Temporal Consistency Improves Generalization in Contextual Offline Meta Reinforcement Learning

Mohammadreza Nakhaei
Aalto University *
mohammadreza.nakhaei@aalto.fi

Aidan Scannell
Toyota Research Institute †
aidan.scannell@tri.global

Kevin Sebastian Luck
Vrije Universiteit Amsterdam
k.s.luck@vu.nl

Joni Pajarinen
Aalto University
joni.pajarinen@aalto.fi

Abstract

Offline meta-reinforcement learning seeks to learn a policy that generalizes to new related tasks online. Context-based methods infer a task representation from transition histories, yet learning an effective task representation without supervision remains challenging. Existing methods relying on contrastive learning learn discriminative task representations, but fail to identify task-specific dynamics, while relying on reconstruction can be insufficient to model long-horizon dependencies, limiting generalization to new tasks. We investigate the impact of temporal consistency in latent space on task representation learning, showing that enforcing multi-step predictions in latent space encourages task representations that are able to capture task-dependent dynamics while preventing representation collapse. We provide theoretical analysis characterizing sources of error in value estimation and show through extensive experiments on MuJoCo, Contextual DeepMind Control, and MetaWorld benchmarks that temporal consistency significantly improves both zero-shot and few-shot generalization. Code is available at https://github.com/MohammadrezaNakhaei/SPC.

1 Introduction

One of the core challenges of reinforcement learning (RL) is *generalization*, where a policy trained on one task typically performs poorly when applied to a related task. Meta-reinforcement learning (meta-RL) addresses this issue by training on a distribution of tasks and learning a policy that can adapt quickly to new tasks [1, 2, 3]. However, most meta-RL methods require online interaction with numerous training tasks, which can be impractical in real-world scenarios. Offline meta-RL (OMRL) instead leverages access to offline datasets of related tasks and aims to learn a generalizable policy with limited additional environment interaction at test time.

A common approach in OMRL is *context encoding* [4, 5, 6, 7, 8, 9, 10]. In these methods, a context encoder maps a history of transitions (the context) to a latent vector termed the *task representation*. Policy and value functions are conditioned on this task representation, which serves as an implicit task identifier without requiring explicit task labels or knowledge of the underlying task variation factors. As a result, the quality of the task representation plays a central role in generalization to unseen tasks.

*Part of this work was conducted during a research visit to Vrije Universiteit Amsterdam.
†Work done whilst at University of Edinburgh.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Many existing methods rely solely on contrastive learning to learn the task representation [4, 11, 12]. While contrastive learning encourages task discrimination, it does not explicitly enforce predictive structure over time. As a result, the context encoder may fail to encode the underlying task variation factors, reflected in the reward function and transition dynamics, into the task representation, thereby limiting generalization. UNICORN [7] partially addresses this issue by including a one-step reconstruction objective as well, but one-step predictions might be insufficient to capture long-horizon, task-dependent dynamics.

Multi-step temporal consistency in latent space [13] forms a robust self-supervised learning signal for representation learning. In contrast to approaches that learn dynamics directly in observation or state space, world models project observations into a latent state representation and model dynamics within this latent manifold. They have been successfully used for decision-time planning and policy learning via imagination in model-based RL [14, 15, 16, 17, 18] and for representation learning in model-free RL [13, 19, 20]. Across these settings, a core theme is that self-supervised representation learning based on self-predictive RL [21] – also known as temporal consistency – provides a strong signal for learning latent state representations. However, directly applying latent dynamics in the OMRL setting is challenging, as the model must infer task-dependent variations without access to explicit task labels.

Contributions: In this work, we show that enforcing temporal consistency through self-predictive representations by jointly training the context encoder and world model yields better task representations than existing methods relying on contrastive learning and reconstruction, aka, one-step prediction. We study the quality of the learned context encoder through the lens of disentanglement and representation metrics, including feature rank, matrix rank, and dormant neuron ratio.

Our main contributions are:

- **C1 Temporal consistency for task inference.** We show that enforcing latent temporal consistency during context encoding yields task representations that capture task variation factors more effectively than reconstruction-based objectives based on disentanglement scores.
- **C2 Theoretical analysis.** We formally characterize the sources of error in value estimation and motivate temporal consistency on the latent space for task representation learning.
- **C3 Extensive empirical evaluation.** Through experiments on MuJoCo, Contextual DeepMind Control, and Meta-World benchmarks, we analyze the task representation quality and show that incorporating temporal consistency into the context encoder training objective significantly enhances few-shot and zero-shot generalization.

2 Background

Context-based Offline Meta-RL In offline meta-RL, we consider a set of training tasks, each modeled as a Markov Decision Process (MDP) $\mathcal{M}_i = \langle \mathcal{S}, \mathcal{A}, R_i, P_i, \gamma, \rho_0 \rangle$, where all tasks share the same state space $\mathcal{S}$, action space $\mathcal{A}$, discount factor $\gamma \in [0, 1]$, and initial state distribution ρ_0, but differ in their reward functions $R_i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and transition dynamics $P_i : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. Each task $\mathcal{M}_i$ is associated with an offline dataset $\mathcal{D}_i$.

The goal of offline meta-RL is to learn a meta-policy π that generalizes to unseen tasks, *i.e.*, maximizes the expected return over a distribution of test tasks:

$$J(\pi) = \mathbb{E}_{\mathcal{M}_i \sim p_{\text{test}}(\mathcal{M})} \left[\mathbb{E}_{\substack{\mathbf{s}_0 \sim \rho_0 \\ \mathbf{a}_t \sim \pi \\ \mathbf{s}_{t+1} \sim P_i(\mathbf{s}_t, \mathbf{a}_t)}} \left[\sum_{t=0}^{T} \gamma^t R_i(\mathbf{s}_t, \mathbf{a}_t) \right] \right]. \tag{1}$$

Context-based methods address task generalization by learning a *context encoder* that infers the underlying task from a small set of transitions. Specifically, a context encoder $E_\theta : \mathcal{S} \times \mathcal{A} \times \mathbb{R} \times \mathcal{S} \to \mathcal{Z}$ maps transitions $(\mathbf{s}_j, \mathbf{a}_j, r_j, \mathbf{s}'_j)$ to a task representation $\mathbf{z}$. The inferred task representation is then used to condition the policy $\pi(\mathbf{a}_t \mid \mathbf{s}_t, \mathbf{z})$, the value function $Q(\mathbf{s}_t, \mathbf{a}_t, \mathbf{z})$, or a learned dynamics model, enabling adaptation to different tasks without explicit task labels.

Self-predictive RL A state encoder $F_\phi : \mathcal{S} \to \mathcal{X}$ maps states $\mathbf{s} \in \mathcal{S}$ to latent states $\mathbf{x} \in \mathcal{X}$, also referred to as abstractions. These latent states are subsequently used by downstream components such

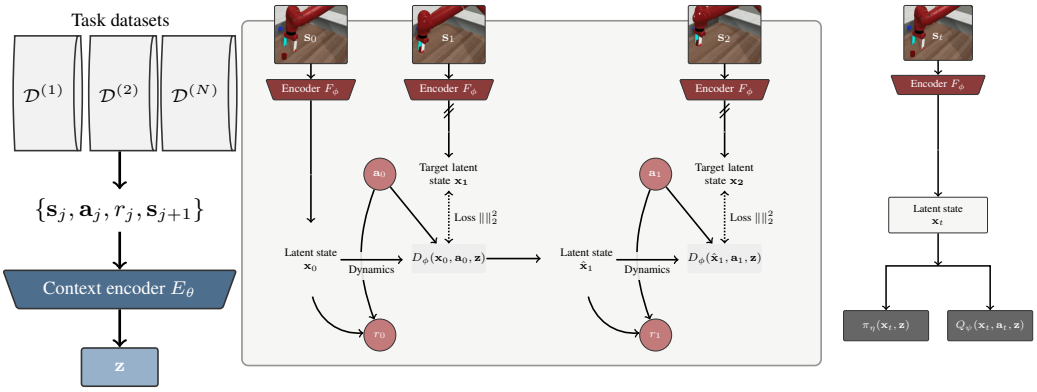


Figure 1: **Contextual World Model. Left:** A context encoder E_θ maps transitions from each task to a task representation $\mathbf{z}$, which serves as an implicit task identifier. **Middle:** An observation encoder maps observations $\mathbf{s}_t$ to latent vectors $\mathbf{x}_t$. A task-conditioned latent dynamics model D_ϕ predicts future latent states given the current latent state, action, and task representation. The world model is trained using multi-step temporal consistency. **Right:** An offline policy is trained using the latent states $\mathbf{x}_t$ and task representation $\mathbf{z}$.

as the policy, value function, or world model. Different methods train the encoder under different objectives and architectural couplings [21, 22, 23].

Self-supervised RL methods exploit (auxiliary) learning signals such as latent temporal consistency and latent reward prediction to train the encoder without relying on state reconstruction. Such an encoder should fulfill the following condition for learning optimal behavior [22]:

$$\text{RP: } \exists\, \hat{R} : \mathcal{X} \times \mathcal{A} \to \mathbb{R} \text{ s.t. } \mathbb{E}[r_t \mid \mathbf{s}_t, \mathbf{a}_t] = \hat{R}(\mathbf{x}_t, \mathbf{a}_t),$$
$$\text{ZP: } \exists\, \hat{P} : \mathcal{X} \times \mathcal{A} \to \Delta\mathcal{X} \text{ s.t. } \mathbb{E}[\mathbf{x}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t] = \hat{P}(\mathbf{x}_t, \mathbf{a}_t),$$

where $\mathbf{x}_t = F_\phi(\mathbf{s}_t)$.

The **RP** condition ensures that the latent state $\mathbf{x}$ contains sufficient information to predict the reward function, while the **ZP** condition enforces temporal consistency in the latent space $\mathcal{X}$. However, satisfying the **ZP** condition alone admits a trivial solution in which the encoder maps all states to a constant vector, resulting in complete representation collapse [23]. When both **RP** and **ZP** conditions are satisfied, the encoder retains the necessary information for predicting returns.

3 Joint Training of Context Encoder and World Model

To investigate temporal consistency for task representation learning in the OMRL setting, we condition a world model on the task representation (computed with the context encoder) and jointly train the world model and context encoder with self-supervision. The policy, conditioned on latent state and task representation, is optimized with offline RL. Figure 1 provides a high-level description of the joint training.

Components The contextual world model and offline RL consist of the following components:

Context encoder:	$\mathbf{z} = \mathbb{E}[E_\theta(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})]$	(2a)
Obs. encoder:	$\mathbf{x}_t = F_\phi(\mathbf{s}_t)$	(2b)
Latent dynamics:	$\hat{\mathbf{x}}_{t+1} = D_\phi(\mathbf{x}_t, \mathbf{a}_t, \mathbf{z})$	(2c)
Reward model:	$\hat{r}_t = R_\phi(\mathbf{x}_t, \mathbf{a}_t, \mathbf{z})$	(2d)
Q-function:	$q = Q_\psi(\mathbf{x}_t, \mathbf{a}_t, \mathbf{z})$	(2e)
Value function:	$v = V_\omega(\mathbf{x}_t, \mathbf{z})$	(2f)
Policy:	$\mathbf{a}_t \sim \pi_\eta(\mathbf{a}_t \mid \mathbf{x}_t, \mathbf{z})$	(2g)

The world model consists of the observation encoder F_ϕ, latent dynamics D_ϕ, and reward model R_ϕ. We denote the encoding as $\Phi(\mathbf{s}) \coloneqq F_\phi(\mathbf{s})$. The policy π_η, value function V_ω, and Q-function Q_ψ are conditioned on both the latent state $\mathbf{x}_t$ and the task representation $\mathbf{z}$. Unlike prior work, we jointly train the world model and the context encoder, allowing task inference to benefit directly from temporal consistency. Importantly, we avoid observation reconstruction and benefit from the powerful representation learning provided by a self-predictive loss.

3.1 Contextual World Model

In context-based OMRL, a context encoder E_θ infers the task from a small set of transitions. The observation encoder F_ϕ is then shared across all tasks, while the latent dynamics D_ϕ and reward model R_ϕ are conditioned on the inferred task representation $\mathbf{z}$. This reflects the fact that tasks differ in dynamics and reward functions.

Task representation. During training, we sample a meta-batch of tasks. For each task i with dataset $\mathcal{D}_i$, the task representation is given by

$$\mathbf{z}^i = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1}) \sim \mathcal{D}_i} \left[E_\theta(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1}) \right]. \tag{3}$$

World model. Observations from all tasks are mapped to continuous latent vectors using Eq. (2b).

The latent dynamics model predicts next latent state $\hat{\mathbf{x}}_{t+1}$ conditioned on the current latent state $\mathbf{x}_t$, action $\mathbf{a}_t$, and task representation $\mathbf{z}$. Rewards are also predicted by a task-conditioned reward model.

We train the context encoder jointly with the world model – observation encoder, latent dynamics, and reward function – using a multi-step temporal consistency (TC) objective, given by

$$\mathcal{L}_{\text{TC}}(\theta, \phi) = \sum_{h=0}^{H-1} \gamma^h \Big(\|(D_\phi(\hat{\mathbf{x}}_{t+h}, \mathbf{a}_{t+h}, \mathbf{z}) - \mathbf{x}_{t+h+1})\|_2^2 + \|R_\phi(\hat{\mathbf{x}}_{t+h}, \mathbf{a}_{t+h}, \mathbf{z}) - r_{t+h}\|_2^2 \Big), \tag{4}$$

where $\hat{\mathbf{x}}_t = F_\phi(\mathbf{s}_t)$ is the initial latent state, $\hat{\mathbf{x}}_{t+h+1}$ is computed from the latent dynamics model, and $\mathbf{x}_{t+h+1} = \text{sg}(F_{\bar{\phi}}(\mathbf{s}_{t+h+1}))$ is the target latent state computed using an exponential moving average (EMA) encoder. Here, H is the prediction horizon, γ is the discount factor, and $\bar{\phi}$ denotes EMA parameters. Additionally, contrastive learning, e.g., the InfoNCE loss (Eq. (29) in Sec. C.9), enhances the discriminability of task representations. We investigate the effects of training the context encoder using temporal consistency, contrastive learning, and their combination on task representation learning and downstream generalization.

3.2 Meta Policy Optimization

After learning the world model, we train a policy using offline reinforcement learning. We condition the policy and value functions on the latent state and task representation. To address distribution shift in offline RL, we adopt Implicit Q-Learning (IQL [24]). In Sec. C.12, we show evidence that this investigation is compatible with other offline RL methods.

IQL avoids out-of-distribution actions by learning a value function using expectile regression

$$\mathcal{L}_V(\omega) = \mathbb{E}\big[\mathcal{L}_2^\tau(Q_{\bar{\psi}}(\mathbf{x}_t, \mathbf{a}_t, \mathbf{z}) - V_\omega(\mathbf{x}_t, \mathbf{z}))\big],$$

where $\mathcal{L}_2^\tau(x) = (\tau - \mathbb{1}[x < 0])x^2$ and $\bar{\psi}$ denotes EMA parameters. The Q-function is trained using standard temporal-difference learning.

The policy is optimized using advantage-weighted regression [25], given by

$$\mathcal{L}_\pi(\eta) = \mathbb{E}\left[\exp(A(\mathbf{x}_t, \mathbf{a}_t, \mathbf{z})/\mathcal{B}) \log \pi_\eta(\mathbf{a}_t \mid \mathbf{x}_t, \mathbf{z})\right],$$

where $A(\mathbf{x}_t, \mathbf{a}_t, \mathbf{z}) = Q_\psi(\mathbf{x}_t, \mathbf{a}_t, \mathbf{z}) - V_\omega(\mathbf{x}_t, \mathbf{z})$ is the advantage and $\mathcal{B}$ is a temperature parameter.

It is worth noting that the world model is used solely for representation learning and not for planning. Specifically, observations are encoded into a latent space, and both the policy and value functions are learned directly over these latent states. Consequently, learning proceeds within the latent MDP induced by the encoder.

3.3 Theoretical Analysis

Our meta policy optimization operates in the learned latent space $(\mathbf{x}_t, \mathbf{z}_t)$ without reconstructing observations. Here we build on the well-known simulation lemma [26] and bound the value error incurred by *(i)* latent abstraction via $\Phi(\mathbf{s})$, *(ii)* learned world-model error, and *(iii)* task inference via the context encoder. All proofs are deferred to Appendix B.

Setup. For task i with MDP $\mathcal{M}_i = \langle \mathcal{S}, \mathcal{A}, R_i, P_i, \gamma, \rho_0 \rangle$ and bounded rewards $|R_i(\mathbf{s}, \mathbf{a})| \le R_{\max}$. Let the learned latent state be $\mathbf{x} = \Phi(\mathbf{s}) \coloneqq F_\phi(\mathbf{s}) \in \mathcal{X}$. Given $(\mathbf{x}, \mathbf{a}, \mathbf{z})$, the learned dynamics model induces a kernel $\widehat{P}_\phi(\cdot \mid \mathbf{x}, \mathbf{a}, \mathbf{z})$ over $\mathcal{X}$ and the learned reward model outputs $\widehat{R}_\phi(\mathbf{x}, \mathbf{a}, \mathbf{z})$. We use the infinite-horizon discounted value in MDP $\mathcal{M}_i$ as

$$V^\pi_{\mathcal{M}_i}(\mathbf{s}) = \mathbb{E}\left[\sum_{t \ge 0} \gamma^t R_i(\mathbf{s}_t, \mathbf{a}_t) \mid \mathbf{s}_0 = \mathbf{s},\ \mathbf{a}_t \sim \pi(\cdot \mid \mathbf{s}_t) \right].$$

Sources of error. We quantify three sources of error:

(1) Latent abstraction error. Let $P_i^\Phi(\mathbf{x}' \mid \mathbf{s}, \mathbf{a})$ be the pushforward transition induced by Φ (Appendix B). We assume there exists a Markov kernel $\bar{P}_i^\Phi(\cdot \mid \mathbf{x}, \mathbf{a})$ and latent reward function $\bar{R}_i^\Phi(\mathbf{x}, \mathbf{a})$ such that

$$\epsilon^P_{\text{abs}} \coloneqq \sup_{\mathbf{s},\mathbf{a}} \left\| P_i^\Phi(\cdot \mid \mathbf{s}, \mathbf{a}) - \bar{P}_i^\Phi(\cdot \mid \Phi(\mathbf{s}), \mathbf{a}) \right\|_1, \tag{5a}$$

$$\epsilon^R_{\text{abs}} \coloneqq \sup_{\mathbf{s},\mathbf{a}} \left| R_i(\mathbf{s}, \mathbf{a}) - \bar{R}_i^\Phi(\Phi(\mathbf{s}), \mathbf{a}) \right|. \tag{5b}$$

(2) World model approximation error (true task). When conditioned on the true task i, define

$$\epsilon^P_{\text{model}} \coloneqq \sup_{\mathbf{x},\mathbf{a}} \left\| \widehat{P}_\phi(\cdot \mid \mathbf{x}, \mathbf{a}, i) - \bar{P}_i^\Phi(\cdot \mid \mathbf{x}, \mathbf{a}) \right\|_1, \tag{6a}$$

$$\epsilon^R_{\text{model}} \coloneqq \sup_{\mathbf{x},\mathbf{a}} \left| \widehat{R}_\phi(\mathbf{x}, \mathbf{a}, i) - \bar{R}_i^\Phi(\mathbf{x}, \mathbf{a}) \right|. \tag{6b}$$

(3) Task inference error (conditioning on $\mathbf{z}$). Let $\mathbf{z} = g_\theta(\mathcal{H})$ be the task representation inferred from history $\mathcal{H}$ collected in task i. Define

$$\epsilon^P_{\text{task}} \coloneqq \sup_{\mathbf{x},\mathbf{a}} \left\| \widehat{P}_\phi(\cdot \mid \mathbf{x}, \mathbf{a}, \mathbf{z}) - \widehat{P}_\phi(\cdot \mid \mathbf{x}, \mathbf{a}, i) \right\|_1, \tag{7a}$$

$$\epsilon^R_{\text{task}} \coloneqq \sup_{\mathbf{x},\mathbf{a}} \left| \widehat{R}_\phi(\mathbf{x}, \mathbf{a}, \mathbf{z}) - \widehat{R}_\phi(\mathbf{x}, \mathbf{a}, i) \right|. \tag{7b}$$

Value error bound. Let $\pi(\mathbf{a} \mid \mathbf{x}, \mathbf{z})$ be any latent policy and define the lifted policy $\pi^\Phi(\mathbf{a} \mid \mathbf{s}, \mathbf{z}) \coloneqq \pi(\mathbf{a} \mid \Phi(\mathbf{s}), \mathbf{z})$. Let $V^{\pi^\Phi}_{\mathcal{M}_i}$ be its value in the original MDP and let $V^\pi_{\widehat{\mathcal{M}}(\mathbf{z})}$ be its value in the learned latent MDP induced by $(\widehat{P}_\phi(\cdot \mid \mathbf{x}, \mathbf{a}, \mathbf{z}), \widehat{R}_\phi(\mathbf{x}, \mathbf{a}, \mathbf{z}))$.

Theorem 3.1 (Value error from abstraction, model learning, and task inference)**.** *For any task i and any latent policy π,*

$$\sup_{\mathbf{s} \in \mathcal{S}} \left| V^{\pi^\Phi}_{\mathcal{M}_i}(\mathbf{s}) - V^\pi_{\widehat{\mathcal{M}}(\mathbf{z})}(\Phi(\mathbf{s})) \right| \le \frac{\epsilon^R}{1-\gamma} + \frac{\gamma R_{\max}}{(1-\gamma)^2} \epsilon^P,$$

$$\textit{with} \qquad \epsilon^R = \epsilon^R_{\text{abs}} + \epsilon^R_{\text{model}} + \epsilon^R_{\text{task}}, \tag{8a}$$

$$\epsilon^P = \epsilon^P_{\text{abs}} + \epsilon^P_{\text{model}} + \epsilon^P_{\text{task}}. \tag{8b}$$

Interpretation. The bound decomposes value error into: *(i)* latent abstraction error (how non-Markov the latent state $\mathbf{x}$ is), *(ii)* world model error (how accurately D_ϕ and R_ϕ match the induced latent MDP), and *(iii)* task inference error (how well $\mathbf{z}$ substitutes for the true task ID for predicting latent dynamics and rewards).

Implications for representation learning. Crucially, the bound in Theorem 3.1 does not require reconstructing observations from the latent state. Unlike reconstruction-based world models, which aim to preserve all information in $\mathbf{s}_t$, our analysis only requires that the learned representations $(\mathbf{x}, \mathbf{z})$ preserve information for predictive control.

Concretely, the abstraction error $(\epsilon^P_{\text{abs}}, \epsilon^R_{\text{abs}})$ is governed by how well the observation encoder Φ groups states with similar future latent dynamics and rewards. Our temporal consistency loss directly targets this by enforcing that $\mathbf{x}_{t+1}$ is predictable from $(\mathbf{x}_t, \mathbf{a}_t, \mathbf{z})$, rather than from raw observations. The world model error $(\epsilon^P_{\text{model}}, \epsilon^R_{\text{model}})$ is reduced by training the latent dynamics and reward models to accurately predict future latent states and rewards under the inferred task. Finally, the task inference error $(\epsilon^P_{\text{task}}, \epsilon^R_{\text{task}})$ is controlled by jointly training the context encoder with the world model, encouraging $\mathbf{z}$ to capture precisely the task-dependent factors needed for prediction.

Together, these objectives yield a latent representation that is sufficient for planning and control, without requiring reconstruction of $\mathbf{s}_t$. We provide a more detailed theoretical analysis in Appendix B.

4 Experiments

We evaluate temporal consistency for task representation learning on multi-task benchmarks from MuJoCo [27], Contextual DeepMind Control (DMC) [28, 29], and Meta-World [30]. Our experiments address the following questions:

RQ1 How does learning the context encoder jointly with the world model's self-supervised temporal consistency loss affect the task representation?

RQ2 Does temporal consistency for task representation learning reduce different sources of error described in our theoretical analysis (Sec. 3.3)?

RQ3 How does incorporating temporal consistency into training the context encoder impact the generalization to unseen tasks compared to prior methods?

Baselines. We compare against the following context-based OMRL methods (more details in Sec. A.2):

- **FOCAL** [4] trains the context encoder using a distance-based contrastive objective that pulls representations from the same task together and pushes different tasks apart.
- **CSRO** [5] extends FOCAL by reducing context distribution shift using CLUB as a mutual information regularizer [31].
- **UNICORN** [7] trains a context encoder using task-conditioned dynamics and reward models with one-step prediction. *UNICORN-SS* further augments this with a contrastive objective, whereas *UNICORN-SUP* relies exclusively reconstruction.
- **AutoRegression** where we extend *UNICORN-SS* to multi-step prediction by performing backpropagation through time in the state space. Comparing this baseline with the contextual world model highlights the role of multi-step temporal consistency in latent space for task representation learning.

Datasets. We generate offline datasets using Dropout Q-learning (DroQ) [32], training a separate agent for each task. Each dataset contains 1000 trajectories collected at different stages of training, ranging from random behavior to near-expert performance over 1M environment steps.

Benchmarks. For MuJoCo and Contextual-DMC, we use 20 training tasks, 10 in-distribution test tasks, and 10 out-of-distribution test tasks. Out-of-distribution tasks differ in the ranges of underlying variation factors.

In Meta-World, we follow standard Meta-RL settings (ML1, ML10, ML45). ML1 evaluates generalization across goals within a single environment. ML10 and ML45 evaluate generalization to unseen environments without task identifiers, using 10 and 45 training environments, respectively, and 5 held-out test environments.

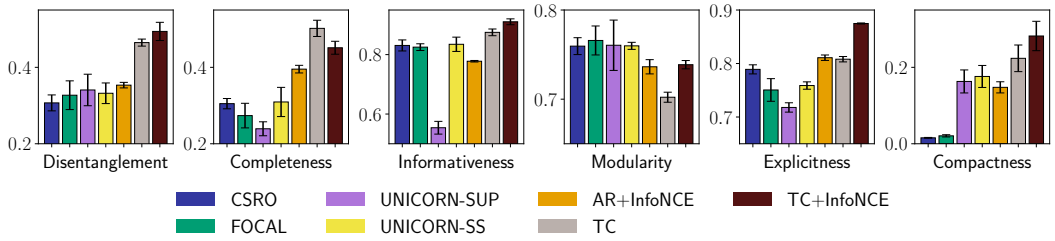


Figure 2: Disentanglement metrics (DCI, InfoMEC) for the Cheetah-length-speed (LS) environment. **Temporal consistency disentangles the variation factors more effectively, while contrastive learning enhances task distinguishability.** TC denotes training the context encoder solely with the temporal consistency objective, and AR denotes auto-regression. Averaged over 6 random seeds.

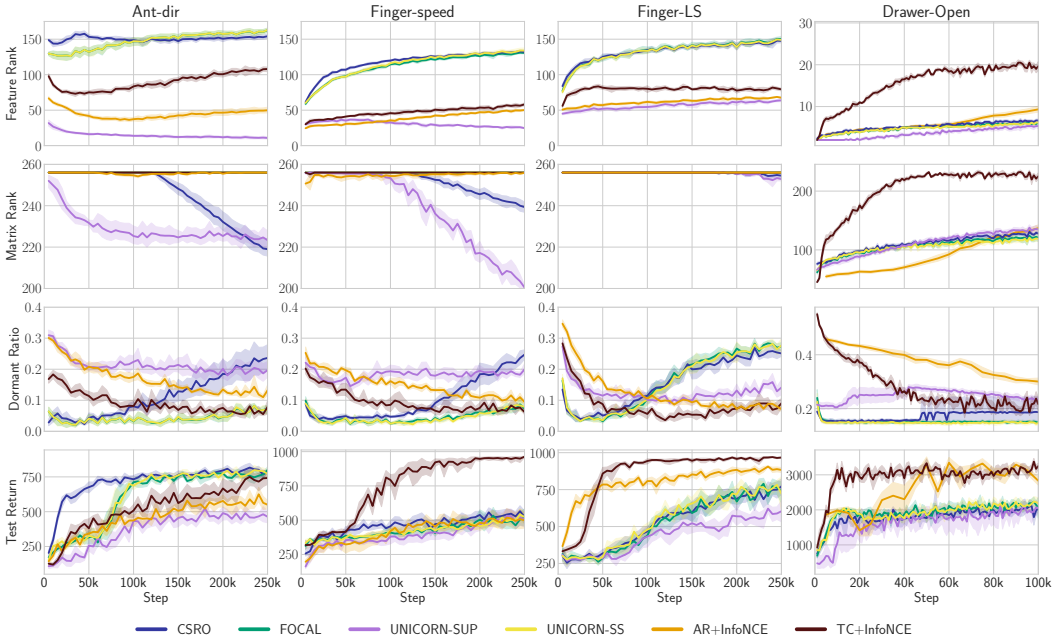


Figure 3: Representation metrics (Feature Rank, Rank, and Dormant ratio) of the context encoder. **Temporal consistency (TC) maintains low dormant neuron ratio and high matrix rank** compared to pure reconstruction (UNICORN-SUP) and auto-regression (AR). The shaded area represents 95% confidence interval across 6 random seeds.

4.1 Investigating Task Representation Learning

We first study how different training objectives affect the quality of the learned task representations. Previous work often relies on t-SNE visualizations to assess task separation, but such visualizations are qualitative and sensitive to hyperparameters. We instead use disentanglement metrics, which measure how well latent dimensions align with underlying task variation factors. We report DCI [33] metrics (disentanglement, completeness, informativeness) and InfoMEC [34] metrics (modularity, explicitness, compactness). Together, these metrics capture whether a task representation is structured, informative, and aligned with true task factors.

Figure 2 reports results for the Cheetah-LS environment, which varies desired speed and torso length. Training the context encoder using latent temporal consistency (*TC*) leads to better disentanglement than reconstruction-based training (UNICORN-SUP) and autoregression (*AR*). Adding contrastive objectives improves task distinguishability, reflected in higher informativeness and explicitness. Combining both objectives yields the strongest overall representations. Sec. C.4 presents disentanglement metrics for more environments, demonstrating consistent behavior.

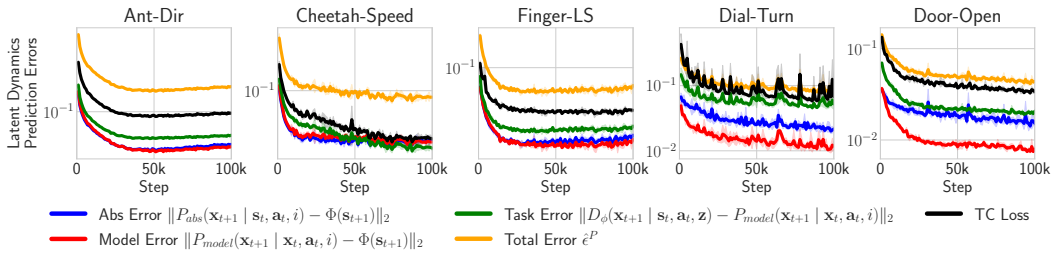


Figure 4: Estimating different error sources. **Joint training of the context encoder and world model reduces all error terms, with task error dominating; their sum upper-bounds the temporal consistency loss.** Abstraction error arises from mapping states to a latent space, model error reflects world model accuracy, and task error measures how well the context encoder identifies the task. Shaded regions denote 95% confidence intervals over three random seeds.

We analyze task representation learning using feature rank, matrix rank, and dormant neuron ratio (Figure 3). The matrix rank [23] quantifies correlations among learned features, while the dormant neuron ratio [35, 36] measures the proportion of neurons with near-zero activations, which limits the expressivity of the network. Given the representation matrix – defined as the activations of the last hidden layer of the context encoder – the feature rank [37, 38, 39, 40] measures feature diversity by identifying the smallest subspace that preserves $(1 - \varepsilon)\%$ of the total variance; throughout our experiments, we set $\varepsilon = 0.01$. Self-prediction learns less correlated features than reconstruction (UNICORN SUP), maintains higher matrix rank, and significantly reduces dormant neurons, suggesting that latent temporal consistency leads to more expressive and robust task representations [36]. Figure 7 provides representation metrics for more environments.

4.2 Error Decomposition

In this section, we conduct experiments to evaluate the theoretical analysis in Sec. 3.3. To quantify each error source, we train separate networks using privileged information, the true task ID i, unavailable in the offline meta-RL setting. Sec. C.2 provides more detail.

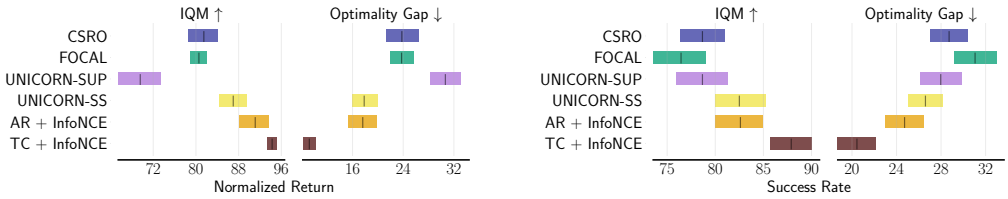
- **Abstraction error** A network $P_{\text{abs}}(\mathbf{x}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t, i)$ is trained to predict the next latent from the state, action, and task ID. The error is the discrepancy between this prediction and the encoded next state $\Phi(\mathbf{s})$ (Eq. (5a)).
- **World modeling error** A network $P_{\text{model}}(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{a}_t, i)$ is trained to predict the next latent from the current latent, action, and task ID. The error is measured against the encoded next state $\Phi(\mathbf{s})$ (Eq. (6a)).
- **Task error** We compare predictions of P_{model} with those of the contextual world model, which conditions on the learned task representation $\mathbf{z}$ instead of task ID i (Eq. (7a)).

Figure 4 illustrates the evolution of these errors during training. All networks are trained and evaluated simultaneously on the same transitions. Jointly training the context encoder and the world model reduces all three error terms, and their sum upper-bounds the temporal consistency loss. Task error is the dominant term in most experiments, indicating that the main source of error is due to estimating the underlying task from previous transitions with the context encoder.

4.3 Generalization to New Tasks

We evaluate whether improved task representations lead to better generalization. During meta-testing, the agent is not given task identifiers and must infer the task online from interaction data. The task representation is initialized to zero and updated at each step using collected transitions.

Figure 5 reports few-shot performance on in-distribution tasks. These results suggest that training the context encoder with both the latent temporal consistency and contrastive objectives improves both in-distribution performance and generalization. Sec. C.1 reports few-shot performance across benchmarks as well as zero-shot results on MuJoCo and Contextual-DMC. We also evaluate generalization to unseen environments in Meta-World ML10 and ML45 (Table 7 in Sec. C.1). Temporal consistency



(a) MuJoCo and Contextual DMC

(b) MetaWorld

Figure 5: Few-shot in-distribution performance. **Temporal consistency in the latent space improves generalization compared to reconstruction and auto-regression** (9 MuJoCo and Contextual DMC environments and 30 MetaWorld environments, each experiment with 6 random seeds).

with contrastive learning achieves higher success rates on both training and test environments, although generalization to entirely new environments remains challenging for all methods. Increasing the diversity of training environments improves test performance.

4.4 Further Results

Sec. C in the appendix presents additional empirical results and ablation studies. Sec. C.5 demonstrates that the learned world model can be used for online planning, where increasing the planning horizon generally improves zero-shot performance at the cost of increased computation. Additionally, incorporating a conservative value function improves planning performance and reduces the need for long-horizon planning. Sec. C.6 compares the proposed framework to DreamerV3 [41] and MetaDT [42], demonstrating better generalization of the proposed method. Sec. C.7 investigates alternative context-encoder architectures, including RNN- and transformer-based architectures, but does not find a significant benefit on the evaluated benchmarks.

Sec. C.8 provides an ablation study of different latent-space formulations, indicating that a discrete latent space with temporal consistency defined as next-code classification improves performance, consistent with the findings of [18]. Sec. C.9 further analyzes the impact of contrastive learning and latent temporal consistency objectives on generalization, demonstrating that incorporating temporal consistency into the context encoder objective improves generalization, particularly on out-of-distribution tasks. Sec. C.10 presents an ablation study on the training horizon in Eq. (4), showing that the proposed framework is robust to this hyperparameter, while longer horizons can negatively affect training stability, potentially due to backpropagation through time. Sec. C.11 demonstrates the performance with different model sizes, indicating that the proposed method scales effectively. Sec. C.12 provides an ablation study of the offline RL backbone used for policy optimization, showing that the proposed framework is relatively agnostic to the choice of offline RL algorithm. Sec. C.13 shows that bounding the task representation $\mathbf{z}$ improves performance across the evaluated bounding functions. Finally, Sec. C.14 compares the computational cost of the proposed framework with the baselines, showing that although each training iteration is more expensive for the proposed framework, it requires fewer iterations to converge, resulting in faster overall training.

5 Related Work

We review world models, self-predictive RL, and context-based offline meta-RL, highlighting how our method bridges gaps between these directions.

World models and self-predictive RL. World models learn compact state representations by mapping observations to a latent state space and learning to predict future latent states and rewards. They have been successfully used in model-based RL [14, 15, 16, 17, 18], and for representation learning in model-free RL [13, 19, 23, 20]. Across both settings, a unifying principle is *self-predictive RL* [21, 13], which provides a strong self-supervised signal for learning latent state spaces. Recent works have also shown that reconstruction is unnecessary and that enforcing multi-step latent consistency leads to improved performance [16, 18, 23]. However, these methods are typically

task-specific or assume access to a known task identifier, and therefore do not address generalization across tasks or environments.

Context-based offline meta-RL. Context-based offline meta-RL (OMRL) methods aim to generalize to unseen tasks by learning a task representation from a context of past transitions. Prior work trains context encoders using contrastive objectives [4, 11], but such approaches are sensitive to context distribution shift between training and testing. CSRO [5] and ER-TRL [8] mitigate this shift by regularizing the dependence between task representations and behavior policies. UNICORN [7] addresses task inference via predictive objectives, but operates directly in the observation space and does not learn a latent state space for control.

We unify these two lines of work. Unlike prior OMRL methods, we learn a latent state space using self-predictive world models without reconstructing observations. By training the world model and context encoder together, both the latent state $\mathbf{x}$ and task representation $\mathbf{z}$ capture the task-dependent predictive structure required for control.

6 Conclusion

This paper investigates temporal consistency for contextual OMRL, where we condition world models on learned task representations and jointly train the world model and context encoder without reconstruction. The resulting self-supervised representations capture task-relevant variation while avoiding representation collapse, leading to improved generalization to unseen tasks. We further provide value error bounds in the latent MDP induced by the observation encoder, showing that accurate control is possible without observation reconstruction. Overall, our results demonstrate that predictive latent representations suffice for generalization in offline meta-RL, and that jointly learning the context encoder and world model is both principled and effective.

Limitations Extending our experiments to accommodate different state and action spaces across tasks is a promising direction for future research. Furthermore, we evaluate temporal consistency for task representation learning in environments with only state information; in principle, the observation encoder architecture could be modified to support visual observations.

Acknowledgments and Disclosure of Funding

We acknowledge CSC – IT Center for Science, Finland, for awarding this project access to the LUMI supercomputer, owned by the EuroHPC Joint Undertaking, hosted by CSC (Finland) and the LUMI consortium through CSC. We acknowledge the computational resources provided by the Aalto Science-IT project. M. Nakhaei was supported by Business Finland (BIOND4.0 - Data Driven Control for Bioprocesses and Aurora - Automated and Connected Machines). A. Scannell was supported by the UK Engineering and Physical Sciences Research Council (grant EP/W002876/1). K. S. Luck was supported by the project ETAPE: Embodiment- and Task-Aware Parameter Embeddings for Robotic Foundation Models, file number NGF.1609.243.075, under the research programme National Growth Fund AiNed XS Europe 24-3, financed by the Dutch Research Council (NWO).

References

[1] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *International Conference on Machine Learning*, 2017.

[2] Luisa Zintgraf, Sebastian Schulze, Cong Lu, Leo Feng, Maximilian Igl, Kyriacos Shiarlis, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: Variational Bayes-adaptive deep RL via Meta-Learning. *Journal of Machine Learning Research*, 22, 2021.

[3] Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, Shimon Whiteson, et al. A Tutorial on Meta-Reinforcement Learning. *Foundations and Trends® in Machine Learning*, 18, 2025.

[4] Lanqing Li, Rui Yang, and Dijun Luo. FOCAL: Efficient Fully-Offline Meta-Reinforcement Learning via Distance Metric Learning and Behavior Regularization. In *International Conference on Learning Representations*, 2021.

[5] Yunkai Gao, Rui Zhang, Jiaming Guo, Fan Wu, Qi Yi, Shaohui Peng, Siming Lan, Ruizhi Chen, Zidong Du, Xing Hu, et al. Context Shift Reduction for Offline Meta-Reinforcement Learning. In *Advances in Neural Information Processing Systems*, 2024.

[6] Renzhe Zhou, Chen-Xiao Gao, Zongzhang Zhang, and Yang Yu. Generalizable Task Representation Learning for Offline Meta-Reinforcement Learning with Data Limitations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[7] Lanqing Li, Hai Zhang, Xinyu Zhang, Shatong Zhu, Yang Yu, Junqiao Zhao, and Pheng-Ann Heng. Towards an Information Theoretic Framework of Context-Based Offline Meta-Reinforcement Learning. In *Advances in Neural Information Processing Systems*, 2024.

[8] Mohammadreza Nakhaeinezhadfard, Aidan Scannell, and Joni Pajarinen. Entropy Regularized Task Representation Learning for Offline Meta-Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 2025.

[9] Jianhao Wang, Jin Zhang, Haozhe Jiang, Junyu Zhang, Liwei Wang, and Chongjie Zhang. Offline Meta Reinforcement Learning with In-Distribution Online Adaptation. In *International Conference on Machine Learning*, 2023.

[10] Hai Zhang, Boyuan Zheng, Tianying Ji, JinHang Liu, Anqi Guo, Junqiao Zhao, and Lanqing Li. Scrutinize What We Ignore: Reining in Task Representation Shift of Context-Based Offline Meta Reinforcement Learning. In *International Conference on Learning Representations*, 2025.

[11] Haoqi Yuan and Zongqing Lu. Robust Task Representations for Offline Meta-Reinforcement Learning via Contrastive Learning. In *International Conference on Machine Learning*, 2022.

[12] Xinyu Zhang, Wenjie Qiu, Yi-Chen Li, Lei Yuan, Chengxing Jia, Zongzhang Zhang, and Yang Yu. Debiased Offline Representation Learning for Fast Online Adaptation in Non-stationary Dynamics. In *International Conference on Machine Learning*, 2024.

[13] Yi Zhao, Wenshuai Zhao, Rinu Boney, Juho Kannala, and Joni Pajarinen. Simplified Temporal Consistency Reinforcement Learning. In *International Conference on Machine Learning*, 2023.

[14] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning Latent Dynamics for Planning from Pixels. In *International Conference on Machine Learning*, 2019.

[15] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. *Nature*, 588, 2020.

[16] Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal Difference Learning for Model Predictive Control. In *International Conference on Machine Learning*, 2022.

[17] Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, Robust World Models for Continuous Control. In *International Conference on Learning Representations*, 2024.

[18] Aidan Scannell, Mohammadreza Nakhaeinezhadfard, Kalle Kujanpää, Yi Zhao, Kevin Sebastian Luck, Arno Solin, and Joni Pajarinen. Discrete Codebook World Models for Continuous Control. In *International Conference on Learning Representations*, 2025.

[19] Scott Fujimoto, Wei-Di Chang, Edward Smith, Shixiang Shane Gu, Doina Precup, and David Meger. For SALE: State-Action Representation Learning for Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*, 2023.

[20] Scott Fujimoto, Pierluca D'Oro, Amy Zhang, Yuandong Tian, and Michael Rabbat. Towards General-Purpose Model-Free Reinforcement Learning. In *International Conference on Learning Representations*, 2025.

[21] Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-Efficient Reinforcement Learning with Self-Predictive Representations. In *International Conference on Learning Representations*, 2021.

[22] Tianwei Ni, Benjamin Eysenbach, Erfan SeyedSalehi, Michel Ma, Clement Gehring, Aditya Mahajan, and Pierre-Luc Bacon. Bridging state and history representations: Understanding self-predictive RL. In *International Conference on Learning Representations*, 2024.

[23] Aidan Scannell, Kalle Kujanpää, Yi Zhao, Mohammadreza Nakhaei, Arno Solin, and Joni Pajarinen. iQRL - Implicitly Quantized Representations for Sample-efficient Reinforcement Learning. *arXiv preprint arXiv:2406.02696*, 2024.

[24] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline Reinforcement Learning with Implicit Q-Learning. In *International Conference on Learning Representations*, 2022.

[25] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-Weighted Regression: Simple and Scalable Off-Policy Reinforcement Learning. *arXiv preprint arXiv:1910.00177*, 2019.

[26] Michael Kearns and Satinder Singh. Near-Optimal Reinforcement Learning in Polynomial Time. *Machine Learning*, 49, 2002.

[27] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A Physics Engine for Model-Based Control. In *International Conference on Intelligent Robots and Systems*, 2012.

[28] Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and Tasks for Continuous Control. *Software Impacts*, 6, 2020.

[29] Sahand Rezaei-Shoshtari, Charlotte Morissette, Francois R Hogan, Gregory Dudek, and David Meger. Hypernetworks for Zero-Shot Transfer in Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 2023.

[30] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement learning. In *Conference on Robot Learning*, 2020.

[31] Pengyu Cheng, Weituo Hao, Shuyang Dai, Jiachang Liu, Zhe Gan, and Lawrence Carin. Club: A Contrastive Log-ratio Upper Bound of Mutual Information. In *International Conference on Machine Learning*, 2020.

[32] Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout Q-Functions for Doubly Efficient Reinforcement Learning. In *International Conference on Learning Representations*, 2022.

[33] Cian Eastwood and Christopher K. I. Williams. A Framework for the Quantitative Evaluation of Disentangled Representations. In *International Conference on Learning Representations*, 2018.

[34] Kyle Hsu, Will Dorrell, James C. R. Whittington, Jiajun Wu, and Chelsea Finn. Disentanglement via Latent Quantization. In *Advances in Neural Information Processing Systems*, 2023.

[35] Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The Dormant Neuron Phenomenon in Deep Reinforcement Learning. In *International Conference on Machine Learning*, 2023.

[36] Shibhansh Dohare, J Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A Rupam Mahmood, and Richard S Sutton. Loss of Plasticity in Deep Continual Learning. *Nature*, 632, 2024.

[37] Clare Lyle, Mark Rowland, and Will Dabney. Understanding and Preventing Capacity Loss in Reinforcement Learning. In *International Conference on Learning Representations*, 2022.

[38] Maksym Andriushchenko, Dara Bahri, Hossein Mobahi, and Nicolas Flammarion. sharpness-Aware Minimization Leads to Low-rank Features. *Advances in Neural Information Processing Systems*, 2023.

[39] Skander Moalla, Andrea Miele, Daniil Pyatko, Razvan Pascanu, and Caglar Gulcehre. No Representation, No Trust: Connecting Representation, Collapse, and Trust Issues in PPO. In *Advances in Neural Information Processing Systems*, 2024.

[40] Johan Obando-Ceron, Walter Mayor, Samuel Lavoie, Scott Fujimoto, Aaron Courville, and Pablo Samuel Castro. Simplicial Embeddings Improve Sample Efficiency in Actor-Critic Agents. *arXiv preprint arXiv:2510.13704*, 2025.

[41] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering Diverse Control Tasks Through World Models. *Nature*, 2025.

[42] Zhi Wang, Li Zhang, Wenhao Wu, Yuanheng Zhu, Dongbin Zhao, and Chunlin Chen. Meta-DT: Offline Meta-RL as Conditional Sequence Modeling with World Model Disentanglement. In *Advances in Neural Information Processing Systems*, 2024.

[43] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An Imperative Style, High-performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, 2019.

[44] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2019.

[45] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

[46] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[47] Diganta Misra. Mish: A Self Regularized Non-Monotonic Neural Activation Function. *arXiv preprint arXiv:1908.08681*, 2019.

[48] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742, 2006.

[49] Vitchyr H Pong, Ashvin V Nair, Laura M Smith, Catherine Huang, and Sergey Levine. Offline meta-reinforcement learning with online self-supervision. In *International Conference on Machine Learning*, pages 17811–17829. Pmlr, 2022.

[50] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-Learning for Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems*, 2020.

[51] Andreas Doerr, Christian Daniel, Martin Schiegg, Nguyen-Tuong Duy, Stefan Schaal, Marc Toussaint, and Trimpe Sebastian. Probabilistic Recurrent State-Space Models. In *International Conference on Machine Learning*, 2018.

[52] Mengdi Xu, Yikang Shen, Shun Zhang, Yuchen Lu, Ding Zhao, Joshua Tenenbaum, and Chuang Gan. Prompting Decision Transformer for Few-Shot Policy Generalization. In *International Conference on Machine Learning*, 2022.

[53] Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taiga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, Aviral Kumar, and Rishabh Agarwal. Stop Regressing: Training Value Functions via Classification for Scalable Deep RL. In *International Conference on Machine Learning*, 2024.

[54] Scott Fujimoto and Shixiang (Shane) Gu. A Minimalist Approach to Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems*, 2021.

Appendices

In this appendix, we provide implementation details in Sec. A, theoretical analysis in Sec. B, and additional results in Sec. C.

Large Language Models

We use large language models (LLMs) to assist with paper writing, including proofreading for typos and grammar errors. We also employ LLMs to generate scripts for visualizing results and creating figures.

A Implementation Details

We used PyTorch [43] for implementation and the AdamW optimizer [44] for training the world model and the Adam optimizer [45] for the other models. All neural networks are implemented as MLPs where each intermediate linear layer is followed by Layer Normalization (*i.e.* pre-norm) [46] and a Mish activation function [47]. Below we summarize the architecture of the world model and policy for the Cheetah-LS environment.

Table 1: Hyperparameters for jointly training context encoder and world model.

Hyperparameter	Value	Description
Data Collection		
Train steps	10^6	
Random steps	5×10^4	Num. random steps at start
Num. eval episodes	50	Num. trajectories in evaluation
Eval. every steps	5×10^4	
Policy MLP dims	$[512, 512]$	
Value Function MLP dims	$[512, 512]$	
Dropout ratio	0.1	
Learning rate	10^{-4}	Value function, entropy coeff
	3×10^{-4}	Policy
Target Entropy	$-\|\mathcal{A}\|_1$	
Batch size	1024	
Discount factor γ	0.99	
Momentum coef	0.005	Soft update target networks
Contextual World Model		
Observation Encoder MLP dims	$[256]$	
Context Encoder MLP dims	$[256, 256]$	
Latent Dynamics and Reward MLP dims	$[512, 512]$	
Task representation dim	10	
Latent dim	512	
Consistency coeff	1.0	
Reward Coeff	1.0	
Discount Factor γ	0.99	
Training Horizon H	5	
Learning Rate	10^{-4}	
Contrastive Obj Weight β	1.0	
Momentum coef	0.005	Soft update target encoder
Offline Meta-RL		
Meta batch size	16	
Batch size	256	
Num. train task	20	mujoco & dmc
	40	metaworld
Num. eval task	10	
Num. OOD task	10	mujoco & dmc
	0	metaworld
Context size	64	
Buffer size	2×10^5	for each task
Discount factor γ	0.99	
(Q-)Value Function MLP dims	$[256, 256]$	
Num. Q Functions	2	
Policy MLP dims	$[256, 256]$	
Learning rate	3×10^{-4}	Policy and (Q-)Value functions
Expectile regression τ	0.8	
Inverse temperature $\mathcal{B}$	3.0	In policy optimization
Momentum coef.	0.005	soft update target networks
Num. test trajectories	3	k few-shot

```
Context Encoder: MLP(
  (net): Sequential(
    (0): NormedLinear(in_features=41, out_features=256, bias=True, act=Mish)
    (1): NormedLinear(in_features=256, out_features=256, bias=True, act=Mish)
    (2): NormedLinear(in_features=256, out_features=256, bias=True, act=Mish)
    (3): Linear(in_features=256, out_features=10, bias=True)
  )
)
World Model: ContextualWorldModel(
  (Encoder): MLP(
    (net): Sequential(
      (0): NormedLinear(in_features=17, out_features=256, bias=True, act=Mish)
      (1): Linear(in_features=256, out_features=512, bias=True)
    )
  )
  (Encoder_tar): MLP(
    (net): Sequential(
      (0): NormedLinear(in_features=17, out_features=256, bias=True, act=Mish)
      (1): Linear(in_features=256, out_features=512, bias=True)
    )
  )
  (Trans): MLP(
    (net): Sequential(
      (0): NormedLinear(in_features=534, out_features=512, bias=True, act=Mish)
      (1): NormedLinear(in_features=512, out_features=512, bias=True, act=Mish)
      (2): Linear(in_features=512, out_features=512, bias=True)
    )
  )
  (Reward): Vectorized [MLP(
    (net): Sequential(
      (0): NormedLinear(in_features=534, out_features=512, bias=True, act=Mish)
      (1): NormedLinear(in_features=512, out_features=512, bias=True, act=Mish)
      (2): Linear(in_features=512, out_features=1, bias=True)
    )
  )]
)
Policy: MLP(
  (net): Sequential(
    (0): NormedLinear(in_features=522, out_features=256, bias=True, act=Mish)
    (1): NormedLinear(in_features=256, out_features=256, bias=True, act=Mish)
    (2): Linear(in_features=256, out_features=12, bias=True)
  )
)
Q-Functions: Vectorized [MLP(
  (net): Sequential(
    (0): NormedLinear(in_features=534, out_features=256, bias=True, act=Mish)
    (1): NormedLinear(in_features=256, out_features=256, bias=True, act=Mish)
    (2): Linear(in_features=256, out_features=1, bias=True)
  )
), MLP(
  (net): Sequential(
    (0): NormedLinear(in_features=534, out_features=256, bias=True, act=Mish)
    (1): NormedLinear(in_features=256, out_features=256, bias=True, act=Mish)
    (2): Linear(in_features=256, out_features=1, bias=True)
  )
)]
Value Function: MLP(
  (net): Sequential(
    (0): NormedLinear(in_features=522, out_features=256, bias=True, act=Mish)
    (1): NormedLinear(in_features=256, out_features=256, bias=True, act=Mish)
    (2): Linear(in_features=256, out_features=1, bias=True)
  )
)
```

Hardware We used AMD Instinct MI250X GPUs to run our experiments. All experiments have been run on a single GPU with 2 CPU workers and 32GB of RAM.

Hyperparameters Table 1 illustrates the hyperparameters for our experiments. We use the same hyperparameters for all of the experiments. For a fair comparison, we use the same network architecture for all the baselines.

A.1 Environments

We evaluated all methods on 3 MuJoCo [27] environments, 6 Contextual DeepMind Control [28, 29] environments, and 50 Meta-World ML1 [3] environments. Table 2 provides details of the

environments we used, including the dimensionality of the observation and action spaces, and the distribution of variation factors. Below we summarize the environments:

- **Ant-direction:** an ant (quadruped) robot moving in different desired directions in different tasks.
- **Hopper-mass:** a hopper (one-legged robot) must move as fast as it can, while the mass is different for different tasks.
- **Walker-friction:** a walker (bi-legged) robot must move as fast as it can, while the friction coefficient is different for different tasks.
- **Cheetah-speed:** a cheetah robot moving forwards/backwards with different desired speeds in different tasks.
- **Finger-speed:** a planar finger robot rotating a body on an unactuated hinge with different desired angular speeds (both directions) for different tasks.
- **Walker-speed:** a walker robot moving forwards/backwards with different desired speeds in different tasks.
- **Cheetah-length-speed (LS):** a cheetah robot moving forwards, where torso length (change in morphology) and/or desired speed are different for each task.
- **Finger-length-speed (LS):** a planar finger robot rotating a body on an unactuated hinge, while the length of the link and/or desired angular speed differ in each task.
- **Walker-length-speed (LS):** a walker robot moving forwards, where torso length (change in morphology) and/or desired speed are different for each task.
- **Meta-World ML1**: consists of 50 robotic manipulation environments featuring a Sawyer arm with various everyday objects. Each environment consists of 50 different tasks where the position of objects and goals is different for each task.
- **Meta-World ML10:** evaluates generalization to new environments. Similar to the ML1 setting, it consists of robotic manipulation environments, where 10 environments are used for training and 5 environments are reserved for testing generalization capabilities. The testing environments share structural similarities with the training environments. During testing, no prior information about the environment (such as environment ID) is provided, and agents must identify and adapt to the environment solely based on interaction data.
- **Meta-World ML45:** evaluates generalization to new environments, similar to the ML10 setting, but with a larger and more diverse set of 45 training environments.

Table 2: Environment used for evaluation of different methods.

ENVIRONMENT	OBS DIM	ACTION DIM	ID VARIATION FACTORS	OOD VARIATION FACTORS
ANT-DIRECTION	29	8	$\theta \sim [-\pi, \pi]$	$\theta \sim [-1.5\pi, -\pi] \cup [\pi, 1.5\pi]$
HOPPER-MASS	11	3	$\log f_m \sim [-1.5, 1.5]$	$\log f_m \sim [-2, -1.5] \cup [1.5, 2]$
WALKER-FRICTION	17	6	$\log f_f \sim [-1.5, 1.5]$	$\log f_f \sim [-2, -1.5] \cup [1.5, 2]$
CHEETAH-SPEED	17	6	$v \sim [-10, -6] \cup [-2, 2] \cup [6, 10]$	$v \sim [-6, -2] \cup [2, 6]$
FINGER-SPEED	17	6	$v \sim [-15, -9] \cup [-3, 3] \cup [9, 15]$	$v \sim [-9, -3] \cup [3, 9]$
WALKER-SPEED	24	6	$v \sim [-5, -3] \cup [-1, 1] \cup [3, 5]$	$v \sim [-3, -1] \cup [1, 3]$
CHEETAH-LENGTH-SPEED	17	6	$v \sim [3, 8]$ $L \sim [0.4, 0.6]$	$v \in \{1, 2, 9, 10\}$ $L \in \{0.3, 0.35, 0.65, 0.7\}$
FINGER-LENGTH-SPEED	9	2	$v \sim [5, 10]$ $L \sim [0.15, 0.25]$	$v \in \{3, 4, 11, 12\}$ $L \in \{0.1, 0.12, 0.27, 0.3\}$
WALKER-LENGTH-SPEED	24	6	$v \sim [2, 4.5]$ $L \sim [0.2, 0.4]$	$v \in \{1, 1.5, 5, 5.5\}$ $L \in \{0.1, 0.15, 0.45, 0.5\}$
META-WORLD	39	4		

A.2 Baselines

In this section, we provide further details of the baselines we compare against.

FOCAL [4] utilizes distance metric learning to train the context encoder. Intuitively, the context encoder should map contexts from the same task closely in the latent task representation $\mathbf{z}$ and contexts from different tasks far apart. They show that pairwise contrastive learning [48] can lead to degenerate task representations, where embeddings from multiple tasks collapse into shared clusters. To encourage discriminative task representations, FOCAL proposes a negative-power variant of the contrastive loss:

$$\mathrm{L}_{\text{FOCAL}}(\phi) = \mathbb{1}\{i = j\}\|\mathbf{z}^i - \mathbf{z}^j\|_2^2 + \mathbb{1}\{i \neq j\}\frac{\beta}{\|\mathbf{z}^i - \mathbf{z}^j\|_2^2 + \epsilon_0},$$

wwhere ϕ denotes the parameters of the context encoder, $\mathbf{z}^i$ is the task representation corresponding to task i, and β is a hyperparameter. Although FOCAL improves over approaches that directly couple task representation learning with Q-learning, it does not explicitly model task-specific dynamics or rewards. Moreover, during inference, it assumes access to a context set sampled from a distribution similar to the training data, which limits applicability under distribution shift.

CSRO [5] utilizes adversarial training to mitigate context distribution shift [49]. Since the learned policy may differ from the behavior policies used to collect the offline data, the context distribution encountered during evaluation can differ substantially from the training distribution. To address this issue, CSRO minimizes the mutual information between the task representation $\mathbf{z}$ and the behavior-dependent components $(\mathbf{s}, \mathbf{a})$. Specifically, it employs the CLUB [31] upper bound on mutual information $\mathcal{I}_{\text{club}} = \mathbb{E}_i[\log p(\mathbf{z}^i \mid \mathbf{s}^i, \mathbf{a}^i)] - \mathbb{E}_j[\log p(\mathbf{z}^j \mid \mathbf{s}^i, \mathbf{a}^i)]$. Since $p(\mathbf{z}^i \mid \mathbf{s}^i, \mathbf{a}^i)$ is unavailable, CSRO approximates it using a variational distribution estimator trained via maximum likelihood:

$$\mathcal{L}_{\text{VD}}(\psi) = -\mathbb{E}[\log q_\psi(\mathbf{z}^i \mid \mathbf{s}^i, \mathbf{a}^i)],$$

where ψ denotes the parameters of the variational estimator. To minimize the mutual information and, subsequently, the context distribution shift, the context encoder is trained to minimize

$$\mathcal{L}_{\min \text{MI}}(\phi) = \mathbb{E}_i[\log q_\psi(\mathbf{z}^i \mid \mathbf{s}^i, \mathbf{a}^i)] - \mathbb{E}_j[\log q_\psi(\mathbf{z}^j \mid \mathbf{s}^i, \mathbf{a}^i)].$$

CSRO trains the context encoder with $\mathcal{L}(\phi) = \mathcal{L}_{\text{FOCAL}} + \mathcal{L}_{\min \text{MI}}$ forming an adversarial objective with variational distribution estimator q_ψ. Although CSRO improves generalization to unseen tasks compared to FOCAL, it still does not explicitly model task-dependent dynamics or rewards.

UNICORN [7] studies context-based OMRL from an information-theoretic perspective, showing that reconstruction (predicting reward and next state) maximizes the $\mathcal{I}(Z; X_t \mid X_b)$ and contrastive learning in FOCAL maximizes $\mathcal{I}(Z; X)$, where X_t and X_b denote the task-dependent and behavior-dependent components of the context, respectively. By maximizing $\mathcal{I}(Z; X_t \mid X_b)$, the reconstruction objective encourages the task representation $\mathbf{z}$ to capture task-specific dynamics and rewards, thereby reducing context distribution shift:

$$\mathcal{L}_{\text{Reconstruction}}(\phi) = \|\mathbf{s}_{t+1} - D_\phi(\mathbf{s}_t, \mathbf{a}_t, \mathbf{z})\|_2^2 + \lambda\|r_t - R_\phi(\mathbf{s}_t, \mathbf{a}_t, \mathbf{z})\|_2^2, \tag{9}$$

where D_ϕ and R_ϕ denote the dynamics and reward models jointly trained with the context encoder. *UNICORN-SUP* trains the context encoder using only $\mathcal{L}_{\text{Reconstruction}}$, whereas *UNICORN-SS* augments the reconstruction objective with contrastive learning: $\mathcal{L}_{\text{Reconstruction}} + \lambda_{\text{FOCAL}}\mathcal{L}_{\text{FOCAL}}$.

AutoRegression: We extend Eq. (9) to perform multi-step prediction using backpropagation through time:

$$\mathcal{L}_{\text{AutoRegression}}(\phi) = \sum_{h=0}^{H-1} \|\mathbf{s}_{t+h+1} - D_\phi(\hat{\mathbf{s}}_{t+h}, \mathbf{a}_{t+h}, \mathbf{z})\|_2^2 + \lambda\|r_t - R_\phi(\hat{\mathbf{s}}_{t+h}, \mathbf{a}_{t+h}, \mathbf{z})\|_2^2,$$

where $\hat{\mathbf{s}}_t = \mathbf{s}_t$ and $\hat{\mathbf{s}}_{t+h+1} = D_\phi(\hat{\mathbf{s}}_{t+h}, \mathbf{a}_{t+h}, \mathbf{z})$. Comparing this baseline with single-step reconstruction isolates the effect of multi-step prediction on task representation learning. Comparing it with our jointly trained world model and context encoder further highlights the importance of latent temporal consistency in task representation learning.

B Theoretical Analysis

Our actor-critic operates in the learned latent space $(\mathbf{x}_t, \mathbf{z}_t)$ without reconstructing observations. We give conditions under which planning/control in $(\mathbf{x}, \mathbf{z})$ is sufficient and derive value-error bounds. Our bounds build on the *simulation lemma* (*e.g.*, [26]), which relates value differences between two MDPs to discrepancies in reward and transition kernels. We extend it in two steps: *(i)* from the original task MDP to the latent MDP induced by the observation encoder Φ, and *(ii)* from conditioning on the true task to conditioning on the learned context representation $\mathbf{z}$.

B.1 Preliminaries and notation

For each task i, let $\mathcal{M}_i = \langle \mathcal{S}, \mathcal{A}, R_i, P_i, \gamma, \rho_0 \rangle$ with bounded rewards $|R_i(\mathbf{s}, \mathbf{a})| \leq R_{\max}$ for all $\mathbf{s}$ and $\mathbf{a}$. We use the infinite-horizon discounted value

$$V_i^\pi(\mathbf{s}) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t R_i(\mathbf{s}_t, \mathbf{a}_t) \mid \mathbf{s}_0 = \mathbf{s},\ \mathbf{a}_t \sim \pi(\cdot \mid \mathbf{s}_t) \right]. \tag{10}$$

Latent state and task representation. Our latent state is

$$\mathbf{x} = \Phi(\mathbf{s}) \coloneqq F_\phi(\mathbf{s}) \in \mathcal{X}, \tag{11}$$

and the context encoder maps a context set $\mathcal{H}_t$ of transitions to a task representation

$$\mathbf{z}_t = g_\theta(\mathcal{H}_t) \coloneqq \mathbb{E}_{(\mathbf{s},\mathbf{a},r,\mathbf{s}') \sim \mathcal{H}_t} \big[E_\theta(\mathbf{s}, \mathbf{a}, r, \mathbf{s}') \big]. \tag{12}$$

At test time $\mathbf{z}_t$ may change as more context is accumulated. In the bounds below, we treat $\mathbf{z}$ as fixed after a context set is collected (matching few-shot evaluation with a fixed context window).

World model (task-conditioned). Given $(\mathbf{x}, \mathbf{a}, \mathbf{z})$, the latent dynamics D_ϕ defines a distribution $\widehat{P}_\phi(\cdot \mid \mathbf{x}, \mathbf{a}, \mathbf{z})$ over $\mathcal{X}$, and the reward model defines $\widehat{R}_\phi(\mathbf{x}, \mathbf{a}, \mathbf{z})$.

Norms. For distributions p, q over a finite set, $\|p - q\|_1 = \sum_x |p(x) - q(x)|$. For bounded functions $V : \mathcal{S} \to \mathbb{R}$, $\|V\|_\infty = \sup_{\mathbf{s}} |V(\mathbf{s})|$.

Simulation lemma. We use a standard simulation-lemma style inequality in total variation (TV).

Lemma B.1 (Simulation lemma (TV form))**.** *Let $M = \langle \mathcal{S}, \mathcal{A}, R, P, \gamma \rangle$ and $\widehat{M} = \langle \mathcal{S}, \mathcal{A}, \widehat{R}, \widehat{P}, \gamma \rangle$ share the same state space $\mathcal{S}$ and action space $\mathcal{A}$, with $|R(\mathbf{s}, \mathbf{a})| \leq R_{\max}$ and $|\widehat{R}(\mathbf{s}, \mathbf{a})| \leq R_{\max}$ for all $(\mathbf{s}, \mathbf{a})$. Assume for all $(\mathbf{s}, \mathbf{a})$:*

$$|R(\mathbf{s}, \mathbf{a}) - \widehat{R}(\mathbf{s}, \mathbf{a})| \leq \epsilon_R, \qquad \|P(\cdot \mid \mathbf{s}, \mathbf{a}) - \widehat{P}(\cdot \mid \mathbf{s}, \mathbf{a})\|_1 \leq \epsilon_P. \tag{13}$$

Then for any stationary policy $\pi(\mathbf{a} \mid \mathbf{s})$,

$$\|V_M^\pi - V_{\widehat{M}}^\pi\|_\infty \leq \frac{\epsilon_R}{1 - \gamma} + \frac{\gamma R_{\max}}{(1 - \gamma)^2}\, \epsilon_P. \tag{14}$$

Proof. See [26]. □

B.2 Step 1: From the original MDP to the latent MDP induced by Φ

Fix a task i and suppress i in notation when clear.

Pushforward (latent) transition. Given P_i on $\mathcal{S}$ and $\Phi : \mathcal{S} \to \mathcal{X}$, we define the pushforward transition for any $(\mathbf{s}, \mathbf{a})$ as

$$P_i^\Phi(\mathbf{x}' \mid \mathbf{s}, \mathbf{a}) \coloneqq \sum_{\mathbf{s}' \in \mathcal{S}} P_i(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})\, \mathbb{1}\{\Phi(\mathbf{s}') = \mathbf{x}'\}. \tag{15}$$

Latent Markov approximation. Because Φ may be many-to-one, the induced process on $\mathbf{x}$ need not be Markov. As such, we define a Markov approximation using conditional expectation within preimages of Φ, given by

$$\bar{P}_i^{\Phi}(\mathbf{x}' \mid \mathbf{x}, \mathbf{a}) \coloneqq \mathbb{E}\left[P_i^{\Phi}(\mathbf{x}' \mid \mathbf{s}, \mathbf{a}) \mid \Phi(\mathbf{s}) = \mathbf{x}\right], \tag{16}$$

$$\bar{R}_i^{\Phi}(\mathbf{x}, \mathbf{a}) \coloneqq \mathbb{E}\left[R_i(\mathbf{s}, \mathbf{a}) \mid \Phi(\mathbf{s}) = \mathbf{x}\right]. \tag{17}$$

Assumption B.2. (Approximately Markov). There exist $\epsilon_{\text{abs}}^P, \epsilon_{\text{abs}}^R \geq 0$ such that for all $(\mathbf{s}, \mathbf{a})$ with $\mathbf{x} = \Phi(\mathbf{s})$ we have

$$\|P_i^{\Phi}(\cdot \mid \mathbf{s}, \mathbf{a}) - \bar{P}_i^{\Phi}(\cdot \mid \mathbf{x}, \mathbf{a})\|_1 \leq \epsilon_{\text{abs}}^P, \qquad |R_i(\mathbf{s}, \mathbf{a}) - \bar{R}_i^{\Phi}(\mathbf{x}, \mathbf{a})| \leq \epsilon_{\text{abs}}^R. \tag{18}$$

Intuitively, states that map to the same $\mathbf{x}$ should induce similar one-step latent transitions and rewards. Essentially, ϵ_{abs}^P represents errors in the transition dynamics due to the encoder's Φ latent space breaking the Markov property. Similarly, ϵ_{abs}^R, quantifies how much reward-relevant information is lost by Φ.

Latent policy lifting. Any latent policy $\pi(\cdot \mid \mathbf{x}, \cdot)$ induces a state-space policy $\pi^{\Phi}(\mathbf{a} \mid \mathbf{s}, \cdot) = \pi(\mathbf{a} \mid \Phi(\mathbf{s}), \cdot)$.

Lemma B.3 (Original MDP $\to$ latent Markov MDP)**.** *Fix task i and latent policy $\pi(\mathbf{a} \mid \mathbf{x}, i)$. Under Assumption B.2, the value of π^{Φ} in $\mathcal{M}_i$ is close to the value of π in the latent Markov MDP $\bar{\mathcal{M}}_i^{\Phi} = \langle \mathcal{X}, \mathcal{A}, \bar{R}_i^{\Phi}, \bar{P}_i^{\Phi}, \gamma \rangle$:*

$$\sup_{\mathbf{s} \in \mathcal{S}} \left| V_{\mathcal{M}_i}^{\pi^{\Phi}}(\mathbf{s}) - V_{\bar{\mathcal{M}}_i^{\Phi}}^{\pi}(\Phi(\mathbf{s})) \right| \leq \frac{\epsilon_{\text{abs}}^R}{1-\gamma} + \frac{\gamma R_{\max}}{(1-\gamma)^2} \epsilon_{\text{abs}}^P. \tag{19}$$

Proof. Consider two (latent) MDPs under the same policy π: one uses the (state-dependent) push-forward kernel $P_i^{\Phi}(\cdot \mid \mathbf{s}, \mathbf{a})$ and reward $R_i(\mathbf{s}, \mathbf{a})$, and the other uses $\bar{P}_i^{\Phi}(\cdot \mid \mathbf{x}, \mathbf{a})$ and $\bar{R}_i^{\Phi}(\mathbf{x}, \mathbf{a})$. Assumption B.2, bounds the one-step reward and transition discrepancies uniformly for any $\mathbf{s}$ with latent state $\mathbf{x}$. Applying Lemma B.1 with $\epsilon_R = \epsilon_{\text{abs}}^R$ and $\epsilon_P = \epsilon_{\text{abs}}^P$ yields (19). □

B.3 Step 2: From the latent MDP to the learned world model (true task ID)

Assumption B.4. (Latent model error with true task). There exist $\epsilon_{\text{model}}^P, \epsilon_{\text{model}}^R \geq 0$ such that for all $(\mathbf{x}, \mathbf{a})$ we have

$$\|\widehat{P}_{\phi}(\cdot \mid \mathbf{x}, \mathbf{a}, i) - \bar{P}_i^{\Phi}(\cdot \mid \mathbf{x}, \mathbf{a})\|_1 \leq \epsilon_{\text{model}}^P, \qquad |\widehat{R}_{\phi}(\mathbf{x}, \mathbf{a}, i) - \bar{R}_i^{\Phi}(\mathbf{x}, \mathbf{a})| \leq \epsilon_{\text{model}}^R. \tag{20}$$

Lemma B.5 (Latent Markov MDP $\to$ learned world model (true task))**.** *Fix task i and latent policy $\pi(\mathbf{a} \mid \mathbf{x}, i)$. Under Assumption B.4,*

$$\|V_{\bar{\mathcal{M}}_i^{\Phi}}^{\pi} - V_{\widehat{\mathcal{M}}_i}^{\pi}\|_{\infty} \leq \frac{\epsilon_{\text{model}}^R}{1-\gamma} + \frac{\gamma R_{\max}}{(1-\gamma)^2} \epsilon_{\text{model}}^P, \tag{21}$$

where $\widehat{\mathcal{M}}_i = \langle \mathcal{X}, \mathcal{A}, \widehat{R}_{\phi}(\cdot, \cdot, i), \widehat{P}_{\phi}(\cdot \mid \cdot, \cdot, i), \gamma \rangle$.

Proof. Apply Lemma B.1 on $\mathcal{S} = \mathcal{X}$ with $\epsilon_R = \epsilon_{\text{model}}^R$ and $\epsilon_P = \epsilon_{\text{model}}^P$. □

Theorem B.6 (Original task MDP vs learned world model (true task))**.** *Fix task i and latent policy $\pi(\mathbf{a} \mid \mathbf{x}, i)$. Under Assumptions B.2 and B.4,*

$$\sup_{\mathbf{s} \in \mathcal{S}} \left| V_{\mathcal{M}_i}^{\pi^{\Phi}}(\mathbf{s}) - V_{\widehat{\mathcal{M}}_i}^{\pi}(\Phi(\mathbf{s})) \right| \leq \frac{\epsilon_{\text{abs}}^R + \epsilon_{\text{model}}^R}{1-\gamma} + \frac{\gamma R_{\max}}{(1-\gamma)^2} (\epsilon_{\text{abs}}^P + \epsilon_{\text{model}}^P). \tag{22}$$

Proof. By the triangle inequality,

$$\left| V_{\mathcal{M}_i}^{\pi^{\Phi}}(\mathbf{s}) - V_{\widehat{\mathcal{M}}_i}^{\pi}(\Phi(\mathbf{s})) \right| \leq \left| V_{\mathcal{M}_i}^{\pi^{\Phi}}(\mathbf{s}) - V_{\bar{\mathcal{M}}_i^{\Phi}}^{\pi}(\Phi(\mathbf{s})) \right| + \left| V_{\bar{\mathcal{M}}_i^{\Phi}}^{\pi}(\Phi(\mathbf{s})) - V_{\widehat{\mathcal{M}}_i}^{\pi}(\Phi(\mathbf{s})) \right|. \tag{23}$$

Bound the first term using Lemma B.3 and the second term using Lemma B.5. □

B.4 Step 3: Adding the context encoder (unknown task at test time)

So far, the latent model is conditioned on the true task ID i. At meta-test time i is unknown and the agent instead conditions on $\mathbf{z}$ inferred via our context encoder.

Assumption B.7. (Task-inference sufficiency in latent space). There exist $\epsilon^P_{\text{task}}, \epsilon^R_{\text{task}} \geq 0$ such that for any task i and any $(\mathbf{x}, \mathbf{a})$, if $\mathbf{z} = g_\theta(\mathcal{H})$ is computed from history collected in task i, then

$$\|\widehat{P}_\phi(\cdot \mid \mathbf{x}, \mathbf{a}, \mathbf{z}) - \widehat{P}_\phi(\cdot \mid \mathbf{x}, \mathbf{a}, i)\|_1 \leq \epsilon^P_{\text{task}}, \qquad |\widehat{R}_\phi(\mathbf{x}, \mathbf{a}, \mathbf{z}) - \widehat{R}_\phi(\mathbf{x}, \mathbf{a}, i)| \leq \epsilon^R_{\text{task}}. \tag{24}$$

This is an approximate sufficiency requirement. That is, we assume that conditioning on $\mathbf{z}$ is nearly as informative as conditioning on i for predicting latent dynamics and rewards.

Lemma B.8 (Extra value error from task inference in the learned latent MDP)**.** *Fix task i and latent policy $\pi(\mathbf{a} \mid \mathbf{x}, \mathbf{z})$. Under Assumption B.7,*

$$\sup_{\mathbf{x} \in \mathcal{X}} \left| V^\pi_{\widehat{\mathcal{M}}(\mathbf{z})}(\mathbf{x}) - V^{\pi_i}_{\widehat{\mathcal{M}}_i}(\mathbf{x}) \right| \leq \frac{\epsilon^R_{\text{task}}}{1-\gamma} + \frac{\gamma R_{\max}}{(1-\gamma)^2} \epsilon^P_{\text{task}}, \tag{25}$$

where $\widehat{\mathcal{M}}(\mathbf{z})$ uses $(\widehat{P}_\phi(\cdot \mid \mathbf{x}, \mathbf{a}, \mathbf{z}), \widehat{R}_\phi(\mathbf{x}, \mathbf{a}, \mathbf{z}))$ and $\pi_i(\mathbf{a} \mid \mathbf{x}) := \pi(\mathbf{a} \mid \mathbf{x}, \mathbf{z})$ is the realized policy after inferring $\mathbf{z}$.

Proof. Apply Lemma B.1 on $\mathcal{S} = \mathcal{X}$ between $\text{MDP}_1 = \widehat{\mathcal{M}}_i$ and $\text{MDP}_2 = \widehat{\mathcal{M}}(\mathbf{z})$, using Assumption B.7 to bound the one-step reward and transition discrepancies. □

Final combined bound (unknown task, learned context). Combining Theorem B.6 and Lemma B.8 gives, for any latent policy π,

$$\sup_{\mathbf{s} \in \mathcal{S}} \left| V^{\pi^\Phi}_{\mathcal{M}_i}(\mathbf{s}) - V^\pi_{\widehat{\mathcal{M}}(\mathbf{z})}(\Phi(\mathbf{s})) \right| \leq \frac{\epsilon^R_{\text{abs}} + \epsilon^R_{\text{model}} + \epsilon^R_{\text{task}}}{1-\gamma} + \frac{\gamma R_{\max}}{(1-\gamma)^2}(\epsilon^P_{\text{abs}} + \epsilon^P_{\text{model}} + \epsilon^P_{\text{task}}). \tag{26}$$

Discussion. The bounds depend on:

1. how Markov the latent state space $\mathbf{x} = \Phi(\mathbf{s})$ is for predicting future latent states and rewards (Assumption B.2),
2. the learned latent model error (Assumption B.4),
3. the quality/sufficiency of task inference via $\mathbf{z}$ (Assumption B.7).

Importantly, none of these require reconstructing $\mathbf{s}$ from $\mathbf{x}$. Instead, they require that $(\mathbf{x}, \mathbf{z})$ preserves predictive information for control.

C Further Results

Sec. C.1 provides additional comparisons with baselines, evaluating few-shot and zero-shot adaptation in both in-distribution and out-of-distribution task settings. Sec. C.2 provides more detail on estimating different error sources and reports them for additional environments. Sec. C.3 shows task representation metrics for additional environments. Sec. C.4 reports detailed disentanglement metrics across methods, highlighting the effects of temporal consistency and contrastive learning objectives on task representation learning. Sec. C.5 reports online planing with the trained world model. Sec. C.6 compares our contextual world model with DreamerV3 [41], a state-of-the-art model-based RL method based on recurrent state-space models (RSSMs), to assess generalization to unseen tasks from history alone. It also compares temporal consistency for task representation to MetaDT [42], a state-of-the-art transformer-based OMRL method.

Ablation studies are presented in Sec. C.7 (different architectures for the context encoder), Sec. C.8 (latent space formulations), Sec. C.9 (contrastive learning and task representation $\mathbf{z}$),Sec. C.10 (Training horizon), Sec. C.11 (model size and parameter count), Sec. C.12 (choice of offline RL algorithm), and Sec. C.13 (bounding strategies for $\mathbf{z}$).

Finally, Sec. C.14 analyzes the computational cost of jointly training the context encoder and world model relative to prior work, demonstrating that the contextual world model is computationally efficient while achieving improved performance.

C.1 Generalization to New Tasks

Table 3 summarizes the results across all environments in the Meta-World ML1 benchmark. We collected datasets for these environments at varying difficulty levels using the same RL algorithm (DroQ) with identical hyperparameters. In certain environments, such as Assembly, the datasets lack successful trajectories. Consequently, various OMRL methods fail to learn the corresponding tasks. In the majority of environments, temporal consistency for joint training of the context encoder and world model outperforms the baselines, achieving higher success rates during few-shot adaptation.

Table 5 summarize zero-shot adaptation performance for in-distribution tasks and Table 6 summarize zero-shot adaptation performance for out-of-distribution tasks. For environments where some variation factors in the out-of-distribution tasks interpolate those seen during training (*e.g.*, [Cheetah, Finger, Walker]-speed), the performance on out-of-distribution tasks is relatively close to in-distribution performance. In contrast, when out-of-distribution generalization requires extrapolating beyond the training variation factors (*e.g.*, [Cheetah, Finger, Walker]-LS, Ant-Dir), a larger performance gap between in-distribution and out-of-distribution tasks is observed. Temporal consistency performs more consistently compared to baselines when generalizing to out-of-distribution tasks, where it significantly outperforms the baselines (two-sided t-test, $p < 0.05$) for both in-distribution and out-of-distribution zero-shot adaptation, respectively, indicating improved generalization to OOD tasks. These results indicate that latent temporal consistency can improve performance on in-distribution tasks by converting the observation space to a latent space and can increase generalization to out-of-distribution tasks by encouraging the context encoder to capture latent dynamics.

Table 7 reports generalization to new environments in ML10 and ML45 settings. Temporal consistency improves generalization on both training and test environments, although generalization to entirely new environments remains challenging for all methods. Increasing the diversity of training environments improves test performance.

Table 3: Few-shot in-distribution performance on all environments in Meta-world benchmarks. Average success rate over 6 random seeds, $\pm$ represents 95% confidence intervals. Since we use DroQ with the same hyperparameters to collect datasets across all environments, some environments may lack successful trajectories. **Bold** indicates the highest value with statistical significance according to the t-test with p-value < 0.05.

Env	FOCAL	CSRO	UNICORN-SUP	UNICORN-SS	AR + InfoNCE	TC + InfoNCE
Assembly	0.0 ± 0.0	1.7 ± 3.3	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Basketball	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	$\mathbf{5.0} \pm 4.4$
Bin-picking	1.7 ± 3.3	6.7 ± 4.1	1.7 ± 3.3	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Box-close	5.0 ± 4.4	16.7 ± 10.9	8.3 ± 6.0	3.3 ± 6.5	5.0 ± 6.7	21.7 ± 6.0
Button-press	93.3 ± 6.5	96.7 ± 4.1	91.7 ± 10.6	93.3 ± 9.7	86.7 ± 10.9	100.0 ± 0.0
Button-press-topdown	78.3 ± 19.9	78.3 ± 7.9	75.0 ± 19.4	90.0 ± 7.2	80.0 ± 13.4	$\mathbf{100.0} \pm 0.0$
Button-press-topdown-wall	21.7 ± 6.0	30.0 ± 10.1	43.3 ± 13.1	40.0 ± 8.8	33.3 ± 21.3	$\mathbf{93.3} \pm 6.5$
Button-press-wall	91.7 ± 6.0	85.0 ± 8.4	80.0 ± 16.0	88.3 ± 3.3	85.0 ± 4.4	98.3 ± 3.3
Coffee-button	98.3 ± 3.3	98.3 ± 3.3	98.3 ± 3.3	98.3 ± 3.3	98.3 ± 3.3	100.0 ± 0.0
Coffee-pull	2.5 ± 4.9	0.0 ± 0.0	1.7 ± 3.3	0.0 ± 0.0	0.0 ± 0.0	5.0 ± 4.4
Coffee-push	15.0 ± 12.1	16.7 ± 8.3	11.7 ± 7.9	20.0 ± 18.2	11.7 ± 6.0	30.0 ± 8.8
Dial-turn	90.0 ± 0.0	93.3 ± 6.5	75.0 ± 9.8	88.3 ± 11.8	81.7 ± 6.0	91.7 ± 6.0
Disassemble	5.0 ± 4.4	16.7 ± 6.5	18.0 ± 7.3	13.3 ± 6.5	28.3 ± 6.0	25.0 ± 6.7
Door-close	100.0 ± 0.0	100.0 ± 0.0	96.7 ± 6.5	95.0 ± 9.8	93.3 ± 9.7	100.0 ± 0.0
Door-lock	90.0 ± 5.1	78.3 ± 19.9	86.7 ± 4.1	86.7 ± 9.7	70.0 ± 13.4	88.3 ± 9.4
Door-open	96.7 ± 6.5	96.7 ± 4.1	60.0 ± 18.9	93.3 ± 4.1	78.3 ± 12.8	100.0 ± 0.0
Door-unlock	91.7 ± 6.0	95.0 ± 4.4	85.0 ± 4.4	90.0 ± 8.8	91.7 ± 6.0	100.0 ± 0.0
Drawer-close	98.3 ± 3.3	100.0 ± 0.0	98.3 ± 3.3	98.3 ± 3.3	96.7 ± 4.1	100.0 ± 0.0
Drawer-open	35.0 ± 8.4	28.3 ± 11.8	55.0 ± 21.9	26.7 ± 12.0	36.7 ± 13.1	56.7 ± 8.3
Faucet-close	81.7 ± 9.4	90.0 ± 7.2	71.7 ± 11.8	96.7 ± 4.1	71.7 ± 18.5	98.3 ± 3.3
Faucet-open	81.7 ± 13.8	88.3 ± 10.6	76.7 ± 23.6	91.7 ± 7.9	88.3 ± 7.9	86.7 ± 9.7
Hammer	35.0 ± 12.1	31.7 ± 11.8	38.3 ± 14.7	36.7 ± 14.9	38.3 ± 11.8	25.0 ± 11.0
Hand-insert	18.3 ± 9.4	23.3 ± 12.0	20.0 ± 8.8	11.7 ± 6.0	21.7 ± 6.0	26.7 ± 9.7
Handle-press	91.7 ± 6.0	93.3 ± 6.5	93.3 ± 8.3	93.3 ± 4.1	91.7 ± 6.0	96.7 ± 4.1
Handle-press-side	96.7 ± 4.1	83.3 ± 6.5	95.0 ± 4.4	96.7 ± 6.5	90.0 ± 7.2	100.0 ± 0.0
Handle-pull	48.3 ± 12.8	40.0 ± 10.1	51.7 ± 14.7	43.3 ± 12.0	48.3 ± 14.7	$\mathbf{73.3} \pm 4.1$
Handle-pull-side	71.7 ± 13.8	73.3 ± 15.7	55.0 ± 11.0	73.3 ± 17.3	45.0 ± 13.1	80.0 ± 7.2
Lever-pull	15.0 ± 11.0	30.0 ± 7.2	21.7 ± 9.4	21.7 ± 10.6	25.0 ± 11.0	30.0 ± 8.8
Peg-insert-side	23.3 ± 14.9	15.0 ± 4.4	35.0 ± 12.1	23.3 ± 13.1	23.3 ± 12.0	31.7 ± 11.8
Peg-unplug-side	66.7 ± 9.7	80.0 ± 15.2	68.3 ± 7.9	65.0 ± 6.7	56.7 ± 16.5	73.3 ± 10.9
Pick-out-of-hole	23.3 ± 6.5	30.0 ± 16.0	1.7 ± 3.3	20.0 ± 11.3	13.3 ± 8.3	6.7 ± 6.5
Pick-place	0.0 ± 0.0	0.0 ± 0.0	1.7 ± 2.2	0.0 ± 0.0	0.0 ± 0.0	1.7 ± 2.2
Pick-place-wall	0.0 ± 0.0	3.3 ± 4.1	0.0 ± 0.0	1.7 ± 3.3	0.0 ± 0.0	0.0 ± 0.0
Plate-slide	58.3 ± 13.8	50.0 ± 7.2	50.0 ± 11.3	58.3 ± 13.8	55.0 ± 18.8	56.7 ± 6.5
Plate-slide-back	10.0 ± 7.2	10.0 ± 7.2	10.0 ± 8.8	26.7 ± 23.6	26.7 ± 10.9	36.7 ± 24.1
Plate-slide-back-side	74.0 ± 14.7	70.0 ± 29.9	70.0 ± 10.1	55.0 ± 18.8	75.0 ± 19.4	58.3 ± 13.8
Plate-slide-side	55.0 ± 18.8	61.7 ± 7.9	70.0 ± 14.3	58.3 ± 12.8	56.7 ± 4.1	67.5 ± 14.7
Push	11.7 ± 6.0	15.0 ± 8.4	8.3 ± 7.9	10.0 ± 5.1	21.7 ± 10.6	$\mathbf{35.0} \pm 8.4$
Push-back	16.7 ± 4.4	18.3 ± 8.6	16.7 ± 5.6	16.7 ± 7.4	15.0 ± 4.4	13.3 ± 2.8
Push-wall	26.7 ± 9.7	30.0 ± 18.9	40.0 ± 14.3	26.7 ± 12.0	8.3 ± 9.4	$\mathbf{70.0} \pm 11.3$
Reach	6.7 ± 6.5	1.7 ± 3.3	5.0 ± 4.4	1.7 ± 3.3	3.3 ± 4.1	6.7 ± 6.5
Reach-wall	8.3 ± 6.0	10.0 ± 5.1	1.7 ± 3.3	2.0 ± 3.9	3.3 ± 4.1	10.0 ± 7.2
Shelf-place	8.3 ± 6.0	0.0 ± 0.0	6.7 ± 6.5	6.7 ± 6.5	5.0 ± 4.4	10.0 ± 7.2
Soccer	18.3 ± 7.9	11.7 ± 12.8	16.7 ± 9.7	21.7 ± 9.4	23.3 ± 10.9	$\mathbf{45.0} \pm 11.0$
Stick-pull	0.0 ± 0.0	0.0 ± 0.0	3.3 ± 6.5	0.0 ± 0.0	0.0 ± 0.0	3.3 ± 4.1
Stick-push	1.7 ± 3.3	8.3 ± 7.9	23.3 ± 10.9	6.7 ± 6.5	11.7 ± 9.4	0.0 ± 0.0
Sweep	63.3 ± 13.1	76.7 ± 13.1	85.0 ± 9.8	68.3 ± 11.8	63.3 ± 8.3	85.0 ± 9.8
Sweep-into	66.7 ± 14.0	70.0 ± 5.1	80.0 ± 14.3	63.3 ± 8.3	75.0 ± 12.1	83.3 ± 6.5
Window-close	98.3 ± 3.3	91.7 ± 12.8	98.3 ± 3.3	95.0 ± 6.7	93.3 ± 6.5	100.0 ± 0.0
Window-open	85.0 ± 6.7	83.3 ± 4.1	88.3 ± 7.9	78.3 ± 11.8	68.3 ± 12.8	96.7 ± 6.5

Table 4: Few-shot in-distribution performance on MuJoCo and Contextual-DMC benchmarks. **Temporal consistency for task representation learning improves in-distribution generalization**. Average returns over 6 random seeds, $\pm$ represents 95% confidence intervals. **Bold** indicates the highest value with statistical significance according to the t-test with p-value < 0.05.

Env	FOCAL	CSRO	UNICORN-SUP	UNICORN-SS	AR + InfoNCE	TC + InfoNCE
Ant-dir	804.0 ± 35.0	798.0 ± 39.3	429.0 ± 30.6	812.9 ± 24.5	653.5 ± 52.5	$\mathbf{863.1} \pm 36.2$
Cheetah-LS	852.2 ± 26.4	831.2 ± 60.0	832.0 ± 52.9	795.9 ± 39.4	930.3 ± 9.4	$\mathbf{944.8} \pm 4.9$
Cheetah-speed	515.7 ± 62.7	576.3 ± 78.2	586.4 ± 40.7	554.3 ± 71.6	591.3 ± 109.1	$\mathbf{764.1} \pm 39.2$
Finger-LS	880.8 ± 39.2	869.2 ± 46.6	753.2 ± 56.3	885.6 ± 14.1	913.0 ± 15.2	$\mathbf{968.0} \pm 5.5$
Finger-speed	609.5 ± 25.0	631.6 ± 56.1	526.9 ± 36.8	515.9 ± 25.5	528.3 ± 129.9	$\mathbf{967.4} \pm 2.0$
Hopper-mass	572.7 ± 13.0	476.4 ± 68.7	442.5 ± 119.2	540.9 ± 36.5	551.2 ± 12.4	$\mathbf{590.6} \pm 3.5$
Walker-LS	875.0 ± 51.5	899.2 ± 41.8	914.2 ± 23.0	880.7 ± 54.7	917.2 ± 19.1	934.6 ± 20.1
Walker-friction	532.3 ± 46.7	521.8 ± 34.4	539.1 ± 13.7	485.5 ± 57.8	526.1 ± 47.8	563.6 ± 33.5
Walker-speed	768.9 ± 30.1	771.2 ± 20.0	518.6 ± 55.3	730.7 ± 48.6	452.1 ± 55.9	$\mathbf{835.7} \pm 37.3$

Table 5: Zero-shot in-distribution performance on MuJoCo and Contextual-DMC benchmarks. **Temporal consistency for task representation learning improves in-distribution generalization**. Average returns over 6 random seeds, $\pm$ represents 95% confidence intervals. **Bold** indicates the highest value with statistical significance according to the t-test with p-value < 0.05.

Env	FOCAL	CSRO	UNICORN-SUP	UNICORN-SS	AR + InfoNCE	TC + InfoNCE
Ant-dir	678.4 ± 40.3	699.3 ± 26.6	366.6 ± 41.6	668.1 ± 46.4	502.5 ± 86.5	$\mathbf{755.7} \pm 43.1$
Cheetah-LS	825.3 ± 36.6	828.3 ± 34.9	841.9 ± 50.9	794.5 ± 44.5	860.8 ± 25.3	$\mathbf{935.0} \pm 11.9$
Cheetah-speed	447.3 ± 73.0	556.5 ± 31.3	447.9 ± 49.1	490.2 ± 82.1	532.5 ± 148.2	$\mathbf{706.4} \pm 33.1$
Finger-LS	863.0 ± 58.3	897.8 ± 41.8	824.0 ± 30.6	885.3 ± 49.6	931.3 ± 8.7	$\mathbf{972.0} \pm 5.0$
Finger-speed	746.7 ± 49.3	773.7 ± 48.5	614.4 ± 45.0	671.9 ± 54.6	638.7 ± 112.5	$\mathbf{943.3} \pm 8.4$
Hopper-mass	535.0 ± 33.3	450.8 ± 79.2	491.6 ± 145.6	533.7 ± 38.3	536.2 ± 16.1	566.0 ± 13.5
Walker-LS	898.9 ± 30.3	882.5 ± 100.3	889.9 ± 36.9	900.0 ± 49.2	871.9 ± 32.6	937.2 ± 9.9
Walker-friction	522.0 ± 32.8	503.7 ± 39.5	476.7 ± 32.4	522.1 ± 21.6	510.1 ± 81.6	578.2 ± 13.6
Walker-speed	653.1 ± 99.4	767.1 ± 31.7	513.1 ± 48.7	598.6 ± 54.4	506.5 ± 43.3	785.7 ± 50.5

Table 6: Zero-shot out-of-distribution performance on MuJoCo and Contextual-DMC benchmarks. **Temporal consistency for task representation learning improves out-of-distribution generalization**. Average returns over 6 random seeds, $\pm$ represents 95% confidence intervals. **Bold** indicates the highest value with statistical significance according to the t-test with p-value < 0.05.

Env	FOCAL	CSRO	UNICORN-SUP	UNICORN-SS	AR + InfoNCE	TC + InfoNCE
Ant-dir	368.8 ± 64.2	399.2 ± 63.9	-211.4 ± 195.5	405.9 ± 38.5	284.8 ± 103.4	410.7 ± 36.9
Cheetah-LS	826.6 ± 20.6	813.9 ± 28.1	795.8 ± 44.3	806.1 ± 28.6	849.3 ± 23.6	865.5 ± 20.4
Cheetah-speed	607.8 ± 160.4	603.5 ± 96.5	554.8 ± 75.9	598.8 ± 103.4	622.8 ± 155.7	756.0 ± 92.5
Finger-LS	786.8 ± 32.6	762.7 ± 57.9	691.5 ± 54.8	816.1 ± 47.7	846.1 ± 18.4	$\mathbf{886.7} \pm 11.8$
Finger-speed	771.3 ± 39.0	822.8 ± 32.3	675.3 ± 66.0	709.6 ± 43.6	598.3 ± 87.6	$\mathbf{948.1} \pm 9.3$
Hopper-mass	547.3 ± 13.3	543.6 ± 32.1	463.9 ± 133.6	550.7 ± 11.3	423.1 ± 22.6	583.4 ± 4.1
Walker-LS	658.0 ± 41.0	611.5 ± 31.6	649.9 ± 50.7	657.6 ± 41.7	705.7 ± 36.2	$\mathbf{788.0} \pm 27.8$
Walker-friction	473.7 ± 36.5	475.1 ± 22.5	435.2 ± 53.8	484.6 ± 25.0	475.6 ± 33.2	474.1 ± 30.3
Walker-speed	659.6 ± 120.1	767.2 ± 24.0	535.5 ± 44.2	623.7 ± 98.0	576.9 ± 44.0	772.5 ± 56.5

Table 7: **Generalization to new environments:** Temporal consistency for task representation learning demonstrates better generalization to unseen environments. Increasing the number of training environments can improve generalization to testing environments. Average success rate over 6 random seeds, $\pm$ represents 95% confidence intervals. **Bold** indicates the highest value with statistical significance according to the t-test with p-value < 0.05.

Env	CSRO	FOCAL	UNICORN-SUP	UNICORN-SS	AR + InfoNCE	TC + InfoNCE
ML10-Train	43.3 ± 7.6	43.3 ± 6.5	25.4 ± 3.7	43.3 ± 3.0	44.4 ± 4.2	$\mathbf{83.5} \pm 6.1$
ML10-Test	3.3 ± 4.1	5.0 ± 5.1	2.5 ± 3.3	4.2 ± 4.7	5.5 ± 4.9	$\mathbf{15.0} \pm 3.9$
ML45-Train	32.8 ± 2.4	32.2 ± 2.9	30.7 ± 1.7	35.9 ± 2.4	42.4 ± 2.1	$\mathbf{58.7} \pm 4.0$
ML45-Ood	8.3 ± 6.0	3.3 ± 4.1	18.3 ± 1.2	18.3 ± 2.6	5.0 ± 4.4	$\mathbf{24.0} \pm 6.7$

C.2 Error Decomposition

To approximately quantify each error source, we train separate networks using privileged information (the true task ID i, which is unavailable in the offline meta-RL setting). Note that while the theoretical analysis in Sec. 3.3 defines each error term as $\sup \|\cdot\|_1$, we approximate these quantities empirically using the ℓ_2-norm $\|\cdot\|_2$.

Abstraction error We train a network $P_{\text{abs}}(\mathbf{x}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t, i)$ to predict the next latent vector from the current state, action, and task ID. The abstraction error is estimated as the discrepancy between this prediction and the encoded next state, where $\mathbf{x}_{t+1} := \Phi(\mathbf{s}_{t+1})$ is the next latent state induced by the observation encoder and $\bar{P}_i^{\phi}$ is the true latent dynamics:

$$\hat{\epsilon}^P_{\text{abs}} = \left\| \underbrace{P_{\text{abs}}(\cdot \mid \mathbf{s}_t, \mathbf{a}_t, i)}_{\approx P_i^{\Phi}(\cdot|\mathbf{s}_t, \mathbf{a}_t)} - \underbrace{\Phi(\mathbf{s}_{t+1})}_{\approx \bar{P}_i^{\phi}(\cdot|\Phi(\mathbf{s}_t), \mathbf{a}_t)} \right\|_2 .$$

World modeling error We train a second network $P_{\text{model}}(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{a}_t, i)$ to predict the next latent vector from the current latent vector, action, and task ID. The world modeling error is estimated as the discrepancy between the predicted next latent state and the induced next latent state by the observation encoder $\Phi(\mathbf{s}_{t+1})$:

$$\hat{\epsilon}^P_{\text{model}} = \left\| \underbrace{P_{\text{model}}(\cdot \mid \mathbf{x}_t, \mathbf{a}_t, i)}_{= \hat{P}_{\phi}(\cdot|\mathbf{x}_t, \mathbf{a}_t, i)} - \underbrace{\Phi(\mathbf{s}_{t+1})}_{\approx \bar{P}_i^{\phi}(\cdot|\Phi(\mathbf{s}_t), \mathbf{a}_t)} \right\|_2 .$$

Task error Finally, we compare the predictions of the learned world model with those of our contextual world model, which conditions on the learned task representation $\mathbf{z}$ instead of the true task ID i:

$$\hat{\epsilon}^P_{\text{task}} = \left\| \underbrace{D_{\phi}(\cdot \mid \mathbf{x}_t, \mathbf{a}_t, \mathbf{z})}_{= \hat{P}_{\phi}(\cdot|\mathbf{x}_t, \mathbf{a}_t, \mathbf{z})} - \underbrace{P_{\text{model}}(\cdot \mid \mathbf{x}_t, \mathbf{x}_t, i)}_{= \hat{P}_{\phi}(\cdot|\mathbf{x}_t, \mathbf{a}_t, i)} \right\|_2 .$$

All networks are trained simultaneously on the same transitions, and all error terms are evaluated on the same transitions. Figure 6 illustrates the evolution of these errors during training. We observe the following:

- Jointly training the context encoder and the world model generally reduces all three error terms over the course of training.
- The estimated task error is generally larger than both the estimated abstraction and modeling errors, highlighting the importance of task representation learning.
- The sum of the decomposed errors upper-bounds the temporal consistency loss

$$\hat{\epsilon}^P_{\text{abs}} + \hat{\epsilon}^P_{\text{model}} + \hat{\epsilon}^P_{\text{task}} \;\geq\; \|\mathbf{x}_{t+1} - D_{\phi}(\cdot \mid \mathbf{x}_t, \mathbf{a}_t, \mathbf{z})\|_2 .$$

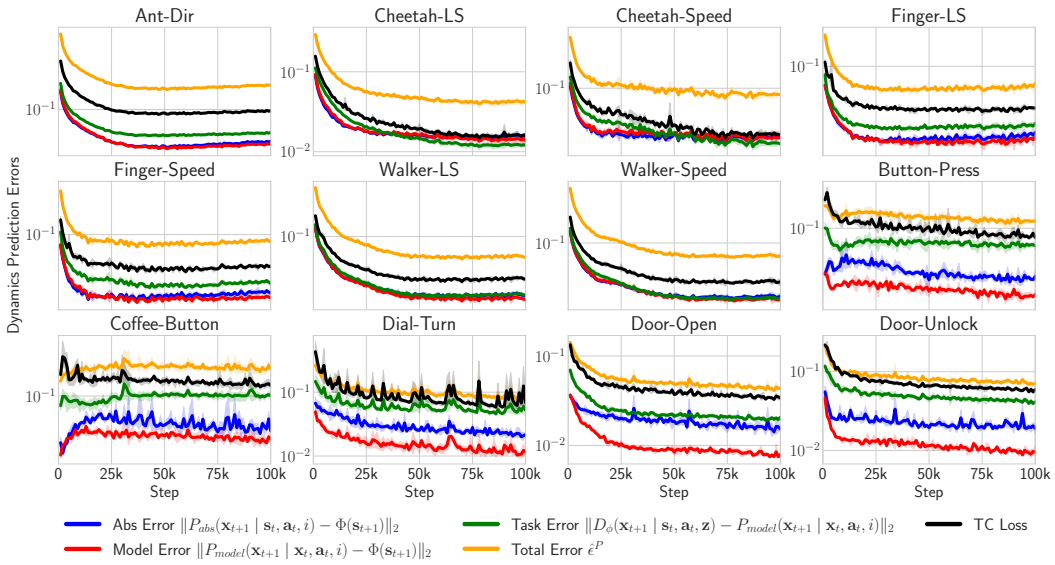


Figure 6: Estimating different error sources. **Joint training of the context encoder and world model reduces all error terms, with task error dominating; their sum upper-bounds the temporal consistency loss.** Abstraction error arises from mapping states to a latent space, model error reflects world model accuracy, and task error measures how well the context encoder identifies the task. Separate networks (P_{abs} and P_{model}) with privileged information (task ID i) were trained for quantifying these error terms. Shaded regions denote 95% confidence intervals over three random seeds.

C.3 Task Representation Metrics

Figure 7 illustrates representation metrics (feature rank, rank, and dormant ratio) along with performance on 10 MetaWorld environments.

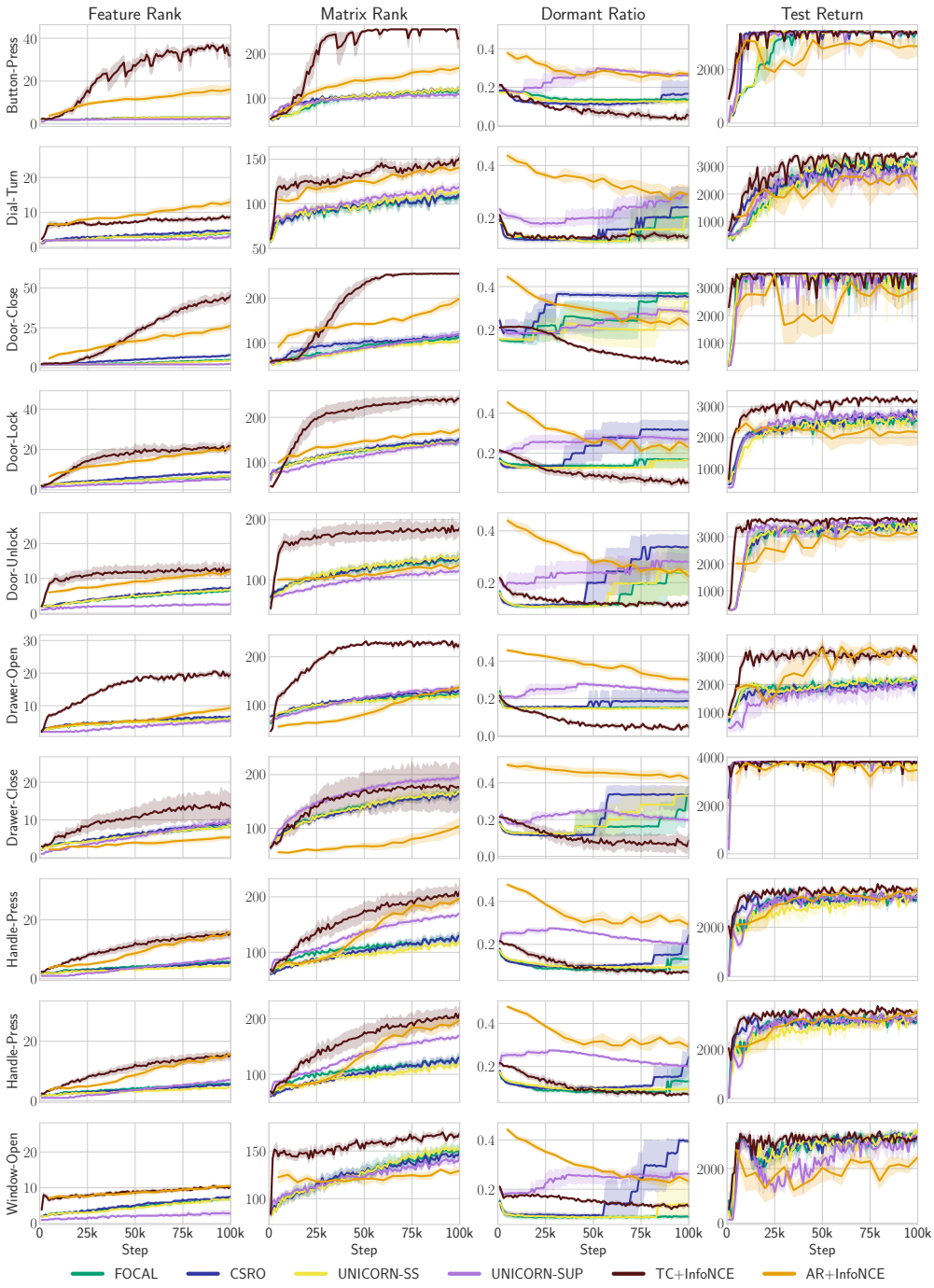


Figure 7: Representation metrics (feature rank, rank, and dormant ratio) of the context encoder across 10 environments from the Meta-World benchmark. **Temporal consistency maintains a low dormant neuron ratio and a high matrix rank while learning more diverse features (higher feature rank).** The shaded area represents the 95% confidence interval across six random seeds.

C.4 Disentanglement Metrics

Previous work often relies on t-SNE visualizations to assess task separation, but such visualizations are qualitative and sensitive to hyperparameters. We instead use disentanglement metrics, which measure how well latent dimensions align with underlying task variation factors. We report DCI [33] metrics (disentanglement, completeness, informativeness) and InfoMEC [34] metrics (modularity, explicitness, compactness). Together, these metrics capture whether a task representation is structured, informative, and aligned with true task factors.

Table 8, Table 9 and Table 10 illustrate disentanglement metrics for Cheetah-LS Finger-LS and Walker-LS environments, respectively. We observe similar trends as Table 8, where training the context encoder based on temporal consistency (TC) results in higher disentanglement than training the context encoder with reconstruction (UNICORN-SUP), and including contrastive learning can improve task distinguishability (informativeness, explicitness). However, while the reconstruction objective (UNICORN-SUP) results in higher disentanglement than contrastive objectives (FOCAL, InfoNCE) in Cheetah-LS (Table 8), in [Finger, Walker]-LS reconstruction objective results in a lower disentanglement score.

Table 8: Disentanglement metrics (DCI, InfoMEC) for the Cheetah-length-speed (LS) environment. **Latent temporal consistency disentangles the variation factors more effectively**, while **contrastive learning enhances task distinguishability**, reflected in informativeness and explicitness. *TC* denotes training the context encoder solely with temporal consistency (Eq. (4)); *FOCAL* represents contrastive learning, *UNICORN-SUP* indicates training with reconstruction (decoder), and *AR* indicates auto-regression, training the context encoder with backpropagation through time in state space. Average metrics over 6 random seeds, $\pm$ represents 95% confidence intervals.

	Disentanglement	Completeness	Informativeness	Modularity	Explicitness	Compactness
FOCAL	0.33 ± 0.11	0.25 ± 0.08	0.82 ± 0.04	0.77 ± 0.04	0.75 ± 0.05	0.02 ± 0.01
CSRO	0.31 ± 0.05	0.32 ± 0.05	0.83 ± 0.04	0.75 ± 0.03	0.78 ± 0.03	0.02 ± 0.00
UNICORN-SS	0.32 ± 0.06	0.29 ± 0.09	0.83 ± 0.04	0.76 ± 0.01	0.76 ± 0.04	0.03 ± 0.01
UNICORN-SUP	0.36 ± 0.09	0.23 ± 0.06	0.54 ± 0.06	0.76 ± 0.05	0.71 ± 0.02	0.18 ± 0.07
AR + InfoNCE	0.41 ± 0.04	0.39 ± 0.05	0.78 ± 0.01	0.74 ± 0.02	0.81 ± 0.01	0.15 ± 0.03
TC	0.45 ± 0.06	0.49 ± 0.07	0.87 ± 0.03	0.70 ± 0.02	0.86 ± 0.02	0.23 ± 0.02
TC + InfoNCE	0.50 ± 0.05	0.46 ± 0.05	0.89 ± 0.02	0.74 ± 0.03	0.87 ± 0.01	0.24 ± 0.04

Table 9: Disentanglement metrics (DCI, InfoMEC) for the Finger Length/Speed environment. Average metrics over 6 random seeds, $\pm$ represents 95% confidence intervals.

	Disentanglement	Completeness	Informativeness	Modularity	Explicitness	Compactness
FOCAL	0.36 ± 0.06	0.30 ± 0.07	0.70 ± 0.01	0.83 ± 0.02	0.80 ± 0.01	0.04 ± 0.01
CSRO	0.36 ± 0.03	0.38 ± 0.08	0.71 ± 0.02	0.82 ± 0.02	0.79 ± 0.01	0.04 ± 0.01
UNICORN-SS	0.33 ± 0.05	0.31 ± 0.08	0.69 ± 0.02	0.79 ± 0.02	0.78 ± 0.01	0.05 ± 0.01
UNICORN-SUP	0.25 ± 0.05	0.34 ± 0.06	0.57 ± 0.01	0.85 ± 0.02	0.73 ± 0.01	0.10 ± 0.04
AR + InfoNCE	0.23 ± 0.05	0.40 ± 0.05	0.63 ± 0.02	0.87 ± 0.02	0.74 ± 0.01	0.08 ± 0.01
TC	0.41 ± 0.03	0.43 ± 0.02	0.70 ± 0.01	0.82 ± 0.02	0.85 ± 0.01	0.13 ± 0.01
TC + InfoNCE	0.46 ± 0.06	0.43 ± 0.04	0.82 ± 0.03	0.82 ± 0.02	0.87 ± 0.02	0.13 ± 0.01

Table 10: Disentanglement metrics (DCI, InfoMEC) for the Walker Length/Speed environment. Average metrics over 6 random seeds, $\pm$ represents 95% confidence intervals.

	Disentanglement	Completeness	Informativeness	Modularity	Explicitness	Compactness
FOCAL	0.33 ± 0.07	0.31 ± 0.06	0.83 ± 0.04	0.69 ± 0.03	0.82 ± 0.02	0.03 ± 0.01
CSRO	0.42 ± 0.08	0.41 ± 0.09	0.79 ± 0.03	0.72 ± 0.03	0.82 ± 0.02	0.03 ± 0.01
UNICORN-SS	0.38 ± 0.07	0.31 ± 0.05	0.84 ± 0.03	0.73 ± 0.03	0.84 ± 0.03	0.04 ± 0.01
UNICORN-SUP	0.20 ± 0.03	0.18 ± 0.07	0.38 ± 0.04	0.80 ± 0.04	0.64 ± 0.02	0.06 ± 0.02
AR + InfoNCE	0.30 ± 0.04	0.21 ± 0.05	0.46 ± 0.04	0.82 ± 0.06	0.67 ± 0.02	0.10 ± 0.04
TC	0.39 ± 0.05	0.27 ± 0.06	0.75 ± 0.04	0.77 ± 0.05	0.86 ± 0.03	0.14 ± 0.02
TC + InfoNCE	0.40 ± 0.09	0.30 ± 0.09	0.84 ± 0.04	0.79 ± 0.04	0.88 ± 0.02	0.14 ± 0.01

C.5 Planning with the World Model

We have shown that jointly training the context encoder and world model improves the performance of offline policy optimization. In the previous experiments, however, we used the world model solely for representation learning. In this section, we investigate whether the trained world model, conditioned on the context encoder, can also be used for online planning.

A key challenge in planning with a learned Q-function in the offline setting is value overestimation for out-of-distribution (OOD) actions. Since the actions selected by the planner may follow a different distribution from those observed in the offline dataset, the learned Q-function can assign erroneously high values to such actions. To mitigate this issue, we use conservative Q-learning (CQL [50]) to train the Q-function, encouraging more conservative value estimates for OOD actions during planning.

Figure 8 illustrates the zero-shot adaptation performance under different planning horizons using online planning. Similar to TD-MPC2 [17] and DC-MPC [18], we use model-predictive path-integral (MPPI) control for planning while updating the task representation $\mathbf{z}$ at each step. We consider two planning variants: planning based solely on the predicted rewards and planning that additionally uses the Q-function to estimate the value of the final state at the end of the planning horizon.

The Q-function can complement short-horizon planning by providing an estimate of the long-term return beyond the planning horizon. In reward-only planning, candidate trajectories are evaluated based only on the rewards predicted within the finite planning horizon, which may favor actions that yield high immediate rewards but lead to unfavorable states in the longer term. In contrast, incorporating the Q-value of the final predicted state provides a bootstrap estimate of the future return beyond the planning horizon. This allows the planner to account for long-term consequences without explicitly extending the planning horizon, potentially improving both decision quality and computational efficiency.

Overall, increasing the planning horizon improves zero-shot adaptation performance, at the cost of increased computational complexity. Longer horizons allow the planner to account for future outcomes over a greater number of steps in the learned latent space. Furthermore, incorporating conservative Q-value estimates into the planner generally improves performance, particularly for shorter planning horizons, where the Q-function provides a more informative estimate of the long-term consequences that cannot be captured within the limited planning horizon.

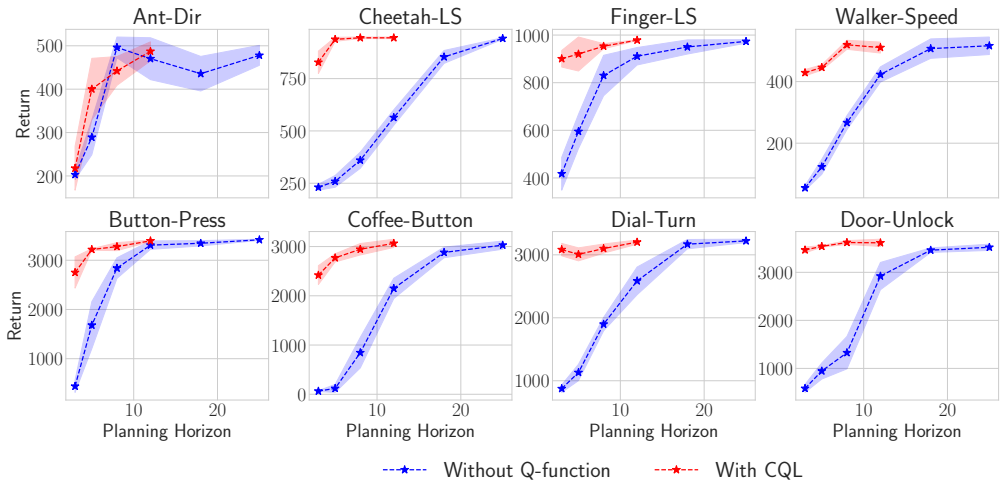


Figure 8: Planning with the trained world model and context encoder for zero-shot adaptation. **Increasing the planning horizon improves performance, while incorporating conservative Q-estimates further improves performance, particularly at shorter planning horizons.** Results are averaged over 6 random seeds; shaded areas represent the standard deviation.

C.6 Comparison to DreamerV3 and MetaDT

DreamerV3 [41] is a model-based RL method that employs a Recurrent State-Space Model (RSSM [51]) for latent dynamics while jointly predicting rewards, observations, and terminations. Its latent space includes both continuous and discrete variables. The policy and value function are optimized within the world model. We use a public PyTorch implementation[3] with default hyperparameters.

MetaDT [42] employs a transformer-based architecture to directly predict actions, without explicitly modeling environment dynamics or the reward function. It extends Prompt-DT [52] by conditioning on a sequence of task representations, where the task encoder is an RNN trained via reconstruction.

Table 11 reports the zero-shot performance on in-distribution tasks. For DreamerV3, we reset the RSSM hidden state at the initial timestep during meta-testing. DreamerV3 struggles to generalize to new tasks in OMRL settings, particularly in environments where optimal policies differ significantly across tasks. For example, in [Cheetah, Finger, Walker]-speed environments, the agent must move both forward and backward at different speeds, and in Ant-dir environments, the agent must move in different directions. On the other hand, DreamerV3 shows better generalization in environments where optimal task-specific policies are more similar, such as Hopper-mass and Walker-friction. We also hypothesize that the policy may exploit inaccuracies in the world model, since the world model is trained solely on static datasets. The policy is optimized to maximize expected return under the world model's predictions, without any penalty for acting in uncertain or poorly modeled regions. In online RL settings, the policy's actions would be executed in the real environment, and the world model would be updated accordingly; however, this corrective mechanism is absent in the offline OMRL scenario.

Overall, jointly training context encoder and world model based on temporal consistency and contrastive learning outperforms Meta-DT in 5/9 environments, matches performance in 3/7, and is outperformed only in Walker-Speed.

Table 11: **DreamerV3 fails to generalize in OMRL settings** and **our contextual world model outperforms MetaDT. Zero-shot generalization to in-distribution tasks**. Average returns/success rates over 3 random seeds, $\pm$ represents 95% confidence intervals.

Environment	TC + InfoNCE (ours)	DreamerV3	MetaDT
Ant-dir	$\mathbf{649.9 \pm 50.7}$	-3.6 ± 3.0	412.3 ± 87.4
Cheetah-LS	$\mathbf{936.5 \pm 10.8}$	584.8 ± 53.2	533.8 ± 113.8
Cheetah-speed	$\mathbf{664.4 \pm 51.7}$	178.7 ± 20.8	439.2 ± 55.8
Finger-LS	966.4 ± 5.5	438.6 ± 85.3	967.3 ± 14.2
Finger-speed	946.9 ± 9.6	187.0 ± 72.7	942.1 ± 4.2
Hopper-mass	$\mathbf{579.9 \pm 9.5}$	555.1 ± 3.6	566.1 ± 12.3
Walker-friction	$\mathbf{580.6 \pm 4.7}$	523.6 ± 41.1	540.2 ± 21.3
Walker-LS	939.7 ± 8.3	643.6 ± 46.0	929.3 ± 11.0
Walker-speed	705.8 ± 70.9	149.9 ± 10.3	$\mathbf{797.7 \pm 57.1}$

[3] https://github.com/NM512/dreamerv3-torch

C.7 Ablation: Architecture of Context Encoder

Similar to previous OMRL methods, we conduct our experiments on deterministic MDPs where the reward function and transition dynamics differ across tasks. On these benchmarks, a common assumption is task-transition correspondence, where :

$$\forall s, a \quad P_i(. \mid s, a) = P_j(. \mid s, a), R_i(. \mid s, a) = R_j(. \mid s, a) \iff \mathcal{M}_i = \mathcal{M}_j \tag{27}$$

Eq. (27) indicates that the context encoder can identify the task from a single transition. Based on this assumption, the context encoder identifies the underlying task by taking the expectation over the sequence of task representations according to Eq. (3).

Table 12 compares the performance of different architectures in the context encoder. *Attention* applies self-attention without positional encoding to compute the task representation via weighted averaging, operating on the same non-temporal context set as *MLP*. By contrast, *(*RNN) and *(*Transformer) process sequential context batches (sequence length 5) during training, and leverage the full available sequence at inference time.

We have noticed that training an RNN-based context encoder with a temporal consistency objective of the world model can hinder performance. Using attention to perform weighted averaging did not provide a noticeable benefit, yet it reduced the performance in 2 environments. Using a transformer-based context encoder improved generalization in *Finger-Speed*, we speculate that this is because when the planar robot is not in contact with the rotating body, its actions do not affect the angular velocity and thus do not influence the reward, making individual transitions less informative for task inference.

Table 12: Ablation study on the context encoder architecture for zero-shot in-distribution generalization. While RNN-based encoders underperform, **transformer-based encoders perform comparably to MLPs and only show clear gains in Finger-Speed, suggesting that averaging representations from single transitions is sufficient in most environments.** Results are averaged over six seeds; $\pm$ denotes 95% confidence intervals, and **bold** indicates statistically significant best results ($p < 0.05$, t-test).

Env	MLP	Attention	GRU	Transformer
Ant-Dir	649.9 ± 50.7	453.5 ± 106.1	230.9 ± 5.5	593.1 ± 60.2
Cheetah-LS	936.5 ± 10.8	930.5 ± 11.2	755.4 ± 0.0	880.5 ± 11.2
Cheetah-Speed	664.4 ± 51.7	669.5 ± 72.6	427.8 ± 25.0	677.5 ± 43.2
Finger-LS	966.4 ± 5.5	967.0 ± 10.3	857.6 ± 81.0	970.0 ± 13.1
Finger-Speed	946.9 ± 9.6	693.6 ± 146.7	568.3 ± 147.8	$\mathbf{970.6 \pm 3.1}$
Walker-LS	939.7 ± 8.3	941.5 ± 9.5	826.2 ± 24.7	937.5 ± 8.4
Walker-Speed	705.8 ± 70.9	698.7 ± 49.2	621.7 ± 28.7	715.4 ± 53.4
Button-Press	100.0 ± 0.0	96.7 ± 4.1	35.0 ± 29.4	100.0 ± 0.0
Coffee-Button	100.0 ± 0.0	100.0 ± 0.0	88.3 ± 6.0	100.0 ± 0.0
Dial-Turn	$\mathbf{100.0 \pm 0.0}$	83.3 ± 10.9	88.7 ± 5.7	90.0 ± 7.1
Door-Close	98.3 ± 3.3	98.3 ± 3.3	100.0 ± 0.0	98.3 ± 3.3
Door-Lock	91.7 ± 10.6	100.0 ± 0.0	90.0 ± 0.0	100.0 ± 0.0
Door-Open	98.3 ± 3.3	100.0 ± 0.0	95.0 ± 7.0	100.0 ± 0.0
Door-Unlock	100.0 ± 0.0	98.3 ± 3.3	100.0 ± 0.0	100.0 ± 0.0
Faucet-Close	98.3 ± 3.3	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
Handle-Press	93.3 ± 9.7	98.3 ± 3.3	50.0 ± 14.1	100.0 ± 0.0

C.8 Ablation: Latent Space Formulation

We isolate the effect of different latent space formulations (classification vs. regression, continuous vs. discrete, bounded vs. unbounded) on the performance. Prior work on RL uses continuous latent spaces with regression losses [16], cosine similarity [13], or bounded and discretized representations [23, 17, 18].

Figure 9 shows that simply bounding or discretizing the latent space does not improve performance in our setting. In contrast, formulating temporal consistency as a classification problem leads to better results. This suggests that the cross-entropy loss, rather than discretization itself, is the main driver of performance gains as reported by [18]. Similarly, [53] demonstrates the advantages of classification compared to regression in training value functions. We provide a detailed comparison in Table 13.

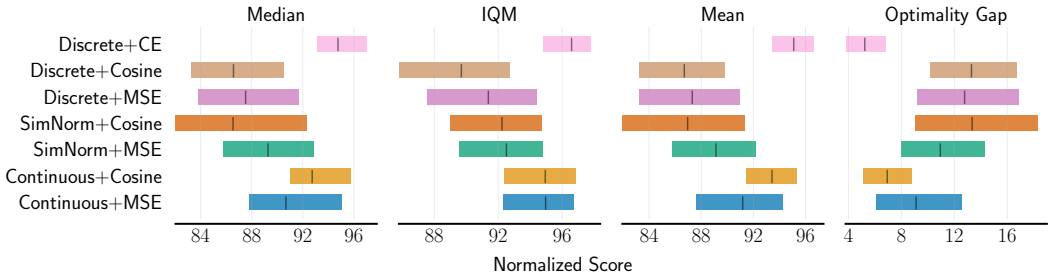


Figure 9: Different world model formulations: IQM and optimality gap of the normalized return for different world modeling methods, evaluated on 9 environments with 6 random seeds per environment. **The main advantage of discretizing the latent space is due to classification** loss (cross-entropy). Bounding or discretizing the latent space alone does not improve performance.

Table 13: Ablation on latent space, different choice on latent space, and temporal consistency loss function. Average returns over 6 random seeds, $\pm$ represents 95% confidence intervals.

Environment	Cont MSE	Cont Cosine	SimNorm MSE	SimNorm Cosine	Dis MSE	Dis Cosine	Dis CE
Ant-dir	841.6 ± 25.9	865.6 ± 41.4	851.5 ± 30.9	854.5 ± 34.0	784.0 ± 54.0	800.7 ± 65.9	863.1 ± 36.2
Cheetah-LS	938.0 ± 9.2	895.8 ± 49.8	898.3 ± 25.1	905.6 ± 25.0	920.0 ± 15.2	897.0 ± 33.2	944.8 ± 4.9
Cheetah-speed	645.3 ± 127.1	735.6 ± 27.6	572.2 ± 49.6	453.9 ± 96.0	647.2 ± 63.3	648.5 ± 49.7	764.1 ± 39.2
Finger-LS	972.2 ± 6.1	971.6 ± 3.6	971.6 ± 4.8	973.7 ± 2.3	974.7 ± 2.2	976.8 ± 2.5	968.0 ± 5.5
Finger-speed	949.4 ± 2.9	932.3 ± 18.9	915.0 ± 63.1	947.3 ± 25.3	929.8 ± 39.7	968.0 ± 1.9	967.4 ± 2.0
Hopper-mass	578.8 ± 17.8	557.0 ± 31.8	563.6 ± 11.7	513.2 ± 71.7	406.0 ± 116.0	429.5 ± 61.0	590.6 ± 3.5
Walker-friction	559.9 ± 20.6	573.7 ± 13.4	502.1 ± 35.5	521.8 ± 49.4	549.4 ± 28.0	451.5 ± 40.0	563.6 ± 33.5
Walker-LS	927.8 ± 23.2	930.1 ± 56.5	942.0 ± 9.1	931.9 ± 31.3	944.9 ± 4.6	938.6 ± 11.6	934.6 ± 20.1
Walker-speed	813.0 ± 30.9	847.6 ± 23.4	843.1 ± 20.6	824.1 ± 31.9	814.6 ± 28.7	831.3 ± 22.2	835.7 ± 37.3
Button-press	100.0 ± 0.0	98.3 ± 3.3	100.0 ± 0.0	96.7 ± 6.5	98.3 ± 3.3	98.3 ± 3.3	100.0 ± 0.0
Coffee-button	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
Door-close	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	98.3 ± 3.3	100.0 ± 0.0
Door-open	100.0 ± 0.0	98.3 ± 3.3	100.0 ± 0.0	96.7 ± 4.1	98.3 ± 3.3	98.3 ± 3.3	100.0 ± 0.0
Door-unlock	95.0 ± 6.7	96.7 ± 4.1	98.3 ± 3.3	96.7 ± 4.1	98.3 ± 3.3	100.0 ± 0.0	100.0 ± 0.0
Drawer-close	98.3 ± 3.3	100.0 ± 0.0	100.0 ± 0.0	98.3 ± 3.3	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
Drawer-open	45.0 ± 8.4	40.0 ± 8.8	63.3 ± 10.9	56.7 ± 9.7	48.3 ± 9.4	48.3 ± 12.8	51.7 ± 6.0
Faucet-close	91.7 ± 6.0	90.0 ± 5.1	98.3 ± 3.3	100.0 ± 0.0	100.0 ± 0.0	93.3 ± 4.1	100.0 ± 0.0
Faucet-open	71.7 ± 17.8	95.0 ± 4.4	86.7 ± 12.0	95.0 ± 4.4	90.0 ± 5.1	90.0 ± 10.1	91.7 ± 6.0
Handle-press	91.7 ± 6.0	93.3 ± 4.1	95.0 ± 4.4	96.7 ± 6.5	96.7 ± 4.1	98.3 ± 3.3	98.3 ± 3.3
Handle-pull	61.7 ± 15.5	70.0 ± 14.3	75.0 ± 12.1	73.3 ± 17.3	70.0 ± 20.9	86.7 ± 8.3	70.0 ± 17.5
Push-wall	53.3 ± 17.3	48.3 ± 12.8	46.7 ± 18.0	45.0 ± 12.1	48.3 ± 19.2	56.7 ± 14.0	63.3 ± 6.5
Sweep	68.3 ± 11.8	78.3 ± 9.4	73.3 ± 9.7	71.7 ± 13.8	81.7 ± 9.4	93.3 ± 4.1	85.0 ± 8.4
Sweep-into	73.3 ± 12.0	73.3 ± 12.0	71.7 ± 15.5	73.3 ± 19.4	71.7 ± 19.9	76.7 ± 9.7	80.0 ± 11.3
Window-close	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	98.3 ± 3.3	100.0 ± 0.0	100.0 ± 0.0
Window-open	100.0 ± 0.0	88.3 ± 9.4	96.7 ± 6.5	96.7 ± 4.1	98.3 ± 3.3	98.3 ± 3.3	98.3 ± 3.3

C.9 Ablation: Contrastive Learning

In this section, we perform an ablation study on the contrastive and world modeling objectives used to train the context encoder. We compare two contrastive learning objectives commonly employed in OMRL: InfoNCE and FOCAL [4] (also referred to as distance metric learning). The FOCAL objective is defined as:

$$\mathrm{L}_{\text{FOCAL}}(\phi) = \mathbb{1}\{i=j\}\|z^i - z^j\|_2^2 + \mathbb{1}\{i\neq j\}\frac{\beta}{\|z^i - z^j\|_2^2 + \epsilon_0}. \tag{28}$$

The InfoNCE objective in our experiment is defined as:

$$\mathcal{L}_{\text{Contrastive}}(\theta) = -\sum_i \log \frac{S(\mathbf{z}^i, \bar{\mathbf{z}}^i)}{\sum_j S(\mathbf{z}^i, \bar{\mathbf{z}}^j)}, \tag{29}$$

where $\bar{\mathbf{z}}^i = \lambda \mathbf{z}^i + (1-\lambda)\bar{\mathbf{z}}^i$ is the moving average of task representations controlled by λ, and $S(\mathbf{z}^i, \mathbf{z}^j) = \exp(-\|\mathbf{z}^i - \mathbf{z}^j\|_2^2/\alpha)$ measures similarity. Positive samples are obtained from the moving average of the same task representation, while negatives are drawn from other tasks. The moving average stabilizes training by smoothing updates. This objective provides a lower bound on the mutual information between tasks and task representations, $I(\mathbf{z}; \mathcal{M})$ [12].

For a fair comparison, we convert the observation space to a latent space for all the methods with the same world model. Table 14 reports few-shot in-distribution testing and Table 15 reports few-shot out-of-distribution testing. Here, *TC* refers to training the context encoder with the world modeling objective (latent temporal consistency), and + indicates the combination of objectives. For environments in Meta-World benchmarks (last 6 rows in Table 14), we observe no significant difference in the performance of different objectives. Training the context encoder solely with the temporal consistency is insufficient, as it fails to distinguish between different tasks. This limitation is particularly pronounced in environments where variation factors affect only the reward function, rather than the transition dynamics (*e.g.*, Ant-dir, where the desired forward direction varies, or environments requiring the inference of desired speed). Only using contrastive learning results in good performance across most tasks, while InfoNCE outperforms FOCAL significantly in certain environments, especially on out-of-distribution testing. Adding the world modeling objective to the contrastive objective has an insignificant impact on in-distribution performance; however, it can improve generalization to out-of-distribution tasks for certain environments. We employ the same relative weighting of the contrastive objective with respect to the world modeling objective across all environments. Overall, combining InfoNCE with the world modeling objective produces more robust results across environments compared to combining FOCAL with the world modeling objective.

Table 14: Ablation on contrastive learning and world modeling, few-shot in-distribution performance. Average returns/success rates over 6 random seeds, $\pm$ represents 95% confidence intervals. **Bold** indicates the highest value with statistical significance according to the t-test with p-value < 0.05.

Environment	FOCAL	InfoNCE	TC	TC+FOCAL	TC+InfoNCE
Ant-dir	841.6 ± 31.1	857.7 ± 42.3	487.5 ± 91.8	859.0 ± 20.5	863.1 ± 36.2
Cheetah-LS	940.0 ± 16.2	937.7 ± 17.7	941.5 ± 16.6	933.2 ± 15.0	944.8 ± 4.9
Cheetah-speed	721.1 ± 54.2	727.3 ± 23.8	395.0 ± 36.2	711.2 ± 94.0	764.1 ± 39.2
Finger-LS	971.0 ± 10.5	968.4 ± 10.8	974.6 ± 5.5	973.7 ± 2.8	968.0 ± 5.5
Finger-speed	789.7 ± 189.6	958.1 ± 5.7	706.1 ± 150.3	770.0 ± 185.2	**967**$.4 \pm 2.0$
Walker-LS	929.8 ± 24.4	947.7 ± 15.1	904.5 ± 24.9	928.9 ± 21.6	934.6 ± 20.1
Walker-speed	622.6 ± 65.9	842.2 ± 35.6	522.1 ± 95.4	552.0 ± 82.4	835.7 ± 37.3
Button-press	100.0 ± 0.0	100.0 ± 0.0	98.3 ± 3.3	100.0 ± 0.0	100.0 ± 0.0
Coffee-button	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
Dial-turn	93.3 ± 4.1	93.3 ± 6.5	98.3 ± 3.3	96.7 ± 4.1	98.3 ± 3.3
Door-open	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	96.7 ± 6.5	100.0 ± 0.0
Door-unlock	98.3 ± 3.3	95.0 ± 6.7	98.3 ± 3.3	98.3 ± 3.3	100.0 ± 0.0
Handle-press	96.7 ± 4.1	96.7 ± 4.1	95.0 ± 6.7	95.0 ± 6.7	98.3 ± 3.3

Table 15: Ablation on contrastive learning and world modeling, few-shot out-of-distribution performance. Average returns over 6 random seeds, $\pm$ represents 95% confidence intervals. **Bold** indicates the highest value with statistical significance according to the t-test with p-value < 0.05. **Combining contrastive learning with latent temporal consistency enhances generalization to out-of-distribution tasks**.

Environment	FOCAL	InfoNCE	TC	TC+FOCAL	TC+InfoNCE
Ant-dir	529.0 ± 41.0	363.2 ± 173.9	203.6 ± 102.9	540.8 ± 111.2	501.8 ± 92.4
Cheetah-LS	864.4 ± 21.7	864.7 ± 29.9	867.2 ± 7.5	876.1 ± 24.7	860.6 ± 10.2
Cheetah-speed	729.0 ± 42.4	754.9 ± 70.6	486.8 ± 111.1	908.7 ± 74.2	$\mathbf{967.7} \pm 10.5$
Finger-LS	836.3 ± 45.5	809.6 ± 53.1	838.3 ± 64.8	860.0 ± 42.7	850.9 ± 41.5
Finger-speed	793.8 ± 189.0	868.5 ± 10.3	766.6 ± 176.2	755.7 ± 185.0	$\mathbf{978.5} \pm 6.0$
Walker-LS	740.2 ± 52.9	793.2 ± 62.2	757.0 ± 65.2	738.9 ± 51.8	792.3 ± 41.3
Walker-speed	619.4 ± 111.4	782.7 ± 32.2	507.7 ± 103.4	568.7 ± 128.5	$\mathbf{833.2} \pm 64.5$

C.10 Ablation: Multi-Step Temporal Consistency

In our framework, we jointly train the world model and the context encoder using a multi-step temporal consistency objective in the latent space (Eq. (4)). In this section, we provide an ablation study on the training horizon h used in this objective.

Figure 10 illustrates the impact of the training horizon h on few-shot in-distribution adaptation performance. Overall, our framework achieves competitive performance across different training horizons. Increasing the training horizon initially improves performance, but performance degrades when the horizon becomes too large. We hypothesize that excessively long training horizons can make optimization more challenging due to backpropagation through time over longer sequences, potentially leading to unstable gradients and making the joint training of the world model and context encoder more difficult.

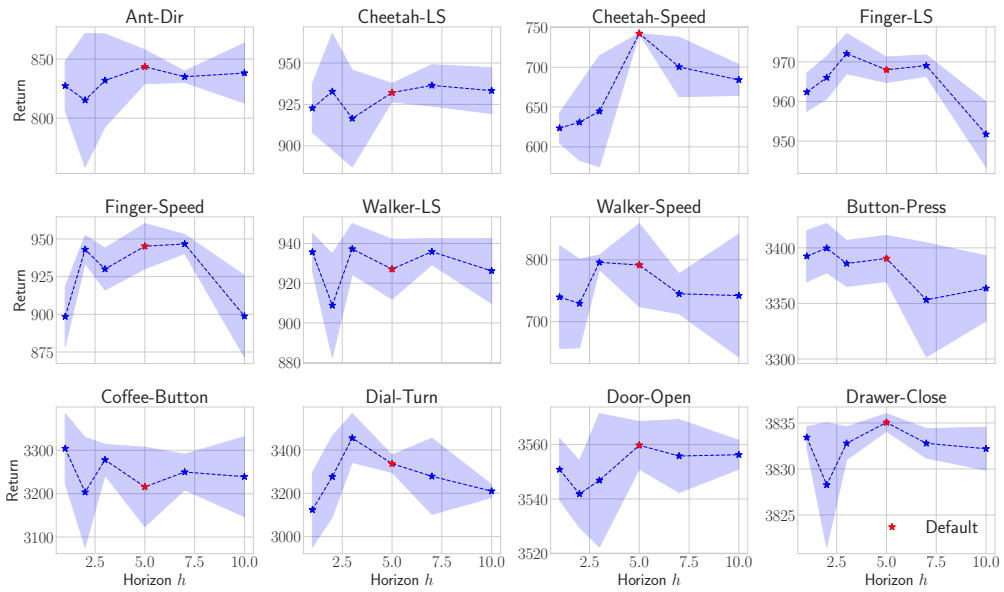


Figure 10: Impact of training horizon used in multi-step temporal consistency on few-shot performance. Results are averaged over 6 random seeds; shaded areas represent the standard deviation.

C.11 Ablation: Number of Parameters

In this section, we investigate how increasing the number of trainable parameters affects performance across different methods. In our experiments, we jointly train the context encoder and world model, where the world model maps the observation space to a latent space, which increases the total number of parameters. Figure 11 illustrates the few-shot in-distribution performance of each method for different model sizes. For the baselines, we vary the number of hidden layers in $\{2, 3\}$ and the number of hidden units in $\{256, 512, 1024\}$, resulting in six model sizes (default is two hidden layers with 256 neurons). To ensure architectural consistency, we apply Layer Normalization and the Mish activation function across all baseline networks. For our contextual world model, we set the latent dimension and the number of neurons in the dynamic head of the world model to $\{64, 128, 256, 512\}$ while setting the number of neurons in the offline RL (IQL) networks to $\{64, 128, 256, 512\}$ (6 combinations, default is 512 latent dimensions and neurons in the dynamic head, 256 neurons for the offline RL networks). Our contextual world model exhibits better scaling with model size: performance generally improves as the number of parameters increases. However, for smaller model sizes, our contextual world model underperforms the baselines in some environments.

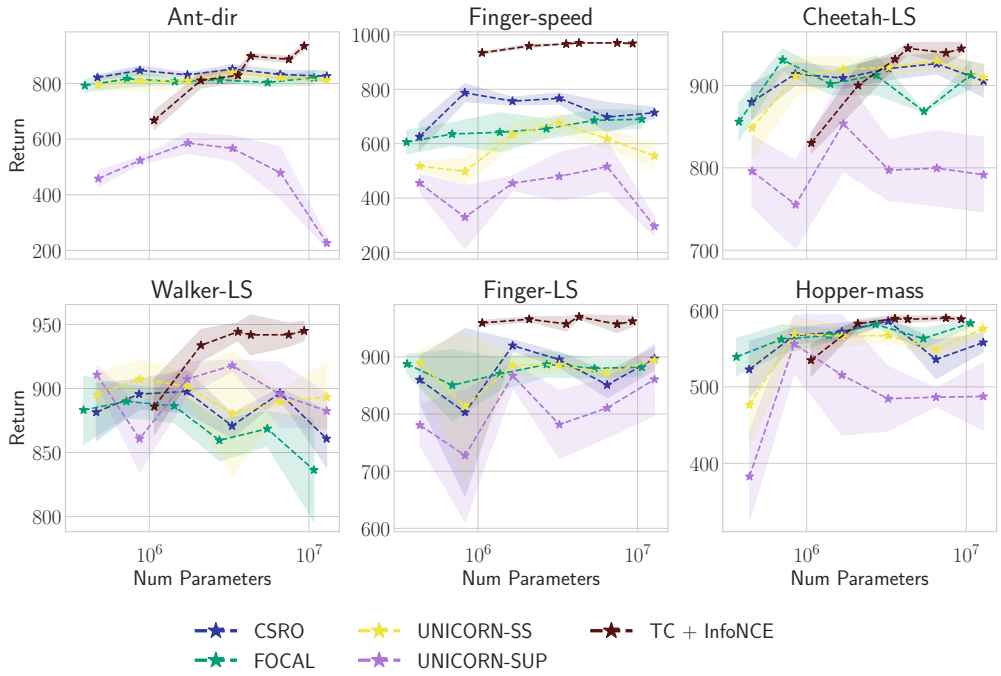


Figure 11: **Jointly training the world model and context encoder scales more effectively.** Few-shot generalization to in-distribution tasks for different model sizes. The shaded area represents the standard deviation over 6 random seeds.

C.12 Ablation: Offline RL

As described in Sec. 3.2, policy optimization with offline data requires regularization to avoid OOD action selection when computing the target for the value function. Offline RL methods address this issue in different ways, and in principle, any offline RL method can be used for policy optimization in our experiments. By default, we use Implicit Q-Learning (IQL) for all methods, which predicts an upper expectile of the TD targets in SARSA style without querying OOD actions. We also evaluate Conservative Q-Learning (CQL, [50]) and TD3+BC [54] for policy optimization, summarized in Table 16. CQL regularizes the value function by reducing the Q-value for OOD actions, resulting in a pessimistic value function. TD3+BC, on the other hand, regularizes the policy to stay close to the behavior policy by adding a behavior cloning objective to the policy optimization. We used one set of hyperparameters (default values) for all methods without further fine-tuning. We find that IQL in general is more robust, performing well in diverse environments. CQL generally performs on par with IQL, even outperforming it significantly in two environments. However, the computation cost of CQL is generally higher than that of IQL. We sometimes observe a performance drop when training for a larger number of steps. TD3+BC generally has lower performance than CQL and IQL in our settings. We hypothesize that fine-tuning the regularization weight for each environment can increase the performance.

Table 16: **Ablation: different offline RL methods for policy optimizations.** Few-shot generalization to in-distribution tasks. Average returns/success rates over 6 random seeds, $\pm$ represents 95% confidence intervals. **Bold** indicates the highest value with statistical significance according to the t-test with p-value < 0.05.

Environment	CQL	IQL (Def)	TD3+BC
Ant-dir	780.7 ± 54.4	$\mathbf{863.1} \pm 36.2$	731.8 ± 29.8
Cheetah-LS	949.8 ± 12.1	943.0 ± 11.5	846.2 ± 60.6
Cheetah-speed	$\mathbf{813.3} \pm 3.5$	751.2 ± 27.9	620.8 ± 133.0
Finger-LS	$\mathbf{987.5} \pm 3.8$	957.1 ± 23.9	330.7 ± 26.7
Finger-speed	970.2 ± 4.3	962.0 ± 9.2	208.7 ± 31.8
Hopper-mass	584.2 ± 1.1	587.5 ± 4.9	159.4 ± 26.1
Walker-friction	585.6 ± 32.6	563.6 ± 33.5	535.6 ± 26.6
Walker-LS	948.0 ± 7.2	937.1 ± 16.6	710.2 ± 84.9
Walker-speed	814.0 ± 42.4	827.6 ± 34.6	655.7 ± 108.2
Button-press	95.0 ± 9.8	100.0 ± 0.0	80.0 ± 5.1
Coffee-button	100.0 ± 0.0	100.0 ± 0.0	91.7 ± 7.9
Dial-turn	90.0 ± 19.6	91.7 ± 10.6	95.0 ± 4.4
Door-open	100.0 ± 0.0	100.0 ± 0.0	23.3 ± 4.1
Door-unlock	100.0 ± 0.0	100.0 ± 0.0	51.7 ± 10.6
Handle-press	60.0 ± 19.6	$\mathbf{93.3} \pm 4.1$	76.7 ± 6.5

C.13 Ablation: Bounding the Task Representation

In this section, we evaluate how bounding the task representation z affects the generalization. By default, we utilize Tanh as the activation function for the context encoder, consistent with prior OMRL methods. Table 17 summarize the results. We compare unbounded representations (Identity), ℓ_2-normalization, FSQ, and Tanh for bounding the latent space. We investigate whether discretizing the task representation with FSQ influences generalization. The results indicate that bounding the task representation significantly enhances generalization in certain environments (*e.g.*, Ant-dir, Cheetah-speed). Discretizing the task representation with FSQ yields no advantages over Tanh, and ℓ_2-normalization is less robust across different environments.

Table 17: **Bounding the task representation enables better generalization in certain environments**, though discretizing with FSQ yields no advantages. Average returns and success rates across 6 random seeds; $\pm$ represents 95% confidence intervals.

Environment	Identity	ℓ_2-Norm	FSQ	Tanh
Ant-dir	452.7 ± 121.3	838.0 ± 72.9	866.0 ± 46.2	863.1 ± 36.2
Cheetah-LS	933.8 ± 10.5	932.3 ± 16.7	938.3 ± 5.8	944.8 ± 4.9
Cheetah-speed	670.3 ± 75.1	778.0 ± 27.8	730.7 ± 30.2	764.1 ± 39.2
Finger-LS	962.2 ± 3.8	956.6 ± 11.4	969.6 ± 5.4	968.0 ± 5.5
Finger-speed	715.4 ± 133.6	855.1 ± 83.8	950.4 ± 30.4	967.4 ± 2.0
Walker-LS	922.7 ± 20.0	924.7 ± 15.7	932.4 ± 42.9	934.6 ± 20.1
Button-press	98.3 ± 3.3	100.0 ± 0.0	98.3 ± 3.3	100.0 ± 0.0
Coffee-button	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
Dial-turn	96.7 ± 4.1	95.0 ± 4.4	78.0 ± 33.6	96.7 ± 4.1
Door-open	95.0 ± 6.7	98.3 ± 3.3	100.0 ± 0.0	100.0 ± 0.0
Door-unlock	98.3 ± 3.3	96.7 ± 4.1	98.3 ± 3.3	100.0 ± 0.0

C.14 Computation Cost

Figure 12 compares the computation cost for different methods. All experiments are conducted using the same hardware, as described in Sec. A, to ensure a fair comparison. Although our contextual world model has a longer training time per step, it generally converges faster than the baselines, compensating for the higher per-step computational cost. During testing, our contextual world model is slightly slower because it first maps the observation to the latent space using the observation encoder, after which the policy produces actions.

UNICORN-SUP trains the context encoder solely using the prediction loss and has the lowest computational cost per training step. However, incorporating contrastive learning can improve task representation learning and, consequently, generalization to new tasks. CSRO and UNICORN-SS aim to reduce context distribution shift by minimizing a CLUB upper bound of mutual information and by adding a prediction loss, respectively. These approaches require additional networks, increasing their computational cost per training step. During test time, all baselines have the same computational cost since the policy and context encoder architectures are identical across the baseline methods.

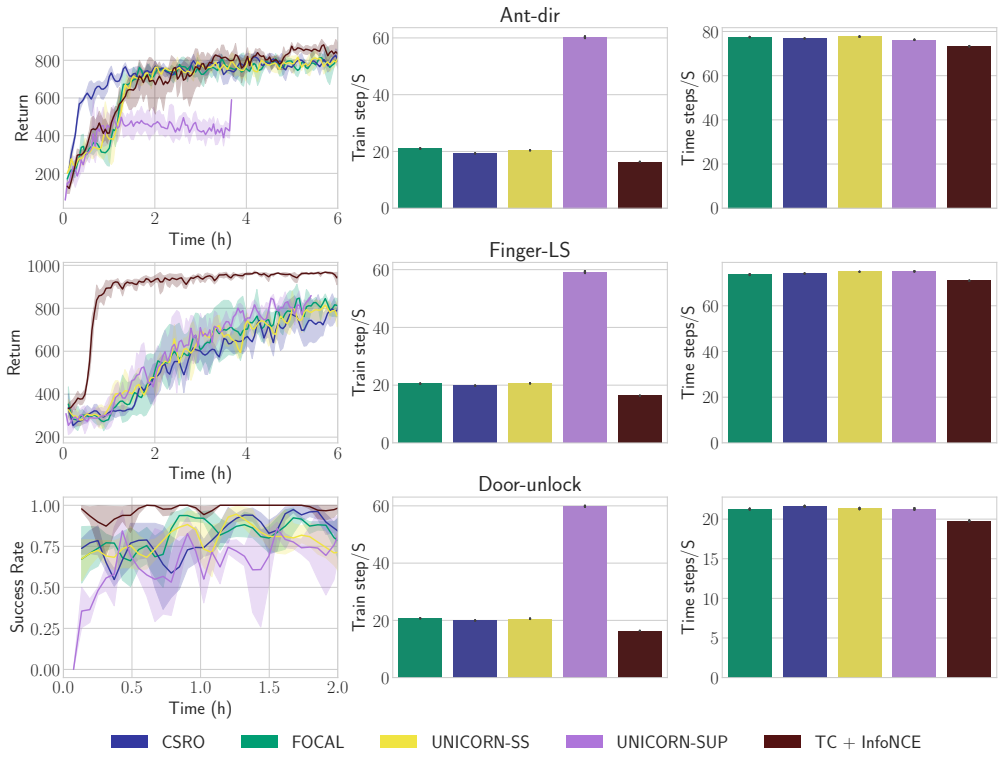


Figure 12: Comparing the computation cost. **Left:** Few-shot performance on in-distribution tasks with training time. **Middle:** Number of training steps (full backpropagation and updating the networks) per second. **Right:** Number of time steps per second during testing. **Our contextual world model is computationally more expensive during training, yet it converges faster.** Mapping the observation space to the latent space adds insignificant computational overhead during testing. Results are averaged over 3 random seeds while considering the standard deviation.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The statements in the abstract and introduction describe the contributions of this paper, where we investigate temporal consistency in offline MetaRL.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discussed the limitations of our study in the conclusion section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The assumptions, lemmas, and proofs for our theoretical analysis are provided in detail in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We describe the details of the experimental setup in the appendix and release the code in the supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We release the code in the supplementary material with anonymous links to the dataset.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We describe the details of experiments in the appendix and provide the code in the supplementary materials.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We conduct our experiments over multiple random seeds, and we report the standard error. We also use the t-test for statistical significance.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We describe the compute resources in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: Our work is foundational research. We do not believe that our work has a particular societal impact in addition to the standard ones of fundamental reinforcement learning research.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This work poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use the official code for the baselines, and we report the source directly when using third parties.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The provided code (in the supplementary materials) is documented.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: We used LLMs for proofreading and checking typos, and visualization. We did not use LLMs as an important, original, or non-standard component of the core methods

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.