An Implementation Guide and Empirical Analysis on the Soft Actor-Critic Algorithm

Che Wang
cw1681

Training for a few million timesteps

- Why study Soft Actor-Critic (SAC)?

  Project Contributions:
- An easy-to-learn Pytorch implementation of SAC
- Additional analysis on what are the most important components for SAC
- Enhancing SAC with recent advances in DL

Why study SAC? Because it's recent, it's powerful and it's robust.

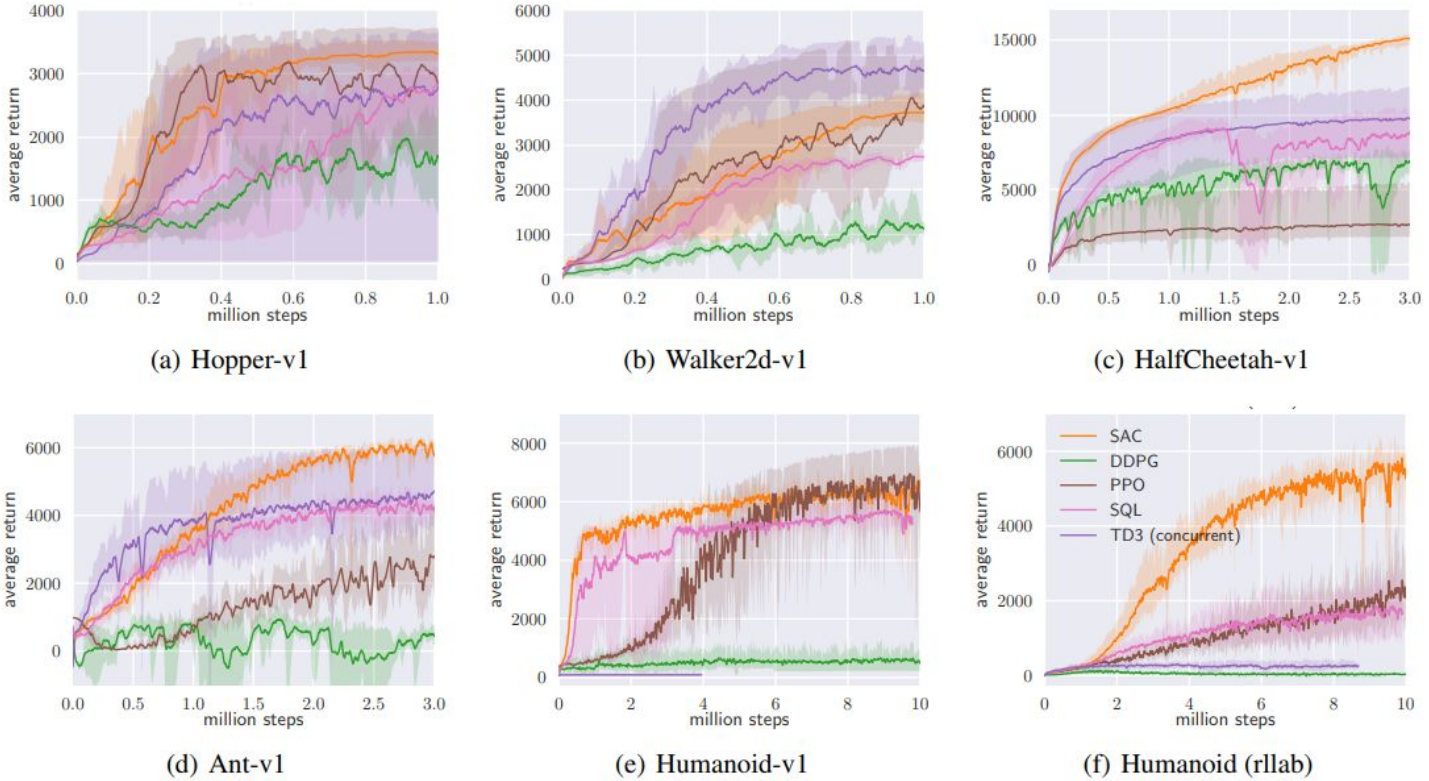


Figure 1. Training curves on continuous control benchmarks. Soft actor-critic (yellow) performs consistently across all tasks and outperforming both on-policy and off-policy methods in the most challenging tasks.

Source: SAC paper at https://arxiv.org/pdf/1801.01290.pdf

SAC core elements: entropy and 5 networks

An entropy term, related to exploration

$$H(P) = \underset{x\sim P}{\mathrm{E}}\left[-\log P(x)\right].$$

A policy network, maps states to action, learns to take action with high q-value and high entropy

$$\pi^* = \arg\max_{\pi} \underset{\tau\sim\pi}{\mathrm{E}}\left[\sum_{t=0}^{\infty}\gamma^t\Big(R(s_t,a_t,s_{t+1}) + \alpha H\left(\pi(\cdot|s_t)\right)\Big)\right],$$

A value network and a target value network, learn the value of a state

$$V^{\pi}(s) = \underset{\tau\sim\pi}{\mathrm{E}}\left[\sum_{t=0}^{\infty}\gamma^t\Big(R(s_t,a_t,s_{t+1}) + \alpha H\left(\pi(\cdot|s_t)\right)\Big)\,\middle|\, s_0 = s\right]$$

2 q-value networks, learn the value of a state-action pair

$$Q^{\pi}(s,a) = \underset{\tau\sim\pi}{\mathrm{E}}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t,s_{t+1}) + \alpha\sum_{t=1}^{\infty}\gamma^t H\left(\pi(\cdot|s_t)\right)\,\middle|\, s_0 = s, a_0 = a\right]$$

Source: OpenAI spinup

SAC is an off-policy algorithm and has a replay buffer
SAC can work with continuous and discrete (with modification) environments

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

Project part One

A Beginner-Friendly Pytorch Implementation of SAC, with focus to the following key points:

a. Consistent with OpenAI Spinup documentation: help our readers to easily get additional references for their study.
b. A concise and minimal implementation. as simple as possible, without redundant structures.
c. Extensive documentation: write comments whenever needed, to make sure every line of the code is easy to understand.
d. Bug-free and Comparable performance: equivalent performance compared to the results in the SAC paper.

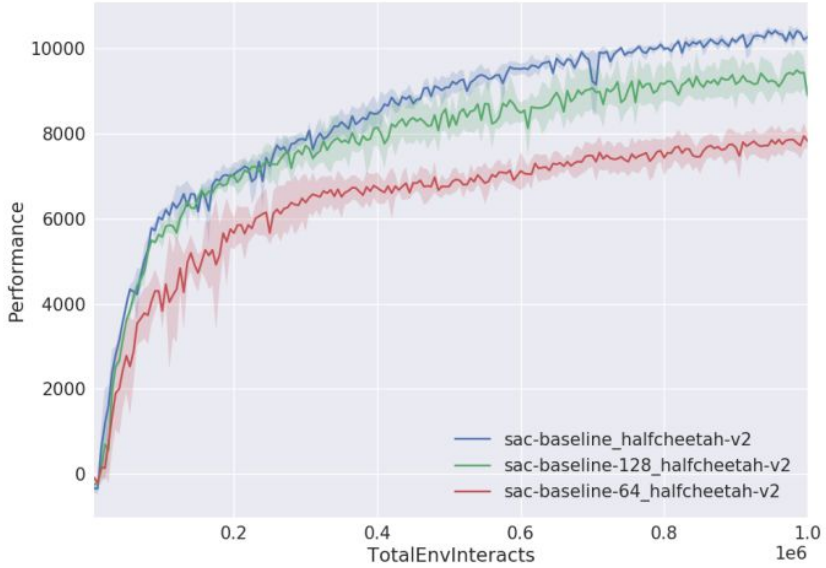
https://spinningup.openai.com
https://github.com/watchernyu/spinningup
https://github.com/watchernyu/hpc_setup

Project part Two Additional Analysis on SAC

Additional analysis on what are the most important components for SAC (not presented in the SAC paper)

a. SAC uses a relatively large hidden layer size, compared to other papers, is this a critical thing?
b. SAC uses a huge replay buffer that can take 1e6 data, can the buffer size be modified for improvement?

Testing environment: HalfCheetah and Ant Mujoco Environment
Each experiment takes ~20 hours to run.



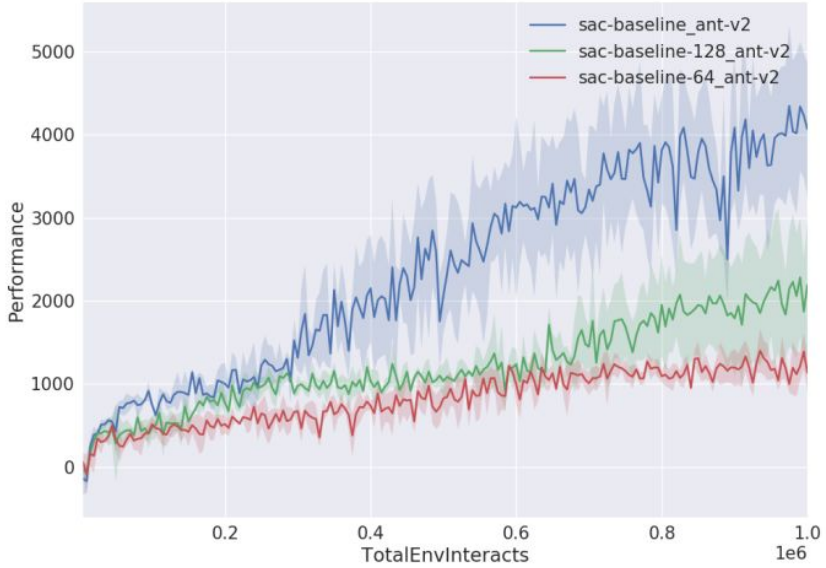
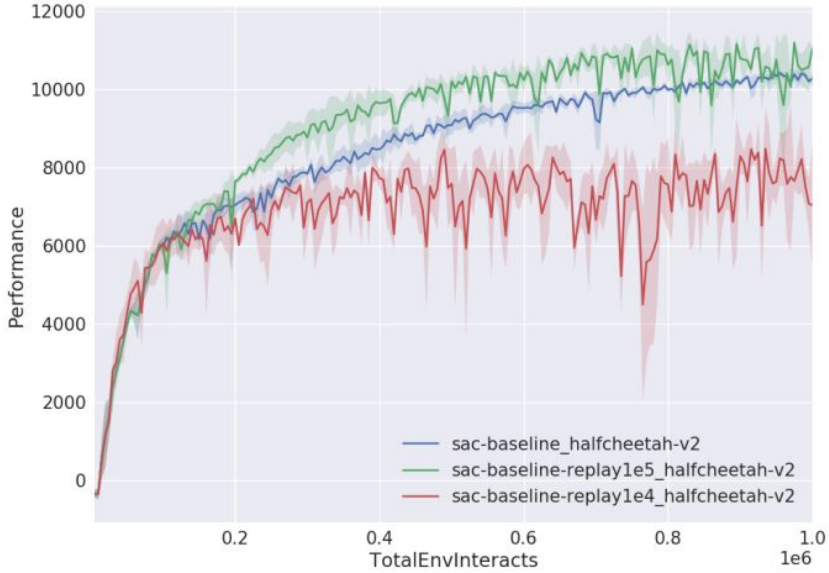


Figure 1: SAC performance comparison on HalfCheetah and Ant Mujoco environments, on different hidden sizes.



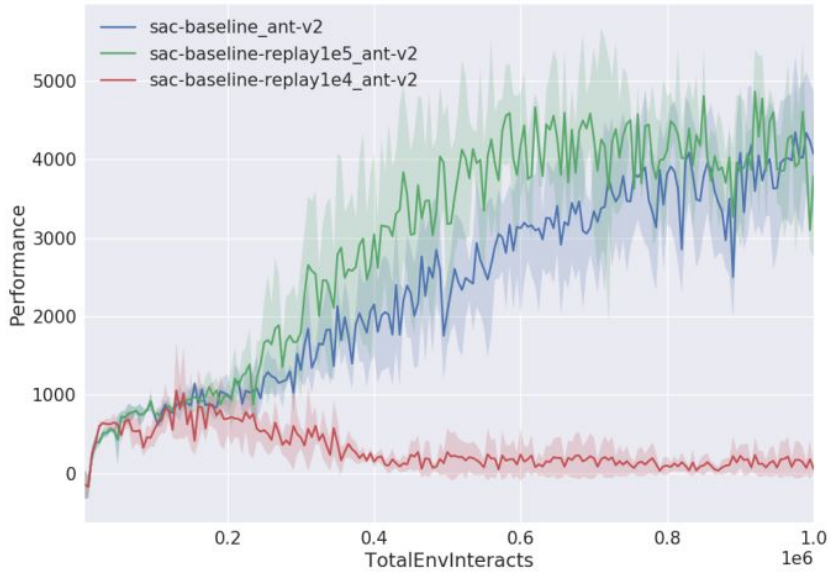


Figure 2: SAC performance comparison on HalfCheetah and Ant Mujoco environments, on different replay buffer sizes.

Conclusions:

- Relatively large networks (compared to other recent DRL works such as PPO) are critical to the performance of SAC, but this also means SAC can be slower in learning from the same amount of data. Hidden size of 256 is approximately 3x slower than a hidden size of 64, which is PPO's default size.
- SAC's replay buffer can be designed in more sophisticated fashion to count for data of different quality and improve performance.

Project part Three Enhancing SAC

Enhancing SAC with recent advances in DL

a. AMSGRAD optimizer in On the Convergence of Adam and Beyond (Reddi2018)
b. Use selu activation unit in Self-Normalizing Neural Networks (Klambauer2017)
c. Adding remaining time to agent observation, in Time Limits in Reinforcement Learning (Pardo2017)

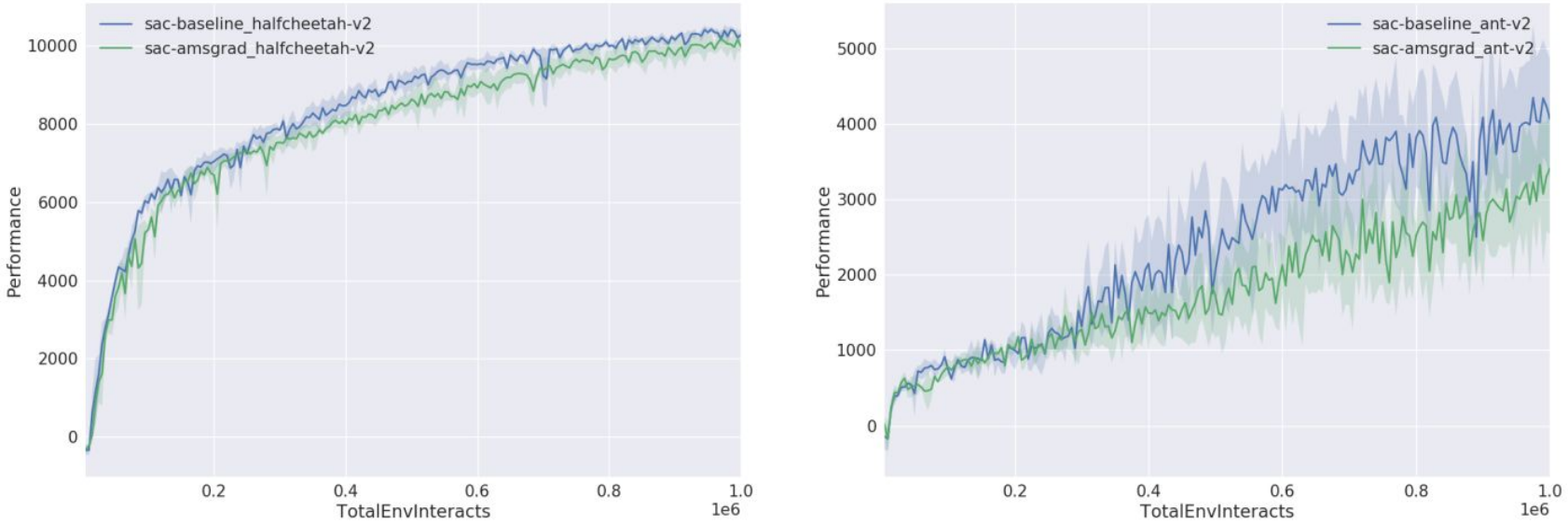


Figure 3: SAC performance comparison on HalfCheetah and Ant Mujoco environments, with Adam and AMSGrad optimizer.

On the convergence of adam and beyond by Reddi et al. (2018) proves that there is a flaw in the convergence of Adam and proposed AMSGrad which has better proven convergence property and empirically make training much faster in an array of supervised learning tasks.

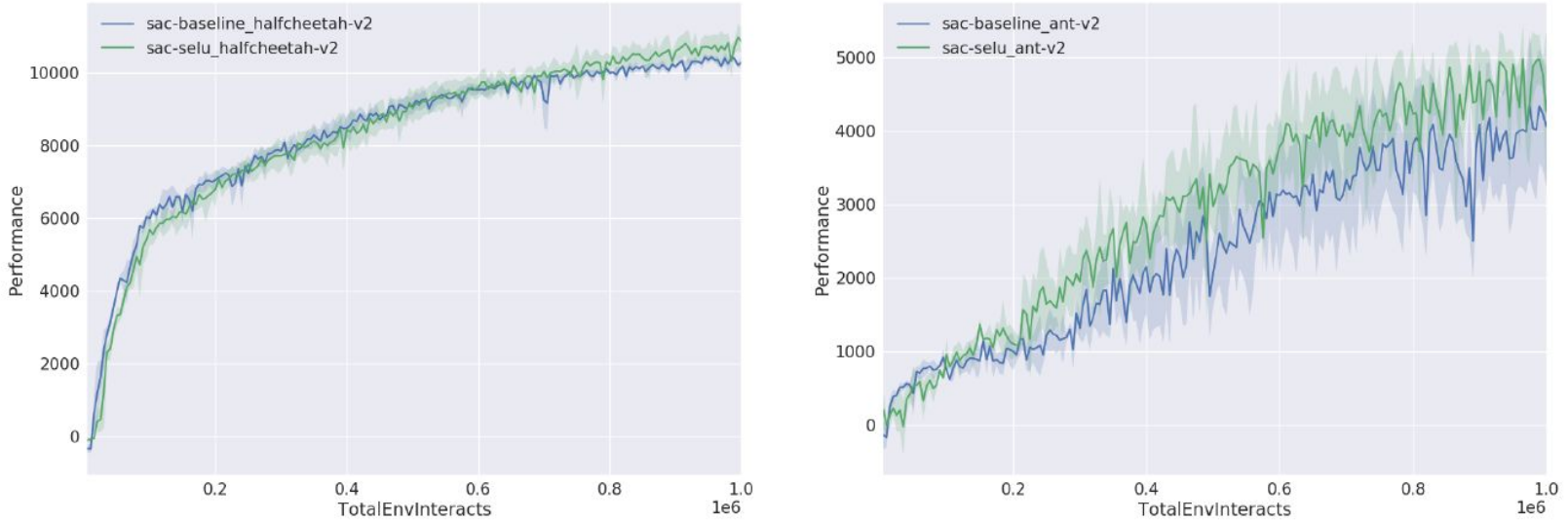


Figure 4: SAC performance comparison on HalfCheetah and Ant Mujoco environments, with relu activation and selu activation, selu activation also comes with a specific type of network weight initialization, details see [16]

Self-Normalizing Neural Networks by Klambauer et al. (2017) proposed selu activation which can automatically normalize output activation after each layer, potentially making training much more stable

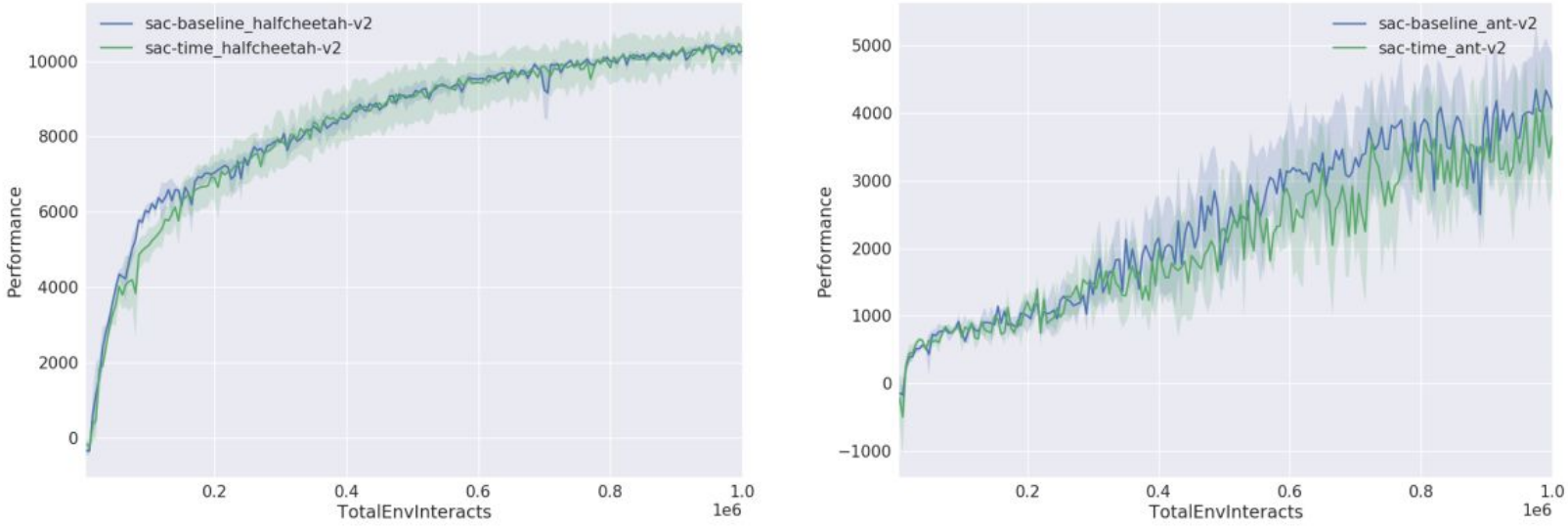


Figure 5: SAC performance comparison on HalfCheetah and Ant Mujoco environments, with baseline and time-aware enhancement.

Time Limits in Reinforcement Learning by Pardo et al. (2017) propose to add remaining environment time as a feature to the observation of RL agent to help them develop time-dependent behaviors that can result in higher return.

Conclusions:

- While AMSgrad makes learning a lot more effective in supervised learning tasks, it seems that it doesn’t work so well with SAC, results show that replacing Adam with AMSgrad decreases performance. More research is needed to see if its usage can be modified to fit the DRL environment (or other DRL algorithms).
- Selu in SAC significantly improve data efficiency in the more difficult Ant task. At the cost of 50% more wall clock time. Doesn’t make a difference on HalfCheetah.
- Being time-aware doesn’t improve performance significantly. It could be the environment termination flag in the SAC code makes time feature redundant.

A tutorial and empirical analysis on Soft Actor-Critic: a state-of-the-art DRL algorithm

- An easy-to-learn Pytorch implementation of SAC
- Additional analysis on what are the most important components for SAC
- Enhancing SAC with recent advances in DL
- (2 out of 5 modifications resulted in better performance than original paper)

Thank You!

Questions are welcome!

References

[1] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.

[2] F. Pardo, A. Tavakoli, V. Levdik, and P. Kormushev, "Time limits in reinforcement learning," *CoRR*, vol. abs/1712.00378, 2017.

[3] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in Neural Information Processing Systems*, pp. 971–980, 2017.

[4] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," 2018.

[5] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033, IEEE, 2012.

[6] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[7] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[9] Z. Huang, S. Zhou, B. Zhuang, and X. Zhou, "Learning to run with actor-critic ensemble," *arXiv preprint arXiv:1712.08987*, 2017.

[10] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.