RL-Course 2025/26: Final Project Report

WayneGradientzky: Rajinish Aneel Bhatia, Bartol Markovinović, Mohd Khizir Siddiqui

February 26, 2026

1 Introduction

The report presents the implementation of three reinforcement learning algorithms by the *WayneGradientzky* team. The algorithms were finally evaluated on the Hockey competition hosted by the Martius Lab. `Hockey` is a 2D-multi-agent, fully observable environment where two agents compete to score goals by hitting a puck into the opponent's goal.
The algorithms implemented are as follows:

- **Task-Driven Model Predictive Control (TDMPC)** by Bartol Markovinović
- **Twin Delayed Deep Deterministic Policy Gradient (TD3)** [**?**] by Rajinish Aneel Bhatia
- **Soft Actor-Critic (SAC)** [**?**] by Mohd Khizir Siddiqui

All the implementations are available on the GitHub repository `https://github.com/BartolMarko/HockeyRL`. Each team member implemented their code individually, with credit given where due. The report covers the fundamentals of each algorithm, team member contributions and the evaluation on the Hockey environment. Finally, we conclude with a discussion of the results and potential future work in.

2 Twin Delayed Deep Deterministic Policy Gradient (TD3)

Twin Delayed Deep Deterministic Policy Gradient (TD3) [**?**] is a model-free, off-policy algorithm for continuous action spaces, improving over DDPG via clipped double Q-learning, delayed policy updates, and target policy smoothing. TD3 maintains two Q-networks (Q_{ϕ_1} and Q_{ϕ_2}) and uses their minimum as the TD target:

$$y = r + \gamma \min_{i=1,2} Q_{\phi_i'}(s', \tilde{a}), \quad \tilde{a} = \pi_{\theta'}(s') + \epsilon, \quad \epsilon \sim \texttt{clip}(\mathcal{N}(0, \sigma), -c, c)$$

Each critic minimizes the squared Bellman error $\mathcal{L}(\phi_i) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}(Q_{\phi_i}(s,a) - y)^2$, while the actor maximizes $J(\theta) = \mathbb{E}_{s\sim\mathcal{D}} Q_{\phi_1}(s, \pi_\theta(s))$ and is updated every 2 critic steps.

2.1 Method

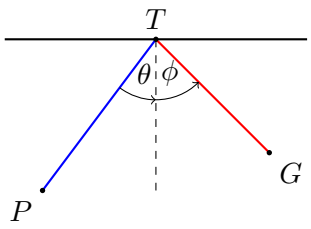
2.1.1 N-Step Returns

Instead of the standard one-step TD target, we use n-step returns to propagate reward signal faster:

$$y = r(s_0) + \gamma r(s_1) + \cdots + \gamma^{n-1} r(s_{n-1}) + \gamma^n q(s_n, \tilde{a})$$

2.1.2 Custom Opponent and Bank-Shot Reward

We noticed early on that the model did not always manage to learn ricochet (bank) shots, so to help it defend against opponents that hit those shots we introdued a hard-coded opponent that always plays them. Given puck position P, goal position G, and T as the target we must shoot at to reach G. Then T_y is known because that is the boundary and we only need to find T_x. We can assume that $T_x \geq P_x$ and $G_x \geq T_x$. If we assume frictionless walls than $\theta = \phi$.



From $\tan\theta = \tan\phi$ we obtain:

$$T_x = \frac{|T_y - P_y|\, G_x + |T_y - G_y|\, P_x}{|T_y - G_y| + |T_y - P_y|}$$

This agent achieves $\approx 94\%$ win rate against the weak opponent and $\approx 80\%$ against the strong opponent. We also shaped the reward by calculating how aligned the agent's direction was to this direction to encourage the learning agent to prefer shots in this direction.

2.1.3 Layer Normalization

Layer Normalization. Layer normalization normalizes activations across the feature dimension for each sample independently. For a given input vector x, it computes:

$$\texttt{LayerNorm}(x) = \frac{x - \mu}{\sigma + \varepsilon}\,\gamma + \beta$$

where μ and σ are the mean and standard deviation of x, and γ, β are learnable scale and shift parameters. In the RL setting this reduces sensitivity to the scale of inputs and gradients, which can vary dramatically during training. We apply LayerNorm after each linear layer in both actor and critic networks.

2.2 Experiments

2.2.1 Training in Phases (Opponent Scheduling)

Training proceeded in three phases. Phase 1 used only the weak opponent. Phase 2 sampled weak/strong opponents with probabilities 30%/70%. Phase 3 used a mixed pool: 20% weak, 20% strong, 10% custom bank-shot opponent, and 50% from a rolling queue of past checkpoints (self-play). An adaptive opponent selection strategy based on Thompson sampling was evaluated as an alternative but underperformed the manual schedule, as shown in Figure **??**.

2.2.2 Self-Play

Self play was the part of the third phase of training and was implemented by keeping a queue of 50 checkpoints and adding a new checkpoint every 50000 timesteps. This ensured that the model had enough learning experience from each of its previous checkpoints to be able to defeat them.

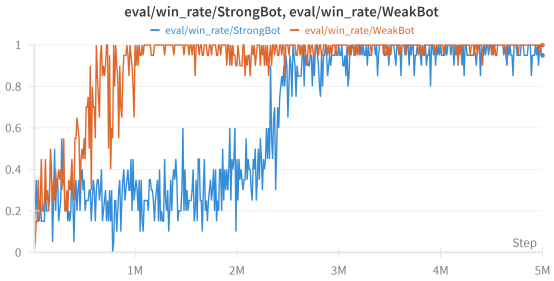


Figure 1: Winrate (trained in phases)

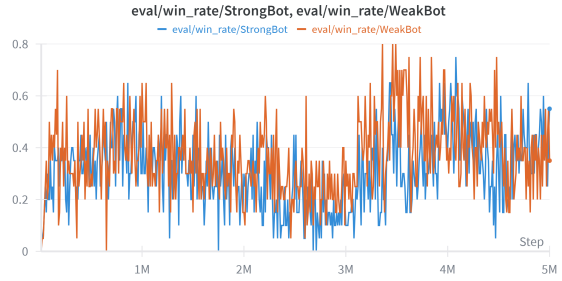


Figure 2: Winrate (trained using Thompson sampling)

2.2.3 N-Step Returns, Episode Mirroring and Environment Scheduling

N-step returns and episode mirroring improve the speed of learning as reflected in (Figure **??**), compared to one-step TD targets. We chose 3-step for the final training. Environment scheduling (30% SHOOTING_MODE, 70% DEFENSE_MODE) was introduced in Phase 3. Figure **??** shows the win rate when the opponent starts with possession and confirms a clear improvement.

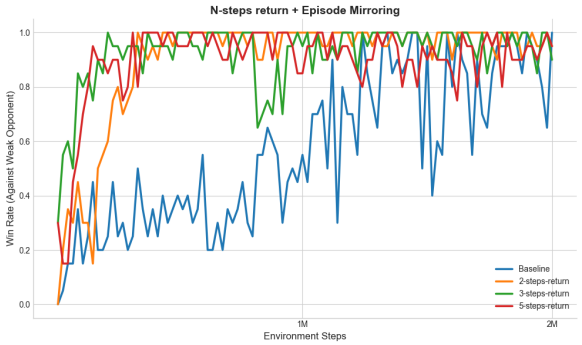


Figure 3: Win rate vs. past checkpoints (n-step returns)

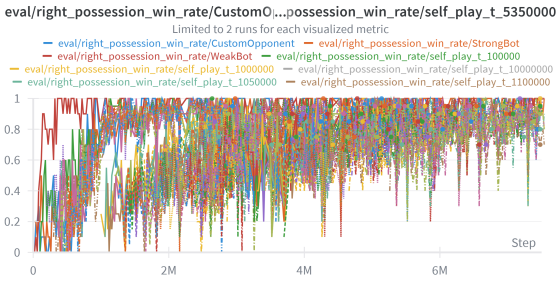


Figure 4: Win rate when opponent starts with possession using environment scheduler

2.2.4 Bank-Shot Reward

The bank-shot preference reward visibly shifts puck density toward the walls (Figures **??**, **??**), confirming that the agent learned to incorporate ricochet shots into its strategy.

2.2.5 Layer Normalization

LayerNorm converged faster against the fixed bots (Figure **??**) but hurt generalization during self-play: the agent progressively drew against all previous checkpoints (Figure **??**) rather than maintaining dominance, so it was excluded from the final model.

3 Discussions and Conclusion

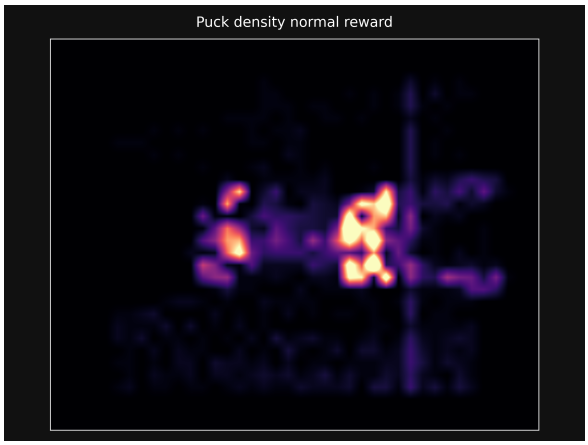


Figure 5: Puck density — standard reward

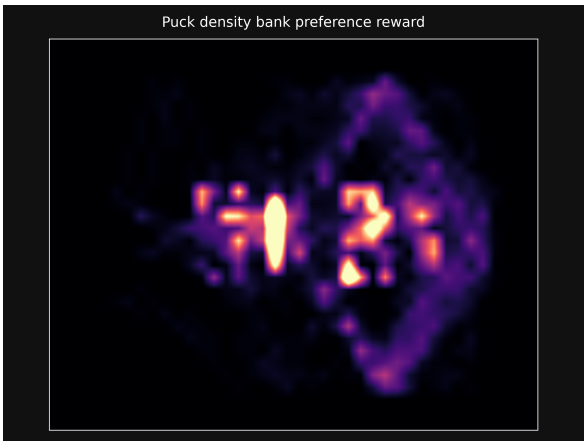


Figure 6: Puck density — bank-shot preference reward

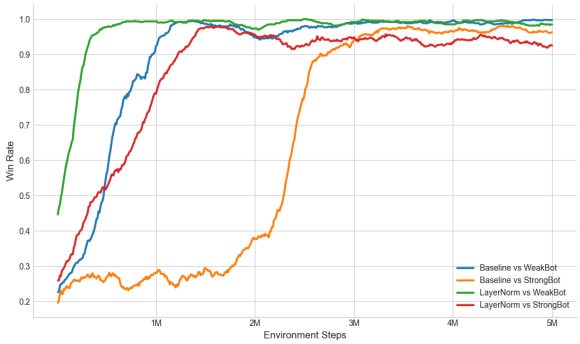


Figure 7: Win rate: baseline vs. LayerNorm

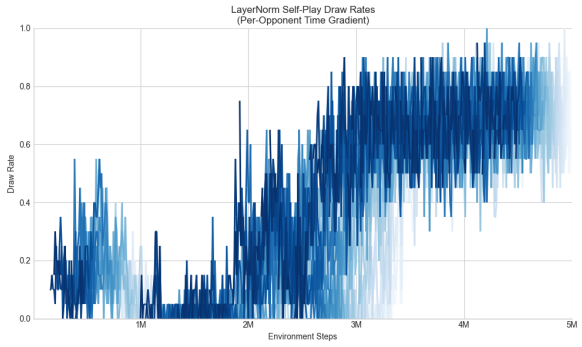


Figure 8: Draw rate of LayerNorm vs. past checkpoints