Offline vs. Online Learning in Model-based RL: Lessons for Data Collection Strategies

Jiaqi Chen , Ji Shi , Cansu Sancaktar , Jonas Frey , Georg Martius

Keywords: Model-based RL, Online Learning, Offline Learning, Active Learning, Exploration

Summary

Data collection is crucial for learning robust world models in model-based reinforcement learning. The most prevalent strategies are to actively collect trajectories by interacting with the environment during online training or training on offline datasets. At first glance, the nature of learning task-agnostic environment dynamics makes world models a good candidate for effective offline training. However, the effects of online vs. offline data on world models and thus on the resulting task performance have not been thoroughly studied in the literature. In this work, we investigate both paradigms in model-based settings, conducting experiments on 31 different environments. First, we showcase that online agents outperform their offline counterparts. We identify a key challenge behind performance degradation of offline agents: encountering Out-Of-Distribution (OOD) states at test time. This issue arises because, without the self-correction mechanism in online agents, offline datasets with limited state space coverage induce a mismatch between the agent's imagination and real rollouts, compromising policy training. We demonstrate that this issue can be mitigated by allowing for additional online interactions in a fixed or adaptive schedule, restoring the performance of online training with limited interaction data. We also showcase that incorporating exploration data helps mitigate the performance degradation of offline agents. Based on our insights, we recommend adding exploration data when collecting large datasets, as current efforts predominantly focus on expert data alone.

Contribution(s)

1. We provide an in-depth analysis of performance degradation in offline model-based agents with practical considerations. We highlight the coupling of model and policy learning as a primary contributing factor beyond the pure OOD challenge.
   Context: In model-free RL, the performance degradation is often linked to limited coverage of offline datasets, which leads to inaccurate value estimates and poor extrapolation of the learned policy (Ostrovski et al., 2021; Yue et al., 2023; 2022). Similar issues plague offline model-based RL (He, 2023; Kidambi et al., 2020; Chen et al., 2023; Yu et al., 2020; Cang et al., 2021). However, an alternative perspective remains overlooked: the influence of data quality and online interaction ratios on the robustness and generalization of world models.
2. We demonstrate that incorporating exploration data with a mixed reward improves the state-space coverage in offline training. This provides insights in how to create the offline dataset such that the performance degradation can be mitigated and competitive task performance can be maintained.
   Context: Existing methods primarily focus on constraining the agent within in-distribution regions for the task (Kidambi et al., 2020; Yu et al., 2020; 2021; Wang et al., 2024; Matsushima et al., 2021) but do not explicitly assess which data collection strategies best support offline training.
3. We propose using the world model loss as a metric to measure the novelty of regions explored by the current policy. It serves as an indicator for when minimal online interactions can help offline agents to efficiently improve performance.
   Context: The approach of self-generated data is mostly investigated in the context of model-free RL (Ostrovski et al., 2021; Lee et al., 2021).

Offline vs. Online Learning in Model-based RL: Lessons for Data Collection Strategies

Jiaqi Chen [1,3], **Ji Shi** [1,2], **Cansu Sancaktar** [1,2], **Jonas Frey** [2,3], **Georg Martius** [1,2]
chenjiaq@ethz.ch, {ji.shi, cansu.sancaktar}@uni-tuebingen.de, jonfrey@ethz.ch, georg.martius@uni-tuebingen.de

[1]**Autonomous Learning Group, Computer Science, University of Tübingen, Germany**
[2]**Max Planck Institute for Intelligent Systems, Tübingen, Germany**
[3]**Robotic Systems Lab, ETH Zurich, Switzerland**

Abstract

Data collection is crucial for learning robust world models in model-based reinforcement learning. The most prevalent strategies are to actively collect trajectories by interacting with the environment during online training or training on offline datasets. At first glance, the nature of learning task-agnostic environment dynamics makes world models a good candidate for effective offline training. However, the effects of online vs. offline data on world models and thus on the resulting task performance have not been thoroughly studied in the literature. In this work, we investigate both paradigms in model-based settings, conducting experiments on 31 different environments. First, we showcase that online agents outperform their offline counterparts. We identify a key challenge behind performance degradation of offline agents: encountering Out-Of-Distribution states at test time. This issue arises because, without the self-correction mechanism in online agents, offline datasets with limited state space coverage induce a mismatch between the agent's imagination and real rollouts, compromising policy training. We demonstrate that this issue can be mitigated by allowing for additional online interactions in a fixed or adaptive schedule, restoring the performance of online training with limited interaction data. We also showcase that incorporating exploration data helps mitigate the performance degradation of offline agents. Based on our insights, we recommend adding exploration data when collecting large datasets, as current efforts predominantly focus on expert data alone.

1 Introduction

Online training of reinforcement learning (RL) agents enables continual adaptation through direct interaction with the environment. However, this approach is often impractical and less scalable in real-world settings due to high data collection costs, safety concerns, or hardware constraints (Kim et al., 2023; Schrittwieser et al., 2021; Huang et al., 2024). To address these limitations, offline RL methods attempt to reuse past experiences, training agents on pre-collected datasets without further environment interaction.

However, offline RL is prone to performance degradation when encountering Out-Of-Distribution (OOD) states, where poor generalization manifests as overestimation errors in value functions, leading to suboptimal action choices (Ostrovski et al., 2021; Yue et al., 2023; 2022). Model-based Reinforcement Learning (MBRL) offers a potential alternative by learning task-agnostic environment dynamics, enabling agents to train policies via model rollouts instead of direct environment interaction (Bruce et al., 2024). In principle, this should help mitigate overestimation errors in value functions and thus promote generalization.

Yet, recent studies have shown that MBRL is still vulnerable to OOD issues, particularly when world models are trained on offline data with insufficient state-space coverage (Yu et al., 2020; Kidambi et al., 2020; Wang et al., 2024), which in turn increases the risk of inaccuracies in the world model—a second source of error distinct from that in value function estimation. While prior works, such as MOPO (Yu et al., 2020), focus on mitigating distributional shift by penalizing model uncertainty during policy deployment, our study takes a fundamentally different approach: rather than addressing OOD errors post hoc, we investigate how data diversity, dataset optimality, and online interaction ratios impact the robustness (i.e. generalization ability) of world models and MBRL policies. By systematically decoupling the roles of the policy and the world model, we aim to provide a deeper understanding of the failure modes of MBRL in offline settings. Our study shifts the focus from uncertainty-penalization techniques to data-driven solutions, offering insights into how data collection strategies influence the reliability and generalization of world models.

In this work, we aim to provide an exhaustive analysis of online and offline data collection paradigms in an MBRL setting and address two key questions: (1) How can we best **leverage offline data** to train a robust world model and (2) **what combination of data collection strategies**—e.g., online and offline, task-oriented and exploration-driven—yields the best performance at the lowest cost across different scenarios? We believe this is a crucial research direction, as analyzing these phenomena from a unified perspective across a wide range of environments can provide valuable insights for future dataset collection.

We employ DreamerV3 (Hafner et al., 2023) across 31 diverse environments on well-established benchmarks including locomotion, manipulation, and numerous other robotic tasks. As shown in Fig. 1, we examine three scenarios: (1) an Active agent training tabula rasa, (2) a Tandem agent replaying the learning history of the Active agent in the same temporal order but with a different random initialization, and (3) a Passive agent with access to the Active agent's full experience from the start, also with a different random initialization. Importantly, both the Tandem and Passive agents do not collect data themselves; instead, they learn offline from data generated by the Active agent.

Our key findings reveal that in a task-oriented setting[1], Tandem and Passive agents underperform compared to the Active agent, primarily due to visiting novel states during evaluation. This OOD tendency stems from the absence of a self-correction mechanism in offline agents. Unlike online agents, which can correct overestimation bias through direct interaction with the environment, offline agents cannot gather relevant data to correct their predictions. This leads to a mismatch between the agent's imagination and real rollouts, ultimately misguiding policy training. We demonstrate that using offline exploration data instead of solely task-oriented data mitigates this problem and, surprisingly, find that expert demonstrations alone are insufficient for high performance in MBRL. However, we showcase that performance can be recovered with minimal environment interactions. Based on these results, we analyze an adaptive fine-tuning agent that can recover the Active agent's performance with just $6\,\%$ of environment interactions relative to its offline dataset. As a result of our large-scale experimental study, we suggest to everyone collecting expert demonstration data to also collect exploration data for sufficient state-space coverage.

Our contributions are as follows:

- **Analysing the process behind performance degradation** in offline model-based agents, along with several practical considerations.
- **Demonstrating the benefits of exploration data** and proposing that a mixed reward function enhances state-space coverage in data collection, preventing performance degradation in offline training while maintaining strong task performance.
- **Examining world-model loss as a metric for targeted active data collection**, thereby substantially enhancing the efficiency of offline agents with minimal additional interactions.

[1]That is, an agent trained solely with task-specific rewards.

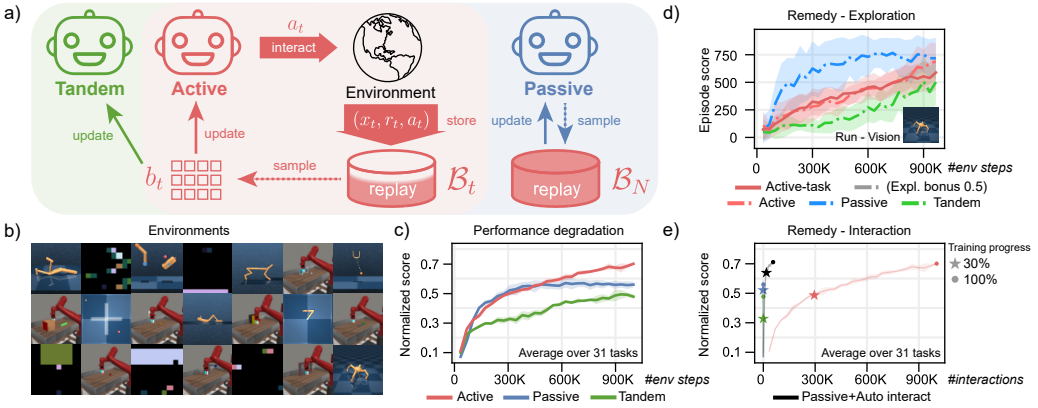

Figure 1: **Investigation of the performance degradation in offline agents and potential remedies.** a) Illustration of Active, Passive, and Tandem agents. The Active agent is trained using online RL and is allowed to interact with the environment. The Passive agent is trained from the full buffer of an Active agent, without performing any additional interactions. The Tandem agent, is also trained offline, but samples batches from the Active agent's replay buffer in the exact same sequence. b) We conduct experiments in 31 tasks across various domains. c) Illustration of the performance degradation in Passive and Tandem agents w.r.t. the Active agent. d-e) exploration data (d) and online interaction (e) effectively mitigate performance degradation observed in offline Passive agents.

2 Method

2.1 Preliminaries

Model-based Reinforcement Learning In this work, we consider environments that can be described by a partially observable Markov Decision Process (POMDP), with high-dimensional observations x_t, which are encoded into latent representations s_t, state-conditioned actions a_t generated by an agent and scalar rewards r_t (conditional on s_t and a_t) generated by the environment. In MBRL, our aim is to learn the latent transition dynamics by a **world model** $\hat{\mathcal{T}}(s_{t+1} \mid s_t, a_t)$ and find an optimal **policy** $\pi(a_t|s_t)$ maximizing the expected discounted return with discount factor γ:

$$\pi^* = \arg\max_{\pi} \mathop{\mathbb{E}}_{\substack{s_t \sim \hat{\mathcal{T}}(\cdot|s_{t-1},a_{t-1}) \\ a_t \sim \pi(a|s_t)}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]. \tag{1}$$

DreamerV3 We use DreamerV3 (Hafner et al., 2023), a state-of-the-art model-based RL method, as the base architecture in all our experiments. Based on the Recurrent State-Space Model (RSSM) (Hafner et al., 2018) summarized in Eq. (2), the world model predicts the latent state $s_t = (h_t, z_t)$ from the previous state and action, where h_t is the deterministic and z_t is the stochastic state component. The estimated observation $\hat{x}_t$, reward $\hat{r}_t$, and continuation flag $\hat{c}_t$ (signalling whether the episode has ended or not) are decoded from the latent states; given by the tuple $\hat{e}_t = (\hat{x}_t, \hat{r}_t, \hat{c}_t)$. The policy has an actor-critic architecture, detailed in Eq. (3). R_t is the discounted return from state s_t. For the off-policy updates of DreamerV3, environment interactions are added to a replay buffer $\mathcal{B} = \{(x_t, a_t, r_t, c_t, \dots)\}_{t=1}^{N}$, where each tuple contains the observation x_t, action a_t, reward r_t, continuation flag c_t, and optionally other variables collected from the environment.

$$\begin{aligned} &\textbf{Sequence model:} && h_t = f_\phi(h_{t-1}, z_{t-1}, a_{t-1}) && \textbf{Encoder:} && z_t \sim q_\phi(z_t \mid h_t, x_t) \\ &\textbf{Dynamics predictor:} && \hat{z}_t \sim p_\phi(\hat{z}_t \mid h_t) && \textbf{Decoder:} && \hat{e}_t \sim p_\phi(\hat{e}_t \mid h_t, z_t) \end{aligned} \tag{2}$$

$$\textbf{Actor:} \quad a_t \sim \pi_\theta(a_t \mid s_t) \qquad \textbf{Critic:} \quad v_\psi(s_t) \approx \mathrm{E}_{p_\phi, \pi_\theta}\left[R_t\right] \tag{3}$$

DreamerV3 minimizes the world model loss, which is a weighted loss of multiple components and is defined in the original paper (Hafner et al., 2023), as shown in Eq. (4).

$$\mathcal{L}(\phi) \doteq \mathrm{E}_{q_\phi}\left[\textstyle\sum_{t=1}^{T}(\beta_{\text{dyn}}\mathcal{L}_{\text{dyn}}(\phi) + \beta_{\text{rep}}\mathcal{L}_{\text{rep}}(\phi) + \beta_{\text{pred}}\mathcal{L}_{\text{pred}}(\phi))\right]. \tag{4}$$

It consists of the dynamics-based loss components given by $\mathcal{L}_{\text{dyn}}$ and $\mathcal{L}_{\text{rep}}$, defined in Eq. (S1), as well as the loss $\mathcal{L}_{\text{pred}}$ from three prediction heads: observation reconstruction, reward estimation, and continuity prediction.

The following three-step cycle is repeated throughout the training process of DreamerV3: (1) The agent interacts with the environment to collect data, adding it to its replay buffer $\mathcal{B}$. Meanwhile, the latent states (h_t, z_t) are updated closed-loop using the current observation x_t and are used to compute the action. (2) The world model is trained on a batch of sequence data uniformly sampled from the replay buffer using the loss function shown in Eq. (4). (3) Open-loop trajectories are generated in imagination by the world model to train the actor and critic networks.

2.2 Learning Agents

In order to investigate the online and offline training paradigms, we design three off-policy agents, as shown in Fig. 1, each representing a different variation of training data collection.

Active agent is the typical RL agent in online RL. It interacts with the environment and performs training steps using the collected data by its own policy. An Active agent can adapt its world model with its own policy rollouts, which is a self-correction mechanism, enabling the agent to learn from its own mistakes (Ostrovski et al., 2021).

Passive agent is trained offline without any environment interactions by uniformly sampling data from the *final* replay buffer $\mathcal{B}_N$ of an Active agent. This gives the Passive agent access to the full data of the Active agent right from the start of the training process, including high-reward trajectories. It serves as a conventional offline agent, trained on static data without interaction or replay dynamics.

Tandem agent is another agent trained offline, but sees the training data in the same order as the Active agent, i.e. the training batches b_t are replayed exactly as they were sampled during the training of the Active agent (Ostrovski et al., 2021). The goal here is to introduce a more controlled offline learning setting than the Passive agent, with the only difference from the Active agent being the model initialization. This setup facilitates easier interpretation of the experimental results.

The offline agents, Passive and Tandem, are initialized independently of the Active agent used for data collection with a different random seed. The pseudocode of the agents is in Appendix A.5.

3 Experiments

We use DreamerV3 for all our experiments (details on hyperparameters can be found in Appendix A). In total, we conducted 2000 experiments using 20 000 GPU hours. All agents are trained from scratch using task-oriented rewards unless specified otherwise.

3.1 Environment Setup

Our experiments are conducted in the Deepmind Control Suite (DMC) (Tunyasuvunakool et al., 2020; Yarats et al., 2022), Metaworld (Yu et al., 2019), and MinAtar (Young & Tian, 2019) domains, including a total of 31 tasks. These are representative environments for robotic locomotion, manipulation, and discrete game tasks. The environment settings mainly follow the default settings in Hafner et al. (2023). The results for all individual experiments and detailed setups are provided in the Appendix B.5 and Appendix A. Whether state or image observations are used is indicated alongside the task name as "proprio" or "vision" respectively. We run 1 million environment steps per task, training every second step. Results are averaged over three random seeds and reported as the mean with a shaded region indicating ± 1 standard deviation, unless stated otherwise. For the Passive and Tandem agents, we keep the same total number of environment and training steps as the Active agent to ensure consistency and comparability; however, without collecting any interaction data, as explained in Appendix A.4.

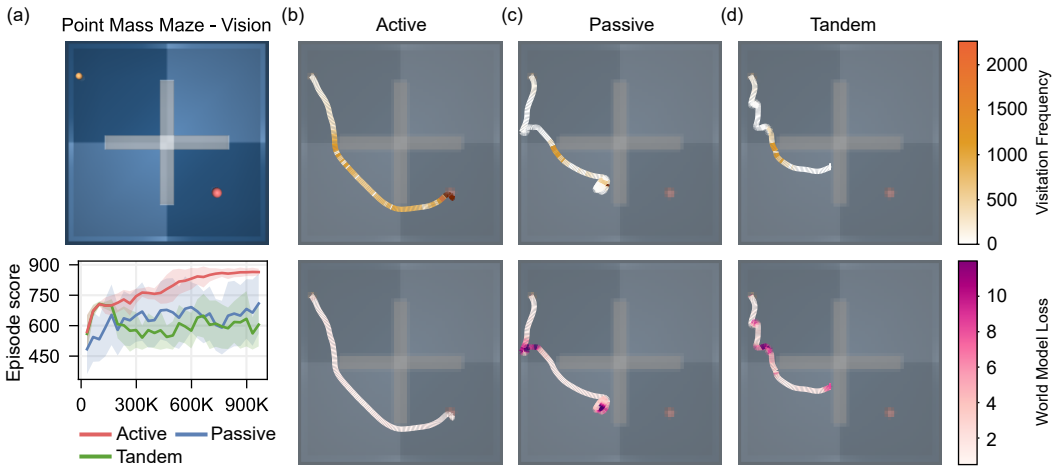

Figure 2: **Example of the degraded performance during offline training in 2D point mass maze environment.** The task is to move the yellow point mass from the top-left initial position to the red marker in the bottom-right of the maze, which is the goal position. The episode score of each agent is shown in (a). In (b-d), we show the point mass trajectory generated by the final model after 1M environment steps. The two heatmaps on the trajectory represent: (1) a count-based frequency of each covered cell that is visited in the replay buffer and (2) world model loss on each visited state. The median visitation frequency along the shown trajectory is 650.0 for Active, 54.5 for Passive, and 37.0 for Tandem.

3.2 Metrics for Analysis

World model loss The mean error of the world model for the prediction of dynamics, observation, reward, and continuity (Sec. 2.1). It is an indicator of the total aleatoric and epistemic model uncertainty and can serve as a simple OOD measure (Yu et al., 2020; Chen et al., 2023).

Episode score The undiscounted sum of rewards over the episode.

The metrics shown in all figures are calculated as follows, unless specified otherwise: (1) Every 5K environment steps, we roll out the agent's policy for a total of 4 episodes. (2) We compute the mean episode score and the mean world model loss across the 4 episodes (see Sec. A.8 for implementation details). Each agent is evaluated in an on-policy manner on its own test-time trajectories. The data distributions of visited states are thus conditioned on the policy and are different for individual agents.

3.3 Toy Example

We first study the performance of all learning agents in a toy environment. We select the point mass maze environment in DMC, where an actuated 2-DoF point mass has to reach the red goal position, as shown in Fig. 2. The results show that only the Active agent successfully solves the task, while both agents trained offline fail, showing degraded performance compared to the Active agent.

Hypothesis: Lack of self-correction causes OOD errors The policy in DreamerV3 is trained purely in the imagination of the world model. As a result, the policy can learn to exploit inaccuracies in the imagination. The Active agent continuously collects data from regions where the world model could be unreliable, specifically for regions where the world model predicts a high reward and, therefore, the policy is likely to visit. Training the world model on the collected data from these regions helps to improve the world model in a targeted manner with respect to the current Active agent's policy. This not only helps to improve the policy to solve the task but also makes the world model adapt to the agent's policy rollouts, ensuring sufficient data coverage around its self-rollouts. Consequently, the agent is unlikely to encounter novel states when rolling out the policy during evaluation.

The agents trained offline lack this critical feedback loop of self-correction. Although the overall training data distribution is the same as the Active agent, differences in sampling sequences (Passive)

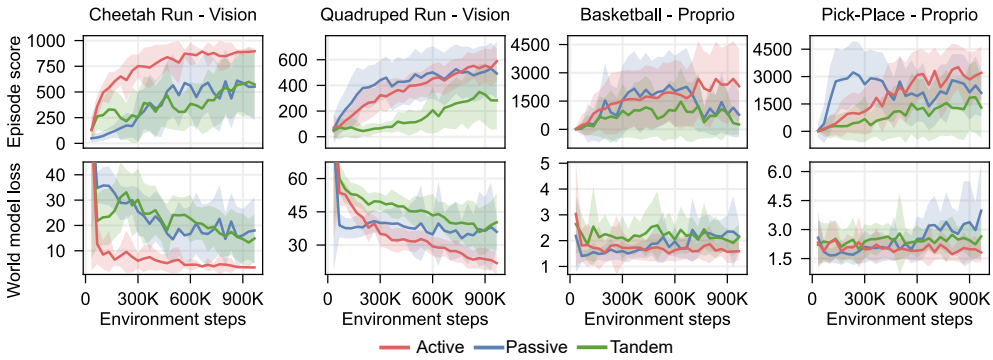

Figure 3: **Episode score and world model loss during evaluation rollouts of 4 selected tasks.** The first two are from DMC and the last two are from the Metaworld domain. The performance degradation of offline agents, including Passive and Tandem, is common across domains and tasks, especially for Tandem agents.

and/or model initializations (Passive and Tandem) lead to distinct policies during training. To effectively improve these policies, the training data generated from the world model's imagination should closely match real rollout performance. However, without self-correction and constrained by data coverage tailored to another agent's policy, the imagination of this limited-capability world model fails to align with real rollouts under its own policy, leading to a persistent discrepancy between imagination and reality in offline training. Consequently, the policy will exploit these inaccuracies during training and be updated blindly to eventually steer the agent toward novel, unvisited areas. During test time, visiting novel states can lead to world model prediction errors and, therefore, suboptimal policy actions. It creates a catastrophic cycle where each compromised action leads to further novel states and additional inaccuracies in the world model until the episode ends or the agent accidentally re-enters into a familiar state.

We observe this behavior in the performance of the three agents as shown in Fig. 2. The Active agent learned to adapt its world model to its own rollouts; therefore, it did not meet any novel states when rolling out the policy for evaluation, as shown by the consistent low world model loss and high visitation frequencies alongside its trajectory. However, this is not the case for the Passive and Tandem agents. From the start, their policies seem to behave anomalously, guiding them towards a suboptimal direction even in the regions familiar to the world model. This is due to the absence of feedback: the policy exploits model inaccuracies, and blind updates cause it to drift further from optimal behavior over time. Since the task-oriented dataset has limited state-space coverage, they inevitably visit novel states. Although both the Passive and Tandem agents re-enter familiar states under arbitrary actions, their compromised policies subsequently lead them to another OOD region, where they cannot recover until the end of the episode—ultimately failing to solve the task.

To summarize, **self-correction ensures sufficient data coverage related to the agent's policy rollouts**, thereby **(1) preventing OOD errors** and **(2) facilitating policy training** by reducing gaps between imaginations and real rollouts. Without self-correction, imagination gaps compromise policy training and push offline agents toward OOD states, where they become trapped in a catastrophic cycle that leads to further performance degradation.

Our hypothesis is generally in line with previous research in model-free RL (Ostrovski et al., 2021; Yue et al., 2023; Emedom-Nnamdi et al., 2023; Kumar et al., 2020b) , which attributes performance degradation to extrapolation errors in Q-values in OOD state-action pairs during training and evaluation. However, in the context of MBRL, the paradigm is shifted from a focus on Q-functions to the coupling of a world model and a policy network.

3.4 Validation across Tasks

The performance degradation phenomenon in offline agents is observed across various tasks and domains, as shown in Fig. 3 and Appendix B.5.2. In tasks such as *Quadruped Run - Vision* and *Pick-Place - Proprio*, the Passive agent initially demonstrates a faster increase in performance but has a larger variance or even experiences performance drops as training progresses. The degraded per-

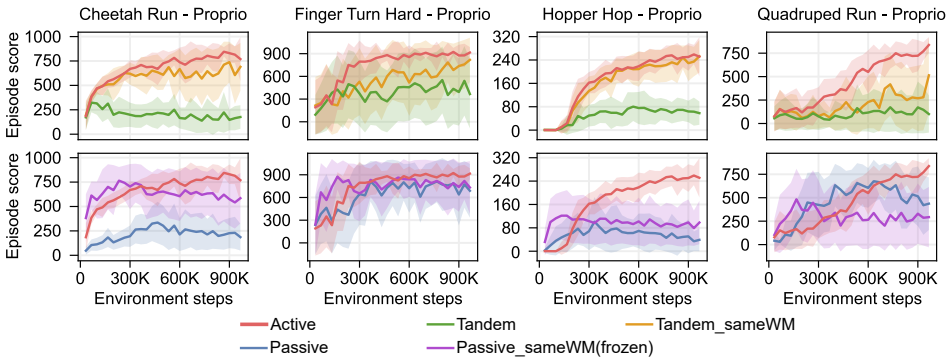

Figure 4: **Performance comparison when keeping an equivalent world model in Passive or Tandem agents to the one of the Active agent** throughout training. Despite utilizing the same world model during training, performance degradation still occurs, albeit to varying degrees.

formance in Passive and Tandem agents is accompanied by a significantly larger world model loss on evaluation episodes than the Active agent. Given that a high world model loss indicates novel states, this observation supports our hypothesis in Sec. 3.3. The discrepancy between imagined and real rollouts in offline agents is shown in Appendix B.1. Our detailed inspections on a timestep level in Appendix B.2 further validate our hypothesis of the catastrophic cycle during testing. Fig. 3 also shows a potential advantage of Passive agents: faster convergence by having access to high reward trajectories from the start of training (validated in Appendix B.3), though additional measures may be necessary to ensure training stability. The results of Tandem agents also follow the findings of degraded performance of the Tandem training regime in Ostrovski et al. (2021) and extend its validity to MBRL. We include more discussions about the Tandem agent and the self-correction mechanism in Appendix B.4.

3.5 Deep Dive into Performance Degradation

3.5.1 OOD in MBRL

Both world model and policy affect performance degradation To decouple the effect of the world model and the policy on the performance degradation, we carry out a more controlled experiment as shown in Fig. 4. In this setup, the Tandem agent's world model replicates that of the Active agent precisely at each training step, which is referred to as Tandem_sameWM. For Passive agents, we keep using the final world model from their Active counterpart for the remainder of training, which is named Passive_sameWM(frozen). The pseudocode of the agents is in Appendix A.5.

After isolating the effect of different world models, we observe that the degradation still persists although the extent of it varies across tasks. In tasks such as *Hopper Hop - Proprio*, the performance degradation of the Tandem_sameWM agent is minimal, while it remains significant in others like *Quadruped - Proprio*. A similar trend appears in Passive_sameWM(frozen) agents. These findings suggest that deviations in both the world model and policy from the Active agent contribute to performance degradation, with their relative impacts depending on the specific task. More discussions about this experiment can be found in Appendix B.4.

What is the difference to supervised learning? In classical supervised learning, a model is optimized on an offline dataset, e.g., for image classification. Training on independent and identically distributed data from different random initializations typically yields similar performance, showing robustness to initialization. Why is this not the case in the MBRL setting, where Tandem agents perform worse than Active agents, despite one expecting the world model to perform equally well across seeds given the same data? This is because offline trained agents will cause states to be visited during policy optimization that are not collected by the Active agent, leading to OOD queries to the model.

3.5.2 World model loss is a pessimistic indicator of performance degradation

The world model loss is due to prediction errors arising from both epistemic and aleatoric uncertainty. Novel states lead to high variance predictions due to epistemic uncertainty induced by insufficient state space coverage during training. Overlaid are errors due to partial observability and environment stochasticity.

In particular, the latter factors can lead to high model loss without significant impacts on performance, depending on whether exact predictions are required for the task at hand.

In addition, even when the agent is in novel states, other factors, e.g. environment constraints, and the policy producing correct actions by coincidence in hallucinations of the world model, can reduce the impact of a poorly performing world model on agent performance. Therefore, the world model loss is a pessimistic indicator of performance degradation.

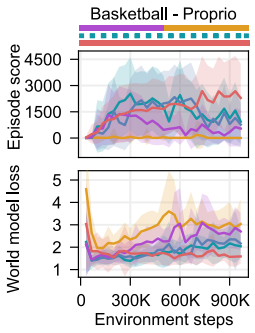

Figure 5: **Performance comparison of Active, Passive as well as Passive agents trained on expert, suboptimal, and mixed data**, which is implemented by splitting the replay buffer of the Active agent in different ways.

3.5.3 Expert data alone exacerbates OOD issues

Expert data is commonly used in offline learning, but compared to data collected by the Active agent, its coverage is more limited to task-specific trajectories, typically capturing only certain ways of solving the task. As a result, states are more likely to be OOD for the world model, resulting in even worse task performance, as shown in Fig. 5, where we treat the second half of the buffer as expert data. As expected, the world model loss evaluated on test-time trajectories is significantly larger than for other agents. For more details, see Appendix B.3.

3.5.4 Considerations in Practical Applications

In further experiments, we find that initializing the Passive agents' weights identically to the Active agents' does not improve task performance. In contrast, even minor differences in the model initialization of Tandem agents compared to Active agents leads to degraded performance, reflecting the chaotic training dynamics of gradient-based optimization. See Appendix B.3 for more details.

4 Potential Remedies from a Data Perspective

Based on the previous analysis, we conclude that insufficient state coverage during training of Passive and Tandem agents limits the generalization capability of the world model, which results in the policy exploiting inaccuracies of the world model during training and eventually visiting OOD states during evaluation. To address this, we propose two strategies for effective agent training with offline datasets: **training on an exploration dataset** and **(adaptively) incorporating self-generated data**.

4.1 Training on Exploration Data

We investigate how training on exploration data affects the performance of Active, Passive and Tandem agents. Here, we use Plan2Explore (Sekar et al., 2020), where the objective is to maximize the information gain of the world model. The exploration reward is calculated as ensemble disagreement, denoted by r_{disag}. We investigate exploration in two modes: (1) pure exploration in a task-free setting, i.e. the agent only maximizes for r_{disag}, (2) a mixed reward setting, where r_{disag} is added as an exploration bonus on top of the task reward:

$$r_t \doteq w_{\text{task}} \cdot r_{\text{task}} + w_{\text{expl}} \cdot r_{\text{disag}}, \tag{5}$$

where w_{task} and w_{expl} weights are normalized such that they sum up to 1. In both cases, the Active agent trains two separate policies: an exploration policy (guided by either pure exploration or the mixed reward, used to collect data), and a task policy (trained only for task evaluation).

For agents trained offline, exploration data in the training set can provide a larger state-space coverage, which can counteract the missing self-correction mechanisms of an active agent. Fig. 6 demonstrates how task-oriented data is narrower compared to exploration data. The addition of exploration data becomes crucial in alleviating the OOD challenge during evaluation, as validated in Fig. 7, where the training data is gathered by an Active agent based on pure exploration rewards r_{disag}. As a result, the Passive agents generally outperform their Active counterparts, and in some tasks even match the performance of the online task-oriented version, while the Tandem agents perform comparably to the Active agents in most cases. Furthermore, the relationship between task performance and world model loss generally also matches the findings in Sec. 3.4. However, some cases in Appendix B.5.4 indicate that world model loss can occasionally be less predictive of task performance. This inconsistency arises as novel regions for the world model shrink with exploration data, leading to lower world model loss—even in regions far from typical task trajectories (more discussions in Appendix B.5.6). In addition, the pure exploration dataset contains numerous trajectories irrelevant to the task, interfering with the effective learning of the task policy. Consequently, task performance becomes increasingly dependent on the task difficulty. For example, in two challenging tasks – *Quadruped Run - Vision* and *Pick-Place - Proprio* – agents trained on pure exploration data have significantly lower performance than those trained with task-oriented data, as shown in Fig. 7.

To this end, we investigate the mixed reward setting, where we add the exploration reward as a bonus, as defined in Eq. (5). This approach allows a more concentrated exploration near the goal, as shown in Fig. 6, preventing the excessive exploration of irrelevant areas that could arise from a purely explorative dataset.

Indeed, in Fig. 8, we show that pure exploration is hardly the best option for the hard tasks like *Quadruped Run - Vision*. The addition of an exploration bonus e.g. $w_{\text{expl}} = 0.5$ together with task rewards in *Quadruped Run - Vision* can lead to an improved task performance compared to runs with pure task rewards, especially in Passive agents. A downside of this approach is the introduction of the hyperparameter w_{expl}, the optimal value of which can depend on the specific task as shown in our experiments in Appendix B.5.1.

4.2 Adding Additional Self-generated Data

We have demonstrated the critical importance of self-correction. However, as training solely on interaction data is expensive, and offline data is often cheaply available; we would like to explore how one can most effectively combine fixed offline data with online interaction data. To analyze this interplay, we first examine a strategy that uses a predetermined schedule for the Passive agent to interact with its environment.

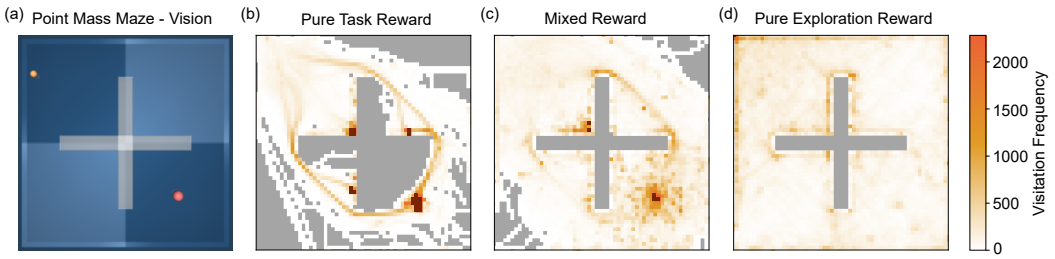

Figure 6: **State visitation in the Point Mass Maze task.** They are calculated using the discretized states from three different Active agents' final replay buffers after 1M environment steps. (b) Agent in a pure task-oriented setting. (c) Agent with a mixed reward: task plus exploration rewards, see Eq. (5) with $w_{\text{expl}} = 0.5$. (d) Agent with pure exploration rewards based on ensemble disagreement (Sekar et al., 2020). The unvisited areas are painted gray, and the outliers that have extremely high values are painted dark red. Here the task-oriented agent only explores limited state space in the map and always follows certain routes towards the goal position, while the two explorative agents visit the space much more equally.

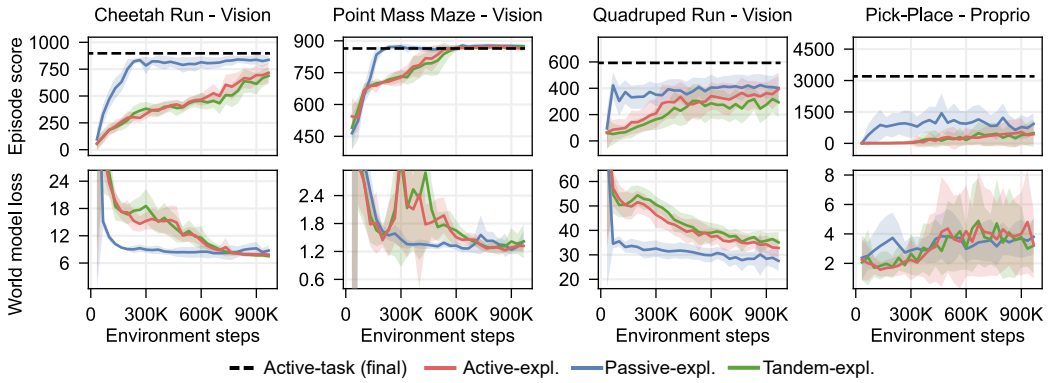

Figure 7: **Performance comparison when training on pure exploration data.** The dataset is generated by the Active-expl. agent with a behavioral policy based on ensemble disagreement (Sekar et al., 2020). We additionally show the baseline performance of a task-oriented Active agent.

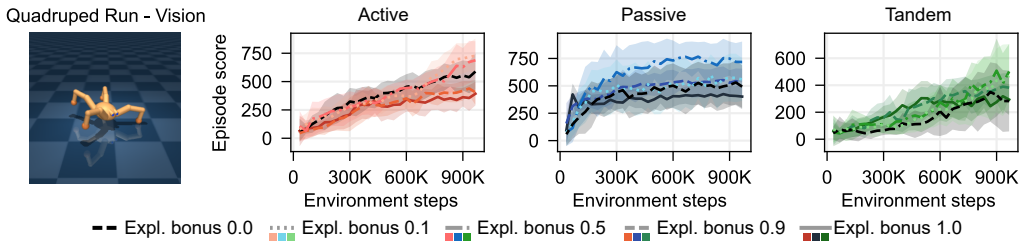

Figure 8: **Training on pure exploration data is not optimal.** Performance comparison when assigning different exploration bonuses w_{expl} in the reward function. The black dashed lines represent pure task-oriented policy without any exploration bonus.

Specifically, for every N environment steps, the Passive agent is allowed to collect 2K-step transitions based on its learned policy. Then the interactive data will be added to expand the replay buffer for later sampling during world model training as usual. By choosing a different N, we can adjust the frequency of interactive data injection. Experiments were conducted with N set to 4K, 20K, and 200K, respectively corresponding to 50%, 10%, and 1% self-generated data. The results are shown in Fig. 9. Accordingly, merely 10% additional self-generated data can already result in a significant improvement in the episode score as well as a notable reduction in the world model loss, recovering the performance of its Active counterpart. In certain environments, such as the *Spaceinvaders* from the MinAtar domain, the Passive agents may already solve the task and have a faster convergence than the Active one; therefore, self-generated data provides no performance increase.

Adaptive interaction Upon examining the results with a fixed schedule, we see that interaction ratios to restore agents' performance vary across tasks. Therefore, we analyze an adaptive interaction schedule based on the insights of OOD states causing degenerate performance. We calculate a ratio by dividing the world model loss on evaluation trajectories by the loss on trajectories in the replay buffer. This ratio measures the novelty of the trajectories visited by the current learned policy compared to those seen during training and enables a single threshold for adding self-generated data across tasks.

We set the threshold for the OOD ratio to 1.35 (see the ablation study in Appendix A.7) and inspect it every 5K environment steps over 4 evaluation episodes. If the OOD ratio exceeds this threshold, the Passive agent collects 2K-step transitions from the environment using its learned policy, denoted as Passive+Auto interact (refer to Appendix A.5 for the agent's pseudocode). As shown in Fig. 9, this strategy fine-tunes self-generated data injection based on task demands, achieving similar performance with less data (5.67% across 31 tasks) compared to an agent that regularly adds 10% self-generated data. The inspection frequency can be reduced to lower evaluation costs. For more results, see Appendix B.5.3. This strategy involves online evaluations. A complete offline evaluation would be desirable, but is outside the scope of this paper. We hope to inspire research in this direction.

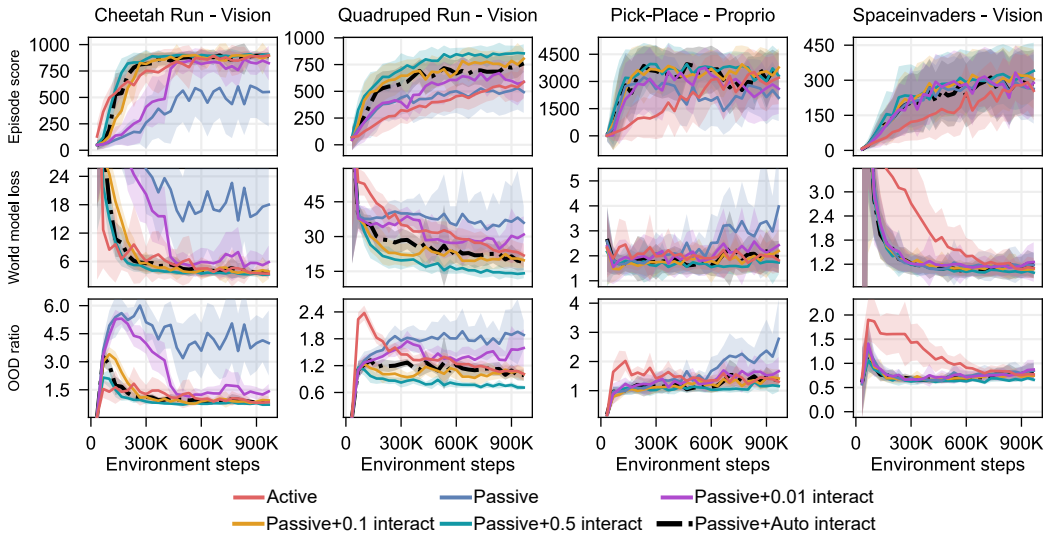

Figure 9: **Performance comparison when allowing adding additional self-generated data for Passive agents.** The Passive+Auto interact agent adds 6.5% self-generated data in Cheetah Run - Vision, 2.9% in Quadruped Run - Vision, 9.8% in Pick-Place - Proprio, and 0.5% in Spaceinvaders. The percentage is calculated w.r.t. to the size of the final replay buffer of Active agents.

5 Related Work

Performance Degradation in Offline Model-based Agents Performance degradation of offline agents is a known phenomenon in MBRL (He, 2023) and is mainly attributed to two factors:

(1) The distribution mismatch between training data and the states visited by the learned policy (Kidambi et al., 2020; Chen et al., 2023; Yu et al., 2020; Cang et al., 2021; Ross & Bagnell, 2012). These inaccuracies in the world model within unseen regions are then exacerbated by compounding errors in multi-step predictions (Asadi et al., 2019; Janner et al., 2019). These accumulated errors in the model-based imagination process based on OOD queries can mislead both policy training (Wang et al., 2024) and planning by overestimation in critics (Sims et al., 2024), ultimately resulting in a performance drop.

(2) The inability of offline agents to self-correct through active data collection (He, 2023; Cang et al., 2021; Yu et al., 2020). Prior works on offline agents (Ostrovski et al., 2021; Tang et al., 2024; Emedom-Nnamdi et al., 2023; Lin et al., 2024) have shown that utilizing data from interactions with the environment introduces a corrective feedback loop (Kumar et al., 2020a), allowing the agent to learn from its own mistakes and consequently improve its task performance.

Building on existing studies, we explore phenomena across various tasks and domains in model-based RL using DreamerV3. Additionally, we investigate the conditions (e.g. the nature and quality of the dataset) that exacerbate distribution mismatches and model inaccuracies.

Remedies to Support Offline Training To address performance degradation in offline model-based agents, many studies add conservatism to their algorithms. One method is to include an uncertainty penalty in the reward function to deter the agent from exploring new states (Kidambi et al., 2020; Yu et al., 2020; 2021; Wang et al., 2024), while another employs trust-region updates to maintain the learned policy's proximity to the data collection policy (Matsushima et al., 2021). RAMBO (Rigter et al., 2022) trains an adversarial environment model that generates pessimistic transitions for OOD state-action pairs, reducing the value function in uncertain regions. In contrast, MAPLE (Chen et al., 2023) enables adaptive agent behavior in OOD regions during deployment, using a context-aware policy based on meta-learning techniques.

While these methods provide insights on mitigating performance degradation in offline MBRL, few address which type of data best facilitates offline training. In model-free RL, studies suggest adding self-generated data (Ostrovski et al., 2021; Lee et al., 2021) and emphasize the importance of diversity and exploration (Mediratta et al., 2024; Suau et al., 2024; Kanitscheider et al., 2021; Kim et al., 2023;

Guo et al., 2022). We extend these ideas to model-based RL with validation in various tasks and domains.

6 Conclusions and Discussions

We study the effect of data collection for model-based RL agents for offline learning. Through a wide range of experiments across various domains, we show that data collection has a huge impact on performance and find that pure offline methods suffer from degraded performance. The reason is that novel states are visited during evaluation. This tendency to visit OOD states arises from the lack of self-correction in offline training on data with limited state-space coverage. The resulting mismatch between imagined and real rollouts misleads policy training and drives agents toward failure. From a data perspective, we identify that training on partially exploratory data collected using a mixed task-exploration reward function is effective in mitigating performance degradation. Importantly, training offline solely on expert data exacerbates performance degradation compared to a typical mixed dataset due to severe OOD issues. Additionally, our experiments show that adding as little as 10% self-generated data at regular intervals can significantly enhance the performance of Passive agents. When we allow the Passive agent to adaptively interact based on its world model loss as a proxy measure of OOD state visitation, we observe a significant performance improvement while minimizing the need for additional interaction data. However, our method still requires evaluation rollouts. An offline measure would be desirable and is left for future research.

Overall, we highlight the importance of sufficient state-space coverage in the training data to train a robust model-based agent, which can be achieved either by an explorative offline dataset or by enabling the agent to learn from its own mistakes. As efforts to collect large-scale real-world data for robotics are increasing, the question arises: What is the best way to collect data to facilitate robust agent training? As model-based RL shows strong task performance and promises efficient fine-tuning and good transfer capabilities for new tasks, we suggest that dataset collection should incorporate exploration data. We plan to extend our experiments to other RL methods and real-world scenarios to identify optimal data collection strategies. We believe that our insights can help design a data-efficient fine-tuning method for robotics foundation models. This will help develop more resilient and adaptable agents capable of performing reliably in complex environments.

Appendix

Acknowledgments

Funded/Co-funded by the European Union (ERC, REAL-RL, 101045454). Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. This work was supported by the Volkswagen Stiftung (No 98 571) and by the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039A. The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting CS. JF is supported by the Max Planck ETH Center for Learning Systems. Georg Martius is a member of the Machine Learning Cluster of Excellence, funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC number 2064/1 – Project number 390727645.

References

Kavosh Asadi, Dipendra Misra, Seungchan Kim, and Michael L. Littman. Combating the compounding-error problem with a multi-step model. *CoRR*, abs/1905.13320, 2019.

Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge (Jimmy) Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah

Bechtle, Feryal Behbahani, Stephanie Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando De Freitas, Satinder Singh, and Tim Rocktäschel. Genie: generative interactive environments. In *International Conference on Machine Learning*, 2024.

Catherine Cang, Aravind Rajeswaran, Pieter Abbeel, and Michael Laskin. Behavioral priors and dynamics models: Improving performance and domain transfer in offline rl. *CoRR*, abs/2106.09119, 2021.

Xiong-Hui Chen, Fan-Ming Luo, Yang Yu, Qingyang Li, Zhiwei Qin, Wenjie Shang, and Jieping Ye. Offline model-based adaptable policy learning for decision-making in out-of-support regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(12):15260–15274, 2023.

Patrick Emedom-Nnamdi, Abram L. Friesen, Bobak Shahriari, Nando de Freitas, and Matt W. Hoffman. Knowledge transfer from teachers to learners in growing-batch reinforcement learning. *CoRR*, abs/2305.03870, 2023.

Caglar Gulcehre, Sergio Gómez Colmenarejo, Ziyu Wang, Jakub Sygnowski, Thomas Paine, Konrad Zolna, Yutian Chen, Matthew Hoffman, Razvan Pascanu, and Nando de Freitas. Regularized behavior value estimation. *CoRR*, abs/2103.09575, 2021.

Zhaohan Daniel Guo, Shantanu Thakoor, Miruna Pîslar, Bernardo Avila Pires, Florent Altché, Corentin Tallec, Alaa Saade, Daniele Calandriello, Jean-Bastien Grill, Yunhao Tang, Michal Valko, Rémi Munos, Mohammad Gheshlaghi Azar, and Bilal Piot. Byol-explore: exploration by bootstrapped prediction. *Advances in Neural Information Processing Systems*, 35:31855–31870, 2022.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Haoyang He. A survey on offline model-based reinforcement learning. *CoRR*, abs/2305.03360, 2023.

Weidong Huang, Jiaming Ji, Borong Zhang, Chunhe Xia, and Yaodong Yang. Safedreamer: Safe reinforcement learning with world models. In *The Twelfth International Conference on Learning Representations*, 2024.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in Neural Information Processing Systems*, 32:12498–12509, 2019.

Ingmar Kanitscheider, Joost Huizinga, David Farhi, William Hebgen Guss, Brandon Houghton, Raul Sampedro, Peter Zhokhov, Bowen Baker, Adrien Ecoffet, Jie Tang, Oleg Klimov, and Jeff Clune. Multi-task curriculum learning in a complex, visual, hard-exploration domain: Minecraft. *CoRR*, abs/2106.14876, 2021.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 21810–21823, 2020.

Hyun Kim, Injun Park, Ingook Jang, Seonghyun Kim, Samyeul Noh, and Joonmyon Cho. Exploring generalization and adaptability of offline reinforcement learning for robot manipulation. In *2023 23rd International Conference on Control, Automation and Systems (ICCAS)*, pp. 1542–1547, 2023.

Aviral Kumar, Abhishek Gupta, and Sergey Levine. Discor: Corrective feedback in reinforcement learning via distribution correction. *Advances in Neural Information Processing Systems*, 33: 18560–18572, 2020a.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020b.

Aviral Kumar, Rishabh Agarwal, Xinyang Geng, George Tucker, and Sergey Levine. Offline q-learning on diverse multi-task data both scales and generalizes. In *The Eleventh International Conference on Learning Representations*, 2022.

Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, volume 164, pp. 1702–1712. PMLR, 2021.

Zhixuan Lin, Pierluca D'Oro, Evgenii Nikishin, and Aaron Courville. The curse of diversity in ensemble-based exploration. In *The Twelfth International Conference on Learning Representations*, 2024.

Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. In *9th International Conference on Learning Representations*, 2021.

Ishita Mediratta, Qingfei You, Minqi Jiang, and Roberta Raileanu. The generalization gap in offline reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Georg Ostrovski, Pablo Samuel Castro, and Will Dabney. The difficulty of passive learning in deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34:23283–23295, 2021.

Marc Rigter, Bruno Lacerda, and Nick Hawes. Rambo-rl: Robust adversarial model-based offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:16082–16097, 2022.

Stéphane Ross and J. Andrew Bagnell. Agnostic system identification for model-based reinforcement learning. In *Proceedings of the 29th International Conference on Machine Learning*, pp. 1905–1912, 2012.

Julian Schrittwieser, Thomas Hubert, Amol Mandhane, Mohammadamin Barekatain, Ioannis Antonoglou, and David Silver. Online and offline reinforcement learning by planning with a learned model. *Advances in Neural Information Processing Systems*, 34:27580–27591, 2021.

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, pp. 8583–8592. PMLR, 2020.

Anya Sims, Cong Lu, and Yee Whye Teh. The edge-of-reach problem in offline model-based reinforcement learning. *CoRR*, abs/2402.12527, 2024.

Miguel Suau, Matthijs T. J. Spaan, and Frans A. Oliehoek. Bad habits: Policy confounding and out-of-trajectory generalization in rl. *RLJ*, 4:1711–1732, 2024.

Yunhao Tang, Daniel Zhaohan Guo, Zeyu Zheng, Daniele Calandriello, Yuan Cao, Eugene Tarassov, Rémi Munos, Bernardo Ávila Pires, Michal Valko, Yong Cheng, and Will Dabney. Understanding the performance gap between online and offline alignment algorithms. *CoRR*, abs/2405.08448, 2024.

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*, 6:100022, 2020.

Xiyao Wang, Ruijie Zheng, Yanchao Sun, Ruonan Jia, Wichayaporn Wongkamjan, Huazhe Xu, and Furong Huang. Coplanner: Plan to roll out conservatively but to explore optimistically for model-based rl. In *The Twelfth International Conference on Learning Representations*, 2024.

Denis Yarats, David Brandfonbrener, Hao Liu, Michael Laskin, Pieter Abbeel, Alessandro Lazaric, and Lerrel Pinto. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. *CoRR*, abs/2201.13425, 2022.

Kenny Young and Tian Tian. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*, 2019.

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 *3rd Annual Conference on Robot Learning*, volume 100, pp. 1094–1100. PMLR, 2019.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in Neural Information Processing Systems*, 34:28954–28967, 2021.

Yang Yue, Bingyi Kang, Xiao Ma, Zhongwen Xu, Gao Huang, and Shuicheng Yan. Boosting offline reinforcement learning via data rebalancing. *CoRR*, abs/2210.09241, 2022.

Yang Yue, Rui Lu, Bingyi Kang, Shiji Song, and Gao Huang. Understanding, predicting and better resolving q-value divergence in offline-RL. *Advances in Neural Information Processing Systems*, 36:60247–60277, 2023.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Implementation Details

A.1 Runtime Overview

Our experiments comprised approximately 2000 runs, totaling 20000 GPU hours. Each run took between 8 and 15 hours, depending on the specific task. All experiments were conducted using NVIDIA RTX 4090 or A100 GPUs.

A.2 Model Hyperparameters

For all experiments, we use the same model size S, defined in Hafner et al. (2023). Each agent, which consists of a world model, an actor network, and a critic network, has a total of 18M optimizable variables. We follow the default values in Hafner et al. (2023) for the training hyperparameters e.g. learning rate and batch size for each component of the agent as well as other hyperparameters. For more details about DreamerV3, please refer to Hafner et al. (2023).

A.3 Environment Hyperparameters

We list the environment hyperparameters in Tab. S1. The implementation of the task *Point Mass Maze* is based on Yarats et al. (2022).

Table S1: Environment hyperparameters for each domain

Hyperparameter	DMC	Metaworld	MinAtar
Image Size	[64,64]	[64,64]	[32,32]
Action Repeat	2	2	1
Episode Truncate	-	-	2500
Parallel Env Num	4	4	4
Train Ratio	512	512	512

A.4 Environment Steps in Offline Agents

Tracking performance metrics relative to environment steps during online training is standard practice in the RL community. This methodology is also applied in the analysis of the offline Tandem agent in Ostrovski et al. (2021), which closely mirrors the behavior of its Active counterpart.

However, the Passive agent—by definition—does not interact with the environment and thus cannot influence environment steps. This poses a challenge for directly comparing its performance with that of the Active and Tandem agents. To ensure comparability across training procedures, we allow the Passive agent to interact with the environment during training in the same manner as an online agent, but without adding the resulting interaction data into its replay buffer. This setup enables the Passive agent to remain trained solely on an offline dataset while allowing performance comparisons based on environment steps, with only minimal code changes required.

A.5 Pseudocode of methods

We add the pseudocode of the Active, Passive, and Tandem agents (in Alg. 1), the variation of Passive and Tandem agents (in Alg. 2) used in decoupled analysis (Sec. 3.5.1), as well as the second

remedy (in Alg. 3) for better clarity. Below, the training of the world model M includes training all components in Eq. (2), while training π includes all components in Eq. (3).

Algorithm 1 Learning agents

Active Agent	Passive Agent	Tandem Agent
Initialize: Replay buffer $\mathcal{B}$ = a few random episodes.	**Initialize:** Replay buffer $\mathcal{B}$ = final $\mathcal{B}_A$.	**Initialize:** Replay buffer $\mathcal{B}$ = final $\mathcal{B}_A$.
World model M + Policy π by seed S_A.	World model M + Policy π by seed S_P.	World model M + Policy π by seed S_T.
for each step i **do**	**for** each step i **do**	**for** each step i **do**
Sample $\mathcal{D}_A^i \sim \mathcal{B}$	Sample $\mathcal{D}_P^i \sim \mathcal{B}$	Copy $\mathcal{D}_T^i = \mathcal{D}_A^i$
Update M using $\mathcal{D}_A^i$	Update M using $\mathcal{D}_P^i$	Update M using $\mathcal{D}_T^i$
Train π in the imagination of M	Train π in the imagination of M	Train π in the imagination of M
Execute π in the env to expand $\mathcal{B}$	-	-
	-	-
Return: Final $\mathcal{B}_A$, π	**Return:** π	**Return:** π

Algorithm 2 Learning agents in decoupled analysis

Active Agent	Passive_sameWM (frozen)	Tandem_sameWM
Initialize: Replay buffer $\mathcal{B}$ = a few random episodes.	**Initialize:** Replay buffer $\mathcal{B}$ = final $\mathcal{B}_A$.	**Initialize:** Replay buffer $\mathcal{B}$ = final $\mathcal{B}_A$.
World model M + Policy π by seed S_A.	World model M = final M_A, only Policy π by seed S_P.	only Policy π by seed S_T.
-		-
		-
for each step i **do**	**for** each step i **do**	**for** each step i **do**
Sample $\mathcal{D}_A^i \sim \mathcal{B}$	Sample $\mathcal{D}_P^i \sim \mathcal{B}$	Copy $\mathcal{D}_T^i = \mathcal{D}_A^i$
Update M_A^i using $\mathcal{D}_A^i$	-	Copy $M = M_A^i$
Train π in the imagination of M	Train π in the imagination of M	Train π in the imagination of M
Execute π in the env to expand $\mathcal{B}$	-	-
	-	-
Return: Final $\mathcal{B}_A$, M_A, π	**Return:** π	**Return:** π

Algorithm 3 Passive agents adding additional self-generated data (K denotes thousand)

Passive Agent	Fixed Schedule	Adaptive Schedule
Initialize: Replay buffer $\mathcal{B}$ = final $\mathcal{B}_A$.	**Initialize:** Replay buffer $\mathcal{B}$ = final $\mathcal{B}_A$.	**Initialize:** Replay buffer $\mathcal{B}$ = final $\mathcal{B}_A$.
World model M + Policy π by seed S_P.	World model M + Policy π by seed S_P.	World model M + Policy π by seed S_P.
for each step i **do**	**for** each step i **do**	**for** each step i **do**
Sample $\mathcal{D}^i \sim \mathcal{B}$	Sample $\mathcal{D}^i \sim \mathcal{B}$	Sample $\mathcal{D}^i \sim \mathcal{B}$
Update M using $\mathcal{D}^i$	Update M using $\mathcal{D}^i$	Update M using $\mathcal{D}^i$
Train π in the imagination of M	Train π in the imagination of M	Train π in the imagination of M
-	**if** $i\%N == 0$ **then** // $N = 4\text{K}, 20\text{K}, 200\text{K}$	**if** $i\%5\text{K} == 0$ and $\text{ood_ratio}_i > \text{thres.}$ **then**
-	Execute π in the env to expand $\mathcal{B}$ by 2K step data	Execute π in the env to expand $\mathcal{B}$ by 2K step data
-		
-		
Return: π	**Return:** Final $\mathcal{B}$, π	**Return:** Final $\mathcal{B}$, π

A.6 Supplementary of DreamerV3

The computation of each component in the world model loss:

$$
\begin{aligned}
\mathcal{L}_{\text{pred}}(\phi) &\doteq -\ln p_\phi(x_t \mid z_t, h_t) - \ln p_\phi(r_t \mid z_t, h_t) - \ln p_\phi(c_t \mid z_t, h_t) \\
\mathcal{L}_{\text{dyn}}(\phi) &\doteq \max\big(1, \mathrm{KL}\big[\mathrm{sg}(q_\phi(z_t \mid h_t, x_t)) \;\big\|\; p_\phi(\hat{z}_t \mid h_t)\big]\big) \\
\mathcal{L}_{\text{rep}}(\phi) &\doteq \max\big(1, \mathrm{KL}\big[q_\phi(z_t \mid h_t, x_t) \;\big\|\; \mathrm{sg}(p_\phi(\hat{z}_t \mid h_t))\big]\big)
\end{aligned}
\tag{S1}
$$

A.7 Ablation Studies

We test different threshold values used in adaptive Passive agents for autonomously adding self-generated interaction data. In Fig. S13, we observe that the majority OOD ratio in Active agents reaches below 2.0 during training. Therefore, we begin with an upper bound threshold value of 2.0 and test four values: 2.0, 1.65, 1.35, and 1.2. It is important to note that this upper bound serves solely as a reference point for initiating the ablation studies and does not imply any dependence of the OOD ratio on the performance of the Active agent. In Fig. S1, we show that although a lower threshold value (e.g. 1.2) could bring more self-generated data (about 10% average) to the replay buffer, the improvement in performance is not significant compared to other higher values. However, a high threshold value (e.g. 2.0) makes the training process less stable, as shown in the relatively low normalized mean score and an increasing tendency of OOD ratio from step 800K, compared to lower threshold values. But generally, the sensitivity of this threshold value to performance is low. One can set a low threshold value if the training budget allows. In the main experiments, we choose a middle threshold value of 1.35, which balances the number of added interaction data and stable performance. Although 1.65 performs similarly with fewer samples in the three test environments, we conservatively adopt 1.35 to ensure greater stability in potentially more challenging settings.

A.8 Evaluation Episodes Details for Metrics Computation

During training, each evaluation phase involves rolling out the agent's policy for a total of 4 episodes. The episode score (described in Sec. 3.2) is computed on the entire episode trajectory. For other metrics—such as world model loss—we collect up to 500 steps per episode using a first-in-first-out (FIFO) buffer. The world model loss is computed over the collected steps and averaged to obtain a mean value for each episode. Additional metrics, such as the value function (described in Sec. A.9), are computed at the first step of the buffer. All metric values are then averaged across the 4 evaluation episodes.

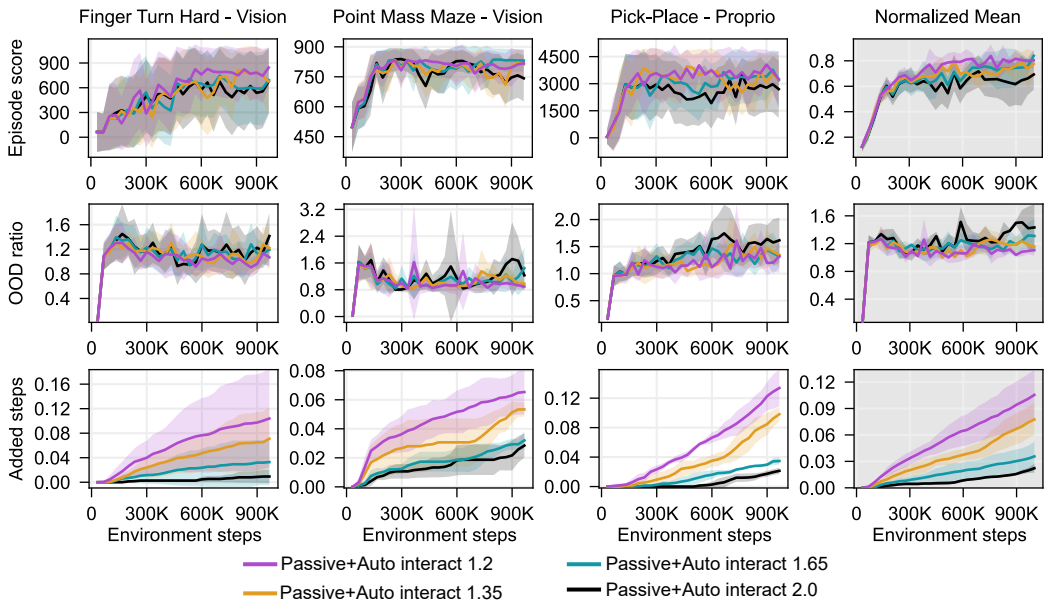

Figure S1: **Ablation studies on threshold value for adaptive Passive agents.** We test four threshold values: 2.0, 1.65, 1.35, and 1.2 in three tasks. The last column shows a normalized mean across tasks. The number of added steps in the third row is shown as a percentage of the original replay buffer size.

A.9 Additional Metrics

Policy input reconstruction loss We train an autoencoder functioning as an OOD detector for the policy inputs. The autoencoder is optimized to minimize the negative log-likelihood (Eq. S2) to reconstruct the policy input. Novel policy inputs, that may compromise the quality of output actions, can be detected using the Mean Squared Error (MSE) reconstruction loss. A higher MSE indicates that the input is likely novel or anomalous, suggesting the input differs significantly from the training distribution and could lead to an unreliable policy action.

$$\mathcal{L}_{\text{recon}}(\phi) \doteq -\ln p_\phi(z_t, h_t \mid \text{encoder}(z_t, h_t)) \tag{S2}$$

Value function The expected discounted return—the cumulative sum of future rewards, as shown in Eq. (1).

The additional metrics are calculated as follows unless specified otherwise: (1) Every 5K environment steps, we roll out the agent's policy for a total of 4 episodes. (2) We compute the policy input reconstruction loss across the 4 episodes. For the value function, we calculate it at the initial state of the collected trajectory per episode and then average these values across the 4 episodes. The implementation details can be found in Sec. A.8.

B Result Analyses

B.1 Discrepancy between Imagination and Real Rollouts

As outlined in Sec. 2.1, the agent's policy utilizes an actor-critic framework, with the critic predicting the value function $V(s)$ for each given state. Since the critic is trained in the imagination of the world model and will subsequently be used to train the actor, it is essential that its value estimates accurately reflect the agent's real rollout conditions. If the actual rollout performs poorly, a correct low-value estimate from the critic can guide the actor's updates in a direction that improves performance. However, in Fig. S2, we show that both Passive and Tandem agents consistently wrongly predict their value functions, assigning high values even when their actual trajectories yield low rewards. Throughout training, the value function estimation error for these offline agents remains significantly

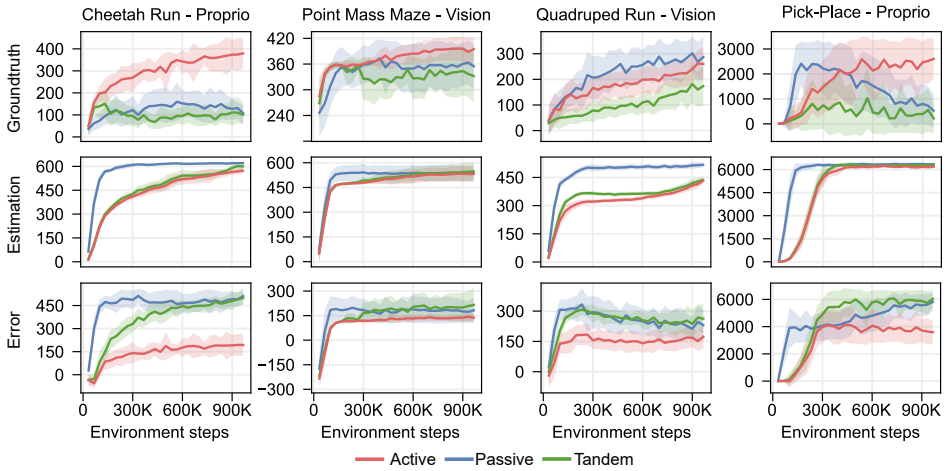

Figure S2: **Value function estimation of each agent in 4 selected tasks.** The value function $V(s)$ is calculated on the initial state of each agent's collected trajectory, which should reflect the actual discounted rewards accumulated across the trajectory. The ground truth value is computed using Monte Carlo estimation from one sample trajectory. The error is computed by subtracting the ground truth value from the estimated value.

higher than that of the Active agent, showing consistent statistical differences across time scales. This finding highlights that, without the self-correction mechanism, offline agents develop a pronounced mismatch between imagined and real rollouts, which is reflected in the discrepancy between estimated and ground-truth returns. This misalignment can lead to suboptimal actor updates in many tasks, ultimately resulting in unstable or degraded performance. Nevertheless, in certain tasks such as *Quadruped Run - Vision*, the Passive agent appears less affected by its value overestimation, suggesting that other task-specific factors may mitigate the impact of inaccurate value estimation.

Moreover, in some tasks such as *Pick-Place - Proprio*, the magnitude of the error is even larger than that of the ground truth, even for the Active agent. This observation raises the question of whether the standard practice of using value function error is the most ideal metric for quantifying the imagination gap. Since value functions are trained via bootstrapping, they are inherently biased and unlikely to converge exactly to Monte Carlo returns. A potentially better alternative could be to compare the average predicted reward (used in boostrapping to train the critic) with the actual reward collected during evaluation. We leave a detailed investigation of such alternatives to future work. The full results including more tasks can be found in Sec. B.5.5.

B.2 Per-step Analysis of Performance Degradation

Compromised policy causes the agent to drift into novel states from familiar regions. This is particularly evident in task (a) *Point Mass Maze - Vision* of Fig. S3. In the timesteps marked by the grey regions—just before entering novel states—the agent exhibits low world model loss, yet its movement direction deviates from the typical task-solving trajectory. This suggests that the policy is already compromised, even before encountering novel states, matching the argument in Sec. 3.3. However, in the locomotion environment (b) *Cheetah Run - Vision*, this is less observable, which could be attributed to the nature of the task—where it is more difficult to visually identify subtle changes in gait compared to tracking the trajectory of a ball.

Catastrophic cycle: novel states disrupt world model and policy output during evaluation. After the agent enters into novel states, the world model will output inaccurate estimations and latent embeddings. Since the policy network relies on these inaccurate latent states as input, this can start the catastrophic cycle where each compromised action leads to further novel states and additional inaccuracies until the episode ends or the agent accidentally re-enters into a familiar state. In Fig. S3, we provide for two test times trajectories the reward, world model loss, and policy reconstruction loss across two tasks. A low task reward is typically accompanied by a high world model loss. A high

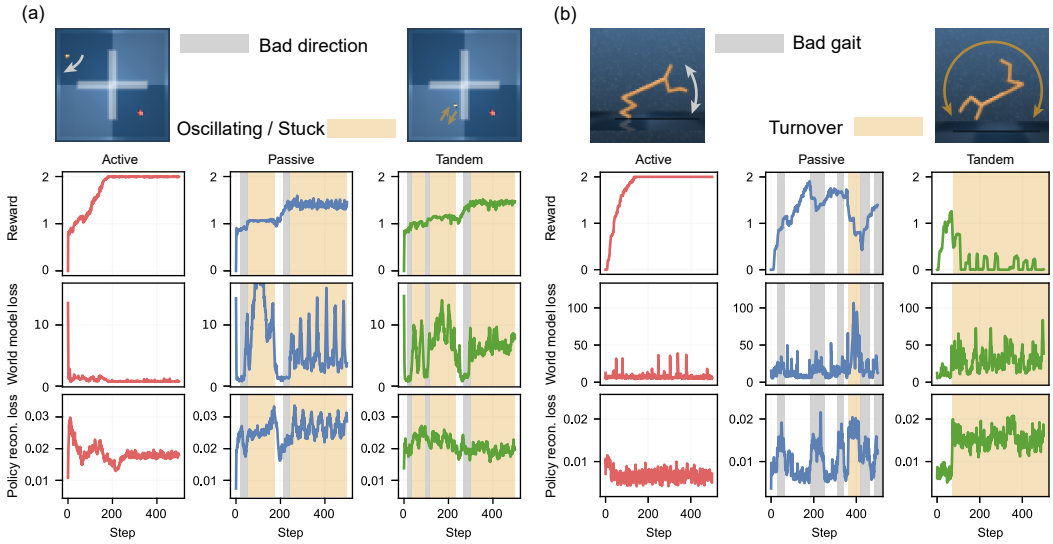

Figure S3: **Stepwise analysis within a single test episode** of the Point Mass Maze - Vision and Cheetah Run - Vision tasks from DMC. The plots show the progression of reward, world model loss, and policy input reconstruction loss at each step as the agent executes actions given by its own policy. Timesteps, where agents exhibit abnormal behavior, are highlighted with yellow and grey regions. Each episode consists of 500 steps, with the environments initialized identically across agents. The agents are the fully trained version after 1M environment steps.

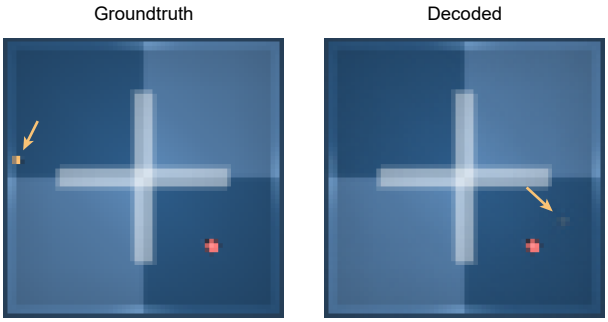

Figure S4: **World model misinterprets the novel states.** In the decoded image (step 84 in Fig. S3) from the world model of the Passive agent in task Point Mass Maze - Vision, the ball vaguely appears near the goal position while in the ground truth observation, it is actually in a novel region to the world model.

world model loss can correlate with a high policy input reconstruction loss, suggesting that the policy is unfamiliar with such inputs and produces compromised actions—an indication of the catastrophic cycle. In both tasks, the catastrophic cycle is evident in the significant failure periods (e.g. oscillation in *Point Mass Maze - Vision* or turnover in *Cheetah Run - Vision*). In addition, the accidental re-entering into a familiar state can also be observed between the neighboring failure periods. However, this accidental re-entering cannot save the agent since its policy is already compromised due to the lack of self-correction mechanism and cannot consistently output a reliable action in familiar states.

World model can sometimes hallucinate and mislead policy in novel states. We observe unexpected instances where the world model hallucinates, as shown in Fig. S4. The decoded image by the world model shows the agent has already reached a position near the goal, while, in fact, it is still far away from the target. It indicates that the world model can hallucinate in the novel states and produce an incorrect mapping of the latent state, misleading the policy to output inadequate actions.

Sometimes, policy input reconstruction loss fails to reflect performance degradation. This is observed in the Tandem agent in *Point Mass Maze - Vision* in Fig. S3. In novel states, a high

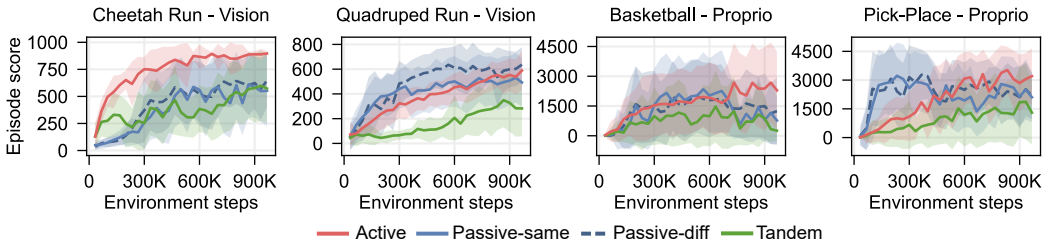

Figure S5: **Model initialization matters not in Passive agents.** Performance comparison when initializing the world model and policy network of Passive agents with the same and different seed w.r.t. the Active agents.

world model loss may still result in a relatively low policy input reconstruction loss, possibly due to erroneous latent state mapping produced by the world model.

B.3 Detailed Results of Considerations in Practical Applications

Advantage of training agents offline Although the performance degradation caused by the OOD issue is prominent in Passive agents, they show potential for faster convergence and more efficient training, as seen in tasks like *Quadruped Run - Vision* and *Pick-Place - Proprio* in Fig. 3. This is because Passive agents have access to high-quality trajectories from the beginning, while Active agents must wait until later in training to encounter those trajectories. We validate this hypothesis in Fig. S7, where Passive agents trained on suboptimal data generally perform worse than those trained on mixed data. It indicates that mixing expert trajectories into suboptimal data helps the performance, which matches the case between the Active (suboptimal data) vs. Passive (mixed data) agent in the early training stage. Therefore, addressing the OOD issue in Passive agents is crucial, as solving it could unlock the potential for highly efficient agent training. However, we do not observe such advantages in Tandem agents.

Different model initialization In this section, we answer the question whether the model initialization affects the performance degradation. In particular, if we initialize the world model and policy network of a Passive agent using the same seed as the Active one, will the performance differ from the independently initialized Passive agent? In Fig. S5, we show that no significant difference in the task performance can be observed with initialization seeds among Passive agents. We also investigate the sensitivity of task performance to the initialization of weights in model networks of Tandem agents. By mixing weights of the identically initialized networks as the Active and those of an independent initialization with different ratios α, it allows us to observe whether a tiny difference in the initialization will cause a big difference in task performance.

$$w \doteq (1-\alpha) \cdot w_{\text{Active}} + \alpha \cdot w_{\text{Tandem}} \tag{S3}$$

In Fig. S6, we observe that even a small deviation from the weights of the Active agent eventually causes a large difference in task performance when training on the identical sequence of training batches each training step.

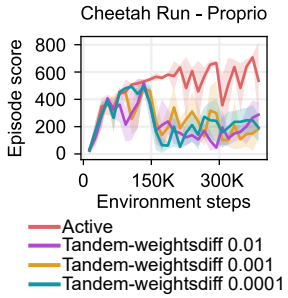

Figure S6: **Performance comparison of the world model and policy network of Tandem agents initialized with mixed weights.** Results shown for different α values (indicated in run name) as defined in Eq. (S3). Results for one seed.

World model overfitting on expert dataset Another popular practice to facilitate training a capable agent is to train the agent on an expert dataset (Kumar et al., 2022). However, in Fig. S7, we find that training on expert data leads to an even worse performance degradation in Passive agents. It is also indicated by the high world model loss with a growing tendency. However, according to the performance of Passive-mixed agents, mixing expert data with suboptimal trajectories can help mitigate this

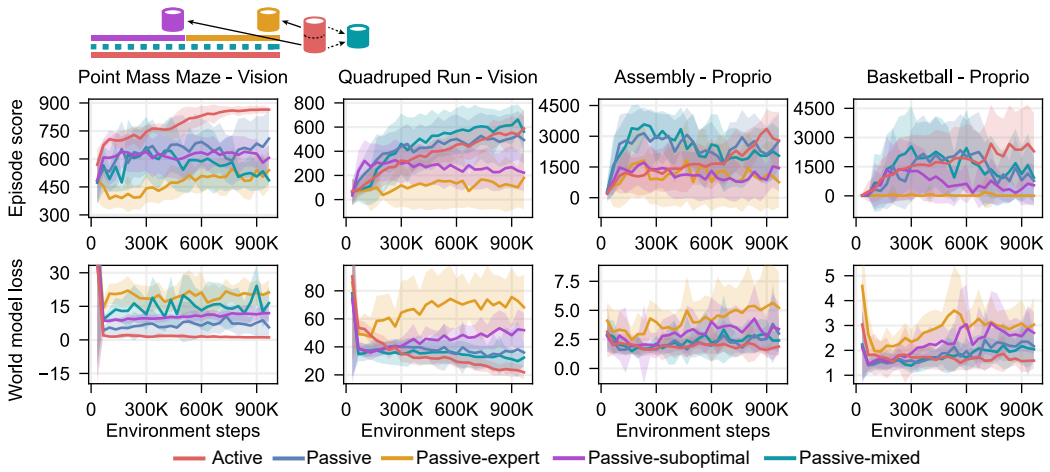

Figure S7: **Performance comparison when training Passive agents on different halves of the replay buffer from the Active.** We split the replay buffer (red bucket) at the 500K environment steps, as shown in the schematic illustration on the Point Mass Maze - Vision. The first half (purple bucket) represents the suboptimal data, while the second half (yellow bucket) mainly contains high-reward expert data. Therefore, Passive-expert, Passive-suboptimal, and Passive-mixed have a halved replay buffer compared to the normal Passive agent. The replay buffer of the mixed agent (turquoise bucket) is uniformly sampled from the whole replay buffer.

issue. The expert dataset primarily consists of monotonic task-solving trajectories, which implies extremely limited state-space coverage. Incorporating suboptimal data expands this coverage during training. It can improve the generalization capability of the world model and reduce the OOD risk during policy rollouts in evaluation. This highlights the importance of broad state-space coverage during training and the need to include exploration-equivalent data to ensure a capable agent. This finding matches results from previous research (Gulcehre et al., 2021; Mediratta et al., 2024; Suau et al., 2024).

World model overfitting on low-dimensional inputs In the *Basketball - Proprio* and *Pick-Place - Proprio* tasks, the performance of the Passive agent declines as the world model loss increases in the second half of the training process. A similar issue is observed in proprioceptive versions of DMC tasks in Appendix B.5.2. It indicates that the world model begins to overfit on the fixed data distribution in the replay buffer, given that the Passive agent is not allowed to add its own interaction data and cannot change the data distribution progressively in the same way as the Active agent. This tendency is more pronounced in the proprioceptive version, likely because the lower input dimension of the world model, compared to image-based observations, makes it more susceptible to overfitting when combined with high model capacity.

B.4 Further Insights on Self-Correction Mechanism

Here, we present additional perspectives on the self-correction mechanism. Further experiments are expected to strengthen the arguments, and we leave them to future work.

Copying data from an online agent is not self-correction In Fig. S9, we observe that the Tandem agents generally suffer from performance degradation w.r.t. the Active agents. They tend to visit OOD states more often because the policy/actor training suffers from misleading critic prediction as shown in Fig. S17. Although the Tandem agents share the exact same training data stream (for the world model training) as the Active agents, they do not benefit from the self-correction in the online agents. Therefore, simply copying the step-wise training data from an online agent is not sufficient for self-correction. Since the Tandem agent begin with different weights and therefore a different policy from the one in the Active agent, it might need different set of training data for effective self-correction.

Indirect self-correction via shared world models and the need for ongoing self-correction From the results in Fig. 4 as discussed in Sec. 3.5, we identify that discrepancies in both world model and policy contribute to performance degradation, with their relative impacts depending on the task. However, from a different perspective, this can be understood as: by using the same world model as the online agent, the Tandem_sameWM agents can improve their performance w.r.t. the original Tandem agents to varying degrees. This suggests that the self-correction in an online-learning agent can indirectly benefit a different policy in another agent via a shared world model. In other words, even if the task policy itself does not engage in the process of self-correction, an online-trained world model—being generally more accurate than its offline-trained counterpart—can still improve task performance in a tandem setting. This could possibly be viewed as an indirect self-correction effect. By contrast, the Passive_sameWM(frozen) agent shows no such benefit, likely because its world model is fixed during training, with no online feedback loop. This may also indicate the importance of an ongoing self-correction process, as in an online-learning agent. If the self-correction stops, then the agent may begin to suffer from policy exploiting the generalization error in the world model, ultimately harming performance. Additional experiments e.g., stopping the online feedback process in the middle of training, could further validate this argument, and we leave such investigations to future work.

Self-correction in exploration setting In Sec. 4.1, we compare online (Active) and offline (Passive, Tandem) agents in a pure exploration setting, where the Active agent collects interaction data via an online-trained exploration policy while training a task policy in parallel. Since data collection is driven by the exploration policy rather than the task policy, one might question whether the Active agent provides any self-correction for the task policy, and whether it serves as a fair baseline for measuring offline degradation. However, as noted earlier, the task policy may still benefit from self-correction indirectly via the shared world model with the exploration policy. We therefore retain it as a baseline, and additionally compare to the online task-oriented agent to support the argument that exploration data can counteract the lack of self-correction and help mitigate the OOD challenge during evaluation in offline agents.

B.5 Complete Results

B.5.1 Results of Agents with Different Exploration Bonus

In Fig. S8, we show all three analyzed tasks with comparison among different exploration bonus values. The optimal exploration bonus w_{expl} is 0.5 for task *Quadruped Run - Vision*, 0.9 for tasks *Point Mass Maze - Vision* and *Pick-Place - Proprio*.

B.5.2 Results of Task-oriented Agents

In Fig. S9 and Fig. S10, we show the complete results in 31 tasks corresponding to the discussion in Sec. 3.4 and Sec. 3.5. The Passive agent initialized using the same seed for the world model and policy network as the Active agent is marked with a suffix "-same", while the different model initialization is marked with "-diff".

B.5.3 Results of Adding Self-generated Data

In Fig. S11, Fig. S12, and Fig. S13, we show the complete results in 31 tasks, where we allow the Passive agents utilize the self-generated data from environmental interaction, corresponding to the discussion in Sec. 4.2. In Tab. S2, we show how many self-generated data is added to the replay buffer by Passive+Auto interact agents. The percentage is calculated using the number of additionally added steps divided by the total number of steps in the original replay buffer. Together with Fig. S14, we show that our adaptive agent **Passive+Auto interact** can converge fast and require minimal interaction data to recover the performance.

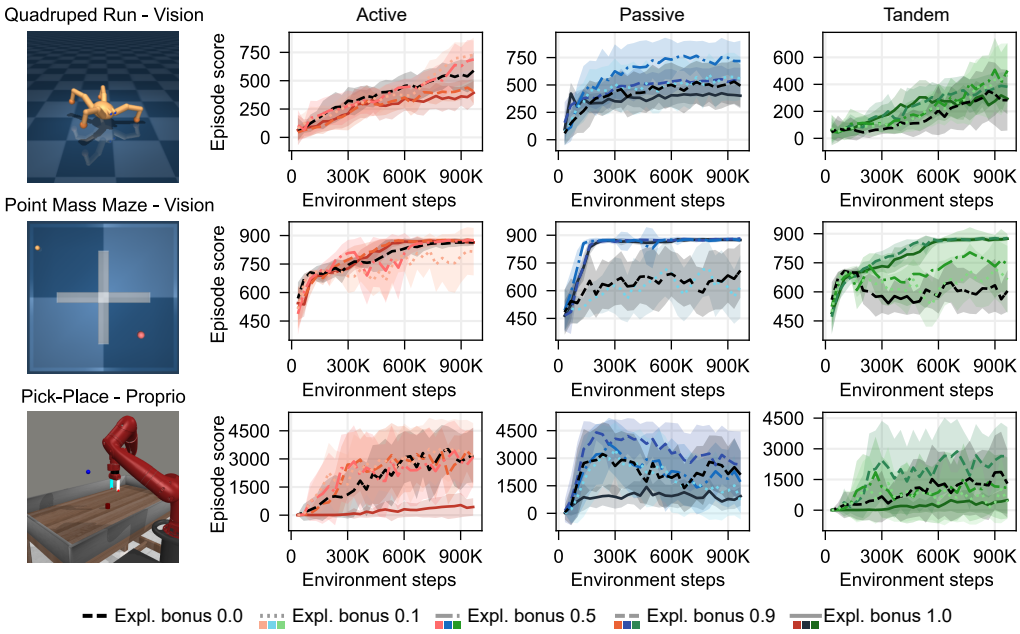

Figure S8: **Different task has different optimal exploration bonus values.** Performance comparison when assigning different exploration bonuses w_{expl} in the reward function. The black dashed lines represent pure task-oriented policy without any exploration bonus.

Table S2: Percentage of added self-generated data by Passive+Auto interact agents

Task	**Percentage (%)**	**Task**	**Percentage (%)**
cheetah_run-proprio	10.44%	walker_walk-proprio	18.27%
cheetah_run-vision	6.53%	walker_walk-vision	7.87%
cup_catch-proprio	0.67%	assembly-proprio	8.04%
cup_catch-vision	9.47%	basketball-proprio	7.16%
finger_turn_hard-proprio	2.53%	button-press-proprio	4.04%
finger_turn_hard-vision	3.47%	lever-pull-proprio	1.20%
hopper_hop-proprio	4.31%	peg-insert-side-proprio	2.31%
hopper_hop-vision	4.00%	pick-place-proprio	9.82%
humanoid_walk-proprio	17.78%	soccer-proprio	14.93%
humanoid_walk-vision	3.60%	window-open-proprio	1.47%
point_mass_maze-proprio	0.00%	asterix-vision	2.68%
point_mass_maze-vision	4.62%	breakout-vision	1.86%
quadruped_run-proprio	2.53%	freeway-vision	0.00%
quadruped_run-vision	2.93%	seaquest-vision	0.07%
reacher_hard-proprio	2.27%	spaceinvaders-vision	0.47%
reacher_hard-vision	20.31%	**Average**	**5.67%**

B.5.4 Results of Explorative Agents

In Fig. S15 and Fig. S16, we show the complete results in 31 tasks using agents with pure exploration rewards, corresponding to the discussion in Sec. 4.1. The Passive agent initialized using the same seed for the world model and policy network as the Active agent is marked with a suffix "-same", while the different model initialization is marked with "-diff".

B.5.5 Results of Value Function Estimation

In Fig. S17, we show the complete results in 7 tasks corresponding to the discussion in Sec. B.1. The Passive and Tandem agents are initialized using a different seed for the world model and policy network from the Active agent.

B.5.6 Outliers in Complete Results

Here, we briefly discuss a few task results that are unexpected or do not match the general trend and are not fully covered in the main body of the paper. A detailed analysis is beyond the scope of this work and is left for future investigation.

Among the results of agents trained on task-specific data in Sec. B.5.2 The world model loss of the Passive_same agent in *Freeway - Vision* shows a clear mismatch with its performance degradation w.r.t. its Active counterpart (i.e., low model loss despite low task performance). This discrepancy may be attributed to the characteristics of the MinAtar domain, where an episode terminates immediately upon a failure condition, rather than running for a fixed length as in DMC or MetaWorld. In the Freeway task, the agent must cross roads while avoiding fast-moving vehicles. Due to the high speed and density of the cars, a failure condition (i.e., being hit) can be easily triggered. As a result, once the agent enters OOD states, it may be immediately terminated before the world model loss can accumulate. This early termination suppresses the average world model loss, making it appear low despite poor performance.

Among the results of agents trained on pure exploration data in Sec. B.5.4 The Passive agent in *Reacher Hard - Vision* performs significantly worse than the Active counterpart even when trained on exploration data. Since this task only gives sparse rewards and the replay buffer is filled with trajectories irrelevant to the task, it is likely that the agent struggles to extract useful learning signals to perform effective policy training, especially in the early stage.

In tasks such as *Finger Turn Hard - Vision* and *Peg Insert Side - Proprio*, the Tandem agents show lower performance compared to the Active agents. However, the scores are still close to the variance range of the Active agents, suggesting that the difference may stem from seed-level randomness. Alternatively, it could reflect a slight benefit from self-correction in the Active agents—even under an exploration setting. Although the task policy is not directly involved in data collection (which is guided by an exploration policy), both policies share the same world model. Therefore, the self-correction effect driven by the exploration policy may still indirectly benefit the task policy through this shared model, with the extent depending on the task characteristics.

Besides, the world model loss of the Tandem agent in tasks *Finger Turn Hard - Vision* and of the Passive agent in *Reacher Hard - Proprio* is less indicative of their performance degradation compared to the Active agent (i.e., the model loss remains low despite low task performance). As mentioned in Sec. 4.1, the broad state-space coverage provided by pure exploration data improves the generalization of the world model, yielding low loss even in regions far from task-relevant trajectories. Therefore, even if performance degradation still happens due to a sparse-reward setting and insufficient task-related information in pure exploration data, its low-reward trajectory may still have low world model loss.

Among the results of agents adding self-generated data in Sec. B.5.3 In tasks such as *Hopper Hop - Proprio*, *Point Mass Maze - Proprio*, and *Walker Walk - Vision*, the Passive agents adding 1% interaction data (Passive+0.01 interact) exhibit large training fluctuations. In some cases, they even underperform compared to the Passive agents without any self-generated data. The fluctuations may stem from the sparse addition of self-interaction data: performance rises when new interaction data is added, but declines as the model begins to overfit on the static dataset during periods without new additions. A possible compounding factor for the worse performance is that the added self-generated trajectories—collected using a different policy—may lie outside the original training distribution.

Such trajectories may inject noise into the world model training by providing inconsistent training signals. Due to their small proportion (1%), they fail to offer meaningful self-correction and instead destabilize the learning process to varying degrees, depending on the task.

Among the results of value function estimation in Sec. B.1 In an earlier standalone offline test using a final checkpoint of the *Pick-Place - Proprio* task, we observed one mismatch between the value function error and the task performance of the Passive agent (i.e., low value prediction error despite a low task score compared to the Active counterpart). However, after rerunning the task with three random seeds and collecting statistics over time, the results aligned with our main claims. This suggests that the earlier observation was likely due to randomness or variance.

Among the results of suboptimal and mixed dataset in Sec. B.3 While mixing expert trajectories is generally beneficial for offline training, we observe an exception in *Point Mass Maze - Vision* in Fig. S7, where the Passive_mixed agent underperforms the Passive_suboptimal agent in the second half of training. A possible reason is that the Active agent learns to solve the task early (already by the end of the first half of the replay buffer), so the second half mostly contains repetitive, task-specific trajectories with little new information—as reflected in its consistently high performance and low variance. As a result, mixing samples from both halves may actually reduce the overall diversity compared to using only the first half, leading to narrower state-space coverage and increasing OOD risk. This is also consistent with the slightly higher world model loss observed in Passive_mixed compared to Passive_suboptimal.

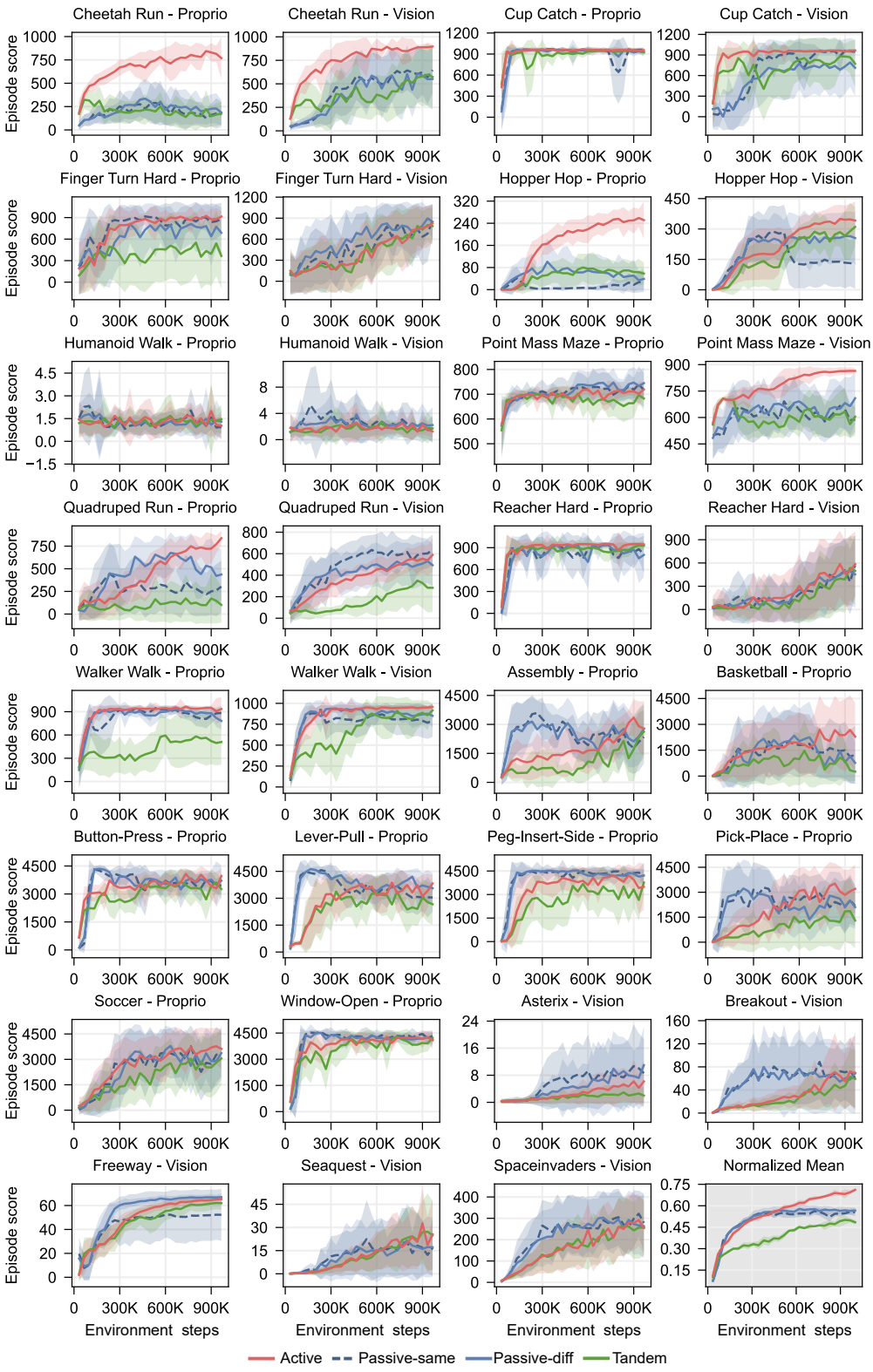

Figure S9: **Episode score of 31 tasks.** The first 18 tasks are from DMC, the subsequent 8 tasks are from Metaworld, and the last 5 are from the MinAtar domain. We also output a normalized mean score across tasks. The Passive-same is Passive agents initialized identically as the Active agents while Passive-diff is independently initialized.

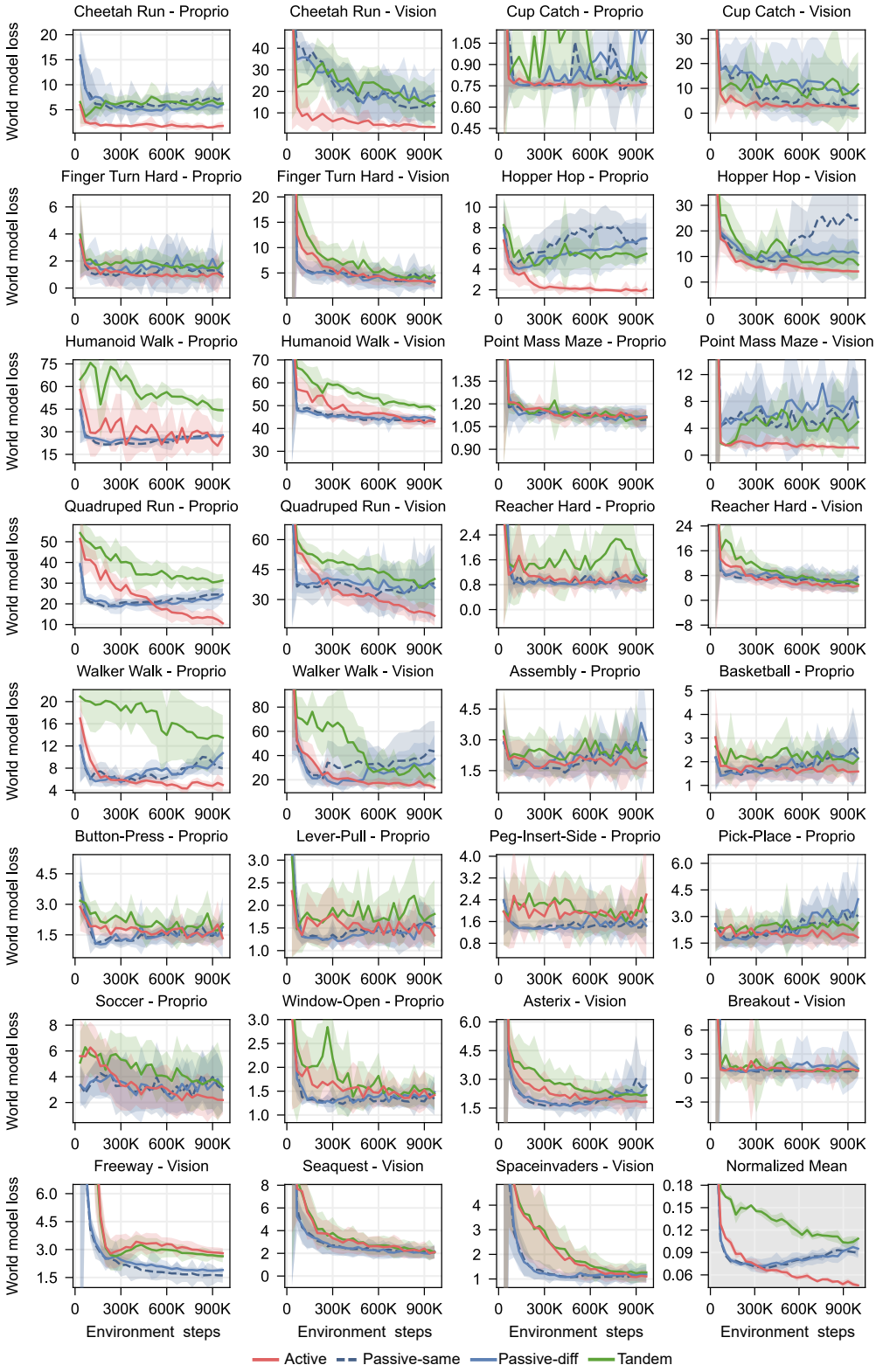

Figure S10: **World model loss of 31 tasks.** In the last subplot, we show an additional normalized mean result across tasks.

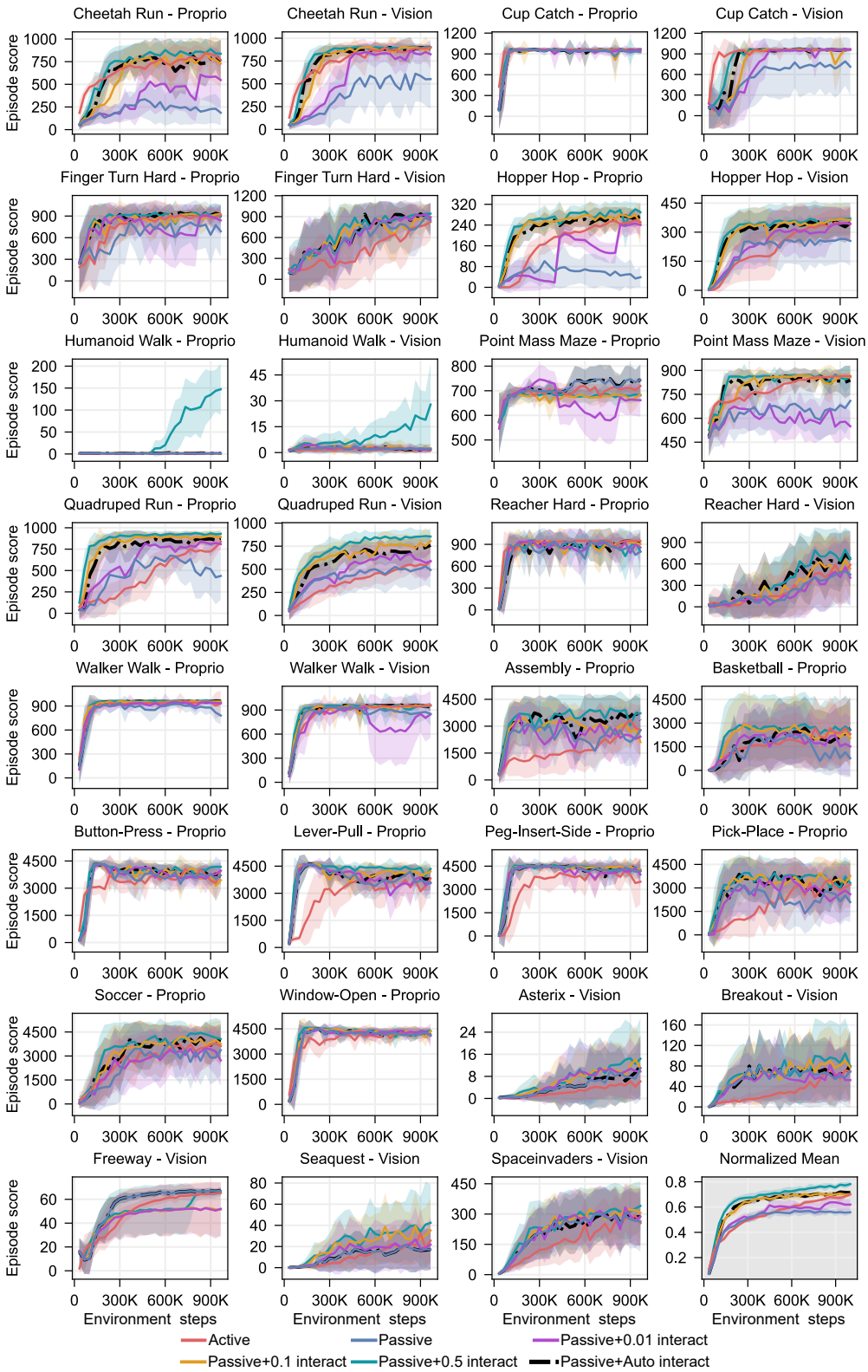

Figure S11: **Episode score of 31 tasks.** In the last subplot, we show an additional normalized mean result across tasks.

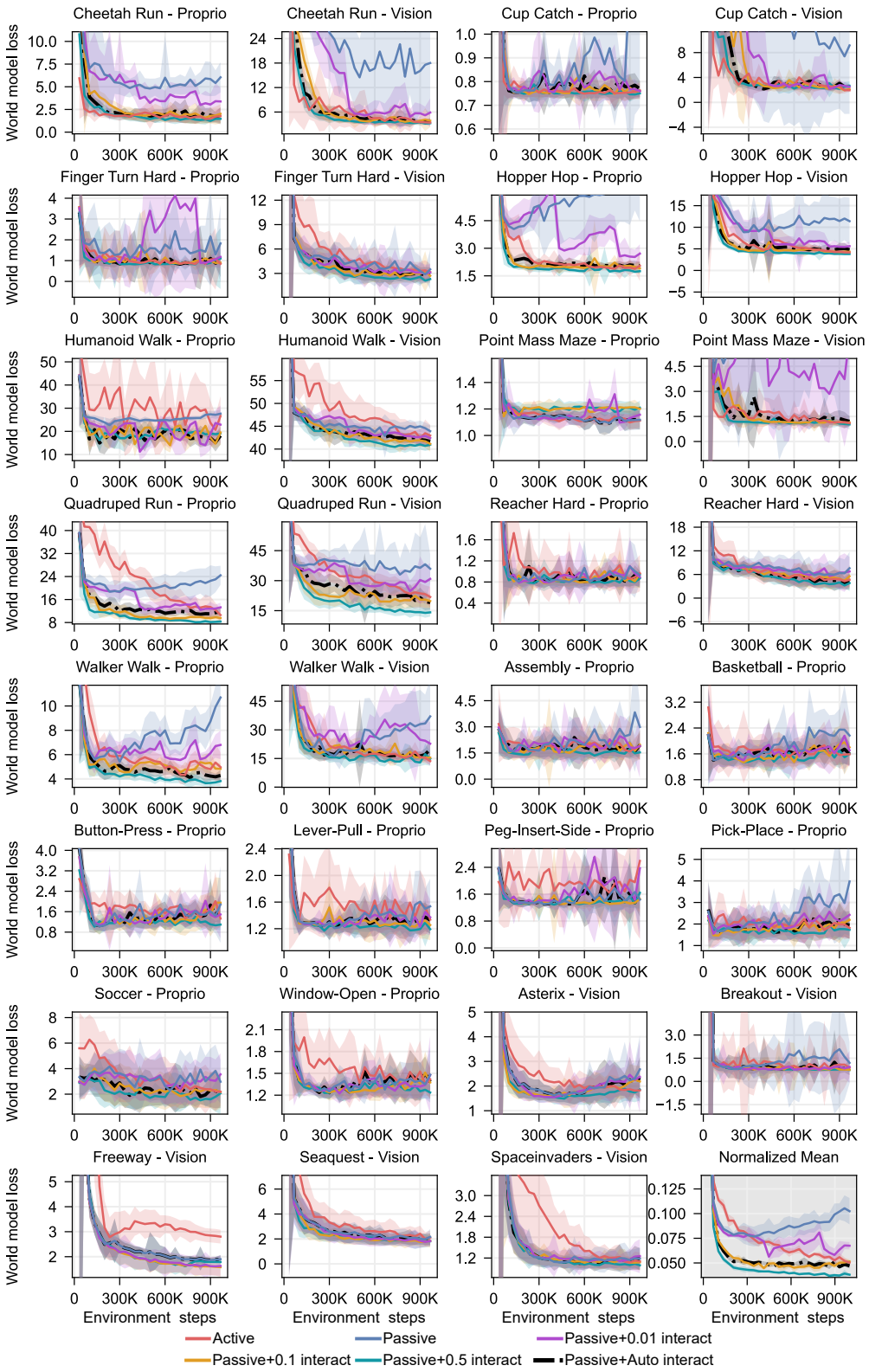

Figure S12: **World model loss of 31 tasks.** In the last subplot, we show an additional normalized mean result across tasks.

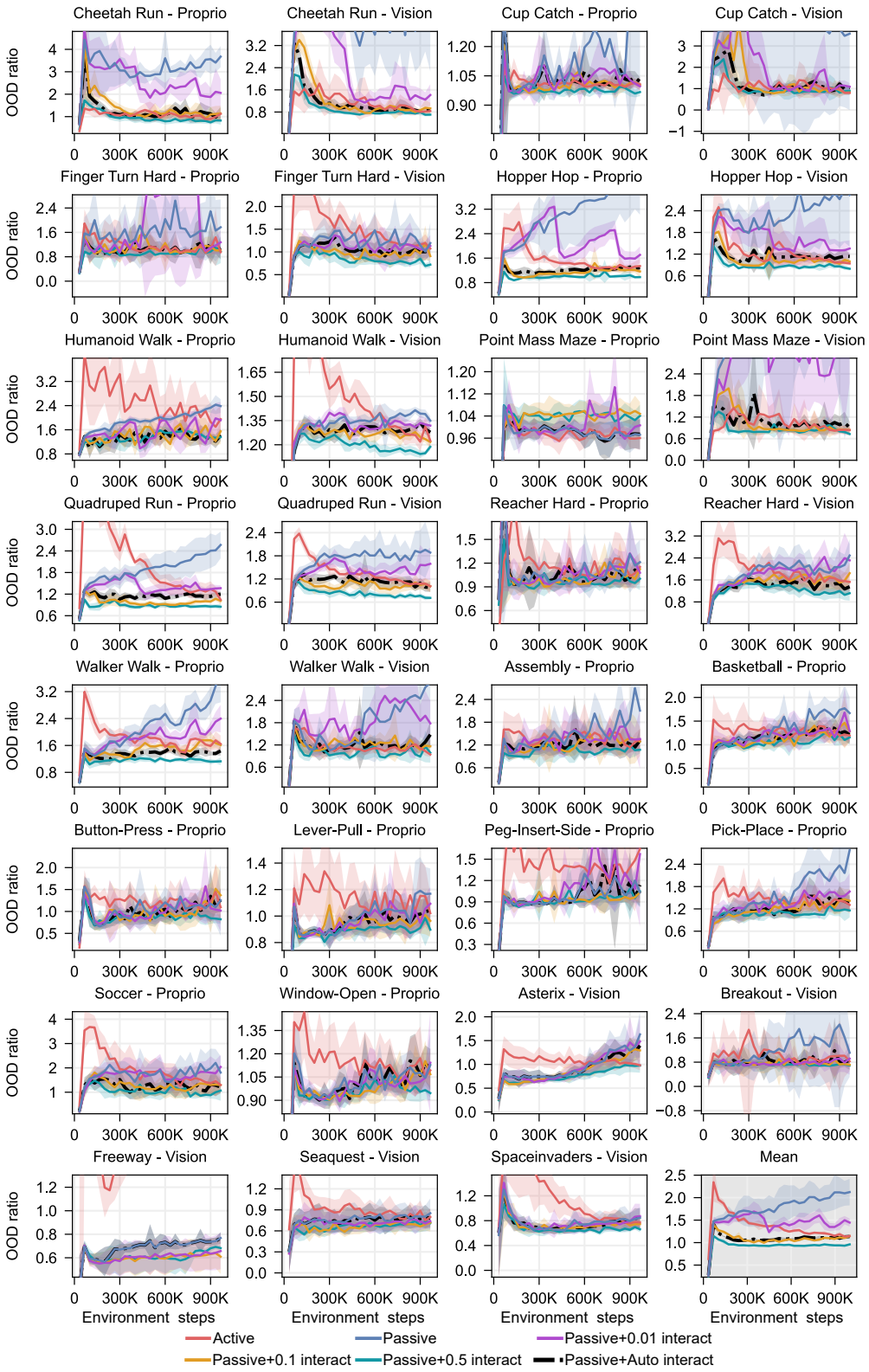

Figure S13: **OOD ratio of 31 tasks.** In the last subplot, we show an additional mean result across tasks.

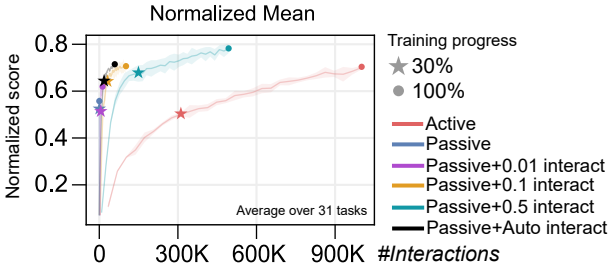

Figure S14: **Performance comparison between different Passive agents allowed environment interaction.** The y-axis is the average normalized episode score across 31 tasks. The x-axis shows how many self-generated interaction data are added to the replay buffer. Generally, an agent with markers closest to the top left corner is the best, having the highest score and requiring minimal self-generated interaction data.

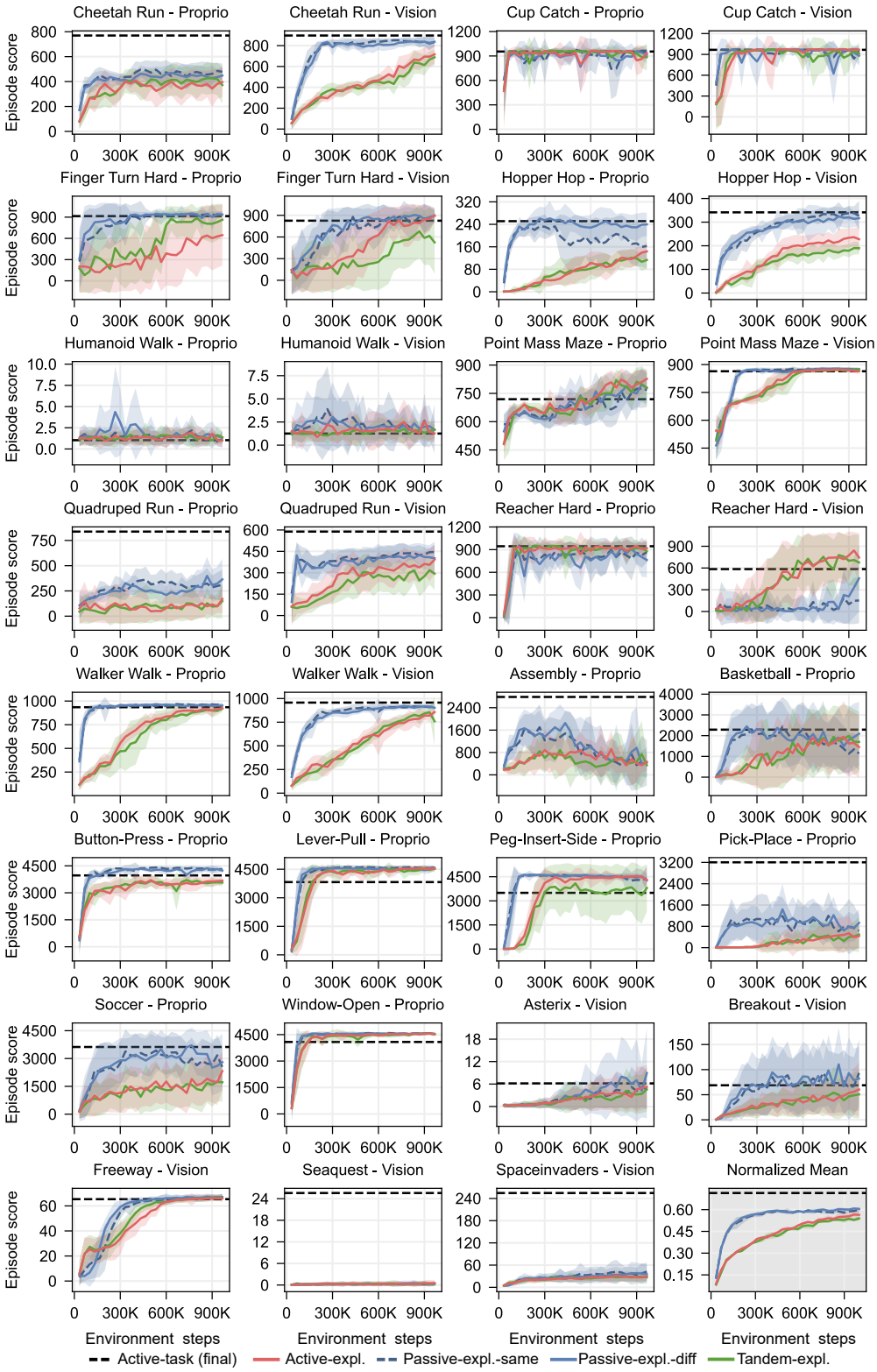

Figure S15: **Episode score of 31 tasks using agents with pure exploration rewards.** We also show the final performance of a task-oriented Active agent as the baseline in black dashed horizontal lines. In the last subplot, we show an additional normalized mean result across tasks.

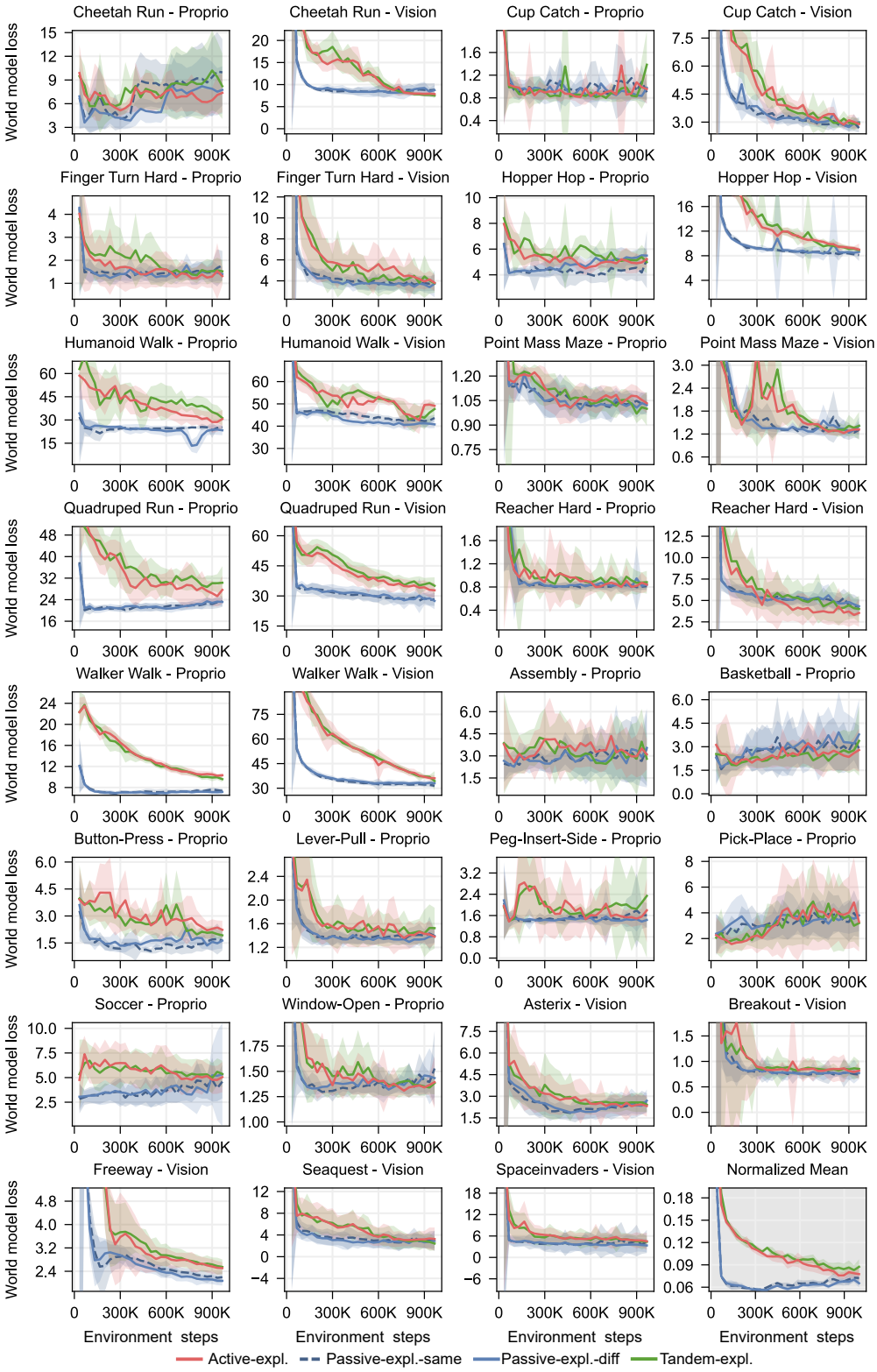

Figure S16: **World model loss of 31 tasks using agents with pure exploration rewards.** In the last subplot, we show an additional normalized mean result across tasks.

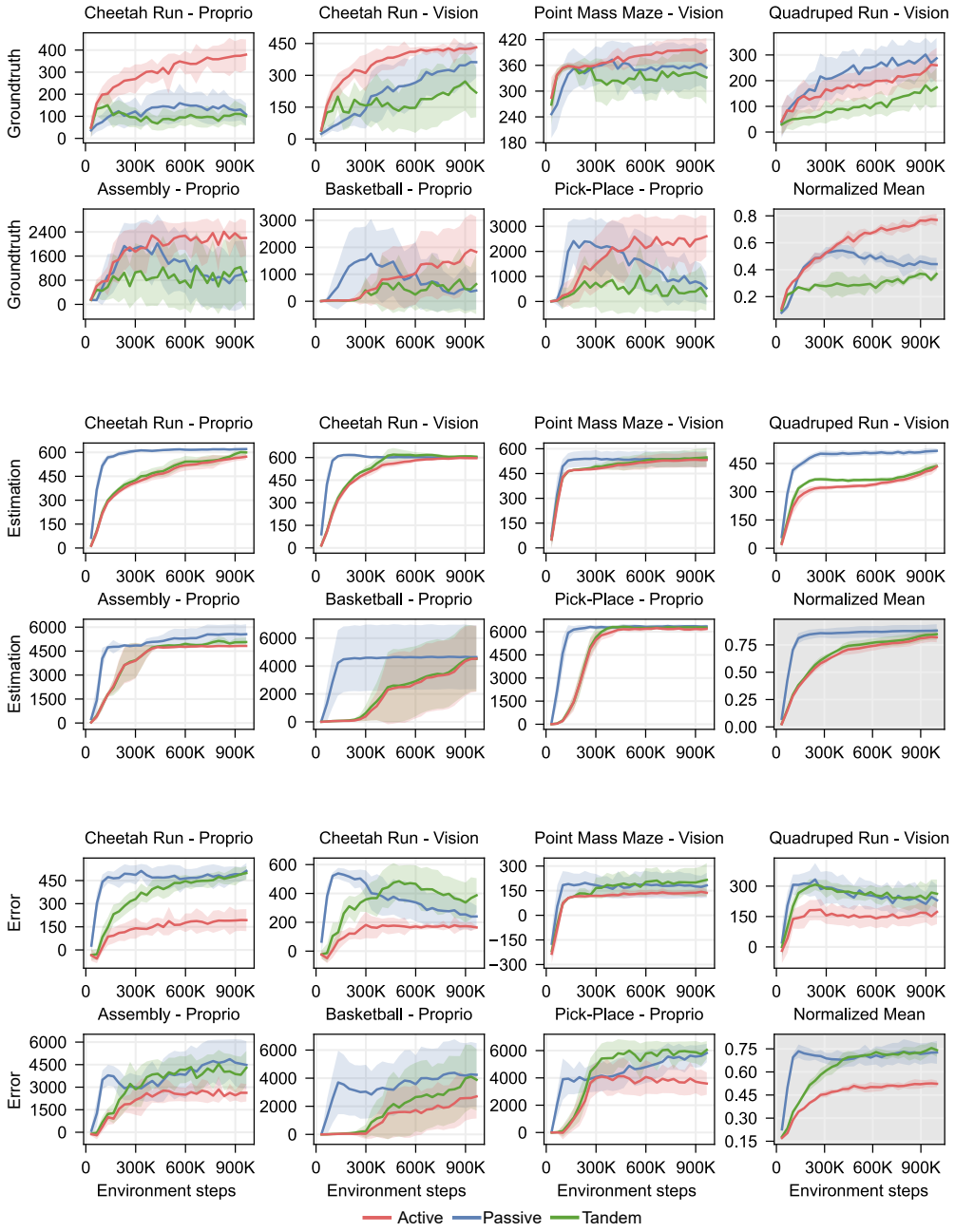

Figure S17: **Value function estimation of 7 tasks.** The value function $V(s)$ is calculated on the initial state of each agent's collected trajectory, which should reflect the actual discounted rewards accumulated across the trajectory. The ground truth value is computed using Monte Carlo estimation from one sample trajectory. The error is computed by subtracting the ground truth value from the estimated value. In the last subplot, we show an additional normalized mean result across tasks.

C Additional Discussions

C.1 Connection to the Original Tandem RL Paper (Ostrovski et al., 2021)

Motivation: addressing performance degradation in model-based offline training Our work is motivated by the performance degradation often observed when training model-based agents offline. While we draw inspiration from the experimental setup of the original Tandem RL paper (Ostrovski et al., 2021), their focus is on model-free methods (specifically DQN), and their analysis centers on the policy/value function. In contrast, we focus on a deeper and more intricate level of analysis—specifically, the interplay between the world model and the policy, which lies at the core of model-based RL. Our study complements the findings in Ostrovski et al. (2021) by extending the tandem framework to the model-based domain, where the sources of degradation are inherently different and less well-understood. Although such degradation and its solutions are sometimes assumed intuitively in the model-based RL community, it has lacked empirical backing—our work fills this gap with systematic and extensive experiments.

Experimental design: similar principles, adapted to a different focus Our experimental design is inspired by the controlled setup of Tandem agents proposed in Ostrovski et al. (2021), where training data stream is held constant to isolate the effect of other factors. We follow the same underlying principle of controlled experimentation to identify causal factors in performance degradation, e.g., by adopting a similar Tandem agent setup. However, we adapt this methodology to the model-based setting, designing new test cases – e.g. fix the same world model to examine whether degradation arises from world model learning alone (Sec. 3.5.1). Moreover, we conduct experiments across a broader range of environments beyond the original Atari focus, evaluating the generality of our findings in diverse model-based RL scenarios.

C.2 Contextualization in Similar Work

Most closely related studies on performance degradation in offline RL focus on model-free methods (Yarats et al., 2022; Ostrovski et al., 2021; Mediratta et al., 2024). Therefore, their conclusions should not be assumed to directly transfer to model-based settings, where the interaction between the world model and policy can introduce different failure dynamics.

Even in model-based works (Ross & Bagnell, 2012), the analysis remains limited—e.g. it could benefit from a deeper investigation into the respective roles of the world model and the policy, as well as from broader evaluation beyond a single-domain setting. Other solution-oriented works like Kidambi et al. (2020) mainly propose algorithmic solutions rather than understanding degradation from a data-centric perspective.

Our goal is not algorithmic novelty, but to fill this analytical gap through controlled and extensive experiments, offering a clearer understanding of degradation in model-based offline RL and identifying what kinds of data can help mitigate it.