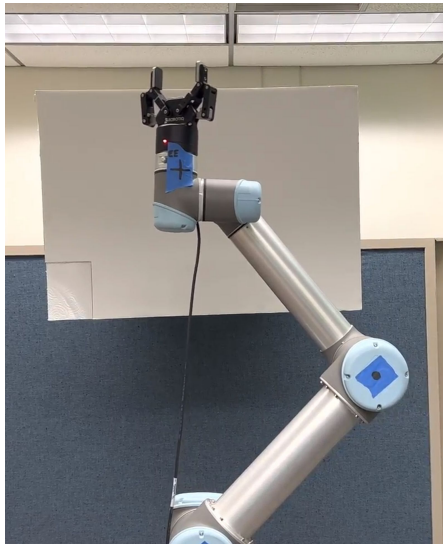
EEE598 Reinforcement Learning

RL in Real World Robotics task training

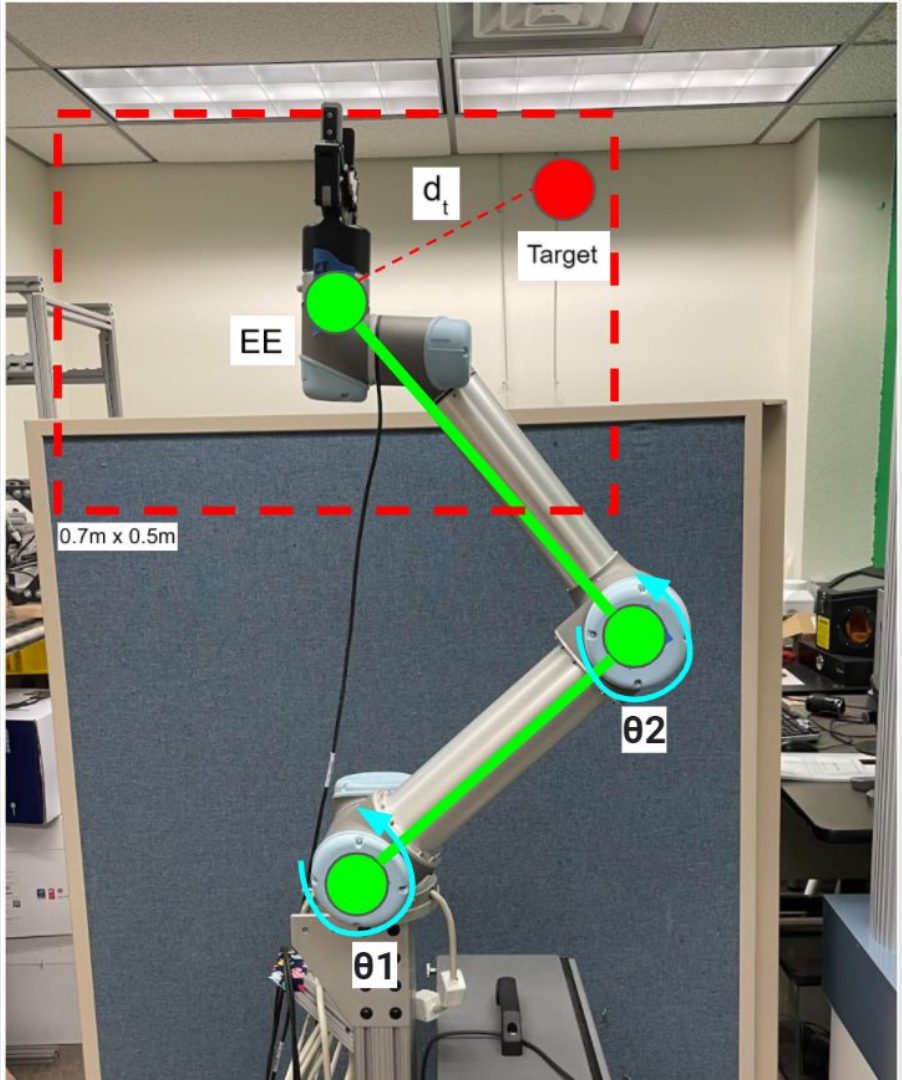
Team 6
George Muhn, Kylel Scott, Jacob Sindorf

Problem Statement

Many real world robotics tasks require complex and detailed calculations in order to accomplish the simplests of tasks. When a robot is successful in these tasks it is solely limited to that specific task. Through reinforcement learning, Robotic tasks such as reaching can be trained and applied to real world robots such as the UR5 to reduce the need for rigorous math and control loops. However, poor documentation and extensive software dependencies hinder the replicability of such projects.



UR5 Reinforcement Learning Setup



Configuration

Action (At ∈ A)

θ(dot)(1,2) ∈ [−0.3, +0.3]

State/Observation

θ, θ(dot), At-1, dt

Reward (Rt ∈ R)

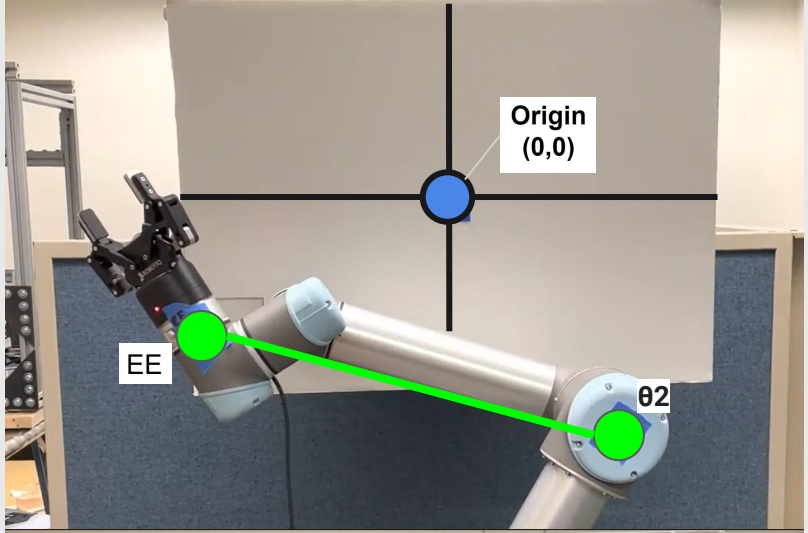
Rt = −dt + exp(−100d2t)

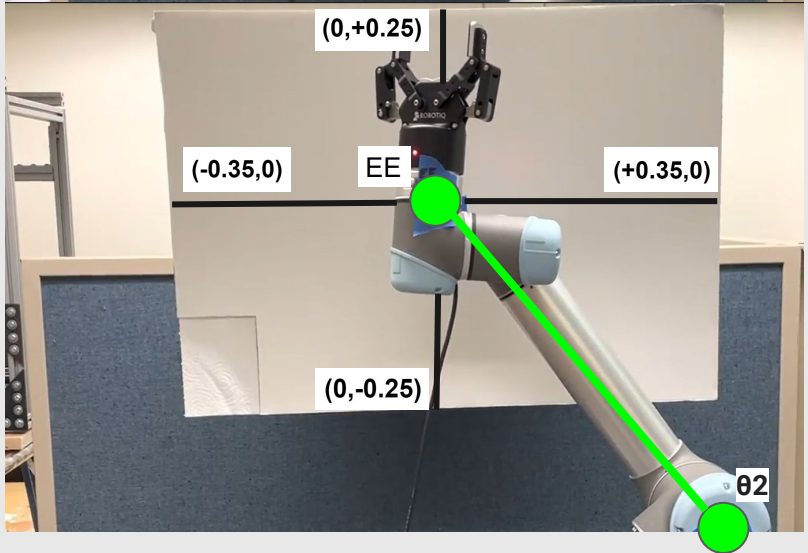
State Dynamics

$$\frac{\partial L}{\partial q^i}(t, \boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)) - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}^i}(t, \boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)) = 0,$$

$$i = 1, \ldots, n.$$

Values and info from (1)





Configuration

Boundary

0.7m by 0.5m, with origin (0,0) in center of the rectangle

Episodes

Start at origin, random target, 4 seconds, reset to Origin

Values and info from (1)

Software Setup

- Ubuntu 64 bit (16.04,20.04)
  - Virtual Machine (VirtualBox)
- Python3.7, Tensorflow 1.15
- Github: OpenAI Baselines, SenseAct, SenseActExperiments*
  - Docker*
- UR5 v 3.3.4.310



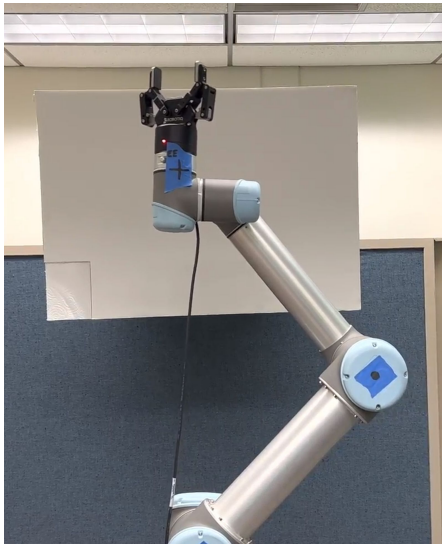



Hardware Setup

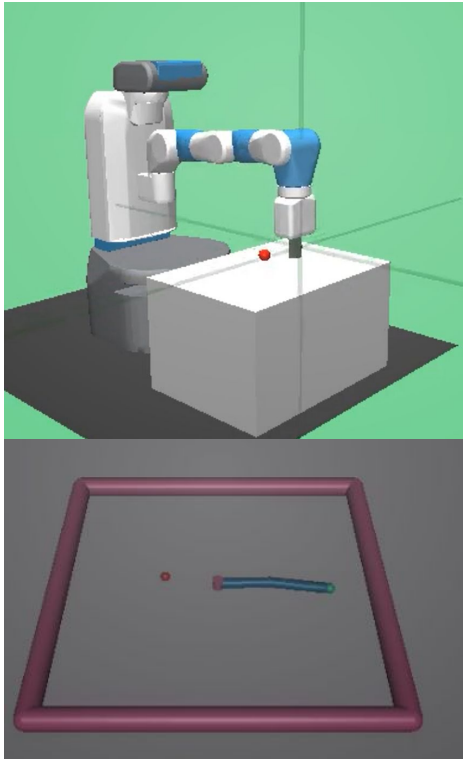
Physical (Our Project) vs Virtual Environment

- Ethernet connection
- IP address dependant

Physical



Virtual



OpenAi FetchReach (3) and Reacher (4)

TRPO and PPO

Trust Region Policy Optimization (TRPO)

$$\underset{\theta}{\text{maximize}}\, \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}},a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta_{\text{old}}}(s,a)\right] \quad \text{subject to } \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|s)\parallel\pi_\theta(\cdot|s))\right]\leq\delta.$$

Proximal Policy Optimization (PPO)

$$L_\theta^{CLIP}=\mathbb{E}_{s,a\sim\pi_{\theta_{\text{old}}}}\left[\min(r_\theta(a|s)A_{\theta_{\text{old}}}(a,s),\text{clip}(r_\theta(a|s),1-\varepsilon,1+\varepsilon)A_{\theta_{\text{old}}}(s,a))\right],$$

Running Experiments

- IP address required
- Hyperparameters
  - Can easily change
- Run train.py
- Each full experiment takes 2.5 hours with no failure

TRPO								
#	Average Return	Hidden Layers	Hidden Size	Batch Size	Step Size	γ	λ	δ_{KL}
1	158.56	2	64	4096	0.00472	0.96833	0.99874	0.02437
2	138.58	1	128	2048	0.00475	0.99924	0.99003	0.01909
3	131.35	4	64	8192	0.00037	0.97433	0.99647	0.31222
4	123.45	4	128	4096	0.00036	0.99799	0.92958	0.01952
5	122.60	4	32	2048	0.00163	0.96801	0.96893	0.00510

Table 3: **Hyper-parameter configurations found by random search:** The table presents the top-5 configurations found for TRPO in [8].

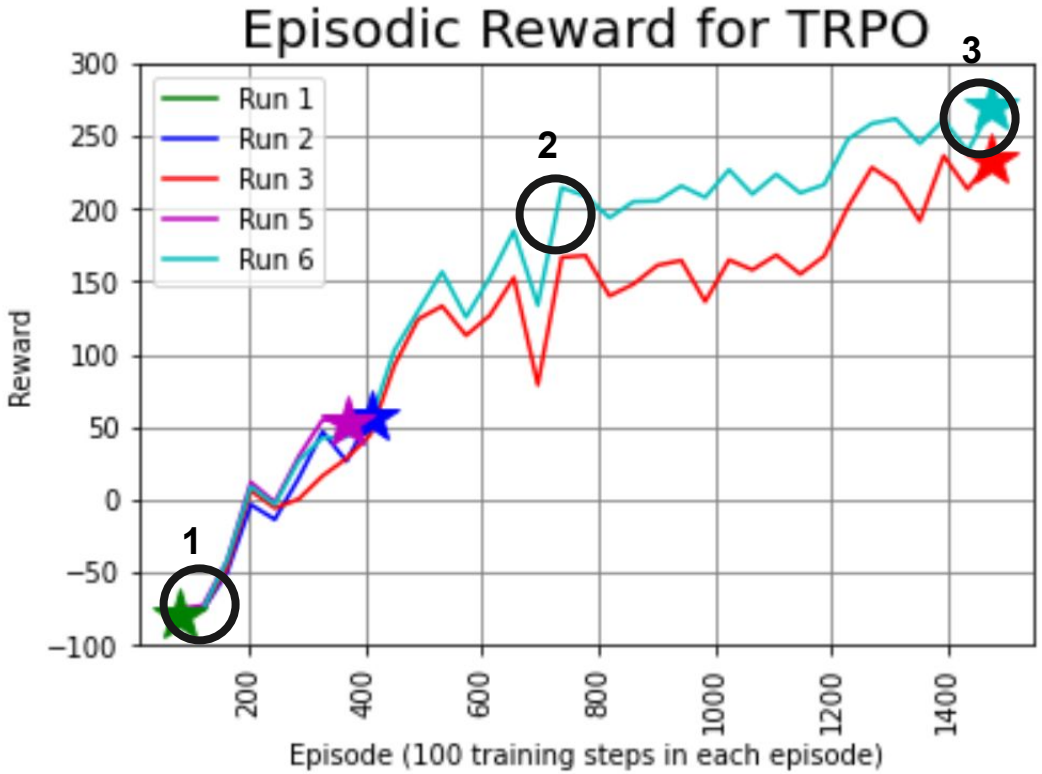
PPO								
#	Average Return	Hidden Layers	Hidden Size	Batch Size	Step Size	γ	λ	Opt. Batch Size
1	176.62	3	64	512	0.00005	0.96836	0.99944	16
2	150.25	1	16	256	0.00050	0.99926	0.98226	64
3	137.92	1	2048	512	0.00011	0.99402	0.90185	8
4	137.26	4	32	2048	0.00163	0.96801	0.96893	1024
5	136.09	1	128	2048	0.00280	0.99924	0.99003	32

Table 4: **Hyper-parameter configurations found by random search:** The table presents the top-5 configurations found for PPO in [8].

Appendix A3 from (2). Hyperparameters are not given in (1), however Lynnerup experimented and documented the top 5 for PPO and TRPO

Results and Comparison

TRPO

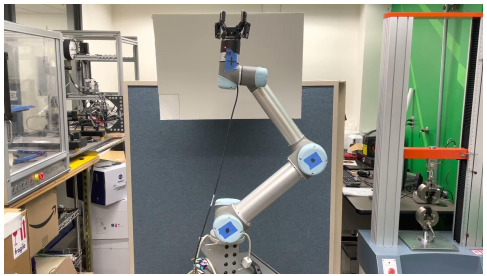


TRPO Reward = -70, Timestep = 5000

1

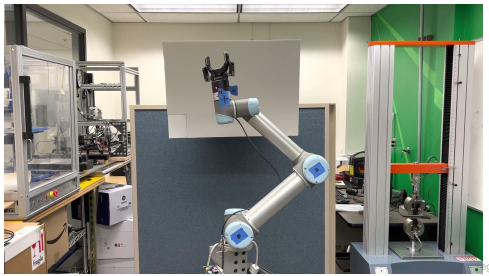
TRPO Reward = 200, Timestep = 77000

2



TRPO Reward = 257, Timestep = 150000

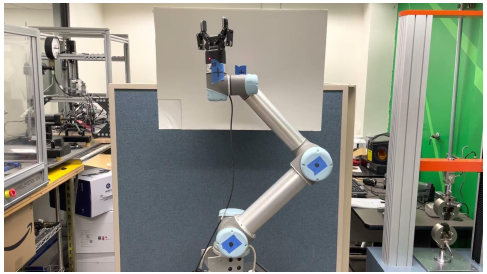
3



PPO

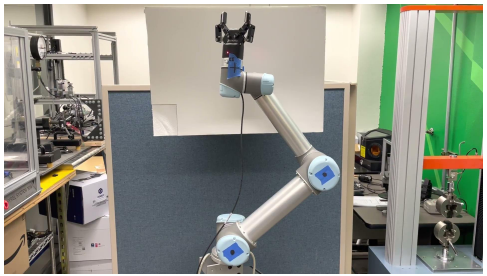
PPO Reward = -50, Timestep = 17000

1



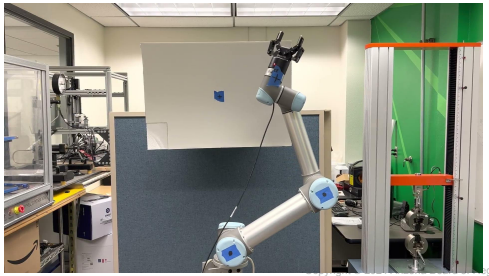
PPO Reward = 70, Timestep = 75000

2



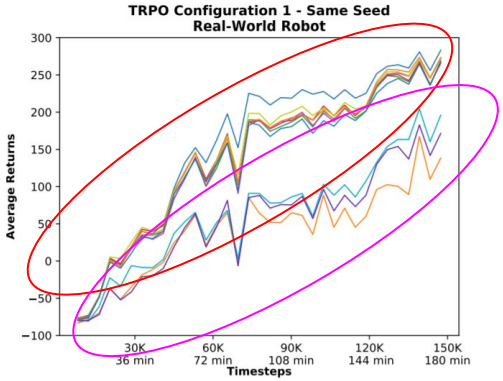
PPO Reward = 180, Timestep = 148000

3



Results Comparison

Figure 2 from (2).
Using TRPO



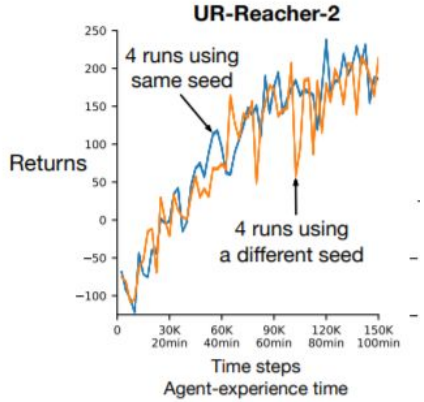
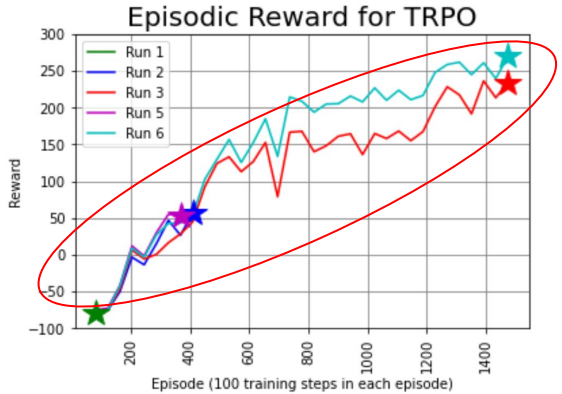
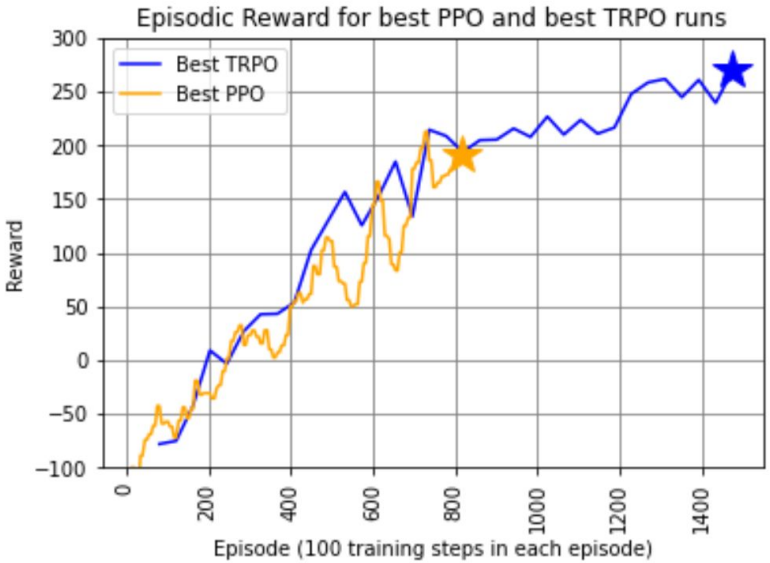


Figure 2 from (1).
Using TRPO



Results Comparison



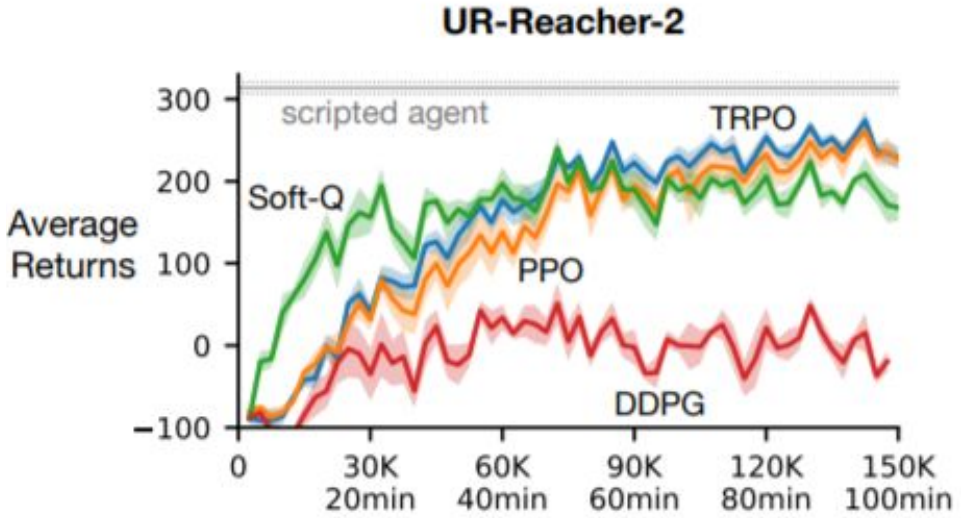


Figure 6 from (1)
(states ‘best hyperparameters’ used, but these are never given)

Discussion and Conclusion

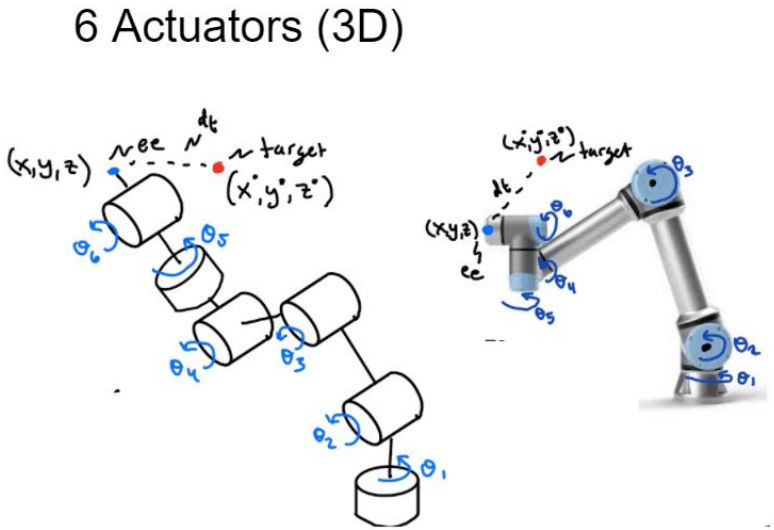
Discussion and Conclusion

- Software dependencies and documentation
- virtual vs physical environment
- Communication and failed runs

- Replicated Results
- Easy to run once started

Next Steps

- Another PPO run
- New Hyperparameters
- 6 DOF



References

[1] Mahmood, A. R., Korenkevych, D., Vasan, G., Ma, W., Bergstra, J. (2018b). Benchmarking reinforcement learning algorithms on real-world robots. In Proceedings of the 2nd Annual Conference on Robot Learning.

[2] Lynnerup, N. A., Nolling, L., Hallam, J., Hasle, R. (2019). A Survey on Reproducibility by Evaluating Deep Reinforcement Learning Algorithms on Real-World Robots. In Proceedings of the 3rd International Conference on Robot Learning - Volume 100. (Osaka, Japan). Proceedings of Machine Learning Research (PMLR)

[3] https://gym.openai.com/envs/FetchReach-v1/

[4] https://gym.openai.com/envs/Reacher-v2/

[5] Mahmood, A. R., Korenkevych, D., Komer,B. J., Bergstra, J. (2018a). Setting up a reinforcement learning task with a real-world robot. In IEEE/RSJ International Conference on Intelligent Robots and Systems.

Questions?