RL-Course 2024/25: Final Project Report

Bayesed Sigma-Algebros: Finn Springorum

February 26, 2025

1 Introduction

The field of reinforcement learning (RL) has a multitude of applications, one of which is the creation of human-like game agents that exhibit intelligent behavior and challenge the player's tactical mind. In this study, we trained RL agents to play Laser Hockey, a custom game environment built based on the Gymnasium Python library [5]. It consists of two players, shaped like the letter D, that compete against each other. The objective is to shoot the puck into the opponent's goal while adhering to specific rules. Players are confined to their respective pitch halves and can attach the ball to their straight edge for a brief period before shooting in the direction that corresponds to their current rotation. Despite its conceptual simplicity, the game's strategic intricacies, which balance offensive and defensive maneuvers while adapting to diverse opponents of both algorithmic and human nature, are noteworthy. Consequently, RL may yield strategies that surpass human intuition and succeed against more straightforward approaches. To address this challenge, we implemented the model-free and off-policy Soft Actor-Critic (SAC) [3] algorithm and extended it with Prioritized Experience Replay (PER) [4].

2 Methods

2.1 Soft Actor-Critic

Soft Actor-Critic (SAC), as proposed by Haarnoja et al. [3], is a model-free off-policy actor-critic deep RL algorithm based on the maximum entropy RL framework. Its primary characteristic is the stochastic actor (policy) that aims to maximize the expected reward while also maximizing the entropy, thus pursuing rewarding strategies while acting as randomly as possible. The SAC was designed for continuous state and action spaces and demonstrated superior performance over prior off-policy methods like the actor-critic Deep Deterministic Policy Gradient (DDPG). SAC offers sample-efficient learning and stability, improved scalability and efficiency, and outperformed conventional RL methods on challenging tasks. Given these properties, it was hypothesized that the SAC would be a highly suitable approach for the Laser Hockey objective.

For a Markov decision process $(\mathcal{S}, \mathcal{A}, p, r)$ with continuous state space $\mathcal{S}$ and action space $\mathcal{A}$, $p_\pi(s_t)$ and $p_\pi(s_t, a_t)$ denote the state and state-action marginals of the trajectory distribution induced by a policy $\pi(a_t|s_t)$. While the standard RL objective considers the expected sum of rewards, the maximum entropy objective extends it by an entropy term:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim p_\pi} \left[r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t)) \right].$$

This formula indicates that the actor aims to increase its entropy as long as the reward is not significantly decreased by this stochastic behavior. Therefore, the temperature parameter α is critical, as it determines the relative importance of the entropy against the reward and, thus, the balance between exploration and exploitation. This objective can be highly beneficial for many tasks, including Laser Hockey, as the policy is incentivized to explore more widely while discarding clearly unpromising avenues. Moreover, it can allocate equal probability mass to actions that seem equally attractive, which is a significant advantage over deterministic policies.

2.2 Implementation

The SAC code was based on a lightweight PyTorch implementation [2] that closely follows the original paper [3] and implements many tuning options. For the normal and target critic, we employed Q-networks that shared the same architecture but served different purposes. Each Q-network had two parallel architectures (Q1 and Q2) for evaluating state-action pairs. The state (18-dimensional) and action (4-dimensional) inputs were concatenated and passed through two hidden layers of 256 units each, followed by an output layer that produced a single Q-value. The use of the minimum of these two Q-functions is known to mitigate positive bias in the policy improvement step. The normal critic was used for training, while the target critic provided stable target values. The target critic was continuously soft-updated using an exponentially moving average with smoothing constant $\tau = 0.005$. For the actor, we used a Gaussian policy, parameterized by the mean and log standard deviation of a Gaussian distribution, which were outputs of a neural network. The network took the 18-dimensional state input, passed it through two hidden layers with 256 units each, and produced a 4-dimensional mean vector and a 4-dimensional log standard deviation vector as outputs. This allowed for a continuous range of actions to be sampled from the policy. In evaluation mode, the mean of the distribution was returned instead. The temperature α was either fixed or automatically tuned during training based on the loss to the target entropy $\mathcal{H}_{\mathcal{T}} = -\dim(A) = -4$ for our action space A. We always used Adam optimizers with a fixed learning rate of 3e-4. The Q-function parameters were trained to minimize the soft Bellman residual

$$L(Q) = \mathbb{E}_{(s_t,a_t)\sim D}\left[\frac{1}{2}\left(Q(s_t,a_t) - r(s_t,a_t) - \gamma \mathbb{E}_{s_{t+1}\sim p}\left[V(s_{t+1})\right]\right)^2\right]$$

based on a batch of transitions sampled from the replay buffer D. The policy was trained to minimize

$$L(\pi) = \mathbb{E}_{s_t\sim D,\ \epsilon_t\sim N}\left[\alpha \cdot \log \pi(f(\epsilon_t; s_t)|s_t) - Q\left(s_t, f(\epsilon_t; s_t)\right)\right].$$

Here, it was reparameterized using a neural network transformation $a_t = f(\epsilon_t; s_t) = \mu_t + \sigma_t \cdot \epsilon_t$, where ϵ_t is sampled from a standard normal distribution, and μ_t is the mean and σ_t the standard deviation output of the policy's neural network, allowing the backpropagation of gradients through the stochastic process.

For the replay buffer, instead of uniformly sampling experiences, we implemented Prioritized Experience Replay (PER) [4] to improve learning efficiency. PER prioritizes stored experiences that were surprising and led to better learning in the past, as indicated by the temporal-difference (TD) error for the Q-function updates. The probability of sampling a transition from the buffer was given by $P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$, where $p_i > 0$ is the priority (TD error) of transition i, and $\alpha = 0.1$ corresponds to the amount of prioritization. The PER changes the memory's data distribution and thus introduces a bias, which was reduced by using importance-sampling weights that were folded into the loss for the Q-learning update. We utilized the SumTree and buffer implementation from [1] and adjusted our existing code to incorporate the importance weighting, temporal-difference errors, and priority updates.

Furthermore, we introduced heuristic changes to the reward function of the Hockey environment, as we observed agents that were anxious of losing games and thus preferred to be inactive when they were

supposed to start the game by touching the puck first. A reward of $+3$ was allotted for the initial touch, with subsequent touches subject to an exponentially decaying reward function (0.99^t). This design aimed to motivate active participation and facilitate the acquisition of effective shooting skills, while ensuring that the primary objective of securing victory in games remained paramount. The rewards for winning ($+10$), drawing (0), and losing (-10) remained unaltered.

2.3 Training

As predicted and supported by manual evaluation of our first trained agents, overfitting was the primary issue during the training phase, as there were only two algorithmic basic opponents (weak and strong), whose strategies were easily exploited by our trained RL agents. To obtain agents with good generalization properties that still perform well against the basic opponents, we introduced partial self-play by utilizing previously saved training checkpoints of the same trained agent as opponents. Specifically, the agent parameters were saved every $1{,}000$ episodes, and for each self-play phase, an earlier version was selected based on an exponential weighting technique with low decay (0.25) to prioritize more recent and thus more challenging checkpoints over older ones. We trained for $15{,}000$ episodes in three phases:

(I) $4{,}000$ episodes against the weak and strong basic opponents, alternating between them every 500 episodes. During this phase, the agent was intended to learn basic gameplay.

(II) $7{,}000$ episodes using a circular system comprising 500 episodes of self-play, 200 episodes against the strong basic opponent, another 500 episodes of self-play, and 200 episodes against the weak basic opponent. This phase aimed to teach the agent to adapt to different opponents while maintaining proficiency against the basic opponents.

(III) $4{,}000$ episodes using a circular system consisting of three sets of 200 episodes of self-play, followed by 100 episodes against the strong basic opponent and 100 episodes against the weak basic opponent. This phase aimed to stabilize the agent through frequent opponent switches.

We used a batch size of 128, a replay memory size of $1{,}000{,}000$, and a discount factor γ of 0.99. We sampled random actions for the first $10{,}000$ training steps, and set the maximum time steps per episode to $1{,}000$, based on our observations in simulations and human play. In total, nine agents were trained by altering one hyperparameter at a time. We focused on the entropy temperature α, as we predicted it to be the most significant one, and trained with four different fixed settings (0, 0.05, 0.2, 0.5). For the other five agents, we enabled automatic entropy tuning, as this was anticipated to be the best-performing version among the temperature variations, and trained agents with reduced learning rate (3e-5), reduced decay (0.9), disabled PER, and hard target critic updates (every 100 parameter updates).

2.4 Evaluation

Every 500 episodes during training, we simulated 25 games against both the weak and strong basic opponents using the current agent in evaluate mode, and saved the statistics. For the final agents, we collected their versions after each of the three training phases and ran a competition between these 27 agents by having them play 520 games against the weak and strong basic opponents, respectively, and 20 games against each other for each pairing. This final evaluation was conducted with a maximum of $2{,}000$ steps per episode. Since a draw and a loss should not be rewarded equally, we used the common scoring system in sports (3 points for a win, 1 for a draw, 0 for a loss) for both training and final evaluation. Our code available on GitHub comprises all mentioned features.

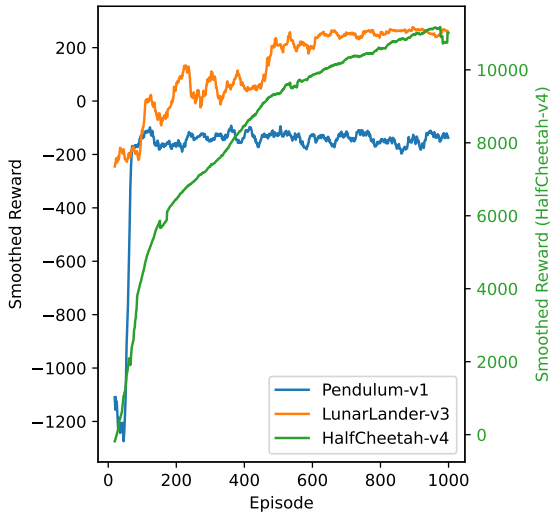

Figure 1: Smoothed reward over 1,000 training episodes of the SAC with PER for three standard Gymnasium environments.

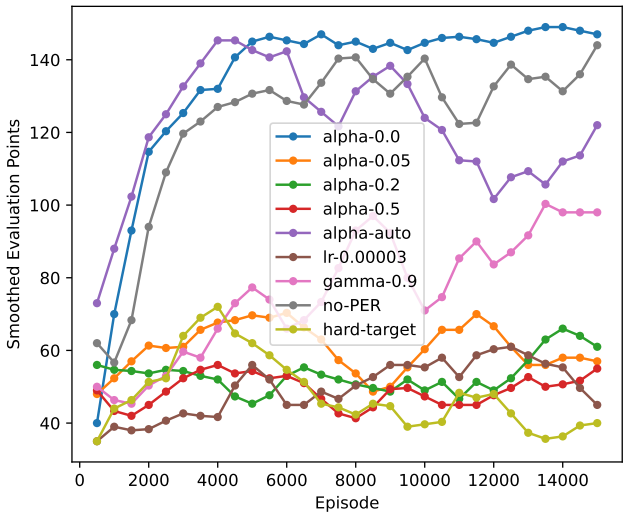

Figure 2: Smoothed evaluation results during the training of the nine Laser Hockey agents, based on 50 games against basic opponents every 500 episodes.

3 Results

Before transferring the SAC and PER implementation to the Hockey environment, we tested its functionality by training SAC models with $\alpha = 0.2$ on three standard Gymnasium [5] environments, verifying that the reward was really maximized and matched prior results obtained with different RL algorithms. As demonstrated by Fig. 1, this was the case for the selected environments.

Due to our divergent training strategy, we measured training performance in terms of points from the 50 evaluation games played every 500 episodes to observe how well the agents perform against the two baseline opponents (Fig. 2). It should be noted that the first 4,000 episodes were solely trained based on the baseline opponents, before the training abruptly switched to self-play. Interestingly, only the performance of the alpha-0.0 agent remained relatively unaffected by this switch, while the other agents exhibited fluctuations in baseline performance during the second and third training phases. Moreover, five of our nine agents failed to significantly improve their baseline performance over time, which resulted in their poor scores in the subsequent final evaluation.

The final evaluation (Tab. 1) revealed significant deviations in performance among the different trained agents. Even the weak and strong basic opponents could not be beaten confidently by all agents, as indicated by the low scores primarily resulting from draws. The agent with disabled entropy (alpha-0.0) demonstrated superior performance against both basic and trained opponents, while the other agents with fixed α failed to learn effective strategies. For the automatically tuned α, only the default and disabled PER settings produced successful agents. We further examined the most promising agents by playing against them, enabling us to identify a clear winner (alpha-0.0_15000), which performed exceptionally well against all kinds of opponents. This agent demonstrated remarkable Hockey abilities, characterized by precise shots based on our movement and position, a balanced offensive and defensive strategy, and the ability to predict our movements and shots. However, similar to the other agents, it struggled with uncommon situations, such as the puck bouncing vertically between the walls. The winning agent and its successor (no-PER_15000) were employed to compete against agents from other students in the RL competition. The former agent ranked 12th out of 116 active participants, while the latter ranked 64th.

Table 1: Final evaluation results of the 27 trained agents after the three training phases (episode checkpoints). For each agent, the "Weak" and "Strong" numbers represent the achieved points against the weak and strong basic opponents, respectively. The "Comp." number indicates the total points from the all-vs.-all competition. The maximum number of points for each mode is 1,560. The colored agents represent the three designated winners based on (I) their evaluation scores and (II) behavior in human play (gold: 1st, silver: 2nd, bronze: 3rd).

	First phase (4000)			Second phase (11000)			Third phase (15000)		
Agent	Weak	Strong	Comp.	Weak	Strong	Comp.	Weak	Strong	Comp.
alpha-0.0	1512	1287	674	1548	1491	948	1551	1503	1101
alpha-0.05	762	790	519	753	646	508	680	597	503
alpha-0.2	509	621	502	552	554	478	801	705	483
alpha-0.5	521	596	498	508	478	492	490	508	458
alpha-auto	1508	1348	592	1411	1177	589	1534	1247	673
lr-0.00003	488	435	499	609	642	517	550	528	507
gamma-0.9	830	604	649	1191	939	799	1174	958	834
no-PER	1510	1209	619	1135	1197	616	1474	1398	808
hard-target	860	566	517	341	445	478	297	444	462

4 Discussion

Our best-performing agent clearly demonstrates the efficacy of actor-critic algorithms in learning complex strategies, such as playing Laser Hockey. It has consistently defeated basic opponents, most contestants, and us human players. Moreover, its performance against other trained agents significantly improved after both the second and third training phases, suggesting that self-play has facilitated better generalization and reduced overfitting. However, we also observed that uncommon events that may not have occurred during training were not adequately addressed. In general, the training of most agents was significantly delayed by draws that dominated the self-play phases, either because no agent managed to score, or because the designated agent did not approach the puck to initiate the game. These games might have provided less informative feedback for the agents and biased the training. Presumably, we should have adhered to the default maximum of 250 steps per episode, as set by the Laser Hockey environment.

The present study was predicated on several heuristics, including the training strategy and reward modification. Moreover, we focused our comparison on α, whose fixed values were potentially excessive, favoring a policy that was too stochastic to effectively exploit rewarding strategies, as indicated by the poor performance of the corresponding agents. In contrast, our best-performing agent had a disabled entropy term and, therefore, no exploration encouragement. This may indicate that the initial random actions, combined with the reward that motivated the agent to touch the puck and thereby reinforced both its goalkeeping and shooting skills, were sufficient to learn a successful and likely quite deterministic strategy. Since all agents with enabled entropy term performed worse, we could not validate the efficacy of the SAC objective, a stochastic actor, or the PER memory in the Laser Hockey environment.

Additional and critical hyperparameters, such as the learning rate, decay factor, or PER settings, have not been sufficiently explored, which is necessary to optimize the potential agent performance. Specifically, it would be essential to test all possible combinations and scales of hyperparameters to maximize the effectiveness of this SAC implementation. Most importantly, it is crucial to compare the SAC agents to agents based on alternative algorithms, including on-policy and model-based versions. Although our competition performance indicates that SAC is appropriate and capable of outperforming many different but potentially untuned approaches, it would be valuable to determine the limitations of our implementation compared to other potentially more suitable and tuned algorithms for this specific environment.

References

[1] Prioritized experience replay implementation. https://github.com/Howuhh/prioritized_experience_replay/tree/main. GitHub repository.

[2] Pytorch soft actor-critic. https://github.com/pranz24/pytorch-soft-actor-critic. GitHub repository.

[3] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[4] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay, 2016.

[5] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.