# Munchausen Reinforcement Learning with Continuous Action Space - Final 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

***Abstract*—Munchausen reinforcement learning (RL) is an improvement of the Q-value updates which was introduced for discrete action spaces [1]. Since many real world applications require a continuous action space, the following work extends a Soft Actor-Critic (SAC) with the Munchhausen idea (M-SAC). The adapted algorithm was tested on common problems such as the four-legged agent of OpenAI Gym [4] and compared to currently existing baselines.**

***Index Terms*—munchausen, reinforcement learning, continuous action space, soft actor critic**

## I. Related Work

The presented idea of Munchausen RL is based on adding the probability of the policy to the reward [1]. This means if an action is likely according to the current policy, it will be additionally rewarded.

$$Q(s_t, a_t) = r_t + \tau[\alpha \ln \pi_\theta(a_t|s_t)]^0_{l_0} + \gamma \mathop{\mathbb{E}}_{s_{t+1}\sim p}[V(s_{t+1})] \quad (1)$$

To adapt the idea of Munchausen RL presented for discrete action spaces to continuous action spaces, the SAC implementation of Stable Baselines3 (SB3) [5] as well as the RL Baselines3 Zoo [6] training framework was used.

## II. Probability space adaptation

The central difference to continuous action spaces is in the representation of the action probabilities. For discrete actions, the probabilities are in the range $[0, 1]$. Hence, the Munchausen term (red in (1)) is 0 for a deterministic policy and negative otherwise. Figure 1a shows an exemplary discrete log probability distribution for the actions (orange) over the training steps. Overlaid is the absolute magnitude and region in which the Munchausen term acts (green). The lower clipping limit at $-1$ (introduced by the authors of the original paper for stability reasons) is shown as a dashed line.

In the continuous action space, the value of the probability density function (PDF) is in the range $[0, +\infty]$. A clipping at 0 would therefore only provide a Munchausen term $\neq 0$ for values of the PDF $< 1$. A possible adjustment which still avoids instability is e.g. a clipping between $[-1, 1]$. In the case of a more deterministic policy, the maximum value of the PDF increases towards $\infty$ while all other values approach 0. Consequently, the Munchausen term becomes a binary factor subject to the clipping limits. Therefore the clipping values need to be tuned to obtain a proper result. By introducing a mean normalization of the Munchausen term before using it to update the target Q-value function, good results have been obtained.

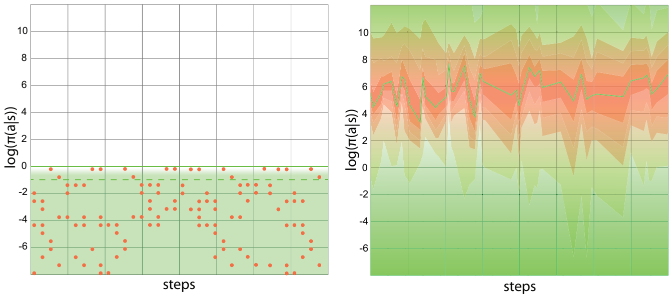
$$\overline{\pi}_\theta(a|s) = \pi_\theta(a|s) - \mathop{\mathbb{E}}_{\substack{a'\in A\\ s'\in S}}[\pi_\theta(a'|s')] \quad (2)$$

The mean normalization was implemented with an additional hyperparameter $\beta$. For $\beta = 0$ it holds independent of the policy that $\mathbb{E}_{s,a}[\ln \overline{\pi}_\theta(a|s) + \beta] = 0$. Consequently, also for a deterministic policy Munchausen term $= 0$ holds. This corresponds to the behavior in the original implementation.

The idea of mean normalization is shown in figure 1b. The example orange distribution represents the log PDF values of the actions plotted over the training steps. The central green line indicates the approximated expected value, which is subtracted for normalization. The green gradient again demonstrates the influence of the Munchausen term.

$$\text{Munchausen term} := \tau\alpha\,[\ln \overline{\pi}_\theta(a_t|s_t) + \beta] \quad (3)$$

$\tau$ is another hyperparameter which scales the Munchausen term. $\alpha$ is a dynamically adjusted entropy coefficient.



(a) Discreet actions (b) Continuous actions

Fig. 1: Munchausen probability space adaption

## III. State based Munchausen

An additional modification to the Munchausen idea is to sample new actions from the current policy based on the observations from the replay buffer.

$$\text{Munchausen term} := \tau\alpha \mathop{\mathbb{E}}_{\tilde{a}\sim\pi_\theta(\cdot|s)}[\ln \overline{\pi}_\theta(\tilde{a}|s_t) + \beta] \quad (4)$$

This leads to the effect that the Munchausen term no longer depends on the action, but only on the state. Therefore, states for which the policy is confident are additionally positively reinforced, while states for which the policy is uncertain are negatively rewarded. The hyperparameter $\beta$ can be interpreted as a balance between good and bad states. For example, with a positive $\beta$, states in which the policy is less certain than average are still rewarded positively.

## IV. Experiment

### A. experiments that have not produced positive results

*1) Original implementation of the Munchausen Paper:* As explained in section II, the original implementation is not adapted to the probability distribution in the continuous case and did not yield any improvements in practice. Especially the version with clipping had such a low impact that almost the same curves were achieved as without Munchausen.

*2) dynamicshift max / min:* To test the new hyperparameter $\beta$, an adaptive normalization was tried that mean normalized (as described in section 4) for a hyperparameter of 0. But for hyperparameters of -1 or 1, the minimum and maximum value, respectively, of the evaluated PDF was used. Thus, the hyperparameter could actually be used as a threshold of what proportion of the actions were positively or negatively rewarded. Due to the sampling of the actions, the minimum and maximum values are very unstable, which is why this attempt was not particularly successful.

*3) Scaling of the update values:* A scaling of the reward, the Munchausen parameter $\tau$ or the entropy coefficient $\alpha$ results in different weights of the individual terms in the Q-Value function update. However, no general improvement could be found.

### B. Experiments with positive results

Using our introduced modification of section III with tuned $\beta$ for every environment, we achieved notable results. In each of them we compared SAC (from [6]) with Munchausen SAC state based (M-SAC$_s$) and present the respective mean reward averaged over three seeds each. Table I lists key parameters.

| 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 |
| $\beta$ | {-10, -1, 0, 10} |

TABLE I: Parameters used for Munchausen RL agents

## V. Results

The results of the standard SAC agent which are used as reverence roughly match the result stated by the SB3 baselines. Our M-SAC$_s$ implementation significantly outperforms the baseline in two environments (see table II). The lower mean reward in the last evaluation of HopperPyBulletEnv-v0 is a bit misleading. In fact, both algorithms perform very similar looking at figure 4.

The following plots show the mean reward achieved by the respective agent in the respective environment during evaluation averaged over three seeds.

| Environments | SAC | M-SAC$_s$ | Improvement |
|---|---|---|---|
| AntPyBulletEnv-v0 | 3550 +/- 97 | **3625 +/- 116** | +2 % |
| HalfCheetahPyBulletEnv-v0 | 2796 +/- 27 | **3301 +/- 188** | +18 % |
| HopperPyBulletEnv-v0 | **2686 +/- 51** | 2551 +/- 102 | -5 % |
| Walker2DPyBulletEnv-v0 | 2154 +/- 131 | **2548 +/- 15** | +18 % |

TABLE II: (Deterministic) mean evaluation reward, state based

| Environments | SAC | M-SAC$_a$ | Improvement |
|---|---|---|---|
| AntPyBulletEnv-v0 | **3550 +/- 97** | 3466 +/- 10 | -2 % |
| HalfCheetahPyBulletEnv-v0 | 2796 +/- 27 | **3063 +/- 216** | +10 % |
| HopperPyBulletEnv-v0 | **2686 +/- 51** | 2648 +/- 11 | -1 % |
| Walker2DPyBulletEnv-v0 | 2154 +/- 131 | **2411 +/- 32** | +12 % |

TABLE III: (Deterministic) mean evaluation reward, action based

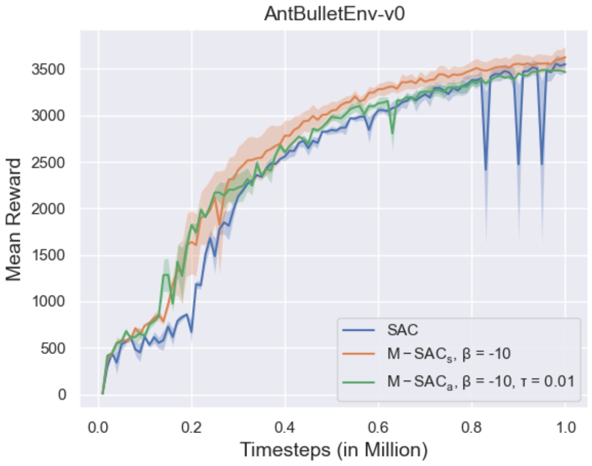


Fig. 2: Results for AntBulletEnv-v0

In addition to higher mean rewards, our idea can demonstrate higher robustness for long trainings which can be observed in the last fifth of training in AntPyBulletEnv-v0 (figure 2) and in MountainCarContinuous-v0 (figure 6) in particular. Somehow SAC forgets already acquired skills some time after convergence in this environment.

Furthermore, our expectation was that by including the Munchausen term in the update, we would obtain a more deterministic policy during training. In order to prove the assumption, we checked the standard deviation since a lower standard

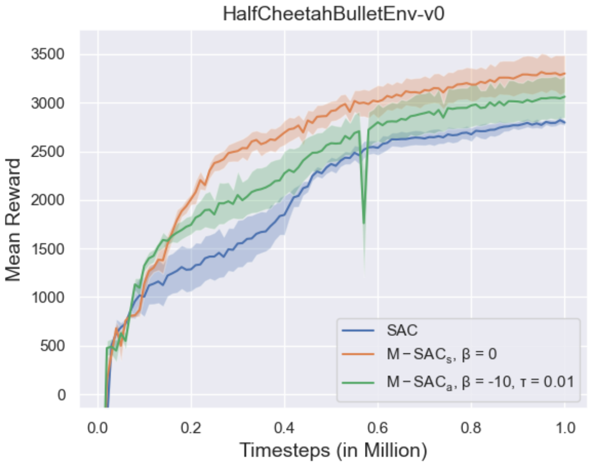


Fig. 3: Results for HalfCheetahPyBulletEnv-v0

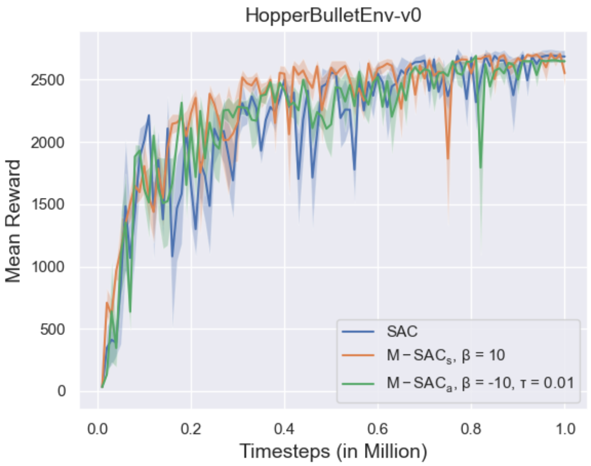


Fig. 4: Results for HopperBulletEnv-v0

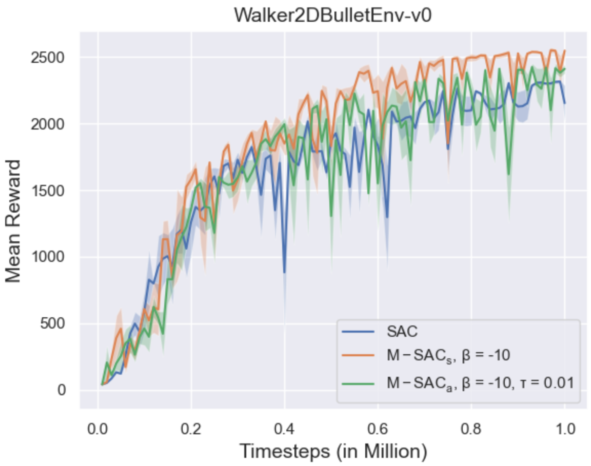


Fig. 5: Results for Walker2DBulletEnv-v0

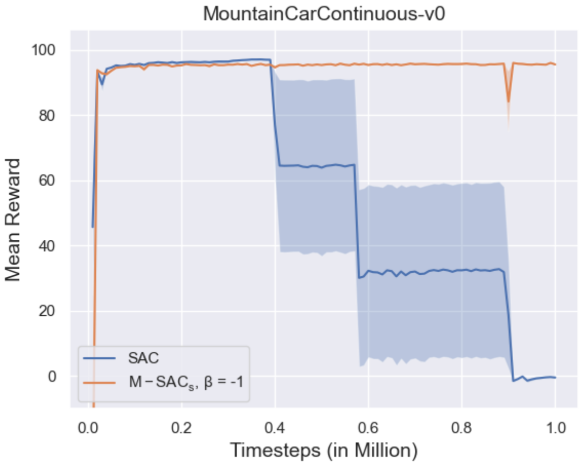


Fig. 6: Results for MountainCarContinuous-v0

deviation would mean the policy is more deterministic and vice versa.

It turned out that Munchausen does not make the policy significantly more deterministic in general, but it does tend to become more deterministic for environments where our algorithm performs well or better than SAC. We observed the most significant result in HalfCheetahPyBulletEnv-v0 (see figure 7) where the standard deviation of M-SAC is indeed lower than in our baselines SAC.

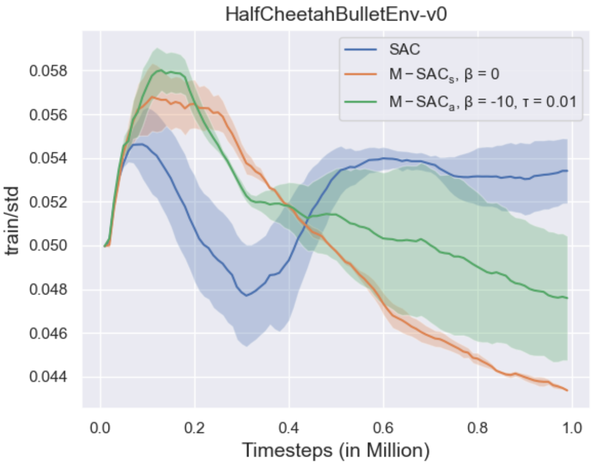


Fig. 7: Standard Deviation in HalfCheetahPyBulletEnv-v0

## References

[1] Nino Vieillard, Olivier Pietquin and Matthieu Geist (2020). Munchausen Reinforcement Learning. arXiv:2007.14430.

[2] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel and Sergey Levine (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, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto and Noah Dormann (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.