OFFLINE META REINFORCEMENT LEARNING WITH BAYESIAN TASK RELABELLING

Anonymous authors
Paper under double-blind review

ABSTRACT

Offline Meta-reinforcement learning faces two challenges when transferring from collected data to online adaptation. Behaviour-dependent task inference arises when an encoder identifies tasks from collector-specific patterns rather than task properties. Belief distribution shift can lead to poor actions when the policy encounters unfamiliar beliefs about the task. We introduce Bayesian Task Relabelling (BTR) to address both challenges using a learned Bayesian task model. We build on the Bayesian linear task model and identify how its conditional reward and next-state likelihoods exclude collector behaviour as separate evidence of task identity. To broaden belief coverage, BTR samples imaginary tasks using a posterior-based approximation and generates their corresponding imaginary rewards along collected histories. The same model combines these imaginary rewards with collected next states to update task beliefs in closed form, providing additional data for policy training. Task sampling is not restricted to training tasks or achieved goals, and reward generation requires no known task-specific reward function. In comparisons with recent or representative baselines trained on the same data, BTR achieves the highest mean return on all eight control benchmarks under online adaptation and seven under offline-context evaluation.

1 INTRODUCTION

Meta-reinforcement learning (Meta-RL) trains an agent to adapt to unseen tasks zero-shot or within a few episodes. Training can proceed online through environment interaction, or offline using a fixed dataset. We focus on offline Meta-RL (OMRL). A common approach is *context-based*: an encoder maps observed interactions to a task representation z, and a policy uses this representation to choose actions (Li et al., 2020; Yuan & Lu, 2022; Gao et al., 2023; Li et al., 2024). After offline training, the agent adapts by collecting rewards and next states, updating its task representation, and acting on that representation.

This separation between offline training and online adaptation creates two challenges. First, *behaviour-dependent task inference*: an encoder can identify training tasks from the data collector's behaviour, then infer the wrong task when that behaviour changes. Second, *belief distribution shift*: a task belief assigns probabilities to tasks given the observed interactions. During adaptation, beliefs unlike those used in policy training can lead to poor actions, even at familiar states.

To address the first challenge, prior methods train representations to retain task information while reducing dependence on collector behaviour. CORRO uses contrastive learning, while CSRO penalises behaviour-related information (Yuan & Lu, 2022; Gao et al., 2023). UNICORN relates these objectives through an information-theoretic view, and MetaSTAR uses reward and next-state prediction to guide representation learning (Li et al., 2024; Qian et al., 2026). However, these objectives do not generally *prevent* an encoder from exploiting collector-specific patterns, such as how frequently particular state–action pairs occur. These correlations can help distinguish training tasks without requiring the encoder to infer task properties, such as the rewards and next states resulting from an action in a given state. This motivates an inference rule that structurally excludes collector behaviour as separate evidence of task identity.

To address the second challenge, prior methods change the experience available for training or adaptation. SMAC collects additional online experience to expose the agent to its own exploration data (Pong et al., 2022). IDAQ selects adaptation episodes to keep context closer to the offline

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.

distribution (Wang et al., 2023). Model-based methods generate additional trajectories for policy training (Lin et al., 2022; Qian et al., 2026). However, online collection is unavailable in a strictly offline setting. Context selection does not teach the policy to handle unfamiliar beliefs, while generated trajectories depend on learned dynamics.

On the other hand, reward relabelling offers a principled way to generate additional data for policy training without new or simulated interaction. A collected history can be reused for another task by replacing its rewards. BOReL applies this technique in OMRL, selecting another training task for relabelling (Dorfman et al., 2021). Hindsight Task Relabelling (HTR) uses it in online Meta-RL, selecting a goal already reached within a collected episode, such as a position the agent visited (Packer et al., 2021). However, both relabelling mechanisms require a known reward function for the selected task. Their task choices are also restricted to existing training tasks or goals already achieved in collected episodes.

To this end, we introduce Bayesian Task Relabelling (BTR), an OMRL framework that uses a learned Bayesian task model to infer tasks and relabel data for policy training. We build on the Bayesian linear task model (BLTM) and its closed-form inference rule introduced by GLiBRL (You & Kurniawati, 2025). BTR samples imaginary tasks using a posterior-based approximation, generates their corresponding imaginary rewards along collected histories, and infers the resulting beliefs using this rule. We then train a belief-conditioned policy on the original and relabelled data for online adaptation. During training, an asymmetric critic receives additional task information, while the policy uses only the state and task belief.

Contributions.

- We identify how the conditional structure of the inherited BLTM excludes collector behaviour as separate evidence of task identity in offline task inference.
- We develop Bayesian task relabelling to broaden belief coverage for offline policy training. It connects posterior-based imaginary-task sampling, learned reward generation, and Bayesian belief updating, without new interaction or simulated dynamics. It requires neither known task-specific reward functions nor achieved hindsight goals, and is not restricted to existing training tasks.
- We achieve the highest mean return against state-of-the-art baselines on all eight benchmarks under online adaptation and seven under offline-context evaluation. Controlled ablations show the benefits of relabelling, task sampling beyond the training set, and the asymmetric critic.

2 BACKGROUND

2.1 OFFLINE META-RL

A task is an MDP $M_\theta = (\mathcal{S}, \mathcal{A}, T_\theta, R_\theta, \gamma)$ with unknown task parameters θ. Tasks share the state and action spaces but can differ in their reward and transition functions. Offline meta-training provides a fixed dataset D_i of interaction tuples (s, a, r, s') for each training task i. Each dataset can come from a different, task-dependent collection policy. A *context* C is a collection of observed interaction tuples used to infer the task. During online adaptation, the agent builds this context through its own actions on an unseen task. The objective is to maximise expected discounted return during adaptation, with discount factor γ.

2.2 BAYES-ADAPTIVE MDPS

A belief $b(\theta)$ is a probability distribution over task parameters given the observed history. It represents both the agent's estimate of the task and its remaining uncertainty. A Bayes-adaptive MDP (BAMDP) includes this belief in its state (Duff, 2002; Ghavamzadeh et al., 2015). Its state is the pair (s, b), called a *hyper-state*. A policy $\pi(a \mid s, b)$ can therefore choose actions using both the physical state and the available task information.

After the agent chooses action a in state s and observes the outcome (r, s'), Bayes' rule gives the updated belief b':

$$b'(\theta) \propto b(\theta)\, p(r, s' \mid s, a, \theta). \tag{1}$$

The update weights the current belief by the likelihood of the observed outcome and normalises it. The next hyper-state is (s', b'). In practice, the policy can receive a vector summary of the belief rather than the full distribution (Zintgraf et al., 2019).

2.3 BAYESIAN LINEAR TASK MODELS

Evaluating equation 1 requires reward and next-state likelihoods and a prior over task parameters. We use the Bayesian linear task model (BLTM) introduced by GLiBRL (You & Kurniawati, 2025). Its linear structure and conjugate priors make the belief update available in closed form while retaining learnable reward and dynamics features.

The likelihood models. Let $\theta_R = (\mu_R, \sigma_R)$ and $\theta_T = (\mu_T, \sigma_T)$ denote the task-specific reward and dynamics model parameters. The reward noise variance is $\sigma_R > 0$, and the transition noise covariance σ_T is diagonal. The feature maps $\phi_R(s, a, s')$ and $\phi_T(s, a)$ are parameterised by η_R and η_T, respectively. For an interaction tuple (s, a, r, s'), the likelihoods are

$$\begin{aligned} p_{\eta_R}(r \mid s, a, s', \theta_R) &= \mathcal{N}\big(r; \phi_R(s, a, s')^\top \mu_R, \sigma_R\big)\,, \\ p_{\eta_T}(s' \mid s, a, \theta_T) &= \mathcal{N}\big(s'; \phi_T(s, a)^\top \mu_T, \sigma_T\big)\,. \end{aligned} \tag{2}$$

Thus μ_R and μ_T parameterise the mean reward and next-state functions. We write $\eta = (\eta_R, \eta_T)$ for the feature parameters and $\theta = (\theta_R, \theta_T)$ for the unknown task parameters.

The task belief update. BLTM places independent conjugate Normal–Gamma priors on the function parameters and noise precisions of both models, using the Matrix-Normal–Gamma form for multivariate next-state predictions. The task priors $p(\theta_R)$, $p(\theta_T)$ are

$$\begin{aligned} p(\theta_R) &= \mathcal{N}(\mu_R; M_R, \sigma_R \Xi_R^{-1}) \,\mathrm{Gamma}\Big(\sigma_R^{-1}; \frac{\nu_R}{2}, \frac{\omega_R}{2}\Big)\,, \\ p(\theta_T) &= \mathcal{MN}(\mu_T; M_T, \Xi_T^{-1}, \sigma_T) \prod_{j=1}^{d_s} \mathrm{Gamma}\Big((\sigma_T^{-1})_{jj}; \frac{\nu_{T,j}}{2}, \frac{\omega_{T,j}}{2}\Big)\,. \end{aligned} \tag{3}$$

The priors are expressed over function parameters and noise precisions. Here, $d_s = \dim(s)$, and $\mathcal{MN}$ denotes the matrix-normal distribution with the stated mean, row covariance, and column covariance. Gamma distributions use the shape–rate convention. The parameters M, Ξ, ν, and ω specify each prior belief.

We define the task belief for context C using these priors and conditional likelihoods:

$$b_C(\theta) = p_\eta(\theta \mid C) \propto p(\theta_R)p(\theta_T) \prod_{(s,a,r,s') \in C} p_{\eta_T}(s' \mid s, a, \theta_T)\, p_{\eta_R}(r \mid s, a, s', \theta_R). \tag{4}$$

Here, $p_\eta(\theta \mid C)$ denotes this conditional-likelihood belief. For task-dependent collectors, it need not equal the full observational posterior (Appendix A.3). The belief b_C assigns density to candidate task parameters according to their prior density and how well they predict the observed outcomes. Conjugacy preserves the Normal–Gamma form, with reward posterior parameters $(M_{R,C}, \Xi_{R,C}, \nu_{R,C}, \omega_{R,C})$ and analogous dynamics parameters. These describe a distribution over the unknown task parameters, not a single task. For example, $M_{R,C}$ is the posterior mean of μ_R; the remaining parameters describe uncertainty in the reward function and reward noise.

Given another interaction tuple, we write the updated belief as $b' = U_\eta(b, (s, a, r, s'))$. For fixed η, U_η is a deterministic Bayesian update. It multiplies the current belief by the tuple's conditional reward and next-state likelihoods and normalises the result. Conjugacy makes this update available in closed form; Appendix A.1 gives the posterior-parameter recursions.

Learning the model offline. Task inference holds η fixed and updates the belief over θ. Model learning instead fits the feature parameters η to collected data. Let $\mathcal{B}$ index sampled tasks, with batches $B_i \subset D_i$. Write $p_\eta(B_i \mid \theta)$ for the product of conditional likelihoods in equation 4. We minimise the negative log marginal likelihood, adding dimension-normalised squared-feature penalties $\mathcal{R}_{\phi_R}, \mathcal{R}_{\phi_T}$ with weights λ_R, λ_T:

$$\mathcal{L}_{\mathrm{model}}(\eta) = -\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \frac{1}{|B_i|} \log \int p(\theta) p_\eta(B_i \mid \theta)\, d\theta + \lambda_R \mathcal{R}_{\phi_R} + \lambda_T \mathcal{R}_{\phi_T}. \tag{5}$$

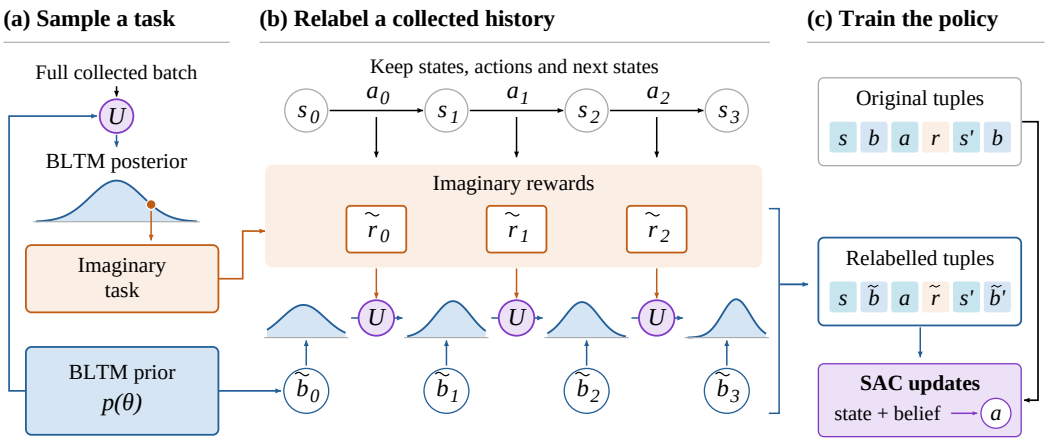

Figure 1: BTR overview. (a) Update the BLTM prior with the collected batch, then sample an imaginary task using the resulting posterior. (b) Generate its corresponding imaginary rewards at each step of the collected history and update the belief after each reward. The same task prior supplies the illustrated initial belief $\widetilde{b}_0$ directly. States, actions, and next states are kept from the collected data. (c) Train the policy on original and relabelled tuples. U denotes the Bayesian update U_η.

Under BLTM, the integral in Equation 5 is analytical, with the exact form delayed to Appendix A.2.

3 METHOD

We develop BTR to train a belief-conditioned policy from offline data for online adaptation. We first examine why BLTM suits offline task inference. We then address the beliefs missing from offline policy training: BTR samples imaginary tasks, generates corresponding imaginary rewards along collected histories, and infers the resulting belief sequences. These rewards and beliefs provide additional data for policy training. Figure 1 and Algorithm 1 summarise the construction.

3.1 WHY BLTM SUITS OFFLINE TASK INFERENCE

We observe that the conditional structure of BLTM rules out a specific shortcut in offline task inference. A collector's behaviour may correlate with task identity, but the posterior in equation 4 does not use that behaviour as separate evidence. Instead, BLTM evaluates each candidate task by how well it predicts the collected rewards and next states, given the corresponding states and actions. This property motivates our use of BLTM, rather than a different inference model.

Proposition 1 (Structural exclusion of collector evidence)**.** *Fix the model parameters η and a task prior independent of the collector. Under equation 4, the relative posterior density of two candidate tasks depends only on their prior densities and their conditional reward and next-state likelihoods. Collector action probabilities and state–action visitation likelihoods contribute no separate factor.*

Appendix A.3 provides the proof. However, different collected histories can still produce different beliefs. Offline policy training may therefore miss beliefs encountered during online adaptation, even when both phases use the same inference rule.

3.2 BAYESIAN TASK RELABELLING

A policy trained on contexts from collected data encounters only the beliefs inferred from those contexts. During online adaptation, the policy may encounter unfamiliar beliefs even at previously seen states. This *belief distribution shift* calls for broader belief coverage in the policy's training data. BTR therefore samples imaginary tasks and generates their corresponding imaginary rewards along collected histories, replacing the original rewards. Sequential belief updates using these rewards can produce beliefs absent from the original training data. Including these beliefs in policy training

aims to reduce the likelihood of encountering unfamiliar beliefs during online adaptation. States, actions, and next states remain those from the collected data, so the construction requires neither new interaction nor simulated dynamics. We now detail this construction, from sampling an imaginary task to forming the policy-training tuples.

Let D_i denote the collected dataset for training task i. Each imaginary task retains task i's dynamics and changes only its rewards; dynamics need not be shared across training tasks. Keeping collected next states avoids introducing dynamics-prediction errors into belief updates and value targets. We sample a batch $B_i \subset D_i$ and infer the full-batch posterior b_{B_i}. Task sampling uses its reward component $p_{\eta_R}(\theta_R \mid B_i)$. The batch constrains which reward parameters are plausible, while the remaining posterior uncertainty allows us to sample alternatives. We write $\widehat{p}_{\eta_R}$ for the sampling approximation described in Appendix A.4 and draw the imaginary task's reward parameters as

$$\widetilde{\theta}_{R,i} \sim \widehat{p}_{\eta_R}(\theta_R \mid B_i). \tag{6}$$

Unlike many task relabelling methods, our task sampling is not restricted to existing training tasks or goals achieved in collected histories. For this imaginary task, we then generate rewards along M segments of history sampled from D_i, each with H steps. Write one collected segment of history as $(s_h, a_h, r_h, s_{h+1})_{h=0}^{H-1}$. At each step, the imaginary reward $\widetilde{r}_h$ replaces the collected reward r_h, while s_h, a_h, and s_{h+1} remain unchanged. We generate $\widetilde{r}_h$ from the imaginary task's BLTM likelihood:

$$\widetilde{r}_h \sim p_{\eta_R}\Big(r \mid s_h, a_h, s_{h+1}, \widetilde{\theta}_{R,i}\Big). \tag{7}$$

The same imaginary task supplies rewards across all M segments of history in this update. Reward noise is sampled independently at each step.

With the imaginary rewards in place, we infer the corresponding beliefs along each segment of history. Let $\widetilde{b}_0$ denote the initial belief for this inference, with initialisation details in Appendix A.4. Starting from $\widetilde{b}_0$, we update the belief after each imaginary reward using that reward and the collected next state:

$$\widetilde{b}_{h+1} = U_\eta(\widetilde{b}_h, (s_h, a_h, \widetilde{r}_h, s_{h+1})). \tag{8}$$

The updated belief $\widetilde{b}_{h+1}$ incorporates the imaginary reward $\widetilde{r}_h$ and collected next state s_{h+1}. It supplies the task information for the next decision. We attach these before/after beliefs to each relabelled interaction tuple and add the complete tuple

$$(s_h, \widetilde{b}_h, a_h, \widetilde{r}_h, s_{h+1}, \widetilde{b}_{h+1})$$

to a batch of relabelled data for policy training. Training on these tuples exposes the policy to the additional beliefs inferred from imaginary rewards. Section 3.3 explains how these tuples enter the actor and critic updates.

3.3 Policy training and online adaptation

We train an actor and critic with SAC (Haarnoja et al., 2018) on batches mixing original data with the relabelled tuples from Section 3.2. Following GLiBRL (You & Kurniawati, 2025), we use posterior-mean features as the belief summary $g(b_C)$. The actor $\pi(a \mid s, g(b_C))$ uses the state and this belief summary. To help the critic estimate returns, we use an asymmetric critic (Pinto et al., 2018) with task information from sampled parameters or full-batch estimates that the actor's context alone does not provide. We supply this information as an additional task summary u_i to the critic $Q(s, g(b_C), a, u_i)$. For relabelled tuples, u_i summarises the sampled reward model and, when used, dynamics estimates from B_i. For original tuples, both components come from the full-batch posterior b_{B_i}. Appendix A.4 specifies these input representations. The actor, on the other hand, must instead only rely on its belief because the task is unknown during online adaptation.

To train these networks, we pair each tuple's reward with the beliefs used at its current and next state. For original data, we infer b_{C_i} from a smaller context $C_i \subset B_i$ containing original rewards. Each original tuple retains its reward and uses b_{C_i} at both s and s'. Each relabelled tuple instead uses $\widetilde{r}_h$, with $\widetilde{b}_h$ at s_h and $\widetilde{b}_{h+1}$ at s_{h+1}, as defined in equation 8. We apply SAC updates to the combined batch. The critic's target combines each tuple's reward with its discounted next-state value. This value uses the next-state belief and the same u_i as the current-state prediction. The actor maximises

the critic's value plus the SAC entropy term. The task model, in contrast, is trained only on collected data via equation 5. Appendix D.1 gives the training pseudocode.

Online adaptation starts from $C = \varnothing$ and $b_C = p(\theta)$. After each action from π, the agent appends (s, a, r, s') to C and updates b_C with U_η. It then acts from s' and the updated belief. Network parameters remain fixed. Each subsequent episode of the same task starts from the latest belief.

4 EXPERIMENTS

We evaluate BTR's adaptation to unseen tasks after offline training on eight environments with varying rewards or dynamics. We compare online adaptation with offline-context evaluation, then test generalisation beyond the training goal range.

4.1 EXPERIMENTAL SETUP

Tasks and datasets. We follow the exact experiment settings in the state-of-the-art method, MetaSTAR (Qian et al., 2026), in our experiments. We evaluate eight environments in total, including Point-Robot-Sparse (Rakelly et al., 2019) and seven MuJoCo locomotion environments (Todorov et al., 2012): Ant-Dir, Cheetah-Vel, Hopper-Param, Walker-Param, and sparse-reward variants of the latter three. Ant-Dir, both Cheetah variants, and Point-Robot-Sparse have uncertain reward functions but fixed dynamics. Tasks differ in target direction, velocity, or goal position. Both Hopper variants and both Walker variants have uncertain dynamics but a fixed reward function. Tasks differ in physical parameters. Episodes last 200 steps, or 60 steps for Point-Robot-Sparse. Following Qian et al. (2026), we collect fixed datasets with task-specific SAC agents for 40 tasks and use the following splits:

- *Standard task split:* We use 30 tasks for training and hold out 10 for testing.

- *ID/OOD split:* We restrict training to narrower goal-angle ranges for Ant-Dir and Point-Robot, and goal-velocity ranges for both Cheetah variants. Five tasks within these ranges are held out for in-distribution (ID) testing. All tasks outside these ranges are held out for out-of-distribution (OOD) testing. Thus, both ID and OOD results evaluate unseen tasks, but only OOD tasks have goals outside the training range. For dynamics-varying environments, we report the standard split's held-out tasks as ID; no OOD split is used.

Appendix D.2 and D.3 specify the details of tasks, the split and the data collection settings.

Evaluation. In *online adaptation*, the agent collects rewards and next states and updates its task representation while acting. Episode 1 begins with 100 random actions, or 30 on Point-Robot-Sparse, before switching to the learned policy. The representation carries over between episodes. In *offline-context evaluation*, the agent acts for one episode with a fixed representation inferred from 100 interaction tuples, or 30 on Point-Robot-Sparse, sampled from the held-out task's dataset.

Methods and reporting. We select the three strongest baselines in MetaSTAR (Qian et al., 2026) for their state-of-the-art results on this benchmark: MetaSTAR, CORRO (Yuan & Lu, 2022) and UNICORN (Li et al., 2024).[1] All methods use the same datasets and task splits, with network parameters fixed during evaluation. Appendix D.4 gives implementation details and hyperparameters. We use three seeds for all return comparisons, following FOCAL (Li et al., 2020) and MetaSTAR. Tables report mean return $\pm$ standard deviation. Curves show the mean with shading spanning $\pm$ standard deviation across seeds.

4.2 PERFORMANCE ON HELD-OUT TASKS

On the standard task split, BTR achieves the highest mean online return on all eight environments, spanning reward-varying and dynamics-varying tasks (Table 1). It reaches 511 on Ant-Dir versus 440 for UNICORN, and 103 on Walker-Param-Sparse versus 83 for MetaSTAR. On Cheetah-Vel,

[1]Direct comparison with methods that restrict relabelling to training tasks require further assumptions: BOReL (Dorfman et al., 2021) requires known reward functions, while GENTLE (Zhou et al., 2024) uses expert data in its policy training. Instead, in Section 4.4 we ablate on the benefit of sampling tasks from the learned posterior rather than restricting relabelling to training tasks on BTR.

Table 1: Held-out-task return. Online: episode 2 with agent-collected context; offline: one episode with a fixed representation from dataset context. Bold/underline: highest/second-highest mean.

	BTR (ours)		MetaSTAR		UNICORN		CORRO	
Environment	online	offline	online	offline	online	offline	online	offline
Ant-Dir	**511 ± 24**	**522 ± 17**	72 ± 68	285 ± 81	440 ± 30	511 ± 41	13 ± 14	14 ± 19
Cheetah-Vel	**-27 ± 2**	**-26 ± 2**	-40 ± 0	-122 ± 37	-52 ± 2	-31 ± 2	-62 ± 9	-33 ± 2
Cheetah-Vel-Sparse	**327 ± 6**	**320 ± 3**	157 ± 9	293 ± 5	318 ± 13	315 ± 6	262 ± 68	312 ± 10
Hopper-Param	**341 ± 22**	**339 ± 20**	312 ± 12	309 ± 17	312 ± 5	321 ± 10	290 ± 8	299 ± 11
Hopper-Param-Sparse	**105 ± 4**	**108 ± 1**	100 ± 1	95 ± 4	96 ± 1	99 ± 1	87 ± 1	87 ± 0
Walker-Param	**450 ± 48**	**468 ± 45**	434 ± 18	372 ± 134	335 ± 10	349 ± 25	427 ± 70	420 ± 83
Walker-Param-Sparse	**103 ± 1**	**105 ± 3**	83 ± 0	73 ± 5	82 ± 0	82 ± 0	75 ± 9	80 ± 2
Point-Robot-Sparse	**29 ± 7**	41 ± 5	20 ± 4	38 ± 5	19 ± 2	**50 ± 2**	10 ± 9	28 ± 2

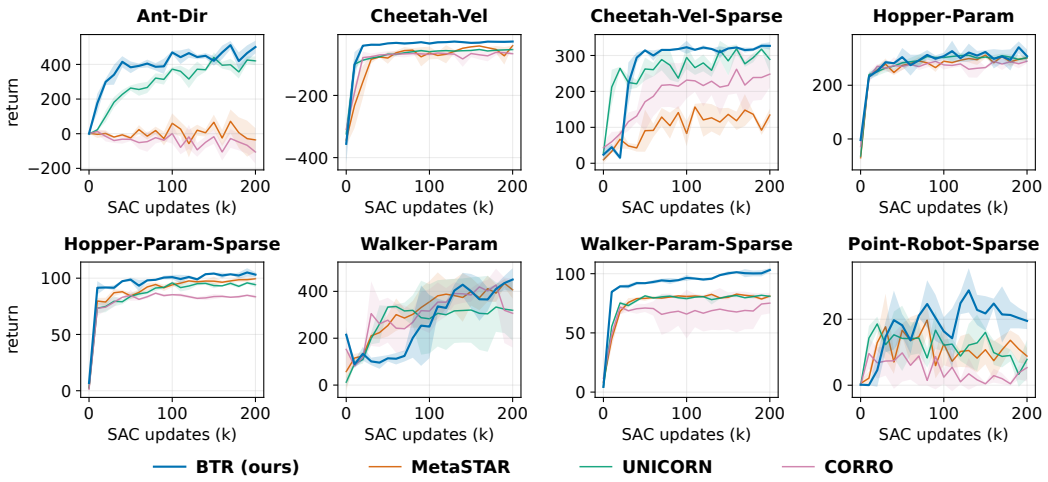

Figure 2: Episode-2 online return on held-out tasks against SAC updates across all eight environments.

it improves the best baseline's -40 to -27. On Point-Robot-Sparse, it reaches 29 versus 20 for MetaSTAR. With offline context, BTR achieves the highest mean return on seven environments and the second-highest on Point-Robot-Sparse. Figure 2 and Figure 4 show online and offline-context training curves, respectively.

4.3 Generalisation beyond the training goal range

We test generalisation to goals outside the training range on the ID/OOD split. We evaluate five online adaptation episodes and report fifth-episode return. From Table 2, on ID goals, BTR leads on all environments. On OOD goals, BTR leads on Ant-Dir and Cheetah-Vel-Sparse, reaching 308 versus 58 for MetaSTAR and 241 versus 221 for UNICORN. Figure 3 shows adaptation across five episodes. For completeness, we report the corresponding offline results in Table 3. As there is no OOD split for dynamics-varying environments, we report their offline results in Table 3, online adaptation results in Table 4 and Figure 5, both with their standard task split as in Section 4.2.

4.4 Ablations

Our ablations remove relabelling or critic task information, restrict sampling to training tasks, or add next-state sampling.

Relabelling's largest online gains on the standard task split are on Ant-Dir (366 to 511) and Walker-Param (216 to 450) (Table 5). On the ID/OOD split (Table 6), it raises OOD return from -67 to -54 on Cheetah-Vel and from 159 to 241 on Cheetah-Vel-Sparse, with a slight decrease in their ID means. This suggests a trade-off between broader task coverage and in-distribution specialisation. The

Table 2: Fifth-episode online return on unseen goals within (ID) and outside (OOD) the training range. Bold/underline: highest/second-highest mean.

	BTR		MetaSTAR		UNICORN		CORRO	
Environment	ID	OOD	ID	OOD	ID	OOD	ID	OOD
Ant-Dir	**518 ± 14**	**308 ± 45**	171 ± 71	<u>58 ± 65</u>	<u>434 ± 40</u>	-9 ± 6	116 ± 43	39 ± 83
Cheetah-Vel	**-24 ± 1**	<u>-54 ± 6</u>	<u>-33 ± 4</u>	**-48 ± 5**	-43 ± 4	-102 ± 17	-67 ± 8	-192 ± 79
Cheetah-Vel-Sparse	**363 ± 3**	**241 ± 5**	269 ± 20	113 ± 13	<u>356 ± 10</u>	<u>221 ± 19</u>	144 ± 42	98 ± 28
Point-Robot-Sparse	**28 ± 5**	7 ± 0	<u>15 ± 11</u>	<u>8 ± 4</u>	14 ± 5	**11 ± 3**	11 ± 1	4 ± 4

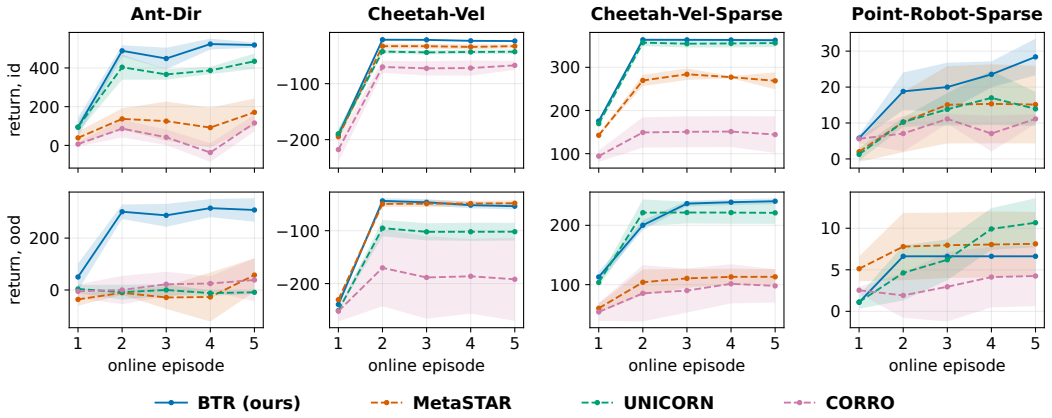

Figure 3: Five-episode adaptation on ID (top) and OOD (bottom) goals. Beliefs carry across episodes; episode 1 includes random actions, following MetaSTAR.

asymmetric critic improves online return on all eight standard-split environments and all four OOD goal sets. Interestingly, reward relabelling in BTR also improves mean return on all four dynamics-varying Hopper and Walker variants without relabelling next states (Table 5). Relabelled histories train the policy on dynamics beliefs updated from collected next states, beyond the fixed-context beliefs used with original data. This broader belief coverage may prepare the policy for incomplete dynamics knowledge during online adaptation.

Sampling beyond training tasks also improves on relabelling with existing tasks from our results in Tables 8 and 9. Following the training-task restriction in BOReL (Dorfman et al., 2021) and GENTLE (Zhou et al., 2024), the *existing task* variant of BTR re-uses another training task's posterior-mean reward parameters for relabelling. We observe that the posterior task sampling in BTR raises OOD return from 89 to 308 on Ant-Dir, -172 to -54 on Cheetah-Vel, and 124 to 241 on Cheetah-Vel-Sparse.

Generating imaginary next states alongside rewards, on the other hand, lowers return in seven of eight environments, including all four dynamics-varying environments (Table 7). Online return falls from 511 to 82 on Ant-Dir and 450 to 239 on Walker-Param. Next-state prediction errors may cause this drop by affecting belief updates and value targets. Reward-only relabelling avoids this error source by retaining collected next states.

5 Related Work

5.1 Online Meta-RL

Online Meta-RL learns adaptation strategies through interaction with training tasks. Gradient-based methods such as MAML learn an initialisation for rapid policy updates (Finn et al., 2017). Black-box methods such as RL^2 adapt through recurrent summaries of past interactions (Duan et al., 2016; Beck et al., 2023). Task-inference methods condition policies on inferred latent task variables. PEARL samples these variables from a learned approximate posterior, while VariBAD supplies the policy

with a representation of task uncertainty (Rakelly et al., 2019; Zintgraf et al., 2019). GLiBRL computes the posterior under its task model in closed form (You & Kurniawati, 2025). The learned features and Bayesian linear inference in BLTM build on ALPaCA (Harrison et al., 2020). ALPaCA learns a feature map and a linear-model prior for closed-form Bayesian regression. GLiBRL extends this construction to reward and dynamics models, including inference over model noise (You & Kurniawati, 2025). We inherit its BLTM and inference rule to generate rewards and beliefs for offline policy training.

Online Meta-RL also uses task generation and relabelling to create additional training data. LDM mixes training-task representations with randomly sampled weights and decodes imaginary rewards (Lee & Chung, 2021). TAVT similarly mixes task embeddings, generates rewards and next states, and penalises differences between the mixed embedding and the representation inferred from generated context (Kim et al., 2025). HTR relabels episodes for achieved goals using a known reward function (Packer et al., 2021). Hindsight Foresight Relabelling (HFR) instead selects training tasks according to predicted return after adapting with the relabelled history, and also tests relabelling with learned reward functions (Wan et al., 2022). All four methods collect online experience during training. BTR instead samples imaginary tasks using a posterior-based approximation fitted to offline data. It generates their rewards and updates beliefs without mixing task representations, restricting task choices to training tasks or achieved goals, or requiring supplied reward functions.

5.2 Offline Meta-RL

Offline Meta-RL learns from fixed task datasets. MACAW adapts policy and value-function parameters through advantage-weighted regression (Mitchell et al., 2020), unlike BTR's belief updates. Meta-DT conditions a sequence policy on world-model task representations and history prompts selected by prediction error (Wang et al., 2024). Representation-based methods such as FOCAL and CORRO learn task representations through metric and contrastive learning, respectively (Li et al., 2020; Yuan & Lu, 2022). CSRO and ER-TRL regularise representations to reduce dependence on collector behaviour (Gao et al., 2023; Nakhaei et al., 2024). UNICORN relates representation objectives through an information-theoretic view (Li et al., 2024). BTR instead uses BLTM's conditional reward and next-state likelihoods to exclude collector behaviour as separate evidence of task identity.

Other methods address the mismatch between offline training data and online adaptation. SMAC collects reward-free online experience and assigns learned rewards (Pong et al., 2022). IDAQ selects adaptation episodes to obtain in-distribution context (Wang et al., 2023). MerPO and MetaSTAR generate model rollouts for policy training (Lin et al., 2022; Qian et al., 2026). BTR instead generates imaginary rewards and infers their corresponding beliefs along collected histories, requiring no new interaction.

Like HFR, BOReL (Dorfman et al., 2021) and GENTLE (Zhou et al., 2024) selects existing training tasks for relabelling. BTR instead samples imaginary tasks from a learned task posterior and generates their rewards. We test this sampling distinction with the existing-task ablation in Section 4.4. CORRO generates alternative rewards and next states as contrastive negatives, including a variant with separate models for each training task (Yuan & Lu, 2022). These negatives train its task encoder, while policy training uses collected data. BTR connects task sampling, reward generation, and belief updating through one Bayesian task model. Each task draw supplies imaginary rewards across a segment of history, producing reward–belief tuples for policy training.

6 Conclusion

We introduced BTR, an offline Meta-RL framework built on BLTM's task inference, which excludes collector behaviour as separate evidence of task identity. BTR samples imaginary tasks using a posterior-based approximation, generates their rewards along collected histories, and infers additional beliefs for policy training. This broadens belief coverage without known reward functions or new interaction. Task sampling extends beyond training tasks and achieved goals. BTR achieves the highest mean return on all eight benchmarks under online adaptation and seven under offline-context evaluation. Ablations show the benefits of relabelling and the asymmetric critic, and support sampling beyond training-task estimates for OOD generalisation.

AI USE STATEMENT

In this work, we use AI (Claude, Anthropic; ChatGPT, OpenAI) for polishing the writing and implementing the code for experiments. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses simulated control benchmarks only; it involves no human subjects, personal data or sensitive applications. The datasets were regenerated from published collection code and will be released with the paper. We see no direct ethical concerns beyond those common to reinforcement learning research.

REPRODUCIBILITY STATEMENT

All eight datasets are regenerated deterministically from the published MetaSTAR collection protocol, The code is provided as supplementary material, and the datasets will be released once the review period concludes. Training settings and selected per-environment configurations are listed in Appendix D.4.

REFERENCES

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A survey of meta-reinforcement learning. *arXiv preprint arXiv:2301.08028*, 2023. URL https://arxiv.org/abs/2301.08028v1.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL https://github.com/jax-ml/jax.

Ron Dorfman, Idan Shenfeld, and Aviv Tamar. Offline meta reinforcement learning – identifiability challenges and effective data collection strategies. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 4607–4618. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/248024541dbda1d3fd75fe49d1a4df4d-Paper.pdf.

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. arXiv 2016.

Michael O'Gordon Duff. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. PhD thesis, University of Massachusetts Amherst, 2002. URL https://moduff.github.io/DuffThesis.pdf.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017. ICML 2017.

Yunkai Gao, Rui Zhang, Jiaming Guo, Fan Wu, Qi Yi, Shaohui Peng, Siming Lan, Ruizhi Chen, Zidong Du, Xing Hu, Qi Guo, Ling Li, and Yunji Chen. Context shift reduction for offline meta-reinforcement learning. *arXiv preprint arXiv:2311.03695*, 2023. NeurIPS 2023.

Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 8(5-6):359–483, 2015. doi: 10.1561/2200000049. URL https://doi.org/10.1561/2200000049.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018. ICML 2018.

James Harrison, Apoorva Sharma, and Marco Pavone. Meta-learning priors for efficient online bayesian regression. In *Algorithmic Foundations of Robotics XIII*, pp. 318–337. Springer International Publishing, 2020. doi: 10.1007/978-3-030-44051-0_19. URL https://doi.org/10.1007/978-3-030-44051-0_19.

Jeongmo Kim, Yisak Park, Minung Kim, and Seungyul Han. Task-aware virtual training: Enhancing generalization in meta-reinforcement learning for out-of-distribution tasks. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 30720–30748. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/kim25ag.html.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Suyoung Lee and Sae-Young Chung. Improving generalization in meta-rl with imaginary tasks from latent dynamics mixture. *arXiv preprint arXiv:2105.13524*, 2021. NeurIPS 2021.

Lanqing Li, Rui Yang, and Dijun Luo. Focal: Efficient fully-offline meta-reinforcement learning via distance metric learning and behavior regularization. *arXiv preprint arXiv:2010.01112*, 2020. ICLR 2021.

Lanqing Li, Hai Zhang, Xinyu Zhang, Shatong Zhu, Yang Yu, Junqiao Zhao, and Pheng-Ann Heng. Towards an information theoretic framework of context-based offline meta-reinforcement learning. *arXiv preprint arXiv:2402.02429*, 2024. NeurIPS 2024.

Sen Lin, Jialin Wan, Tengyu Xu, Yingbin Liang, and Junshan Zhang. Model-based offline meta-reinforcement learning with regularization. *arXiv preprint arXiv:2202.02929*, 2022. ICLR 2022.

Eric Mitchell, Rafael Rafailov, Xue Bin Peng, Sergey Levine, and Chelsea Finn. Offline meta-reinforcement learning with advantage weighting. *arXiv preprint arXiv:2008.06043*, 2020. ICML 2021.

Mohammadreza Nakhaei, Aidan Scannell, and Joni Pajarinen. Entropy regularized task representation learning for offline meta-reinforcement learning. *arXiv preprint arXiv:2412.14834*, 2024. AAAI 2025.

Charles Packer, Pieter Abbeel, and Joseph E. Gonzalez. Hindsight task relabelling: Experience replay for sparse reward Meta-RL, 2021. URL https://arxiv.org/abs/2112.00901.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018. doi: 10.15607/RSS.2018.XIV.008. URL https://www.roboticsproceedings.org/rss14/p08.html.

Vitchyr H Pong, Ashvin V Nair, Laura M Smith, Catherine Huang, and Sergey Levine. Offline meta-reinforcement learning with online self-supervision. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning*

Research, pp. 17811–17829. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/pong22a.html.

Fuyuan Qian, Menglong Zhang, Song Wang, and Quanying Liu. Behavior-invariant task representation learning with transformer-based world models for offline meta-reinforcement learning. *arXiv preprint arXiv:2606.00780*, 2026. ICML 2026.

Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. *arXiv preprint arXiv:1903.08254*, 2019. ICML 2019.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL https://www.jmlr.org/papers/v9/vandermaaten08a.html.

Michael Wan, Jian Peng, and Tanmay Gangwani. Hindsight foresight relabeling for meta-reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=P7OVkHEoHOZ.

Jianhao Wang, Jin Zhang, Haozhe Jiang, Junyu Zhang, Liwei Wang, and Chongjie Zhang. Offline meta reinforcement learning with in-distribution online adaptation. *arXiv preprint arXiv:2305.19529*, 2023. ICML 2023.

Zhi Wang, Li Zhang, Wenhao Wu, Yuanheng Zhu, Dongbin Zhao, and Chunlin Chen. Meta-dt: Offline meta-rl as conditional sequence modeling with world model disentanglement. *arXiv preprint arXiv:2410.11448*, 2024. NeurIPS 2024.

Jingyang You and Hanna Kurniawati. Meta-RL with bayesian linear task models, 2025. URL https://arxiv.org/abs/2512.20974.

Haoqi Yuan and Zongqing Lu. Robust task representations for offline meta-reinforcement learning via contrastive learning. *arXiv preprint arXiv:2206.10442*, 2022. ICML 2022.

Renzhe Zhou, Chen-Xiao Gao, Zongzhang Zhang, and Yang Yu. Generalizable task representation learning for offline meta-reinforcement learning with data limitations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(15):17132–17140, 2024. doi: 10.1609/aaai.v38i15.29658. URL https://ojs.aaai.org/index.php/AAAI/article/view/29658.

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. ICLR 2020.

A Inference and theoretical analysis

A.1 Closed-form belief updates

The reward belief is parameterised by (M_R, Ξ, ν, ω): the posterior mean of the reward-function parameters, their precision factor, and two noise-posterior parameters. The transition belief has the analogous parameters for each output dimension. The learned model parameters are held fixed during each inference calculation. We abbreviate the update U_η introduced in Section 2.3 as U.

Closed-form update. For a reward feature $\phi_R = \phi_R(s, a, s')$ and target $y = r$, conjugacy gives

$$\begin{aligned} \Xi' &= \Xi + \phi_R \phi_R^\top, \quad M_R' = \Xi'^{-1}(\Xi M_R + \phi_R y), \\ \nu' &= \nu + 1, \qquad\qquad \omega' = \omega + y^2 + M_R^\top \Xi M_R - M_R'^\top \Xi' M_R'. \end{aligned} \tag{9}$$

The implementation evaluates this recursion by caching Ξ^{-1}, giving a cost of $O(d_R^2)$ per reward update. The same calculation applies to each next-state coordinate. Applying these updates from the prior gives the context belief b_C. In the reported configurations, the actor uses posterior-mean features without an uncertainty input; Appendix A.4 defines this summary.

For n interaction tuples with fixed features, write $y_i = \phi_{R,i}^\top \mu_R^\star + \varepsilon_i$, where $\mu_R^\star$ is the true reward-function parameter vector and ε_i is the reward noise. With $G_n = \sum_{i=1}^n \phi_{R,i} \phi_{R,i}^\top$, the posterior mean satisfies

$$M_{R,n} - \mu_R^\star = (\Xi_0 + G_n)^{-1} \left[\Xi_0 (M_{R,0} - \mu_R^\star) + \sum_i \phi_{R,i} \varepsilon_i \right]. \tag{10}$$

A.2 Closed-form marginal likelihood

We derive the normalising integral in equation 5 for the likelihoods and priors in equation 2–equation 3. The reward model has one scalar output. Since the dynamics noise covariance is diagonal, the same calculation applies independently to each next-state coordinate.

One scalar-output regression. For a batch B of n interaction tuples, let $\Phi \in \mathbb{R}^{n \times d}$ contain the evaluated feature vectors as rows and let $y \in \mathbb{R}^n$ contain the targets. Write $\mu \in \mathbb{R}^d$ for the function parameters and $\tau = \sigma^{-1}$ for the noise precision. Suppressing the reward or dynamics subscripts, the model is

$$\begin{aligned} p(y \mid \Phi, \mu, \tau) &= \mathcal{N}(y; \Phi\mu, \tau^{-1} I_n), \\ p(\mu \mid \tau) &= \mathcal{N}(\mu; M, (\tau\Xi)^{-1}), \\ p(\tau) &= \frac{(\omega/2)^{\nu/2}}{\Gamma(\nu/2)} \tau^{\nu/2 - 1} \exp(-\omega\tau/2). \end{aligned} \tag{11}$$

Here, Ξ is positive definite, $\nu, \omega > 0$, I_n is the identity matrix, and Γ is the Gamma function. All integrations below use the precision variable τ, consistent with equation 3.

The batch posterior has parameters

$$\begin{aligned} \Xi_B &= \Xi + \Phi^\top \Phi, \quad M_B = \Xi_B^{-1}(\Xi M + \Phi^\top y), \\ \nu_B &= \nu + n, \qquad\quad \omega_B = \omega + y^\top y + M^\top \Xi M - M_B^\top \Xi_B M_B. \end{aligned} \tag{12}$$

These are the batch form of equation 9. Completing the square gives

$$\|y - \Phi\mu\|^2 + (\mu - M)^\top \Xi (\mu - M) + \omega = (\mu - M_B)^\top \Xi_B (\mu - M_B) + \omega_B. \tag{13}$$

Integrating the task parameters. First integrate the Gaussian-distributed function parameters μ. The factor $\exp[-\tau(\mu - M_B)^\top \Xi_B (\mu - M_B)/2]$ integrates over μ to $(2\pi)^{d/2} \tau^{-d/2} |\Xi_B|^{-1/2}$, where $|\cdot|$ denotes a determinant. Combining this with the likelihood and prior normalisers yields

$$\int p(y \mid \Phi, \mu, \tau) p(\mu \mid \tau)\, d\mu = (2\pi)^{-n/2} \tau^{n/2} \frac{|\Xi|^{1/2}}{|\Xi_B|^{1/2}} \exp\Big[-\frac{\tau}{2} (\omega_B - \omega) \Big]. \tag{14}$$

Multiplying by the Gamma prior and integrating the precision then gives

$$\begin{aligned} p(y \mid \Phi) &= (2\pi)^{-n/2} \frac{|\Xi|^{1/2}}{|\Xi_B|^{1/2}} \frac{(\omega/2)^{\nu/2}}{\Gamma(\nu/2)} \int_0^\infty \tau^{\nu_B/2-1} \exp(-\omega_B \tau/2)\, d\tau \\ &= \pi^{-n/2} \frac{|\Xi|^{1/2}}{|\Xi_B|^{1/2}} \frac{\Gamma(\nu_B/2)}{\Gamma(\nu/2)} \frac{\omega^{\nu/2}}{\omega_B^{\nu_B/2}}. \end{aligned} \tag{15}$$

The second line uses the Gamma integral $\int_0^\infty t^{a-1} e^{-bt}\, dt = \Gamma(a) b^{-a}$ for $a, b > 0$. Thus the scalar-output log marginal likelihood is

$$\begin{aligned} \ell(\Phi, y; M, \Xi, \nu, \omega) :=\ & \log p(y \mid \Phi) \\ =\ & -\frac{n}{2} \log \pi + \frac{1}{2} \big(\log |\Xi| - \log |\Xi_B| \big) \\ & + \log \Gamma(\nu_B/2) - \log \Gamma(\nu/2) + \frac{\nu}{2} \log \omega - \frac{\nu_B}{2} \log \omega_B. \end{aligned} \tag{16}$$

Combining rewards and dynamics. For $B = \{(s_l, a_l, r_l, s'_l)\}_{l=1}^n$, the reward design matrix Φ_R has rows $\phi_R(s_l, a_l, s'_l)^\top$, and $y_R = (r_l)_{l=1}^n$. The dynamics design matrix Φ_T has rows $\phi_T(s_l, a_l)^\top$, and $y_{T,j} = (s'_{l,j})_{l=1}^n$ contains the targets for coordinate j.

Let $M_{T,:,j}$ be column j of the dynamics prior mean. Define

$$\begin{aligned} \ell_R(B) &= \ell(\Phi_R, y_R; M_R, \Xi_R, \nu_R, \omega_R), \\ \ell_{T,j}(B) &= \ell(\Phi_T, y_{T,j}; M_{T,:,j}, \Xi_T, \nu_{T,j}, \omega_{T,j}). \end{aligned}$$

The reward and dynamics priors are independent. The diagonal column covariance also makes the columns of μ_T independent, with independent Gamma precisions. Hence the integral used in equation 5 factorises as

$$\log Z_\eta(B) := \log \int p(\theta) p_\eta(B \mid \theta)\, d\theta = \ell_R(B) + \sum_{j=1}^{d_s} \ell_{T,j}(B). \tag{17}$$

This integrates the conditional outcome likelihoods evaluated on the collected tuples. It does not introduce a likelihood for the collector's action choices.

Feature-learning loss. For fixed prior parameters and batch size, only the posterior determinant and scale terms in equation 16 depend on η. The contribution of one batch to the negative log marginal likelihood therefore simplifies to

$$\begin{aligned} -\frac{1}{n} \log Z_\eta(B) = \frac{1}{2n} \Big[& \log |\Xi_{R,B}| + \nu_{R,B} \log \omega_{R,B} \\ & + d_s \log |\Xi_{T,B}| + \sum_{j=1}^{d_s} \nu_{T,j,B} \log \omega_{T,j,B} \Big] + c, \end{aligned} \tag{18}$$

where c is independent of η. The dynamics determinant appears d_s times because each output uses the same feature precision $\Xi_{T,B}$. Averaging over task batches and adding the squared-feature penalties gives equation 5, up to this constant. The implementation stores each noise scale as $\bar{\omega}_B = \omega_B / \nu_B$ and uses $\log(\bar{\omega}_B/2)$ in the scale term. This changes only the discarded constant when the prior is fixed. The full expression in equation 16 retains all prior normalisers.

A.3 Proof of structural exclusion

For Proposition 1, fix η and the collector-independent prior. Let $\theta = (\theta_R, \theta_T)$ and $\vartheta = (\vartheta_R, \vartheta_T)$ be two candidate tasks with positive prior densities and likelihoods. Dividing their posterior densities in equation 4 cancels the common normalising constant:

$$\frac{b_C(\theta)}{b_C(\vartheta)} = \frac{p(\theta)}{p(\vartheta)} \prod_{(s,a,r,s') \in C} \frac{p_{\eta_T}(s' \mid s, a, \theta_T) p_{\eta_R}(r \mid s, a, s', \theta_R)}{p_{\eta_T}(s' \mid s, a, \vartheta_T) p_{\eta_R}(r \mid s, a, s', \vartheta_R)}. \tag{19}$$

Every data-dependent factor compares reward and next-state predictions at a collected state–action pair. There is no additional factor for the probability of the collector choosing that input. This proves the stated exclusion. □

The statement concerns the conditional belief defined in equation 4. It does not identify that belief with the full observational posterior for every offline collector.

A.4 Relabelling and policy-input details

Posterior-based task sampling. Conditional on σ_R, the reward posterior $p_{\eta_R}(\mu_R \mid \sigma_R, B_i)$ has precision $\sigma_R^{-1}\Xi_{R,B_i}$. Inverting this precision gives its covariance:

$$\mu_R \mid \sigma_R, B_i \sim \mathcal{N}\Big(M_{R,B_i}, \sigma_R \Xi_{R,B_i}^{-1}\Big),$$
$$\Xi_{R,B_i} = \Xi_R + \sum_{(s,a,r,s')\in B_i} \phi_R(s,a,s')\phi_R(s,a,s')^\top.$$

For reward posterior parameters $(M_{R,B_i}, \Xi_{R,B_i}, \nu_{R,B_i}, \omega_{R,B_i})$, the Gamma posterior over precision σ_R^{-1} gives the mean noise variance

$$\widehat{\sigma}_{R,i} = \mathbb{E}_{b_{B_i}}[\sigma_R] = \frac{\omega_{R,B_i}}{\nu_{R,B_i} - 2}, \qquad \nu_{R,B_i} > 2.$$

The posterior approximation $\widehat{p}_{\eta_R}$ in equation 6 fixes the noise variance at this posterior mean and samples the imaginary task's reward parameters as follows:

$$\widetilde{\mu}_{R,i} \sim \mathcal{N}\Big(M_{R,B_i}, \widehat{\sigma}_{R,i}\Xi_{R,B_i}^{-1}\Big), \qquad \widetilde{\theta}_{R,i} = (\widetilde{\mu}_{R,i}, \widehat{\sigma}_{R,i}).$$

The reward likelihood in equation 7 then becomes

$$\widetilde{r}_h \sim \mathcal{N}\big(\phi_R(s_h, a_h, s_{h+1})^\top \widetilde{\mu}_{R,i}, \widehat{\sigma}_{R,i}\big).$$

Starting beliefs and collected tuples. Each segment of history independently starts from the prior with probability 0.5 and from b_{C_i} otherwise. The context C_i contains k collected interaction tuples from B_i and retains their original rewards. It is not relabelled before initialising the segment of history. The starting belief therefore need not be the posterior used to sample the imaginary task. Equation 8 gives the filter's response to the imaginary rewards from that starting point.

For original policy-training tuples, b_{C_i} is used at both the current and next state. The implementation selects the first k rows of the sampled batch as context and masks the remaining original rows out of the policy loss. Relabelled tuples receive sequentially updated beliefs and are all included. Original terminal flags are retained for original tuples; relabelled tuples use non-terminal targets within the sampled fixed-horizon segments of history.

Actor and critic inputs. Write f_R and f_T for the deterministic feature transformations of reward-function and dynamics-function parameters. In the configurations described in the main text, the actor summary is

$$g(b_C) = \big[f_T(M_{T,C}),\ f_R(M_{R,C})\big],$$

with the dynamics component omitted when it is not used by the policy. Reward features are the raw or normalised posterior mean $M_{R,C}$. Dynamics features are the normalised, flattened matrix $M_{T,C}$. The environment-specific choices are listed in Table 12. None of the reported configurations includes posterior uncertainty in the actor input.

The critic receives this same state–belief input plus

$$u_i^{\text{rel}} = \big[f_R(\widetilde{\mu}_{R,i}),\ f_T(M_{T,B_i})\big], \qquad u_i^{\text{real}} = \big[f_R(M_{R,B_i}),\ f_T(M_{T,B_i})\big].$$

The dynamics component is again omitted when unused. Neither summary includes the noise variance. Within a critic update, the same additional summary is attached to the current and next state–belief pair. Actor actions are computed without this summary, including when evaluating them through the critic during training.

B ADDITIONAL RESULTS

B.1 OFFLINE-CONTEXT EVALUATION

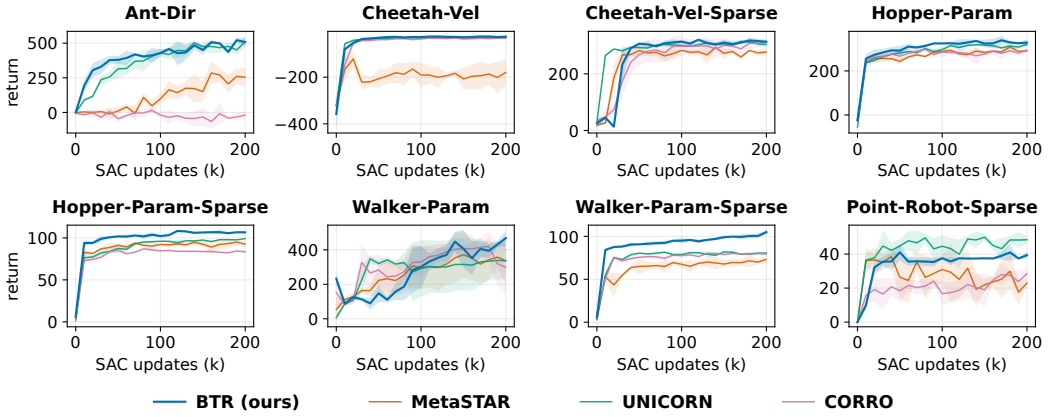

Figure 4: Offline-context return against SAC updates on the standard task split in all eight environments.

Table 3: Offline-context return on ID goals, OOD goals, and held-out dynamics tasks which do not have OOD split. Bold/underline: highest/second-highest mean. BTR leads on seven environments with ID split, and three tasks out of four OOD split.

	BTR		MetaSTAR		UNICORN		CORRO	
Environment	ID	OOD	ID	OOD	ID	OOD	ID	OOD
Ant-Dir	**540** $\pm$ **32**	**309** $\pm$ **57**	311 $\pm$ 106	152 $\pm$ 74	390 $\pm$ 169	-7 $\pm$ 5	106 $\pm$ 20	45 $\pm$ 25
Cheetah-Vel	**-19** $\pm$ **1**	**-44** $\pm$ **2**	-76 $\pm$ 15	-92 $\pm$ 3	-66 $\pm$ 54	-125 $\pm$ 95	-34 $\pm$ 14	-60 $\pm$ 8
Cheetah-Vel-Sparse	**369** $\pm$ **2**	**246** $\pm$ **27**	360 $\pm$ 10	122 $\pm$ 12	294 $\pm$ 113	146 $\pm$ 57	190 $\pm$ 141	116 $\pm$ 21
Point-Robot-Sparse	32 $\pm$ 15	11 $\pm$ 3	38 $\pm$ 20	6 $\pm$ 0	**51** $\pm$ **1**	**13** $\pm$ **0**	38 $\pm$ 13	10 $\pm$ 6
Hopper-Param	**339** $\pm$ **20**	/	309 $\pm$ 17	/	321 $\pm$ 10	/	299 $\pm$ 11	/
Hopper-Param-Sparse	**108** $\pm$ **1**	/	95 $\pm$ 3	/	99 $\pm$ 1	/	87 $\pm$ 0	/
Walker-Param	**468** $\pm$ **45**	/	372 $\pm$ 134	/	349 $\pm$ 25	/	420 $\pm$ 83	/
Walker-Param-Sparse	**105** $\pm$ **3**	/	73 $\pm$ 5	/	82 $\pm$ 1	/	80 $\pm$ 2	/

B.2 FIVE-EPISODE ONLINE ADAPTATION

Table 4: Fifth-episode online return on held-out ID dynamics tasks which do not have OOD split. Bold/underline: highest/second-highest mean. BTR leads on all environments.

Environment	BTR	MetaSTAR	UNICORN	CORRO
Hopper-Param	**329 ± 2**	305 ± 8	<u>314 ± 5</u>	298 ± 9
Hopper-Param-Sparse	**106 ± 2**	<u>100 ± 0</u>	99 ± 1	87 ± 0
Walker-Param	**448 ± 34**	<u>441 ± 17</u>	336 ± 111	431 ± 62
Walker-Param-Sparse	**103 ± 1**	<u>83 ± 0</u>	82 ± 1	73 ± 12

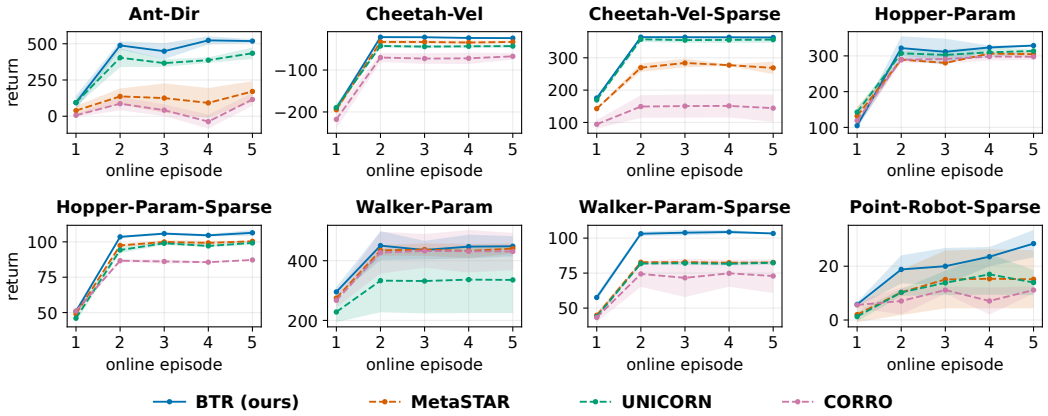

Figure 5: Five-episode online adaptation on all ID tasks, with beliefs carried across episodes. Episode 1 starts with 100 random actions, or 30 on Point-Robot-Sparse. BTR leads on all environments.

B.3 TASK REPRESENTATIONS

BTR builds on GLiBRL's task representations (You & Kurniawati, 2025) and contributes the Bayesian task-relabelling procedure. We include these visualisations for completeness, documenting the representations used in our experiments. Both analyses use seed 1 on the standard task split, with 20 sampled dataset contexts per task. Each context contains 100 interaction tuples, or 30 on Point-Robot-Sparse.

Figure 6 compares distances between training-task representations with differences in their goals. We average each task's representations across contexts before computing pairwise Euclidean distances. Tasks are ordered by goal. Ant-Dir's circular order is cut at the largest angular gap. Brighter cells indicate larger representation distances within each panel.

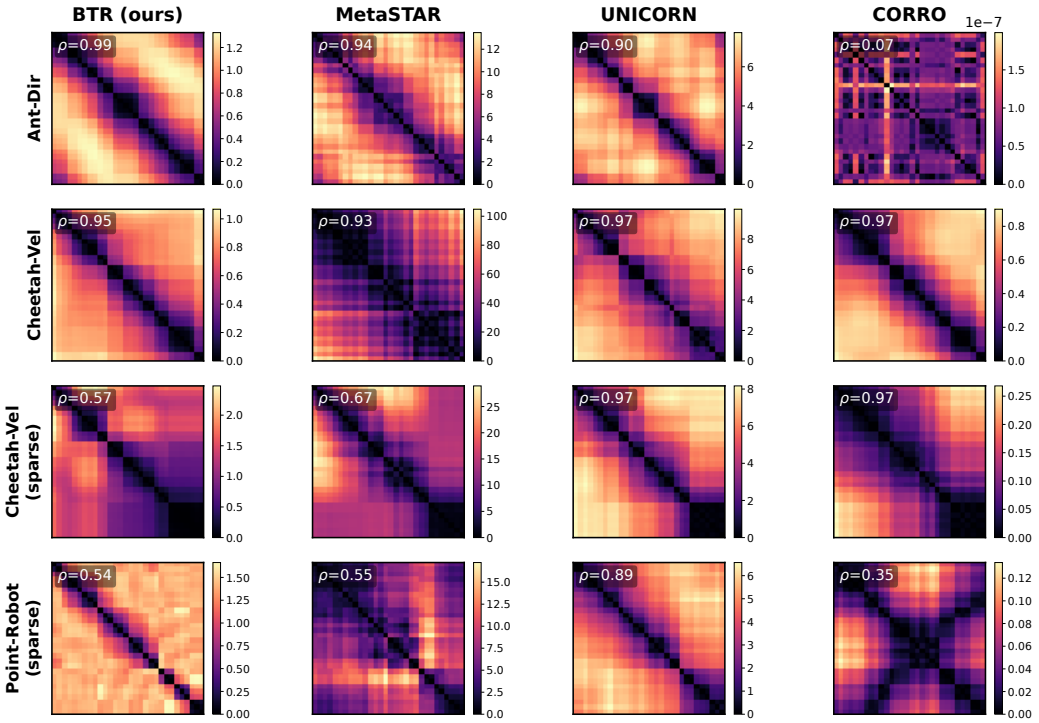

Figure 6: Distances between mean representations of 30 training tasks, ordered by goal. Rows: environments; columns: methods; colour scales: panel-specific. ρ: Spearman correlation with goal distances over unordered task pairs, using shortest circular distances for Ant-Dir and absolute scalar differences elsewhere.

For the ten held-out tasks, Figure 7 shows one point per context and one colour per task.

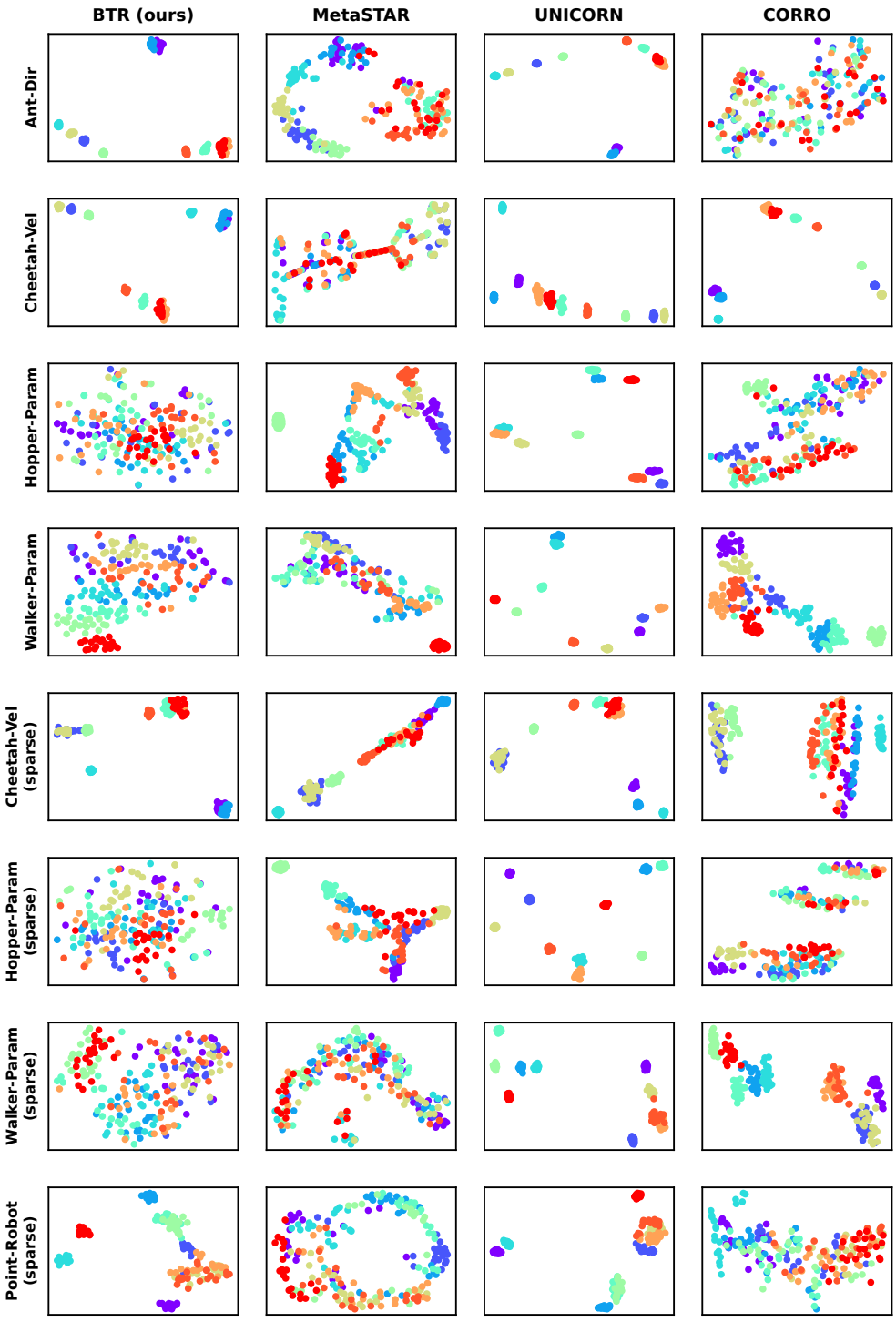

Figure 7: t-SNE projections (van der Maaten & Hinton, 2008) of original task representations across eight environments. Rows: environments; columns: methods. Each panel contains 20 contexts per task from ten held-out tasks.

C ABLATIONS

We test relabelling, the critic's additional task information, task sampling beyond the training set, and next-state sampling. The *no relabelling* variant uses only original tuples; the *symmetric critic* retains relabelling but omits u_i. The task-sampling comparisons follow these component ablations. Other settings are unchanged. Bold marks the highest unrounded mean within each protocol and test group.

Table 5: Single-component ablations on the standard task split. Online: episode 2; offline: fixed dataset context.

	online (episode 2)			offline context		
Environment	BTR	no relabelling	symmetric critic	BTR	no relabelling	symmetric critic
Ant-Dir	**511 ± 24**	366 ± 32	426 ± 12	**522 ± 17**	363 ± 29	443 ± 17
Cheetah-Vel	-27 ± 2	**-25 ± 2**	-33 ± 1	-26 ± 2	**-24 ± 1**	-32 ± 1
Cheetah-Vel-Sparse	327 ± 6	**332 ± 14**	295 ± 13	320 ± 3	**331 ± 3**	288 ± 1
Hopper-Param	**341 ± 22**	312 ± 15	335 ± 31	**339 ± 20**	326 ± 12	328 ± 5
Hopper-Param-Sparse	**105 ± 4**	98 ± 2	99 ± 1	**108 ± 1**	105 ± 1	99 ± 4
Walker-Param	**450 ± 48**	216 ± 71	354 ± 76	**468 ± 45**	223 ± 48	391 ± 66
Walker-Param-Sparse	**103 ± 1**	102 ± 2	98 ± 0	**105 ± 3**	99 ± 2	99 ± 2
Point-Robot-Sparse	**29 ± 7**	22 ± 9	21 ± 3	41 ± 5	42 ± 4	**42 ± 4**

Relabelling improves online means in six of eight environments, with the largest gains on Ant-Dir and Walker-Param (Table 5). Both Cheetah variants have lower online and offline-context means with relabelling. The asymmetric critic improves all eight online means.

Table 6: Single-component ablations on the ID/OOD split. Fifth-episode online return on ID and OOD goals.

	BTR		no relabelling		symmetric critic	
Environment	ID	OOD	ID	OOD	ID	OOD
Ant-Dir	**518 ± 14**	**308 ± 45**	374 ± 59	144 ± 14	484 ± 20	271 ± 31
Cheetah-Vel	-24 ± 1	**-54 ± 6**	**-22 ± 2**	-67 ± 13	-28 ± 1	-68 ± 9
Cheetah-Vel-Sparse	363 ± 3	**241 ± 5**	**365 ± 3**	159 ± 9	359 ± 8	172 ± 33
Point-Robot-Sparse	**28 ± 5**	**7 ± 0**	28 ± 5	**7 ± 5**	21 ± 9	4 ± 3

On the ID/OOD split, relabelling improves OOD means on Ant-Dir and both Cheetah variants (Table 6). The Cheetah OOD gains accompany two-point ID decreases. This pattern suggests a trade-off between broader task coverage and in-distribution specialisation. Point-Robot-Sparse retains its rounded OOD mean of 7. The asymmetric critic improves all four OOD means.

Table 7: Next-state sampling on the standard task split. BTR relabels rewards only; + next state also samples task dynamics and one-step next states at collected state–action pairs. Other settings are unchanged. Online: episode 2; offline: fixed dataset context.

	online (episode 2)		offline context	
Environment	BTR	+ next state	BTR	+ next state
Ant-Dir	**511 ± 24**	82 ± 18	**522 ± 17**	95 ± 13
Cheetah-Vel	**-27 ± 2**	-95 ± 15	**-26 ± 2**	-87 ± 7
Cheetah-Vel-Sparse	**327 ± 6**	235 ± 12	**320 ± 3**	231 ± 21
Hopper-Param	**341 ± 22**	278 ± 11	**339 ± 20**	295 ± 24
Hopper-Param-Sparse	**105 ± 4**	90 ± 4	**108 ± 1**	97 ± 3
Walker-Param	**450 ± 48**	239 ± 33	**468 ± 45**	239 ± 49
Walker-Param-Sparse	**103 ± 1**	84 ± 2	**105 ± 3**	82 ± 2
Point-Robot-Sparse	29 ± 7	**29 ± 2**	**41 ± 5**	41 ± 3

Tables 8 and 9 test whether relabelling benefits from sampling tasks beyond the training set. BOReL and GENTLE select another training task for reward relabelling (Dorfman et al., 2021; Zhou et al., 2024). We implement this task-selection restriction within BTR. For each source task, the *existing task* variant selects a different training task from the current batch.

Table 8: Posterior task sampling versus training-task-restricted relabelling on the standard task split. Online: episode 2; offline: fixed dataset context.

	online (episode 2)		offline context	
Environment	BTR	existing task	BTR	existing task
Ant-Dir	**511 ± 24**	354 ± 42	**522 ± 17**	389 ± 11
Cheetah-Vel	**-27 ± 2**	-77 ± 13	**-26 ± 2**	-86 ± 17
Cheetah-Vel-Sparse	**327 ± 6**	318 ± 5	320 ± 3	**321 ± 9**
Hopper-Param	**341 ± 22**	325 ± 24	**339 ± 20**	335 ± 3
Hopper-Param-Sparse	**105 ± 4**	105 ± 1	108 ± 1	**110 ± 1**
Walker-Param	**450 ± 48**	410 ± 128	**468 ± 45**	389 ± 158
Walker-Param-Sparse	**103 ± 1**	103 ± 2	**105 ± 3**	102 ± 1
Point-Robot-Sparse	29 ± 7	**37 ± 3**	**41 ± 5**	40 ± 2

On the standard split, posterior task sampling raises online return from 354 to 511 on Ant-Dir and from -77 to -27 on Cheetah-Vel. Their offline-context returns also improve, from 389 to 522 and from -86 to -26, respectively.

Table 9: Posterior task sampling versus training-task-restricted relabelling on the ID/OOD split. Fifth-episode online return on ID and OOD goals.

	BTR		existing task	
Environment	ID	OOD	ID	OOD
Ant-Dir	**518 ± 14**	**308 ± 45**	460 ± 41	89 ± 24
Cheetah-Vel	**-24 ± 1**	**-54 ± 6**	-36 ± 5	-172 ± 19
Cheetah-Vel-Sparse	**363 ± 3**	**241 ± 5**	358 ± 6	124 ± 20
Point-Robot-Sparse	**28 ± 5**	**7 ± 0**	27 ± 4	6 ± 5

Posterior task sampling improves OOD return on all four environments. The largest gains are on Ant-Dir (89 to 308), Cheetah-Vel (-172 to -54), and Cheetah-Vel-Sparse (124 to 241). These results support sampling beyond training-task estimates when generating data for policy training.

D IMPLEMENTATION DETAILS AND HYPERPARAMETERS

D.1 PSEUDOCODE

Algorithm 1 summarises BTR; Algorithm 2 gives the BLTM routines inherited from GLiBRL (You & Kurniawati, 2025). Inference updates the belief, while model learning updates η. Appendix A.4 defines the starting beliefs and policy inputs $g(b_C)$, u_i^{real}, and u_i^{rel}.

Algorithm 1 BTR: one offline training iteration

Require: Collected datasets $\{D_i\}$, prior $p(\theta)$, model parameters η, actor π, critic Q

Initialise a batch $\mathcal{T} \leftarrow \varnothing$ for policy training; sample task indices $\mathcal{B}$
for each $i \in \mathcal{B}$ **do**
  Sample $B_i \subset D_i$ and a smaller context $C_i \subset B_i$
  Infer b_{B_i} and b_{C_i} with Infer_η ▷ Algorithm 2
  Sample an imaginary task: $\widetilde{\theta}_{R,i} \sim \widehat{p}_{\eta_R}(\theta_R \mid B_i)$
  Form the critic summaries u_i^{real} and u_i^{rel} ▷ Appendix A.4
  for each of M sampled segments of history $(s_h, a_h, r_h, s_{h+1})_{h=0}^{H-1}$ from D_i **do**
    Initialise $\widetilde{b}_0$ as specified in Appendix A.4
    for $h = 0, \ldots, H-1$ **do**
      Generate $\widetilde{r}_h \sim p_{\eta_R}(r \mid s_h, a_h, s_{h+1}, \widetilde{\theta}_{R,i})$ in place of r_h
      Update $\widetilde{b}_{h+1} \leftarrow U_\eta(\widetilde{b}_h, (s_h, a_h, \widetilde{r}_h, s_{h+1}))$
      Add $(s_h, \widetilde{b}_h, a_h, \widetilde{r}_h, s_{h+1}, \widetilde{b}_{h+1}, u_i^{\text{rel}})$ to $\mathcal{T}$
    end for
  end for
  Add $(s, b_{C_i}, a, r, s', b_{C_i}, u_i^{\text{real}})$ to $\mathcal{T}$ for each $(s, a, r, s') \in C_i$
end for
Update $Q(s, g(b_C), a, u_i)$ and $\pi(a \mid s, g(b_C))$ with SAC on $\mathcal{T}$, holding η fixed
If scheduled, update η on fresh collected batches using ModelStep_η

Algorithm 2 BLTM belief inference and model learning

Require: Prior $p(\theta)$ and model parameters $\eta = (\eta_R, \eta_T)$

function $\text{Infer}_\eta(C)$
  Initialise $b \leftarrow p(\theta)$
  for each interaction tuple $(s, a, r, s') \in C$ **do**
    $b \leftarrow U_\eta(b, (s, a, r, s'))$ ▷ Closed-form update in Appendix A.1
  end for
  return $b_C \leftarrow b$
end function

function $\text{ModelStep}_\eta(\{B_i\}_{i \in \mathcal{B}})$
  Compute each batch's posterior parameters using equation 12
  Evaluate the log marginal likelihoods $\log Z_\eta(B_i)$ ▷ Appendix A.2
  Minimise $\mathcal{L}_{\text{model}}(\eta)$ in equation 5 with one gradient-based update
  return η
end function

D.2 Environments and task splits

Table 10 summarises the episode lengths, task parameters, and goal ranges. Reward definitions and dynamics perturbations below follow the collection implementation used for our datasets.

Table 10: Environment and task ranges. ID and OOD columns describe the ID/OOD split. Dynamics tasks use the standard held-out-task split instead. Physical parameters are multiplied by c^u, with $u \sim \mathcal{U}[-3, 3]$; $c = 1.3$ for damping and 1.5 otherwise.

Environment	Steps	Task parameter	Full range	ID range	OOD range
Ant-Dir	200	Direction angle	$[0, 2\pi)$	$[\pi/4, 7\pi/4]$	$[0, \pi/4) \cup (7\pi/4, 2\pi)$
Cheetah-Vel	200	Target velocity	$[0, 3]$	$[0.5, 2.5]$	$[0, 0.5) \cup (2.5, 3]$
Cheetah-Vel-Sparse	200	Target velocity	$[0, 3]$	$[0.5, 2.5]$	$[0, 0.5) \cup (2.5, 3]$
Point-Robot-Sparse	60	Goal angle	$[0, \pi]$	$[\pi/8, 7\pi/8]$	$[0, \pi/8) \cup (7\pi/8, \pi]$
Hopper-Param	200	Mass, inertia, damping, friction	$c^{[-3,3]}$	–	–
Hopper-Param-Sparse	200			–	–
Walker-Param	200	Mass, friction	$1.5^{[-3,3]}$	–	–
Walker-Param-Sparse	200			–	–

Reward-varying tasks. Let v denote forward velocity, (v_x, v_y) planar velocity, and a the action. Ant-Dir rewards velocity along goal direction g: $\cos(g)v_x + \sin(g)v_y$. Cheetah-Vel uses $-|v - v^\star| - 0.05\|a\|^2$ for target velocity $v^\star$. Its sparse variant replaces the velocity term with $2 - |v - v^\star|$ inside a tolerance of 0.5, and zero outside, retaining the control cost. Point-Robot-Sparse uses goals on a unit semicircle and gives reward 1 within radius 0.2 of the goal, and 0 otherwise.

Dynamics-varying tasks. Hopper varies 41 mass, inertia, damping, and friction multipliers. Walker varies 32 mass and friction multipliers. Their dense rewards are $v - 10^{-3}\|a\|^2$ and $v + 1 - 10^{-3}\|a\|^2$, respectively. Both sparse variants use $0.8 - |v - 1.5|$ when $|v - 1.5| \leq 0.5$, and zero otherwise, before subtracting the same control cost.

Task splits. Task seeds 1–30 are used for training and 31–40 for standard testing. For the ID/OOD split, goals outside the ID range are held out as OOD tasks. The five largest-seed tasks inside the range form the ID test set. The remaining tasks are used for training. The resulting train/ID-test/OOD-test counts are $26/5/9$ for Ant-Dir, $20/5/15$ for both Cheetah environments, and $27/5/8$ for Point-Robot-Sparse.

D.3 Data collection

Each of the 40 tasks has its own SAC collector. We retain its replay data from initial exploration through training. The MuJoCo collectors use two hidden layers of 128 units, 50 initial random episodes, and 500 iterations of two episodes followed by 200 updates. Point-Robot-Sparse uses two hidden layers of 32 units and 300 iterations with 500 updates each. Collector learning rates are 3×10^{-4}, the batch size is 256, and the target-update rate is 0.005. The fixed entropy coefficient is 0.2 and the discount is 0.99, except on Point-Robot-Sparse, where they are 0.01 and 0.9.

D.4 Implementation and training

Baseline implementation. We implement BTR and all baseline policy-training stages in JAX (Bradbury et al., 2018) for efficiency. CORRO retains its official PyTorch (Paszke et al., 2019) code for training the conditional VAE and contrastive encoder. We then transfer the encoder weights to JAX and train its attention aggregator and SAC policy. UNICORN is reimplemented entirely in JAX, including its jointly trained task encoder and reward–next-state decoder. MetaSTAR is also fully ported,

retaining its transformer world model, model-generated data, and conservative policy objective. We check these ports against the official PyTorch implementations through encoder-output comparisons for CORRO and return comparisons on the same collected datasets for all three baselines.

Model and policy training. Tables 11 and 12 give common and environment-specific BTR settings.

Table 11: Common BTR training settings. Environment-specific choices are listed in Table 12.

Parameter	Value
SAC update budget	200k; evaluation every 10k updates
Task batch size	16
Collected-data batch size per task	200 interaction tuples
Task-model update frequency	Every 10 SAC updates
Actor and critic hidden layers	256×256, ReLU
Feature hidden layers	32×32, with layer normalisation (Ba et al., 2016)
Feature optimiser	Adam (Kingma & Ba, 2015), learning rate 10^{-3}, gradient clipping at 1
Feature norm penalties	$\lambda_T = 5 \times 10^{-2}$, $\lambda_R = 10^{-2}$
Learned reward feature dimension	$d_R = 8$
Target-network update rate	5×10^{-3}
Initial belief for relabelling	Prior or original-context belief, each with probability 0.5
Asymmetric critic	Enabled, except in the symmetric-critic ablation

Table 12: Environment-specific settings. A: Ant-Dir; C: Cheetah-Vel; CS: Cheetah-Vel-Sparse; H: Hopper-Param; HS: Hopper-Param-Sparse; W: Walker-Param; WS: Walker-Param-Sparse; P: Point-Robot-Sparse.

Parameter	A	C	CS	H	HS	W	WS	P
State / action dimension	27/8	20/6	20/6	12/3	12/3	18/6	18/6	2/2
Dynamics feature dimension d_T	32	32	32	8	8	8	8	32
Reward feature dimension d_R	8	8	8	8	8	8	8	8
Dynamics input to policy	None	None	None	Linear	Linear	Linear	Linear	None
Reward-mean input	Raw	Raw	Raw	Norm.	Norm.	Norm.	Norm.	Norm.
Relabelled segments of history M	20	5	5	5	10	5	20	3
Steps per segment of history H	10	10	10	10	10	20	10	10
Actor learning rate ($\times 10^{-4}$)	5	5	5	3	5	3	5	3
Critic learning rate ($\times 10^{-4}$)	5	5	5	4	5	4	5	4
Discount	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.9

Belief initialisation and policy inputs. The conjugate prior uses identity precision factors Ξ_R, Ξ_T, with $\omega_R = \nu_R = 2$ and $\omega_{T,j} = \nu_{T,j} = d_s + 1$. Prior means M_R, M_T start at $10^{-4}, 10^{-5}$, respectively. The prior then remains fixed. Policy inputs use normalised, flattened dynamics posterior means and raw or normalised reward posterior means (Table 12).

Optimisation and evaluation. Actors and critics use Adam. Cheetah and Point-Robot use gradient clipping at 1, no critic layer normalisation, and actor and entropy updates every SAC step. Elsewhere, critics use layer normalisation, with actor updates every two steps and entropy updates every 100. Automatic entropy tuning starts at 0.2 for Cheetah and Point-Robot and 0.1 elsewhere. Evaluation contexts contain 100 tuples, or 30 on Point-Robot-Sparse. We report online episode 2 for the standard comparison and episode 5 for ID/OOD and extended online adaptation.