RL-Course 2025/26: Final Project Report

Ansel Cheung, Jannik Mänzer, Niklas Abraham

January 26, 2026

Abstract

This report presents our application and comparative study of modern Reinforcement Learning (RL) algorithms—specifically, Twin Delayed Deep Deterministic Policy Gradient (TD3), Soft Actor-Critic (SAC), and the model based TD-MPC2—within the challenging Laser Hockey environment. We discuss the environment's design, its state and action spaces, and the unique characteristics making it a compelling testbed for RL research. Our work details the methodological approaches taken with these algorithms, summarizes key experimental findings, and reflects on lessons learned in this multi agent, continuous control setting.

1 Introduction

1.1 Environment Overview

The Laser Hockey environment [2] is a custom reinforcement learning benchmark built on the Gymnasium Python API and powered by the Box2D physics engine. In this multi agent setting, two agents each control a hockey stick and compete to score goals by striking a puck into the opponent's net. The environment is designed such that both agents receive identical but mirrored observations at each timestep, which eliminates the need for side specific learning strategies.

$$s_t = (\underbrace{x_1, y_1, \theta_1, v_{x,1}, v_{y,1}, \omega_1}_{\text{Player 1}}, \underbrace{x_2, y_2, \theta_2, v_{x,2}, v_{y,2}, \omega_2}_{\text{Player 2}}, \underbrace{x_p, y_p, v_{x,p}, v_{y,p}}_{\text{Puck}}, \underbrace{t_{\text{puck},1}, t_{\text{puck},2}}_{\text{Puck possession time}})$$

(1)

At each timestep, the agent receives an 18 dimensional continuous state vector that captures the complete game state. This observation includes the position and orientation of both players relative to the center of the field, along with their linear and angular velocities. The state also contains the puck's position and velocity, as well as time remaining for each player's puck possession in keep mode, which ranges from 0 to 15 seconds. Player 1 refers to the agent currently being controlled, while Player 2 represents the opponent. The observation structure ensures that when the viewpoint switches during training, the indices are automatically mirrored so that each agent always perceives itself as Player 1, maintaining a consistent learning perspective.

The action space consists of a 4 dimensional continuous vector that controls the agent's stick. The first two components specify forces applied for stick movement in the x and y directions, while the third component controls the torque applied to adjust the stick's angle. The fourth component is a thresholded scalar that determines whether to release the puck when the agent is in possession.

Reward feedback in this environment is sparse and goal oriented. An agent receives a reward of +10 for scoring a goal, 10 for conceding one, and 0 for a draw. Additionally, there is a small reward signal

for maintaining proximity to the puck, which helps guide exploration during early learning stages. Each episode begins with all entities positioned at the center of the field and terminates when a goal is scored or a timeout occurs. The environment supports both scripted opponents, such as the BasicOpponent, and learning agents as adversaries, making it a versatile and challenging benchmark for continuous control and multi agent reinforcement learning research.

2 Method

2.1 TD-MPC 2 - Niklas Abraham

TD-MPC2 [1] is a model-based reinforcement learning algorithm that learns a world model to predict future states and rewards, and uses this model to select actions through planning. The core idea is to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$ in a Markov Decision Process with an infinite horizon. The policy is constructed to maximize the expected discounted return. In TD-MPC2, this is achieved by learning a world model and selecting actions by planning with the learned models.
For planning, TD-MPC2 employs the Model Predictive Control (MPC) framework, in which actions are optimized based on planning over action sequences of a finite horizon H:

$$\pi(s_t) = \arg\max_{a_1,\ldots,a_H} \mathbb{E}_\pi \left[\sum_{\tau=0}^{H} \gamma^\tau r(s_{t+\tau}, a_{t+\tau}) \right]. \tag{2}$$

The return of each trajectory is estimated by simulating action sequences through the learned world model. However, this approach often leads to only locally optimal policies. To address this limitation, TD-MPC2 additionally utilizes a value function to guide the planning process and improve the policy toward a more globally optimal solution.
Rather than predicting raw future observation states, TD-MPC2 learns to predict a maximally useful latent representation for accurately estimating the outcomes of action sequences. The algorithm is composed of five distinct neural network components that interact in a coordinated manner, as illustrated in Figure 1.

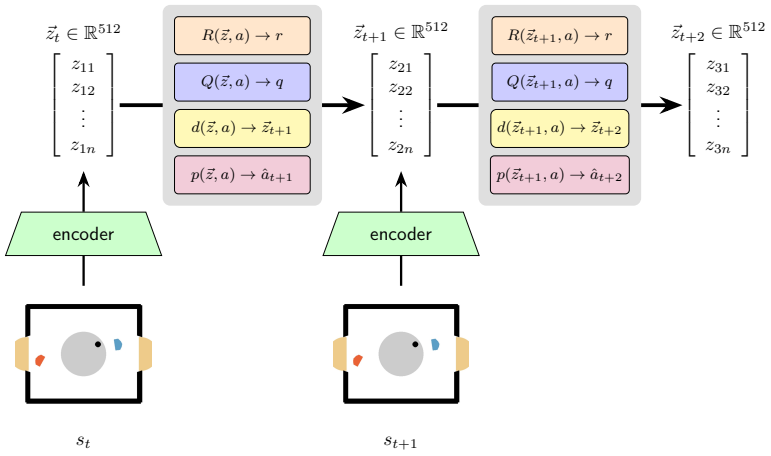

Figure 1: TD-MPC2 agent architecture. The reward, Q-value, dynamics, and action heads are grouped together in the world model, with the latent space flow shown in the background.

- **Encoder Network:** The encoder $h : \mathcal{S} \to \mathcal{Z}$ is a feedforward neural network that maps the current state to a latent representation $\vec{z} \in \mathbb{R}^{512}$. In our implementation, we use a 512 dimensional latent space.
- **Latent Dynamics Network:** The dynamics model $d : \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}$ takes the current latent state $\vec{z}_t$ and action a as input and predicts the next latent state $\vec{z}_{t+1}$.

- **Reward Network:** The reward predictor $R : \mathcal{Z} \times \mathcal{A} \to \mathbb{R}$ estimates the reward r for a given latent state-action pair $R(\vec{z}, a) \to r$.
- **Termination Network:** This component predicts whether the episode will terminate in the next step, which is particularly relevant when the puck is about to enter the goal.
- **Q-Network Ensemble:** The Q-function consists of an ensemble of five Q-networks that estimate the action value $Q(\vec{z}, a) \to q$, representing the expected discounted return starting from latent state $\vec{z}$ and taking action a. In practice, two samples are drawn from the ensemble and the minimum is taken to reduce overestimation bias.
- **Policy Network:** The policy $p : \mathcal{Z} \times \mathcal{A} \to \mathcal{A}$ maps latent states and actions to predicted actions $p(\vec{z}, a) \to \hat{a}$ and is used to guide the sample based trajectory optimization.

2.1.1 Planning with MPPI

For local planning, TD-MPC2 employs Model Predictive Path Integral (MPPI) control [3] to sample and evaluate action sequences, with the policy network guiding the sampling process. At each decision step, we estimate the parameters $\mu^*, \sigma^* \in \mathbb{R}^{H \times m}$ of a time dependent multivariate Gaussian distribution with diagonal covariance that maximizes the expected return:

$$\mu^*, \sigma^* = \arg\max_{\mu,\sigma} \mathbb{E}_{a_t, a_{t+1}, \ldots, a_{t+H} \sim \mathcal{N}(\mu, \sigma^2)} \left[\gamma^H Q(\vec{z}_{t+H}, a_{t+H}) + \sum_{\tau=t}^{H} \gamma^\tau R(\vec{z}_\tau, a_\tau) \right]. \tag{3}$$

This optimization is solved iteratively by sampling action sequences from $\mathcal{N}(\mu, \sigma^2)$, evaluating their expected returns, and updating μ and σ based on a weighted average of the best performing samples. The termination model predicts when an episode ends and stops further sampling in that sequence. To accelerate convergence, the policy network p is used to generate a fraction of the action sequences, and the initial values of μ and σ are warm started from the previous decision step, shifted by one time step.

2.1.2 Architecture and Training

All network components are implemented as multi layer perceptrons (MLPs) with linear layers and Mish activation functions. Following [1], to mitigate exploding gradients, the latent representation $\vec{z}$ is projected into L fixed dimensional simplices using a softmax operation. This enforces sparsity in the latent space and greatly stabilizes training.
The training procedure employs an experience replay buffer $\mathcal{B}$ that maintains complete episode trajectories comprising state transitions, actions, and reward signals. During each training iteration, the world model parameters are optimized by sampling contiguous subsequences of length $H + 1$ from $\mathcal{B}$. The optimization objective is formulated as a joint loss function that simultaneously trains the dynamics, reward, and value prediction components:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1})_{t=0}^{H} \sim \mathcal{B}} \left[\sum_{t=0}^{H} \lambda^t \left(\|\vec{z}_{t+1} - \text{sg}(h(s_{t+1}))\|_2^2 + \text{CE}(\hat{r}_t, r_t) + \text{CE}(\hat{q}_t, q_t) \right) \right], \tag{4}$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operator that prevents backpropagation through the encoder, $\vec{z}_{t+1} = d(\vec{z}_t, a_t)$ represents the predicted next latent state, $\hat{r}_t = R(\vec{z}_t, a_t)$ and $\hat{q}_t = Q(\vec{z}_t, a_t)$ denote the predicted reward and Q-value respectively, and $\lambda \in (0, 1]$ is a temporal discount factor that exponentially decays the contribution of predictions at longer horizons. The temporal difference (TD) target for Q-value estimation is computed as $q_t = r_t + \gamma \bar{Q}(\vec{z}_{t+1}, p(\vec{z}_{t+1}, a_{t+1}))$, where $\bar{Q}$ represents an exponential moving

average of the Q-network parameters to stabilize training. Following the TD-MPC2 formulation, both reward and value predictions are cast as discrete regression tasks in a log-transformed space, with optimization performed via cross-entropy minimization using soft target distributions.
The policy network p is optimized according to a maximum entropy reinforcement learning objective that encourages exploration while maximizing expected returns:

$$\mathcal{L}_p(\theta) = \mathbb{E}_{(s_t,a_t)_{t=0}^{H}\sim\mathcal{B}}\left[\sum_{t=0}^{H}\lambda^t\left(\alpha Q(\vec{z}_t, p(\vec{z}_t, a_t)) - \beta\mathcal{H}(p(\cdot|\vec{z}_t))\right)\right], \tag{5}$$

where the latent state trajectory is recursively computed as $\vec{z}_{t+1} = d(\vec{z}_t, a_t)$ with initial condition $\vec{z}_0 = h(s_0)$, and $\mathcal{H}(p(\cdot|\vec{z}_t))$ quantifies the entropy of the policy distribution conditioned on the latent state. The hyperparameters $\alpha > 0$ and $\beta > 0$ control the relative weighting between the value maximization and entropy regularization terms, thereby preventing premature policy collapse to deterministic behavior during the optimization process.

3 Results

References

[1] N. Hansen, H. Su, and X. Wang. Td-mpc2: Scalable, robust world models for continuous control, 2024.

[2] G. Martius. Hockey environment. https://github.com/martius-lab/hockey-env, 2023.

[3] G. Williams, A. Aldrich, and E. Theodorou. Model predictive path integral control using covariance variable importance sampling, 2015.