TOLD-ZERO: Generalization of TD-MPC2 to Discrete Action Space

Naicheng He [*1] **Kaicheng Guo** [*1] **Wanjia Fu** [*1]

Abstract

TDMPC-2 is a model-based reinforcement learning algorithm on continuous action control. In this work we present TOLD-ZERO, a generalization of the original paper that specializes in discrete action tasks and uses Monte-Carlo Tree Search as a local trajectory optimization method. We will try to argue the role of planning in model-based RL in both continuous and discrete action tasks, specifically by bench-marking algorithms with or without planning on the LightZero Environment. [1]

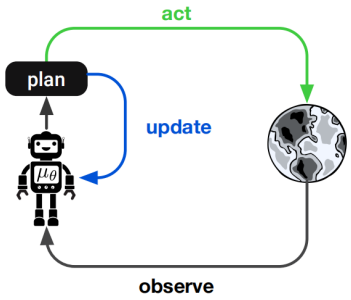


Figure 1. Model-based approximate policy iteration.

1. Introduction

Model-based Reinforcement Learning (Hamrick et al., 2021), featuring combinations of learning and planning in a variety of ways, has seen a lot of success in recent years. Significant advances have been made in many games on discrete and continuous action spaces where the artificial intelligent agent beats world-class human players. A lot of these algorithms, however, such as DQN (Mnih et al., 2013) and SAC (Haarnoja et al., 2018), have poor data efficiency and face challenges when dealing with a diverse range of tasks.

One family of algorithm that have shown great capabilities in solving complex discrete-action games with deep MCTS search is the Zeros starting from MuZero(Schrittwieser et al., 2020). From AlphaZero (Silver et al., 2017), EfficientZero (Ye et al., 2021), to Sampled EfficientZero (Wang et al., 2024), Gumbel MuZero (Kao et al., 2022), and Stochastic MuZero (Antonoglou et al., 2022). They perform a wide variety of tasks such as Go, CartPole, Pendulum, and Atari. Although AlphaZero and MuZero only work for discrete tasks, Sampled EfficientZero also works for continuous tasks by sampling actions.

As illustrated in Figure 1, all of the MBRL algorithms consist of a world model and a local trajectory optimization algorithm that serves as a policy prior and a target for learning. They can also be classified into *decision time* planning, which refers to using the model to select a policy, and *background* planning, which refers to using the model to update a policy. The role of policy search is key to constructing targets in policy learning, and for generating more informative data distribution (Hamrick et al., 2021).

The Zero-family algorithms use variations of Monte-Carlo Tree Search to as a policy guidance, while in TDMPC-2 (Hansen et al., 2024), MPPI (Model Predictive Path Integral) plays this local trajectory optimization role. (Hamrick et al., 2021) investigates the role of mcts in Muzero algorithm. A natural question to ask is, can we experiment on the role of lanning with MPPI for TD-MPC2?

1.1. On the role of planning in tdmpc-2

In the first part of our project, we benchmark TD-MPC2 by running it on Pendulum, Bipedal Walker, and Dog Run with or without MPPI. We also did ablations by changing the horizon length $\mathcal{H}$ and removing the terminal value function. We finally compared the benchmarked performance of Sampled EfficientZero and TD-MPC2 on Pendulum. On the other hand, TD-MPC2 performs very well on a wide range of complex continuous control tasks, but making it work for discrete action spaces is still an open problem. However, given the comparative data efficiency and high performance of MCTS, we are interested in incorporating the TOLD world model in TD-MPC2 with MCTS.

1.2. TOLD-ZERO

We thus try to generalize TDMPC-2 to discrete action space tasks. We do this to ultimately achieve a cross-task ability of playing games like Atari. The key difference will be its

[1] Code available at tdmpc2-discrete

ability to solve multiple tasks with one set of parameters and we've found that the convergence rate of TOLD-ZERO on tasks such as Cartpole is faster than all the Zero-family algorithms.

2. Related Work

For the main sections of our project, which are to investigate the significance of MPPI in TD-MPC2 and to generalize TD-MPC2 to discrete action spaces, we need to reprsent two pieces of work that are crucial for readers' understanding.

The first one is TD-MPC2, which stands for Temporal Difference Learning for Model Predictive Control. It utilizes an MPC planning algorithm that takes into account short term action planning trajectory and long term terminal value function.

The other is MuZero, which incorporates Monte Carlo Tree Search in model-based reinforcement learning for discrete action space tasks. This serves as a potentially powerful planning algorithm which shows better data efficiency and performance.

2.1. TD-MPC2

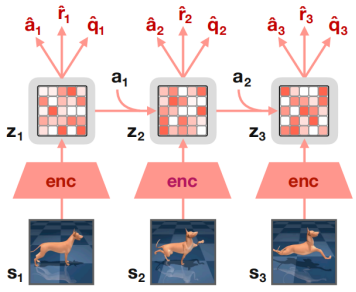


Figure 2. TD-MPC2 TOLD Model Architecture

As shown in Algorithm 2.1, for a given horizon length, MPPI (Model Predictive Path Integral) (Hansen et al., 2022) will sample actions until the end of that trajectory, which is a planning in the short term. It also takes into account planning in the long term by adding a terminal value function into

$$\phi_\Gamma \triangleq \mathbb{E}_\Gamma \left[\gamma^H Q_\theta(\mathbf{z}_H, \mathbf{a}_H) + \sum_{t=0}^{H-1} \gamma^t R_\theta(\mathbf{z}_t, \mathbf{a}_t) \right], \quad (1)$$

It will then update the mean and standard deviation from which to sample future actions, using the following equations:

Algorithm 1 TD-MPC (*inference*)

Require: θ : learned network parameters
μ^0, σ^0: initial parameters for $\mathcal{N}$
N, N_π: num sample/policy trajectories
$\mathbf{s}_t, H$: current state, rollout horizon

Encode state $\mathbf{z}_t \leftarrow h_\theta(\mathbf{s}_t)$ ◁ *Assuming TOLD model*
for each iteration $j = 1..J$ **do**
  Sample N traj. of len. H from $\mathcal{N}(\mu^{j-1}, (\sigma^{j-1})^2\mathrm{I})$
  Sample N_π traj. of length H using π_θ, d_θ
  // Estimate trajectory returns ϕ_Γ using $d_\theta, R_\theta, Q_\theta$, starting from $\mathbf{z}_t$ and initially letting $\phi_\Gamma = 0$:
  for all $N + N_\pi$ trajectories $(\mathbf{a}_t, \mathbf{a}_{t+1}, \ldots, \mathbf{a}_{t+H})$ **do**
    for step $t = 0..H-1$ **do**
      $\phi_\Gamma = \phi_\Gamma + \gamma^t R_\theta(\mathbf{z}_t, \mathbf{a}_t)$ ◁ *Reward*
      $\mathbf{z}_{t+1} \leftarrow d_\theta(\mathbf{z}_t, \mathbf{a}_t)$ ◁ *Latent transition*
    end for
    $\phi_\Gamma = \phi_\Gamma + \gamma^H Q_\theta(\mathbf{z}_H, \mathbf{a}_H)$ ◁ *Terminal value*
  end for *// Update parameters μ, σ for next iteration:*
  $\mu^j, \sigma^j =$ Equation 2
end for
return $\mathbf{a} \sim \mathcal{N}(\mu^J, (\sigma^J)^2\mathrm{I})$

$$\mu^j = \frac{\sum_{i=1}^k \Omega_i \Gamma_i^\star}{\sum_{i=1}^k \Omega_i}, \quad \sigma^j = \sqrt{\frac{\sum_{i=1}^k \Omega_i (\Gamma_i^\star - \mu^j)^2}{\sum_{i=1}^k \Omega_i}}, \quad (2)$$

2.2. MuZero

MuZero (Schrittwieser et al., 2020) is one of the MBRL (Model Based Reinforcement Learning) algorithms that incorporates MTCS (Monte Carlo Tree Search).

It consists of four components: model, search, acting, and learning. (Hamrick et al., 2021)

Within the MTCS framework, MuZero learns three functions at the same time. They are the learned policy function which provides a policy for selecting action at each state; the learned value function, which approximates the value at each state; and the immediate reward function.

2.2.1. Model

MuZero plans in a hidden state. The model μ_θ is parametrized with the parameter θ. At each time step, the model is represented by a combination of three functions: a representation function, a dynamics function, and a prediction function.

The representation function encodes past observations.

The dynamics function predicts an immediate reward r^k and an internal state s^k based on past observations and current action.

The prediction function predicts a policy and generates a hidden state that will be pased in the model.

This part is illustrated in part B of Figure 3.

2.2.2. SEARCH

Search for MuZero involves the MCTS. Beginning at a root node s^0, a simulation is carried out based on a search policy until a previously unexplored node is reached. The search policy also leverages exploration and exploration, and picks actions based on the number of counts that this node has been reached. The value and reward received at the last node will then propagate backwards to update the value and reward of the root.

This is illustrated in part A of Figure 3.

2.2.3. ACTING

After Search, an action is sampled from the search policy. After the reward is received, we will put the observation, state, reward, and action into a replay buffer.

2.2.4. LEARNING

The model is trained jointly to predict the reward, value, and policy for future timesteps. MuZero will not unnecessarily predict future hidden states and observations, so this increases data efficiency for such a model-based reinforcement learning method.

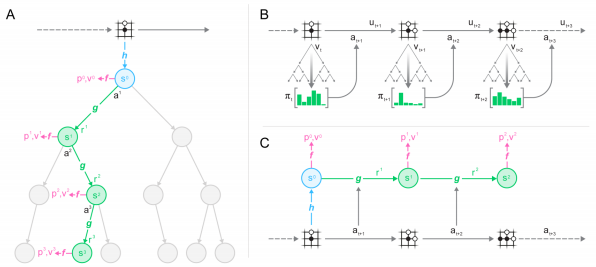


Figure 3. Planning, acting, and training in MuZero

3. Benchmarking

LightZero(Niu et al., 2023) is a unified benchmarking framework for the "Zero" family algorithms. These algorithms incorporate MCTS (Monte Carlo Tree Search) with reinforcement learning, and can work across a variety of task on disrete and continuous actions spaces.

TD-MPC2 (Hansen et al., 2024), in comparison, uses MPPI (Model Predictive Path Integral) instead of MCTS. As one of the MPC (Model Predictive Control) algorithms, MPPI involves a horizon and a terminal value function, which thus plans differently from MCTS. In addition, it works only for tasks on continuous action spaces.

In this section, we will present the benchmarking of TD-MPC2 on LightZero framework on continuous tasks.

3.1. Methodology

3.1.1. ENVIRONMENT

The LightZero framework uses the Ding Wrapper, while the TD-MPC2 tasks are in the Open-AI gym environments. We hereby adaped TD-MPC2 to work in Ding-compatible environments in order to compare its performance with Zero-family algorithms.

3.1.2. TASKS

For benchmarking, we mainly focused on three of the tasks on continuous action spaces: 1) Pendulum; 2) Bipedal Walker; 3) Dog Run (Mujoco). All three are very classic planning tasks used to implement benchmarking.

3.1.3. ABLATIONS

In terms of ablations, we focused on three experiments.

The first one is comparing with and without MPPI. This is carried out on Pendulum, Bipedal Walker, and Dog Run.

The second ablation is comparing different horizon lengths, while also taking out the terminal value function from the bellman equation. This is only run on the Pendulum task.

The third ablation is the comparison between TD-MPC2 and Sampled EfficientZero on the continuous task of Pendulum.

3.2. Results

3.2.1. MPPI

As shown in top left of Figure 3, the task of Pendulum converges at around 0.0026 million environmental steps both when MPPI is and is not used.

When the trajectory is planned using MPPI, the returns will reach values closer to 0, but overall in terms of the stability of returns after convergence, the performance of TD-MPC2 on the task of Pendulum with or without MPPI is very similar.

Similar trends can be observed in the top right and bottom left of Figure 3, which are run on Bipedal Walker and Dog Run, respectively.

Therefore, we've shown that MPPI is not a very well performing planning algorithm for complex continuous control tasks. The reason why TD-MPC2 has high performance can be attributed to other reasons which will be analyzed later.

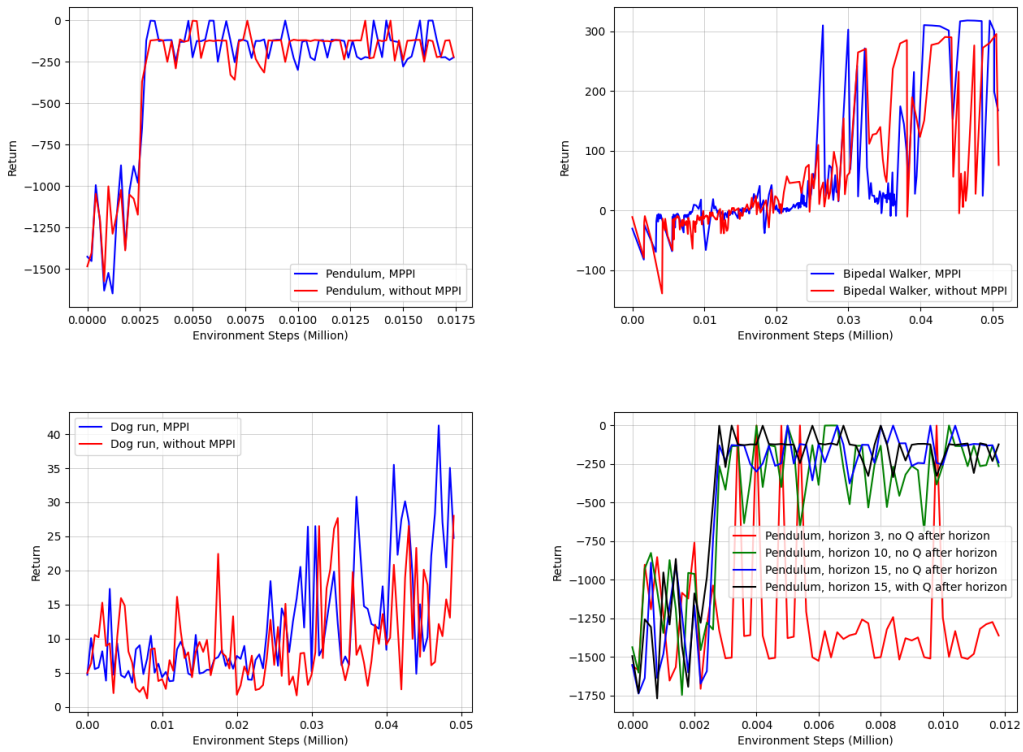


Figure 4. The first three plots are the benchmarking of TD-MPC2 with or without MPPI on Pendulum (top left), Bipedal Walker (top right), and Dog Run (bottom left), respectively. The last plot (bottom right) is the benchmarking of TD-MPC2 with different horizon lengths and with or without terminal value function Q.

As shown in the bottom right of Figure 3, we can see that when horizon is very small, i.e. 3, and we don't consider the terminal value condition, then the performance is very unstable and the agent fails to converge on an optimal policy. Otherwise, the difference between whether the terminal value condition is considered or not doesn't influence the ultimate convergence performance.

Given that the terminal value function is an important part of the MPPI algorithm, this suggests again that MPPI doesn't improve performance much.

3.2.2. Sampled EfficientZero vs TD-MPC2

From Figure B, the blue curve corresponds to the performance of TD-MPC2 without MPPI on Pendulum, and the red curve corresponds to the performance of Sampled EfficientZero on Pendulum.

We can see that TD-MPC2 converges a lot faster to a return very close to zero even without incorporating MPPI algorithm.

This is very likely due to the large number of trainable parameters in the TD-MPC2 model.

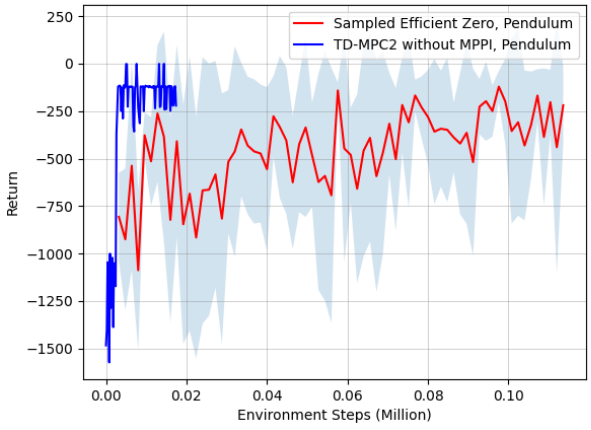


Figure 5. Comparison of benchmarking performance of Sampled EfficientZero and TD-MPC2 on Pendulum. The shaded areas correspond to one standard deviation above and below average returns.

4. TOLD-ZERO

TOLD-ZERO is a model-based RL algorithm that generalize TDMPC-2(Hansen et al., 2024) to discrete action space. Our approach integrates the Task-Oriented Latent Dynamics(TOLD) model as a world model, with the MCTS(Schrittwieser et al., 2019) search algorithm. In the following sections, we will first provide an overview of the TOLD model. We will change the policy update rule and policy network to adapt to discrete action tasks. An integration of MCTS as a local trajectory optimization method is shown to be necessary later.

4.1. Task-Oriented Latent Dynamics model

In this section, we provide a brief introduction to the Task-Oriented Latent Dynamics (TOLD) model, focusing particularly on components relevant to adapting it for discrete action settings.

Task-Oriented Latent Dynamics(Hansen et al., 2022) is designed to only model elements of the environment that are predictive of reward. Compare with other model such as Dreamer-V3(Hafner et al., 2023) attempting to model the environment itself, TOLD model could focus more on the reward-related information without the distractions from the image details.

Components: TOLD model consists of five components h_θ, d_θ, R_θ, Q_θ, π_θ and predict the following:

$$\begin{aligned} \text{Representation:} \quad & z_t = h_\theta(s_t) \\ \text{Latent dynamics:} \quad & z_{t+1} = d_\theta(z_t, a_t) \\ \text{Reward:} \quad & r'_t = R_\theta(z_t, a_t) \\ \text{Value:} \quad & q'_t = Q_\theta(z_t, a_t) \\ \text{Policy:} \quad & a'_t \sim \pi_\theta(z_t) \end{aligned}$$

We optimize our policy by maximizing the following equation. It is similar to one of SAC's in the sense of it incorporates entropy.

Policy objective.(Hansen et al., 2024) The policy prior p is a stochastic maximum entropy policy that learns to maximize the objective

$$L_p(\theta) \coloneqq \mathbb{E}_{(s,a)\sim\mathcal{B}}\left[\sum_{t=0}^{H} \lambda^t \left(Q_\theta(z_t, p(z_t)) - \beta H(p(\cdot|z_t))\right)\right]$$
$$z_{t+1} = d_\theta(z_t, a_t), z_0 = h_\theta(s_0)$$

Algorithm: The TOLD training algorithm in TDMPC can be found in Appendix A. (Hansen et al., 2022)

4.2. Task-Oriented Latent Dynamics model in discrete action settings

We now derive a discrete action version of the above TOLD model. One thing to note is that policy objective, full objective in TDMPC-2 still holds in discrete action settings. We need to make the following two important changes to the process of optimising these objective functions:

1. **Policy Representation**: Rather than output mean and covariance, the policy network is changed to output probability distribution of all possible actions by adding a softmax layer. The policy function changes from $\pi : S \to \mu, \sigma$ to $\pi : S \to [0, 1]^A$

2. **Policy Update**: Calculating entropy in the policy loss function should also be different. The entropy $H(\pi(\cdot|s_t))$ in a discrete action space is calculated using the probabilities from the softmax output:

$$H(\pi(\cdot|s_t)) = -\sum_a \pi(a|s_t) \log \pi(a|s_t)$$

Then plug in entropy term to policy objective, we can derive the new policy objective.

4.3. A Bare-TOLD in discrete action result

We reimplement TOLD model in jax and adapt the above changes. We then run the algorithm on discrete action environment Cartpole-v0. We obtain the following result:

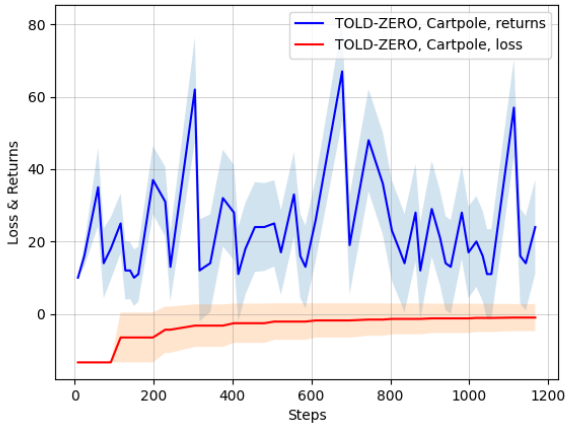


Figure 6. This figure shows the performance of the TOLD model in the Cartpole-v0 environment. We observe that the model is very unstable and reward does not behave as expected.

From the graph, we can conclude the following: The policy implemented in the TOLD model struggles with stability in the discrete action space and have minimal improvement

in actual performance over time. This suggests that while the policy is capable of generating actions, these actions are not effectively leading to the maximization of long-term rewards. There are potential issuses in policy convergence, which could be due to inadequate exploration, the sensitivity of the policy gradient to stochasticity in the environment, or possibly suboptimal hyperparameter settings.

4.4. MCTS as a Policy Improvement Tool

One feature of Monte Carlo Tree Search based algorithm is its ability to perform "precise and sophisticated lookahead". From (Hamrick et al., 2021), we can conclude that this lookahead support learning stronger policies. Therefore, we incorporate Muzero(Schrittwieser et al., 2019) planning stage to TOLD model, which serves as a policy improvement tool.

As detailed in Section 2, we know Muzero has four components: Model, Search, Acting, Learning. In our adaptation, we only incorporate the Search and Acting components into the TOLD model. This selective integration allows us to enhance the policy network without altering the TOLD model's internal structure.

4.5. TOLD-ZERO result

In this section, we present the performance of the TOLD-ZERO model on discrete action environments, specifically focusing on the Cartpole-v0 environment. The results are summarized in Figure 7, which tracks the model's learning progress over 1200 steps.

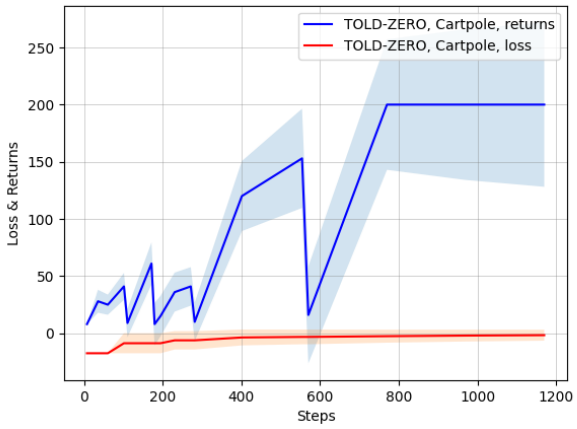


Figure 7. This figure shows the performance of the TOLD model in the Cartpole-v0 environment. We observe that the model converge at approximate 800 steps.

Conclusion:

1. It's clear that the returns increase significantly and tend to stabilize around 800 steps, indicating that the model is achieving higher scores in the task as it learns.
2. The loss line, on the other hand, starts relatively low and slightly decreases as training progresses. This pattern indicates that the model is consistently refining its predictions and minimizing discrepancies between its predictions and the actual outcomes.

Therefore, we can conclude that the performance of the TOLD-ZERO model in the Cartpole-v0 environment is quite promising. We also show result of Cartpole-v1 in Appendix A, which verify our conclusion.

5. Conclusion

In this work, we explored MBRL (Model-Based Reinforcement Learning) and planning algorithms based on MTCS (Monte Carlo Tree Search).

First, we investigated the value of MPPI by benchmarking TD-MPC2 on LightZero framework continuous tasks. By doing ablations with and without MPPI, changing horizon and removing terminal value function, as well as comparing with Sampled EfficientZero, we've shown that MPPI doesn't necessarily improve the performance of MPC on relatively simple continuous tasks such as Pendulum, Bipedal Walker, and Dog Run.

Next, we present TOLD-ZERO, which is our generalization of TD-MPC2 to discrete action space by combining MCTS with TOLD world model. Our new algorithm has been proved to result in convergence on Cartpole within 25 episodes, which is better than all the Zero-family algorithms.

6. Future Directions

The future direction of our work is gonna be two-fold. First regarding our investigation on the role of MPPI on tdmpc-2, we will have to perform experiments on more complex environments like dm-control and mujoco to get a better understanding in more complex environments. We suspect that the role of MPPI will get more important in more complex tasks. Secondly, TOLD-ZERO is proven to work on Cartpole environments without any hyper parameter tuning, but we are still left to do more complex tasks like Vision-based Cartpole, Atari, or Go. The latter was especially challenging because of the computation power it requires.

References

Antonoglou, I., Schrittwieser, J., Ozair, S., Hubert, T. K., and Silver, D. Planning in stochastic environments with a learned model. In *International Conference on Learning*

Representations, 2022. URL https://openreview.net/forum?id=X6D9bAHhBQ1.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Hamrick, J. B., Friesen, A. L., Behbahani, F., Guez, A., Viola, F., Witherspoon, S., Anthony, T., Buesing, L., Veličković, P., and Weber, T. On the role of planning in model-based deep reinforcement learning, 2021.

Hansen, N., Wang, X., and Su, H. Temporal difference learning for model predictive control. 2022.

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

Kao, C.-Y., Guei, H., Wu, T.-R., and Wu, I.-C. Gumbel muzero for the game of 2048. In *2022 International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pp. 42–47, 2022. doi: 10.1109/TAAI57707.2022.00017.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning, 2013.

Niu, Y., Pu, Y., Yang, Z., Li, X., Zhou, T., Ren, J., Hu, S., Li, H., and Liu, Y. Lightzero: A unified benchmark for monte carlo tree search in general sequential decision scenarios, 2023.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T. P., and Silver, D. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588:604 – 609, 2019. URL https://api.semanticscholar.org/CorpusID:208158225.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T., and Silver, D. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, December 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-03051-4. URL http://dx.doi.org/10.1038/s41586-020-03051-4.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017.

Wang, S., Liu, S., Ye, W., You, J., and Gao, Y. Efficientzero v2: Mastering discrete and continuous control with limited data, 2024.

Ye, W., Liu, S., Kurutach, T., Abbeel, P., and Gao, Y. Mastering atari games with limited data, 2021.

A. TOLD algorithm

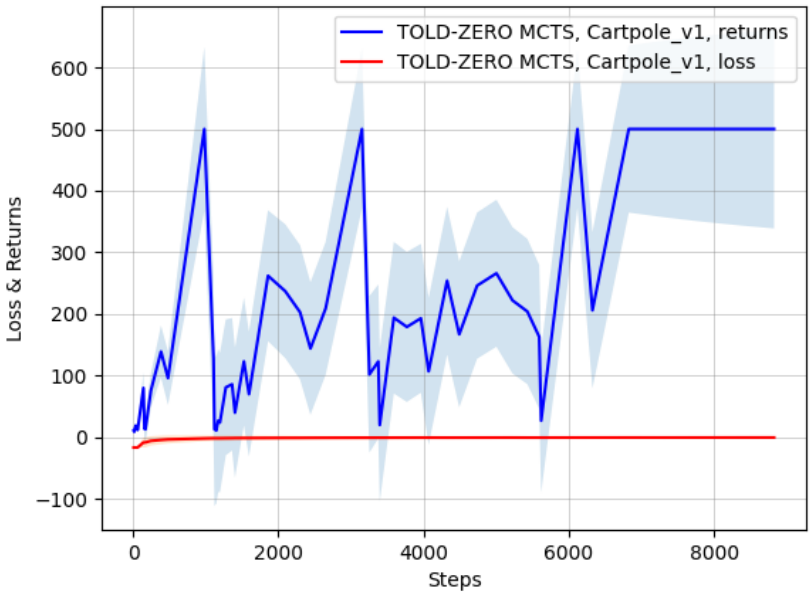
This algorithm is taken from (Hansen et al., 2022)

Algorithm 2 TOLD (training)

Require: θ, θ^-: randomly initialized network parameters $\eta, \tau, \lambda, \beta$: learning rate, coefficients, buffer

while not tired **do**
  // Collect episode with TD-MPC from $s_0 \sim p_0$:
  for step $t = 0\ T$ **do**
    $a_t \sim \Pi_\theta(\cdot|h_\theta(s_t))$
    $(s_{t+1}, r_t) \sim T(\cdot|s_t, a_t), R(\cdot|s_t, a_t)$ {Step environment}
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(s_t, a_t, r_t, s_{t+1})\}$ {Add to buffer}
  end for
  // Update TOLD using collected data in $\mathcal{B}$:
  for num updates per episode **do**
    $\{(s_t, a_t, r_t, s_{t+1})_{t:t+H}\} \sim \mathcal{B}$ {Sample trajectory}
    $z_t = h_\theta(s_t)$ {Encode first observation}
    $J = 0$ {Initialize J for loss accumulation}
    for $i = t\ t + H$ **do**
      $\tilde{r}_i = R_\theta(z_i, a_i)$ {Equation 8}
      $\tilde{q}_i = Q_\theta(z_i, a_i)$ {Equation 9}
      $z_{i+1} = d_\theta(z_i, a_i)$ {Equation 10}
      $\tilde{a}_i = \pi_\theta(z_i)$ {Equation 11}
      $J \leftarrow J + \lambda^i C(z_{i+1}, \tilde{r}_i, \tilde{q}_i, \tilde{a}_i)$ {Equation 7}
    end for
    $\theta \leftarrow \theta - \frac{1}{H}\eta\nabla_\theta J$ {Update online network}
    $\theta^- \leftarrow (1 - \tau)\theta^- + \tau\theta$ {Update target network}
  end for
end while

B. Cartpole-v1 result



C. TOLD-ZERO implementation details

Table 1. Configuration Parameters

Parameter	**Value**
Seed	0
Max Steps	250,000
Encoder	
Encoder Dim	256
Number of Encoder Layers	2
World Model	
MLP Dim	512
Latent Dim	512
Value Dropout	0.01
Number of Value Nets	5
Number of Bins	101
Symlog Min	-10
Symlog Max	10
Simnorm Dim	8
Learning Rate	3×10^{-4}
Encoder Learning Rate	1×10^{-4}
Predict Continues	False
Data Type	bfloat16
Tabulate	False
TDMPC2	
MPC	False
MCTS	False
Horizon	3
MPPI Iterations	6
Population Size	512
Policy Prior Samples	32
Number of Elites	64
Min Plan Std	0.05
Max Plan Std	2
Temperature	0.5
Batch Size	256
Discount	0.99
Rho	0.5
Consistency Coefficient	20
Reward Coefficient	0.1
Continue Coefficient	0.1
Value Coefficient	0.1
Entropy Coefficient	1×10^{-3}
Tau	0.01