Adapting Under Friction: Exploring Continual Learning in Dynamic Environments

Evan Li [*1] **Justin Phillips** [*1] **Max Zuo** [*1] **Yeganeh Kordi** [*1]

Abstract

Continual Learning (CL) in Reinforcement Learning (RL) presents a significant challenge, as agents must adapt to a sequence of tasks without catastrophic forgetting. This paper extends the TD-MPC2 algorithm to address the CL problem. By leveraging its model-based approach and multi-task learning capabilities, we aim to improve the CL performance of state-of-the-art RL algorithms. Our proposed method focuses on preserving past knowledge while acquiring new skills, enabling agents to adapt to changing environments efficiently. Additionally, we explore potential optimizations to improve the computational efficiency of TD-MPC2, enabling faster training and deployment. Through experiments on the Slippery Ant environment, we demonstrate the significant challenges of the continual learning Slippery Ant environment and provide possible paths for improving on TD-MPC2 for dynamic environments.

1. Introduction

Reinforcement learning (RL) has achieved remarkable success in applications ranging from robotic control to strategic decision-making in complex environments. However, traditional RL methods often rely heavily on handcrafted reward functions and substantial computational resources, which limits their scalability and adaptability to real-world settings. These constraints have driven growing interest in self-supervised reinforcement learning, which leverages intrinsic objectives to guide policy learning without explicit supervision. By reducing dependence on external reward signals, self-supervised RL enables more robust and efficient learning processes, especially in resource-constrained or dynamically changing environments.

[*]Equal contribution [1]Brown University. Correspondence to: Evan Li <evan_li1@brown.edu>.

Submitted to the Brown Self Supervised Learning Workshop (BSSL),

Recent advancements in training large models on internet-scale datasets have shown that generalist models, which perform a wide variety of tasks, can be achieved through the availability of enormous datasets and scalable architectures (Brown et al., 2020; He et al., 2022; Kirillov et al., 2023). However, extending this paradigm to embodied control tasks faces significant challenges, such as the reliance on near-expert trajectories for behavior cloning and the absence of scalable continuous control algorithms capable of processing uncurated, diverse datasets (Reed et al., 2022; Brohan et al., 2023). RL offers a promising framework for overcoming these issues, yet existing RL algorithms are often constrained by single-task learning paradigms, task-specific hyperparameters, and limited robustness to variability in action spaces and data quality (Lillicrap et al., 2016; Haarnoja et al., 2018).

In this work, we introduce HTD-MPC2, a self-supervised framework aimed at addressing these challenges by advancing the capabilities of model-based RL. The framework draws inspiration from the scalability principles demonstrated by TD-MPC2 (Hansen et al., 2024b), a generalist RL algorithm designed to learn across multiple task domains, embodiments, and action spaces without reliance on domain-specific knowledge. HTD-MPC2 builds on this foundation with a focus on ensuring latent state consistency for reliable temporal predictions and adopts a decoder-free architecture to minimize computational overhead. A key innovation is the framework's ability to emphasize predictions several states ahead, rather than merely predicting the next state, which aligns with the scaling benefits observed in TD-MPC2 and facilitates generalization to unseen conditions.

Through experimentation, we evaluated HTD-MPC2 in the Slippery Ant environment (Dohare et al., 2021) on episode reward return. The results demonstrate how well TD-MPC2 already performs, as well as just how challenging the dynamic Slippery Ant environment task is.

2. Background

2.1. Reinforcement Learning

Reinforcement Learning (RL) is a framework for sequential decision-making where an agent learns to map states to ac-

tions to maximize cumulative rewards through interaction with an environment. RL methods are broadly categorized into model-free approaches, such as Q-learning and policy gradient methods, which learn directly from experience, and model-based approaches, which learn a model of the environment for planning and policy improvement (Moerland et al., 2020).

2.2. Self-Supervised Learning in RL

Self-Supervised Learning (SSL) in RL focuses on learning representations and skills without explicit rewards or labels by exploiting the structure and dynamics of the environment (Laskin et al., 2021). Notable SSL approaches include Unsupervised Reinforcement Learning (URL), which leverages intrinsic motivation strategies like curiosity-driven (Pathak et al., 2017) and diversity-driven exploration (Eysenbach et al., 2018), and empowerment (Mohamed & Rezende, 2015).

Auxiliary tasks, such as predicting future states and inverse dynamics (Pathak et al., 2017), further enhance SSL by guiding representation learning and enabling transferable skill acquisition. Recent SSL advances have improved sample efficiency, generalization, and robustness across applications like robotics, video games, and continuous control (Hansen et al., 2024a; Laskin et al., 2021).

2.3. Model Predictive Control in RL

Model Predictive Control (MPC) optimizes action sequences over a finite horizon, considering system constraints and future behavior. Despite challenges like computational complexity and model sensitivity (Mayne, 2014), advancements such as TD-MPC2 address these limitations by using an implicit world model to predict rewards and value estimates without reconstructing observations (Hansen et al., 2024b).

TD-MPC2 performs trajectory optimization in latent space and extends return estimates beyond the planning horizon, demonstrating robust performance across diverse continuous control tasks with scalable world models and adaptability to complex domains (Brown et al., 2020).

2.4. TD-MPC2

The implicit world model is central to TD-MPC2's approach to reinforcement learning, consisting of five key components that learn and predict environment dynamics:

- **Encoder** $z = h(s, e) \rightarrow$ Maps observations to their latent representations.
- **Latent dynamics model** $z' = d(z, a, e) \rightarrow$ Models (latent) forward dynamics.
- **Reward model** $\hat{r} = R(z, a, e) \rightarrow$ Predicts reward r of a transition.
- **Terminal value model** $\hat{q} = Q(z, a, e) \rightarrow$ Predicts discounted sum of rewards (return).
- **Policy prior model** $\hat{a} = p(z, e) \rightarrow$ Predicts next action a^* that maximizes Q.

All components are implemented as multi-layer perceptrons.

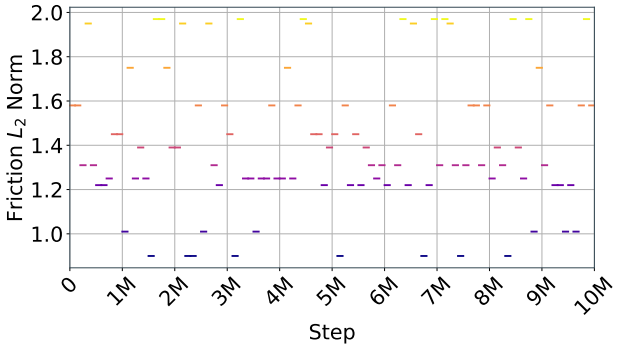

Figure 1. The norm of the 11 unique friction parameters over all timesteps.

3. Experimental Setup

All our experiments are conducted in the Slippery Ant environment (Dohare et al., 2021) implemented with Brax (Freeman et al., 2021) and Mujoco with XLA (Todorov et al., 2012).

3.1. Slippery Ant

In the Slippery Ant environment, after N number of steps, the friction parameters (f_s, f_d, f_r) of the environment model are randomly resampled from a distribution of values. The default friction parameters are $[1, 0.5, 0.5]$. We resample our friction parameters from log uniform (LU) distributions, ($f_s \sim LU(\frac{1}{2}, 2)$, $f_d \sim LU(\frac{1}{4}, 1)$, $f_r \sim LU(\frac{1}{4}, 1)$), every 100,000 steps. To explore revisiting previous friction parameters, we first randomly sample a set of 10 friction parameters plus the original friction parameters, then permute and recycle such parameters in random order throughout training (Figure 1). All agents follow the same permutation of friction parameters, and all begin and end with the original friction parameters. We train our agents for a total of 10 million steps.

3.2. Infrastructure

Experimenting with continual learning requires training a model for significantly longer than it would normally take for it to converge on a single task, as we wish to examine model behavior under a number of changing task conditions

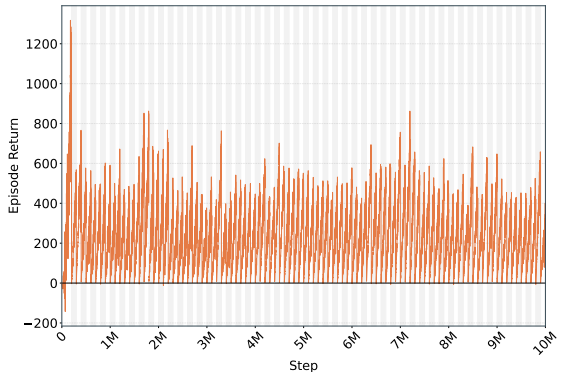

Figure 2. Baseline performance of TD-MPC2 (Horizon=3). Friction changes/environment resets are indicated with gray and white bands.

over long periods of time, while still being able to iterate quickly.

To accelerate our experiments, we utilized a public JAX implementation of TD-MPC2 in (Flandermeyer, 2024), which enabled substantial computational speedups, ranging from 5 to 10 times faster than the original implementation in PyTorch (Paszke et al., 2017). From there, we switched to Mujoco with XLA (MJX) (Todorov et al., 2012) and Brax (Freeman et al., 2021). This enabled another 4-6x speedup, bringing experiment times from $\tilde{2}4$ hours down to ¡6 hours. This speedup was quintessential for conducting extensive experiments under our continual learning framework.

4. Understanding in TD-MPC2

To better understand our baseline method, TD-MPC2 (Hansen et al., 2024b), we run a few experiments exploring how environmental changes affect it. We run a baseline evaluation, an experiment where we freeze the dynamics model after a certain number of steps, and an experiment where the friction parameters are appended to the model's observations.

Baseline Evaluation We first evaluate the performance of TD-MPC2 in the Slippery Ant environment (Dohare et al., 2021) with the default friction parameters. We train the agent for 10 million steps, shown in Figure 2. The agent performs well in the environment. However, there seems to be no understanding of the model of the friction parameters over time: even after revisiting previously seen parameters, the model does not adapt and improve past previous performance. This suggests that the model is not learning to generalize across different friction parameters.

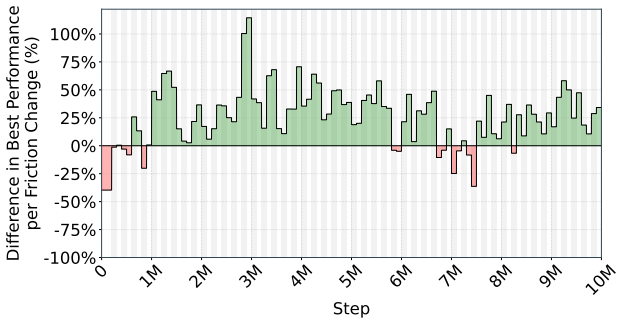

Figure 3. **Frozen Dynamics Experiment:** Percentage difference of frozen-dynamics model and baseline model performance. A positive percentage means the frozen-dynamics model performed better than the baseline. Both models had a horizon of 3.

Frozen Dynamics We then evaluate what happens when we freeze the model responsible for predicting the next state: After 200,000 steps, before the first friction parameters change, we freeze the dynamics model. We do not remove the corresponding world state consistency loss, which is instead backpropagated through the rest of the world model.

Interestingly, we find that freezing the dynamics model before the first friction change significantly improves the model performance (Figure 5). Without freezing the dynamics model, the responsibility for understanding dynamics becomes heavily focused on the dynamics model itself. However, the action model does not directly receive information from the dynamics model, only the world model. When we freeze the dynamics model, the world model is directly and immediately updated, which increases the pressure on the world state representation to contain more information about dynamics. This information is then passed directly to the action model, making it easier and faster for it to improve. However, we note two things:

1. Freezing the dynamics model still does not allow TD-MPC2 to learn from past experiences. The model still struggles when transitioning between friction settings.

2. The consistency loss is still required. Crucially, removing consistency loss completely after the first friction change results in performance equivalent to the baseline.

Friction Observation In this experiment, we pass the current friction parameters (f_s, f_d, f_r) as additional observation features to the model. We find this does not improve performance when compared against the baseline and is arguably worse (Figure 5). This suggests the model does not easily learn to associate the explicit friction parameters with

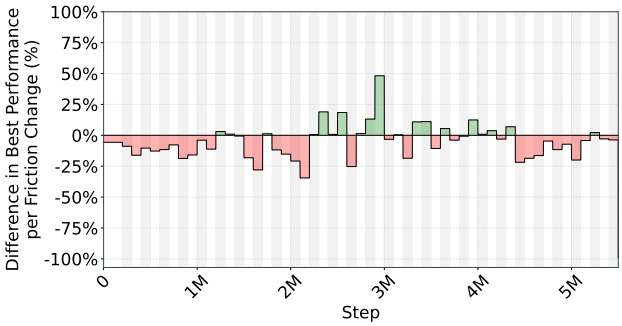

Figure 4. **Friction Observation Experiment:** Percentage difference of friction observation model and baseline model performance. A positive percentage means the friction observation model performed better than the baseline. Both models had a horizon of 3.

the understanding of the dynamics required to act differently under different friction settings.

5. Method

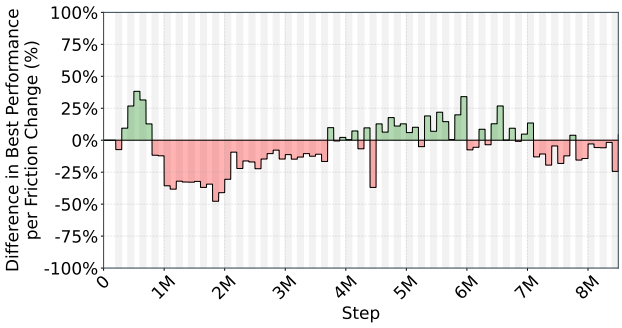

Figure 5. **Multi-Horizon State Prediction:** Percentage difference in performance between the Multi-Horizon model (using horizons of 5 and 10 steps) and the baseline model (horizon=10). Positive percentages indicates the Multi-Horizon model performed better than baseline.

Self-Supervised Latent State Consistency for Long-Horizons On top of TD-MPC2, we add additional dynamics models for simultaneously predicting latent states over multiple time horizons (e.g., $n = 5, 10$ steps ahead) while enforcing self-consistency of our latent states across all time horizons independently. This objective aligns the latent representations of the agent's world model with the underlying environment dynamics, reducing compounding errors and improving the quality of learned representations.

The prediction loss is weighted by $\rho^{(t+n)}$, where ρ is a discount factor, t represents the starting timestep, and n is the prediction horizon, prioritizing earlier predictions and shorter horizons while progressively refining long-term consistency. The world model—comprising an encoder $E(s)$, dynamics model $f(z_t, a_t)$, reward model $r(z_t, a_t)$, value model $V(z_t)$, and policy model $\pi(z_t)$—is trained jointly.

By enforcing self-consistency of latent spaces across several time horizons, our technique reduces error accumulation during the planning phase of TD-MPC2, enabling the agent to generate more accurate action sequences. Furthermore, the shared computation across predictions maintains computational efficiency while establishing a natural curriculum that improves the agent's ability to predict and plan over both short and extended horizons.

6. Conclusion

In this work, we investigate the state-of-the-art model-based reinforcement learning technique, TD-MPC2 (Hansen et al., 2024b), in a continual learning framework. We implemented and experimented with the Slippery Ant environment (Dohare et al., 2021) in Brax (Freeman et al., 2021) and MJX (Todorov et al., 2012). We explored the role of different components of TD-MPC2 contributing to its performance and found that while the dynamics modeling helps performance in a static environment, the decoupling of dynamics and world state representations hurts performance in a continual learning environment. In our Multi-horizon technique, we introduce modules specifically designed for long-horizon self-consistency, which we couple with next-state predictors. This increases the number of axes on which we apply self-consistency loss to better model the environment. However, we find such a method alone is insufficient to improve performance in the Slippery Ant environment. We hope our analyses of such experiments lead to a path for better continual model-based RL techniques.

References

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33: 1877–1901, 2020.

Dohare, S., Sutton, R. S., and Mahmood, A. R. Continual backprop: Stochastic gradient descent with persistent randomness. *arXiv preprint arXiv:2108.06325*, 2021.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity

is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Flandermeyer, S. tdmpc2-jax, 2024. URL https://github.com/ShaneFlandermeyer/tdmpc2-jax.

Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., and Bachem, O. Brax - a differentiable physics engine for large scale rigid body simulation, 2021. URL http://github.com/google/brax.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. Soft actor-critic algorithms and applications. *ArXiv*, abs/1812.05905, 2018.

Hansen, N., Lin, Y., Su, H., Wang, X., Kumar, V., and Rajeswaran, A. Modem: Accelerating visual model-based reinforcement learning with demonstrations. In *Published as a Conference Paper at ICLR*, 2024a.

Hansen, N., Su, H., and Wang, X. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2024b.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P., and Girshick, R. Segment anything. *arXiv:2304.02643*, 2023.

Laskin, M., Yarats, D., Liu, H., Lee, K., Zhan, A., Lu, K., Cang, C., Pinto, L., and Abbeel, P. Urlb: Unsupervised reinforcement learning benchmark. 2021.

Lillicrap, T., Hunt, J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2016.

Mayne, D. Q. Model predictive control: Recent developments and future promise. *Automatica*, 50(12):2967–2986, 2014. doi: 10.1016/j.automatica.2014.10.128.

Moerland, T. M., Broekens, J., and Jonker, C. M. Model-based reinforcement learning: A survey. *arXiv preprint arXiv:2006.16712*, 2020.

Mohamed, S. and Rezende, D. J. Variational information maximisation for intrinsically motivated reinforcement learning. *Advances in neural information processing systems*, 28, 2015.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.