CONTINUAL REINFORCEMENT LEARNING VIA BEHAVIORAL FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual reinforcement learning (CRL) requires agents that adapt quickly to new tasks, reuse past knowledge, and avoid forgetting. Existing methods address parts of this trade-off but rely on restrictive assumptions, such as slow gradient-based adaptation or known task boundaries. We ask whether Behavioral Foundation Models (BFMs), a recent class of zero-shot RL methods, can serve as a backbone for CRL. BFMs are appealing: switching to a new task-conditioned policy does not require task-specific gradient updates, the unsupervised objective is reward-agnostic, and the induced policy family may mitigate forgetting if coverage is maintained. However, BFMs are designed for offline settings and fail when applied naively online: zero-shot inference assumes a reward model, and online updates on a single task cause coverage collapse. To enable BFMs as continual online agents, we introduce a generic three-phase pipeline: reward-free offline pre-training; reward-model-free online task inference, which switches policies without updating their parameters; and continual updates through a coverage-preserving buffer. We evaluate continual sequences from DM Control and Continual World to characterize where BFMs succeed, where pre-training coverage becomes a bottleneck, and how each component contributes to stability and adaptation.

1 INTRODUCTION

Humans naturally acquire skills and reuse learned behaviors to adapt quickly to new tasks without forgetting key skills, such as walking or riding a bike. Developing artificial agents with similar capabilities is crucial for open-ended, real-world settings where downstream tasks change and agents must adapt quickly. Yet the dominant paradigm remains training to convergence on a single task and deploying the frozen agent. This limits scalability: each task requires a manually crafted agent, hindering transfer and reuse. Continual reinforcement learning aims to address this gap by designing agents that learn new behaviors, transfer knowledge, and adapt sequentially. In CRL, these desiderata form the stability-plasticity trade-off: agents should use prior knowledge to solve tasks and learn new ones without forgetting. Existing techniques often rely on restrictive assumptions. For instance, gradient-based continual adaptation methods (Anand & Precup, 2023) mitigate forgetting in many settings but introduce two trade-offs: adaptation requires gradient updates, slowing response to new tasks, and policy updates can cause forgetting, which must be addressed via regularization, replay, or structural separation. Latent-based methods, such as RL2 Duan et al. (2016), PEARL Rakelly et al. (2019), and VariBAD Zintgraf et al. (2019), learn an embedding of the current task and condition the policy on it. However, they generalize mainly within the meta-training task distribution, making them less robust to CRL's non-stationary rewards.

Another line of work incrementally builds policies for new tasks or reuses previous ones Gaya et al. (2022). This mitigates forgetting but requires downstream task knowledge and task-specific policy training. Thus, quick adaptation, resilience to forgetting, and learning new behaviors remain hard to reconcile, yet are necessary for open-ended, non-stationary tasks Javed & Sutton (2024).

We investigate Behavioral Foundation Models (BFMs) as a backbone for continual reinforcement learning. BFMs (Touati & Ollivier, 2021; Touati et al., 2022; Agarwal et al., 2024; Jajoo et al., 2026) are zero-shot reinforcement learning methods pre-trained with reward-free data, usually collected by coverage-maximizing techniques (Eysenbach et al., 2018; Burda et al., 2018; Liu & Abbeel, 2021). Given a downstream task, they return an approximately optimal policy without additional

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

gradient steps, provided pre-training covers the relevant behaviors (Eysenbach et al., 2021). The properties of BFMs are appealing for continual reinforcement learning: (i) adaptation to a new task is fast because it only requires switching the latent-conditioned policy, without retraining policy parameters, (ii) the unsupervised pre-training objective is reward-agnostic and could, in principle, be robust to non-stationary rewards (Zahavy et al., 2021), and (iii) using an implicit policy family might reduce forgetting since BFMs retain behaviors spanning a broad class of rewards. Whether these properties hold in practice, during online updates under non-stationary rewards, crucially depends on maintaining *coverage*, a question that is underinvestigated in the literature. BFMs are currently designed and evaluated for the offline setting, and using them naively in a sequential, online scenario fails (Sun et al., 2025): zero-shot policy inference procedures require a reward model, online updates with data originating from a specific task cause coverage collapse, and there is currently no system to integrate online experience without degrading the online representation.

Contribution. Our central contribution is a generic interface that turns offline-pretrained BFMs into continual online agents. First, an offline pre-training phase provides a warm-start representation from reward-free data. Second, we use online task inference (Sikchi et al., 2025; Rupf et al., 2025) to retrieve a meaningful policy from online interactions alone, without access to a reward model. Third, to integrate new experience while avoiding coverage collapse, we introduce a *coverage buffer* that combines offline rehearsal with filtered online transitions. Together, these components separate fast task adaptation from model optimization: switching tasks only requires inferring a new latent, while the BFM continually learns from less-frequent gradient updates using the coverage buffer. We evaluate this pipeline across custom continual sequences of DM Control tasks and Continual World, characterizing where it works, where it does not, and how its components interact. Our goal is not to claim that BFMs are superior to existing CRL baselines, but to provide a systematic study of BFMs as continual learners and a framework for keeping them stable under online, non-stationary updates.

2 Related Work

With successor-feature based Behavioral Foundation Models (BFMs) we refer to reward-free pre-trained agents that learn a latent-conditioned family of behaviors and retrieve policies for downstream rewards without task-specific training or planning (Sikchi et al., 2025). Successor features and USFAs introduced reusable representations that separate shared dynamics from task-specific rewards and support transfer through generalized policy improvement (Barreto et al., 2017; Borsa et al., 2018; Zahavy et al., 2021). More recent methods learn broader policy representations directly: FB (Touati & Ollivier, 2021; Touati et al., 2022) factorizes policy successor measures, PSM Agarwal et al. (2024) constructs an affine basis of feasible occupancies, and RLDP Jajoo et al. (2026) learns reward-free features through latent dynamics prediction. These approaches target zero-shot transfer under fixed dynamics, but do not study continual adaptation online with non-stationary rewards while updating the representation and preserving behavioral coverage.

Online extensions of BFMs address either reward-free pre-training or downstream adaptation. On the pre-training side, these methods aim to improve the quality of the learned BFM through online exploration. FBEE (Urpí et al., 2025) uses ensemble disagreement over FB value predictions to guide data collection, while DVFB (Sun et al., 2025) augments FB with a separate exploration value function trained using a contrastive-entropy intrinsic reward. On the adaptation side, ReLA and LoLA improve an initial zero-shot policy through online optimization in the BFM latent space, whereas OpTI-BFM uses optimistic task inference to identify a suitable latent policy directly from online rewards, without requiring a reward model (Sikchi et al., 2025; Rupf et al., 2025). However, these methods consider either online pre-training or adaptation to a single downstream task. They do not continually update the underlying BFM across sequential task changes while preserving its behavioral coverage.

Continual RL includes methods based on knowledge retention, modular policies, and task inference; we refer to Khetarpal et al. (2022) for a broader review. Distillation and rehearsal methods, such as Progress & Compress, TAPD, and ClonEx-SAC, preserve previous behaviors through knowledge consolidation or actor cloning (Schwarz et al.; Hafez & Erekmen, 2024; Wolczyk et al., 2022). Modular approaches such as CSP and CompoNet remove forgetting by keeping an expanding or compositional policy family, but relying on task-specific training (Gaya et al., 2022; Malagon et al., 2025). Latent-context methods infer the current task from interaction histories: VariBAD performs

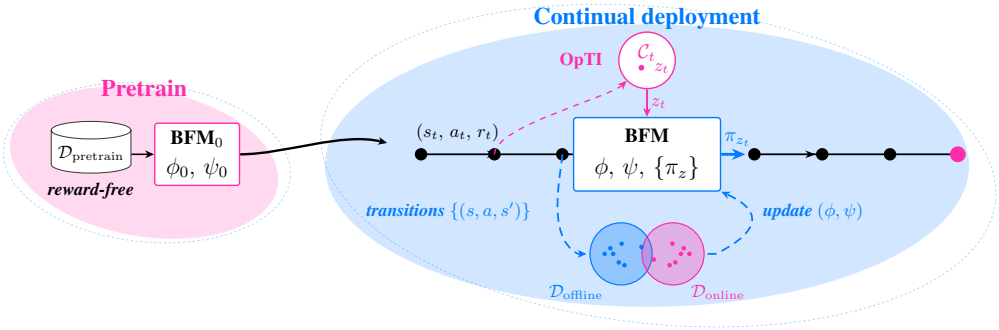


Figure 1: Three-phase pipeline for continual BFMs: offline pre-training, online task inference with OpTI, execution of π_{z_t}, and representation updates from a coverage-preserving filtered buffer.

explicit latent task inference, while 3RL combines a recurrent policy with experience replay for task-agnostic continual adaptation (Zintgraf et al., 2019; Caccia et al., 2023). PT-DQN instead separates slowly accumulated permanent knowledge from a transient value function that rapidly tracks the current task (Anand & Precup, 2023). BFMs differ by representing multiple behaviors via latent-conditioned policy family, enabling policy retrieval without training a new task-specific policy, although continual updates still require preserving behavioral coverage.

3 Preliminaries

Continual Reinforcement Learning We consider a continual reinforcement learning (CRL) problem, where the dynamics are stationary, while the reward changes at discrete task boundaries (Khetarpal et al., 2022; Abel et al., 2023). Formally, the environment is an MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R(t) \rangle$, with stationary dynamics $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ and a reward function $R(t) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, that is piecewise stationary in time: there exists a sequence of points $t_0 = 0 < t_1, t_2, \ldots$ such that in each interval $[t_k, t_{k+1})$ the reward is stationary $R_k : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. We refer to such an interval as *task* k. The agent observes s_t and r_t at each step, acts according to a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, and is agnostic to *when* the task changes or *which* task is active. Within a single task k, the standard objective maximized by the agent is the discounted return $v_k^\pi(s) = \mathbb{E}_\pi \left[\sum_{i=0}^{\infty} \gamma^i R_k(s_{t+i}, a_{t+i}) \right]$. Due to non-stationary rewards, a globally optimal policy might not exist; instead, learning is an endless adaptation where the agent continuously searches for better behavior rather than converging to a fixed one (Abel et al., 2023).

Successor Measures and Successor Features. For a policy π we write $Q^\pi = M^\pi r$ where M^π is the discounted occupancy measure over future state-action pairs, formally $M^\pi(s_0, a_0, X) = \sum_{t=0}^{\infty} \gamma^t \Pr((s_t, a_t) \in X \mid s_0, a_0, \pi), \ \forall X \subseteq \mathcal{S} \times \mathcal{A}$. M^π encodes the discounted visitation frequency of state-action pairs under π. This decoupling is what makes zero-shot policy extraction possible. Outside the tabular case, M^π is intractable to represent. The *successor features* (SF) framework (Barreto et al., 2017; Borsa et al., 2018) enables learning an approximation by assuming the existence of a feature map $\phi(s) : \mathcal{S} \to \mathbb{R}^d$, such that rewards $r(s) = \phi(s)^\top w$ can be obtained by a linear composition of $\phi(s)$ and a task weight vector w. The successor features are defined as $\psi^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) \mid s_0 = s, a_0 = a \right]$. Given $\psi^\pi(s, a)$, and any w in the span of ϕ, the value function for π can be recovered as $Q^\pi(s, a) = \psi^\pi(s, a)^\top w$.

Behavioral Foundation Models BFMs, learn successor features for multiple policies, via three objects: a state representation ϕ, a *universal successor feature* $\psi : \mathcal{S} \times \mathcal{A} \times \mathcal{Z} \to \mathbb{R}^d$ (Borsa et al., 2018), where $\psi(s, a, z)$ is the successor feature of policy π_z, and a policy family $\{\pi_z\}_{z \in \mathcal{Z}}$ indexed by a latent task variable $z \in \mathcal{Z} \subset \mathbb{R}^d$. These three objects are jointly learned so that for each z, the policy π_z is optimal for the linear reward $r_z(s) = \phi(s)^\top z$:

$$\psi(s, a, z) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t \phi(s_{t+1}) \mid s, a, \pi_z \Big], \quad \pi_z(s) = \operatorname{argmax}_a \psi(s, a, z)^\top z. \tag{1}$$

Once a downstream task r is fixed, we extract an optimal policy zero-shot by identifying the z_r associated with that reward via linear regression of r onto ϕ. Practically, given a few labeled interactions $\mathcal{D} = \{s_i, r_i\}_{i=0}^{N}$, we recover the latent z_r corresponding to r as

$$\begin{aligned} z_r &= \text{argmin}_{z \in \mathcal{Z}} \, \mathbb{E}_{s \sim \mathcal{D}} \Big[\big(r(s) - \phi(s)^\top z \big)^2 \Big] \\ &= \mathbb{E}_{s \sim \mathcal{D}} \Big[\phi(s) \phi(s)^\top \Big]^{-1} \mathbb{E}_{s \sim \mathcal{D}} \Big[\phi(s) \, r(s) \Big]. \end{aligned} \tag{2}$$

Then π_{z_r} is the near-optimal policy that maximizes r, with value function $Q_r^{\pi_{z_r}}(s, a) \approx \psi(s, a, z_r)^\top z_r$. The latent z maps the policy π_z to the reward $r_z = \phi(s)^\top z$. This procedure extracts near-optimal policies even for unseen rewards (Touati & Ollivier, 2021). Zero-shot inference samples in-distribution transitions from the pre-training dataset $\mathcal{D} = \{s_i, r_i\}_{i=0}^{N}$, labels them with a reward model $\hat{r}$ for the current downstream task, and obtains the corresponding latent z_r via linear regression as in Eq. equation 2.

The three objects $(\phi, \psi, \{\pi_z\}_{z \in \mathcal{Z}})$ provide a common interface for zero-shot policy retrieval. Existing SF-based BFMs (Touati & Ollivier, 2021; Agarwal et al., 2024; Jajoo et al., 2026) share this interface but differ in each component's training objective. For example, Forward-Backward (FB) (Touati & Ollivier, 2021; Touati et al., 2022) learns ϕ through a B map that factorizes the successor measure. Proto-Successor Measure (PSM) (Agarwal et al., 2024) learns ϕ through an affine decomposition of policy occupancy, and RLDP (Jajoo et al., 2026) learns ϕ by latent dynamics prediction. We treat a BFM $(\phi, \psi, \{\pi_z\})$ as a black box, so our results are *agnostic* to its instantiation.

Coverage Intuitively, *coverage* is the set of tasks, or equivalently rewards, for which the BFM can return a near-optimal policy. Geometrically, each policy π induces a discounted state-action occupancy M^π, and the feasible occupancies under MDP dynamics P form a convex polytope $\mathcal{C}$ whose vertices are deterministic policies, each optimal for some reward (Eysenbach et al., 2021). Representing $\mathcal{C}$ is infeasible, so the BFM approximates a subset $\mathcal{C}_{\text{BFM}} = \{M^{\pi_z} : z \in \mathcal{Z}\} \subset \mathcal{C}$, corresponding to the realizable reward class $\mathcal{R}_\phi$, i.e., rewards spanned by ϕ. By *coverage maximization*, we mean training the BFM so $\mathcal{C}_{\text{BFM}}$ contains behaviors relevant to downstream tasks. Relevance is nontrivial and domain dependent; in practice, BFMs are trained on data from diverse policies, typically via coverage-maximizing exploration (Burda et al., 2018; Liu & Abbeel, 2021), so observed experience spans as many vertices of $\mathcal{C}$ as possible.

4 BFMs AS CONTINUAL LEARNERS

A BFM $(\phi, \psi, \{\pi_z\}_{z \in \mathcal{Z}})$ trained with reward-free data returns near-optimal policies for fixed downstream tasks and has properties appealing for continual RL: its unsupervised loss is reward-agnostic, task adaptation switches π_z by recomputing z without task-specific gradient updates, and the learned policy family $\{\pi_z\}_{z \in \mathcal{Z}}$ retains behaviors for the reward class spanned by ϕ, *structurally* mitigating forgetting. Phase 3 continually updates the shared representation with gradients on a slower timescale, separate from the critical path of switching policies when the reward changes. However, these properties hold only when training data cover the relevant behaviors. In a continual scenario with unknown task sequences, naively updating on the current task stream causes the training distribution to *collapse* to the current occupancy. Preserving coverage is crucial for online BFM updates.

We address this with the three-phase pipeline in Figure 1. *Phase 1* performs offline pre-training for a stable initialization. *Phase 2* replaces standard zero-shot inference, which assumes a reward model and in-distribution data, with online task inference (Rupf et al., 2025) that estimates a near-optimal z_t from interaction alone. *Phase 3* continually updates $(\phi, \psi, \{\pi_z\}_{z \in \mathcal{Z}})$ while preserving coverage through a buffer that mixes offline pre-training data with filtered online transitions. Our contribution is to connect these phases into a continual BFM learner and to introduce the *coverage buffer*, which mitigates coverage collapse during online updates. Phases 1 and 2 use existing techniques and are reviewed only for notation and completeness.

4.1 OFFLINE PRE-TRAINING

We first pre-train the BFM offline on reward-free transitions $\mathcal{D}_{\text{pretrain}} = \{s_i, a_i, s_i'\}$ collected to cover the reachable state-action space. The representations $(\phi, \psi, \{\pi_z\}_{z \in \mathcal{Z}})$ are trained with an

unsupervised loss $\mathcal{L}_{BFM}(\phi, \psi)$; our pipeline is agnostic to the particular loss. The resulting model $(\phi, \psi, \{\pi_z\}_{z\in\mathcal{Z}})$ defines the agent's representational *coverage*, equivalently the reward class $\mathcal{R}_\phi$ for which near-optimal policies can be retrieved zero-shot. The following phases use online experience to infer policies and continually update $(\phi, \psi, \{\pi_z\}_{z\in\mathcal{Z}})$.

4.2 Online task inference

In principle, the pre-trained model can return a near-optimal policy for any reward in $\mathcal{R}_\phi$. Standard zero-shot inference assumes a reward model $\hat{r}$ that labels a subset of the pre-training data, $\mathcal{D}_{\text{inf}} = \{s_i, \hat{r}(s_i)\}_{i=1}^n$, from which a latent z_r is inferred and executed via π_{z_r}; see Eq. equation 2 and Eq. equation 1. In a continual setting, however, the task and reward model are unknown, and online data may be out of distribution. We therefore use OpTI-BFM (Rupf et al., 2025), which casts online task inference as a linear bandit over $\mathcal{Z}$. At each step t, the agent observes s_t, receives reward r_t, and updates the least-squares task estimate

$$\hat{z}_t = V_t^{-1} \sum_{i=0}^{t} \phi(s_i) r_i, \quad \text{and} \quad V_t = \lambda I_d + \sum_{i=0}^{t} \phi(s_i)\phi(s_i)^\top, \tag{3}$$

with confidence set $\mathcal{C}_t = \left\{ z \in \mathbb{R}^d \mid \|z - \hat{z}_{t-1}\|_{V_{t-1}} \leq \beta_t \right\}$. Optimism then selects $z_t \in \operatorname{argmax}_{z\in\mathcal{C}_t} \max_{w\in\mathcal{C}_t} w^\top \psi(s_t, \pi_z(s_t), z)$. The embedding z_t is recomputed every n steps, and the agent follows π_{z_t} between updates. The collected pairs $(\phi(s_t), r_t)$ update $(\hat{z}_t, V_t)$ in Eq. equation 3. This phase produces embeddings $\{z_t\}$ and online transitions $\{(s_t, a_t, s'_t, r_t)\}$, which populate the *coverage buffer* introduced next.

4.3 Continual Updates of the BFM

At this stage, the BFM is no longer frozen and we update its parameters $(\phi, \psi, \{\pi_z\}_{z\in\mathcal{Z}})$ online, using transitions collected by acting in the environment and stored in $\mathcal{D}_{\text{online}}$. Since the BFM loss is unsupervised, in principle, we can update its parameters regardless of which task the agent is facing. However, in practice, naively updating the model online causes *coverage collapse*: the model $(\phi, \psi, \{\pi_z\}_{z\in\mathcal{Z}})$ relies on the empirical training distribution induced by $\mathcal{D}_{\text{pretrain}}$; if we continually update on data $\mathcal{D}_{\text{online}}$ coming from the on-policy distribution of the current task, we eventually degrade the representation of other tasks. In FB (Touati & Ollivier, 2021, Theorem 2), they make this explicit: z_r extraction in Eq. equation 2 and successor features in $\psi(s, a, z)$ in Eq. equation 1 are learned with respect to a measure ρ on the state space $\mathcal{S}$, which is typically the empirical training distribution. The distribution ρ regulates the feature covariance $G_\rho = \mathbb{E}_\rho[\phi(s)\phi(s)^\top]$ and reward moment $b_\rho = \mathbb{E}_\rho[\phi(s)r(s)]$, and hence the accuracy of $z_r = G_\rho^{-1} b_\rho$, as well as the accuracy of $\psi(s, a, z)$. If we update the model only on transitions collected from the current task k, then ρ will slowly concentrate towards the occupancy of π_{z_k}, z inference becomes inaccurate in directions of the ϕ space not visited by π_{z_k}, and ψ will no longer be accurate for policies conditioned on $z' \neq z_k$. Thus, keeping coverage during online updates reduces to maintaining a ρ whose support spans the ϕ directions relevant for the current downstream task.

Intuitively, to safely update the model, we can filter online interactions to retain transitions (s, a, s') that increase the diversity of represented ϕ directions, discarding those that appear redundant. However, this alone is not enough. First, if the agent faces the same task r_k over a long horizon, filtering cannot prevent ρ from concentrating on the single task occupancy induced by π_{z_k}. Second, even when $\mathcal{D}_{\text{online}}$ is diverse in feature space, the empirical distribution it induces may differ significantly from the pre-training distribution, causing representations to drift.

4.3.1 Coverage buffer.

The *coverage buffer* addresses these failure modes by maintaining two sources of data and mixing them on every update, similar in spirit to CLEAR (Rolnick et al., 2019). The first source is a fixed offline dataset $\mathcal{D}_{\text{offline}} \subseteq \mathcal{D}_{\text{pretrain}}$ sampled uniformly from the pre-training dataset and kept frozen during continual interactions. This split acts as a rehearsal mechanism, preserving the broad support of the pre-training distribution, preventing ρ from concentrating on the single task distribution, and providing a fixed source of in-distribution data to constrain the representations $(\phi, \psi, \{\pi_z\}_{z\in\mathcal{Z}})$ from drifting far from the pre-trained values. The second source is an evolving buffer $\mathcal{D}_{\text{online}}$ that *filters*

online transitions and favors ϕ directions underrepresented by the buffer (the score used for selection is introduced in the next section).

During each online update, a batch $\mathcal{B}$ of size B is drawn by combining samples from the online and offline buffers according to a mixing ratio $\alpha \in [0, 1]$, which defines the offline-online ratio as follows

$$\mathcal{B} = \{(s, a, s') \sim \mathcal{D}_{\text{online}}\}_{i=1}^{\alpha B} \cup \{(s, a, s') \sim \mathcal{D}_{\text{offline}}\}_{i=1}^{(1-\alpha)B}. \tag{4}$$

At each update step, the model $(\phi, \psi\,\{\pi_z\}_{z\in\mathcal{Z}})$ is updated with the batch $\mathcal{B}$, incorporating newly collected online transitions while keeping a stable representation via the broad offline dataset. Intuitively, the parameter α controls a stability-plasticity trade-off: lower α preserves pre-training coverage, higher α accelerates learning of newly collected ϕ directions. We ablate the effect of α in Appendix E.1. The desired effect is both to expand the known behavior when pre-training exhibits poor coverage, and condition the model on the current task without forgetting. We study this aspect in Appendix E.2, showing the benefit of continual updates on a BFM pre-trained in a reduced-coverage regime.

4.3.2 Online split selection mechanism

Unlike a standard FIFO buffer, the online split implements a selection mechanism that favors transitions pointing in underrepresented directions of the learned ϕ-space and removes transitions that appear redundant. This idea is similar to the coverage-maximizing buffer in Isele & Cosgun (2018), which ranks experiences using a KD-tree on the raw state space. Instead, we score transitions using the same object that Opti-BFM uses to build the confidence set, the inverse design matrix V_t of ϕ in Eq. equation 3. We use this feature-space criterion as a practical proxy for preserving diverse training support, rather than as a formal guarantee of behavioral coverage. Let $M = |\mathcal{D}_{\text{online}}|$. We construct a similar information matrix for $\mathcal{D}_{\text{online}}$, as follows:

$$V = \frac{1}{M}\sum_{i=0}^{M} \phi(s_i')\phi(s_i')^\top + \epsilon I_d, \quad \text{with} \quad V \in \mathbb{R}^{d\times d}, \tag{5}$$

where ϵI is a regularization term. Intuitively, this matrix captures how much mass ϕ places in each direction of the feature space for $\mathcal{D}_{\text{online}}$, and its inverse emphasizes the underrepresented directions. Practically, given a transition (s, a, s'), we can compute a score

$$L(s') = \phi(s')^\top V^{-1}\phi(s'). \tag{6}$$

High values of $L(s')$ indicate that $\phi(s')$ points in a direction that the buffer covers poorly, while low values indicate the transition is probably redundant. Intuitively, we can use L to rank the points inside $\mathcal{D}_{\text{online}}$. Each time the agent collects a new transition (s_t, a_t, s_t'), we compute $L(s_t')$ and compare it with the lowest scoring transition in the buffer $L^* = \min_{s'\in\mathcal{D}_{\text{online}}} L(s')$. When $L(s_t') > L^*$, we remove the transition $(s, a, s') \in \operatorname{argmin}_{s'\in\mathcal{D}_{\text{online}}} L(s')$ by replacing it with (s_t, a_t, s_t').

5 Experiments

We characterize BFMs as continual learners across the three stages of the pipeline: (i) the base *frozen* BFM, (ii) *Opti BFM* with online task inference (Rupf et al., 2025), and (iii) *Continual BFM*, the full pipeline with continual updates via coverage buffer. We conduct two experiments, a comparison against diverse CRL baselines on DM Control (Tassa et al., 2018) and Continual World (CW) (Wołczyk et al., 2021), and we test Continual BFMs on a longer run on DMC, where tasks change randomly and more frequently comparing Continual BFMs only. This latter scenario differs from standard benchmark on CRL and is used to highlight BFMs' capabilities in a more challenging scenario. Additionally, we run ablations on the coverage buffer to understand the optimal offline-to-online data ratio and the effects of continual updates in poor coverage regimes.

5.1 Experimental setup

Baselines. We compare BFMs variants against six baselines of diverse nature. Three SAC-based (Haarnoja et al., 2018) approaches: FT-1, FT-L2, EWC (Wołczyk et al., 2021; Kirkpatrick et al., 2017) which exploit task boundary knowledge to mitigate forgetting, a continuous-action version

of PT-DQN (Anand & Precup, 2023) based on SAC which maintain two value networks: a slow updating one to retain past knowledge and a fast updating one to adapt to the current task. VariBAD (Zintgraf et al., 2019) a bayesian latent approach with a PPO (Schulman et al., 2017) agent, and Replay-based Recurrent RL (3RL) a task-agnostic SAC-based extension of experience replay using a latent variable to encode the task. The goal is not to establish that Continual BFMs are better than existing gradient-based or latent-based approaches; rather, we want to characterize (i) how each pipeline stage contributes to continual metrics, (ii) identify where BFMs are effective and where they are not, and (iii) evaluate whether the coverage buffer improves online performance and isolate the contribution of its components.

Setup. For the BFMs backbone, we consider Forward-Backward (FB) (Touati & Ollivier, 2021; Touati et al., 2022) and Proto-Successor Measure (PSM) (Agarwal et al., 2024). For online task inference, we use OpTI (Rupf et al., 2025). For DM Control, each BFM is pre-trained on the ExoRL dataset collected using RND exploration (Yarats et al., 2022; Burda et al., 2018). For CW10, we combine expert trajectories for the ten CW10 tasks, filtered from the MetaWorld MT50 dataset used by HarmoDT (Hu et al., 2024), with an additional 1M transitions per task collected using RND. Full dataset and pre-training evaluations are reported in Appendix A.

Protocol and metrics. We evaluate on three DM Control domains and Continual World CW10. The default DM Control sequences are *Walker*: stand, walk, run, flip; *Cheetah*: run, run backward, flip, flip backward; and *Quadruped*: walk, run, stand, jump, roll, roll fast. CW10 follows the standard ten-task sequence; all exact task orders are reported in Appendix D.2. Tasks are presented sequentially for $\Delta = 500$K environment steps in DM Control and $\Delta = 1$M steps in CW10. We run five random seeds. Boundary-free methods receive observations, actions, and scalar rewards, but no task identity or task-boundary signal; boundaries are used to switch the environment reward and for evaluation. Boundary-aware baselines are explicitly marked in Table 1.

Following Wołczyk et al. (2021); Gaya et al. (2022); Lopez-Paz & Ranzato (2022), let $P_{i,j}$ denote normalized performance on task i after completing task j, with $i \leq j$. For DM Control, episodic return is normalized by the maximum return of 1000; for CW10, $P_{i,j}$ is the success rate. With K tasks, endpoint accuracy is $\mathrm{ACC} = {}^1\!/_K \sum_{i=1}^{K} P_{i,K}$. Forgetting is $F = {}^1\!/_{K-1} \sum_{i=1}^{K-1} (P_{i,i} - P_{i,K})$; the final task is excluded because it cannot yet be forgotten. Negative forgetting corresponds to backward transfer. We report mean and standard deviation across seeds.

Method	TB	Quadruped		Walker		Cheetah		CW10	
		ACC ↑	F ↓	ACC ↑	F ↓	ACC ↑	F ↓	ACC ↑	F ↓
FT-1	✓	0.20 ± 0.01	0.77 ± 0.02	0.29 ± 0.02	0.81 ± 0.02	0.24 ± 0.02	0.58 ± 0.03	0.11 ± 0.02	0.57 ± 0.09
FT-L2	✓	0.80 ± 0.06	-0.00 ± 0.01	0.76 ± 0.04	-0.00 ± 0.00	0.64 ± 0.01	0.00 ± 0.00	0.20 ± 0.21	0.00 ± 0.00
EWC	✓	0.81 ± 0.06	-0.00 ± 0.01	0.78 ± 0.07	0.01 ± 0.01	0.65 ± 0.01	-0.00 ± 0.00	0.12 ± 0.04	0.00 ± 0.01
PT-SAC	✗	0.19 ± 0.03	0.75 ± 0.05	0.24 ± 0.03	0.78 ± 0.03	0.24 ± 0.04	0.56 ± 0.06	0.07 ± 0.06	0.49 ± 0.18
3RL	✗	0.26 ± 0.23	0.08 ± 0.03	0.72 ± 0.16	0.08 ± 0.13	0.37 ± 0.04	0.39 ± 0.03	0.08 ± 0.04	0.32 ± 0.07
VariBAD (PPO)	✗	0.43 ± 0.22	-0.01 ± 0.09	0.20 ± 0.07	-0.05 ± 0.06	0.21 ± 0.02	0.15 ± 0.07	0.07 ± 0.04	0.17 ± 0.07
FB	✗	0.21 ± 0.06	0.00^*	0.22 ± 0.04	0.00^*	0.17 ± 0.06	0.00^*	0.12 ± 0.04	0.00^*
OpTI-FB	✗	0.66 ± 0.03	0.00^*	0.70 ± 0.01	0.00^*	0.11 ± 0.02	0.00^*	0.05 ± 0.02	0.00^*
Continual FB	✗	0.74 ± 0.02	-0.03 ± 0.02	0.70 ± 0.05	0.01 ± 0.06	0.21 ± 0.06	-0.05 ± 0.06	0.03 ± 0.04	-0.01 ± 0.05
PSM	✗	0.27 ± 0.03	0.00^*	0.26 ± 0.01	0.00^*	0.10 ± 0.02	0.00^*	0.00 ± 0.00	0.00^*
OpTI-PSM	✗	0.55 ± 0.02	0.00^*	0.53 ± 0.08	0.00^*	0.13 ± 0.02	0.00^*	0.02 ± 0.03	0.00^*
Continual PSM	✗	0.65 ± 0.02	-0.06 ± 0.05	0.62 ± 0.05	0.05 ± 0.07	0.20 ± 0.05	0.00 ± 0.11	0.01 ± 0.02	0.00 ± 0.03

Table 1: Results on DM Control (DMC) and Continual World (CW). Gradient-based baselines are shown above the frozen and continual BFM variants. Results are averaged over five random seeds. The *TB* column indicates whether a method requires task-boundary information. For frozen BFM variants, forgetting is marked with $*$ because their parameters do not change during the sequence.

5.2 Main continual-learning results

Comparison against baselines. Table 1 shows that Continual BFMs provide a promising trade-off between accuracy and forgetting without requiring task-boundary information. On *Walker* and *Quadruped*, where the pretrained model already contains useful policies, continual updates further

improve or preserve ACC while maintaining near-zero forgetting. Continual FB clearly outperforms the other boundary-free baselines on Quadruped and obtains a similar ACC to 3RL on Walker, but with substantially lower forgetting and variance. It also approaches the performance of FT-L2 and EWC, despite these methods using task-boundary information. On *Cheetah*, instead, all pretrained BFM variants are weaker. Continual updates improve over the frozen models but do not close the accuracy gap with FT-L2, EWC, and 3RL, suggesting that the proposed pipeline can refine a useful pretrained behavioral space but cannot compensate for insufficient pretraining coverage. Nevertheless, Continual BFMs retain lower forgetting than the other boundary-free approaches. On CW10, all BFM variants obtain low performance. This is consistent with previous observations that current BFMs struggle on manipulation tasks (Mohan et al., 2026). This remains a limitation of current BFMs and shows that continual updates alone are insufficient in this regime. Overall, the results suggest that successor-feature-based BFMs are promising backbones for boundary-free CRL when pretraining provides sufficient behavioral coverage. Moreover, the proposed pipeline is agnostic by design to the specific BFM instantiation and can directly benefit from future improvements in BFM pretraining.

5.3 Ablations and stress tests

Coverage-buffer ablation. Table 2 isolates the two components of the coverage buffer: mixing offline and online data, and filtering online transitions with the score in Eq. equation 6. Updating only on offline data ($\alpha = 0\%$) performs similarly to frozen OpTI-FB, showing that the gains do not arise merely from additional optimization. A moderate online fraction ($\alpha = 25\%$) improves ACC from 0.66 to 0.74 on Quadruped and from 0.10 to 0.22 on Cheetah, while retaining low forgetting. Conversely, using only online data ($\alpha = 100\%$) degrades Walker and Cheetah, consistent with coverage collapse. Replacing score-based selection with FIFO has the largest effect on Quadruped (0.60 versus 0.74 ACC), while its effect is smaller on Walker and Cheetah. Appendix E.1 reports the complete sweep over mixing ratios.

Online mix	**Quadruped**		**Walker**		**Cheetah**	
	ACC ↑	F ↓	ACC ↑	F ↓	ACC ↑	F ↓
OpTI frozen	0.66 ± 0.02	-0.00 ± 0.04	0.70 ± 0.01	0.00 ± 0.01	0.10 ± 0.01	-0.00 ± 0.03
$\alpha = 0\%$	0.65 ± 0.04	-0.02 ± 0.07	0.70 ± 0.01	-0.00 ± 0.01	0.13 ± 0.04	-0.02 ± 0.08
$\alpha = 25\%$	**0.74** ± 0.02	-0.03 ± 0.03	**0.71** ± 0.02	0.02 ± 0.03	**0.22** ± 0.04	-0.04 ± 0.08
$\alpha = 100\%$	0.70 ± 0.02	**-0.04** ± 0.03	0.57 ± 0.10	0.08 ± 0.05	0.08 ± 0.04	0.05 ± 0.05
FIFO	0.60 ± 0.04	0.05 ± 0.05	0.70 ± 0.02	**-0.03** ± 0.04	0.21 ± 0.02	0.02 ± 0.07

Table 2: Coverage-buffer ablation for Continual FB on the default DM Control sequences. α is the fraction of online samples in each update batch. FIFO uses $\alpha = 25\%$ but replaces score-based filtering with first-in first-out replacement. The full ratio sweep is reported in Appendix E.1.

Long-horizon stability. We evaluate Continual BFMs for 20M environment steps on separate *Walker*, *Quadruped*, and *Cheetah* streams. Each stream contains 200 task intervals sampled randomly from the domain's default tasks without consecutive repetitions; the active task changes every $\Delta = 100$K steps. At the end of each interval, we evaluate the current BFM on every task in that domain. These boundaries are used only for evaluation and are not provided to the agent. This experiment uses three random seeds and asks whether repeated coverage-buffer updates remain stable relative to the initial model, rather than repeating the baseline comparison in Table 1. Figure 2 (Top) averages evaluation returns across tasks and environments. Continual FB and PSM remain above their frozen pretraining references and show no cumulative decline over the full stream. Complete per-task curves, which expose behavior hidden by the aggregate, are reported in Appendix D.4.

Reduced pre-training coverage. To control the quality of the initial behavioral space, we partition standardized Walker observations from ExoRL into $K = 20$ clusters using k-means. We construct three datasets by retaining transitions from only the largest $N \in \{5, 10, 15\}$ clusters and pretrain a separate FB model on each dataset. Smaller N yields both narrower state-space support and a weaker initial model. We then compare Continual FB ($\alpha = 25\%$) with its frozen OpTI-FB counterpart over the default Walker sequence using five seeds; Figure 2 (Bottom) shows the online trajectories and

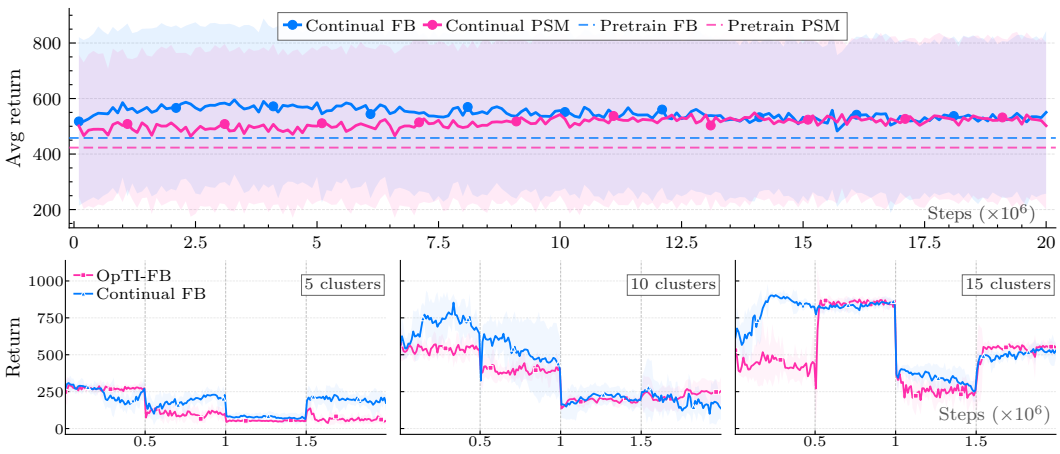


Figure 2: Top: aggregate evaluation return for Continual FB and PSM over the 20M-step long-DMC streams; dashed lines are the corresponding frozen pretraining references. Bottom: online return on the Walker sequence for Continual FB and frozen OpTI-FB when FB is pretrained on the largest $N \in \{5, 10, 15\}$ of 20 state-space clusters. Shaded regions show variability across evaluations.

Table 3 reports the continual metrics. Continual updates consistently improve ACC by 0.02–0.04, but introduce some forgetting. More importantly, the large gap between models initialized with 5 and 15 clusters remains, showing that online updates partially recover missing coverage but do not remove dependence on the initial behavioral space. Full pretraining and per-task results are reported in Appendix E.2.

N	OpTI-FB ACC	Continual ACC	Continual F
5	0.13±0.03	**0.15±0.01**	0.03±0.06
10	0.34±0.03	**0.37±0.05**	0.03±0.08
15	0.49±0.06	**0.53±0.03**	0.10±0.08

Table 3: Reduced-coverage Walker results. Frozen OpTI-FB has structural $F = 0$.

Limitations. The results show that continual updates can refine a useful pre-trained behavioral space, but do not remove the dependence on the underlying BFM. In the reduced-coverage experiment, improvements remain bounded by the initial model, while on CW10 current BFMs fail to provide useful behaviors for most tasks, consistently with previous observations that BFMs struggle on manipulation domains (Mohan et al., 2026). At the same time, the pipeline is agnostic to the particular successor-feature-based BFM, and improvements in reward-free pre-training or BFM representations can be incorporated without changing the overall continual-learning framework.

6 Conclusions

In this work, we investigated BFMs as backbones for CRL and introduced a three-phase pipeline, agnostic to the underlying successor-feature-based BFM, that enables continual updates through a coverage-preserving buffer. Our results show that BFMs are promising boundary-free continual agents: when pretraining provides a useful behavioral space, they adapt effectively, maintain low forgetting, and benefit from continual updates over long horizons. At the same time, their performance remains limited by the quality and expressiveness of the pretrained model, and current BFMs still struggle in more complex domains, such as manipulation tasks. An important future direction is to complement continual updates with coverage-driven online exploration, allowing the agent to discover relevant behaviors that are missing from the initial model. It would also be interesting to study different buffer selection mechanisms and use deviation regret to evaluate their long-term benefits (Elelimy et al., 2025).

7 AI USE STATEMENT

In this work, we used generative AI tools for generate synthetic data sets, design or provide feedback on research methodology or experiments, implement methods, assist with translation. We have not used generative AI tools for help develop theoretical models or conceptual frameworks, formulate mathematical claims, clean and reformat dataset, support qualitative and thematic data analysis, interpret results, and provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses are not applicable to this work. Additionally, we used generative AI tools for create or modify scientific figures or images, create or edit software code, discover research topics or identify gaps, brainstorming, edit a research paper to improve readability, identify relevant literature. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

David Abel, André Barreto, Benjamin Van Roy, Doina Precup, Hado P van Hasselt, and Satinder Singh. A definition of continual reinforcement learning. *Advances in Neural Information Processing Systems*, 36:50377–50407, 2023.

Siddhant Agarwal, Harshit Sikchi, Peter Stone, and Amy Zhang. Proto successor measure: Representing the space of all possible solutions of reinforcement learning. 2024.

Nishanth Anand and Doina Precup. Prediction and control in continual reinforcement learning. *Advances in neural information processing systems*, 36:63779–63817, 2023.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P Van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Diana Borsa, André Barreto, John Quan, Daniel Mankowitz, Rémi Munos, Hado Van Hasselt, David Silver, and Tom Schaul. Universal successor features approximators. *arXiv preprint arXiv:1812.07626*, 2018.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Massimo Caccia, Jonas Mueller, Taesup Kim, Laurent Charlin, and Rasool Fakoor. Task-agnostic continual reinforcement learning: Gaining insights and overcoming challenges, 2023. URL https://arxiv.org/abs/2205.14495.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Esraa Elelimy, David Szepesvari, Martha White, and Michael Bowling. Rethinking the foundations for continual reinforcement learning. *arXiv preprint arXiv:2504.08161*, 2025.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. The information geometry of unsupervised reinforcement learning. *arXiv preprint arXiv:2110.02719*, 2021.

Jean-Baptiste Gaya, Thang Doan, Lucas Caccia, Laure Soulier, Ludovic Denoyer, and Roberta Raileanu. Building a subspace of policies for scalable continual learning. *arXiv preprint arXiv:2211.10445*, 2022.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Muhammad Burhan Hafez and Kerim Erekmen. Continual deep reinforcement learning with task-agnostic policy distillation. *Scientific Reports*, 14(1):31661, 2024.

Shengchao Hu, Ziqing Fan, Li Shen, Ya Zhang, Yanfeng Wang, and Dacheng Tao. Harmodt: Harmony multi-task decision transformer for offline reinforcement learning. *arXiv preprint arXiv:2405.18080*, 2024.

David Isele and Akansel Cosgun. Selective experience replay for lifelong learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Pranaya Jajoo, Harshit Sikchi, Siddhant Agarwal, Amy Zhang, Scott Niekum, and Martha White. Regularized latent dynamics prediction is a strong baseline for behavioral foundation models. *arXiv preprint arXiv:2603.15857*, 2026.

Khurram Javed and Richard S Sutton. The big world hypothesis and its ramifications for artificial intelligence. In *Finding the Frame: An RLC Workshop for Examining Conceptual Frameworks*, 2024.

Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476, 2022.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114 (13):3521–3526, 2017.

Hao Liu and Pieter Abbeel. Aps: Active pretraining with successor features. In *International Conference on Machine Learning*, pp. 6736–6747. PMLR, 2021.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning, 2022. URL https://arxiv.org/abs/1706.08840.

Mikel Malagon, Josu Ceberio, and Jose A Lozano. Self-composing policies for scalable continual reinforcement learning. *arXiv preprint arXiv:2506.14811*, 2025.

Aditya Mohan, Caleb Chuck, Siddhant Agarwal, and Amy Zhang. Is manipulation hard for successor measures? In *Workshop on Reinforcement Learning Beyond Rewards @ Reinforcement Learning Conference 2026*, 2026.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pp. 5331–5340. PMLR, 2019.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in neural information processing systems*, 32, 2019.

Thomas Rupf, Marco Bagatella, Marin Vlastelica, and Andreas Krause. Optimistic task inference for behavior foundation models. *arXiv preprint arXiv:2510.20264*, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Jonathan Schwarz, Jelena Luketina, Wojciech M Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & Compress: A scalable framework for continual learning.

Harshit Sikchi, Andrea Tirinzoni, Ahmed Touati, Yingchen Xu, Anssi Kanervisto, Scott Niekum, Amy Zhang, Alessandro Lazaric, and Matteo Pirotta. Fast adaptation with behavioral foundation models. *arXiv preprint arXiv:2504.07896*, 2025.

Jingbo Sun, Songjun Tu, Haoran Li, Xin Liu, Yaran Chen, Ke Chen, Dongbin Zhao, et al. Unsupervised zero-shot reinforcement learning via dual-value forward-backward representation. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. *Advances in Neural Information Processing Systems*, 34:13–23, 2021.

Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist? *arXiv preprint arXiv:2209.14935*, 2022.

Núria Armengol Urpí, Marin Vlastelica, Georg Martius, and Stelian Coros. Epistemically-guided forward-backward exploration, July 2025.

Maciej Wołczyk, Michał Zając, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Continual world: A robotic benchmark for continual reinforcement learning. *Advances in Neural Information Processing Systems*, 34:28496–28510, 2021.

Maciej Wolczyk, Michał Zając, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Disentangling transfer in continual reinforcement learning. *Advances in Neural Information Processing Systems*, 35:6304–6317, 2022.

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. *arXiv preprint arXiv:2201.13425*, 2022.

Tom Zahavy, Andre Barreto, Daniel J Mankowitz, Shaobo Hou, Brendan O'Donoghue, Iurii Kemaev, and Satinder Singh. Discovering a set of policies for the worst case reward. *arXiv preprint arXiv:2102.04323*, 2021.

Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. *arXiv preprint arXiv:1910.08348*, 2019.

A PRE-TRAINING OF BFMS

In the first phase of our pipeline, we pre-train the BFM on a pre-collected dataset. For the *DM Control* suite, we use the offline *ExoRL* dataset, with RND (Burda et al., 2018) as an exploration bonus. For *Continual World*, we use an expert dataset from Hu et al. (2024), containing data for `MT50` tasks of MetaWorld from which we filter only tasks for CW10 sequence and mix them with additional data collected via RND exploration, which amounts to 1M transitions per task. We evaluate the performance of the pre-trained BFM for each benchmark and task. This value will be used later as a reference to indicate the maximum value attainable by the pre-trained BFM, for comparison against *Continual BFMs*. Table 4 reports pre-train evaluations, showing the average return (or success for CW) over 10 rollouts. For both DM Control and CW, we infer z sampling a batch of observations and rewards (5×10^5 samples) from the dataset (Touati & Ollivier, 2021).

Table 4: Average return results (and success) from *pre-trained* FB and PSM over DM Control and Continual World benchmarks. Used as a reference for the best possible return achievable for each task.

Domain	**Task**	**Return**		**Success**	
		FB	**PSM**	**FB**	**PSM**
Quadruped	walk	474.3 ± 31.5	$\mathbf{497.1 \pm 47.8}$	–	–
	run	356.9 ± 13.4	$\mathbf{366.5 \pm 23.2}$	–	–
	stand	$\mathbf{686.2 \pm 76.0}$	620.4 ± 89.4	–	–
	jump	$\mathbf{564.7 \pm 60.1}$	534.7 ± 39.7	–	–
	roll	$\mathbf{434.3 \pm 202.8}$	281.0 ± 72.9	–	–
	roll_fast	$\mathbf{495.9 \pm 91.9}$	257.1 ± 8.6	–	–
	Average	$\mathbf{\textit{502.0 \pm 79.3}}$	$\textit{426.1 \pm 46.9}$	–	–
Walker	stand	771.1 ± 11.1	$\mathbf{945.6 \pm 14.3}$	–	–
	walk	740.8 ± 29.6	$\mathbf{741.9 \pm 18.0}$	–	–
	run	$\mathbf{448.8 \pm 18.7}$	358.0 ± 4.0	–	–
	flip	$\mathbf{693.5 \pm 23.8}$	585.6 ± 27.3	–	–
	Average	$\mathbf{\textit{663.6 \pm 20.8}}$	$\textit{657.8 \pm 15.9}$	–	–
Cheetah	run	$\mathbf{48.7 \pm 10.8}$	48.6 ± 8.8	–	–
	run_backward	43.3 ± 15.8	$\mathbf{63.2 \pm 17.2}$	–	–
	walk	180.0 ± 9.2	$\mathbf{228.9 \pm 16.2}$	–	–
	walk_backward	248.8 ± 66.6	$\mathbf{298.0 \pm 128.8}$	–	–
	flip	256.4 ± 16.8	$\mathbf{307.2 \pm 28.3}$	–	–
	flip_backward	$\mathbf{396.3 \pm 142.1}$	318.0 ± 73.4	–	–
	Average	$\textit{195.6 \pm 43.5}$	$\mathbf{\textit{210.7 \pm 45.4}}$	–	–
CW10	hammer-v2	$\mathbf{540.3 \pm 103.9}$	207.3 ± 121.8	0.00 ± 0.00	0.00 ± 0.00
	push-wall-v2	91.1 ± 50.6	$\mathbf{383.2 \pm 483.8}$	0.00 ± 0.00	0.00 ± 0.00
	faucet-close-v2	4408.4 ± 341.1	$\mathbf{4601.1 \pm 27.1}$	1.00 ± 0.00	1.00 ± 0.00
	push-back-v2	$\mathbf{20.2 \pm 16.6}$	3.5 ± 3.3	$\mathbf{0.30 \pm 0.46}$	0.00 ± 0.00
	stick-pull-v2	$\mathbf{18.4 \pm 10.7}$	11.7 ± 4.1	0.00 ± 0.00	0.00 ± 0.00
	handle-press-side-v2	$\mathbf{3977.7 \pm 777.3}$	3088.8 ± 1611.2	$\mathbf{1.00 \pm 0.00}$	0.90 ± 0.30
	push-v2	25.9 ± 13.0	$\mathbf{223.3 \pm 314.1}$	0.00 ± 0.00	$\mathbf{0.10 \pm 0.30}$
	shelf-place-v2	7.1 ± 12.8	$\mathbf{22.6 \pm 44.5}$	0.00 ± 0.00	0.00 ± 0.00
	window-close-v2	566.0 ± 225.1	$\mathbf{1176.4 \pm 586.3}$	$\mathbf{1.00 \pm 0.00}$	0.20 ± 0.40
	peg-unplug-side-v2	16.8 ± 7.6	$\mathbf{17.0 \pm 6.2}$	0.00 ± 0.00	0.00 ± 0.00
	Average	$\textit{967.2 \pm 155.9}$	$\mathbf{\textit{973.5 \pm 320.2}}$	$\mathbf{\textit{0.33 \pm 0.05}}$	$\textit{0.22 \pm 0.10}$

B PSEUDOCODE AND COMPLEXITY DISCUSSION

Pseudocode for the full pipeline is shown in Algorithm 2, while pseudocode for coverage buffer is presented in Algorithm 1. The computational cost of insertion is dominated by V^{-1}: the outer product $\phi(s')$ costs $O(d^2)$ for each of the M transitions, plus $O(d^3)$ for the inversion, $O(Md^2 + d^3)$ in total. Since $d \ll M$, the bottleneck is the outer product. Practically, we avoid recomputing V^{-1} at

every insertion; instead, we cache its value and refresh every n replacements. In this way, the score computation $L(s')$ is cheaper, costing only $O(d^2)$ per transition.

Algorithm 1 Coverage Buffer

Require: ϕ; $\mathcal{D}_{\text{pretrain}}$; batch size B; ratio α; capacity M; refresh rate n; regularizer ϵ

$\mathcal{D}_{\text{offline}} \leftarrow$ uniform sample from $\mathcal{D}_{\text{pretrain}}$; $\mathcal{D}_{\text{online}} \leftarrow$ first M transitions
$c \leftarrow 0$ ▷ Counter for V^{-1} update
UPDATE() ▷ build V^{-1} and cached scores L
procedure SAMPLE( )
  return $\{\cdot \sim \mathcal{D}_{\text{online}}\}_1^{\alpha B} \cup \{\cdot \sim \mathcal{D}_{\text{offline}}\}_1^{(1-\alpha)B}$
end procedure
procedure INSERT((s_t, a_t, s'_t))
  $L_t \leftarrow \phi(s'_t)^\top V^{-1} \phi(s'_t)$ ▷ $O(d^2)$, cached V^{-1}
  $\tau^- \leftarrow \arg\min_{s' \in \mathcal{D}_{\text{online}}} L(s')$ ▷ $O(M)$ on cached L
  if $L_t > L(\tau^-)$ **then**
    $\mathcal{D}_{\text{online}} \leftarrow \mathcal{D}_{\text{online}} \setminus \{\tau^-\} \cup \{(s_t, a_t, s'_t)\}$; $L(s'_t) \leftarrow L_t$
    if $(c \leftarrow c + 1) \bmod n = 0$ **then** UPDATE()
  end if
end procedure
procedure UPDATE( )
  $V \leftarrow \frac{1}{M} \sum_{s' \in \mathcal{D}_{\text{online}}} \phi(s')\phi(s')^\top + \epsilon I_d$; $V^{-1} \leftarrow$ INVERSE(V) ▷ $O(Md^2 + d^3)$
  $L(s') \leftarrow \phi(s')^\top V^{-1} \phi(s')$ $\forall s' \in \mathcal{D}_{\text{online}}$
end procedure

C BASELINES DESCRIPTION

We briefly introduce the baselines used to compare with BFM-based continual learners.

Fine-Tuning (FT). FT-1 Wołczyk et al. (2021) is the simplest continual learning baseline, and consists of sequentially training a single policy across tasks. After completing task k, the learned parameters $\theta_k^\star$ are used to initialize training on task $k+1$, without any mechanism to preserve previous knowledge. The optimization objective is simply

$$\mathcal{L}_{\text{FT}}(\theta) = \mathcal{L}_{\text{RL}}^{(k+1)}(\theta), \tag{7}$$

where $\mathcal{L}_{\text{RL}}$ denotes the underlying RL loss (e.g., PPO, SAC, or DQN); here we use SAC Haarnoja et al. (2018). Although FT maximizes plasticity, it requires *task boundary* knowledge and typically suffers from *catastrophic forgetting* because all parameters can adapt to the new task.

Fine-Tuning with L2 Regularization (FT-L2). FT-L2 Wołczyk et al. (2021) mitigates forgetting by penalizing deviations from the parameters learned on the previous task. However, this penalty can be added only when the task changes, so *task boundary* knowledge is required. Let $\theta_k^\star$ denote the parameters after learning task k. Training on task $k+1$ minimizes

$$\mathcal{L}_{\text{FT-L2}}(\theta) = \mathcal{L}_{\text{RL}}^{(k+1)}(\theta) + \lambda \left\|\theta - \theta_k^\star\right\|_2^2, \tag{8}$$

where $\lambda > 0$ controls the strength of the regularization. This method assumes that every parameter is equally important, which often limits adaptation when only a subset of parameters is actually responsible for previous knowledge.

Elastic Weight Consolidation (EWC). EWC Kirkpatrick et al. (2017) extends FT-L2 by recognizing that not all parameters contribute equally to previously learned tasks. Instead of applying a uniform penalty, it weights each parameter according to its estimated importance, approximated by the diagonal *Fisher Information Matrix*. The optimization objective becomes

$$\mathcal{L}_{\text{EWC}}(\theta) = \mathcal{L}_{\text{RL}}^{(k+1)}(\theta) + \frac{\lambda}{2} \sum_i F_i(\theta_i - \theta_{k,i}^\star)^2, \tag{9}$$

Algorithm 2 Continual BFM

Require: $\mathcal{D}_{\text{pretrain}}$; mixing ratio α; batch size B; regularizer ϵ, BFM $\phi, \psi, \{\pi_z\}_{z\in\mathcal{Z}}$

// Phase 1 — Offline pre-training ▷ Sec. 4.1

$(\phi, \psi) \leftarrow \arg\min_{\phi,\psi}\ \mathcal{L}_{\text{BFM}}(\phi, \psi)$ on $\mathcal{D}_{\text{pretrain}}$

// Phase 2 — Online task inference (OpTI-BFM) ▷ Sec. 4.2

init coverage buffer: $\mathcal{D}_{\text{offline}} \sim \mathcal{D}_{\text{pretrain}}$, $\mathcal{D}_{\text{online}} \leftarrow \emptyset$

$V_0 \leftarrow \lambda I_d$; $\quad \hat{z}_0 \leftarrow 0$

for $t = 0, 1, 2, \ldots$ **do**

  $\mathcal{C}_t \leftarrow \{z : \|z - \hat{z}_{t-1}\|_{V_{t-1}} \le \beta_t\}$

  $z_t \leftarrow \arg\max_{z\in\mathcal{C}_t} \max_{w\in\mathcal{C}_t} w^\top \psi(s_t, \pi_z(s_t), z)$ ▷ optimism

  act with π_{z_t}; observe (s_t, a_t, s'_t, r_t)

  $V_t \leftarrow V_{t-1} + \phi(s_t)\phi(s_t)^\top$; $\quad \hat{z}_t \leftarrow V_t^{-1} \sum_{i\le t} \phi(s_i) r_i$

  // Phase 3 — Continual update ▷ Sec. 4.3

  INSERT(s_t, a_t, s'_t) ▷ coverage buffer, Alg. 1

  $\mathcal{B} \leftarrow$ SAMPLE(α, B) ▷ mix offline/online

  $(\phi, \psi) \leftarrow (\phi, \psi) - \eta \nabla \mathcal{L}_{\text{BFM}}(\phi, \psi; \mathcal{B})$

end for

where F_i is the Fisher importance of parameter θ_i. Thus, parameters critical for previous tasks are strongly constrained, while less important parameters remain free to adapt, balancing retention and plasticity. Like FT-L2, this requires *task boundary* knowledge.

Permanent-Transient SAC (PT-SAC). PT-SAC is a continuous-action extension of PT-DQN, a continual RL method introduced by Anand & Precup (2023). It decomposes the action-value function into permanent and transient components,

$$Q^{(\text{PT})}(s, a) = Q_\theta^{(\text{P})}(s, a) + Q_w^{(\text{T})}(s, a). \tag{10}$$

The permanent critic is updated less frequently to capture knowledge useful across tasks, while the transient critic is updated more often to adapt to the current task. The transient component is updated at every step using the SAC objective, and the actor is optimized against the combined critic. On a fixed schedule, information is consolidated into the permanent component and the transient residual is reset. The schedule is chosen independently of observed task switches, so PT-SAC does not receive task-boundary or task-identity information. Its frequency controls the stability-plasticity trade-off.

VariBAD. VariBAD (Zintgraf et al., 2019) is a Bayes-adaptive meta-RL method that infers a latent representation of the active task from interaction history. A recurrent variational encoder maps the trajectory $\tau_{0:t} = (s_0, a_0, r_0, \ldots, s_t)$ to an approximate posterior $q_\phi(z \mid \tau_{0:t})$ over a latent task variable z. The agent then acts according to a latent-conditioned policy:

$$a_t \sim \pi_\theta(\cdot \mid s_t, z_t), \qquad z_t \sim q_\phi(\cdot \mid \tau_{0:t}), \tag{11}$$

and can therefore adapt as new evidence is observed. The encoder is trained with a variational objective using reward and transition reconstruction, while the policy is trained with PPO in our experiments. VariBAD does not require task-boundary information but relies on the current task being identifiable from the observed trajectory.

Replay-based Recurrent RL (3RL). 3RL (Caccia et al., 2023) is a task-agnostic continual RL method that combines experience replay with recurrent task inference. A GRU encodes a recent history of states, actions, and rewards into a context representation,

$$z_t = \text{GRU}\big((s_i, a_i, r_i)_{i=t-h}^{t-1}\big), \tag{12}$$

which is provided to the SAC actor and critics. The recurrent representation allows the agent to adapt its behavior from recent experience without observing the active task ID. During training, sequences sampled from replay are used both to learn the context encoder and to rehearse experience collected earlier, reducing interference and forgetting. Our task-agnostic implementation does not provide the agent with task identities or explicit task-boundary signals.

D RESULTS ON DM CONTROL AND CONTINUAL WORLD

D.1 METRICS

Following Wołczyk et al. (2021); Gaya et al. (2022); Lopez-Paz & Ranzato (2022), we evaluate BFMs and baselines on standard CRL metrics such as accuracy (ACC), backward transfer (BWT), and forgetting (F). Let Δ be the duration of a task, and K the total number of tasks. Each task $k \in \{0, \dots, K\}$ lasts for a time window $[t_k, t_k + \Delta)$. We denote as $P_{i,j}$ the normalized performance on task i *after* training on task j is complete, with $i \leq j$. We define endpoint accuracy as $\text{ACC} = {}^1\!/_K \sum_{i=1}^{K} P_{i,K}$. Forgetting $F = {}^1\!/_{K-1} \sum_{i=0}^{K-1} \big(P_{i,i} - P_{i,K}\big)$ compares the performance of task i after training on task i with its performance after all tasks are completed. The last task is excluded, as it cannot be forgotten. Backward transfer (BWT) happens when forgetting is negative, and can be read as $-F$ in the results.

D.2 ADDITIONAL DETAILS

We provide additional details and plots for the DM Control and Continual World benchmarks introduced in Table 1, and for the longer DM Control experiment summarized in Figure 2 (Top).

We consider twelve baselines: three gradient-based continual RL approaches requiring task boundaries (FT-1, FT-L2, and EWC), three boundary-free techniques (PT-SAC, 3RL, and VariBAD), and three variants of each BFM (Base BFM, OpTI BFM, and Continual BFM). We validate the pipeline with Forward Backward (FB) Touati & Ollivier (2021); Touati et al. (2022) and Proto Successor Measure (PSM) Agarwal et al. (2024).

We compare baselines and BFMs on two benchmarks: *DM Control* Tassa et al. (2018), using *Walker*, *Cheetah*, and *Quadruped*, and *Continual World* Wołczyk et al. (2021). For each environment, we define custom *task sequences* in Table 5. The continual learner faces these tasks sequentially, each for Δ timesteps. We perform two experiments, detailed below.

Env name	Sequence	Len.	Tasks
Quadruped	default	6	walk, run, stand, jump, roll, roll_fast
	long	200	jump x36, roll x32, roll_fast x43, run x31, stand x29, walk x29
Walker	default	4	stand, walk, run, flip
	long	200	flip x59, run x50, stand x41, walk x50
Cheetah	default	4	run, run_backward, flip, flip_backward
	long	200	flip x50, flip_backward x59, run x41, run_backward x50
CW10	default	10	hammer-v2, push-wall-v2, faucet-close-v2, push-back-v2, stick-pull-v2, handle-press-side-v2, push-v2, shelf-place-v2, window-close-v2, peg-unplug-side-v2

Table 5: Task sequences for each environment. For DM Control, *long* sequences are created by randomly sampling *default* tasks 200 times, without consecutive repeats. We report how often each task is sampled.

D.3 DEFAULT CONTINUAL LEARNING ON DM CONTROL AND CW

We compare baselines and BFMs on four environments (Walker, Cheetah, Quadruped, CW10) using the *default* task sequence. For DM Control, each task runs for $\Delta = 500$K timesteps. For Continual World, agents act on each task for $\Delta = 1$M timesteps. Section 5 discusses these results, summarized in Table 1. Here, we add plots comparing baselines and BFMs: Figures 3 and 4 show per-task returns on the *default* DM Control sequence against boundary-aware and boundary-free baselines, respectively. Figures 5 and 6 show the same comparison for Continual World CW10.

D.4 LONG HORIZON ON DM CONTROL

The previous experiment on *default* task sequence tell us that BFMs are competitive against standard CRL baselines, and that using the coverage buffer and continually update the agent online always

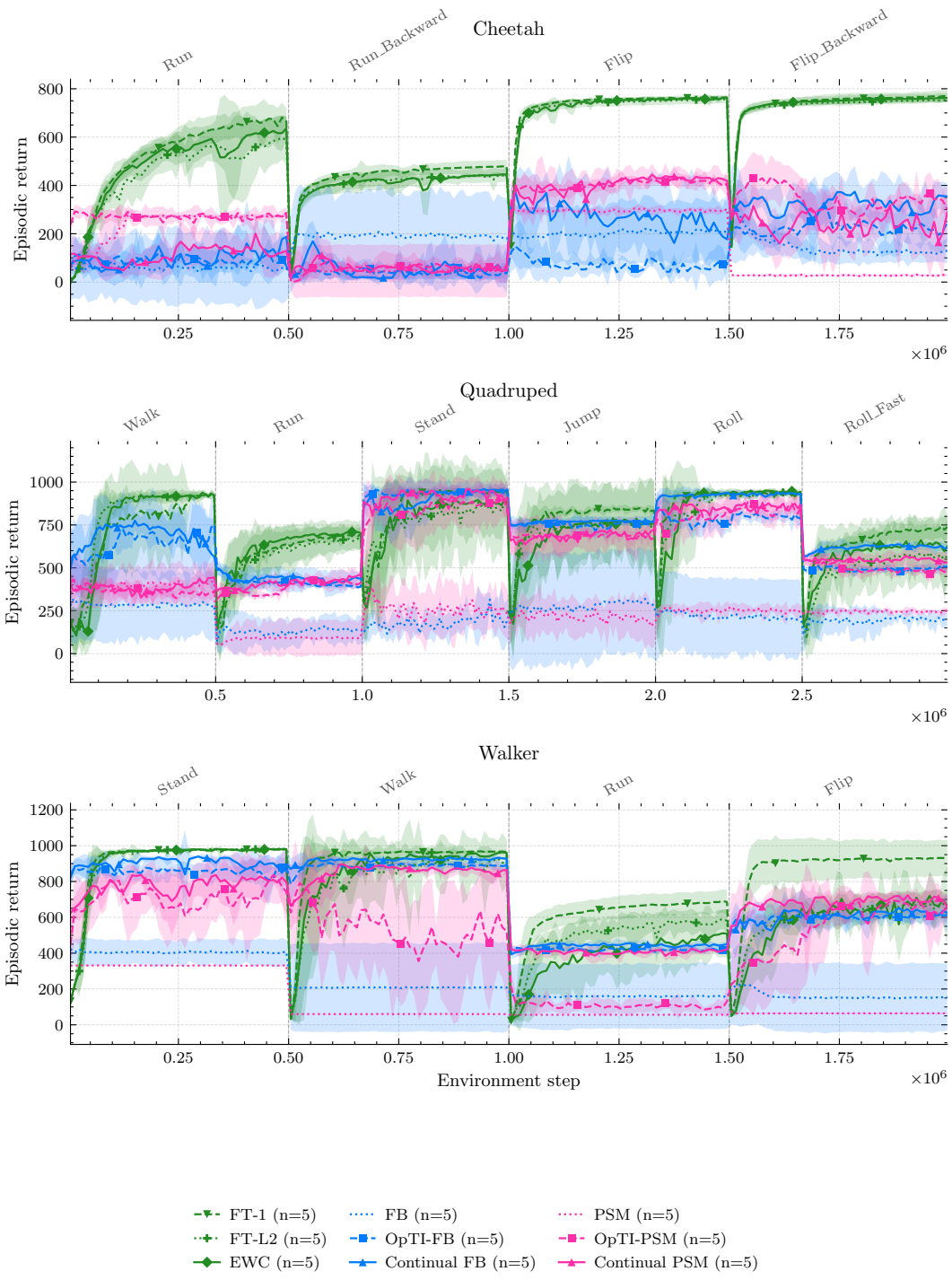


Figure 3: Online episodic return for DM Control environments against boundary-aware techniques, using the *default* task sequence. Each task lasts $\Delta = 500$K timesteps. Each baseline is averaged over $n = 5$ random seeds.

improve performance over frozen models. However, it is hard to tell whether coverage buffer can be stable on a long horizon continual learning scenario. For this reason we run an additional experiment where the agent faces a *fast changing* task sequence for 20M timesteps. For each environment, Cheetah, Walker and Quadruped we test both *Continual BFMs* on the *long* task sequence detailed in

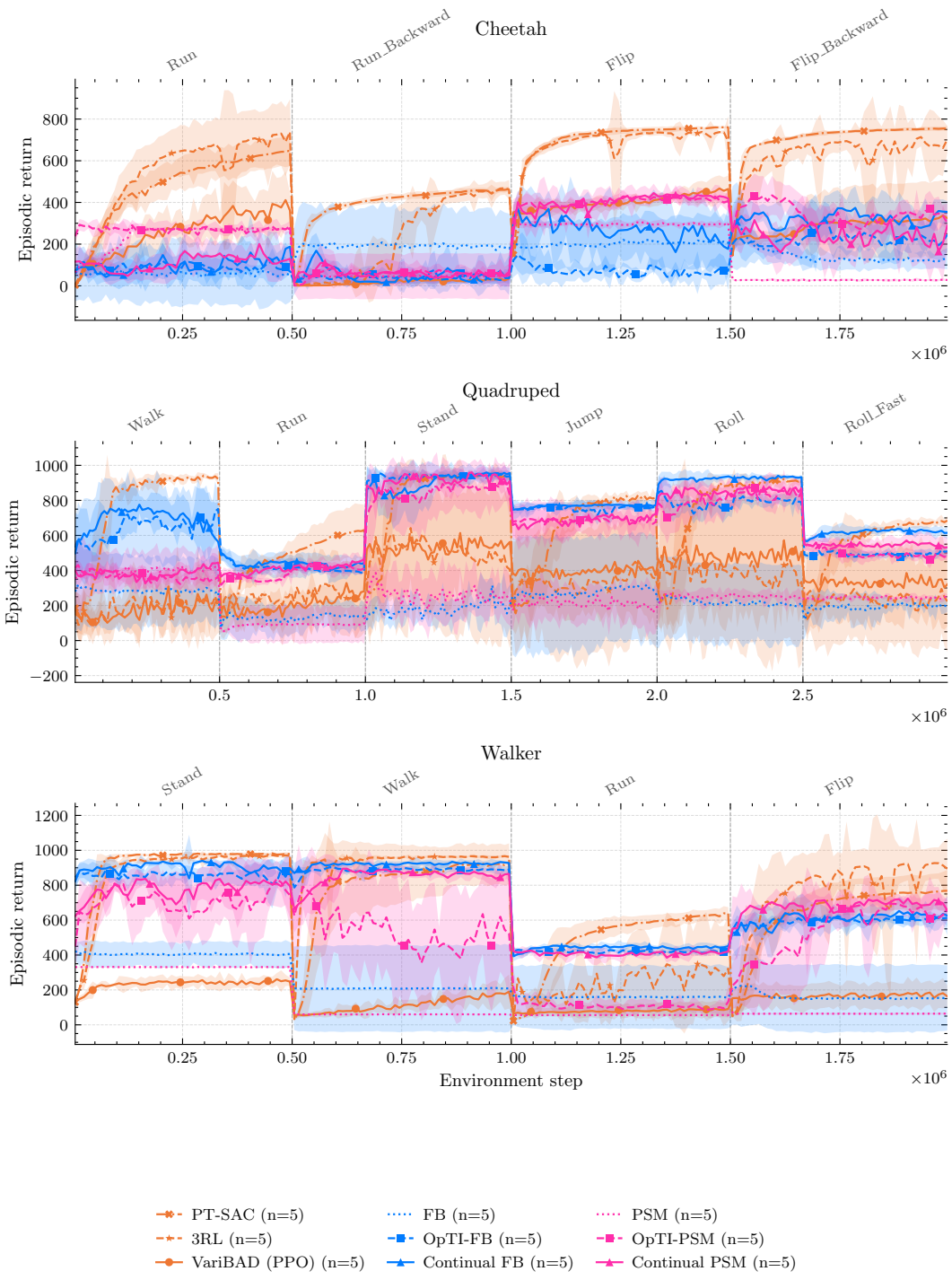


Figure 4: Online episodic return for DM Control environments against boundary-free techniques, using the *default* task sequence. Each task lasts $\Delta = 500$K timesteps. Each baseline is averaged over $n = 5$ random seeds.

Table 5. This is a randomly generated sequence of 200 tasks each lasting for $\Delta = 100$K timesteps. For each environment, we report the evaluations of the average return at the *end* of each tasks (every $\Delta = 100$K) for a total of 200 tasks. Figure 2 (Top) shows a summary of this experiment, where we average each env and task return. Here, we report full evaluation plots for each DM Control

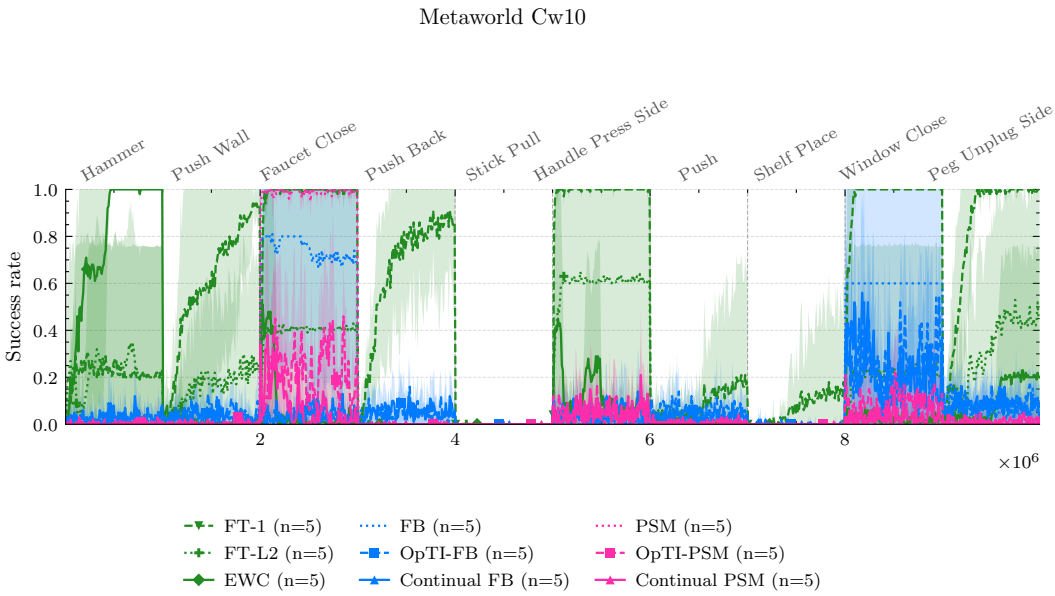


Figure 5: Online episodic return for Continual World with task sequence CW10 against boundary-aware techniques. Each task lasts $\Delta = 1$M timesteps, for 10M timesteps total. Each baseline is averaged over $n = 5$ random seeds.

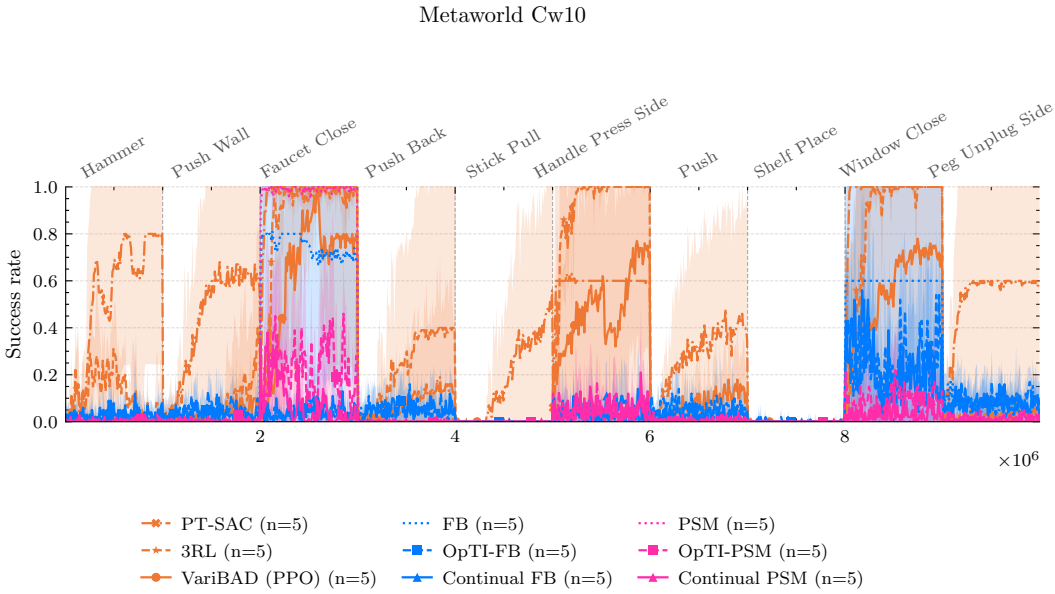


Figure 6: Online episodic return for Continual World with task sequence CW10 against boundary-free techniques. Each task lasts $\Delta = 1$M timesteps, for 10M timesteps total. Each baseline is averaged over $n = 5$ random seeds.

environment: Figure 8 shows evaluation for each task across the 20M timesteps for Quadruped, similarly for Walker in Figure 7 and Cheetah in Figure 9.

In each plot, we compare *Continual FB* and *Continual PSM* against a fixed reference: the pre-train performance for that task of the base BFM, showed as a dashed line in the plots and already introduced in Table 4. From these plots we can draw two insights: (i) continual updates via coverage buffer almost always improve the return or in the worst case it keep the perfomance close to the frozen reference and (ii) in a longer horizon task with frequent change of tasks, there is no drop in

performance and the return stays constant, showing almost no forgetting over time signaling that coverage buffer updates are safe. For completeness we report the full evaluation over 20M timesteps for Walker in Figure 7, for Quadruped in Figure 8 and Cheetah in Figure 9

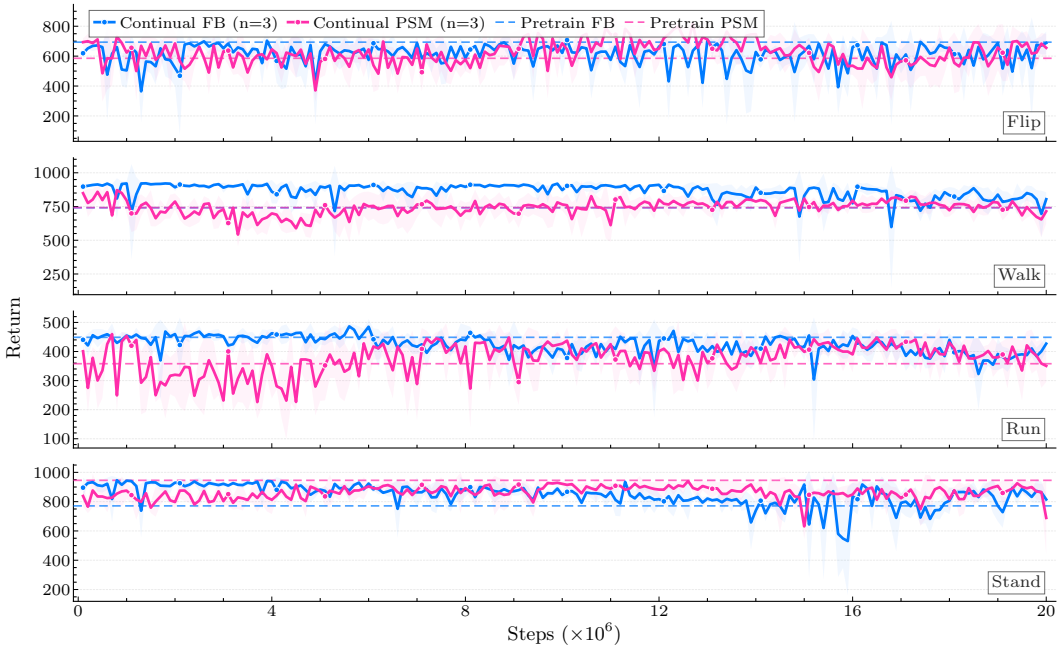


Figure 7: Evaluation return per task on long horizon continual Walker. Every $\Delta = 100$K timesteps, when the task changes randomly, the agent is evaluated on each task for 10 rollouts, for a total of 200 task change amounting to 20M timesteps. Each baselines is averaged over 3 random seeds

E Ablations

In this section, we run two ablations to characterize the behavior of the coverage buffer. Specifically, (i) we want to understand empirically whether there is an optimal ratio between offline and online data that promotes learning while avoiding coverage collapse, and if this ratio is environment dependent or not. Then, (ii) we want to characterize whether continual updates via coverage buffer can recover the performance in scenarios where the pre-train model exhibits poor coverage.

E.1 Ablation on online-offline data ratio

During continual interactions, in the last stage of the pipeline, the agent updates its representation by sampling a batch of *unsupervised* data $\mathcal{B}$ from the coverage buffer, according to Eq. equation 4, where α controls the percentage of *online* data in the buffer. In this ablation, we investigate how different offline-online ratios α impact the learning and continual metrics such as accuracy and forgetting. To this extent, we report results on each environment, for *Continual FB*, varying the ratio in the range $\alpha = \{0\%, 10\%, 25\%, 50\%, 75\%, 100\%\}$ and comparing the results to the frozen *Opti FB* which does not update online. Notice that $\alpha = 0\%$ is not the same as *Opti FB*, the agent still continues to update during continual interactions but only on offline data and benefit from additional training. We include it to show that the improvements are not due to additional training but specifically from collected online interactions. Additionally, we challenge our selection mechanism introduced in Eq. equation 6 to score online collected transitions. We want to understand whether it improves over a simple first-in first-out mechanism, to this extent, we set the selection mechanism to be a simple FIFO i.e., we set the score function to be constant and use a ratio $\alpha = 25\%$.

Table 6 shows results for $\alpha = \{0\%, 10\%, 25\%, 50\%, 75\%, 100\%\}$, plus frozen Opti FB and Continual FB with FIFO selection. The best ratio empirically converges near 25% across environments. Quadruped is more resilient to aggressive online updates, while Walker and Cheetah show stronger

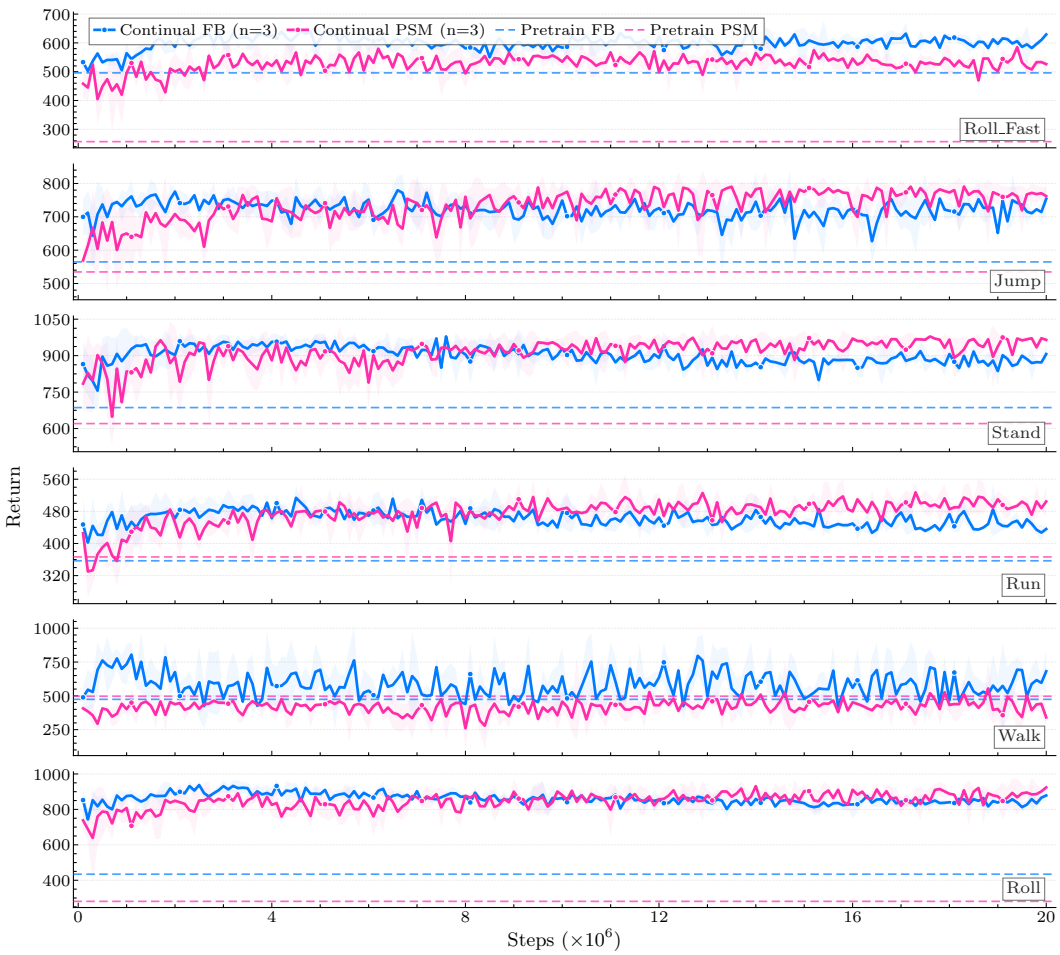


Figure 8: Evaluation return per task on long horizon continual Quadruped. Every $\Delta = 100$K timesteps, when the task change randomly, the agent is evaluated on each task for 10 rollouts, for a total of 200 task change amount to 20M timesteps. Each baselines is averaged over 3 random seeds

Online mix	**Quadruped**		**Walker**		**Cheetah**	
	ACC ↑	F ↓	ACC ↑	F ↓	ACC ↑	F ↓
OpTI frozen	0.66 ± 0.02	-0.00 ± 0.04	0.70 ± 0.01	0.00 ± 0.01	0.10 ± 0.01	-0.00 ± 0.03
$\alpha = 0\%$	0.65 ± 0.04	-0.02 ± 0.07	0.70 ± 0.01	-0.00 ± 0.01	0.13 ± 0.04	-0.02 ± 0.08
$\alpha = 10\%$	0.70 ± 0.04	0.00 ± 0.07	$\mathbf{0.71} \pm 0.04$	$\mathbf{-0.03} \pm 0.05$	$\mathbf{0.22} \pm 0.04$	$\mathbf{-0.11} \pm 0.08$
$\alpha = 25\%$	$\mathbf{0.74} \pm 0.02$	-0.03 ± 0.03	$\mathbf{0.71} \pm 0.02$	0.02 ± 0.03	$\mathbf{0.22} \pm 0.04$	-0.04 ± 0.08
$\alpha = 50\%$	0.73 ± 0.03	0.01 ± 0.04	0.64 ± 0.05	0.04 ± 0.09	0.16 ± 0.06	0.00 ± 0.11
$\alpha = 75\%$	0.73 ± 0.03	-0.02 ± 0.02	0.49 ± 0.13	0.18 ± 0.10	0.16 ± 0.07	-0.01 ± 0.14
$\alpha = 100\%$	0.70 ± 0.02	$\mathbf{-0.04} \pm 0.03$	0.57 ± 0.10	0.08 ± 0.05	0.08 ± 0.04	0.05 ± 0.05
FIFO	0.60 ± 0.04	0.05 ± 0.05	0.70 ± 0.02	$\mathbf{-0.03} \pm 0.04$	0.21 ± 0.02	0.02 ± 0.07

Table 6: Coverage buffer mixing ratio α ablation. For each environment we run *Continual FB* with different α configurations and a frozen *Opti FB* on *default* task sequences. We consider a coverage buffer with ratio in $\alpha = \{0\%, 10\%, 25\%, 50\%, 75\%, 100\%\}$, using the selection mechanism of Eq. equation 6. FIFO means that online collection of new transition has a first in first out mode, without our coverage selection mechanism, for FIFO mechanism we used the best ratio 25%.

coverage collapse at higher α. All environments exhibit a U-shaped trend: performance is lower as α approaches zero or 100%, indicating no gains and coverage collapse, respectively. Walker benefits from online updates, but gains over the frozen model are small, likely because the pre-trained

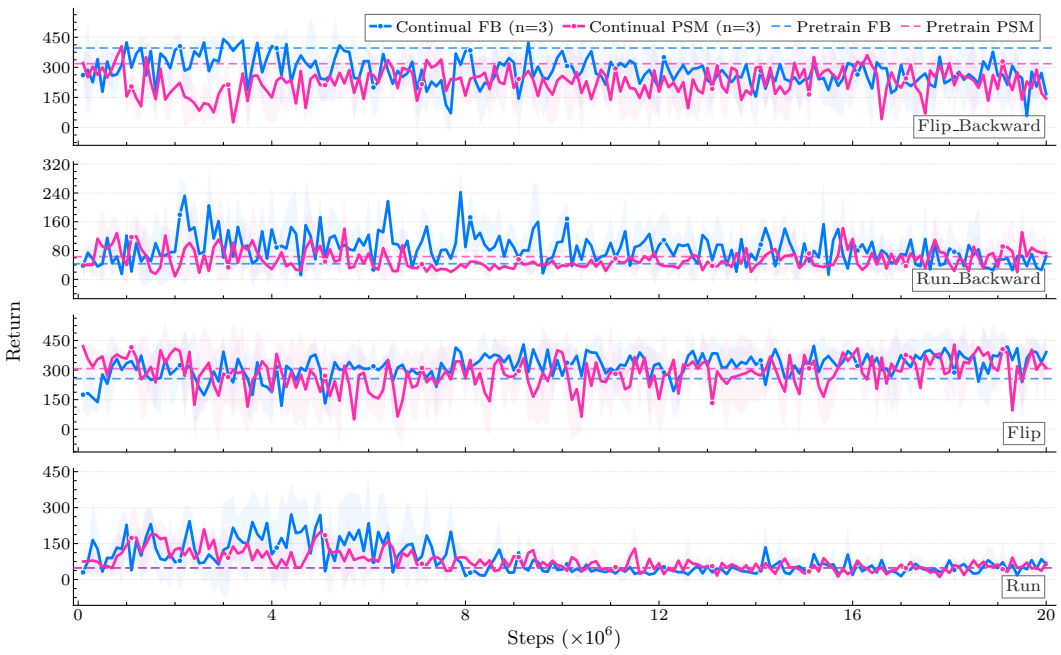


Figure 9: Evaluation return per task on long horizon continual Cheetah. Every $\Delta = 100$K timesteps, when the task change randomly, the agent is evaluated on each task for 10 rollouts, for a total of 200 task change amount to 20M timesteps. Each baselines is averaged over 3 random seeds

representation is already strong for those tasks. Quadruped and Cheetah benefit substantially, with Cheetah *doubling* performance. The selection mechanism has a strong effect on Quadruped and a modest effect on Walker and Cheetah.

E.2 Ablation on pre-trained model coverage

A crucial question is whether the continual BFM pipeline remains stable when pre-training coverage is *poor*, and whether online updates can improve performance.

To create a reduced-coverage dataset, we partition the original state space into $K = 20$ clusters with k-means on standardized observations. We then build datasets of decreasing coverage by keeping only the N largest clusters, for $N = \{5, 10, 15\}$. Dropping the least populated clusters removes rarely visited state-space regions, increasing sparsity as N decreases. We pre-train one BFM for each dataset $\mathcal{D}^i_{\text{pretrain}}$, yielding lower performance as coverage decreases. Table 7 shows Walker performance for $N \in \{5, 10, 15\}$.

We run *Continual FB* on base BFMs with different coverage and compare against the same frozen BFMs equipped with the Opti framework. Table 8 shows the results. When coverage is low, Continual FB improves performance to a level better than or similar to the frozen counterpart, suggesting that continual updates can recover from poor coverage but remain bounded by the pre-trained representation. Forgetting is not catastrophic, though slightly higher than in the full-coverage runs. Figure 10 shows episodic return over the full task sequence.

F Coverage buffer filtering

To visualize coverage-buffer filtering, we show its evolution during continual execution. Figures 12, 13, and 11 show t-SNE projections of ϕ-features for buffer transitions, with one snapshot at the end of each task for Walker, Cheetah, and Quadruped. Each snapshot overlays the offline rehearsal split $\mathcal{D}_{\text{offline}}$ (blue) with the online split $\mathcal{D}_{\text{online}}$ filtered by Eq. equation 6 (red). These visualizations show that score-based selection populates ϕ-space regions that complement the rehearsal split, and help inspect coverage collapse: a collapsed buffer would shrink the online split into one region, whereas

Table 7: Per-task average return for FB on the DM Control Walker domain, averaged over 30 rollouts, when pre-trained on $\{5, 10, 15\}$ clusters of the ExoRL dataset.

Domain	**Task**	**Return**	
		FB	**PSM**
Walker (5 clusters)	stand	132.5 ± 66.1	–
	walk	51.9 ± 18.1	–
	run	39.5 ± 9.6	–
	flip	40.9 ± 30.5	–
	Average	*66.2* $\pm$ *31.1*	–
Walker (10 clusters)	stand	374.6 ± 50.8	–
	walk	500.1 ± 134.7	–
	run	79.0 ± 13.5	–
	flip	64.9 ± 6.9	–
	Average	*254.6* $\pm$ *51.5*	–
Walker (15 clusters)	stand	734.9 ± 53.0	–
	walk	884.7 ± 20.6	–
	run	401.3 ± 12.5	–
	flip	570.4 ± 38.5	–
	Average	*647.8* $\pm$ *31.2*	–

Table 8: Coverage ablation comparing Opti FB and Continual FB when the base BFM is pre-trained on a reduced-coverage dataset. The number of clusters N measures coverage: smaller N indicates lower coverage and lower average performance. Continual FB uses $\alpha = 25\%$ and $\Delta = 500$K for each task. The task sequence is the Walker *default*.

Env.	Variant	ACC ↑	Forgetting ↓
walker (5C)	OpTI-FB	0.13 ± 0.03	**0.00***
	Continual FB	**0.15** $\pm$ 0.01	0.03 ± 0.06
walker (10C)	OpTI-FB	**0.34** $\pm$ 0.03	**0.00***
	Continual FB	**0.37** $\pm$ 0.05	0.03 ± 0.08
walker (15C)	OpTI-FB	0.49 ± 0.06	**0.00***
	Continual FB	**0.53** $\pm$ 0.03	0.10 ± 0.08

here it remains spread throughout the sequence. We project $5{,}000$ points per split using t-SNE, computed independently per snapshot.

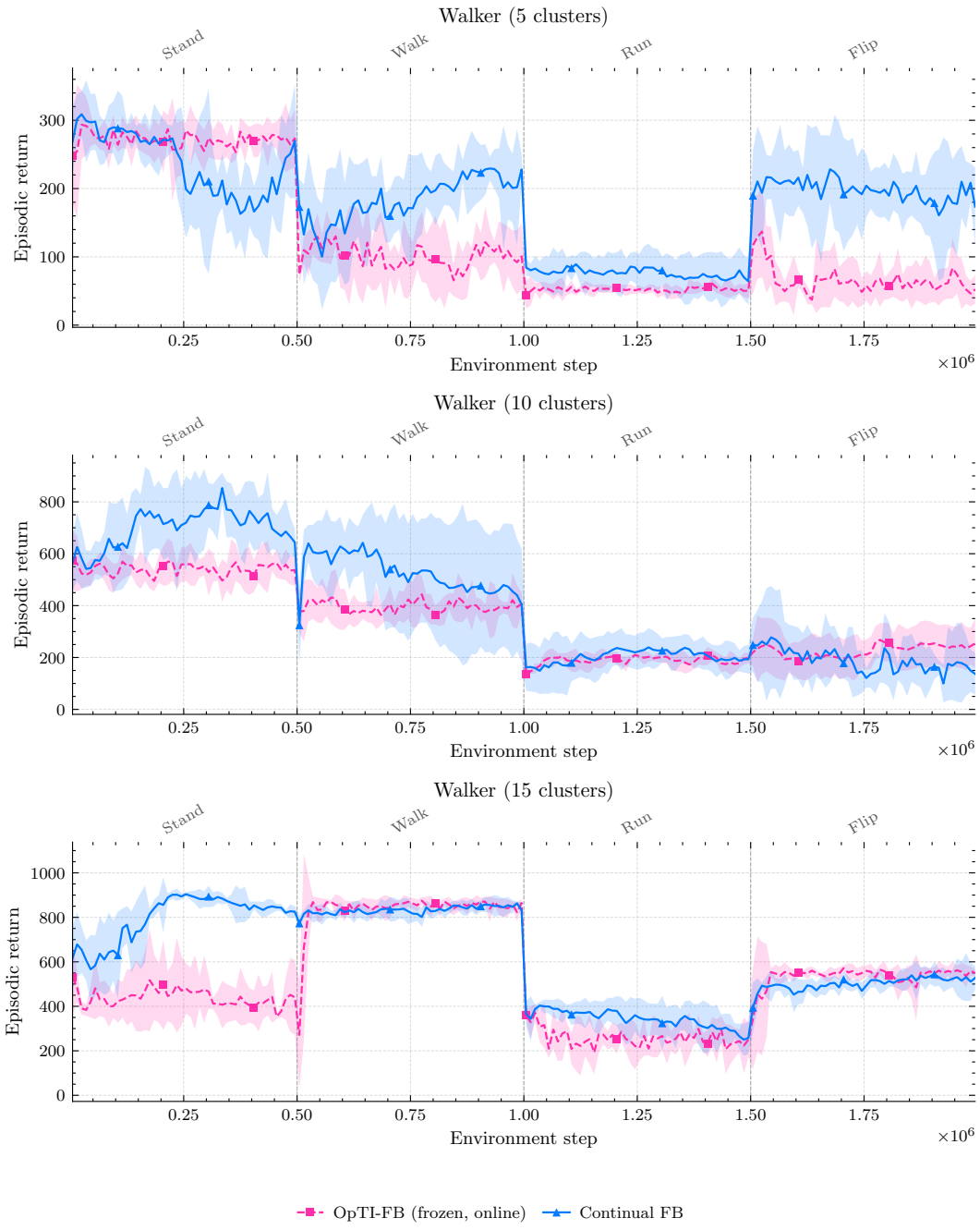


Figure 10: Per-task online episodic return on the DM Control Walker continual sequence, for Continual FB pre-trained on varying clusters of the ExoRL dataset.

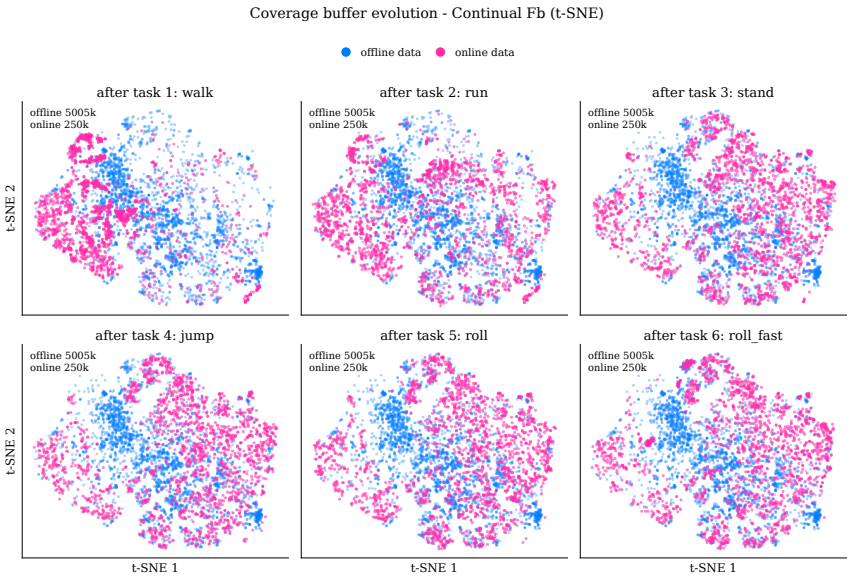


Figure 11: Evolution of the coverage in the online split of the coverage buffer for the Quadruped environment.

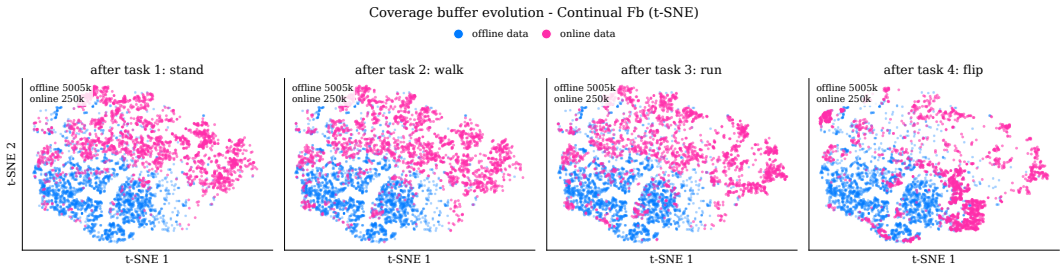


Figure 12: Evolution of the coverage in the online split of the coverage buffer for the Walker environment.

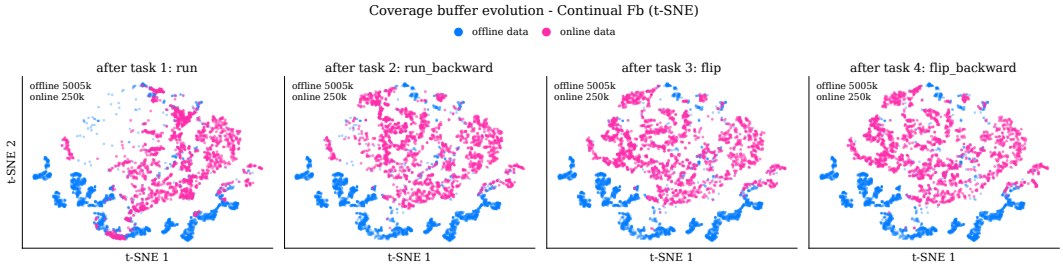


Figure 13: Evolution of the coverage in the online split of the coverage buffer for the Cheetah environment.

G HYPERPARAMETERS

We report the hyperparameters used for our baselines. Each BFM includes base, Opti, and continual variants.

FB

Parameter	Value	Description
Base BFM		
B	1024	online gradient batch size
W_z	50,000	rolling reward-inference window
K_z	500	rolling z refresh period
ρ_{on}	0.25	online fraction for continual replay mix
K_{cov}	1,000	coverage-buffer refresh period
z_{dim}	50	latent goal dimension
K_{train}	100/200	online update period, DMC / CW10
Adaptation	rolling reward inference	frozen pre-trained FB model
OpTI-specific		
λ	0.01/1.0	BLR prior precision, DMC / CW10
β	0.05/1.0	optimism scale, DMC / CW10
n	128	candidate z samples
γ_{decay}	0.9999/0.99995	posterior decay, DMC / CW10
κ	0	posterior covariance floor
K_{OpTI}	10	OpTI posterior/action-z refresh period
Adaptation	OpTI BLR	frozen FB model, no gradient updates
Continual-specific		
η_F, η_B, η_π	2×10^{-5}	online fine-tuning learning rates
$\|\nabla\|_{\max}$	1.0	online gradient clipping
Replay mix	offline + coverage online	coverage policy: leverage

PSM

Parameter	Value	Description
Base BFM		
B	1024	online gradient batch size
W_z	50,000	rolling reward-inference window
K_z	500	rolling z refresh period
ρ_{on}	0.25	online fraction for continual replay mix
K_{cov}	1,000	coverage-buffer refresh period
z_{dim}	50	latent goal dimension
K_{train}	200	online update period
b_{seed}	16	seed-code bit width
Adaptation	rolling reward inference	frozen pre-trained PSM model
OpTI-specific		
λ	$0.01/1.0$	BLR prior precision, DMC / CW10
β	$0.05/1.0$	optimism scale, DMC / CW10
n	128	candidate z samples
γ_{decay}	$0.9999/0.99995$	posterior decay, DMC / CW10
κ	0	posterior covariance floor
K_{OpTI}	10	OpTI posterior/action-z refresh period
Adaptation	OpTI BLR	frozen PSM model, no gradient updates
Continual-specific		
$\eta_\phi, \eta_\psi, \eta_\pi$	2×10^{-5}	online fine-tuning learning rates
$\|\nabla\|_{\max}$	1.0	online gradient clipping
N_{repr}	0	skip representation-only warmup online
Replay mix	offline + coverage online	coverage policy: leverage

FT-1

Parameter	Value	Description
B	128	gradient batch size
N_{warm}	10,000	random-action warmup per task
$N_{\text{upd.after}}$	1,000	no updates before this many task steps
K_{upd}	50	environment-step interval between update bursts
$\|\mathcal{D}\|$	10^6	replay-buffer capacity
Task boundary	reset buffer + Adam state	no regularizer or consolidation
Actor/critic	multi-head SAC	one head per task

FT-L2

Parameter	Value	Description
B	128	gradient batch size
N_{warm}	10,000	random-action warmup per task
$N_{\text{upd.after}}$	1,000	no updates before this many task steps
K_{upd}	50	environment-step interval between update bursts
$\|\mathcal{D}\|$	10^6	replay-buffer capacity
λ_{L2}	10^5	anchor penalty coefficient
Anchor	$\theta^\star \leftarrow \theta_{\text{actor}}$	updated at each task boundary
Penalty	$\lambda_{\text{L2}} \sum_k \|\theta - \theta_k^\star\|_2^2$	cumulative actor anchor

EWC

Parameter	Value	Description
B	128	gradient batch size
N_{warm}	10,000	random-action warmup per task
$N_{\text{upd.after}}$	1,000	no updates before this many task steps
K_{upd}	50	environment-step interval between update bursts
$\lvert\mathcal{D}\rvert$	10^6	replay-buffer capacity
λ_{EWC}	10^4	EWC penalty coefficient
N_F	$1280/2560$	Fisher samples, DMC / CW10
B_F	256	Fisher batch size
Fisher	analytical Gaussian	cumulative single-anchor Fisher

PT-SAC / PT-DQN-style P/T

Parameter	Value	Description
B	128	gradient batch size
N_{warm}	10,000	random-action warmup after a scheduled reset
$N_{\text{upd.after}}$	1,000	no updates before this many steps after reset
K_{upd}	50	environment-step interval between update bursts
$\lvert\mathcal{D}\rvert$	10^6	replay-buffer capacity
η_P	10^{-4}	permanent critic learning rate
N_P	1000	permanent distillation steps per scheduled consolidation
B_P	256	permanent distillation batch size
Q	$Q_P + Q_T$	permanent/transient value decomposition
Transient reset	enabled	Q_T reset to zero residual on a fixed schedule

3RL

Parameter	Value	Description
B	256	gradient batch size
N_{warm}	10,000	random-action warmup per task
K_{upd}	50	environment-step interval between update bursts
U	10	gradient updates per burst
H	15	history length
d_c	50	GRU context dimension
f_{curr}	0	minimum current-task replay fraction

VariBAD

Parameter	Value	Description
B	128	PPO/VAE batch context from rollout
$\eta_{\pi,V}$	3×10^{-4}	PPO actor/value learning rate
η_{VAE}	10^{-3}	encoder/decoder learning rate
H	15	history length
d_z	5	latent dimension
d_h	128	encoder hidden dimension
β_{KL}	0.1	KL loss coefficient
N_{PPO}	2048	rollout length
E_{PPO}	10	PPO epochs per rollout
ϵ_{clip}	0.2	PPO clipping coefficient

Experiments were run on a SLURM-managed GPU cluster using NVIDIA A40 GPUs. Main training runs used one GPU and 8 CPU threads per run, with memory requests of 15-16 GB. Auxiliary preprocessing, evaluation, and ablation jobs used the same GPU type and CPU allocation, with memory requests up to 16 GB.