KOREATECH
한국기술교육대학교

Temporal Difference Learning for Model Predictive Control

Nicklas Hansen, Xiaolong Wang, Hao Su
UC San Diego

Chang-Hun Ji
2022. 4. 13
LINK@KoreaTech
http://link.koreatech.ac.kr

Abstract

◈ Data-driven model predictive control

- Advantages
  - Improving sample efficiency through model learning
  - Better performance as computational budget for planning increases
- Challenges
  - High cost to plan over long horizons
  - Obtaining an accurate model of the environment

◈ Combine the strengths of model-free and model-based methods

Task-oriented latent dynamics model

+

Model Predictive Control
(short horizon, model-based methods)

+

Terminal value function
(long term return, model-free methods)

➡

TD-MPC
(Temporal Difference learning for Model Predictive Control)

Introduction

- Model-based RL
  - Planning
    - Prohibitively expensive to plan over long horizons
  - Using a learned model to improve sample-efficiency
    - Model biases likely to propagate to the policy

Can we instead augment model-based planning with the strengths of model-free learning?

Preliminaries

◈ Model Predictive Control(MPC)

$$\Pi_{\theta}^{\text{MPC}}(\mathbf{s}_t) = \arg \max_{\mathbf{a}_{t:t+H}} \mathbb{E}\left[\sum_{i=t}^{H} \gamma^i \mathcal{R}(\mathbf{s}_i, \mathbf{a}_i)\right]$$

- Π is traditionally implemented as a trajectory optimization procedure
- To make the problem tractable, one typically obtains a local solution
  - local solution: Estimating optimal actions $a_{t:t+H}$ over a finite horizon H
  - Executing the first action a_t
- γ is typically set to 1
- A solution can be found by iteratively fitting parameters of a family of distributions
  - Parameters: μ, σ for a multivariate Gaussian with diagonal covariance
  - Using the derivative-free Cross Entropy Method (CEM; Rubinstein (1997))
  - Sample trajectories generated by a model

Preliminaries

❖ Model Predictive Control(MPC)

$$\Pi_\theta^{\text{MPC}}(\mathbf{s}_t) = \arg\max_{\mathbf{a}_{t:t+H}} \mathbb{E}\left[\sum_{i=t}^{H} \gamma^i \mathcal{R}(\mathbf{s}_i, \mathbf{a}_i)\right]$$

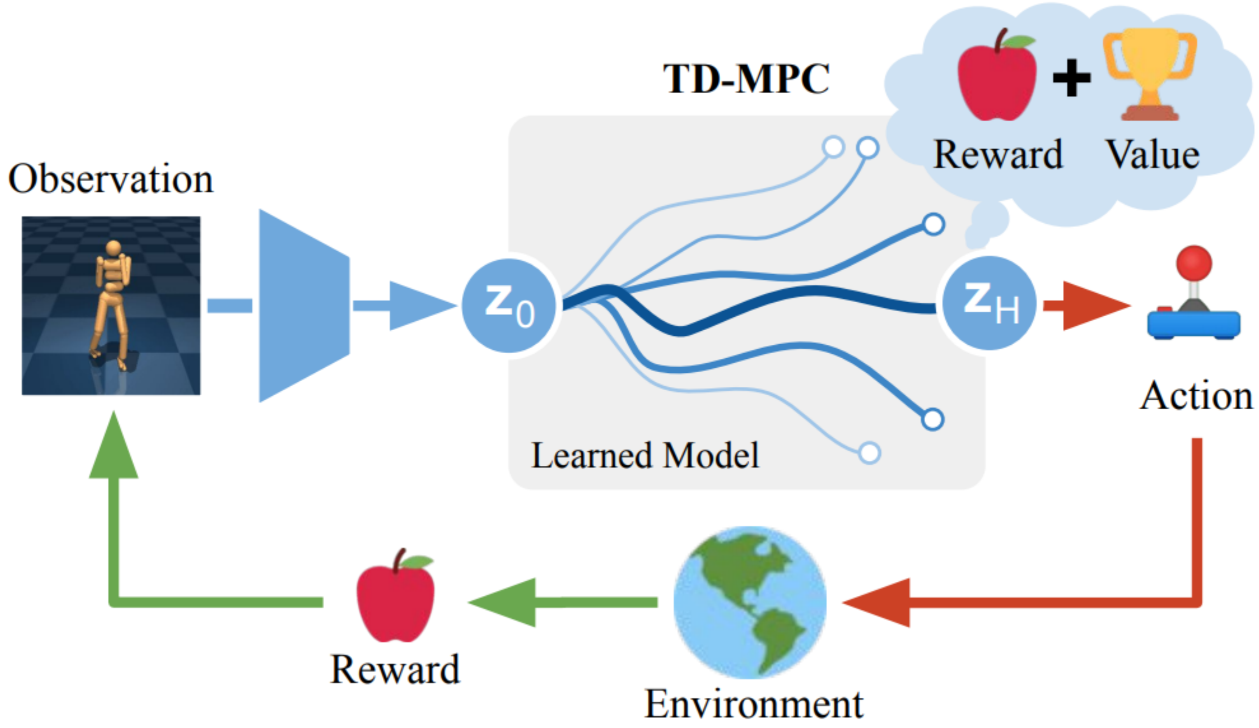
- As opposed to fitted Q-iteration, MPC is not predictive of long-term rewards
- When a value function is known, it can be used in conjunction with MPC to estimate discounted return at state s_{t+H} and beyond

TD-Learning for Model Predictive Control

- TD-MPC
  - A framework that combines MPC with a task-oriented latent dynamics model and terminal value function jointly learned using TD-learning in an online RL setting.
  - Components, Notation
    - Control for planning
      - Model Predictive Path Integral (MPPI; Williams et al. (2015))
      - Π_θ
    - Task-Oriented Latent Dynamics Model

$$
\begin{aligned}
&\text{Representation:} && \mathbf{z}_t = h_\theta(\mathbf{s}_t) \\
&\text{Latent dynamics:} && \mathbf{z}_{t+1} = d_\theta(\mathbf{z}_t, \mathbf{a}_t) \\
&\text{Reward:} && \hat{r}_t = R_\theta(\mathbf{z}_t, \mathbf{a}_t) \\
&\text{Value:} && \hat{q}_t = Q_\theta(\mathbf{z}_t, \mathbf{a}_t)
\end{aligned}
$$

    - A parameterized policy
      - π_θ
    - Sampled trajectory
      - Γ
    - Total return of Γ
      - ϕ_Γ



TD-Learning for Model Predictive Control

◈ TD-MPC

Algorithm 1 TD-MPC *(inference)*

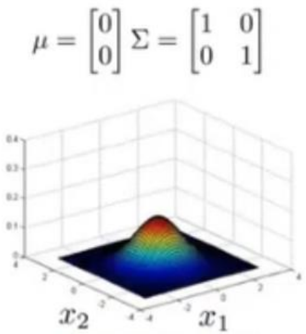
Require: θ : learned network parameters
μ^0, σ^0: initial parameters for $\mathcal{N}$
N, N_π: num sample/policy trajectories
$\mathbf{s}_t, H$: current state, rollout horizon

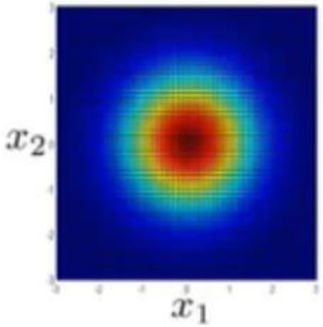
```
Encode state z_t ← h_θ(s_t)                                  ◁ Assuming TOLD model
for each iteration j = 1..J do
    Sample N traj. of len. H from N(μ^{j−1}, (σ^{j−1})^2 I)
    Sample N_π traj. of length H using π_θ, d_θ
    // Estimate trajectory returns φ_Γ using d_θ, R_θ, Q_θ,
      starting from z_t and initially letting φ_Γ = 0:
    for all N + N_π trajectories (a_t, a_{t+1}, ..., a_{t+H}) do
        for step t = 0..H − 1 do
            φ_Γ = φ_Γ + γ^t R_θ(z_t, a_t)                    ◁ Reward
            z_{t+1} ← d_θ(z_t, a_t)                          ◁ Latent transition
        φ_Γ = φ_Γ + γ^H Q_θ(z_H, a_H)                        ◁ Terminal value
    // Update parameters μ, σ for next iteration:
    μ^j, σ^j = Equation 4 (and Equation 5)
return a ∼ N(μ^J, (σ^J)^2 I)
```

TD-Learning for Model Predictive Control

◈ Model Predictive Path Integral(MPPI)

- An MPC algorithm that iteratively updates parameters for a family of distributions
- MPPI's updatable parameters
  - A time-dependent multivariate Gaussian with diagonal covariance's means, standard deviations
    - ➢ μ, σ





  - Starting from initial parameters

$$(\mu^0, \sigma^0)_{t:t+H},\ \mu^0, \sigma^0 \in \mathbb{R}^m,\ \mathcal{A} \in \mathbb{R}^m$$

    - ➢ independent parameters for each action over a horizon of length H
- Sampling action

$$\mathbf{a}_t \sim \mathcal{N}(\mu_t^{j-1}, (\sigma_t^{j-1})^2 \mathrm{I})$$

TD-Learning for Model Predictive Control

- TD-MPC
  - Independently sample N trajectories using rollouts generated by the learned model d_θ
  - estimate the total return ϕ_Γ of a sampled trajectory Γ

Task-oriented latent dynamics model

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

Long term return,
Terminal value function,
Model-free methods

MPC,
short horizon,
model-based methods

Task-oriented latent dynamics model

+

Model Predictive Control
(short horizon, model-based methods)

+

Terminal value function
(long term return, model-free methods)

TD-MPC
(Temporal Difference learning for Model Predictive Control)

TD-Learning for Model Predictive Control

◈ TD-MPC

– Updates parameters

1) Select top-k returns $\emptyset_{\Gamma}^{*}$
2) Obtain new parameters μ_j , σ_j at iteration j from a $\emptyset_{\Gamma}^{*}$-normalized empirical estimate

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

,where $\Omega_i = e^{\tau(\phi_{\Gamma,i}^\star - \max_g(\phi_{\Gamma,g}^\star))}$

τ = temperature parameter

Γ_i^* = ith top-k trajectory

3) After a fixed number of iterations J, the planning procedure is terminated
4) A trajectory(action) is sampled from the final return-normalized distribution over action sequences.

– Plan at each decision step t and execute only the first action

- Employ receding-horizon MPC to produce a feedback policy

TD-Learning for Model Predictive Control

❖ TD-MPC

– Warm start
  - Trajectory optimization at each step t by reusing the 1-step shifted mean μ obtained at the previous step
  - Always use a large initial variance to avoid local minima

– Exploration on planning
  - We find that the rate at which σ decays varies wildly between tasks, leading to (potentially poor) local optima for small σ

$$\sigma^j = \max\left(\sqrt{\frac{\sum_{i=1}^{N} \Omega_i (\Gamma_i^\star - \mu^j)^2}{\sum_{i=1}^{N} \Omega_i}},\ \epsilon\right)$$

  - Linearly increase the planning horizon from 1 to H

– Policy-guided trajectory optimization
  - We augment the CEM sampling with additional samples from π_θ

TD-Learning for Model Predictive Control

- TD-MPC

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 $\mathbf{z}_t \leftarrow h_\theta(\mathbf{s}_t)$ $\triangleleft$ *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 $(\mathbf{a}_t, \mathbf{a}_{t+1}, \dots, \mathbf{a}_{t+H})$ **do**
    for step $t = 0..H-1$ **do**
      $\phi_\Gamma = \phi_\Gamma + \gamma^t R_\theta(\mathbf{z}_t, \mathbf{a}_t)$ $\triangleleft$ *Reward*
      $\mathbf{z}_{t+1} \leftarrow d_\theta(\mathbf{z}_t, \mathbf{a}_t)$ $\triangleleft$ *Latent transition*
    $\phi_\Gamma = \phi_\Gamma + \gamma^H Q_\theta(\mathbf{z}_H, \mathbf{a}_H)$ $\triangleleft$ *Terminal value*
  // Update parameters μ, σ for next iteration:
  μ^j, σ^j = Equation 4 (and Equation 5)
return $\mathbf{a} \sim \mathcal{N}(\mu^J, (\sigma^J)^2\mathrm{I})$

Task-Oriented Latent Dynamics Model

❖ Task-Oriented Latent Dynamics (TOLD)

- Jointly learned together with a terminal value function using TD-learning
- Rather than attempting to model the environment itself, our TOLD model learns to only model elements of the environment that are predictive of reward, which is a far easier problem.
- Components

$$
\begin{array}{ll}
\text{Representation:} & \mathbf{z}_t = h_\theta(\mathbf{s}_t) \\
\text{Latent dynamics:} & \mathbf{z}_{t+1} = d_\theta(\mathbf{z}_t, \mathbf{a}_t) \\
\text{Reward:} & \hat{r}_t = R_\theta(\mathbf{z}_t, \mathbf{a}_t) \\
\text{Value:} & \hat{q}_t = Q_\theta(\mathbf{z}_t, \mathbf{a}_t) \\
\text{Policy:} & \hat{\mathbf{a}}_t \sim \pi_\theta(\mathbf{z}_t)
\end{array}
$$

- We find it sufficient to implement all components of TOLD as purely deterministic MLPs.

Task-Oriented Latent Dynamics Model

- Task-Oriented Latent Dynamics (TOLD)
  - Objective for prediction of representation, latent dynamics, reward, value

$$\mathcal{J}(\theta;\Gamma) = \sum_{i=t}^{t+H} \lambda^{i-t} \mathcal{L}(\theta;\Gamma_i)$$

$$\begin{aligned}\mathcal{L}(\theta;\Gamma_i) = & \underbrace{c_1 \|R_\theta(\mathbf{z}_i,\mathbf{a}_i) - r_i\|_2^2}_{\text{reward}} \\ & + \underbrace{c_2 \|Q_\theta(\mathbf{z}_i,\mathbf{a}_i) - (r_i + \gamma Q_{\theta^-}(\mathbf{z}_{i+1},\pi_\theta(\mathbf{z}_{i+1})))\|_2^2}_{\text{value}} \\ & + \underbrace{c_3 \|d_\theta(\mathbf{z}_i,\mathbf{a}_i) - h_{\theta^-}(\mathbf{s}_{i+1})\|_2^2}_{\text{latent state consistency}}\end{aligned}$$

If we use max Q value by planning,
the computation's cost is extremely high

$$\max_{\mathbf{a}_t} Q_{\theta^-}(\mathbf{z}_t,\mathbf{a}_t)$$

Task-Oriented Latent Dynamics Model

- Task-Oriented Latent Dynamics (TOLD)
  - Objective for policy

$$\mathcal{J}_\pi(\theta;\Gamma) = -\sum_{i=t}^{t+H} \lambda^{i-t} Q_\theta(\mathbf{z}_i, \pi_\theta(\mathrm{sg}(\mathbf{z}_i)))$$

Task-Oriented Latent Dynamics Model

◈ TOLD

Algorithm 2 TOLD *(training)*

Require: θ, θ^-: randomly initialized network parameters
$\eta, \tau, \lambda, \mathcal{B}$: learning rate, coefficients, buffer

while not tired **do**
  // Collect episode with TD-MPC from $\mathbf{s}_0 \sim p_0$*:*
  for step $t = 0...T$ **do**
    $\mathbf{a}_t \sim \Pi_\theta(\cdot | h_\theta(\mathbf{s}_t))$ ◁ *Sample with TD-MPC*
    $(\mathbf{s}_{t+1},\ r_t) \sim \mathcal{T}(\cdot | \mathbf{s}_t, \mathbf{a}_t),\ \mathcal{R}(\cdot | \mathbf{s}_t, \mathbf{a}_t)$ ◁ *Step env.*
    $\mathcal{B} \leftarrow \mathcal{B} \cup (\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ ◁ *Add to buffer*
  // Update TOLD using collected data in $\mathcal{B}$*:*
  for num updates per episode **do**
    $\{\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1}\}_{t:t+H} \sim \mathcal{B}$ ◁ *Sample traj.*
    $\mathbf{z}_t = h_\theta(\mathbf{s}_t)$ ◁ *Encode first observation*
    $J = 0$ ◁ *Initialize J for loss accumulation*
    for $i = t...t+H$ **do**
      $\hat{r}_i = R_\theta(\mathbf{z}_i, \mathbf{a}_i)$ ◁ *Equation 8*
      $\hat{q}_i = Q_\theta(\mathbf{z}_i, \mathbf{a}_i)$ ◁ *Equation 9*
      $\mathbf{z}_{i+1} = d_\theta(\mathbf{z}_i, \mathbf{a}_i)$ ◁ *Equation 10*
      $\hat{a}_i = \pi_\theta(\mathbf{z}_i)$ ◁ *Equation 11*
      $J \leftarrow J + \lambda^{i-t} \mathcal{L}(\mathbf{z}_{i+1}, \hat{r}_i, \hat{q}_i, \hat{\mathbf{a}}_i)$ ◁ *Equation 7*
    $\theta \leftarrow \theta - \frac{1}{H} \eta \nabla_\theta J$ ◁ *Update online network*
    $\theta^- \leftarrow (1-\tau)\theta^- + \tau\theta$ ◁ *Update target network*

Experiments

- ◈ Baselines
  - – SAC
  - – LOOP
    - A hybrid algorithm that extends SAC with planning and a learned model
  - – MPC with a ground truth simulator
  - – **CURL**
    - Contrastive Unsupervised Representations for Reinforcement Learning
  - – **DrQ**
    - Data-regularized Q
    - Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels
  - – PlaNet
  - – Dreamer
  - – MuZero
  - – EfficientZero
  - – Ablations
    - Using state predictor
    - Without the latent consistency loss

Experiments

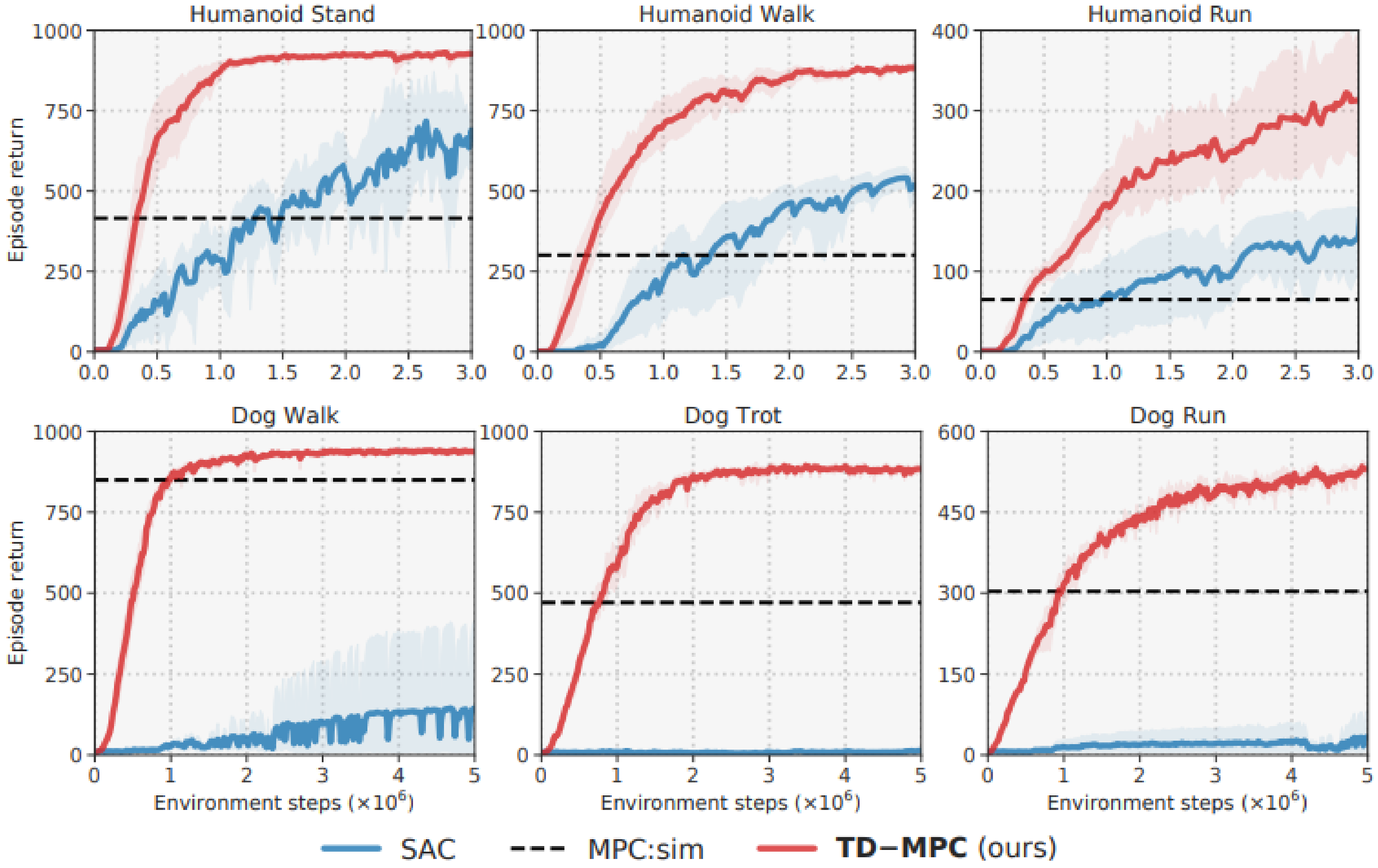
◈ Baselines

Method	Model objective	Value	Inference	Continuous	Compute
SAC	✗	✓	Policy	✓	Low
QT-Opt	✗	✓	CEM	✓	Low
MPC:sim	Ground-truth model	✗	CEM	✓	High
POLO	Ground-truth model	✓	CEM	✓	High
LOOP	State prediction	✓	Policy w/ CEM	✓	Moderate
PlaNet	Image prediction	✗	CEM	✓	High
Dreamer	Image prediction	✓	Policy	✓	Moderate
MuZero	Reward/value pred.	✓	MCTS w/ policy	✗	Moderate
EfficientZero	Reward/value pred. + contrast.	✓	MCTS w/ policy	✗	Moderate
TD-MPC (ours)	Reward/value pred. + latent pred.	✓	CEM w/ policy	✓	Low

MuZero's loss function: $l_t(\theta) = \sum_{k=0}^{K} l^r(u_{t+k}, r_t^k) + l^v(z_{t+k}, v_t^k) + l^p(\pi_{t+k}, \mathbf{p}_t^k) + c||\theta||^2$

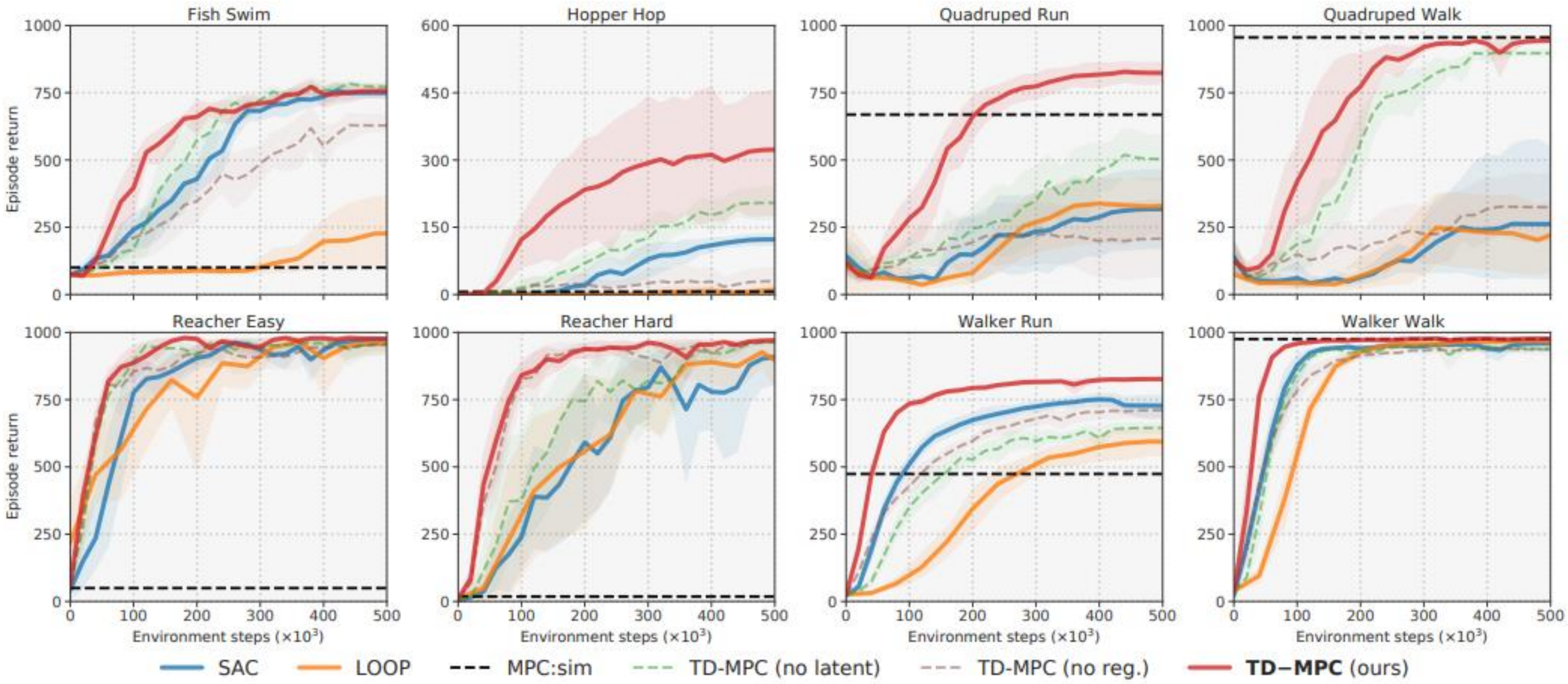
Experiments

- Environments
  - 6 Humanoid and Dog locomotion tasks with high-dimensional state and action spaces



Experiments

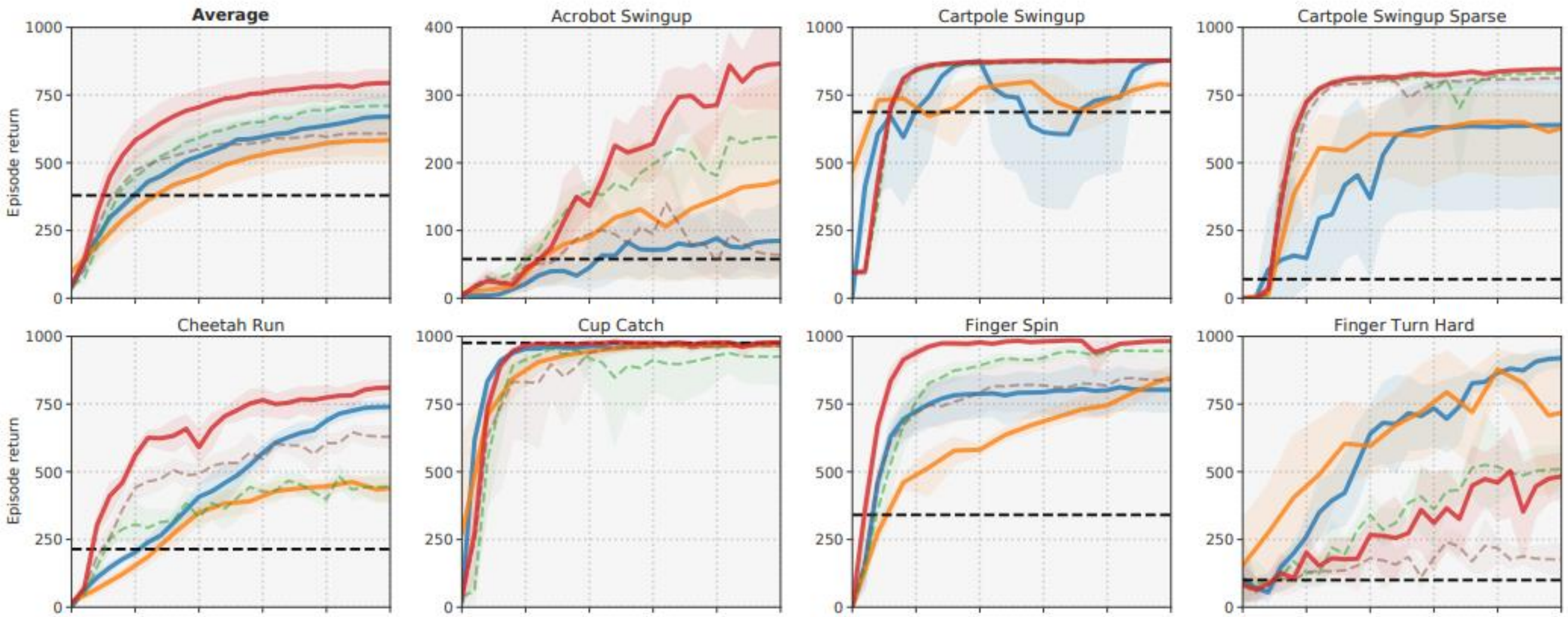
- Environments
  - 15 diverse continuous control tasks from DM-Control, 6 of which have sparse rewards



Experiments

◈ Environments

– 15 diverse continuous control tasks from DMControl, 6 of which have sparse rewards



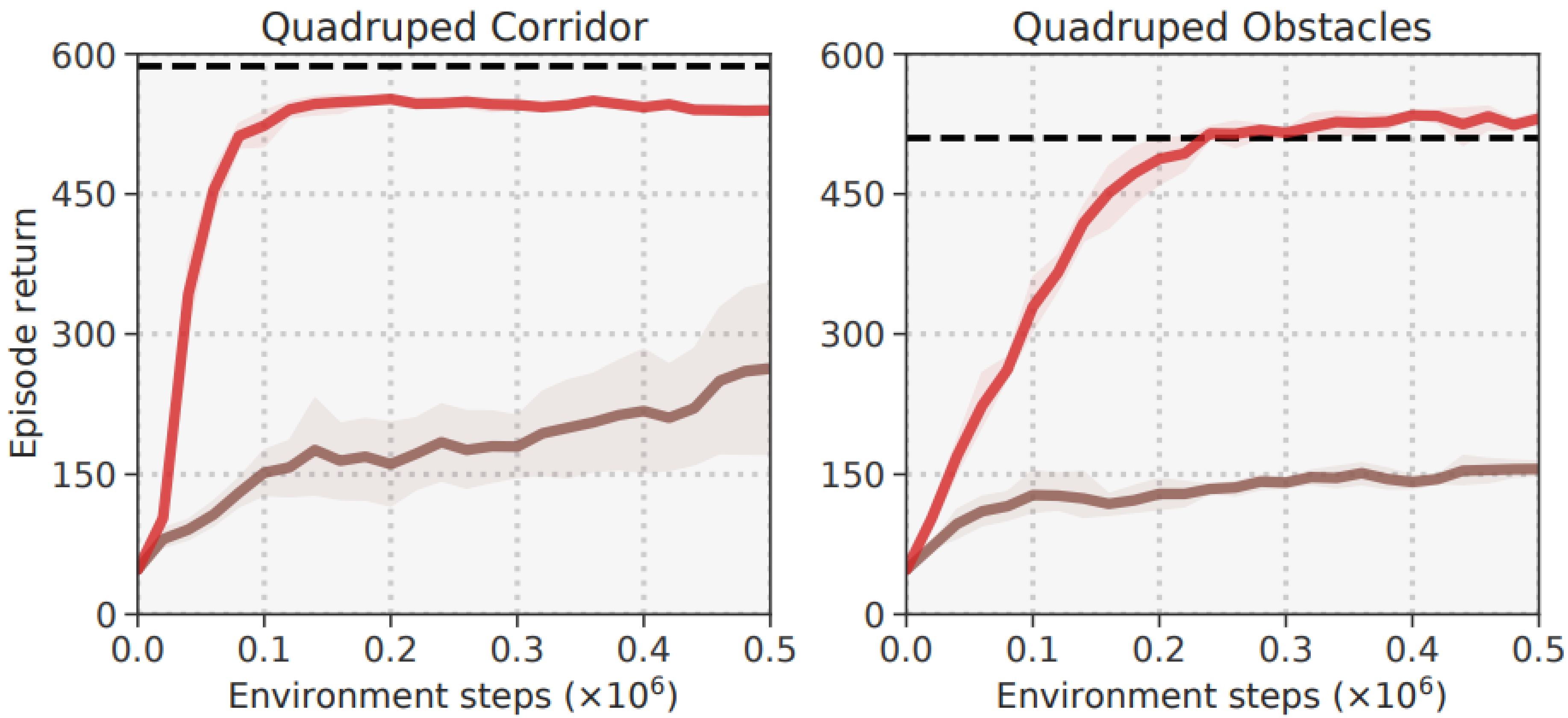
Experiments

- Environments
  - 6 image-based tasks from the data-efficient DMControl 100k benchmark.

	Model-free				Model-based				Ours
100k env. steps	SAC State	SAC Pixels	CURL	DrQ	PlaNet	Dreamer	MuZero*	Eff.Zero*	**TD-MPC**
Cartpole Swingup	812±45	419±40	597±170	**759±92**	563±73	326±27	219±122	**813±19**	**770±70**
Reacher Easy	919±123	145±30	517±113	601±213	82±174	314±155	493±145	**952±34**	628±105
Cup Catch	957±26	312±63	772±241	**913±53**	710±217	246±174	542±270	**942±17**	**933±24**
Finger Spin	672±76	166±128	779±108	**901±104**	560±77	341±70	–	–	**943±59**
Walker Walk	604±317	42±12	344±132	**612±164**	221±43	277±12	–	–	**577±208**
Cheetah Run	228±95	103±38	**307±48**	**344±67**	165±123	235±137	–	–	222±88

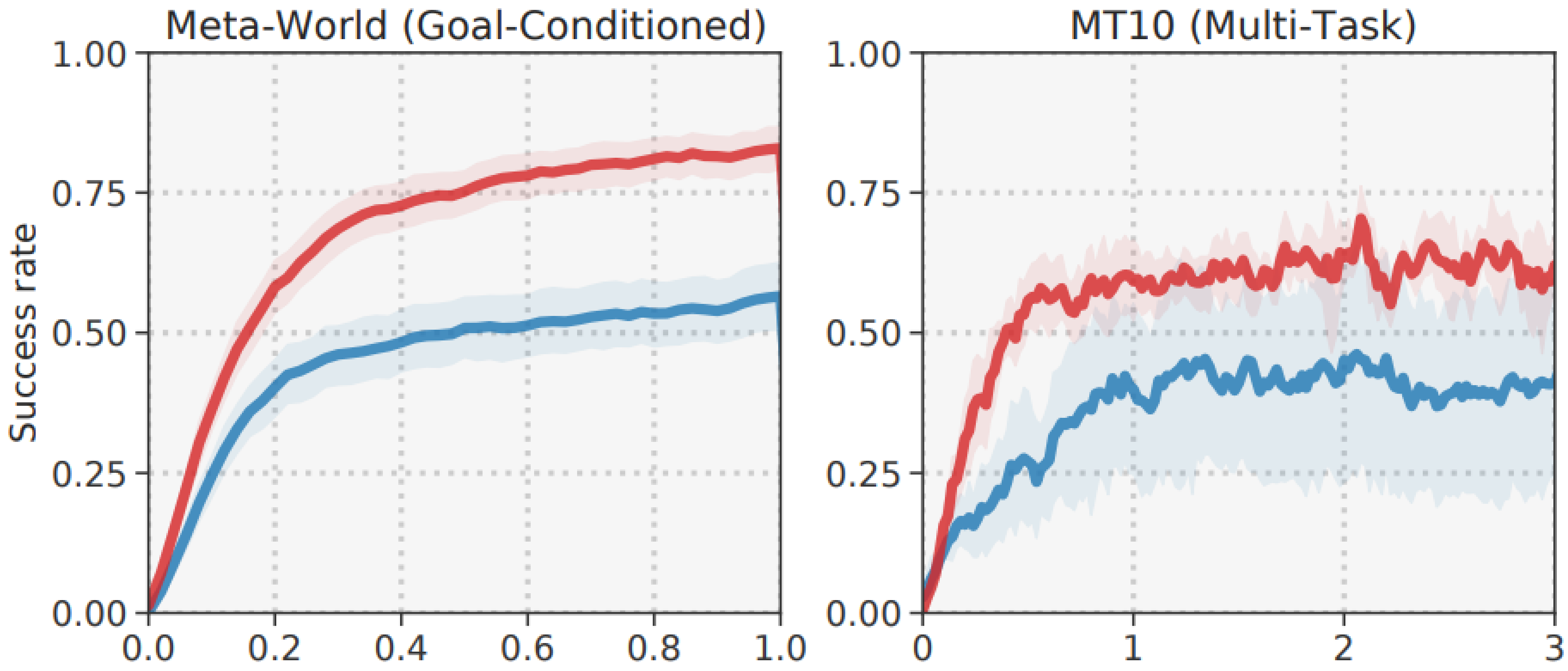
Experiments

- Environments
  - 2 multi-modal (proprioceptive data + egocentric camera) 3D locomotion tasks in which a quadruped agent navigates around obstacles.



Experiments

- Environments
  - 50 goal-conditioned manipulation tasks from MetaWorld, as well as a multi-task setting where 10 tasks are learned simultaneously.



감사합니다.