Simplified Temporal Consistency Reinforcement Learning

Yi Zhao [1] **Wenshuai Zhao** [1] **Rinu Boney** [2] **Juho Kannala** [2] **Joni Pajarinen** [1]

Abstract

Reinforcement learning (RL) is able to solve complex sequential decision-making tasks but is currently limited by sample efficiency and required computation. To improve sample efficiency, recent work focuses on model-based RL which interleaves model learning with planning. Recent methods further utilize policy learning, value estimation, and, self-supervised learning as auxiliary objectives. In this paper we show that, surprisingly, a simple representation learning approach relying only on a latent dynamics model trained by latent temporal consistency is sufficient for high-performance RL. This applies when using pure planning with a dynamics model conditioned on the representation, but, also when utilizing the representation as policy and value function features in model-free RL. In experiments, our approach learns an accurate dynamics model to solve challenging high-dimensional locomotion tasks with online planners while being 4.1× faster to train compared to ensemble-based methods. With model-free RL without planning, especially on high-dimensional tasks, such as the Deepmind Control Suite Humanoid and Dog tasks, our approach outperforms model-free methods by a large margin and matches model-based methods' sample efficiency while training 2.4× faster.

1. Introduction

Deep reinforcement learning (DRL) has shown promising results in games (Schrittwieser et al., 2020; Bellemare et al., 2013), animation (Peng et al., 2018) and robotics (Wu et al., 2022; Andrychowicz et al., 2020; Lee et al., 2020). However, DRL is data-demanding requiring millions of data points to train limiting the applicability of DRL in real-world scenarios. To make DRL more sample-efficient, motivated by the success of self-supervised learning in both vision and language tasks, a series of recent works (Ma et al., 2020; Schwarzer et al., 2021; Laskin et al., 2020) introduce self-supervised auxiliary losses. The aim is to improve representation learning for policy and value functions. The adopted self-supervised losses include image reconstruction (Yarats et al., 2021; Ha & Schmidhuber, 2018; Watter et al., 2015) and maximizing the similarity between two augmentations of the same image (contrastive training) (Laskin et al., 2020). However, most of these methods demonstrate the effectiveness of the proposed method on pixel-based tasks. And their conclusions can not always directly transfer to

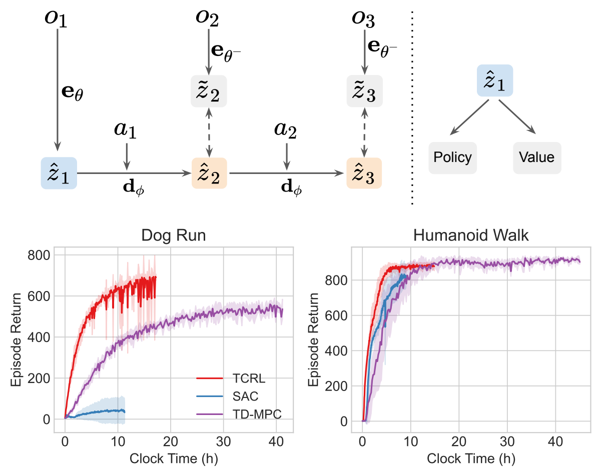

Figure 1. **Top**: Model architecture for computing our temporal consistency reinforcement learning (TCRL) latent state $\hat{z}_t$. At each time step t the model encodes observation o_t into a latent state $\hat{z}_t$ using a neural network. The latent dynamics model d_ϕ predicts the next latent state $\hat{z}_{t+1}$ using $\hat{z}_t$ and action a_t. A standard momentum encoder $\mathbf{e}_{\theta-}$ prevents collapse of the latent state representation. We use the cosine loss between the latent state and the momentum encoded latent state, denoted by a dashed line, for training. The reward function, omitted for clarity, is trained with the standard MSE loss. This simple model works surprisingly well providing a trained dynamics model for planning experiments, and, in model-free RL experiments, the trained latent states $\hat{z}_t$ are used as inputs to policy and value functions. **Bottom**: Episodic returns of TCRL compared to SAC and TD-MPC with respect to computing time on high-dimensional Humanoid and Dog tasks. Agents are trained with 5 million environment steps, and we plot 5 runs with shaded areas denoting 95% confidence intervals.

[1]Department of Electrical Engineering and Automation, Aalto University, Finland [2]Department of Computer Science, Aalto University, Finland. Correspondence to: Yi Zhao <yi.zhao@aalto.fi>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

state-based tasks, where a compact state representation is already available.

Recently, Zhang et al. (2021); McInroe et al. (2021); Schwarzer et al. (2021) exploit the temporal relationship of sequential observations and propose to facilitate representation learning by predicting future observations or future latent states. These methods usually train an image encoder and a latent dynamics model *jointly* with the value function. Hansen et al. (2022); Schrittwieser et al. (2020); Ye et al. (2021); Ghugare et al. (2023) extend these methods by leveraging the learned latent dynamics model to improve the policy and show promising results. For example, TD-MPC is the first documented method that solves high-dimensional dog tasks from DeepMind Control Suite (Tunyasuvunakool et al., 2020) (DMC) by leveraging elaborate decision-time planning using the learned dynamics. Since all these methods learn and use the encoder, latent dynamics model, and value functions jointly, it is difficult to distinguish which parts contribute to performance improvements.

This paper investigates the role of latent temporal consistency in state-based reinforcement learning. Specifically, we propose a simple but effective method, called TCRL (temporal consistency RL) to learn a state encoder and a latent dynamics model jointly, as shown in Figure 1. We carefully strip away complexities of previous methods and report empirical results on both i) dynamics model learning as well as ii) policy learning. We show that the trained model is accurate enough to be used for online planning to solve high-dimensional locomotion tasks. Compared to PETS (Chua et al., 2018), without using ensembles, probabilistic models, and state normalization, our method is able to solve high-dimensional DMC tasks, such as Quadruped Walk and Dog Walk ($\mathcal{A} \in \mathbb{R}^{38}, \mathcal{O} \in \mathbb{R}^{223}$) while being **4.1** $\times$ faster to train than ensemble-based methods which fail in these tasks. Also, even though in state-based tasks, inputs are already compact and have physical meaning, we still observe a big performance boost when training a *model-free* agent in the learned latent space instead of the observation space. Our method improves the performance on challenging locomotion tasks compared to strong model-free baselines and matches sample efficiency compared to state-of-the-art model-based methods while being faster to train. Furthermore, thanks to our minimalistic algorithm and implementation[1], we conduct extensive well-controlled experiments to analyze the usage of temporal consistency in state-based RL. We find that i) temporal consistency is more compatible with the cosine loss than the MSE loss; ii) joint training of the dynamics model and the value function introduces additional instability; iii) although our learned dynamics model is accurate to support online planning on high-dimensional tasks, we fail to observe strong benefits of using it in policy or value function learning.

[1]Source Code of TCRL: https://github.com/zhaoyi11/tcrl

2. Related Work

Learning dynamics models for planning Learning an accurate dynamics model from interactions with the environment is crucial for model-based reinforcement learning (MBRL). When low-dimensional observations are accessible, it's common to learn a dynamics model in the observation space. PILCO (Deisenroth & Rasmussen, 2011) and PETS (Chua et al., 2018) stress the importance of capturing the model's uncertainty and explicitly incorporating the uncertainty into planning. Many recent MBRL methods (Janner et al., 2019; Clavera et al., 2020; Yu et al., 2020; Kidambi et al., 2020) employ the ensemble model used in PETS by default. However, its computational complexity grows linearly with the number of ensembles. And Lutter et al. (2021) shows that these methods are struggling to handle high-dimensional tasks. Instead, we show that by leveraging the latent temporal consistency, we can learn a simple dynamics model to achieve better performance without using ensembles or probabilistic models.

Representation learning in RL Representation learning has been investigated for decades in the context of RL. Singh et al. (1994); Dearden & Boutilier (1997); Andre & Russell (2002); Li et al. (2006); Mannor et al. (2004); Ruan et al. (2015); Jiang et al. (2015); Abel et al. (2016) identify or learn a compact representation of state or action space but are usually limited to relatively simple environments. Gelada et al. (2019); Zhang et al. (2021) extend these ideas to solve challenging tasks leveraging neural networks. Many recent works facilitate RL research by utilizing good insights from self-supervised learning (SSL). SSL aims to learn meaningful features without labels (Kingma & Welling, 2014; Higgins et al., 2017; Oord et al., 2018; Chen et al., 2020; He et al., 2020; Grill et al., 2020). For example, Hafner et al. (2019; 2020; 2021; 2023) learn good representations (latent space) in a VAE-like (Kingma & Welling, 2014) way by reconstructing observations and then do planning and policy learning in the latent space. And Hansen et al. (2022); Schwarzer et al. (2021); McInroe et al. (2021) learn representations via contrastive learning (Hadsell et al., 2006). SSL is mainly used in visual control tasks (Yarats et al., 2021; Ha & Schmidhuber, 2018; Laskin et al., 2020; Huang et al., 2022). However, the effectiveness of SSL in state-based tasks has not been clearly verified (Yang & Nachum, 2021), since in this case, a compact representation with physical meaning is available. Also, *none* of these methods try to investigate the accuracy of the learned dynamics model. Our paper aims to fill this gap.

Learning value-oriented dynamics models Recent work learns dynamics models without predicting future observations. Silver et al. (2017); Oh et al. (2017); Tamar et al. (2016); Schrittwieser et al. (2020); Ye et al. (2021) learn a dynamics model that only predicts rewards and values over multiple time steps and uses the learned model for planning. The state-of-the-art method TD-MPC (Hansen et al., 2022), closest work to ours, uses a temporal consistency constraint in the latent space. TD-MPC shows good performance on continuous control tasks by training a latent dynamics model jointly with value functions and planning with the learned dynamics model. Through extensive experiments, we argue *none* of these are required to solve high-dimensional tasks such as the Humanoid and Dog tasks. Instead, we train the value function separately from the dynamics model and learn the policy and value function in a model-free way. Without using elaborate planning, our TCRL achieves competitive performance and is **2.4**$\times$ faster to train. Our results suggest that learning a high-quality state representation is a key factor in solving these challenging tasks.

3. Method

Components As shown in Figure 1, our TCRL model includes four components:

$$\begin{aligned} \text{Encoder}: &\ z_t = \mathbf{e}_\theta(o_t) \\ \text{Transition}: &\ z_{t+1}, r_t = \mathbf{d}_\phi(z_t, a_t) \\ \text{Value}: &\ q_t = \mathbf{q}_\psi(z_t, a_t) \\ \text{Policy}: &\ a_t \sim \pi_\eta(z_t) \end{aligned} \tag{1}$$

An encoder $\mathbf{e}_\theta$ maps an observation o_t into a latent state z_t. In our method, two encoders are used, named online encoder $\mathbf{e}_\theta$ and momentum encoder $\mathbf{e}_{\theta^-}$. We calculate the target latent state $\tilde{z}_t$ using the momentum encoder with parameters θ^- which are the exponential moving average (EMA) of the online encoder parameters θ. Based on the latent state z_t and action a_t, a latent state at the next time step z_{t+1} as well as an immediate reward r_t are predicted. When inputs are pixels (see Appendix A), the encoder $\mathbf{e}_\theta$ is parameterized by a convolutional neural network (CNN), while MLPs are used when inputs are states. Also, both the latent dynamics model $\mathbf{d}_\phi$ and the value function $\mathbf{q}_\psi$ as well as the policy π_η are represented by MLPs.

In our method, the prediction happens in the *latent* space instead of observation space. In section 4.1, we show that this is a key factor enabling us to learn an accurate dynamics model to support online planning. Furthermore, the value function and policy take the learned latent states instead of observations. In section 4.2, we suggest this is crucial for solving challenging high-dimensional Humanoid and Dog tasks. Although common in pixel-based tasks, our method explicitly demonstrates that in state-based tasks where a compact representation is already available, we can still benefit from using the latent states learned by temporal consistency.

Learning the encoder and latent dynamics The latent dynamics model is trained by accurately predicting H-step rewards $\tilde{r}_{t:t+H}$ and target latent states $\tilde{z}_{t+1:t+1+H}$. Specifically, for a H-step trajectory $(o_t, a_t, r_t, o_{t+1})_{t:t+H}$ drawn from the replay buffer $\mathcal{D}$, starting from an initial observation o_t, we first use the online encoder $\mathbf{e}_\theta$ to obtain a latent representation $\hat{z}_t = \mathbf{e}_\theta(o_t)$. Then, conditioning on action sequences $a_{t:t+H}$, the transition function $\hat{z}_{t+1}, \hat{r}_t = \mathbf{d}_\phi(\hat{z}_t, a_t)$ is applied iteratively to predict future rewards $\hat{r}_{t:t+H}$ and latent states $\hat{z}_{t:t+H}$. The target rewards $\tilde{r}_{t:t+H}$ are just immediate rewards from the sampled trajectory, while the target latent states $\tilde{z}_{t:t+H}$ are calculated by the momentum encoder as $\tilde{z}_{t:t+H} = \mathbf{e}_{\theta^-}(o_{t:t+H})$. Given multi-step predictions and targets, the training objective is simply to minimize the discounted sum of the MSE loss of rewards and the negative cosine distance of latent states:

$$\sum_{h=0}^{H} \gamma^h \Big[||\hat{r}_{t+h} - \tilde{r}_{t+h}||_2^2 - \big(\frac{\hat{z}_{t+h}}{||\hat{z}_{t+h}||_2}\big)^\top \big(\frac{\tilde{z}_{t+h}}{||\tilde{z}_{t+h}||_2}\big) \Big]. \tag{2}$$

This temporal consistency is used in several previous papers (Schwarzer et al., 2021; Hansen et al., 2022; McInroe et al., 2021). Perhaps Hansen et al. (2022) is the most related paper proposing the TD-MPC method. In addition to TD-MPC using planning at decision-time, we differ from TD-MPC in two aspects: i) we use the cosine loss but TD-MPC uses the MSE loss; ii) TD-MPC learns a latent dynamics model jointly with value functions. Learning value functions jointly with the dynamics model, so-called *value-oriented* dynamics, enforces the dynamics model to encode strong task-specific information, which potentially makes it hard to generalize to new tasks (Zhang et al., 2018). Moreover, Section 5 highlights additional training instability issues that joint training may cause.

Learning the policy and value function For learning the policy and value function, we adopt a deep deterministic policy gradient method (DDPG) (Lillicrap et al., 2016) augmented with n-step returns (Watkins, 1989; Peng & Williams, 1994) following Yarats et al. (2022) with the only difference that instead of using the original observation o_t we use the latent state $z_t = \mathbf{e}_\theta(o_t)$ as input to the policy and value function. Specifically, the value function is updated by minimizing:

$$\begin{aligned} \mathcal{L}_\psi &= \mathbb{E}_{\tau\sim\mathcal{D}}\big[(\mathbf{q}_{\psi_k}(z_t, a_t) - y)^2\big], \forall k \in 1, 2 \\ y &= \sum_{h=0}^{n-1} \gamma^h r_{t+h} + \gamma^n \min_{k=1,2} \mathbf{q}_{\psi_k^-}(z_{t+n}, a_{t+n}) \end{aligned} \tag{3}$$

where $a_{t_n} = \pi_\eta(z_{t+n}) + \epsilon$ and the noise ϵ is sampled from a clipped Gaussian distribution $\epsilon \sim \mathcal{N}(0, \sigma^2)$ with a linearly decayed σ as in DrQv2(Yarats et al., 2022). Following previous model-free methods (Fujimoto et al., 2018; Haarnoja et al., 2018; Hasselt, 2010), we use double Q networks $\mathbf{q}_{\psi_{1,2}}$ as well as two delayed Q networks $\mathbf{q}_{\psi^-_{1,2}}$. We update the actor's parameters using the loss

$$\mathcal{L}_\eta = -\mathbb{E}_{\tau\sim\mathcal{D}}\big[\min_{k=1,2}\mathbf{q}_{\psi_k}(z_t, a_t)\big] \tag{4}$$

that maximizes Q-value for the actor. Again, $z_t = \mathbf{e}_\theta(o_t)$, action $a_t = \pi_\eta(z_t) + \epsilon$ and we do not update the encoder's parameters with actor's gradients.

4. Experiments

We evaluate our TCRL approach in several continuous DMC control tasks. DMC uses scaled rewards, where the maximal episodic returns are 1000 for all tasks. We evaluate our method in *two* separate settings:

- **Dynamics learning.** In the first setting, there is *no* value function or policy involved. Instead, we directly use the latent dynamics model for online planning calling the approach *TCRL-dynamics*. The aim is to answer whether temporal consistency can be used to learn an accurate latent dynamics model.
- **Policy learning.** In the second setting, we train the policy and value functions in a model-free way using latent states in place of original observations calling the approach *TCRL*. This experiment aims to investigate whether the latent representation trained by temporal consistency benefits policy and value function learning.

4.1. Evaluation of the Latent Dynamics Model

Evaluation metrics Learning an accurate dynamics model is critical for model-based RL research. Finding a proper way to evaluate the learned dynamics model is still an open problem. Unlike usual supervised learning, the mean square error on the test set does not directly reflect the model's performance in planning (Lutter et al., 2021; Lambert et al., 2020).

Considering the primary usage of a dynamics model is planning, we directly evaluate the learned model with its planning results using Model Predictive Path Integral (MPPI) (Camacho & Alba, 2013; Williams et al., 2015). MPPI is a population-based *online* planning method that iteratively improves the policy $a_{t:t+H}$ with samples. In each iteration j, N trajectories are sampled according to the current policy $a^j_{t:t+H}$. Then, K trajectories with higher returns $\sum_{h=0}^{H} r^j_{t+h}(s^j_{t+h}, a^j_{t+h})$ are selected. Next, we calculate an improved policy $a^{j+1}_{t:t+H}$ by taking the weighted average over the selected top-K trajectories, where the weights are calculated by taking the softmax over returns of the top-K trajectories. After J iterations, the first action of $a^J_{t:t+H}$ is executed.

In our MPPI implementation, *no* value function or policy network is involved. Instead, we iteratively learn the dynamics model by i) collecting experiences via the MPPI planner, and ii) improving the dynamics model using collected data by optimizing Equation 2. In this way, all sampled action sequences are evaluated by the learned latent dynamics model alone, thus the planning performance can reflect the model's accuracy. We consider the following comparison methods to learn the dynamics model:

- **Stochastic ensemble model** (PETS) learns an ensemble of stochastic neural networks predicting both mean and standard deviation of next states and rewards. Similarly to Chua et al. (2018), we only predict the one-step future since uncertainty propagation through multiple time steps is unclear (Lutter et al., 2021). PETS is commonly used in many model-based RL methods when inputs are in compact states.
- **Deterministic ensemble model** (EnsDet) uses an ensemble of deterministic neural networks to predict *multi-step* future observations and rewards. EnsDet's architecture is similar to our method with the difference of predicting the next observations instead of the next latent states enabling an experimental comparison between observation predictions and latent space predictions.

Although our primary focus is on state-based tasks, we also test our method on pixel-based tasks to show that it is general. In the experimental comparison with PlaNet (Hafner et al., 2019) in Appendix A, our method obtains comparable or better performance on six commonly used pixel-based benchmark tasks. Detailed ablation studies on pixel-based tasks are in Appendix A.3.

Planning results In Figure 2, we compare our method with the stochastic ensemble model (PETS) and deterministic ensemble model (EnsDet) on eight state-based tasks from DMC, including six commonly used benchmarking tasks and two challenging high-dimensional tasks: Quadruped Walk and Dog Walk. Although simple and without using ensembles, our method either matches or outperforms comparison methods in the tested tasks. The high-dimensional action and observation spaces $\mathcal{A} \in \mathbb{R}^{38}, \mathcal{O} \in \mathbb{R}^{223}$ make solving the Dog Walk task, even for a strong model-free baseline (Haarnoja et al., 2018), difficult. In fact, to the best of our knowledge, TCRL-dynamics is the first documented method that can control the Pharaoh Dog walking forward using online planning with the learned dynamics model.

Chua et al. (2018) discuss the importance of predicting both aleatoric and epistemic uncertainty to achieve good planning performance. However, in both Chua et al. (2018) and our experiments, the algorithms are tested in deterministic envi-

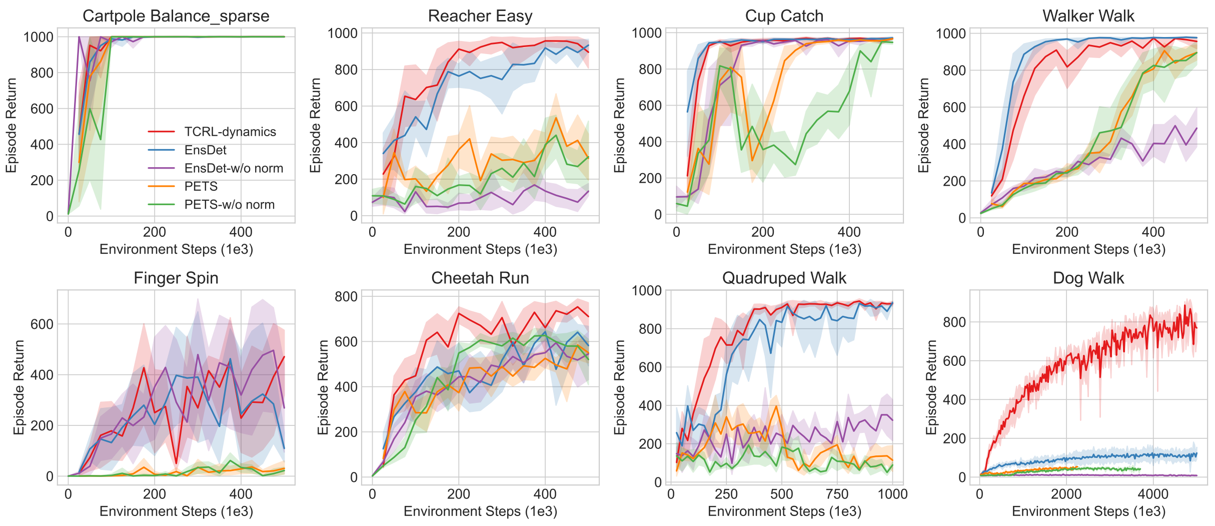

Figure 2. Planning performance on DMC continuous control benchmarks. We plot the mean across 5 runs. Each run includes 10 trajectories. Shaded areas denote 95% confidence intervals. Without using ensembles, our TCRL-dynamics consistently outperforms ensemble-based methods w.r.t. planning performance but with $\sim 4\times$ less training time. Especially on the challenging Dog Walk task, TCRL-dynamics outperforms previous methods by a large margin, which makes it a good candidate to learn the dynamics model in model-based RL.

ronments, which makes predicting the aleatoric uncertainty less motivated. In fact, in our experiments, TCRL-dynamics and EnsDet outperform PETS on all tasks. This evidence forces us to rethink the role of predicting aleatoric uncertainty in these commonly used deterministic tasks. Also, compared to PETS, both our method and EnsDet show the importance of using the multi-step training objective while it is not well compatible with the stochastic model used in PETS since properly propagating the uncertainty over multi-steps is challenging. Compared to EnsDet, our results show the superiority of predicting in the latent space instead of the observation space. We discuss this more in Section 5.

Moreover, we find that both PETS and EnsDet require state normalization. However, state normalization is not common in off-policy methods as this may introduce additional instability. Our assumption is that for off-policy methods, the data distribution in the replay buffer keeps changing during training which leads to the unstable mean and standard deviation for normalizing states. But TCRL-dynamics achieves good performance without state normalization, making it attractive to be adopted in model-based RL.

As mentioned before, we also evaluate our method on pixel-based tasks (Appendix A). On six visual control tasks, we show that our simple method matches PlaNet's results, which require an RNN and separated deterministic and stochastic paths to model the dynamics. Also, our method is easy to implement and **5.51** $\times$ faster to train. Our results show that this simple model can be competitive to be used in both pixel-based and state-based MBRL.

4.2. Evaluation of the Policy

We show that predicting in the latent space enables us to learn an accurate dynamics model to support online planning. Now, we try to introduce a policy and value to solve challenging tasks. We evaluate our TCRL methods on 24 continuous control tasks from DMC. Please check Appendix E for full results. We select 8 representative tasks to compare with other both model-based and model-free methods w.r.t sample efficiency and compute efficiency. We consider the following methods in our experiments:

- **TD-MPC** achieves promising performance on challenging tasks with *value-oriented* latent dynamics model, where the latent dynamics model is learned jointly with the value function, as well as elaborate decision-time planning. We include TD-MPC to demonstrate that *none* of these techniques are crucial to achieving good performance on challenging tasks, but state representation matters.
- **SAC** is a strong model-free baseline that achieves good performance among different DMC tasks. We include SAC to show how much the policy learning benefits from the state representation even when compact state representation is available.
- **REDQ** (Chen et al., 2021) is a strong model-free baseline that is built upon SAC but with a higher Update-To-Data ratio. We include it because this model-free method outperforms many strong model-based methods, such as PETS (Chua et al., 2018), MBPO (Janner

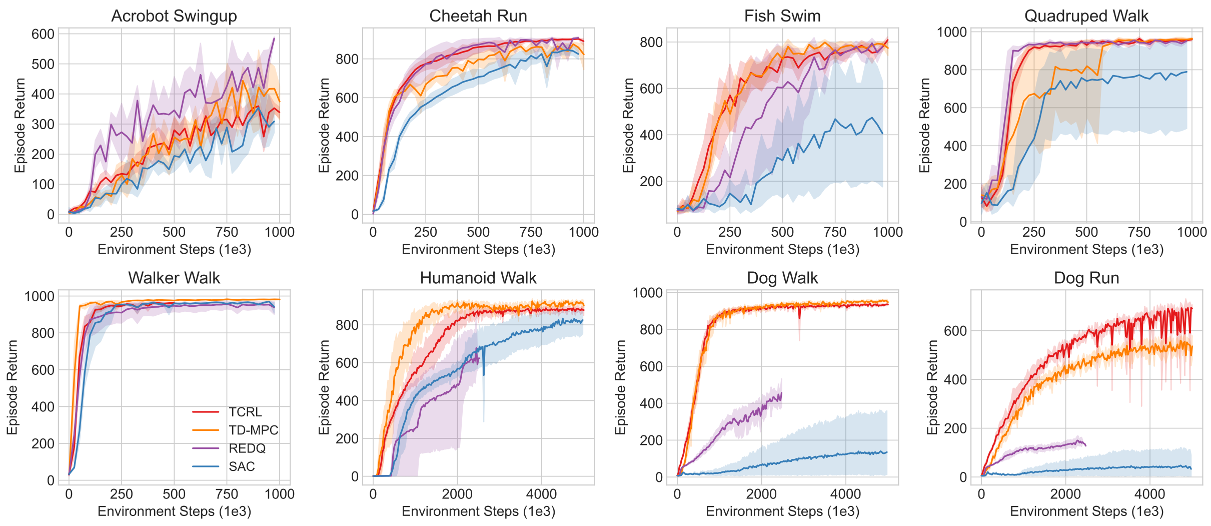

Figure 3. Policy performance on eight selected DMC tasks. We plot the mean across 5 runs, and each run includes 10 trajectories. The shaded areas denote 95% confidence intervals. Our TCRL outperforms strong model-free baselines SAC and REDQ by a large margin on challenging Humanoid and Dog locomotion control tasks. We also achieve comparable performance compared to the state-of-the-art model-based method TD-MPC without using elaborate planning. REDQ and TD-MPC take 18.3× and 2.4× training time respectively compared to TCRL. Full results on 24 continuous control tasks are shown in Appendix E.

et al., 2019), and MVE (Feinberg et al., 2018), with respect to sample efficiency.

We also compared ALM (Ghugare et al., 2023), a recent model-based approach learning representation, latent-space model, and policy jointly. But we fail to achieve good performance in DMC benchmark environments possibly due to hyperparameters in the official ALM implementation being tuned for OpenAI Gym (Brockman et al., 2016). We include ALM results in Appendix C.2.

Policy results Figure 3 compares the performance of TCRL with strong baselines on eight continuous control tasks, including challenging Humanoid and Dog domains. TCRL outperforms SAC on all tested tasks. Especially on complex tasks, such as Fish Swim, Humanoid Walk, Dog Walk, and Dog Run, our method outperforms SAC by a large margin. Since we also learn policy and value functions in a model-free way similar to SAC, the major performance improvement is from the state representation. Our results show strong evidence that state representation is critical even when a compact state representation is available, and temporal consistency can extract useful features that benefit policy and value function learning.

REDQ uses randomized ensembled double Q-learning to achieve a high update-to-data ratio. It achieves better performance than SAC on most tasks except the Humanoid Walk task. TCRL outperforms or matches REDQ on all tasks except the Acrobot Swingup task without increasing the update-to-data ratio and thus being 18.3× faster to train. Notice that TCRL and REDQ improve sample efficiency via orthogonal ways and can be easily combined.

To recapitulate, compared to TD-MPC, TCRL i) uses a cosine loss on latent states; ii) achieves comparable performance without a value-oriented latent dynamics model; iii) does not perform decision-time planning. The results show that using temporal consistency to extract useful state representations is a key factor to success in solving Humanoid and Dog tasks. Because we do not use decision-time planning, which is computationally heavy, our method is 2.4× faster to train. Furthermore, unlike TD-MPC, our method does not need to select pre-task action repeat (control frequency), making it easier to use. Moreover, Section 5 analyzes why TCRL achieves better performance on the Dog Run task.

5. Empirical Analysis

This section empirically studies TCRL with special attention to answering: i) why does TCRL work well? ii) which training objective should be used? iii) can we use the learned dynamics in policy learning? More comparison studies regarding hyperparameters and the choice of different training objectives are shown in Appendix B.

Why does TCRL work well? Experiments in dynamics model learning and policy learning show that one driving factor of TCRL's success is temporal consistency training extracting a useful state representation. Although the state

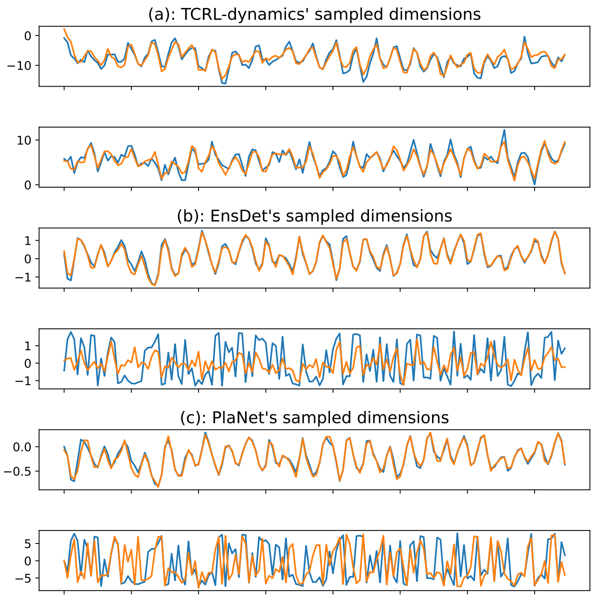

Figure 4. Comparison of open-loop predictions. Blue curves are ground truths and orange curves are predicted by learned dynamics models. **Top:** two sampled dimensions z_t from a Quadruped Walk trajectory in the latent space learned by TCRL-dynamics. **Middle:** two sampled dimensions o_t from the same trajectory but in the normalized observation space. The orange curves are predicted by EnsDet. **Bottom:** two sampled dimensions o_t from the same trajectory and the orange curves are reconstructed observations from the PlaNet model.

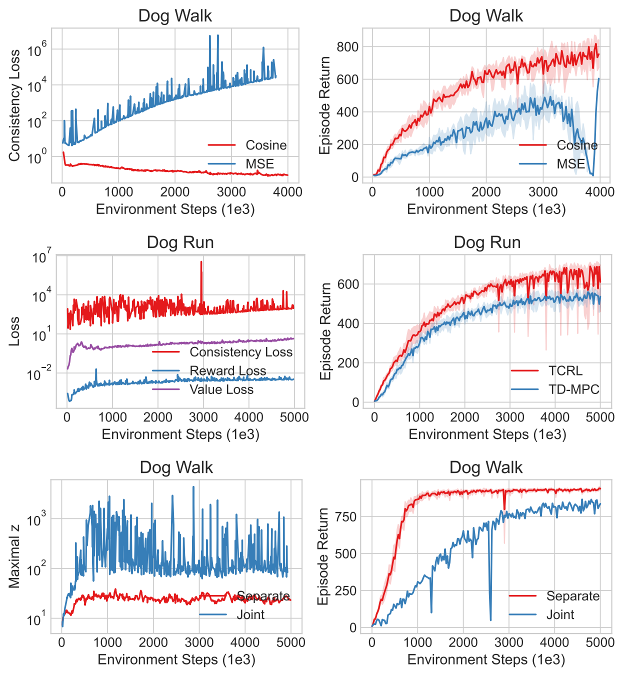

Figure 5. Comparison of choosing different training objectives. **Top**: we compare MSE with cosine loss in Equation 2 in our *TCRL-dynamics* variant. **Middle**: when jointly training the value function and the dynamics model with the MSE loss, as in TD-MPC, the consistency loss dominates the loss and leads to worse performance. **Bottom**: when jointly training the value function and the dynamics model with the cosine loss, the latent space keeps expanding, leading to performance degeneration.

from the simulator is already a compact representation and has physical meanings, the state still suffers from several possible drawbacks: i) not translation invariant; ii) not rotation invariant and the double-cover issue of quaternions; iii) magnitude differences between different dimensions; iv) strong temporal discontinuity.

The first two drawbacks are partially avoided by handcrafted features, e.g., DMC removes the Cheetah's position from the observation and represents angles by their sine and cosine in the Cartpole. For the third one, perhaps the most straightforward solution is to normalize the states. As shown in Figure 2, normalizing states introduces an obvious planning performance boost. However, as mentioned before, normalizing the states may introduce additional instability, and most off-policy methods (Haarnoja et al., 2018; Fujimoto et al., 2018; Lillicrap et al., 2016) do not utilize normalization. The fourth drawback is caused by the nature of locomotion tasks, that is when a robot contacts the ground, the acceleration and sensor readings change significantly.

In Figure 4, we compare open-loop predictions of the TCRL-dynamics, EnsDet, and PlaNet. Given an initial observation and an action sequence, we predict the future states up to 150 steps with different methods. The blue curves are ground truths while the orange curves are predictions. All plots are from the same Quadruped Walk trajectory but with different sampled dimensions. Figure 4(a) shows the open-loop predictions in the latent space learned by TCRL-dynamics. Figure 4(b) plots the predictions in the normalized observation space by EnsDet. Figure 4(c) shows reconstructed observations from PlaNet. Qualitatively our learned latent space is smoother and future latent states are easier to predict. The magnitude differences between dimensions and temporal discontinuity make it hard to train an accurate dynamics model by EnsDet and PlaNet.

Also, our method can ease optimization, according to Tian et al. (2021), the exponential moving average of the momentum encoder can be seen as an automatic curriculum. In the beginning, the training target $\tilde{z}_t$ is close to the prediction setting an easier target for training. Then, the target gradually becomes hard and then tends to stabilize as it converges.

Training objective As mentioned before, different from TD-MPC, we use cosine loss to measure the distance be-

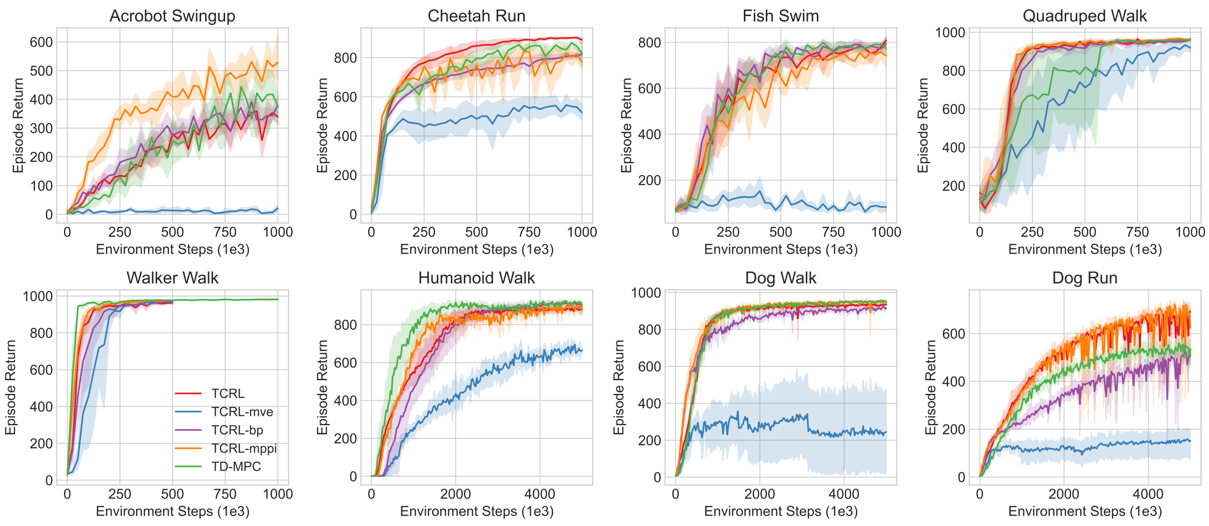

Figure 6. Comparison of different ways to use the learned dynamics in policy or value function learning.

tween the predicted and the target latent states and train a latent dynamics model separately with value functions. We now analyze our design choices.

For the choice of the loss function, previous self-supervised learning methods commonly use the cosine loss (He et al., 2020; Grill et al., 2020) and demonstrate performance drops with the MSE loss. The same evidence appears in RL setting (Schwarzer et al., 2021). We found that in the *TCRL-dynamics* variant, where *no* policy or value function is involved, using MSE loss works well in most tasks. However, we do observe a severe performance decrease on the Dog Walk task where observations are complex as in the top row of Figure 5. With the MSE loss, training tends to be unstable instead of collapse (Schwarzer et al., 2021). Whilst TCRL-dynamics, trained with cosine loss, is much more stable during training and achieves better asymptotic performance.

We now discuss the case when jointly training value functions and a dynamics model. When the MSE loss is used as in TD-MPC, we found that training is stabilized due to the regularization from value function learning. However, as shown in the middle row of Figure 5, the consistency loss (shown in the log scale) will dominate the overall loss. This will lead to a sub-optimal solution as observed in the Dog Run task as well as visual control tasks (Hansen et al., 2022).

Since we found that cosine loss works better than MSE loss in the TCRL-dynamics variant, a natural idea is to use cosine loss to replace the MSE loss in TD-MPC, which can potentially solve the dominating issue. However, as shown in the bottom right plot of Figure 5, a worse performance is observed. To explain it, we should notice that learning a value function is not supervised training. Many ways have been proposed to stabilize training, but perturbation during training still exists. The bottom left plot of Figure 5 shows the maximal latent state value (in the log scale) in training batches. As we can see, the magnitude of the latent space keeps growing. When training the value function and the latent dynamics model separately as in TCRL, the latent space is much more stable and achieves better performance.

Except for training stability, training a value function and dynamics model separately potentially enables better generalization ability as the encoder does not encode strong task-specific features from the value function (Sekar et al., 2020; Zhang et al., 2018). We will leave this as future work.

Utilizing the dynamics model in policy training Here, we use the learned latent dynamics model to train the policy and value function by i) model-based value expansion (TCRL-mve) (Feinberg et al., 2018), where we use the dynamics model to rollout a short trajectory and calculate k-step returns used to train the value function; ii) updating the policy by backpropagating through the dynamics model (TCRL-bp) (Deisenroth & Rasmussen, 2011; Hafner et al., 2020); iii) policy-guided decision-time planning as in TD-MPC (TCRL-mppi).

Although the TCRL-dynamics variant showed that our learned latent dynamics model is accurate to support planning, Figure 6 shows performance drops with the TCRL-mve and TCRL-bp variants, which is a common issue in model-based RL (Feinberg et al., 2018; Chua et al., 2018). In RL, the policy tends to exploit model errors decreasing asymptotic performance. We found that on most tasks, TCRL-bp achieves reasonable performance, but performance drops happen on the Dog Run task. However, as shown in Figure 6, decision-time planning (TCLR-mppi) benefits from the learned dynamics model achieving better performance on the Acrobot Swingup and Dog Run tasks than both TCRL and TD-MPC. TCLR-mppi is a safe way to utilize the learned dynamics model as the dynamics model only influences the collection of informative samples, but is not directly involved in policy or value function updating. Note that TCLR-mppi does not use prioritized experience replay (Schaul et al., 2016) and pre-task action repeat (control frequency) as used in TD-MPC, which may in some tasks lead to slightly worse performance than TD-MPC, such as in Cheetah Run.

6. Conclusion

In this paper, we propose a simple yet efficient way to learn a latent representation based on temporal consistency. We show that the learned representation benefits policy learning to solve challenging state-based Humanoid and Dog tasks, outperforming model-free methods by a large margin and matching model-based methods with respect to sample efficiency but being 2.4 $\times$ faster to train. Furthermore, we show that the learned latent dynamics model is accurate enough to support online planners to solve high-dimensional tasks, outperforming previous ensemble-based methods but being much faster to train. Yet we believe that our method can be improved in different ways, for example, by using the learned model to improve the policy or value function learning, or extending to visual control tasks.

Acknowledgements

We acknowledge the computational resources provided by the Aalto Science-IT project and CSC, Finnish IT Center for Science, and, funding by Academy of Finland (345521).

References

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., and Riedmiller, M. A. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018.

Abel, D., Hershkowitz, D. E., and Littman, M. L. Near optimal behavior via approximate state abstraction. In *International Conference on Machine Learning*, volume 48, pp. 2915–2923, 2016.

Andre, D. and Russell, S. J. State abstraction for programmable reinforcement learning agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 119–125, 2002.

Andrychowicz, M., Baker, B., Chociej, M., Józefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., and Zaremba, W. Learning dexterous in-hand manipulation. *International Journal of Robotics Research*, 39(1), 2020.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Camacho, E. F. and Alba, C. B. *Model Predictive Control*. Springer science & business media, 2013.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. E. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, volume 119, pp. 1597–1607, 2020.

Chen, X., Wang, C., Zhou, Z., and Ross, K. W. Randomized ensembled double q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2021.

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Clavera, I., Fu, Y., and Abbeel, P. Model-augmented actor-critic: Backpropagating through paths. In *International Conference on Learning Representations*, 2020.

Dearden, R. and Boutilier, C. Abstraction and approximate decision-theoretic planning. *Artificial Intelligence*, 89 (1-2):219–283, 1997.

Deisenroth, M. and Rasmussen, C. E. Pilco: A model-based and data-efficient approach to policy search. In *International Conference on machine learning*, 2011.

Feinberg, V., Wan, A., Stoica, I., Jordan, M. I., Gonzalez, J. E., and Levine, S. Model-based value expansion for efficient model-free reinforcement learning. In *International Conference on Machine Learning*, 2018.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1587–1596, 2018.

Furuta, H., Kozuno, T., Matsushima, T., Matsuo, Y., and Gu, S. S. Co-adaptation of algorithmic and implementational innovations in inference-based deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 9828–9842, 2021.

Gelada, C., Kumar, S., Buckman, J., Nachum, O., and Bellemare, M. G. Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, pp. 2170–2179, 2019.

Ghugare, R., Bharadhwaj, H., Eysenbach, B., Levine, S., and Salakhutdinov, R. Simplifying model-based RL: learning representations, latent-space models, and policies with one objective. In *International Conference on Learning Representations*, 2023.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. Bootstrap your own latent–a new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21271–21284, 2020.

Ha, D. and Schmidhuber, J. Recurrent world models facilitate policy evolution. *Advances in Neural Information Processing Systems*, 31, 2018.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1861–1870, 2018.

Hadsell, R., Chopra, S., and LeCun, Y. Dimensionality reduction by learning an invariant mapping. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pp. 1735–1742. IEEE, 2006.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pp. 2555–2565, 2019.

Hafner, D., Lillicrap, T. P., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

Hafner, D., Lillicrap, T. P., Norouzi, M., and Ba, J. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021.

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

Hansen, N., Su, H., and Wang, X. Temporal difference learning for model predictive control. In *International Conference on Machine Learning*, volume 162, pp. 8387–8406, 2022.

Hasselt, H. Double q-learning. In *Advances in Neural Information Processing Systems*, volume 23, 2010.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

Heess, N., Wayne, G., Silver, D., Lillicrap, T., Erez, T., and Tassa, Y. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

Higgins, I., Matthey, L., Pal, A., Burgess, C. P., Glorot, X., Botvinick, M. M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

Huang, T., Wang, J., and Chen, X. Accelerating representation learning with view-consistent dynamics in data-efficient reinforcement learning. *arXiv preprint arXiv:2201.07016*, 2022.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Jiang, N., Kulesza, A., and Singh, S. Abstraction selection in model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 179–188, 2015.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. Morel: Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21810–21823, 2020.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Lambert, N. O., Amos, B., Yadan, O., and Calandra, R. Objective mismatch in model-based reinforcement learning. In *Learning for Dynamics and Control (L4DC)*, volume 120, pp. 761–770, 2020.

Laskin, M., Srinivas, A., and Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pp. 5639–5650, 2020.

Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V., and Hutter, M. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47):eabc5986, 2020.

Li, L., Walsh, T. J., and Littman, M. L. Towards a unified theory of state abstraction for mdps. In *International Symposium on Artificial Intelligence and Mathematics*, 2006.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

Lutter, M., Hasenclever, L., Byravan, A., Dulac-Arnold, G., Trochim, P., Heess, N., Merel, J., and Tassa, Y. Learning dynamics models for model predictive agents. *arXiv preprint arXiv:2109.14311*, 2021.

Ma, X., Chen, S., Hsu, D., and Lee, W. S. Contrastive variational reinforcement learning for complex observations. In *Conference on Robot Learning*, volume 155, pp. 959–972, 2020.

Mannor, S., Menache, I., Hoze, A., and Klein, U. Dynamic abstraction in reinforcement learning via clustering. In *International Conference on Machine Learning*, pp. 71, 2004.

McInroe, T., Schäfer, L., and Albrecht, S. V. Learning temporally-consistent representations for data-efficient reinforcement learning. *arXiv preprint arXiv:2110.04935*, 2021.

Nguyen, T. D., Shu, R., Pham, T., Bui, H., and Ermon, S. Temporal predictive coding for model-based planning in latent space. In *International Conference on Machine Learning*, pp. 8130–8139, 2021.

Oh, J., Singh, S., and Lee, H. Value prediction network. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Okada, M. and Taniguchi, T. Dreaming: Model-based reinforcement learning by latent imagination without reconstruction. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4209–4215, 2021.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

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, 2017.

Peng, J. and Williams, R. J. Incremental multi-step q-learning. In *Machine Learning Proceedings 1994*, pp. 226–232. Elsevier, 1994.

Peng, X. B., Abbeel, P., Levine, S., and Van de Panne, M. deepmimic: example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)*, 37(4):1–14, 2018.

Pont-Tuset, J., Perazzi, F., Caelles, S., Arbeláez, P., Sorkine-Hornung, A., and Van Gool, L. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017.

Ruan, S., Comanici, G., Panangaden, P., and Precup, D. Representation discovery for mdps using bisimulation metrics. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. In *International Conference on Learning Representations*, 2016.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609, 2020.

Schwarzer, M., Anand, A., Goel, R., Hjelm, R. D., Courville, A. C., and Bachman, P. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2021.

Sekar, R., Rybkin, O., Daniilidis, K., Abbeel, P., Hafner, D., and Pathak, D. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, pp. 8583–8592, 2020.

Silver, D., Hasselt, H., Hessel, M., Schaul, T., Guez, A., Harley, T., Dulac-Arnold, G., Reichert, D., Rabinowitz, N., Barreto, A., et al. The predictron: End-to-end learning and planning. In *International Conference on Machine Learning*, pp. 3191–3199, 2017.

Singh, S., Jaakkola, T., and Jordan, M. Reinforcement learning with soft state aggregation. In *Advances in Neural Information Processing Systems*, volume 7, 1994.

Stone, A., Ramirez, O., Konolige, K., and Jonschkowski, R. The distracting control suite–a challenging benchmark for reinforcement learning from pixels. *arXiv preprint arXiv:2101.02722*, 2021.

Tamar, A., Wu, Y., Thomas, G., Levine, S., and Abbeel, P. Value iteration networks. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Tian, Y., Chen, X., and Ganguli, S. Understanding self-supervised learning dynamics without contrastive pairs. In *International Conference on Machine Learning*, pp. 10268–10278, 2021.

Tunyasuvunakool, S., Muldal, A., Doron, Y., Liu, S., Bohez, S., Merel, J., Erez, T., Lillicrap, T., Heess, N., and Tassa, Y. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

Watkins, C. J. C. H. *Learning from Delayed Rewards*. PhD thesis, King's College, 1989.

Watter, M., Springenberg, J., Boedecker, J., and Riedmiller, M. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

Williams, G., Aldrich, A., and Theodorou, E. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015.

Wu, P., Escontrela, A., Hafner, D., Abbeel, P., and Goldberg, K. Daydreamer: World models for physical robot learning. In *Conference on Robot Learning*, volume 205, pp. 2226–2240, 2022.

Yang, M. and Nachum, O. Representation matters: Offline pretraining for sequential decision making. In *International Conference on Machine Learning*, pp. 11784–11794, 2021.

Yarats, D., Zhang, A., Kostrikov, I., Amos, B., Pineau, J., and Fergus, R. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 10674–10681, 2021.

Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. Mastering visual continuous control: Improved data-augmented reinforcement learning. In *International Conference on Learning Representations*, 2022.

Ye, W., Liu, S., Kurutach, T., Abbeel, P., and Gao, Y. Mastering atari games with limited data. In *Advances in Neural Information Processing Systems*, volume 34, pp. 25476–25488, 2021.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. Mopo: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems*, volume 33, pp. 14129–14142, 2020.

Zhang, A., Satija, H., and Pineau, J. Decoupling dynamics and reward for transfer learning. In *International Conference on Learning Representations*, 2018.

Zhang, A., McAllister, R. T., Calandra, R., Gal, Y., and Levine, S. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2021.

A. Planning Performance in Pixel-based Tasks

A.1. Pixel-based Control Tasks

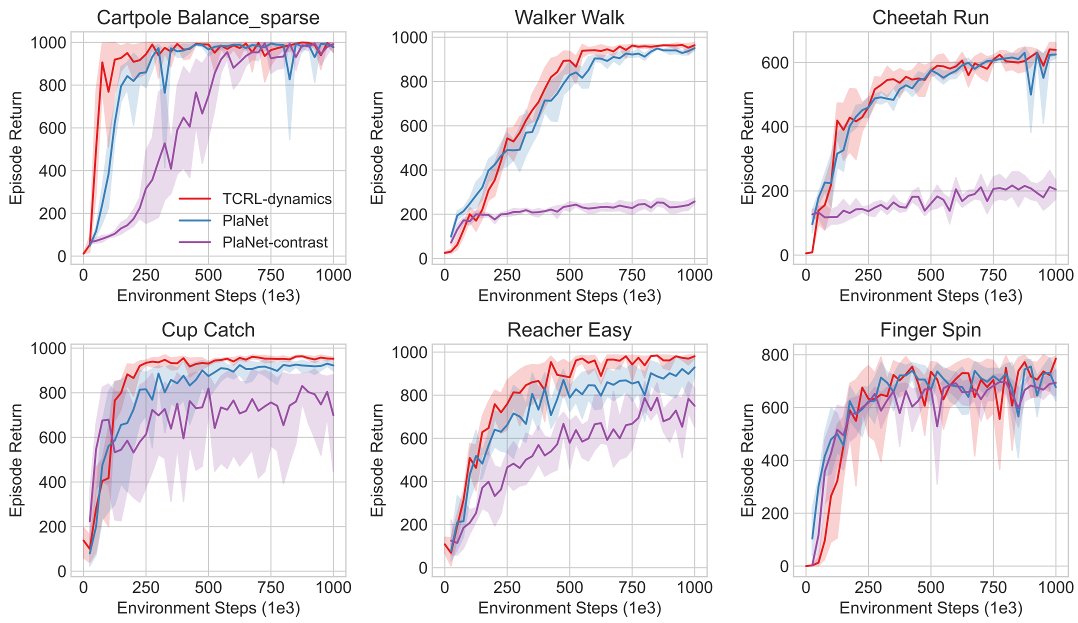

Figure 7. Planning performance on pixel-based DMC benchmarks. We plot the mean across 5 runs. Each run includes 10 trajectories. Shaded areas denote 95% confidence intervals. Our TCRL-dynamics matches PlaNet while being 5.51× faster to train. TCRL-dynamics outperforms PlaNet with contrastive objective by a large margin.

In Figure 7, we compare TCRL-dynamics with PlaNet (Hafner et al., 2019) and PlaNet with contrastive loss (Hafner et al., 2019; Ma et al., 2020). To extend TCRL-dynamics from state-based tasks to visual control tasks, we use a convolutional neural networks (CNN) to represent the encoder and inputs are three stacking images. Following Yarats et al. (2021), we use random crop data augmentation. PlaNet is a strong baseline in pixel-based tasks. It uses the recurrent state space model (RSSM), which uses an RNN and separated deterministic and stochastic paths to model the dynamics. PlaNet-contrast replaces the sequential VAE-like loss in PlaNet with a contrastive loss. We re-implement PlaNet using Pytorch (Paszke et al., 2019) to have a fair computational time comparison. To match our method, the learning rate is changed from 1e-3 to 5e-4, and the state dimension is increased from 30 to 50. Furthermore, we use MPPI instead of CEM as the planner to select actions. We use the same hyperparameters as in state-based tasks listed in Appendix B, but increase the population, elite size and iteration to 1000, 100 and 10 respectively. We verify that, with MPPI, our implementation outperforms the original results (Hafner et al., 2019).

Though our method only uses tacking images as inputs and adopts a simple architecture, it achieves competitive performance on most tasks and outperforms PlaNet on Cup Catch and Reacher Easy tasks. In these two tasks, the ball or the goal position only takes tens of pixels. As mentioned by Okada & Taniguchi (2021), the RSSM can still obtain low reconstruction error even totally ignoring these pixels. Our method does not suffer from this issue since we do not reconstruct the observations. Our method also obviously outperforms PlaNet-contrast, showing the effectiveness of the temporal consistency objective.

Furthermore, due to the simplicity of our method, it is much faster to train. We evaluate the training time on a single RTX 2080Ti GPU. On the Cartpole Blance_sparse task, PlaNet takes 15.6 hours for 500 episodes while TCRL-dynamics only takes **2.83** hours, which is **5.51** × faster than PlaNet. We hope our results can speed up model-based RL research.

A.2. Distracting Control Tasks

In the real world, observations are complex, while observations from DMC have plain backgrounds making it easy to distinguish interested objects. To test the performance of the learned model under complex observation, we test our methods as well as two pixel-based baselines using distracting control suite (Stone et al., 2021). Distracting control suite modifies

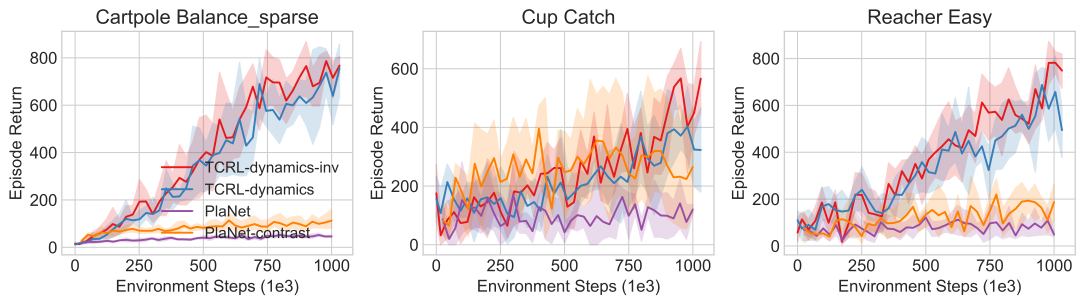

Figure 8. Planning results on distracting pixels-based tasks. We plot the mean across 5 runs and the shaded areas denote 95% confidence intervals. TCRL-dynamics outperforms PlaNet and PlaNet-contrast by a large margin.

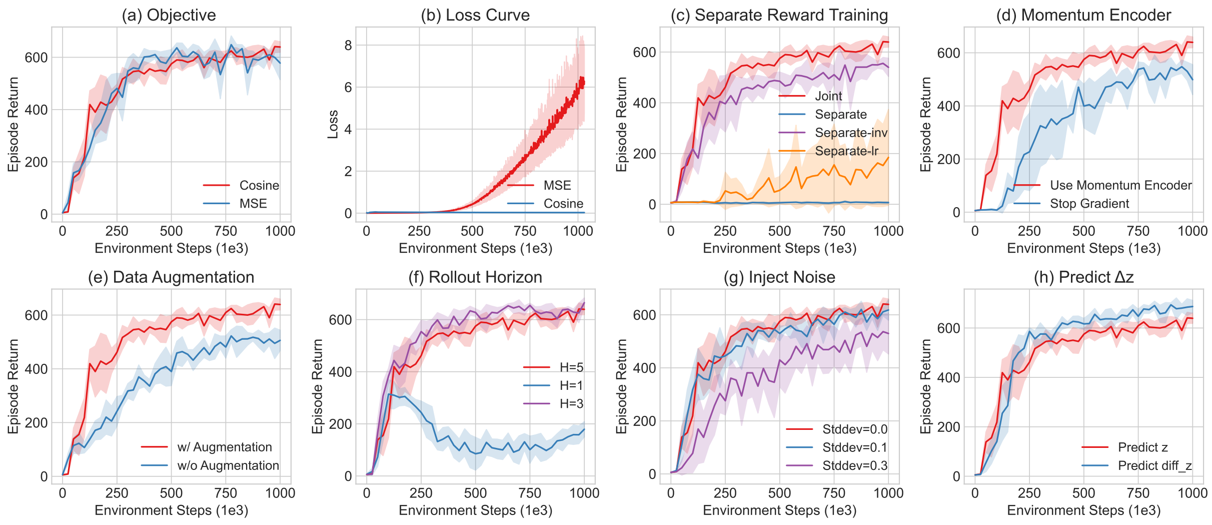

Figure 9. Ablation results on pixels-based tasks. The experiments are conducted on the Cheetah Run task.

upon standard DMC by adding natural backgrounds, color, and camera distractions. In our experiments, only natural background distractions are considered. Images of background distractions are video frames sampled from the DAVIS dataset (Pont-Tuset et al., 2017). In each episode, the background will be re-sampled from the dataset. We consider a simpler case where we only sample images from four pre-sampled videos.

To enable the dynamic model to focus on controllable objects, we add an additional *inverse dynamic model* (Pathak et al., 2017). The inverse dynamic model takes two adjacent latent states z_t, z_{t+1} as inputs and predicts the corresponding action a_t. As shown in Figure 8, our method is more robust to background distraction compared to PlaNet and PlaNet-contrast. We notice that PlaNet fails to solve these tasks completely. This is because PlaNet learns the dynamic model based on a VAE-like loss, which means it reconstructs the visual observation. However, on this distracting control suite, the backgrounds keep changing, making it hard to model. Also, only a few pixels are related to the control task, which can be easily ignored by PlaNet. While our method does not require to reconstruct observations, making it more robust in the distracting tasks. Furthermore, we observe that the inverse dynamic model helps the learning on both Cup Catch and Reacher Easy tasks.

A.3. Ablation Study for Visual Control

Representation collapse in pixel-based tasks In our method, perhaps the most urgent concern is whether there is a representation collapse, that is, whether there exists a trivial solution where the minimal loss is obtained when the encoder maps all observations into the same constant and the transition function is identical. In Figure 9(a-d), with Cheetah Run tasks, we investigate three factors: i) the loss function (MSE loss vs. cosine loss), ii) the jointly trained reward function and

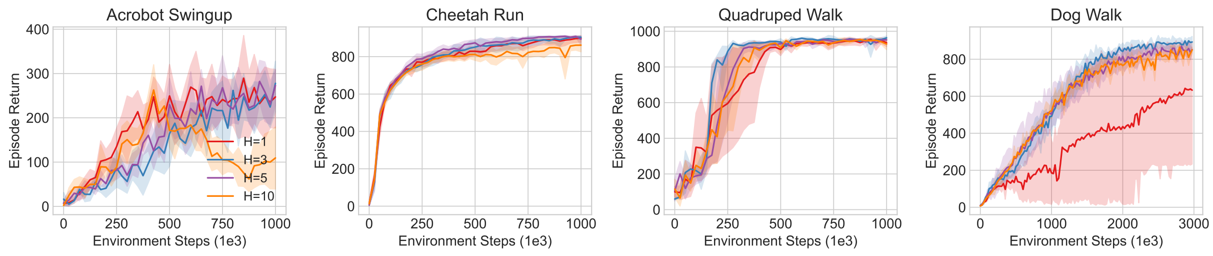

Figure 10. Comparison of model rollouts horizon (H) during training. We vary it from $[1, 3, 5, 10]$, and we find $H = 5$ works the best.

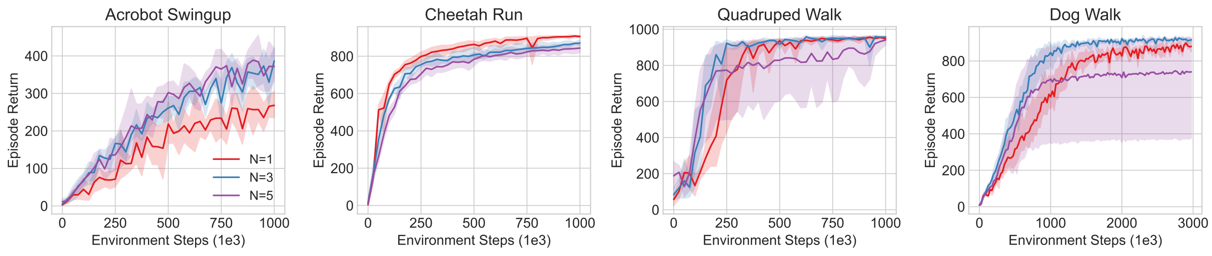

Figure 11. Comparison of choosing different N in N-step TD learning. We vary if from $[1, 3, 5]$ and we find $N = 3$ works the best.

iii) the momentum encoder.

Unlike (Schwarzer et al., 2021), when using the MSE loss over latent states, we do not observe representation collapse and the control performance is reasonable (See Figure 9(a-b)). However, the loss keeps growing, making training potentially unstable as discussed in section 5, while the cosine loss nicely behaves, which is aligned with our conclusion in state-based tasks, as in section 5.

When the reward function is trained separately from the encoder and the transition function (Figure 9c), representation collapse happens. Tian et al. (2021); Grill et al. (2020) mention that using a near-optimal projector avoids representation collapse, thus we adopt different learning rates for the transition model and the rest, specifically, the learning rate of the transition model is 5 times larger than the rest. It helps somehow, but in some runs, the collapse still happens. Furthermore, when we jointly train the inverse dynamic model with the encoder and the transition model, the representation collapse is successfully prevented. Our suggestion is to jointly train the latent dynamic model with the reward function whenever available and use the inverse dynamic model if necessary.

In Figure 9(d), when calculating the target latent states $\tilde{z}_t$ we compare usage of the momentum encoder $\tilde{z}_t = \mathbf{e}_{\theta^-}(o_t)$ by instead using the online encoder $\tilde{z}_t = \text{stop\_grad}(\mathbf{e}_\theta(o_t))$. We find that when the reward function is trained jointly, the second method is still able to avoid collapse but control performance drops.

What matters for model learning in pixel-based tasks? In Figure 9(e-h), we investigate several factors that are important to the model's performance, including (i) data augmentation, (ii) rollout length during training, (iii) injecting noise into latent states, and (iv) predicting the latent state difference $\hat{z}_{t+1} - \hat{z}_t$.

We find that data augmentation and multi-step prediction are critical to performance in pixel-based tasks, which is aligned with previous methods (Schwarzer et al., 2021; Hansen et al., 2022). Especially when using one-step prediction error during training ($H = 1$), the control performance drops dramatically, suggesting the advantage of using multi-step prediction. Nguyen et al. (2021) injects Gaussian noise to the latent state to smooth the dynamics, however, we do not find obvious benefits of using it. Also, injecting noise $\epsilon \sim \mathcal{N}(0, 0.3^2)$ as in Nguyen et al. (2021) hurts the performance. Furthermore, predicting the temporal difference of $\Delta z = z_{t+1} - z_t$ is slightly helpful but is not the major factor.

B. More Ablation Study in State-based Tasks

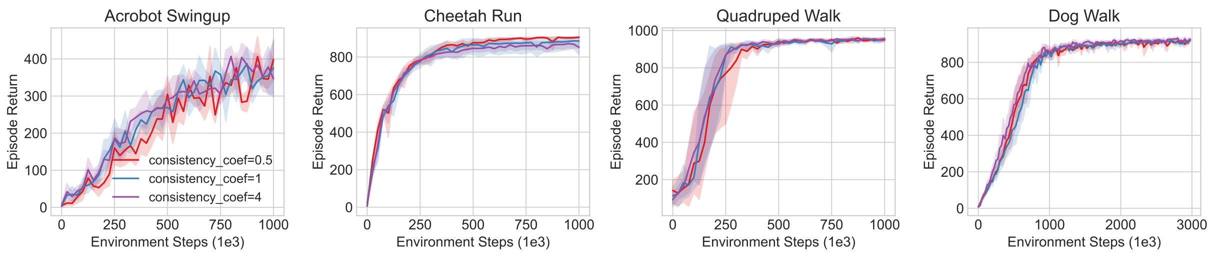

Figure 12. Comparison of different coefficients of the consistency_loss. We vary it from $[0.5, 1, 4]$ and we find the performance is robust w.r.t this hyperparameter. Thus we choose it as 1 in our experiments.

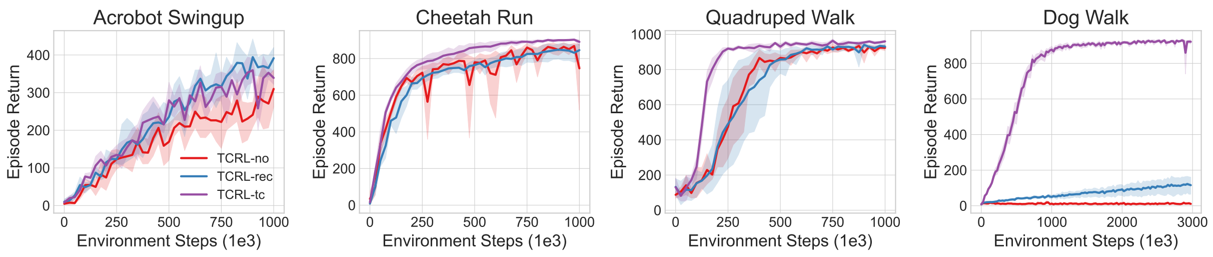

Figure 13. Comparison of different objective functions. We compare our method using temporal consistency loss with reconstruction loss (TCRL-rec) and dropping the loss by predicting rewards only (TCRL-no). All methods work reasonably on medium-level tasks but only our method can solve the complex Dog Walk task.

Hyperparameters We incrementally show each of the influences of three hyperparameters: model rollout horizon, n-step TD learning and coefficient of the consistency loss. We first decide the model rollout horizon during training, which is noted as H in Equation 2. We vary it from $[1, 3, 5, 10]$. As shown in Fig 10, we find that when $H = 1$, the performance decrease in the Dog Walk task, but if setting it as $H = 10$, performance drops are observed in Acrobot Swingup. Thus we set it as 5 in our experiments.

We use N-step TD learning as in Equation 3. Now we compare the influences of choosing different N by varying it from $[1, 3, 5]$. From Figure 11, we find that using $N = 3, 5$ improves the performance in Acrobot Swingup, and using $N = 5$ hurts the performance in Quadruped Walk and Dog Walk. Thus we choose $N = 3$ in our experiments.

Lastly, we decide on the coefficient of the consistency loss in Equation 2. We fix the weight of reward loss as 1 and test the coefficient of the consistency loss from $[0.5, 1, 4]$, as shown in Figure 12. We find that the algorithm is robust w.r.t this hyperparameter, so we select it as 1 in our experiment.

Different objective functions Our method exploits temporal consistency, and we compared it (TCRL-tc) with the contrastive objective in the pixel-based tasks. We now compare it with other training objectives in state-based tasks. Specifically, we compared it with i) reconstructing the observations, named TCRL-rec, and ii) without adding a loss function on the latent states, where the latent dynamic model is learned to accurately predict future rewards, named TCRL-no. As shown in Figure 13, our method using the temporal consistency loss works the best. Other objectives can also achieve reasonable performance on medium-level tasks, but TCRL is the only method that is able to solve the Dog Walk task (achieve more than 800 episodic returns). This experiment shows the effectiveness of leveraging temporal consistency in RL.

C. Baselines

C.1. More Baselines

We further compare our method with following baselines:

- **TD3**-like method is a strong model-free baseline. It is used in DrQv2 (Yarats et al., 2022), with two major differences

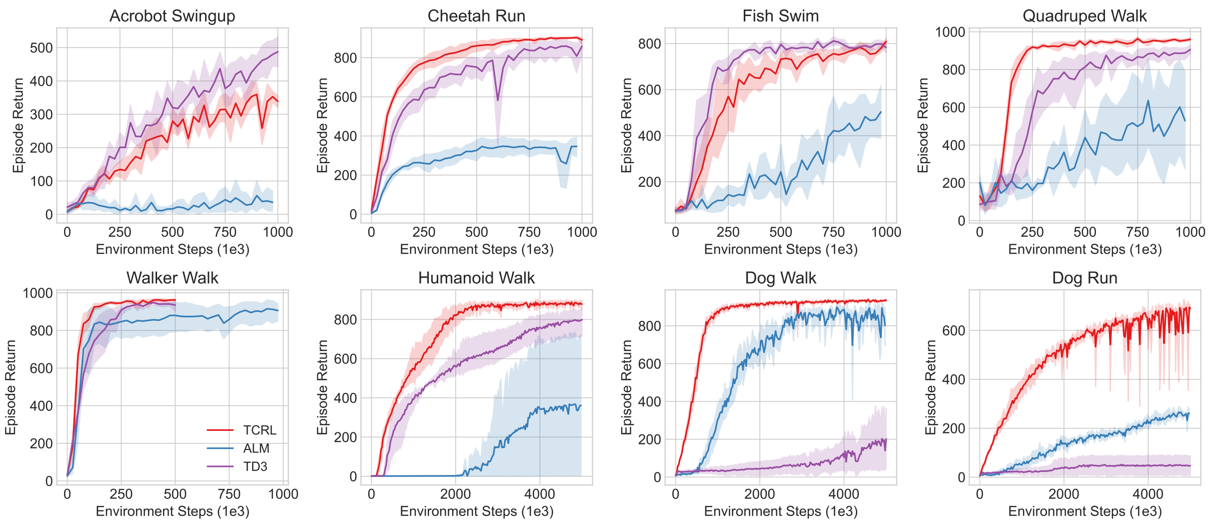

Figure 14. Policy performance on eight DMC tasks compared to ALM and TD3. We plot 5 random seeds with 95% confidence intervals presented by shaded areas.

from the original TD3 method (Fujimoto et al., 2018): i) it uses N-step TD learning and ii) it doesn't use the target policy network. We use it as the backend in TCRL to learn the policy and Q functions, thus the major difference between this baseline and TCRL is whether to use temporal consistency training. We include this baseline to stress the influence of representation learning.

- **ALM** (Ghugare et al., 2023) learns the representation, latent-space dynamics and the policy jointly. The policy is updated by recurrently backpropagating stochastic gradients (Heess et al., 2015). We reproduce ALM's results with author's source code. We change the testing suits from OpenAI Gym (Brockman et al., 2016) to DeepMind Control Suite (Tunyasuvunakool et al., 2020) without changing hyperparameters. We fail to achieve good performance in the tested environments but this may be due to the improper hyperparameters.

- **Dreamer V3** (Hafner et al., 2023) is the latest model-based reinforcement learning method that achieves strong performance on a diverse set of domains. The results of Dreamer V3, together with DDPG (Lillicrap et al., 2016) and MPO (Abdolmaleki et al., 2018), shown in Table 1 are from the original Dreamer V3 paper (Hafner et al., 2023). Notice that Dreamer V3 aims to work over a diverse set of domains, so the hyperparameters may not be optimized for continuous control tasks used in DMC, which potentially leads to lower performance.

Compared to the TD3-like baseline, we notice that TCRL hurts the performance on the Acrobot Swingup task and slightly on the Fish Swim task. However, TCRL outperforms the TD3-like baseline by a large margin on complex Humanoid and Dog domains.

C.2. Extended Description of Baselines

In this section, we describe how to obtain the baseline results and what modifications are made to have fair comparisons among different algorithms in our experiments.

- PETS: We implement PETS by referencing the public codebases[2] [3]. We align hyperparemeters with our TCRL-dynamics, but used a ensemble of dynamics models. To have similar amount of parameters of each model as our method, the reward and transition function share the common two-layer MLPs $[512, 512]$ and use separate heads with two-layer MLPs $[512, 512]$.

[2]Code, Library of model-based RL: https://github.com/facebookresearch/mbrl-lib

[3]Code, MBPO : https://github.com/zhaoyi11/mbpo-pytorch

Table 1. Policy performance on 15 DMC tasks compared with DDPG, MPO and Dreamer V3.

Task	DDPG	MPO	Dreamer V3	TCRL
Acrobot Swingup	92.7	80.6	154.5	**279.6** (σ 52.1)
Cartpole Swingup	**863.9**	**857.7**	**850.0**	**860.3** (σ 4.7)
Cartpole Swingup Sparse	**627.5**	519.9	468.1	**624.0** (σ 255.0)
Cheetah Run	576.9	612.3	575.9	**860.7** (σ 32.3)
Cup Catch	905.5	800.6	**958.2**	**976.3** (σ 2.1)
Finger Spin	753.6	766.9	**937.2**	849.1 (σ 13.3)
Finger Turn Easy	462.2	430.4	**745.4**	597.9(σ 228.1)
Finger Turn Hard	286.3	250.8	**841.0**	487.8 (σ 339.9)
Hopper Hop	24.6	37.5	111.0	**146.2** (σ 61.4)
Hopper Stand	388.1	279.3	573.2	**664.8** (σ 313.7)
Pendulum Swingup	748.3	**829.8**	766.0	**830.3** (σ 21.0)
Reacher Easy	**921.8**	**954.4**	**947.1**	**938.5** (σ 88.3)
Reacher Hard	**944.2**	**914.1**	**936.2**	**935.7** (σ 66.1)
Walker Run	530.0	539.5	632.7	**717.7** (σ 46.7)
Walker Walk	**948.7**	**924.9**	**935.7**	**955.6** (σ 19.0)
Mean	605.0	586.6	695.5	**715.0**
Medium	627.5	612.3	766.0	**830.3**

- TD-MPC: We test TD-MPC using their original code[4] by changing the learning rate from 1e-3 to 3e-4 and set the update frequency from one update per environment step to every two environment steps. We also enlarge the encoder from [256, 256] to [512, 512] to have same architecture as TCRL.
- SAC: We obtain SAC's results by running the code implemented with Pytorch[5] and make a few changes to have a fair comparison. We change the architecture of the actor and critic networks from [1024, 1024] to [512, 512, 512], add LayerNorm and Tanh nonlinear functions after the first layer according to the recommendations from Furuta et al. (2021). We further replace the ReLU nonlinear function with ELU and change batch size from 1024 to 512. Furthermore, we set action repeat as two.
- REDQ: We implement REDQ by modifying the SAC's implementation with the reference of author's implementation[6]. We set the update-to-data ratio as 10 and reduce it to 1 for the Fish Swim task since performance collapse is observed on this task with a high ratio (10).
- ALM: We obtain the ALM's results by re-running the authors' implementation[7]. Except changing the testing environments from OpenAI Gym to DMC, we change the update frequency from one update per environment step to every two environment steps. We also increase the latent dimension of Humanoid and Dog to 100.

D. Hyperparameters

In this section, we list important hyparparameters used in both TCRL and TCRL-dynamics. For details, please check the released code[8]. For TCRL-dynamics, we use the same hyperparameters as TCRL for learning the encoder and the latent dynamics model, thus we only list additional hyperparameters used for planning.

E. Full Results

[4]Code, TD-MPC: https://github.com/nicklashansen/tdmpc
[5]Code, SAC: https://github.com/denisyarats/pytorch_sac
[6]Code, REDQ: https://github.com/watchernyu/REDQ
[7]Code, ALM: https://github.com/RajGhugare19/alm/tree/7f1afdfd92f212a9deaf81e47e8b529b4aec2ee0
[8]Code, TCRL: https://github.com/zhaoyi11/tcrl

Table 2. Important Hyperparameters used in TCRL and TCRL-dynamics.

Hyperparameter	**Value**
TCRL	
Seed episode	10
Action repeat	2
Update frequency	2
Replay buffer size	Unlimited
Replay sampling strategy	Uniform
Batch size	512
Learning rate	3e-4
Optimizer	Adam
MLPs	[512, 512]
MLP activation	ELU
Latent Dimension	100 (Dog, Humanoid)
	50 (otherwise)
Momentum coefficient (τ)	0.005
Discount (γ)	0.99
Rollout horizon (H)	5
Rollout discount	0.9
N-step TD	3
Reward coefficient	1
Temporal coefficient	1
Policy stddev schedule	Linear(1.0, 0.1, 50) (easy)
	Linear(1.0, 0.1, 150) (medium)
	Linear(1.0, 0.2, 500) (hard)
Policy stddev clip	0.3
TCRL-dynamics	
Planner	MPPI
Plan horizon (H)	12
Population size	512
Num. of elite	64
Iteration	6
Temperature	0.5
Momentum	0.1
Reuse solution	True
Action repeat	2 (Dog, Walker, Finger)
	4 (Reacher, Quadruped, Cheetah)
	6 (Cup)
	8 (Cartpole)

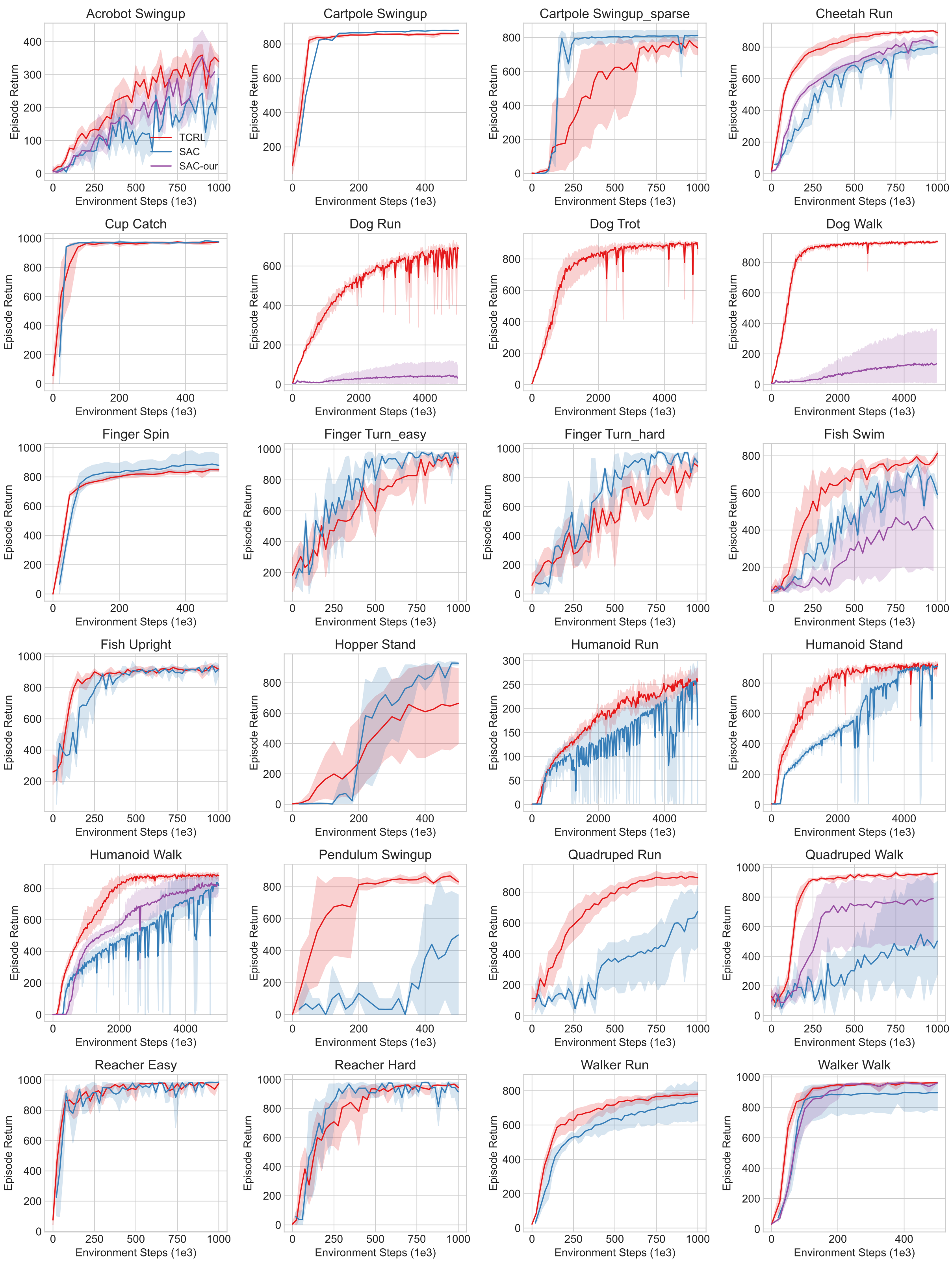

Figure 15. TCRL's results on 24 continuous control tasks from DMC. We plot 5 random seeds with 95% confidence intervals presented by shaded areas. SAC's results are from the public GitHub repository[4] and SAC-our's results are from Figure 3.