Efficient exploration is a crucial challenge in deep reinforcement learning. Several methods, such as behavioral priors, are able to leverage offline data in order to efficiently accelerate reinforcement learning on complex tasks. However, if the task at hand deviates excessively from the demonstrated task, the effectiveness of such methods is limited. In our work, we propose to learn features from offline data that are shared by a more diverse range of tasks, such as correlation between actions and directedness. Therefore, we introduce state-independent temporal priors, which directly model temporal consistency in demonstrated trajectories, and are capable of driving exploration in complex tasks, even when trained on data collected on simpler tasks. Furthermore, we introduce a novel integration scheme for action priors in off-policy reinforcement learning by dynamically sampling actions from a probabilistic mixture of policy and action prior. We compare our approach against strong baselines and provide empirical evidence that it can accelerate reinforcement learning in long-horizon continuous control tasks under sparse reward settings.

1 Introduction

Exploration is a fundamental issue in reinforcement learning (RL): in order for an agent to maximize its reward signal, it needs to adequately cover its state space and observe the outcome of its actions. This becomes increasingly difficult when dealing with large, continuous state and action spaces, as is the case in many real-world applications. Despite a large and fruitful body of research on exploration (Bellemare et al., 2016; Osband et al., 2016; Tang et al., 2017; Osband et al., 2018; Azizzadenesheli et al., 2018; Burda et al., 2019; Dabney et al., 2021; Ecoffet et al., 2021; Vulin et al., 2021), most general-purpose algorithms remain based on $\epsilon$-greedy exploration (Mnih et al., 2015) or entropy-regularized Gaussian policies (Haarnoja et al., 2018). In the absence of an informative reward signal, both methods rely on uniformly sampling actions from the action space, independently of the history of the agent. Unfortunately, in sparse reward settings, achieving positive returns by uncorrelated exploration becomes exponentially less likely as the horizon length increases (Dabney et al., 2021).

A promising approach to achieve efficient exploration is that of using a behavioral prior to guide the policy (Pertsch et al., 2020; Tirumala et al., 2020; Singh et al., 2021). Typically, this is learned from expert trajectories as a state-conditional action distribution. Behavioral priors are able to foster directed exploration (Singh et al., 2021), by assuming a strong similarity between the agent and expert tasks. However, an agent should ideally be able to produce efficient explorative behaviors across a broad range of diverse tasks, even if they are rather different from those demonstrated (Parisì et al., 2021). Similarly, it should be possible to leverage information collected on simpler tasks to enable learning in more complex scenarios (Florensa et al., 2017; Gehring et al., 2021).

In this paper, we thus propose to focus on the temporal structure of demonstrations rather than on task-specific strategies. We find that this choice enables knowledge transfer from simple offline trajectories to a diverse family of more complex tasks. Our method, which we dub TEMporal Priors for exploration in Off-policy Reinforcement Learning (TempoRL), introduces temporal priors as state-independent non-Markovian action distributions $\tilde{\pi}(a_t|a_{t-1})$, modeling promising actions conditioned on past actions. We find

## TempoRL: Temporal Priors for Exploration in Off-Policy Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

Efficient exploration is a crucial challenge in deep reinforcement learning. Several methods, such as behavioral priors, are able to leverage offline data in order to efficiently accelerate reinforcement learning on complex tasks. However, if the task at hand deviates excessively from the demonstrated task, the effectiveness of such methods is limited. In our work, we propose to learn features from offline data that are shared by a more diverse range of tasks, such as correlation between actions and directedness. Therefore, we introduce state-independent temporal priors, which directly model temporal consistency in demonstrated trajectories, and are capable of driving exploration in complex tasks, even when trained on data collected on simpler tasks. Furthermore, we introduce a novel integration scheme for action priors in off-policy reinforcement learning by dynamically sampling actions from a probabilistic mixture of policy and action prior. We compare our approach against strong baselines and provide empirical evidence that it can accelerate reinforcement learning in long-horizon continuous control tasks under sparse reward settings.
Figure 1: TempoRL: A temporal prior is trained on task-agnostic trajectories, such as reaching random goals in a reaching task (top left). Actions are then sampled from a dynamic mixture between the temporal prior and the policy in downstream learning of more complex tasks. Our method works with both vector-based and image-based state inputs.

that this class of priors is sufficient for capturing desirable properties for exploration, such as directedness and temporal correlation. This information is often learnable from few expert trajectories collected in simple environments (e.g. reaching uniformly sampled end-effector positions for a robotic arm, see Figure 1). Since such demonstrations do not necessarily contain information that is specific to the task at hand, they can be considered task-agnostic trajectories. Temporal priors can then be used to guide exploration in more complex tasks, in which observed states are not guaranteed to belong to the expert state distribution, and behavioral priors are, as a result, inherently limited.

Furthermore, we propose a principled manner of integrating action priors into the Soft Actor Critic framework (Haarnoja et al., 2018). In downstream learning of more complex tasks, our method samples actions from a dynamic mixture between the policy and temporal prior. Crucially, we derive an update rule for learning the mixing coefficient by directly optimizing a (max-entropy) RL objective. Our dynamic integration allows the policy to receive strong guidance when needed, while crucially retaining the ability to explore diverse actions when the prior is no longer beneficial.

In our experiments, we first analyze the choice of conditioning variables for action priors by measuring their effectiveness in accelerating downstream RL. We then focus on temporal priors and verify their capability to produce correlated and directed behavior. Most importantly, we provide empirical evidence that TempoRL can leverage task-agnostic trajectories to accelerate learning. For this purpose, our method is evaluated against several baselines in complex long-horizon control tasks with sparse rewards. Finally, we present further applications of TempoRL, including leveraging offline state-based trajectories to (a) accelerate visual RL or (b) guide exploration when the task is corrupted by biased observations.

Our contributions can be organized as follows:

1. We propose learnable non-Markovian action priors conditioned on past actions. We show that sampling from these priors produces directed and correlated trajectories.

2. We introduce a principled manner of integrating action priors into the Soft Actor Critic framework (Haarnoja et al., 2018).

3. We show how temporal priors can be learned from few expert trajectories on simple tasks (e.g. reaching) and used to improve exploration efficiency in more complex, unseen tasks (e.g. opening a window), even across fundamentally different settings (e.g., from non-visual to visual RL).
After discussing our method’s novelty and related literature in Section 2, we introduce our setting in Section 3. The method is described in Section 4, while empirical evidence of its effectiveness is reported in Section 5. Finally, Section 6 contains a brief closing discussion of our work. We make our code available for research purposes.

2 Related Work

While we limit this section to essential topics, further discussion can be found in Appendix F.

Temporal-Extended Exploration Several works have attempted to directly address the inability of traditional methods, such as $\epsilon$-greedy or uniform action sampling (Lillicrap et al., 2016; Haarnoja et al., 2018), to produce correlated trajectories. A recent study (Dabney et al., 2021) highlights this issue and shows how repeating random actions for multiple steps is sufficient to significantly accelerate Rainbow (Hessel et al., 2018) on the Atari benchmark (Bellemare et al., 2013). Similarly, Amin et al. (2021) propose a non-learned policy inspired by the theory of freely-rotating chains in polymer physics to collect initial explorative trajectories in continuous control tasks. Both methods pinpoint a fundamental issue, but rely on scripted policies which are hand-crafted for a particular family of environments. On the other hand, our prior is learned, and does not require engineering an explorer, which can be unfeasible for complex tasks. Moreover, our integration of priors into off-policy RL is not limited to initial trajectories and dynamically adjusts the likelihood of sampling actions from the prior.

Hierarchical Reinforcement Learning Another approach to tackle exploration-hard tasks is to rely on a hierarchical decomposition into different levels of temporal and functional abstraction (Parr & Russell, 1998; Dietterich, 2000; Sutton et al., 1999; Dayan & Hinton, 2000). For instance, tasks can be decomposed into high level planning and a set of low-level policies, often referred to as skills (Konidaris & Barto, 2007; Eysenbach et al., 2019) or options (Sutton et al., 1999; Bacon et al., 2017). This effectively reduces the planning horizon and allows efficient solving of complex tasks (Bacon et al., 2017; Vezhnevets et al., 2017; Nachum et al., 2018; Levy et al., 2019; Christen et al., 2021). In general, such approaches only target temporal correlation within skills, as each skill is generally selected independently from the last one. Incidentally, our method is not designed to achieve temporal abstraction, but can be interpreted in a hierarchical framework (Schafer et al., 2021), in which a high-level criterion (the mixing function) governs a probabilistic choice between an explorer (temporal prior) and an exploiter (policy). Our method’s application is also inherently related to the works of Florensa et al. (2017); Gehring et al. (2021), which propose to extract skills on simple training tasks and deploy them in more complex scenarios. In contrast with these methods, training a temporal prior does not require access to the training environment, but only to offline trajectories.

Behavioral Priors Behavioral priors generally represent state-conditional action distributions, modeling promising actions for the current state of the environment. Such priors can be learned jointly with the policy in the context of KL-regularized RL (Tirumala et al., 2019, 2020), which in some cases restricts the information available to the prior (Galashov et al., 2019). A second approach consists in learning behavioral priors from expert policies on related tasks. This is the case for several works (Peng et al., 2019; Pertsch et al., 2020, 2021; Ajay et al., 2021) which adopt a Gaussian behavioral prior in a latent skill-space. In particular, Pertsch et al. (2020) report that a prior is crucial to guiding a high-level actor in an HRL framework. An important contribution to the field is made by PARROT (Singh et al., 2021), which focuses on a visual setup and introduces a flow-based transformation of the action space to allow arbitrarily complex prior distributions. We extend this idea to prior action distributions that are not conditioned on the current state or a part thereof, but rather on past actions, and are therefore non-Markovian. Moreover, we propose a novel, more flexible integration of the prior distribution into the learning algorithm. Finally, we overcome the reliance on a strong similarity between states observed by the expert and the agent.

1https://sites.google.com/view/tempo-rl
3 Background

3.1 Setting

Reinforcement learning (RL) is the problem that an agent faces when learning to interact with a dynamic environment. Albeit with a slightly different definition, we formalize the environment as a goal-conditioned Markov Decision Process (gc-MDP) (Nasiriany et al., 2019), that consists of a 6-tuple $(S, A, G, R, T, \gamma)$, where $S$ is the state space, $A$ is the action space, $G \subseteq S$ is the goal space, $R : S \times G \rightarrow \mathbb{R}$ is a scalar reward function, $T : S \times A \rightarrow \Pi(S)$ a probabilistic transition function that maps state-action pairs to distributions over $S$ and, finally, $\gamma$ is a discount factor. Assuming goals to be drawn from a distribution $p_G$, the objective of an RL agent can then be expressed over a time horizon $T$ as finding a probabilistic policy $\pi^* = \arg\max_\pi \mathbb{E}_{g \sim p_G} \sum_{t=0}^{T-1} \gamma^t R(s_t, g)$, with $s_t \sim T(s_{t-1}, a_{t-1})$ and $a_{t-1} \sim \pi(s_{t-1}, g)$. In order to simplify notation, from this point on, we will implicitly include the goal into the state at each time step: $s_t \leftarrow (s_t, g)$.

We focus on long-horizon control problems with continuous state and action spaces and sparse rewards, i.e., non-zero only after task completion. Although our method can be generally applied to off-policy RL methods, we build upon Soft Actor Critic (Haarnoja et al., 2018), due to its wide adoption in these settings.

Finally, in contrast with several behavioral prior approaches (Galashov et al., 2019; Pertsch et al., 2020; Singh et al., 2021), we adopt a more general and challenging setting. First, we do not assume that prior information on the structure of the state space is available. Second, while we also assume access to a collection of trajectories $D_{\text{exp}} = \{(s_0^i, a_0^i, s_1^i, a_1^i, \ldots, s_N^i, a_N^i)\}_{i=0}^L$, we do not require high quality trajectories to be collected on a task that is similar to the one at hand. From this point on, we refer to the task used for collecting data as the training task, and to the task the RL agent has to complete as the downstream task.

3.2 Behavioral Priors

A behavioral prior $\pi(a|s)$ (Pertsch et al., 2020; Singh et al., 2021) is a state-conditional probability distribution over the action space (cf. Figure 2a). Behavioral priors can be trained to assign high probability to useful actions with respect to the current state, and hence be used to accelerate RL. A behavioral prior can only guide the policy effectively as long as prior information on the structure of the state space is available (Galashov et al., 2019), or the prior has been trained on data collected on a closely related task (Pertsch et al., 2020; Singh et al., 2021).

In our settings, the structure of observations is unknown and expert trajectories $D_{\text{exp}}$ are task-agnostic or collected on simple tasks. As a consequence, the distribution of training states might not match the distribution of states produced by downstream tasks: behavioral priors will then be evaluated on out-of-distribution samples and their performance will degrade drastically, as shown in Section 5.1.

4 Method

Our method relies on the integration of a learned temporal prior into an off-policy RL algorithm. Thus, we first define and discuss the class of action priors of interest, and later describe how they can be integrated into existing off-policy algorithms.

4.1 Temporal Priors

Within the settings outlined in Section 3.1 it is still possible to extract and transfer knowledge from the expert dataset $D_{\text{exp}}$ to the agent. In this case, conditioning an action prior on the current state might be both insufficient and counterproductive, as a state-conditional action prior would receive out-of-distribution samples as inputs, due to the distribution shift between states from the training and downstream task. For this reason, we shift our focus towards modeling the temporal correlation of expert trajectories.

In order to recover this information, we thus propose to learn a temporal prior. A temporal prior is a non-Markovian, state-independent action prior, representing a probability distribution over the action space that is conditioned on past actions: $\bar{\pi}(a_1|a_0^{t-1})$ (see Figure 2b, c). By renouncing the informativeness of...
states with respect to expert actions, temporal priors retain the ability to model temporal correlation, which crucially enables knowledge transfer to tasks with a fundamentally different state distribution with respect to demonstrations. This is indeed a viable strategy in a hard-exploration setting, when no prior information is available on the state space and no reward is observed: in this case, extracting any information from the state remains challenging. Our empirical evidence suggests that the simplest form of temporal priors, i.e., \( \bar{\pi}(a_t|a_{t-1}) \) (see Figure 2c) is surprisingly competitive with variants conditioned on multiple past actions (cf. Section 5.1) and is sufficient to capture non-trivial temporal relations.

Independently of the conditioning variables, temporal priors can conveniently be modeled as conditional generative models and trained through empirical risk minimization. For the purpose of this paper, we choose to use the conditional variant of the Real Non Value Preserving Flow (Dinh et al., 2017; Ardizzone et al., 2019), which is well suited for Euclidean action spaces (Singh et al., 2021). Moreover, our integration in the SAC framework allows arbitrarily complex prior distributions, which Real NVP Flows are in principle able to capture.

In the context of Real NVP Flows, training samples are actions \( a \in \mathcal{A} \), paired with conditioning variables \( a_{t-1} \), thus the learned mapping is \( a = f_\theta(z; a_{t-1}) \), with \( z \sim \mathcal{N} \). Since \( f_\theta \) is invertible, it is possible to analytically compute the likelihood of a single training pair \((a_t, a_{t-1})\) and maximize its expected value through standard gradient-based optimization techniques. An empirical justification of this choice is found in Appendix B, while implementation details are reported in Appendix H. For a complete introduction to Real NVP Flows, we refer the reader to Dinh et al. (2017).

One final concern regards the nature of the data for training the temporal prior. The main requirements for the training data are two: (1) the task on which the data is collected needs to share the same action space of the downstream task and (2) the training trajectories should display the desired qualities of correlation and directness. In general, we adopt task-agnostic expert trajectories generated by achieving random goals in simple environments. Such simple trajectories can be produced from scratch using standard RL or generated by a scripted policy. We remark that, in contrast with existing approaches (Pertsch et al., 2020; Singh et al., 2021), this framework poses very weak requirements on the similarity between the task used for data collection and the target task. As we show in Section 5.3, this allows our method to transfer knowledge across different tasks and settings, such as from a simple reaching task with access to the true state of the system to a window-closing task in a visual RL setting. Limitations of our method with respect to training data are discussed in Section 6.

### 4.2 Soft Actor Critic with TempoRL

The main challenge introduced by temporal priors stems from their non-Markovianity, which renders existing integrations schemes unsuitable (Singh et al., 2021; Tirumala et al., 2020). For this reason, we introduce a novel method to integrate action priors in an off-policy RL framework. While our method can be formally derived for Markovian priors, it can also be generalized to non-Markovian action distributions by allowing further bias in Q-value estimates. In the following derivations, we will consider a general state-dependent non-Markovian prior \( \bar{\pi}(a_t|H_t) \) with \( H_t = (s_t, a_{t-1}^t) \), but the integration remains applicable to temporal or behavioral priors alike (for brevity, we will interchangeably use the compact notation \( \bar{\pi}(a_t) \)). The key strategy consists of sampling actions from a dynamically weighted mixture between the policy and a fixed prior distribution. We demonstrate it as an integration into the SAC framework.

![Figure 2: Graphical models of different priors: from left to right, a behavioral prior, a general temporal prior and a single-step temporal prior. Solid arrows represent the environment’s transition function \( T \), while dashed arrow indicate conditional modeling.](image)
SAC is an off-policy actor critic algorithm that trains a stochastic policy $\pi_\phi$ and state-action value function estimator $Q_\theta$ in an off-policy way; its objective is designed to pursue rewards while maximizing the entropy of its policy. When prior knowledge on the structure of the environment or task is available, simply sampling actions from a maximal entropy policy $\pi_\phi$ may not be optimal. On the other hand, blindly sampling from a fixed action prior $\bar{\pi}$ prevents exploitation of reward signals as well as any behavior which is not encoded in the prior. Ideally, it is desirable to control the degree to which actions are sampled from the prior.

We propose to achieve this in a natural way by sampling actions from a mixture $\bar{\pi}$ between the policy $\pi_\phi$ and the prior $\bar{\pi}$:

$$a_t \sim \bar{\pi}(\cdot|s_t) = (1 - \lambda_t)\pi_\phi(\cdot|s_t) + \lambda_t \bar{\pi}(\cdot), \quad (1)$$

where the mixing parameter $\lambda_t$ is bounded ($0 \leq \lambda_t \leq 1$) and computed dynamically at each step $t$.

In principle, it is desirable to bias the mixture $\bar{\pi}$ towards the prior $\bar{\pi}$ or the policy $\pi$, depending on which of the two is more likely to reach the goal. Interestingly, if the mixing weight is computed through a parameterized function of states $\lambda_t = \Lambda_\omega(s_t)$, this behavior can be naturally recovered by maximizing the maximum entropy objective with respect to the mixture $\bar{\pi}$:

$$\arg\max_{\pi_\phi, \Lambda_\omega} \mathbb{E}_{\bar{\pi}} \left[ \sum_{t=0}^{\infty} \gamma^t \left( R(s_t, a_t) + \alpha \mathcal{H}(\pi_\phi(\cdot|s_t)) \right) \right]. \quad (2)$$

We note that, while this expectation is computed over trajectories sampled from the mixture $\bar{\pi}$, the entropy term $\mathcal{H}(\pi_\phi(\cdot|s_t))$ is computed on the policy $\pi_\phi$ alone, as it is not desirable to incentivize exploration of states according to the prior $\bar{\pi}$’s entropy.

Through straightforward derivations (see Appendix A), it is possible to derive learning objectives for the policy $\pi_\phi$ and for the Q-estimator $Q_\theta$, as well as for a mixing function $\Lambda_\omega$. Given a distribution $D$ of observed transitions, the loss functions can be defined as:

$$J_{\pi_\phi} = - \mathbb{E}_{(s) \sim D} \left[ (1 - \Lambda_\omega(s)) \left( Q_\theta^\pi(s, a) - \alpha \log \pi_\phi(a|s) \right) \right]$$

with $a \sim \pi_\phi(\cdot|s)$, \quad (3)

$$J_{Q_\theta} = \mathbb{E}_{(s, a, s') \sim D} \left[ \left( Q_\theta^\pi(s, a) - y_t(s, a, s') \right)^2 \right],$$

where the target for the Q-value is computed as

$$y_t(s, a, s') = R(s, a) + \gamma \left( Q_\theta^\pi(s', \bar{a}') - \alpha \log \pi_\phi(a'|s') \right)$$

with $\bar{a}' \sim \bar{\pi}(\cdot|s')$, $a' \sim \pi_\phi(\cdot|s')$, \quad (5)

and finally

$$J_\Lambda = - \mathbb{E}_{(s) \sim D} \left[ \Lambda_\omega(s) \left( Q_\theta^\pi(s, \bar{a}) - Q_\theta^\pi(s, a) \right) \right]$$

with $\bar{a} \sim \bar{\pi}(\cdot)$, $a \sim \pi_\phi(\cdot|s)$.

\textbf{Algorithm 1 SAC with TempoRL}

1: Train action prior $\bar{\pi}(a_t|H_t)$
2: Initialize parameters $\theta, \phi, \omega$
3: for each iteration do
4: Observe $s_0$
5: Initialize history $H_0 = (s_0)$
6: for each environment step do
7: $\lambda_t = \Lambda_\omega(s_t)$
8: $a_t \sim \bar{\pi} = (1 - \lambda_t)\pi(a_t|s_t) + \lambda_t \bar{\pi}(a_t|H_t)$
9: $s_{t+1} \sim T(s_t, a_t)$
10: $H_{t+1} = H_t \cup (a_t, s_{t+1})$
11: end for
12: for each gradient step do
13: Update $\theta, \phi, \omega$
14: end for
15: end for

The three objectives can be empirically estimated and minimized through standard procedures, as reported in Haarnoja et al. [2018] and in Appendix A. We further note that, as expected, Equation 6 encourages
high mixing weights $\lambda_t = \Lambda_T(s_t)$ in case Q-values for actions sampled from the prior $\bar{\pi}$ are higher than those for actions sampled from the policy $\pi_\phi$, and vice versa (see Appendix D for an empirical analysis on the evolution of $\lambda$ during training).

Algorithm 1 summarizes (in blue) the necessary modifications for integrating the action prior into the SAC framework. Namely, actions are sampled from a mixture (line 8) weighted according to the output of a mixing function (line 7). Finally, the history of the agent needs to be initialized (line 5) and updated at each step (line 10). Update rules for $\theta$, $\phi$ and $\omega$ (line 13) are computed by minimizing the objectives in Equations 3, 4 and 6.

5 Experiments

We evaluate our method in a series of experiments to empirically validate our contributions. First, in Section 5.1 we compare the effectiveness of different action priors to justify the choice of state-independent conditioning. In Section 5.2, we verify that sampling from a temporal prior can produce correlated and state-covering behavior, without the need to hand-craft an exploration policy. Next, we show how our method can improve exploration efficiency in unseen long-horizon tasks by comparing against various baselines in state-based RL (Section 5.3). Finally, we present a proof of concept of different applications for temporal priors, i.e., in visual RL and when dealing with biased observations (Section 5.4). Ablations for our method and baselines are provided in Appendix B.

Baselines We now present a brief introduction to the baselines we consider, while implementation details are provided in Appendix H.3.

- **SAC**: vanilla Soft Actor Critic [Haarnoja et al., 2018].
- **SAC+BC**: SAC with warm-started policy through behavior cloning.
- **SAC-PolyRL**: SAC with locally self-avoiding walks [Amin et al., 2021].
- **PARROT-state**: flow-based behavioral prior enforced through a transformation of the action space [Singh et al., 2021]. We benchmark a state-based variant in non-visual settings.

Environments We evaluate our method on three types of domains, namely robotic manipulation, maze navigation and robotic locomotion:

- **meta-world**: robot manipulation tasks from the MT10 benchmark in the publicly available meta-world suite [Yu et al., 2020].
- **point-maze**: qualitatively different 2D maze structures, navigated by a point-like agent [Pitis et al., 2020].
- **gym-mujoco**: widely adopted continuous control environments for locomotion of simulated robots [Brockman et al., 2016; Todorov et al., 2012].

The first two suites define diverse tasks that share similar underlying dynamics (e.g. navigating mazes with different structures, or interacting with different objects), and are therefore used to benchmark the ability to transfer knowledge from offline trajectories to complex downstream tasks (Section 5.1, 5.2, 5.3). In this case, offline trajectories are collected in simple training tasks that involve reaching uniformly sampled goals in an empty environment (reach for meta-world and room for point-maze). RL agents are then trained and
Figure 4: Comparison of downstream performance of action priors with different conditioning variables, reported on the training task (left) and averaged over downstream tasks (right).

We now empirically show how the effectiveness of an action prior depends on the conditioning variables and on the similarity between the training and the downstream tasks. For this purpose, we train several variants of flow-based action priors on offline reaching trajectories and integrate them into a SAC learner as described in Section 4. In particular, we compare temporal priors conditioned on action sequences of different lengths (1, 2, 5, 10), non-Markovian priors conditioned on the previous state-action pair and a behavioral prior (conditioned on the state alone). In Figure 4, we report learning curves for the training task (reach), as well as averaged learning curves over all downstream tasks in meta-world. We observe that all priors are capable of guiding downstream RL as long as the task at hand matches the training task. In this case, state-conditional priors achieve slightly faster exploration by leveraging state-related and task-specific information. However, we find that including the state in the conditioning variables (as done in $\bar{\pi}(a_t|s_t)$ and $\bar{\pi}(a_t|s_t, a_{t-1})$) can jeopardize the ability to transfer knowledge to a different task. We hypothesize that this is due to a mismatch between the state distribution used for training the prior and that generated by the downstream task.

On the other hand, temporal priors are not conditioned on states by design, prove to be a capable alternative across both settings and are able to transfer knowledge to unseen tasks. While conditioning on longer action sequences can improve performance, we note that single-action-conditional models $\bar{\pi}(a_t|a_{t-1})$ are sufficient for capturing non-trivial temporal dependencies within our settings. Hence, they will be the focus of the remaining experiments.

Figure 5: A qualitative comparison of sampled exploration trajectories in a robotic reaching task and in an empty 2D maze. Our method achieves directed behavior while covering most of the state space. SAC fails to cover the full state space, while SAC-PolyRL fails to reach distant areas consistently.

5.1 Conditioning Variables for Action Priors

The gym-mujoco environments do not offer a straightforward way to define tasks, and are therefore used to evaluate a proof-of-concept experiment on biased observations (Section 5.4).

More details can be found in Appendix C, while visual examples of environments are provided in Figure 3.
5.2 Correlation and State Coverage

In this experiment, we show how a one-step state-independent temporal prior $\bar{\pi}(a_t|a_{t-1})$ can generate correlated and directed behavior, which leads to a more complete coverage of the state space during exploration. To this end, we sample 20 random trajectories of 500 steps each with our method and two relevant baselines in the room and reach environments.

As shown in Figure 5, the temporal prior produces directed behavior which covers most of the state space. As expected, uniform sampling (which approximates SAC’s exploration) fails to reach the boundaries of the environment; SAC-PolyRL is capable of producing correlated and directed behaviors, but only after careful design and tuning. This qualitative assessment is consistent with the quantitative evaluation in Table 5.3, which reports state space coverage and radius of gyration squared (Amin et al., 2021) (see Appendix H.1 for details).

5.3 Transfer Learning

Our main results are obtained by comparing our method against several baselines in downstream learning tasks with a vectorial state space, as presented in Figure 6. We report learning curves averaged across downstream tasks, as well as on training tasks for reference. Results for each task are disentangled in Appendix I.

As expected, we observe that the performance of behavioral priors depends on the similarity of between the task at hand and expert demonstrations. This is the case for PARROT-state, which instantly solves the training tasks (reach and room), as its behavioral prior already represents a strong policy. On tasks which are significantly different from the training tasks, PARROT-state is however unable to guide the policy, as it receives out-of-distribution states, resulting in low average performance. On the other hand, TempoRL is largely more capable of transferring knowledge to unseen tasks, while rapidly catching up with PARROT-state in the training tasks.

Other baselines are in general less effective across the benchmarks: Vanilla SAC only makes progress on easier tasks, which can be achieved even with weak exploration.

Figure 7: State coverage metrics for our method and baselines. TempoRL’s trajectories are locally directed and cover the state space well in both environments.

<table>
<thead>
<tr>
<th></th>
<th>$U^2_g$</th>
<th>% Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>reach</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAC</td>
<td>0.006±0.001</td>
<td>0.137±0.01</td>
</tr>
<tr>
<td>SAC-PolyRL</td>
<td>0.025±0.001</td>
<td>0.272±0.01</td>
</tr>
<tr>
<td>TempoRL (ours)</td>
<td><strong>0.053±0.009</strong></td>
<td><strong>0.357±0.02</strong></td>
</tr>
<tr>
<td>room</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAC</td>
<td>0.005±0.001</td>
<td>0.333±0.02</td>
</tr>
<tr>
<td>SAC-PolyRL</td>
<td>0.026±0.002</td>
<td>0.880±0.04</td>
</tr>
<tr>
<td>TempoRL (ours)</td>
<td><strong>0.054±0.008</strong></td>
<td><strong>0.963±0.04</strong></td>
</tr>
</tbody>
</table>
SAC-PolyRL is able to produce good explorative trajectories through its hand-crafted policy and improves on SAC, but fails to explore after its initial phase and does not resolve harder tasks. As previously reported by Singh et al. (2021), initializing SAC through behavioral cloning (SAC+BC) can help guide exploration without constraining the policy, but it fails to generalize across tasks and is regularly outperformed by stronger methods.

5.4 Additional Applications
We finally provide proofs of concept for possible applications of TempoRL. Per-task results are available in Appendix I.

Visual RL Temporal priors allow the state space of the downstream task to be defined arbitrarily. Hence, they also allow transfer to the visual RL setting, which avoids reliance on a low-dimensional vectorized state space and is purely based on RGB observations. To this end, we compare TempoRL with Vanilla SAC and with PARROT in its original, visual setup, i.e., by conditioning its behavioral prior on images. We remark that, in this case, PARROT needs access to additional data (i.e., RGB observations) compared to TempoRL. We report the results in Figure 8 which indicate that our findings hold in visual settings, as TempoRL retains most of its exploration capability.

Biased Downstream Learning A final potential application for temporal priors is that of guiding exploration when downstream learning is corrupted by biased observations. This scenario could model a form of sim-to-real transfer, in which training trajectories were cheaply collected in simulation, but the physical testbed was miscalibrated, resulting in biased observations. We provide experimental results in this setup based on gym-mujoco environments (Ant-v2, HalfCheetah-v2, Swimmer-v2, Hopper-v2, Walker2d-v2). Training trajectories are collected by a SAC agent that was fully trained on dense rewards. In downstream learning, the reward is instead sparse (i.e. only positive when the displacement of the CoM from its initial position surpasses a sparsity threshold $\bar{x}$), and significant exploration is required. Moreover, the observations in downstream learning are corrupted by a fixed additive bias $s = s + b$. Figure 9 compares test performance of downstream agents when guided by a behavioral prior (PARROT-state) or by a temporal prior (TempoRL), after 500k steps and averaged across environments. We observe that behavioral priors are capable of extracting a near-optimal policy from the training data, which transfers perfectly to the sparsified environments if the observations are uncorrupted. However, as bias increases, their performance sharply drops. On the other hand, temporal priors are less effective in unbiased conditions, but their performance is generally retained as bias increases. More details on this experiment are presented in Appendix G.3.

6 Discussion and Conclusion
Limitations While temporal priors are able to generalize to a broader set of tasks with respect to behavioral priors, they are data-driven, and as such will only reconstruct behavior that appears in the offline dataset of trajectories they are trained on. For instance, when training on demonstrations for reach, temporal priors will not encode any grasping strategy. While such behavior can still be recovered by the policy $\pi$, there is no incentive to do so. As a result, successfully transferring to tasks that require additional strategies, such as grasping (e.g., pick-place) remains an open challenge. Secondly, as shown in Figure 6 behavioral priors remain superior when the downstream task can be solved by the policy used for collecting expert trai-
jectories. As temporal priors do not model state-action relationships, they can only provide weaker guidance in this privileged case. For this reason, a strategy to combine benefits from behavioral and temporal priors would represent an interesting direction for the future.

**Conclusion** In this paper we proposed a method for improving exploration efficiency in off-policy reinforcement learning. In particular, we introduced temporal priors, a family of non-Markovian, state-independent and flow-based action priors, and proposed a principled manner for integration into an off-policy reinforcement learning framework. While leveraging a relatively simple idea, temporal priors represent a powerful method for exploration. As shown by significant empirical evidence, they can massively accelerate learning across a diverse set of tasks, while solely requiring a modest amount of offline, task-agnostic trajectories.

**Broader Impact Statement**

Our main contribution revolves on accelerating and enabling reinforcement learning in environments with a strong exploration component. As a consequence, we believe that concerns with respect to our method are for the most part aligned with general RL research. For instance, improved sample efficiency could on one hand accelerate the process of automation, which might have a negative impact on societal equality, and on the other hand democratize access to powerful RL methods by lowering the amount of required resources. Due to the general nature of our method, we believe that we do not introduce fundamentally new risks.

**References**


A Objective Derivation

The purpose of this section is that of describing how learning objectives for the policy \( \pi \), the Q-estimator \( Q^\pi \) and the mixing function \( \Lambda \) can be derived from a max-entropy objective; while in practice each function is respectively parameterized by \( \phi, \theta \) and \( \omega \), in order to ease notation we will only explicitly mention these parameters when introducing empirical loss estimates. The objective is defined as

\[
\mathbb{E}_{\tau \sim \pi} \left[ \sum_{t=0}^{\infty} \left( \gamma^t R(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t)) \right) \right],
\]

(7)

where the expectation is computed over trajectories generated by the mixture \( \tilde{\pi} = (1 - \lambda_t) \pi + \lambda_t \bar{\pi} \), \( \bar{\pi} \) is a fixed action prior and \( \lambda_t = \Lambda(s_t) \) is the mixing weight. The optimization of Equation (7) can be performed by modeling the action prior \( \bar{\pi}(\cdot) \) as a Markovian distribution over the action space \( \mathcal{A} \); we note that non-Markovian action priors do not necessarily satisfy Equation (9) and therefore can introduce bias in Q-value targets (Equation (11)). The steps of these derivations follow those reported in Haarnoja et al. (2018), as our method is designed to fit within the proposed framework. First, let us formally introduce the Q-function for the mixture \( \tilde{\pi} \):

\[
Q^\tilde{\pi}(s,a) = \mathbb{E}_{\tau \sim \tilde{\pi}} \left[ \sum_{t=0}^{\infty} \left( \gamma^t R(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t)) \right) \right] s_0 = s, a_0 = a.
\]

(8)

We can then formulate the Bellman Equation and explicitly unravel the entropy term:

\[
Q^\tilde{\pi}(s,a) = \mathbb{E}_{s' \sim T(\cdot|s,a)} \left[ R(s,a) + \gamma \left( Q^\tilde{\pi}(s', \tilde{a}') + \alpha \mathcal{H}(\pi(\cdot|s')) \right) \right]
\]

\[
= \mathbb{E}_{s' \sim T(\cdot|s,a)} \mathbb{E}_{\tilde{a}' \sim \tilde{\pi}(\cdot|s')} \left[ R(s,a) + \gamma \left( Q^\tilde{\pi}(s', \tilde{a}') + \alpha \mathbb{E}_{a' \sim \pi(\cdot|s')} \left[ \log(\pi(a'|s')) \right] \right) \right]
\]

\[
= \mathbb{E}_{s' \sim T(\cdot|s,a)} \mathbb{E}_{\tilde{a}' \sim \tilde{\pi}(\cdot|s')} \left[ R(s,a) + \gamma \left( Q^\tilde{\pi}(s', \tilde{a}') - \alpha \log(\pi(a'|s')) \right) \right].
\]

(9)

The right-hand side of Equation (9) can be estimated via Monte Carlo sampling and used as a target for Q-estimates, which can then be trained by minimizing a standard MSE loss. In practice, as is done for SAC, two separate parameterized Q-function estimators \( Q^\tilde{\pi}_i, i=1,2 \) are used to prevent overestimating Q-values. Additionally, we also adopt target networks \( \{Q^\tilde{\pi}_{i,\text{target}}\}_{i=1,2} \), updated via Polyak averaging. As a result, when sampling a batch \( B \) from a replay buffer, the empirical estimates for Q-losses are:

\[
\hat{J}_{Q^i} = \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} \left( Q^i(s,a) - \hat{y}_t(s', r, d) \right)^2,
\]

(10)

where the target for the Q-value is computed as

\[
\hat{y}_t(s', r, d) = r + \gamma (1 - d) \left( \min_{i=1,2} Q^i_{\text{target}}(s', \tilde{a}') - \alpha \log \pi_{\tilde{\phi}}(a'|s') \right) \quad \text{with } \tilde{a}' \sim \tilde{\pi}(\cdot|s'), a' \sim \pi_{\tilde{\phi}}(\cdot|s').
\]

(11)

and \( r, d \) stand for the reward and done signal, respectively. As the prior \( \bar{\pi} \) is fixed, optimizing the mixture policy \( \bar{\pi} = (1 - \lambda) \pi + \lambda \bar{\pi} \) only involves the optimization of \( \pi \) and \( \lambda \). The policy \( \pi \) can be trained to minimize the KL-divergence with the soft-max of the Q-function, or alternatively to maximize the value function \( V^\tilde{\pi}(s) \). Our method relies on the second option and trains both \( \pi \) and \( \Lambda \) to maximize the value function \( V^\tilde{\pi}(s) \), which can be formulated as follows:

\[
V^\tilde{\pi}(s) = \mathbb{E}_{\tilde{a} \sim \tilde{\pi}(\cdot|s)} \mathbb{E}_{a \sim \pi(\cdot|s)} \left[ Q^\tilde{\pi}(s, \tilde{a}) - \alpha \log(\pi(a|s)) \right].
\]

(12)
Interestingly, the value function $V^\pi$ can be decomposed as

$$V^\pi(s) = \mathbb{E}_{\tilde{a} \sim \tilde{\pi}(.|s)}[Q^\pi(s, \tilde{a})] - \alpha \mathbb{E}_{a \sim \pi(.|s)}[\log(\pi(a|s))].$$

$$= \left( \int_{\tilde{a} \in A} Q^\pi(s, \tilde{a}) \tilde{\pi}(\tilde{a}|s) \right) - \alpha \mathbb{E}_{a \sim \pi(.|s)}[\log(\pi(a|s))].$$

$$= \left( \int_{\tilde{a} \in A} Q^\pi(s, \tilde{a}) (\lambda \tilde{\pi}(\tilde{a}) + (1 - \lambda)\pi(\tilde{a}|s)) \right) - \alpha \mathbb{E}_{a \sim \pi(.|s)}[\log(\pi(a|s))].$$

$$= \lambda \left( \int_{\tilde{a} \in A} Q^\pi(s, \tilde{a}) \tilde{\pi}(\tilde{a}) \right) + (1 - \lambda) \left( \int_{a \in A} Q^\pi(s, a) \pi(a|s) \right) - \alpha \mathbb{E}_{a \sim \pi(.|s)}[\log(\pi(a|s))].$$

where $\lambda = \Lambda(s)$ is the mixing weight computed on the current state. When differentiating this objective with respect to the parameters $\phi$ of the policy network $\pi$, the first term does not contribute to gradients, resulting in

$$\max_{\phi} V^\pi(s) = \max_{\phi} \mathbb{E}_{a \sim \pi_{\phi}(.|s)}[1 - \lambda)Q^\pi_{\phi}(s, a) - \alpha \log(\pi_{\phi}(a|s))].$$

Computing the expectation over actions can then be circumvented by using the reparameterization trick, which enables expressing actions as $a_{\phi}(\xi, s)$, where $\xi \sim \mathcal{N}$ is sampled from a standard Gaussian. In practice, considering the two Q-networks, the policy network can be optimized by minimizing the empirical loss estimated over a batch $B$:

$$\hat{J}_{\pi_{\phi}} = \frac{1}{|B|} \sum_{(s, a) \in B} \left( 1 - \Lambda_{\omega}(s) \right) \left( \min_{\theta \in \{1, 2\}} Q_{\theta}(s, \pi_{\phi}(a|s)) - \alpha \log(\pi_{\phi}(a|s)) \right) \text{ with } a \sim \pi_{\phi}(.|s),$$

which can be minimized via standard first-order optimization techniques.

The last learning objective to recover is that for the parameterized mixing function $\Lambda_{\omega}$. Once again, we obtain this by maximizing the value function $V^\pi(s)$. Starting from Equation 13, we can now drop the final term, which does not depend on $\omega$:

$$\max_{\omega} V^\pi(s) = \max_{\omega} \Lambda_{\omega}(s) \mathbb{E}_{\tilde{a} \sim \tilde{\pi}(.)} \left(Q^\pi_{\phi}(s, \tilde{a}) + (1 - \Lambda_{\omega}(s)) \mathbb{E}_{a \sim \pi_{\phi}(.|s)}[Q^\pi_{\phi}(s, a)] \right)$$

$$= \max_{\omega} \Lambda_{\omega}(s) \mathbb{E}_{\tilde{a} \sim \tilde{\pi}(.)} \left(Q^\pi_{\phi}(s, \tilde{a}) \right) + \mathbb{E}_{a \sim \pi_{\phi}(.|s)}[Q^\pi_{\phi}(s, a)] - \Lambda_{\omega}(s) \mathbb{E}_{a \sim \pi_{\phi}(.|s)}[Q^\pi_{\phi}(s, a)]$$

$$= \max_{\omega} \Lambda_{\omega}(s) \mathbb{E}_{\tilde{a} \sim \tilde{\pi}(.)} \left(Q^\pi_{\phi}(s, \tilde{a}) \right) - \Lambda_{\omega}(s) \mathbb{E}_{a \sim \pi_{\phi}(.|s)}[Q^\pi_{\phi}(s, a)] - \Lambda_{\omega}(s) \mathbb{E}_{a \sim \pi_{\phi}(.|s)}[Q^\pi_{\phi}(s, a)]$$

$$= \max_{\omega} \Lambda_{\omega}(s) \mathbb{E}_{\tilde{a} \sim \tilde{\pi}(.)} \left[Q^\pi_{\phi}(s, \tilde{a}) - Q^\pi_{\phi}(s, a) \right].$$

In practice, the loss estimate for training the mixing network $\Lambda_{\omega}$ can be computed over a batch $B$ as

$$\hat{J}_{\Lambda_{\omega}} = \frac{1}{|B|} \sum_{(s, a) \in B} \left[ \Lambda_{\omega}(s) \left( \min_{i=1, 2} Q^\pi_{\phi_i}(s, \tilde{a}) - \min_{i=1, 2} Q^\pi_{\phi_i}(s, a) \right) \right] \text{ with } \tilde{a} \sim \tilde{\pi}(.), a \sim \pi_{\phi}(.|s).$$

The gradient of this objective with respect to the mixing weight $\lambda = \Lambda(s)$ is intuitively the difference in Q-values between actions drawn from the prior and policy. As training proceeds, and the policy $\pi$ improves, the gradient turns negative and encourages lower mixing weights, gradually abandoning the prior $\tilde{\pi}$’s guidance.
Figure 10: Performance on downstream meta-world tasks when modeling a one-step temporal prior through various generative models.

B Model Ablation

We now set out to provide empirical backing for an important design choice. Our policy mixing approach grants freedom in choosing generative models capable of describing complex distributions, as computing distance metrics to the prior is not required. We compare the performance of Real NVP Flows with a Conditional VAE (Sohn et al., 2015), a non-parametric conditional Least Squares Density Estimator (Sugiyama et al., 2010), and an MLP modeling a deterministic prior. As shown in Figure 10, we found flow-based modeling to be both competitive and practical.

C Data Ablation

Figure 11 includes data quantity ablations (with 75%, 50% and 25% of training data), and data quality ablations (by substituting 25%, 50% and 75% of the training data with uniformly sampled actions). In the latter case, we observe a sharp drop in final performance (averaged over downstream meta-world tasks), while we find TempoRL to remain viable in low-data regimes.

D Study on Learned Mixing Weights

In Figure 12, we plot the mixing weight \( \lambda \) during the course of training against returns, averaged over the downstream meta-world tasks. As formally described in Section 4, the mixing weight generally decreases as returns increase; in particular, we observe \( \lambda \to 0 \) for solved tasks and \( \lambda \to 1 \) for unsolved tasks. We found this trend to be consistent for most states visited. We also note that the initial trajectory is influenced by the initialization discussed in Appendix H.4.

E Study on Temporal Correlation

In order to better quantify the concept of temporally correlated exploration, we plot the PSD (averaged over action dimensions) of action sequences sampled from the training set, from our learned prior, as well as those obtained by sampling from a uniform distribution on room. In Figure 13 we note prominent low frequencies for both the training data and TempoRL trajectories.

F Additional Related Works

Meta-RL and Task Inference. The idea of quickly adapting to unseen tasks is a fundamental concept in meta-reinforcement learning (Schmidhuber [1987]; Wang et al., 2017). While TempoRL has a similar goal, it crucially relies on training a policy from scratch for downstream learning and is not designed for zero-shot adaptation. A line of research in meta-RL, referred to as context-based (Wang et al., 2017; Mishra et al., 2018; Rakelly et al., 2019), performs task inference in order to identify the current task and quickly extrapolate how to maximize returns. A task representation can in practice be extracted from recent experiences. Interestingly, temporal priors can in principle handle the same type of input, i.e. sequences of recent action-state pairs. However, instead of producing an explicit representation of the current task, temporal priors directly model an action distribution.
Learning from Demonstrations TempoRL is aligned with existing methods that rely on demonstrations to accelerate RL on complex tasks \cite{Nair2018, Rajeswaran2018, Christen2019}. In principle, the cost of acquiring few expert trajectories can be small compared to the significant engineering effort required in their absence \cite{OpenAI2019}. In most cases, such trajectories are required to be near-optimal and collected in the same environment \cite{Schaal1997} or from the same distribution of tasks \cite{Singh2021}. In our case, expert trajectories can be task-agnostic, unlabeled and collected on a significantly different task, as we focus on reconstructing correlated exploration trajectories instead of exploitative behavior.

G Environment Details

G.1 Robot Manipulation

We rely on the meta-world suite \cite{Yu2020} for robot manipulation experiments. It consists of a simulated 7 DoF Sawyer arm, implemented in the MuJoCo physics engine \cite{Todorov2012}. By default, states are represented as 36-dimensional vectors across all tasks. The state contains the 3D location and aperture of the gripper, the 3D location and quaternion of one object (e.g. door or window), measured for the current and previous time step. Goals are represented as the desired 3D location of the end effector or object, according to the task. Actions are 4-dimensional vectors containing a 3D movement and a 1D control over the aperture of the effector.

We additionally render 64x64 RGB images as observations for the visual setup in Section 5.4, using the camera angle corner (as can be seen in Figure 3). Instead of the originally proposed dense reward, we adopt a binary sparse reward which is non-zero only upon task completion. For more details, we refer to the original implementation \cite{Yu2020}.

G.2 Maze Navigation

Our point-maze environments are adapted from \cite{Pitis2020}. The state consists of the 2D position of the agent, which can be actuated via a velocity-controller through 2D actions. The goal space matches the state spaces in dimensions and representation. The reward signal is 1 when the Euclidean distance to the goal is lesser than 1.2 units, else it is 0. We experiment with two different layouts (renderings can be found in Figure 3):

- **room** is a large, 29×29 square room. The starting position is at the center, and the goal is initialized randomly in one of the 4 corners at each reset. Trajectories for training the priors are obtained from this environment. Only for Figure 5 the size was increased to 81×81, to allow and better visualize long trajectories.
- **corridor** is a larger u-shaped corridor with three parts of lengths 60, 3 and 60 respectively, assembled at 90° clockwise rotations. The structure of the corridor can therefore be contained in a 60 × 3 rectangle. Starting and goal position are fixed and located at opposite ends of the corridor.
Swimmer-v2  |  HalfCheetah-v2  |  Hopper-v2  |  Ant-v2  |  Walker2d-v2

Figure 14: Overview of the different gym-mujoco environments presented in Section 5.4.

- **maze** is a maze presenting intersections and dead ends. Starting and goal positions are fixed, and the optimal policy requires 66 steps.

For further details, we refer directly to the published codebase (URL in Footnote 1).

### G.3 Continuous Control

Due to their prevalence as a benchmark, we use gym-mujoco environments to evaluate TempoRL and baselines in the presence of biased observations. We evaluate methods across 5 popular environments, namely Ant-v2, HalfCheetah-v2, Hopper-v2, Swimmer-v2 and Walker2d-v2 (see Figure 14). Additionally, we parameterize each environment with two variables, namely the bias $b$ and the sparsity parameter $\bar{x}$. Observations in downstream learning are corrupted as $s \leftarrow s + b$, and rewards are sparsified as $r_t = 1_{x \geq \bar{x}}$ where $x$ is the L2 distance between the initial and current position of the CoM. In order to aggregate data across environments, we empirically find an environment-specific maximum sparsity threshold $\bar{x}_{max}$ for which TempoRL fails to reach meaningful rewards. For each environment, we then evaluate all methods on a set of 6 sparsity coefficients scaled linearly from 0 to $\bar{x}_{max}$, that are referred to as percentages in Figure 9 and 19. The maximum sparsity thresholds $\bar{x}_{max}$ is 12 for HalfCheetah-v2, 2 for Ant-v2, Hopper-v2 and Swimmer-v2 and 1 for Walker-v2.

### H Implementation Details

#### H.1 Metrics

All metrics reported in this paper (e.g. cumulative returns per episode) are averaged over 10 random seeds (excluding gym-mujoco experiments and visual RL experiments, which are respectively limited to 5 and 2 seeds for computational reasons); mean learning curves are smoothed over 5 steps and shaded areas represent standard deviation across seeds.

We rely on two metrics for measuring the quality of exploration trajectories in Section 5.2:

- **%Coverage** divides the reachable state space in $n$ cubic buckets ($n = 1000$ for reach and $n = 100$ for room) and reports the ratio between the number of buckets visited by a set of 20 trajectories of length 500 and the total number of buckets.

- **Radius of Gyration Squared** measures the spread in visited states, averaged over all trajectories, and is adapted from Amin et al. (2021). Given a set $T$ of $n$ trajectories, the metric can be computed as:

  $$U_g^2(T) = \frac{1}{\delta n} \sum_{\tau \in T} \frac{1}{|\tau| - 1} \sum_{s \in \tau} d^2(s, \bar{\tau}),$$

  where a trajectory $\tau$ is modeled as a sequence of states $(s_i)_{i=0}^{|\tau|-1}$, $d^2(\cdot, \cdot)$ measures the Euclidean distance and $\bar{\tau} = \frac{1}{|\tau|} \sum_{s \in \tau} s$. We additionally normalize the metric by $\delta$, which measures the diagonal of the box containing reachable states.

#### H.2 Data Collection

Training tasks for point-maze and meta-world are relatively simple, and allow collecting demonstrations through a scripted policy. Since these environments do not include obstacles and allow direct control over the agent position (in 2D for room and in 3D for reach), scripted policies simply receive the 2D/3D positions of
the agent and of its goal, and output a distance vector. This vector is then corrupted with isotropic Gaussian noise and scaled to fit within action limits. A simple implementation of scripted policies is available as part of our codebase (see URL in Footnote [1]). For each training environment (reach or room), we collect 4000 trajectories of 500 steps each. Goals for the scripted policy are sampled uniformly from the reachable state space. On goal achievement, a new goal is sampled, once again uniformly. In the context of gym-mujoco experiments, 1000 trajectories of 500 steps each are instead collected by fully-trained SAC Haarnoja et al. (2018) agents. The same expert datasets are then used for training temporal priors, behavioral priors or behavioral cloning.

H.3 Baselines

SAC We build upon the implementation provided by SpinningUp (Achiam, 2018), which is reported to be roughly on-par with the best results achieved on gym-mujoco (Brockman et al., 2016). For simplicity, we do not use automatic entropy tuning and keep $\alpha$ constant during learning. Nonetheless, our method could easily tune $\alpha$ dynamically at the expense of increased complexity, as is described in Haarnoja et al. (2018). Moreover, we introduce n-step returns (Hessel et al. 2018) with $n = 10$. We experimented with importance sampling for off-policy correction, but, similarly to what is reported by Hessel et al. (2018), we observed no empirical benefit. All remaining hyperparameters are reported in Table 1. We anticipate that all baselines also rely on n-step returns.

SAC+BC Behavioral cloning (BC) is performed on the entire dataset $D$ for 10 epochs, by maximizing the log-likelihood of the Gaussian policy with respect to expert actions. The optimizer and batch size used for BC are the same as for downstream learning.

SAC-PolyRL SAC-PolyRL (Amin et al., 2021) replaces the initial uniform exploration phase of SAC with trajectories collected by a hand-crafted policy. While SAC’s hyperparameters are unvaried, SAC-PolyRL specific parameters are tuned from those reported in various settings in the original paper. We use $\theta = 0.35$, $\sigma^2 = 0.017$ and $\beta = 0.01$.

PARROT An official implementation for PARROT (Singh et al. 2021) is not available at the time of writing. We reproduce the training routine and downstream application and adopt all hyperparameters reported in the original paper. Generative models are trained until convergence (100 epochs) using a batch size of 400 samples and Adam (Kingma & Ba, 2015) as an optimizer, with a learning rate of 0.0001, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a weight decay penalty of $1e^{-6}$. In addition to the image-based version of PARROT used in Section 5.4, we introduce a version with a vectorized state space, dubbed PARROT-state, since the method is originally only applicable in visual settings. The only modification consists in replacing the image encoder with a 3-layer MLP with 256 neurons per layer and ReLU activations. In this setting, the input to the encoder is therefore vector-based (non-visual).

H.4 TempoRL

Prior We model all families of temporal priors with conditional Real NVP Flows (Ardizzone et al., 2019), sharing the same architecture across all experiments. The invertible transformation $f_\theta$ is a composition of 6 coupling layers, each followed by a batch-normalization layer (Dinh et al., 2017). For each layer, a 3-layer MLP with 128 hidden units per layer is used to preprocess the conditioning input. Scale and transition networks are also implemented as a shared 3-layer MLP with 128 hidden units, whose output is split in two to provide scale and shift coefficients. All MLPs use ReLU activations. We train our temporal priors following the same protocol used with behavioral priors (100 epochs, a batch size of 400 samples and Adam (Kingma & Ba, 2015) with a learning rate of 0.0001, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a weight decay penalty of $1e^{-6}$).

Downstream Learning Our method does not introduce additional hyperparameters with respect to SAC, excluding those involved with the design and optimization of the neural network parameterizing the mixing function $\Lambda_\omega$. This is implemented as a 2-layer ReLU network with 128 units per layer and a sigmoid for output activation. We report that scaling the gradients for $\omega$ by a fixed coefficient $\epsilon$ was found beneficial for a stabler training of the mixing network ($\epsilon = 1e^{-9}$ for meta-world, point-maze and $\epsilon = 1e^{-7}$ for
Table 1: Hyperparameters for SAC.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Epochs</strong></td>
<td>125</td>
</tr>
<tr>
<td><strong>Steps Per Epoch</strong></td>
<td>4e3</td>
</tr>
<tr>
<td><strong>Steps of Initial Exploration</strong></td>
<td>1e4</td>
</tr>
<tr>
<td><strong>Steps Before Training</strong></td>
<td>1e3</td>
</tr>
<tr>
<td><strong>Environment Steps per Iteration</strong></td>
<td>50</td>
</tr>
<tr>
<td><strong>γ</strong></td>
<td>0.99</td>
</tr>
<tr>
<td><strong>Polyak Averaging Rate</strong></td>
<td>0.995</td>
</tr>
<tr>
<td><strong>Inverse of Reward Scale (α)</strong></td>
<td>0.2 for meta-world and gym-mujoco, 0.02 for point-maze</td>
</tr>
<tr>
<td><strong>Batch Size</strong></td>
<td>100</td>
</tr>
<tr>
<td><strong>Optimizer</strong></td>
<td>Adam</td>
</tr>
<tr>
<td><strong>β₁</strong></td>
<td>0.9</td>
</tr>
<tr>
<td><strong>β₂</strong></td>
<td>0.999</td>
</tr>
<tr>
<td><strong>Learning Rate</strong></td>
<td>0.001</td>
</tr>
<tr>
<td><strong>Hidden Units</strong></td>
<td>256</td>
</tr>
<tr>
<td><strong>Hidden Layers</strong></td>
<td>2</td>
</tr>
<tr>
<td><strong>Hindsight Replay Ratio</strong></td>
<td>4</td>
</tr>
<tr>
<td><strong>Replay Size</strong></td>
<td>5e5 for vector-based RL, 2e5 for image-based RL</td>
</tr>
</tbody>
</table>

As exploration is mainly driven by the prior, TempoRL also relies on lower α parameters, namely α = 0.01 for meta-world, point-maze and α = 0.005 for gym-mujoco.

Finally, we introduce two implementation choices motivated by the prior knowledge that, during early stages of training, exploration should mainly be driven by samples from the action prior \( \bar{\pi} \). First, while SAC samples action uniformly for the first 10000 steps to encourage exploration, TempoRL directly samples actions from its prior instead. Second, we initialize the bias parameter of the final layer of the mixing network \( \Lambda_\omega \) to \( \sigma^{-1}(\lambda_0) \) in order to target mixing weights around \( \lambda_0 = 0.95 \) during early training.

All remaining hyperparameters are shared with SAC (see Table 1).

**Test-time Policy** At test time, instead of sampling an action from the mixture between prior \( \bar{\pi} \) and policy \( \pi \), the agent executes an action corresponding to the mean of the policy distribution. We note that this is consistent to what is done in SAC [Haarnoja et al. (2018)].

I Detailed Results

This section reports individual results were aggregated in previously occurring figures. In particular, Figure 15 and Figure 16 report performance for each meta-world and point-maze task, respectively. Figures 17 and 18 also report performance for the two suites, but focus on visual RL settings. Finally, Figure 19 disentangles the results from Figure 9 across the five mujoco-gym environments.

For completeness, we also include plots for meta-world tasks that remain out of reach of all tested methods, namely peg-insert-side, pick-place and push. In general, these tasks require both deep exploration and precise grasping and control. While TempoRL remains capable of deep exploration, precise manipulation constitutes a challenge, as expert demonstrations do not contain grasping behavior (see Section 6). In visual settings, due to increased complexity, we observed a drop in performance on reach, button-press and door-open across all methods.
Figure 15: Learning curves for each downstream meta-world task.

Figure 16: Learning curves for each downstream point-maze task.

Figure 17: Learning curves for each downstream meta-world task in visual settings.

Figure 18: Learning curves for each downstream point-maze task in visual settings.
Figure 19: Mean performance on each gym-mujoco environment for different sparsity thresholds and bias norms.