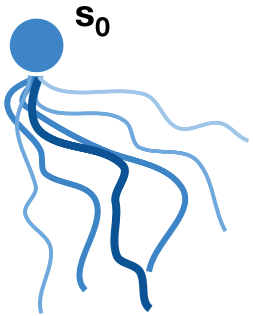
Temporal Difference Learning for Model Predictive Control

Nicklas Hansen, Xiaolong Wang*, Hao Su*

UC San Diego

Data-Driven Model Predictive Control

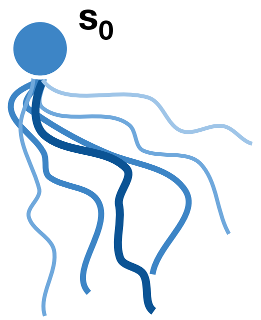
- Plan using a ***learned*** model of the environment
- Objective $\mathbb{E}_{\Gamma \sim \Pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\right]$ intractable



(repeat for ∞ steps)

Data-Driven Model Predictive Control

- Plan using a ***learned*** model of the environment
- Objective $\mathbb{E}_{\Gamma \sim \Pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\right]$ intractable
- Instead find ***locally optimal*** trajectory $\mathbb{E}_{\Gamma \sim \Pi_\theta}\left[\sum_{t=0}^{H} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\right]$
- **Two major challenges:**
  - Compounding model errors
  - Cost of long-horizon planning



(repeat for H steps)

How can TD-learning help MPC?

- Learning a ***terminal value function*** by TD-learning
    - MPC yields temporally ***local*** optimal solutions
    - A value function approximates the ***globally*** optimal solution

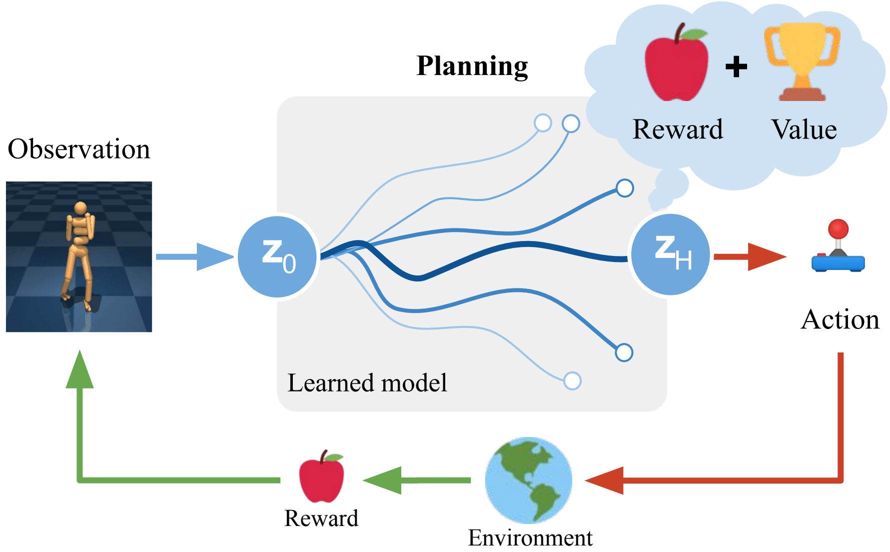
- Learning a ***task-oriented*** representation
    - Model-based RL typically ***models everything*** in the environment
    - Model-free RL only retains information ***predictive of reward***

TD-MPC

Inference *(planning)*

- Planning in latent space
- Return estimate:

$$\mathbb{E}_{\Gamma}\left[\underbrace{\gamma^{H} Q_{\theta}(\mathbf{z}_{H}, \mathbf{a}_{H})}_{\textbf{Value}} + \underbrace{\sum_{t=0}^{H-1} \gamma^{t} R_{\theta}(\mathbf{z}_{t}, \mathbf{a}_{t})}_{\textbf{Rewards}}\right]$$

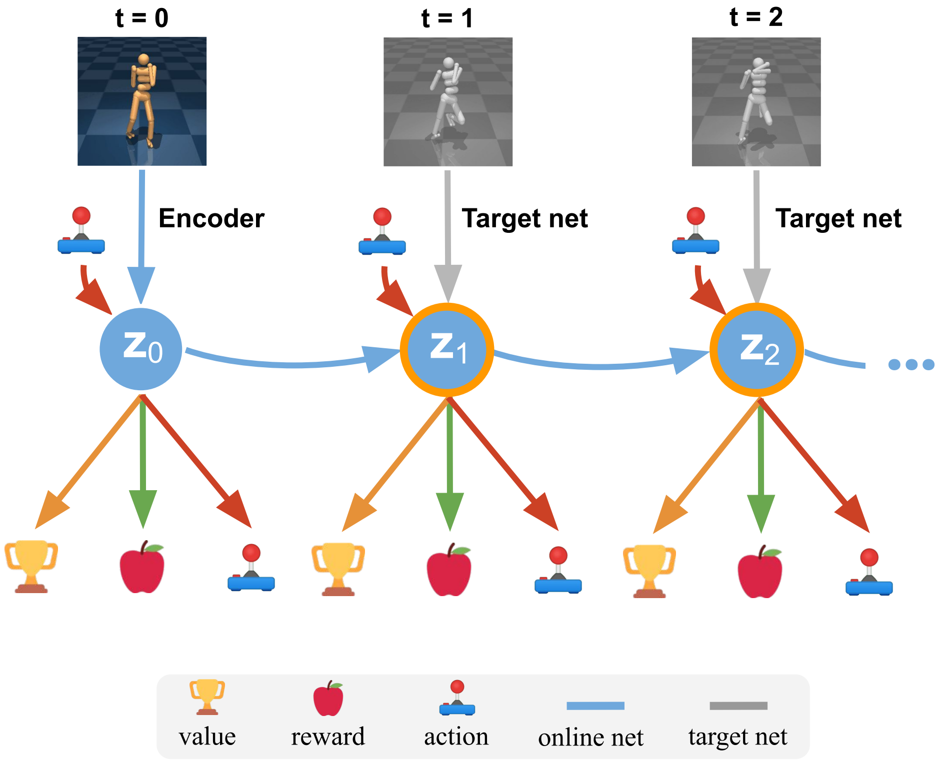


TD-MPC

Task-Oriented Latent Dynamics (**TOLD**) model

- Model only parts of environment that are ***predictive of reward***
- Learned ***jointly*** with value function ***using TD-learning***

TD-MPC


t = 0
t = 1
t = 2
Encoder
Target net
Target net
z_0
z_1
z_2
value
reward
action
online net
target net

TD-MPC

TOLD minimizes the objective

$$\mathcal{J}(\theta;\Gamma) = \sum_{i=t}^{t+H} \lambda^{i-t} \mathcal{L}(\theta;\Gamma_i)\,, \tag{7}$$

where

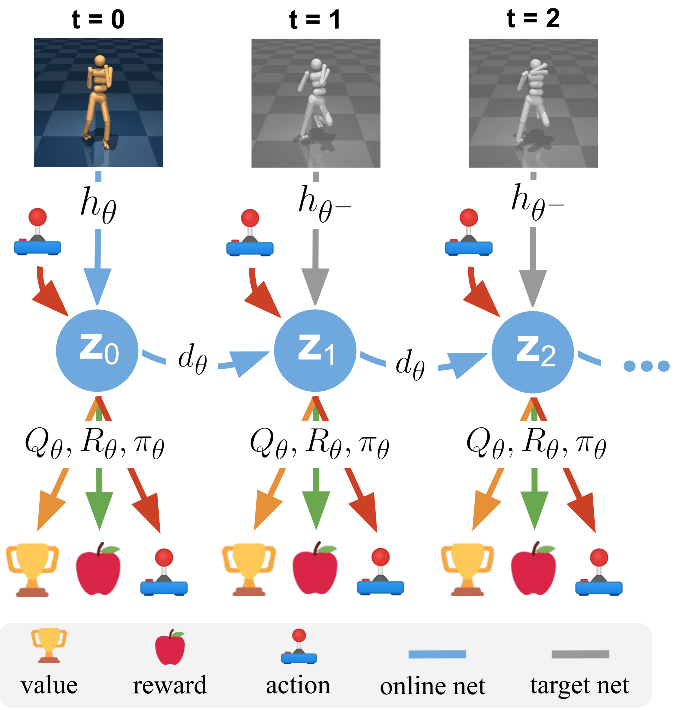
$$\mathcal{L}(\theta;\Gamma_i) = c_1 \underbrace{\|R_\theta(\mathbf{z}_i,\mathbf{a}_i) - r_i\|_2^2}_{\text{reward}} \tag{8}$$

$$+\, c_2 \underbrace{\|Q_\theta(\mathbf{z}_i,\mathbf{a}_i) - (r_i + \gamma Q_{\theta^-}(\mathbf{z}_{i+1}, \pi_\theta(\mathbf{z}_{i+1})))\|_2^2}_{\text{value}} \tag{9}$$

$$+\, c_3 \underbrace{\|d_\theta(\mathbf{z}_i,\mathbf{a}_i) - h_{\theta^-}(\mathbf{s}_{i+1})\|_2^2}_{\text{latent state consistency}} \tag{10}$$

and the policy minimizes

$$\mathcal{J}_\pi(\theta;\Gamma) = -\sum_{i=t}^{t+H} \lambda^{i-t} Q_\theta(\mathbf{z}_i, \pi_\theta(\mathrm{sg}(\mathbf{z}_i)))\,, \tag{11}$$



TD-MPC

Why learn a policy?

- **Planning:** policy ***action proposals*** speed up convergence
- **Learning:** estimating Q-targets via planning is ***very slow***; use policy instead

$$\mathcal{L}(\theta;\Gamma_i) = c_1 \underbrace{\|R_\theta(\mathbf{z}_i,\mathbf{a}_i) - r_i\|_2^2}_{\text{reward}} \tag{8}$$

$$+ c_2 \underbrace{\|Q_\theta(\mathbf{z}_i,\mathbf{a}_i) - (r_i + \gamma Q_{\theta^-}(\mathbf{z}_{i+1}, \pi_\theta(\mathbf{z}_{i+1})))\|_2^2}_{\text{value}} \tag{9}$$

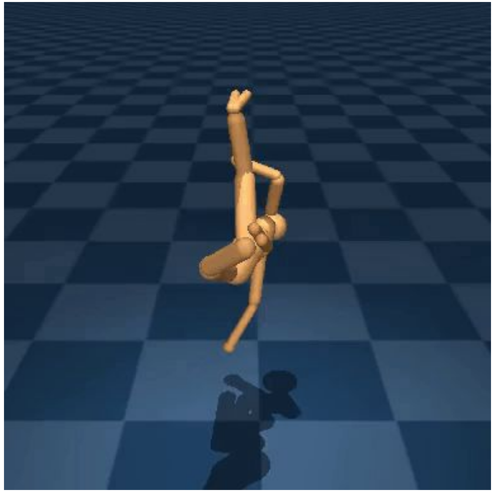
$$+ c_3 \underbrace{\|d_\theta(\mathbf{z}_i,\mathbf{a}_i) - h_{\theta^-}(\mathbf{s}_{i+1})\|_2^2}_{\text{latent state consistency}} \tag{10}$$

Results

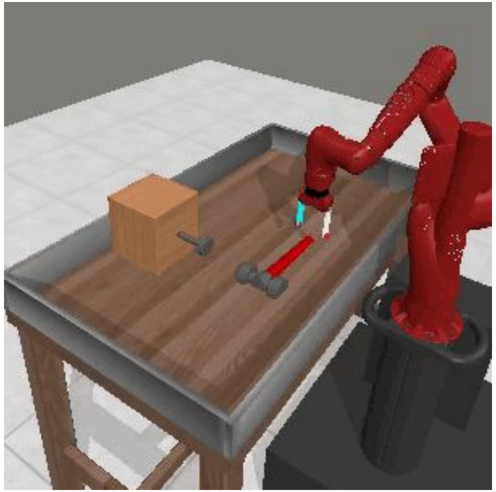
TD-MPC solves ***challenging*** continuous control problems

Dog Run

Humanoid Run



Hammer



Results

TD-MPC solves ***challenging*** continuous control problems

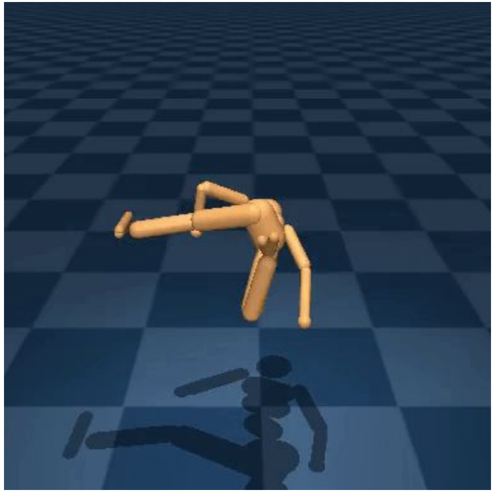
SAC

TD-MPC

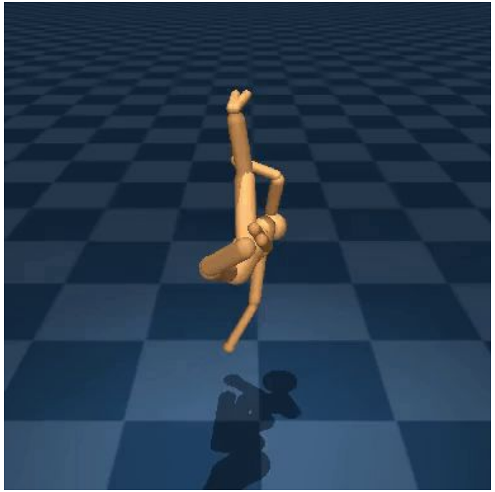
Results

TD-MPC solves ***challenging*** continuous control problems

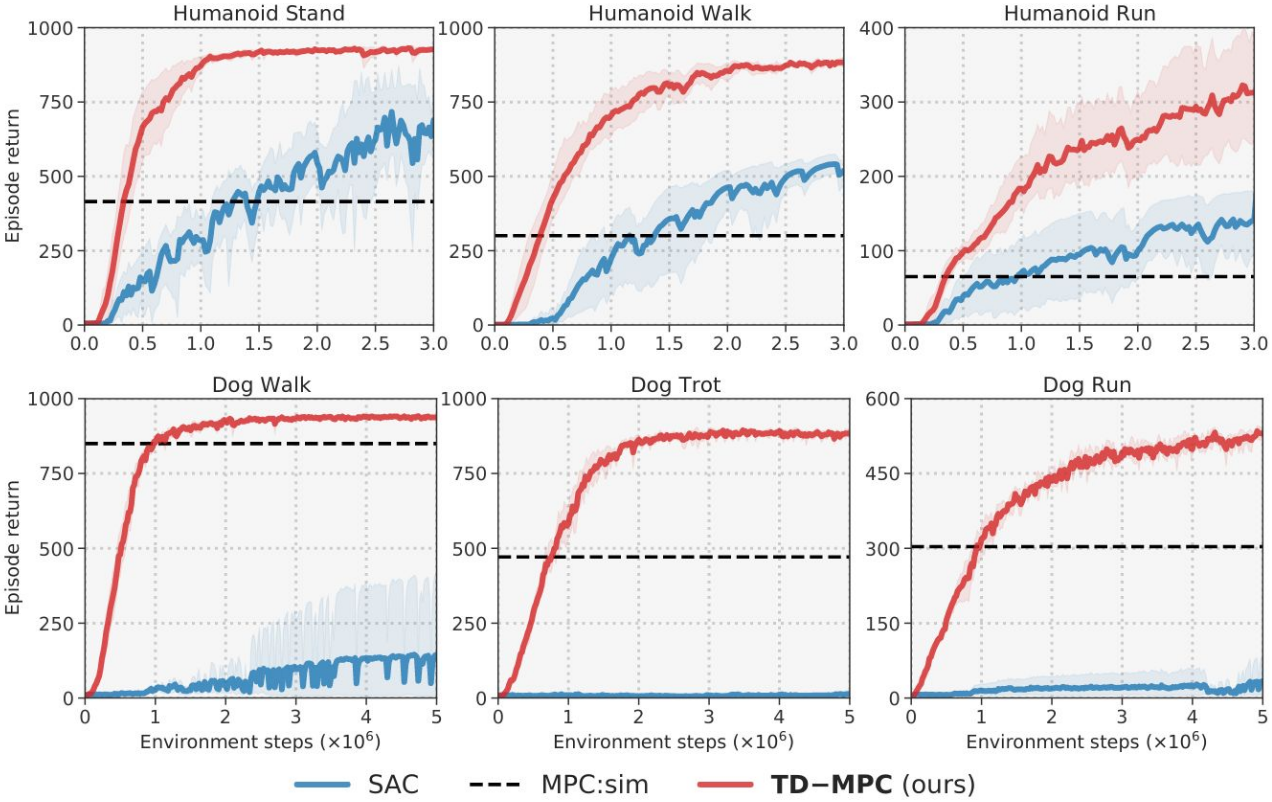
SAC



TD-MPC

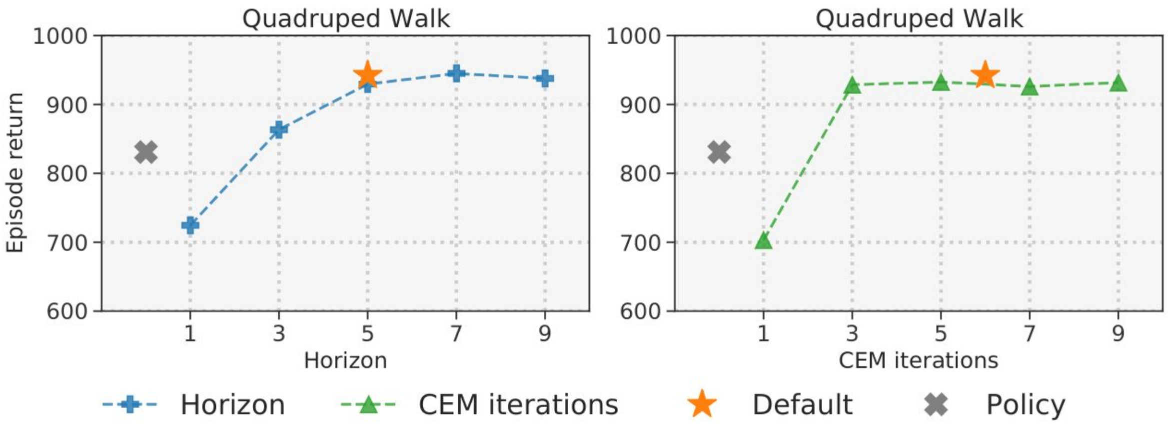


Results


Humanoid Stand
Humanoid Walk
Humanoid Run
Dog Walk
Dog Trot
Dog Run
Episode return
Environment steps (×10^6)
SAC
MPC:sim
TD−MPC (ours)

Results

More **planning** → better **performance**



Variable budget ***at test-time***

Results

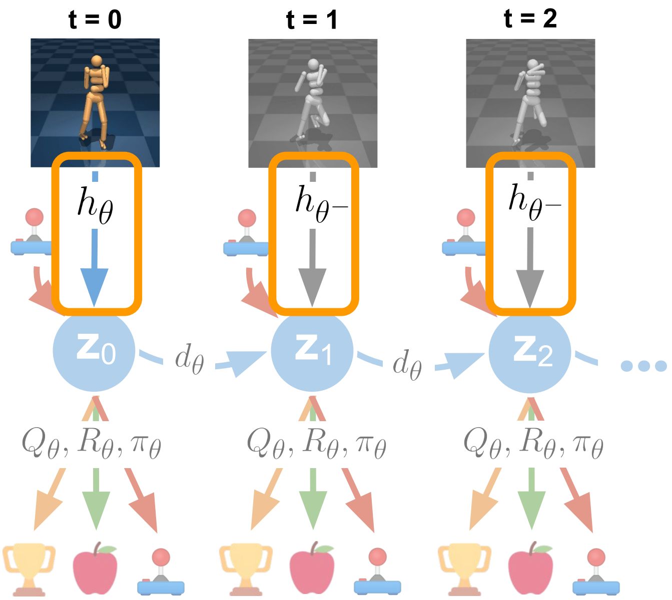
Replace MLP encoder with CNN → competitive performance on ***image-based RL***

	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

Results

TD-MPC is ***input-agnostic***; just change ***h***

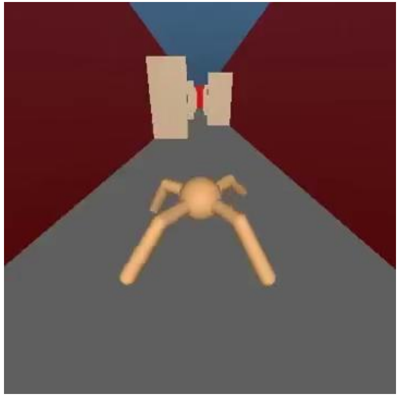
- Trivially extended to multi-modal RL



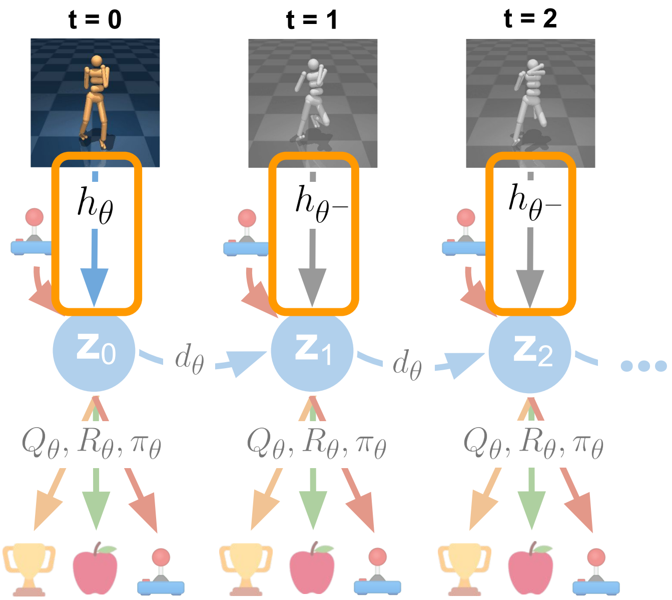
Results

TD-MPC is ***input-agnostic***; just change ***h***

- Trivially extended to multi-modal RL



Proprioceptive data + egocentric camera



Results

TD-MPC matches the ***time to solve*** of SAC but uses far less data

	Walker Walk				Humanoid Stand	
Wall-time (h)	SAC	LOOP	MPC:sim	**TD-MPC**	SAC	**TD-MPC**
time to solve ↓	0.41	7.72	0.91	0.47	9.31	9.39
h/500k steps ↓	1.41	18.5	–	5.60	1.82	12.94

nicklashansen.github.io/td-mpc