# Munchausen Reinforcement Learning with Continuous Action Space - Milestone Report

Marcel Brucker
*Technical University of Munich*
Munich, Germany
marcel.brucker@tum.de

Finn Süberkrüb
*Technical University of Munich*
Munich, Germany
finn.sueberkrueb@tum.de

## I. Used Tools

Stable Baselines3 (SB3) [5] is the next major version of Stable Baselines containing a set of reliable implementations of reinforcement learning (RL) algorithms like Soft Actor-Critic (SAC) in PyTorch.
RL Baselines3 Zoo [6] is a training framework for RL built on SB3 and contains scripts and trained agents to benchmark against.
OpenAI Gym [8] is a toolkit for developing and comparing RL algorithms. The gym open-source library gives you access to a standardized set of environments.
PyBullet Gymperium [7] is an open-source implementation of the OpenAI Gym MuJoCo environments using BulletPhysics' python wrapper pybullet.

## II. Experiment

We performed experiments on four different environments of PyBullet Gymperium

- AntPyBulletEnv-v0
- HalfCheetahPyBulletEnv-v0
- HopperPyBulletEnv-v0
- Walker2DPyBulletEnv-v0

comparing SAC with Munchausen SAC (M-SAC) and present the respective mean reward averaged over three seeds each.
For SAC we used optimized parameters from SB3 and only added the Munchausen term in a M-SAC subclass with the fixed parameters specified in the original Munchausen paper [1] (see table I).

## III. (Preliminary) Results

The results achieved by the SAC agent roughly match the result stated by SB3 demonstrating successful reproduction of their baselines. Our M-SAC implementation, in comparison, demonstrates similar performance. We can even observe a notable superiority of M-SAC in the environment "HalfCheetahPyBulletEnv-v0" as illustrated in table II and figure 2 in particular.

As expected, when switching from a stochastic policy (table II) to a deterministic policy, an improvement (table III) is found. For SAC it is with $(4\,\%)$ slightly higher than for M-SAC $(2.5\,\%)$. One possible interpretation would be that M-SAC already tends to a more deterministic policy during training. However, so far this hypothesis could not be verified by experiments.

| Parameter | Value |
|---|---|
| Base (Adam) SAC parameters | |
| number of timesteps | 1M |
| learning rate | 0.00073 |
| buffer size | 300000 |
| batch size | 256 |
| discount factor | 0.98 |
| Actor network structure | FC 256 - FC 256 - FC (Out) |
| Critic network structure | FC 256 - FC 256 - FC 1 |
| Activation functions | ReLU |
| optimizer | Adam (lr = 0.00073) |
| Munchausen-RL specific parameters | |
| $\alpha$ (entropy temperature) | (Automatically adjusted) $\approx 0.02$ |
| $\tau$ (Munchausen scaling term) | 0.9 |
| $l_0$ (clipping value) | -1 |

TABLE I
Parameters used for Munchausen RL agents

| Environments | SAC | M-SAC |
|---|---|---|
| AntPyBulletEnv-v0 | 3550 +/- 97 | 3458 +/- 49 |
| HalfCheetahPyBulletEnv-v0 | 2796 +/- 27 | 2891 +/- 108 |
| HopperPyBulletEnv-v0 | 2686 +/- 51 | 2719 +/- 14 |
| Walker2DPyBulletEnv-v0 | 2154 +/- 131 | 2177 +/- 100 |

TABLE II
Training results, stochastic

| Environments | SAC | M-SAC |
|---|---|---|
| AntPyBulletEnv-v0 | 3726 +/- 12 | 3534 +/- 16 |
| HalfCheetahPyBulletEnv-v0 | 2829 +/- 11 | 2899 +/- 19 |
| HopperPyBulletEnv-v0 | 2738 +/- 7 | 2772 +/- 11 |
| Walker2DPyBulletEnv-v0 | 2319 +/- 13 | 2303 +/- 8 |

TABLE III
Training results, deterministic

The following plots show the mean reward achieved by the respective agent in the respective environment averaged over three seeds. For a cleaner visualization the curves are slightly smoothed.

Because of the scaling, the Munchausen term only makes up a small fraction of $\approx 0.001\,\%$ in the Q value update. Figure 4 shows how the training behaves with larger Munchausen scalings. The weaker results may be due to the fact that smaller differences will strongly influence the update and thus force

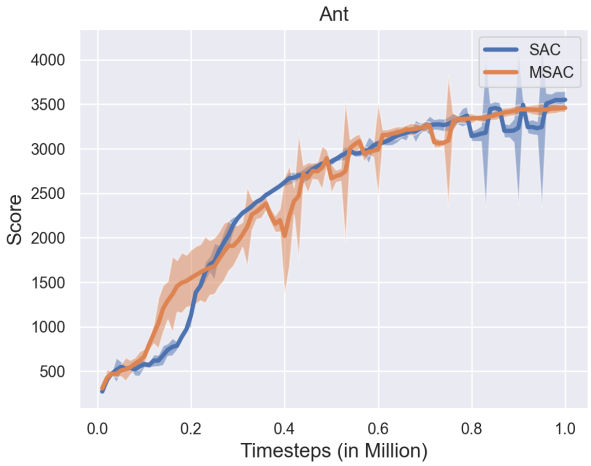


Fig. 1. Results for AntPyBulletEnv-v0

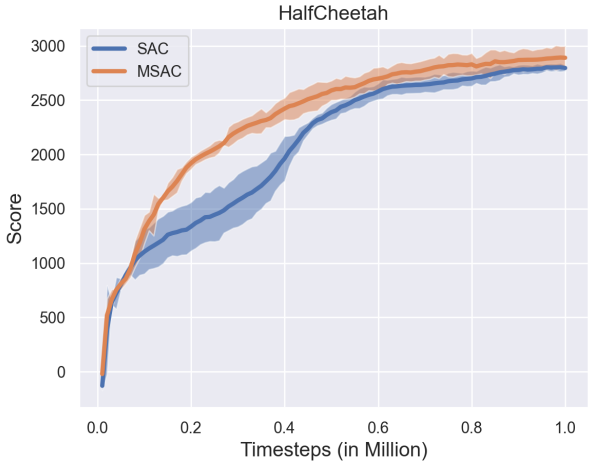


Fig. 2. Results for HalfCheetahPyBulletEnv-v0

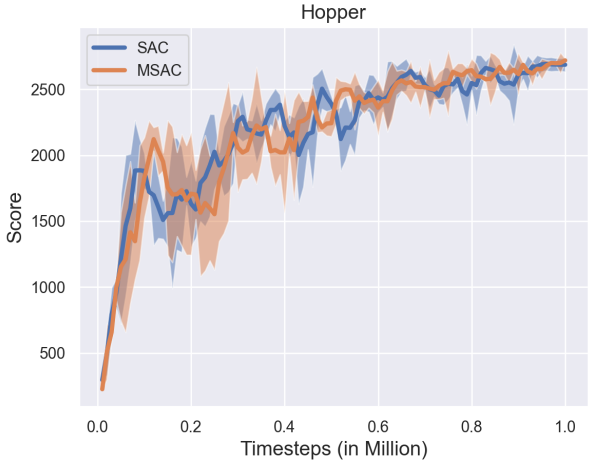


Fig. 3. Results for HopperPyBulletEnv-v0

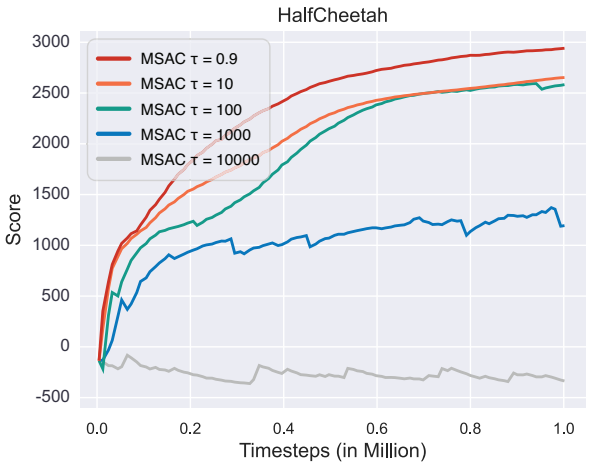


Fig. 4. Results for HalfCheetahPyBulletEnv-v0 with different Munchausen scaling factors

a deterministic policy at a very early stage. That the policy becomes more deterministic at high scaling factors was proven by a low entropy. The explanation here cannot be applied to the results in table II and III.

## IV. Next Steps

A central question to be clarified in the further project is how the clipping values should be chosen. In the discrete action space the probability for actions is in the range $[0, 1]$. The logarithmic probability is in the range $[-\infty, 0]$. In the continuous action space, however, the logarithmic probability is in the range $[-\infty, +\infty]$. In fact, our previous test achieved the best results when the clipping is still used in the range $[-1, 0]$. However, this does not sound reasonable, since the influence of the Munchausen term is in most cases $= 0$, since the logarithmic probabilities of the continuous actions are often $> 0$. A simple shift of the range to e.g. $[1, +\infty]$ has an effect on a similar range as the Munchausen term in the discrete action space but the influence in the Q function update is reinforcing by the positive values and not suppressing, like in the discrete action space, which is a fundamental difference.

## References

[1] Nino Vieillard, Olivier Pietquin and Matthieu Geist (2020). Munchausen Reinforcement Learning. arXiv:2007.14430.
[2] Tuomas Haarnoja et. al(2019). Soft Actor-Critic Algorithms and Applications. arXiv:1812.05905.
[3] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, Sergey Levine (2018). Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. arXiv:1801.01290.
[4] Make a four-legged creature walk forward as fast as possible. https://gym.openai.com/envs/Ant-v2/.
[5] Antonin Raffin et. al (2019). Stable Baselines3. GitHub repository, https://github.com/DLR-RM/stable-baselines3
[6] Antonin Raffin (2020). RL Baselines3 Zoo. GitHub repository, https://github.com/DLR-RM/rl-baselines3-zoo
[7] Benjamin Ellenberger (2018-2019). PyBullet Gymperium. GitHub repository, https://github.com/benelot/pybullet-gym
[8] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang and Wojciech Zaremba (2016). OpenAI Gym. arXiv:1606.01540.