RL-Course 2025/26: Final Project Report

Team: Toast

February 26, 2026

1 Introduction

In this report, we present our results and findings for the final project of the Reinforcement Learning lecture. The final project environment is Laser Hockey environment in which there are two players and a puck, where each player matches to score goals with the puck.
In general, continuous control tasks involving rigid body physics—such as moving a player to strike a puck in this hockey environment—present several challenges for Reinforcement Learning agents. The agent must learn not only spatial positioning, but also the management of physical actions over time. The agent must learn how to score while also defending opponent's score attempts.
For this task, we implemented and evaluated two state-of-the-art off-policy actor-critic algorithms:

- **Soft Actor-Critic (SAC)** – Mert Bilgin 7034879
- **Twin Delayed Deep Deterministic Policy Gradient (TD3)** – Kevin Van Le 7314700

While TD3 [2] relies on a deterministic policy with target policy smoothing, SAC is based on the maximum entropy framework [4, 5]. This is the fundamental difference, which encourages the agent to maximize both the expected return and the policy entropy of the SAC.
To further address the physical nature of the environment, we adapted our SAC exploration strategy to utilize Pink Noise [1], a temporally correlated noise structure, contrasting it with standard uncorrelated noise. For SAC, we also utilized the updated version of the original SAC paper in which the entropy regularizer parameter α is also learned during the training. [5]
For TD3 we added dropout and weight decay to the Actor and Critic Network and decay those values alongside the policy noise linearly over the number of training steps.
We then structured two different reward shaping to have a diverse action pool as opponents. In terms of experiment, we created consecutive rounds for which we use some of the trained models for training and some of them as hold-out opponents to see the progress over unknown models.

2 Method

2.1 Reward Shaping

2.1.1 Offensive Reward Shaping (The "Attack" Wrapper)

Since the base environment only rewards the agent when a goal is scored, the agent receives very little feedback during early training, making it harder to learn meaningful behavior. To address this, we built a

custom wrapper (`AttackRewardWrapper`) that adds intermediate rewards to guide the agent toward offensive play. The shaped reward R_t^{attack} is defined as:

$$R_t^{\text{attack}} = r_{\text{b},t} + \underbrace{\lambda_w \mathbb{I}_{\text{win},t} - \lambda_l \mathbb{I}_{\text{loss},t}}_{\text{Amplified Game Signal}} + \underbrace{\lambda_{\text{close}} d_t (1 - \mathbb{I}_{\text{touched},t})}_{\text{Fading Closeness Bonus}} + \underbrace{\lambda_{\text{touch}} \mathbb{I}_{\text{just\_touched},t}}_{\text{Touch Bonus}} \tag{1}$$

The $r_{\text{b},t}$ is the base reward. The closeness bonus encourages the agent to move toward the puck early on, but it is turned off the moment the agent touches the puck via the $(1 - \mathbb{I}_{\text{touched},t})$ term. This prevents the agent from simply hovering near the puck to collect easy rewards and instead forces it to take the next step: control the puck and attempt a shot.

2.1.2 Defensive Reward Shaping (The "Defense" Wrapper)

For the defensive setting, we shift the reward signal to prioritize keeping the puck away from our own goal and actively engaging with it in our own half. The shaped reward R_t^{defense} is defined as:

$$R_t^{\text{def}} = r_{\text{b},t} + \underbrace{\lambda_w \mathbb{I}_{\text{win},t} - \lambda_l \mathbb{I}_{\text{loss},t}}_{\text{Asymmetric Game Signal}} + \underbrace{\lambda_{\text{close}} c_t \mathbb{I}^{-}_{\text{touched},t}}_{\text{Fading Closeness Bonus}} + \underbrace{\lambda_{\text{zone}} \max(0,\, x_{\text{puck},t})}_{\text{Zone Bonus}} + \underbrace{\lambda_{\text{int}} \mathbb{I}_{\text{touch},t} \mathbb{I}_{x_{\text{puck}}<0}}_{\text{Interception Bonus}} \tag{2}$$

where $\lambda_w = 5$, $\lambda_l = 15$, $\lambda_{\text{close}} \in \{2.0,\, 0.3\}$ (before and after first touch respectively), $\lambda_{\text{zone}} = 0.2$, and $\lambda_{\text{int}} = 0.5$. The win/loss bonuses are asymmetric, meaning that a goal is penalized three times more than scoring is rewarded, reflecting the defensive priority of the wrapper. The zone bonus gives a small reward whenever the puck is in the opponent's half, and the interception bonus rewards the agent for touching the puck in its own defensive half. We created defensive reward to have as different opponents as possible.

2.2 Soft Actor-Critic (SAC)

Soft Actor-Critic (SAC) is a state-of-the-art, off-policy actor-critic algorithm designed specifically for continuous action spaces. In general, SAC learns a stochastic policy (the actor) and two Q-functions (the critics) to evaluate actions. The agent is trained to maximize expected task rewards while acting as randomly as possible by also maximizing the entropy.
Where standard RL solely maximizes the expected sum of rewards, SAC alters this objective to incorporate an entropy maximization term as below:

$$J(\pi) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))] \tag{3}$$

where $\mathcal{H}$ is the entropy of the policy and α is the temperature parameter which controls the trade-off between exploitation and exploration.
To mitigate the overestimation bias which is common in value-based methods, we employ clipped double Q-learning. The target value for the critic updates is computed as:

$$y = r + \gamma(1-d)\left[\min_{i=1,2} Q_{\text{target},i}(s', a') - \alpha \log \pi_\phi(a'|s')\right] \tag{4}$$

The actor network π_ϕ outputs the mean μ and covariance Σ of a Gaussian distribution. Sampled actions are strictly squashed using a $\tanh$ function to bound them to the environment's permissible action space $[-1, 1]$.

2.3 Automatic Entropy Tuning

We implemented automatic entropy tuning [5], treating the temperature α as a learnable parameter. It is continuously updated by minimizing the following loss formulation:

$$J(\alpha) = \mathbb{E}_{a\sim\pi_t}[-\alpha \log \pi_t(a|s_t) - \alpha\bar{\mathcal{H}}] \tag{5}$$

where $\bar{\mathcal{H}}$ denotes the target entropy heuristically set to the negative dimension of the action space. This mechanism gracefully permits aggressive exploration during early training phases (high α) and converges toward efficient exploitation once a dominant strategy emerges (low α). Below, in **Figure-1**, we compare constant α's versus the learned α. This is one the major improvement for SAC. When the temperature is too high, SAC tries to maximize randomness more than expected reward. When, we take smaller temperature, it starts to converge auto learner, yet still lags behind in the evaluation.

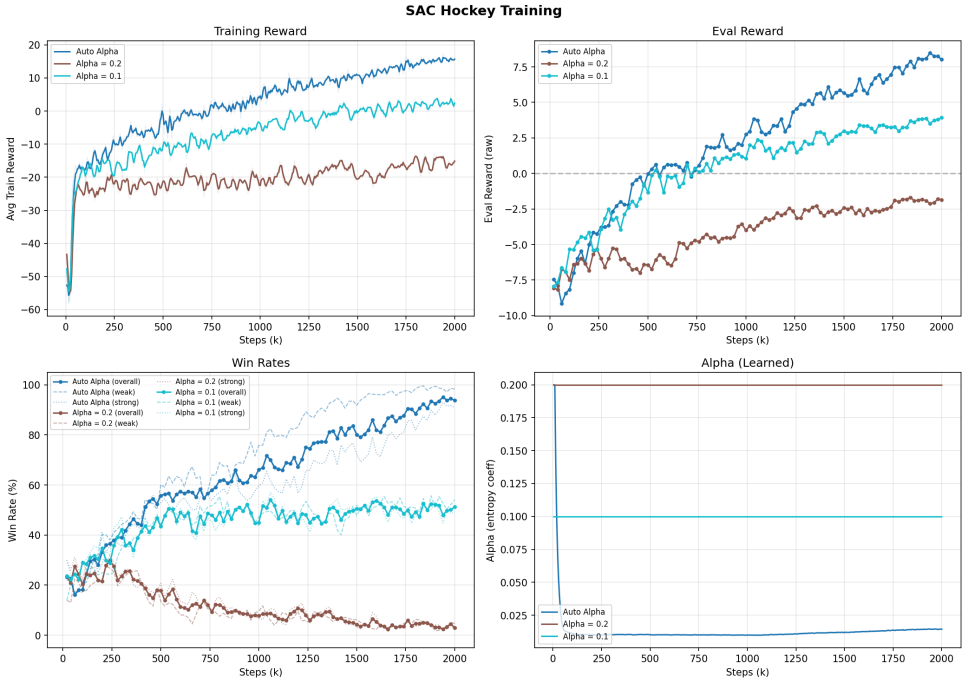


Figure 1: SAC Auto α vs. Constant α

2.4 Pink Noise Exploration

To generate temporally correlated exploration, we sample noise characterized by a power-law Power Spectral Density (PSD) defined as $S(f) \propto 1/f^{\beta}$. We explicitly selected $\beta = 1$, which mathematically corresponds to pink noise. This noise is generated in the frequency domain utilizing the Timmer & König algorithm, followed by an inverse Fast Fourier Transform (FFT) to produce coherent time-domain sequences. During the actor's reparameterization trick, the standard independent white noise $\epsilon \sim \mathcal{N}(0, 1)$ is substituted with our pre-generated pink noise sequence:

$$a = \tanh(\mu_\phi(s) + \sigma_\phi(s) \odot \epsilon_{\text{pink}}) \tag{6}$$

This temporal correlation should allow the agent to commit to a directional physical force across several consecutive frames. However, if you can observe the below **Figure-2**, the pink noise can make

a difference in the scenarios with constant temperature. If the temperature learner is enabled, there is no difference in the evaluation. We think that temporal dependency doesn't add much value for this environment as the length of the episodes are relatively short.

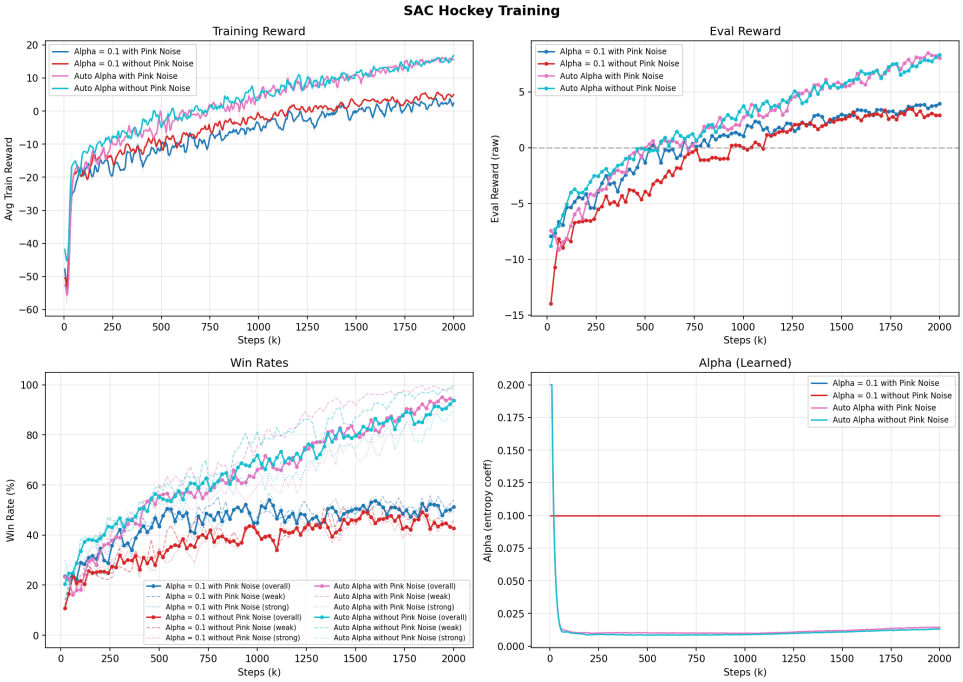


Figure 2: SAC Pink Noise Effect

2.5 Twin Delayed Deep Deterministic Policy Gradient (TD3)

Twin Delayed Deep Deterministic Policy Gradient (TD3) [2] acts as an advanced extension of the Deep Deterministic Policy Gradient (DDPG) [7] algorithm, specifically structured to alleviate the overestimation bias and ensuring policy degradation prevalent in standard actor-critic formulations. In contrast to SAC's stochastic policy, TD3 optimizes a deterministic policy $\mu_\phi(s)$.
The main issue with DDPG is that its single Q-function approximator is prone to overestimating Q-values, which causes the policy to exploit these inaccurate value estimates. To mitigate this, TD3 introduces three critical modifications over standard DDPG:

1. Clipped Double-Q Learning: Instead of relying on a single critic as in DDPG, TD3 maintains two independent Q-functions Q_{θ_1} and Q_{θ_2} and uses the smaller of both Q-values as the target value [6]. This counteracts the overestimation that potentially arises when only a single Q-function is used. The target value is computed as:

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\theta_{i,\text{targ}}}(s', a'(s')) \tag{7}$$

where γ denotes the discount factor and d indicates whether the state is terminal.

2. Target Policy Smoothing: A failure that can occur in DDPG is that the Q-function approximator develops incorrect sharp peaks for certain actions, which the deterministic policy would then quickly exploit. TD3 addresses this by adding clipped noise to the target action:

$$a'(s') = \text{clip}\left(\mu_{\phi_{\text{targ}}}(s') + \text{clip}(\epsilon, -c, c),\ a_{\text{low}},\ a_{\text{high}}\right), \quad \epsilon \sim \mathcal{N}(0, \sigma) \tag{8}$$

This serves as a regularizer by smoothing out the Q-function over similar actions, which makes the learning less prone to errors in the value estimates.

3. Delayed Policy Updates: TD3 updates the policy and the target networks at a lower frequency than the Q-functions. This reduces the volatility that normally arises in DDPG because of how a policy update changes the target.
Given these three modifications, both Q-functions are trained by minimizing the mean squared Bellman error against the shared target:

$$L(\theta_i, \mathcal{D}) = \underset{(s,a,r,s',d)\sim\mathcal{D}}{\mathbb{E}} \left[\left(Q_{\theta_i}(s,a) - y(r,s',d)\right)^2\right] \tag{9}$$

The policy is then learned by maximizing the expected Q-value under the first critic:

$$\max_{\phi} \underset{s\sim\mathcal{D}}{\mathbb{E}} \left[Q_{\theta_1}(s, \mu_\phi(s))\right] \tag{10}$$

Finally, the target networks for both the critics and the actor are updated via Polyak averaging:

$$\phi_{\text{targ}} \leftarrow \rho\,\phi_{\text{targ}} + (1-\rho)\,\phi, \quad \theta_{i,\text{targ}} \leftarrow \rho\,\theta_{i,\text{targ}} + (1-\rho)\,\theta_i \tag{11}$$

Since TD3 employs a deterministic policy, the agent would not sufficiently explore the action space on its own. Therefore, during training, exploration is facilitated by adding uncorrelated Gaussian noise to the actions selected by the policy.

2.5.1 TD3 Improvements

This idea is motivated by the same principle as annealing the epsilon in epsilon-greedy action selection [9]. At the start of training, higher values of dropout [8] and noise encourage the agent to explore a wider range of actions and also prevent the networks from overfitting to early experiences. As training progresses and the policy becomes more stable, these values are linearly reduced which allows for a gradual shift from exploration towards exploitation.

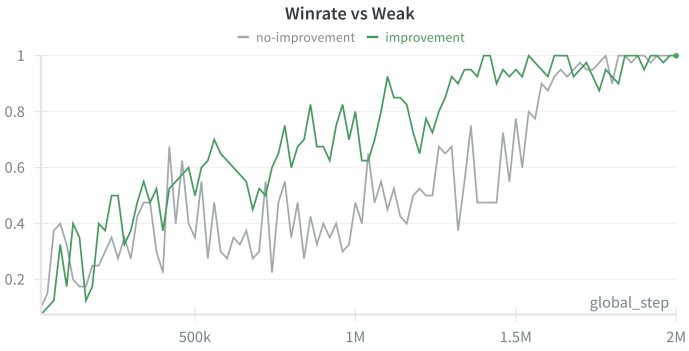


Figure 3: Win rate vs weak bot with and without improvement

To validate the improvement, a model with and without improvement was trained on the weak bot and evaluated on the weak bot as well, however in Figure 3 both models start to converge to 100% winrate indicating no changes with or without improvement.

3 Experiment

3.1 Experimental Setup

Our training approach consists of two training phases which both employ the same round-based training approach but with a different goal:

Phase 1: Creating fixed opponents The goal of Phase 1 is to create fixed opponents to be used in the evaluation and training sets for Phase 2. Otherwise, there is no way to assess the generalizability of our trained models. To address this, we first trained a model for each algorithm and reward shape pair using the fixed training set, which only consists of the weak and strong bots. This concludes the first round. In the second round, we added the latest model of each algorithm-reward shape pair from the first round to enrich the training of subsequent models with a more diverse set of opponents. In subsequent rounds, each model of an algorithm and reward shape pair in the training opponent set will be replaced by the latest model from the previous round.

Phase 2: Main round-based training In this phase, we use the models created in the previous phase to add to be included in the fixed training set and also to create the evaluation set. To ensure generalizability, we make sure that those sets are not overlapping.
The difference between training in phase 1 and 2 is that instead of using the latest algorithm-reward shape model of the previous rounds to be included in the next round, we use the algorithm-reward shape model that yields the highest average reward on the evaluation.
To ensure a fair comparison between the rounds, each round trains the algorithm-reward shape model from scratch so no continual training from previous rounds is used.
The goal of the main round is to create a model that improves in performance on the evaluation set over the number of rounds.

3.2 Results

Figure 4 shows that, with round-based training, the performance in subsequent rounds increases. Notably, there is a big improvement in round 2 after the latest trained models from round 1 are included as training opponents. From round 2 onward, the mean reward continues to grow but the gains become smaller. The win rate increases in subsequent rounds, whereas the draw rate remains roughly the same. The loss rate drops from around 22% in round 1 to about 11% by round 5, showing that the agents learn to avoid losing rather than just converting draws into wins.
But the gains become smaller. The win rate increases in subsequent rounds, whereas the draw rate remains roughly the same.

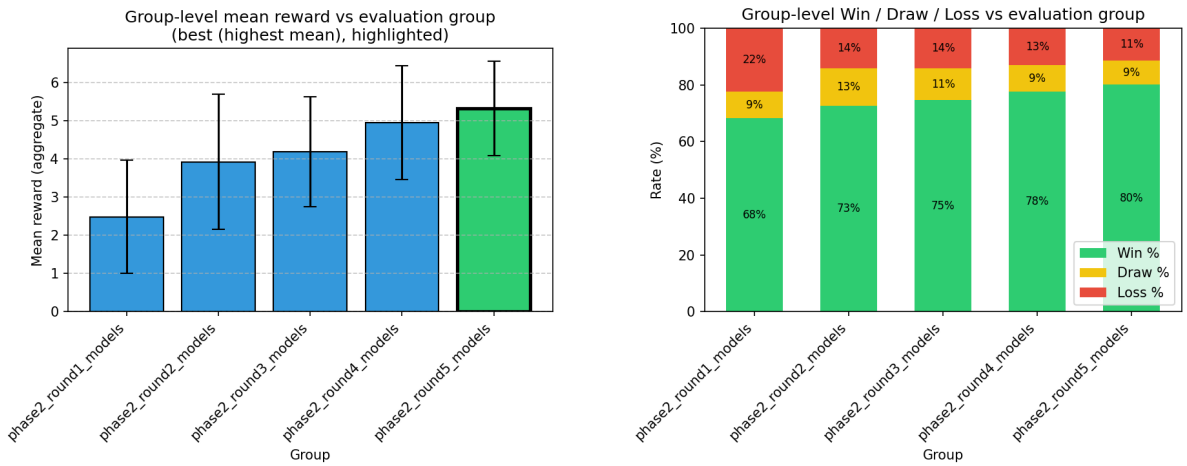
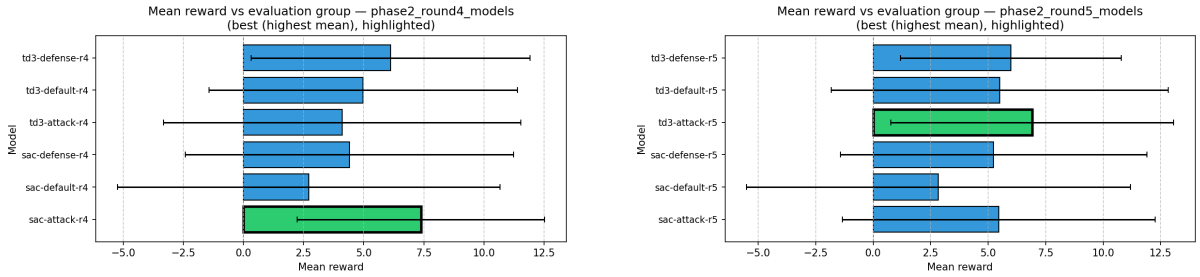


Figure 4: Round-level performance on evaluation set

Figure 5 provides a closer look at the performance of each algorithm-reward shape pair. SAC with the attack reward shape achieves the highest mean reward in round 4 and TD3 with the attack reward shape achieves the highest mean reward in round 5. SAC-attack-r4 has a slightly higher mean reward than td3-attack-r5. The small gap between the best performers also is reflected in their respective ranks in the competition: sac-attack-r4 (Toast-sac) placed 24th and TD3-attack-r5 (Toast-td3) placed 27th.



(a) Mean reward (Phase 2, Round 4)

(b) Mean reward (Phase 2, Round 5)

Figure 5: Within-group reward on evaluation set — Phase 2 Rounds 4 and 5

Figure 6 aggregates performance across all rounds by algorithm-reward shape. The attack reward shape yields the highest mean reward for SAC, while sac-default performs noticeably worse. This suggests that the attack-focused reward signal provides a stronger learning signal for SAC. The TD3 variants are more consistent across reward shapes, with all three performing similarly. Overall, the choice of reward shape has a larger impact on SAC than on TD3.

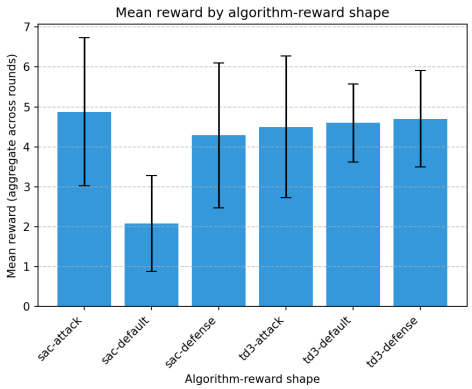


Figure 6: Average algorithm-reward shape performance on evaluation set

3.3 Selecting the competition models

As above-mentioned we opted for the "sac-attack-r4" and "td3-attack-r5" as the algorithm-specific models, as both performed showed the highest mean in reward across all rounds.
Additionally, we created a final team model called "Toast" using the TD3 algorithm and the attack reward shape. This model was trained using all of the models that we had created thus far, as we hypothesized that the more different opponents the model sees during training, the better it will perform in the competition. This follows the same principle as in classic deep learning, where a diverse set of training data generally allows for better performance on unseen data [3]. The competition results clearly highlight the importance of adding more opponents, as the team model ended up in 3rd place (according to the last meeting), compared to our Phase 2 models, which ended up in 24th and 27th place respectively.

References

[1] O. Eberhard, J. Hollenstein, C. Rosenbaum, R. Legaspi, et al. Pink noise is all you need: Colored noise exploration in deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2023.

[2] S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. 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 1587–1596, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[3] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[4] 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.

[5] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[6] H. v. Hasselt. Double q-learning. In *Proceedings of the 24th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'10, page 2613–2621, Red Hook, NY, USA, 2010. Curran Associates Inc.

[7] 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, 2019.

[8] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, Jan. 2014.

[9] R. Sutton and A. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 9(5):1054–1054, 1998.