-Exploitation Trade-off

Contributions

- Soft Actor-Critic Algorithm
  - Combines elements of Q-learning and entropy regularization
- Automatic Temperature Tuning
  - Use of entropy regularization adds stability
- Sample Efficiency
  - Improved compared to other RL algorithms, reducing number of sample required for training
- Handling Continuous action Spaces
  - Addresses challenges posed by continuous spaces, showing real-world application
- Exploration vs Exploitation Balance
- Scalability/Application

Problem Setting

Learning Problem Definition:

- a policy search in a Markov Decision Process (MDP) – $(\mathcal{S}, \mathcal{A}, p, r)$

state space: $\mathcal{S}$ action space: $\mathcal{A}$

Policy for a particular action: $\pi(\mathbf{a}_t|\mathbf{s}_t)$.

for action: $\mathbf{a}_t \in \mathcal{A}$ and $\mathbf{s}_t \in \mathcal{S}$

Maximum Entropy Reinforcement Learning: $\pi^* = \arg\max_{\pi} \sum_t \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi} \left[r(\mathbf{s}_t, \mathbf{a}_t) + \alpha\mathcal{H}(\pi(\,\cdot\,|\mathbf{s}_t))\right]$

Bellman Equations:

$$J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\mathcal{D}} \left[\frac{1}{2}\left(Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \left(r(\mathbf{s}_t, \mathbf{a}_t) + \gamma\,\mathbb{E}_{\mathbf{s}_{t+1}\sim p}\left[V_{\bar{\theta}}(\mathbf{s}_{t+1})\right]\right)\right)^2\right]$$

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t\sim\mathcal{D},\epsilon_t\sim\mathcal{N}} \left[\alpha \log \pi_\phi(f_\phi(\epsilon_t;\mathbf{s}_t)|\mathbf{s}_t) - Q_\theta(\mathbf{s}_t, f_\phi(\epsilon_t;\mathbf{s}_t))\right]$$

$$J(\alpha) = \mathbb{E}_{\mathbf{a}_t\sim\pi_t}\left[-\alpha \log \pi_t(\mathbf{a}_t|\mathbf{s}_t) - \alpha\bar{\mathcal{H}}\right]$$

Full Proof and Additional Reading

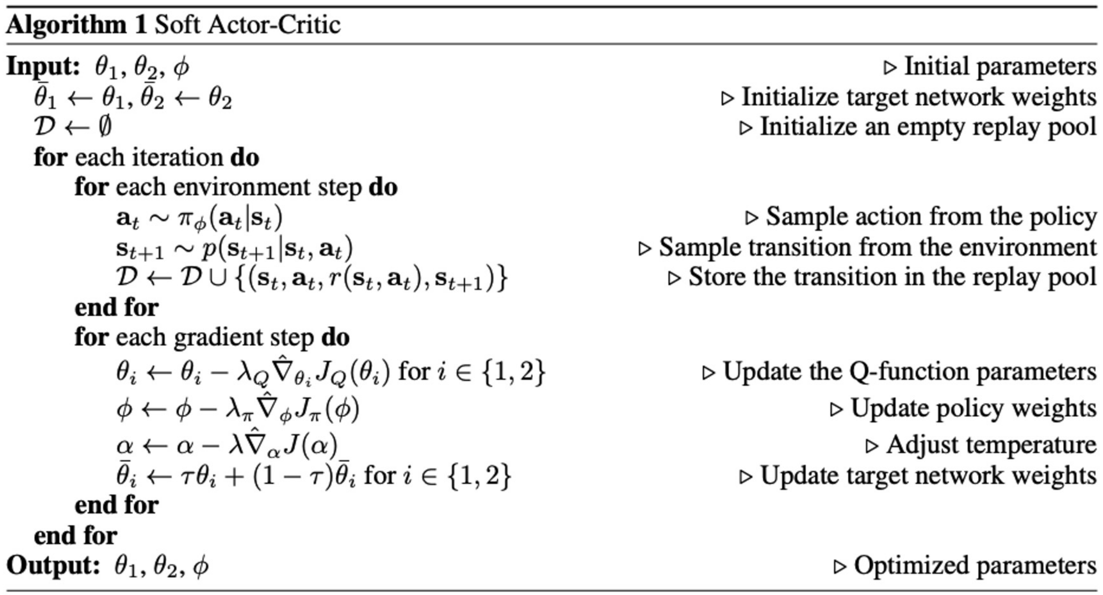
- Principle of Maximum Entropy
- Appendix B.1, B.2, B.3 for full proofs
- Dual Gradient Descent
- Addressing Function Approximation Error in Actor-Critic Methods
- Double Q-Learning

Related Work

Maximum Entropy Inverse Reinforcement Learning (Ziebart et al., 2008)

- Developed technique of creating solutions to Markov Decision Problems by utilizing the principle of maximum entropy
- Limited by its discrete state requirements, not applied to robotics

Algorithm Breakdown



Algorithm 1 Soft Actor-Critic

```
Input: $\theta_1, \theta_2, \phi$                                    ▷ 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: $\theta_1, \theta_2, \phi$                                   ▷ Optimized parameters
```

- Initialize parameters, network weights, and relay pool
- Each env step: sample action, sample state space for transition, and store the transition info
- Each gradient step: Update Q-function, policy weights, temperature, and network weights
- Output parameters

Simulated Benchmark Experimental Setup

Using OpenAI Gym benchmark Suite on different environments with different robotic control tasks

Tasks include Hopper-v2, Walker2d-v2, HalfCheetah-v2, Ant-v2, Humanoid-v2, and Humanoid (rllab)

Train on:

- five different instances of each algorithm with different random seeds
- each performing one evaluation rollout every 1000 environment steps.

Hypothesis:

- SAC will outperform the other baseline models in comparison on the same task in both learning speed and overall performance on numerous robot tasks
- SAC's automatic temperature tuning will improve performance by eliminating manual tuning, improving performance across multiple environments

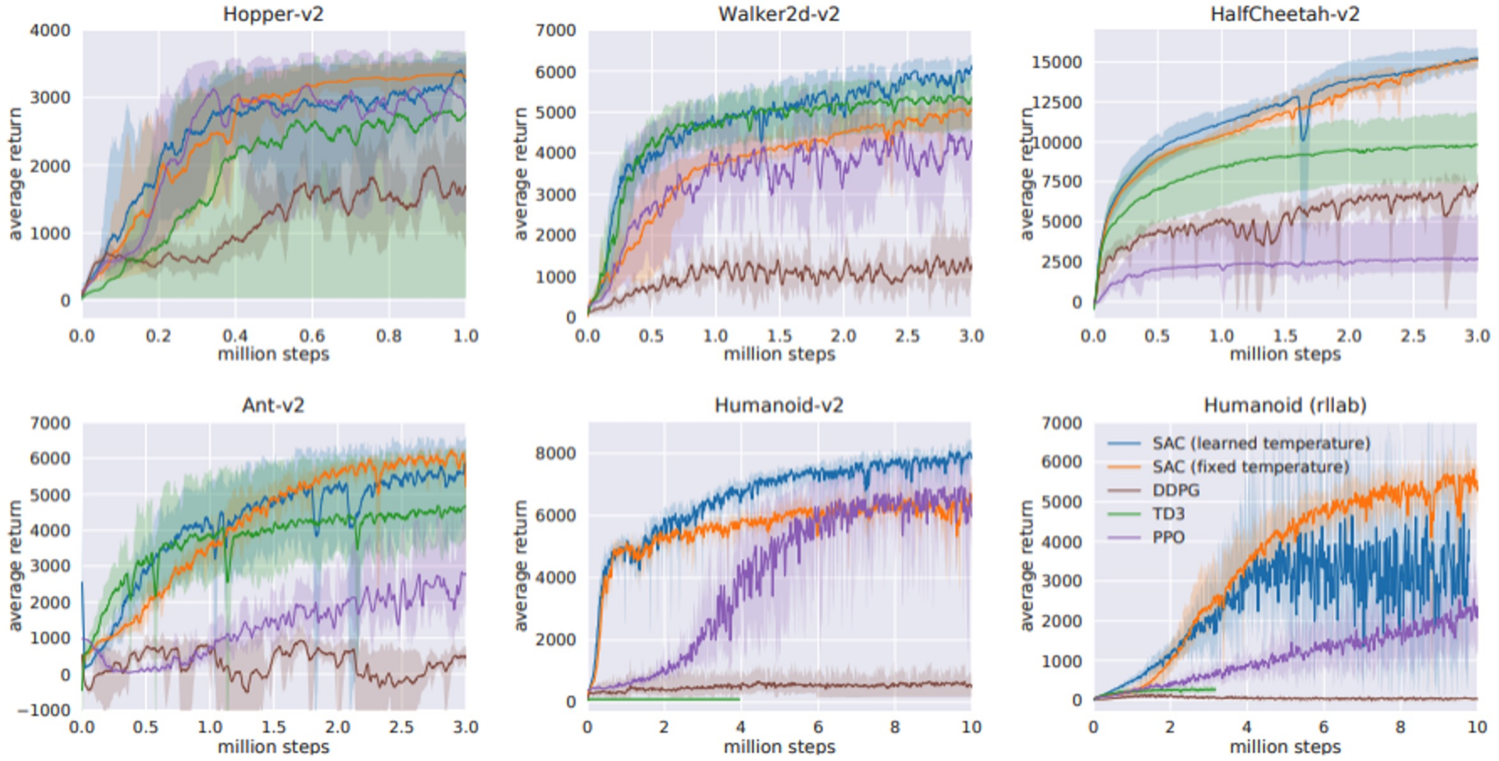
Simulated Benchmark Experimental Setup cont.

Baseline – Compare the performance of the Soft Actor-Critic (**SAC**) to the following:

- **DDPG** (Deep Deterministic Policy Gradient) : off policy
  - closely associated with Q-learning
  - adapted for environments with continuous action spaces
- **PPO** (Proximal Policy Optimization) : on-policy
  - focuses on optimizing a policy for max expected cumulative rewards
- **SQL** (Soft Q-Learning) : off-policy
  - extends Q-learning with being stochastic and soft value functions, but is a policy optimizer
- **TD3** (Twin Delayed Deep Deterministic Policy Gradient) : off-policy
  - learns deterministic policies, explores through added noise.

Experimental Results

Performance Metrics: Average Return



Quadrupedal Locomotion Experimental Setup

Minitaur robot

- 4 legs, eight direct-drive actuators controlling legs, two actuators per leg form sagittal plane movement
- Motor encoders measure motor angles, and an IMU measures orientation and angular velocity of the Minitaur’s base

Hypothesis:

- SAC can allow the robot to learn walking patterns in real world situations, even with limited robot robustness (can lose balance easily and break)
- The learned policy can be generalized to different terrains without prior training

Quadrupedal Locomotion Experimental Setup cont.

To evaluate, a limited number of environment steps (160k) and episodes were enforced to demonstrate real world training times (about 2 hours), equating to about 400 episodes with maximum length of 500 steps

The robot was also placed on different terrains without being restrained.

- slopes
- wooden block obstacles
- stairs

Successes in the new environment will support the authors claim for generalizability.

Results

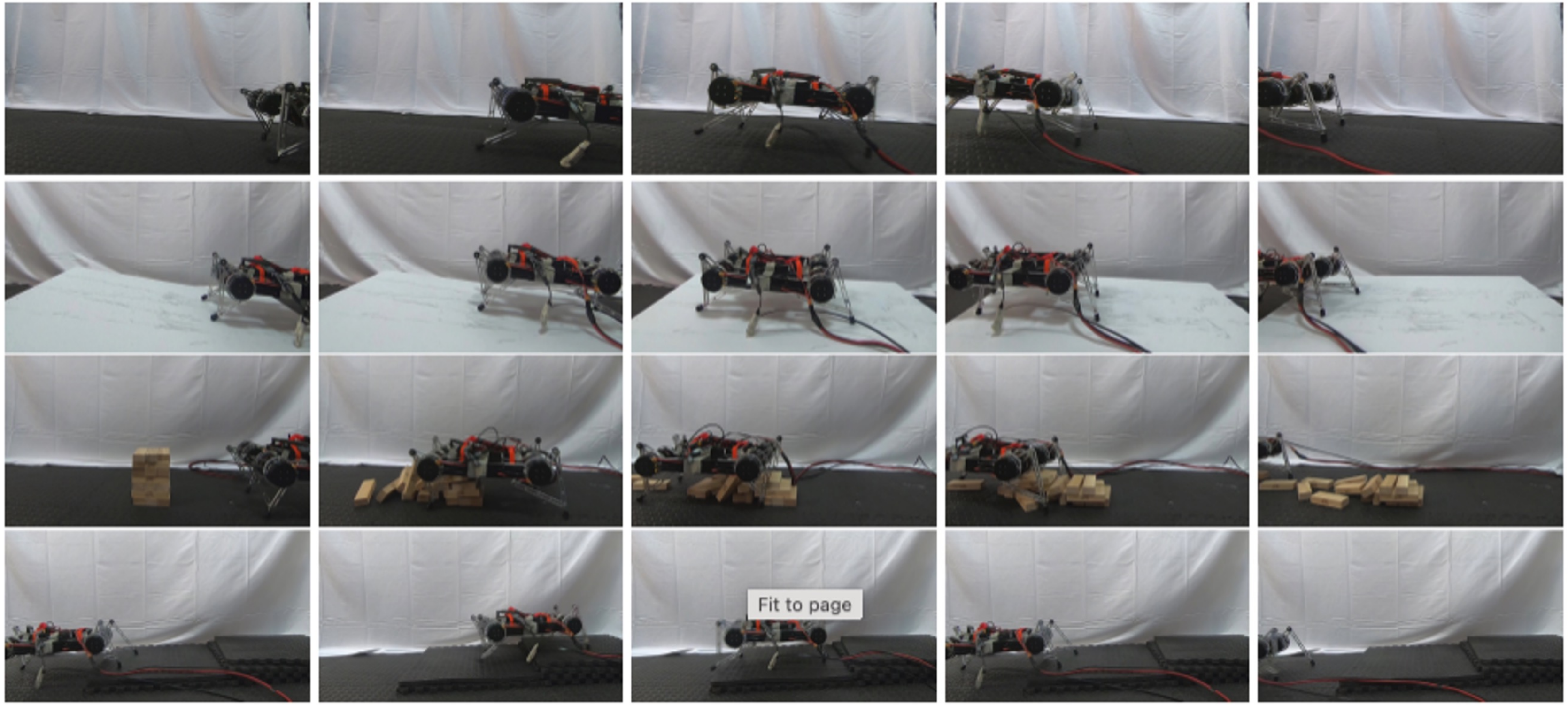


Figure 2: We trained the Minitaur robot to walk on flat terrain (first row) and tested how the learned gait generalizes to unseen situations (other rows)

54 min
5x

Dexterous Hand Manipulation Experimental Setup

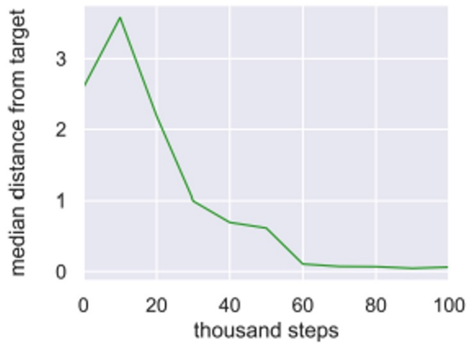
The “dclaw” hand

- 3 fingered, 9 DoFs controlled by Dynamixel servo-motors
- Environment is perceived by raw RGB images, which are processed in a neural network, forcing the policy to learn from these images

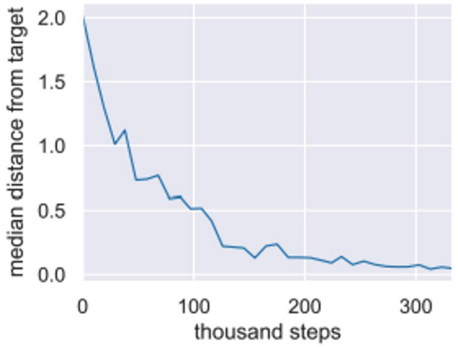
Hypothesis:

- SAC can enable 3-finger dexterous robotic hand to complex tasks directly from raw RGB images, without pre training or simulation
- The learned policy can perceive the valve, its orientation from the images, generated coordinated finger movements to rotate the valve
- Demonstrate the ability to learn the task without direct knowledge of the valve’s position or orientation

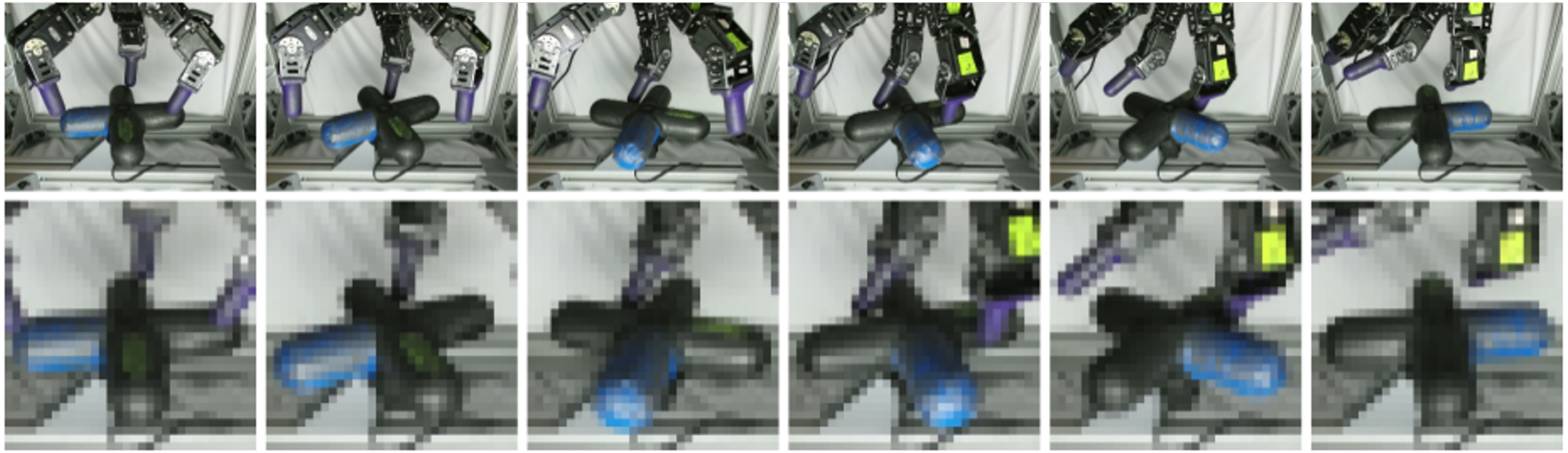
Results



(a) Learning from valve joint angle.



(b) Learning from images.



Discussion of Results

Conclusions from experiments

- The success of SAC learning via off-policy reinforcement learning is successful at training a model in real time to handle baseline environments, as well as generalize to environment variations
- Demonstrated the ability for sample efficiency will the addition of a continuous action space

The authors seemed to be content with the boosted performance in the simulation experiment, which was supported by the graphs demonstrating increased average return over other models

But the other experiments lacked many of the quantitative data to back up its claims of state-of-the-art superiority to other off-policy deep RL.

- Trainings of the real robot using other algorithms could have demonstrated a true comparison

Critique / Limitations / Open Issues

One of the main limitations to this approach is the expertise required to understand the algorithm/model, possibly making it difficult for the average computer scientist/researcher to put the algorithm into practice.

Large limitation of relatively large training times are required for the agent to be able to perform the given task. While hopeful and a step forward for general purpose robotics, it lacks testing in real world environments, ones that may not give the 4-20 hours to train on.

Use of valve location vs RGB images to train shows possibility of multimodal inputs to learn on.

Real world environment tasks are limited in evaluation and lack comparison against other models.

Future Work for Paper / Reading

“To our knowledge, these results represent the first evaluation of deep reinforcement learning for real-world training of underactuated walking skills with a quadrupedal robot, as well as one of the most complex dexterous manipulation behaviors learned with deep reinforcement learning end-to-end from raw image observations.”

Next Steps:

- Evaluate algorithm on new, novel environments
- Stress test the learned policies to evaluate their limits
- Learn and evaluate the algorithm on more situations and tasks

Extended Readings

How to train your robot with deep reinforcement learning: lessons we have learned

Continuous control with deep reinforcement learning

Model-Free Real-Time Autonomous Control for a Residential Multi-Energy System Using Deep Reinforcement Learning

Summary

- The authors tackle the problem of creating a general purpose model free deep RL algorithm that can efficiently handle continuous action spaces
- Solving this problem is a large leap towards general purpose robotics, without large amounts of pre-training, a problem that often causes a lot of overhead for most models
- Prior work was limited in this sample efficiency and lack of true actor-critic models
- Key insights:
  - Off-policy, entropy learning temperature, and Actor-Critic models allow for complex, dexterous policy learning behaviors
  - deep reinforcement learning can also be successful with different modes of input