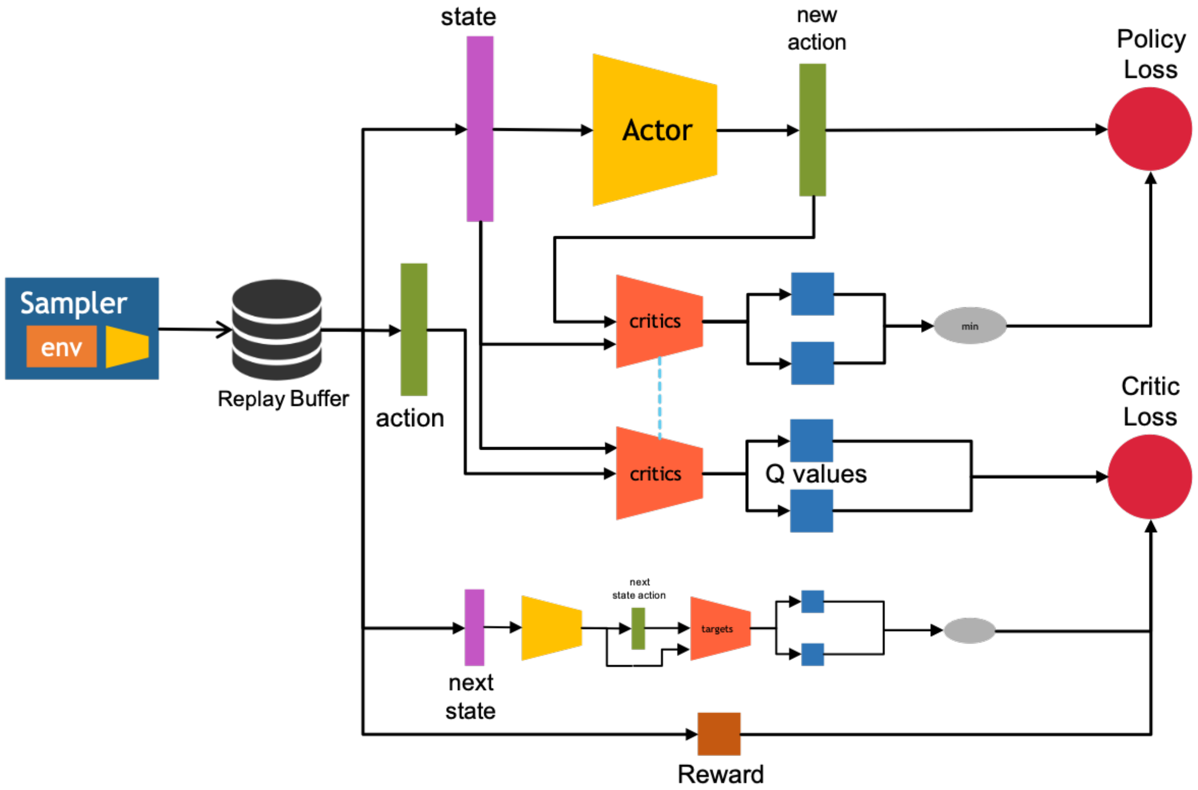
RL Final Project
SAC vs TD3

Team Kexbot

Sebastian Koch (SAC) and Onno Eberhard (TD3)

Soft Actor-Critic (SAC)



- Off-Policy Algorithm
- Stochastic policy with entropy regularization
- Entropy coefficient controls Exploration/Exploitation tradeoff

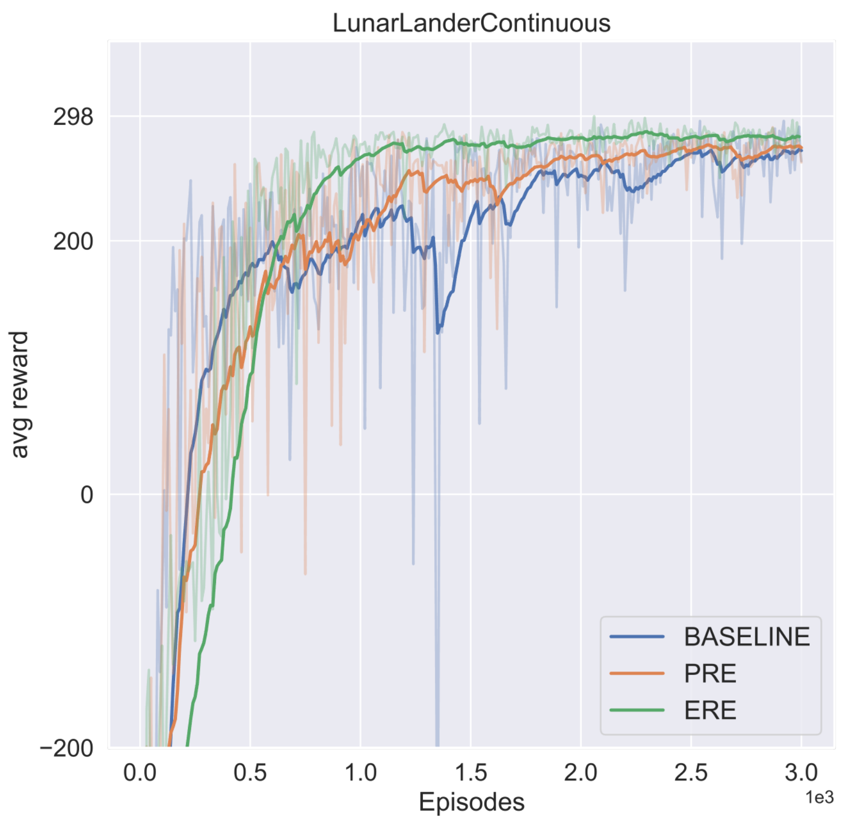
$$\underbrace{\pi^*}_{\text{optimal policy}} = \arg\max_{\pi} \mathop{\mathrm{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t\left(\underbrace{R(s_t, a_t, s_{t+1})}_{\text{Reward}} + \underbrace{\alpha H\left(\pi(\cdot|s_t)\right)}_{\text{Entropy}}\right)\right]$$

Prioritized Experienced Replay with Emphasizing Recent Experiences

- Normal Replay Buffer samples random
- <u>PRE:</u>
- Idea: Sample important states more often
  - Assign probability to the samples depending on their TD-Error

- <u>ERE:</u>
- Idea: Sample recent states more often
  - Calculate how many old samples are important

- Weightening the Loss needed as a bias is introduced



Training with SAC

Train Shooting

Train Defending

Train against Basic Strong Opponent

Train Self-Play

- Train agent against itself
- One agent learns, the other exploits its policy
- If the trained agent has an avg. score above a certain threshold, update to opponent
- Repeat

Twin Delayed DDPG (TD3)

- Off-Policy, tries to improve DDPG
- <u>Clipped Double-Q Learning</u> to help overcome overestimation of $Q(s, a)$:

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_{i,\text{targ}}}(s', a'(s')),$$

- <u>Target Policy Smoothing</u>: Add noise to target actions to prevent exploitation of errors in Q-functions

$$a'(s') = \text{clip}\left(\mu_{\theta_{\text{targ}}}(s') + \text{clip}(\epsilon, -c, c), a_{Low}, a_{High}\right), \qquad \epsilon \sim \mathcal{N}(0, \sigma)$$

- <u>Delayed Policy Updates</u>: Update the Q-functions more frequently than the policy network.

Policy Update Clipping

Algorithm 1 Twin Delayed DDPG

Input: initial policy parameters θ, Q-function parameters ϕ_1, ϕ_2, empty replay buffer $\mathcal{D}$
Set target parameters equal to main parameters $\theta_{\text{targ}} \leftarrow \theta$, $\phi_{\text{targ},1} \leftarrow \phi_1$, $\phi_{\text{targ},2} \leftarrow \phi_2$
repeat
  Observe state s and select action $a = \text{clip}(\mu_\theta(s) + \epsilon, a_{Low}, a_{High})$, where $\epsilon \sim \mathcal{N}$
  Execute a in the environment
  Observe next state s', reward r, and done signal d to indicate whether s' is terminal
  Store (s, a, r, s', d) in replay buffer $\mathcal{D}$
  If s' is terminal, reset environment state.
  if it's time to update **then**
    for j in range(however many updates) **do**
      Randomly sample a batch of transitions, $B = \{(s, a, r, s', d)\}$ from $\mathcal{D}$
      Compute target actions

$$a'(s') = \text{clip}\left(\mu_{\theta_{\text{targ}}}(s') + \text{clip}(\epsilon, -c, c), a_{Low}, a_{High}\right), \qquad \epsilon \sim \mathcal{N}(0, \sigma)$$

      Compute targets

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', a'(s'))$$

      Update Q-functions by one step of gradient descent using

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} \left(Q_{\phi_i}(s, a) - y(r, s', d)\right)^2 \qquad \text{for } i = 1, 2$$

      if j mod `policy_delay` $= 0$ **then**
        Update policy by one step of gradient ascent using

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} Q_{\phi_1}(s, \mu_\theta(s))$$

        Update target networks with

$$\phi_{\text{targ},i} \leftarrow \rho \phi_{\text{targ},i} + (1 - \rho)\phi_i \qquad \text{for } i = 1, 2$$
$$\theta_{\text{targ}} \leftarrow \rho \theta_{\text{targ}} + (1 - \rho)\theta$$

      end if
    end for
  end if
until convergence

- Action is clipped to valid range everywhere except in policy update
- Why not there as well?
- Might help with NaNs

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} Q_{\phi_1}(s, \mu_\theta(s))$$

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} Q_{\phi_1}(s, \text{clip}(\mu_\theta(s), a_{\text{low}}, a_{\text{high}}))$$

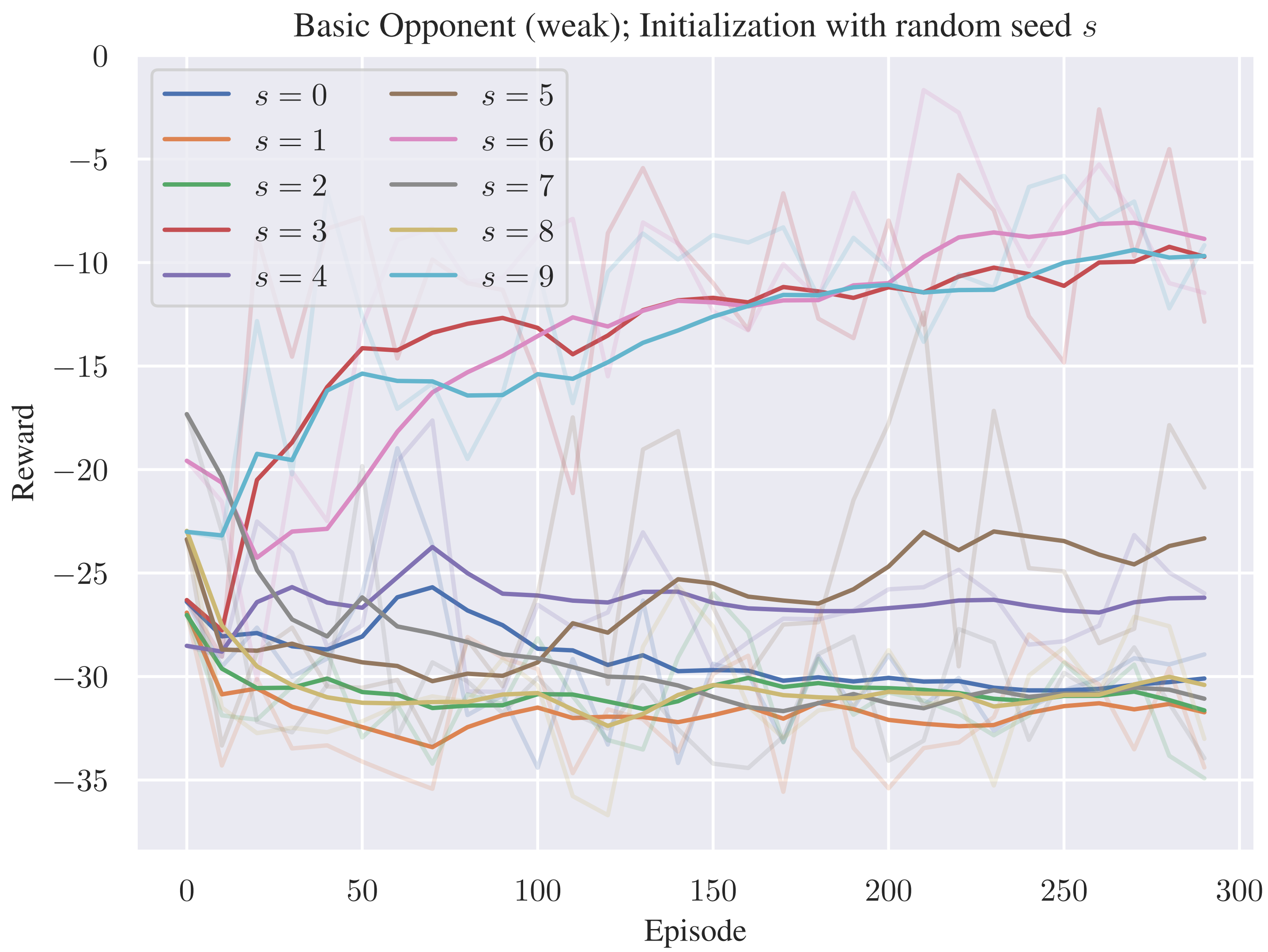
Solving Lunar Lander

Solving Hockey

(Seed: The worst hyperparameter.™)

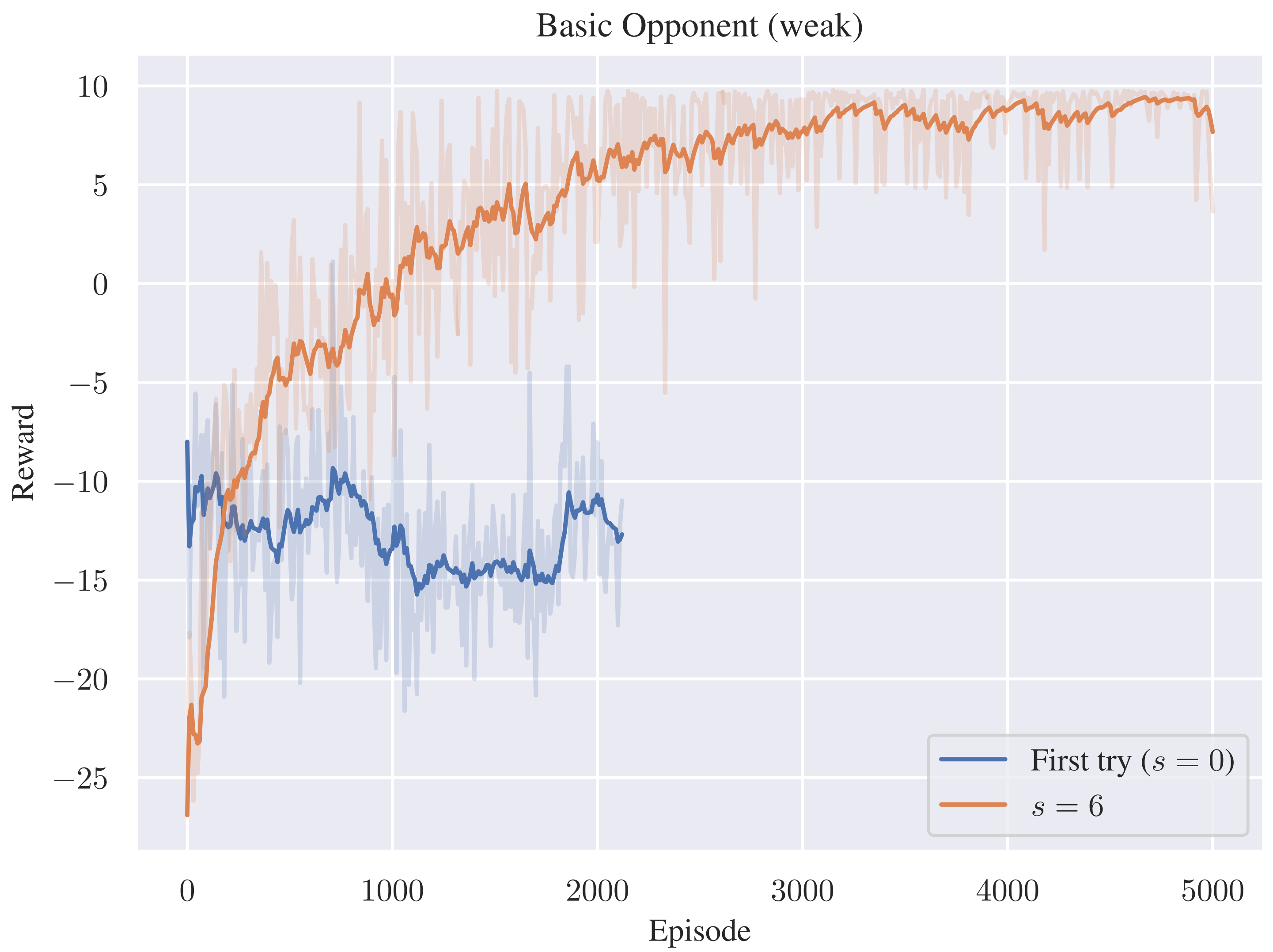
Solving Hockey

(Seed: The worst hyperparameter.™)



Solving Hockey

(Seed: The worst hyperparameter.™)



Thank you for listening!