Zero-Shot Reinforcement Learning Under Partial Observability

Anonymous authors
Paper under double-blind review

Keywords: Zero-shot RL, Behaviour Foundation Models, POMDPs.

Summary

Recent work has shown that, under certain assumptions, zero-shot reinforcement learning (RL) methods can generalise to *any* unseen task in an environment after reward-free pre-training. Access to Markov states is one such assumption, yet, in many practical applications, the Markov state is only *partially observed* via unreliable or incomplete observations. Here, we explore how the performance of standard zero-shot RL methods degrades when subjected to partially observability, and show that, as in single-task RL, memory-based architectures are an effective remedy. We evaluate our *memory-based* zero-shot RL methods in domains where we simulate unreliable states by adding noise or dropping them randomly, and in domains where we simulate incomplete observations by changing the dynamics between training and testing rewards without communicating the change to the agent. In these settings, our proposals show improved performance over memory-free baselines, which we pay for with slower, less stable training dynamics.

Contribution(s)

1. We explore the empirical failure modes state-of-the-art zero-shot RL methods (specifically forward-backward representations, or FB) given partially observed (noisy) states.
   Context: None
2. We present a new architecture called FB with memory (FB-M) which has a memory-based forward model F, backward model B and policy π. Though we develop our method within the FB framework, our proposals are fully compatible with other zero-shot RL methods.
   Context: Prior zero-shot RL methods, including FB (Touati & Ollivier, 2021) and USF-based HILP (Borsa et al., 2018; Park et al., 2024b), are memory-free.
3. We show that, in aggregate, FB-M outperforms memory-free FB and HILP, as well as a naïve observation-stacking baseline, in domains where the states are noisy or randomly dropped, or where there is a change in dynamics function between training and testing.
   Context: None
4. We report better performance when the memory model is a GRU than when it is a transformer or S4d model.
   Context: This aligns with Morad et al. (2023)'s finding that GRUs were the most performant memory model on POPGym, a partially observed single-task RL benchmark.

Zero-Shot Reinforcement Learning Under Partial Observability

Anonymous authors
Paper under double-blind review

Abstract

Recent work has shown that, under certain assumptions, zero-shot reinforcement learning (RL) methods can generalise to *any* unseen task in an environment after reward-free pre-training. Access to Markov states is one such assumption, yet, in many real-world applications, the Markov state is only *partially observable*. Here, we explore how the performance of standard zero-shot RL methods degrades when subjected to partially observability, and show that, as in single-task RL, memory-based architectures are an effective remedy. We evaluate our *memory-based* zero-shot RL methods in domains where the states, rewards and a change in dynamics are partially observed, and show improved performance over memory-free baselines. Our anonymised code is available via: https://anonymous.4open.science/r/rlc2025/.

1 Introduction

Large-scale unsupervised pre-training has proven an effective recipe for producing vision (Rombach et al., 2022) and language (Brown et al., 2020) models that generalise to unseen tasks. The zero-shot reinforcement learning (RL) problem setting (Touati et al., 2023) requires us to produce sequential decision-making agents with similar generality. It asks, informally: can we pre-train agents from datasets of reward-free trajectories such that they can immediately generalise to *any* unseen reward function at test time? A family of methods called *behaviour foundation models* (BFMs) (Touati & Ollivier, 2021; Jeen et al., 2024; Pirotta et al., 2024) theoretically solve the zero-shot RL problem under certain assumptions (Touati & Ollivier, 2021), and empirically return near-optimal policies for many unseen goal-reaching and locomotion tasks (Touati et al., 2023).

These results have assumed access to Markov states that provide all the information the agent requires to solve a task. Though this is a common assumption in RL, for many interesting problems, the Markov state is only *partially observed* via unreliable or incomplete observations (Kaelbling et al., 1998). Observations can be unreliable because of sensor noise or issues with telemetry (Meng et al., 2021). Observations can be incomplete because of egocentricity (Tirumala et al., 2024), occlusions (Heess et al., 2015) or because they do not communicate a change to the environment's task or dynamics context (Hallak et al., 2015).

How do BFMs fare when subjected to partial observability? That is the primary question this paper seeks to answer, and one we address in three parts. First, we expose the mechanisms that cause the performance standard BFMs to degrade under partial observability (Section 4.1). Second, we repurpose methods that handle partial observability in single-task RL for use in the zero-shot RL setting, that is, we add *memory models* to the BFM framework (Section 4.2, Figure 1). Third, we conduct experiments that test how well BFMs augmented with memory models manage partially observed states (Section 5.2) and partially observed changes in dynamics (Section 5.3). We conclude by discussing limitations and next steps.

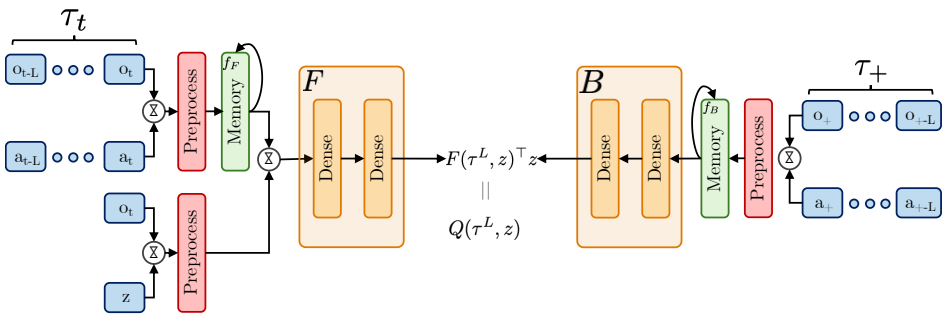

Figure 1: **BFMs with memory**. In the case of FB, the forward model F and backward model B condition on the output of memory models that compress trajectories of observations and actions. According to standard FB theory, their dot product predicts $M^{\pi_z}(\tau_0, \tau_+)$, the successor measure from initial trajectory τ_0 to future trajectory τ_+, from which a Q function can be derived. Figure 8 illustrates memory-free FB for comparison.

2 Related Work

2.1 Zero-shot RL

Offline RL An important part of the zero-shot RL problem is that agents must pre-train on static datasets (Section 3). This is the realm of offline RL (Lange et al., 2012; Levine et al., 2020), where regularisation techniques (Kumar et al., 2020; Kidambi et al., 2020; Fujimoto & Gu, 2021) are used to minimise the *distribution shift* between the offline data and online experience (Kumar et al., 2019b). In this work, we only train on high-coverage datasets to isolate the problem of partial observability, so do not require such regularisation, but past work has repurposed these for zero-shot RL (Jeen et al., 2024). Standard offline RL methods are trained with respect to one downstream task, so cannot generalise to new tasks at test time, as specified by the zero-shot RL problem.

Goal-conditioned RL For goal-reaching tasks, zero-shot goal generalisation can be achieved with via *goal-conditioned* RL (GCRL) (Schaul et al., 2015; Andrychowicz et al., 2017). Here, policies are trained to reach any goal state from any other state. Past work has focused on constructing useful goal-space encodings, with contrastive (Eysenbach et al., 2022), state-matching (Ma et al., 2022), and hierarchical representations (Park et al., 2024a) proving effective. However, GCRL methods do not reliably generalise to *dense* reward functions that cannot be codified by a goal state,[1] and so cannot be said to solve the general zero-shot RL problem.

Behaviour foundation models To date, BFMs have shown the best zero-shot RL performance because they provide a mechanism for zero-shot generalising to *both* goal-reaching and dense reward functions.[2] They build upon successor representations (Dayan, 1993), universal value function approximators (Schaul et al., 2015), successor features (Barreto et al., 2017) and successor measures (Blier et al., 2021). State-of-the-art methods instantiate these ideas as either universal successor features (USFs) (Borsa et al., 2018; Park et al., 2024b) or forward-backward (FB) representations (Touati & Ollivier, 2021; Touati et al., 2023; Jeen et al., 2024). No works have yet explored the zero-shot RL performance of these methods under partial observability.

2.2 Partial Observability

States Most past works assume it is the *state* that is partially observed. This is usually the result of noisy (Meng et al., 2021), occluded (Heess et al., 2015), aliased (Whitehead & Ballard, 1990), egocentric (Tirumala et al., 2024) or otherwise unreliable observations. Standard solutions methods use histories of observations and actions to compute *beliefs* over the true state via (approximate) Bayesian inference (Cassandra et al., 1994; Kaelbling et al., 1998) or via memory-based architectures (Schmidhuber, 1990; Bakker, 2001; Hausknecht & Stone, 2015; Ha & Schmidhuber, 2018).

[1] Examples include any locomotion task *e.g.* Walker-`run` in the DeepMind Control Suite.
[2] A formal justification of this statement is left for Section 3.

Dynamics Sometimes, parameters that modulate the underlying *dynamics* change and are not communicated to the agent via the state. Given sets of training and testing dynamics parameters, *generalisation* is a measure of the agent's average-case performance on the test set (Packer et al., 2018; Cobbe et al., 2019). If the agent trains and tests on the same set of dynamics, *robustness* is a measure of the agent's worst-case performance on this set (Nilim & El Ghaoui, 2005; Morimoto & Doya, 2005; Mankowitz et al., 2019). Generalisation can be improved via regularisation (Farebrother et al., 2018), data augmentation (Tobin et al., 2017; Raileanu et al., 2020; Ball et al., 2021), or dynamics context modelling (Seo et al., 2020; Lee et al., 2020). Robustness can be improved with adversarial dynamics selection (Rajeswaran et al., 2016; Jiang et al., 2021; Rigter et al., 2023).

Rewards In some cases, the utility of an action for a task may only be partially reflected in the standard one-step reward (Minsky, 1961; Sutton, 1984). Such a situation arises when the reward signal is delayed (Arjona-Medina et al., 2019) or is dependent on the entire trajectory (i.e. episodic) (Liu et al., 2019). These have traditionally been handled with sophisticated techniques that learn surrogate reward functions (Raposo et al., 2021; Arjona-Medina et al., 2019), tune discount factors (Fedus et al., 2019), or utilise eligibility traces (Xu et al., 2020), among other methods.

Each of the above methods were developed to form of partial observability, but memory-based architectures are, in principle, general enough to solve all of them (Kaelbling et al., 1998). Indeed, Ni et al. (2021) find that a standard, but well-implemented, recurrent policy and critic can outperform methods specialised for each setting. Our proposed method (Section 4.2) is heavily informed by this finding, and is designed to be agnostic to the specific way in which partial observability arises.

3 Preliminaries

POMDPs A partially observable Markov decision process (POMDP) $\mathcal{P}$ is defined by $(\mathcal{S}, \mathcal{A}, \mathcal{O}, R, P, O, \mu_0, \gamma)$, where $\mathcal{S}$ is the set of Markov states, $\mathcal{A}$ is the set of actions, $\mathcal{O}$ is the set of observations, and μ_0 is the initial state distribution (Åström, 1965; Kaelbling et al., 1998). Let $s_t \in \mathcal{S}$ denote the Markov state at time t. When action $a_t \in \mathcal{A}$ is executed, the state updates via the transition function $s_{t+1} \sim P(\cdot|s_t, a_t)$, and the agent receives a scalar reward $r_{t+1} \sim R(s_{t+1})$ and observation $o_{t+1} \sim O(\cdot|s_{t+1}, a_t)$. The observation provides only partial information about the underlying Markov state. The agent samples actions from its policy $a_t \sim \pi(\cdot|\tau_t^L)$, where $\tau_t^L = (a_{t-L}, o_{t-L+1}, \dots, a_{t-1}, o_t)$ is a *trajectory* of the preceding L observations and actions. We use $\mathcal{T}^L$ to denote the set of all possible trajectories of length L. The policy is optimal in $\mathcal{P}$ if it maximises the expected discounted future reward *i.e.* $\pi^* = \arg\max_\pi \mathbb{E}[\sum_{t\geq 0} \gamma^t R(s_{t+1})|s_0, a_0, \pi]$, where $\mathbb{E}[\cdot|s_0, a_0, \pi]$ denotes an expectation over state-action sequences $(s_t, a_t)_{t\geq 0}$ starting at (s_0, a_0) with $s_t \sim P(\cdot|s_{t-1}, a_{t-1})$ and $a_t \sim \pi(\cdot|\tau_t^L)$, and $\gamma \in [0, 1]$ is a discount factor.

Partially observable zero-shot RL In the standard zero-shot RL problem setting, states are fully observed. For pre-training, the agent has access to a static offline dataset of reward-free transitions $\mathcal{D} = \{(s_i, a_i, s_{i+1})\}_{i=1}^{|\mathcal{D}|}$, generated by an unknown behaviour policy. At test time, a task R_{test} is revealed via a small dataset of reward-labelled states $\mathcal{D}_{\text{labelled}} = \{(s_i, R_{\text{test}}(s_i))\}_{i=1}^{k}$ where typically $k \leq 10,000$. The agent must return a policy for this task with no further planning or learning.

In this work, we consider the extended problem setting of *partially observable zero-shot RL*. Here, the agent has access to an offline pre-training dataset of reward-free length-L trajectories, $\mathcal{D} = \{\tau_i^L\}_{i=1}^{|\mathcal{D}|}$, each of which is a sequence of partial observations and actions. As before, a task R_{test} is revealed at test time, for which the agent must return a policy. The task is specified by a small dataset of reward-labelled observation-action trajectories, where the reward is assumed to be a function of the final Markov state in the trajectory, $\mathcal{D}_{\text{labelled}} = \{(\tau_i^L, R_{\text{test}}(s_i^L))\}_{i=1}^{k}$.

Behaviour foundation models We build upon the forward-backward (FB) BFM which predicts successor measures (Blier et al., 2021). The successor measure $M^\pi(s_0, a_0, \cdot)$ over $\mathcal{S}$ is the cumulative discounted time spent in each future state s_{t+1} after starting in state s_0, taking action a_0, and following policy π thereafter. Let ρ be an arbitrary state distribution and $\mathbb{R}^d$ be an embedding space. FB representations are composed of a *forward* model $F : S \times A \times \mathbb{R}^d \to \mathbb{R}^d$, a *backward* model

$B : S \to \mathbb{R}^d$, and set of polices $\pi(s, z)_{z \in \mathbb{R}^d}$. They are trained such that

$$M^{\pi_z}(s_0, a_0, X) \approx \int_X F(s_0, a_0, z)^\top B(s) \rho(\mathrm{d}s) \qquad \forall \ s_0 \in \mathcal{S}, a_0 \in \mathcal{A}, X \subset \mathcal{S}, z \in \mathbb{R}^d, \quad (1)$$

$$\pi(s, z) \approx \arg\max_a F(s, a, z)^\top z \qquad \forall \ (s, a) \in \mathcal{S} \times \mathcal{A}, z \in \mathbb{R}^d, \quad (2)$$

where $F(s, a, z)^\top z$ is the Q function (critic) formed by the dot product of forward embeddings with a *task embedding* z. During training, candidate task embeddings are sampled from $\mathcal{Z}$, a prior over the embedding space. During evaluation, the test task embeddings are inferred from $\mathcal{D}_{\text{labelled}}$ via:

$$z_{\text{test}} \approx \mathbb{E}_{(s, R_{\text{test}}(s)) \sim \mathcal{D}_{\text{labelled}}}[R_{\text{test}}(s) B(s)], \quad (3)$$

and passed as an argument to the policy.

4 Zero-Shot RL Under Partial Observability

In this section, we adapt BFMs for the partially observable zero-shot RL problem. In Section 4.1, we explore the ways in which standard BFMs fail in this setting. Then, in Section 4.2, we propose new methods that address these failures. We develop our methods in the context of FB, but our proposals are fully compatible with USF-based BFMs. We leave their derivation to Appendix D for brevity.

4.1 Failure Mode of Existing Methods

FB solves the zero-shot RL problem in two stages. First, a generalist policy is pre-trained to maximise FB's Q functions for all tasks sampled from the prior $\mathcal{Z}$ (Equation 1). Second, the test task is inferred from reward-labelled states (Equation 3) and passed to the policy. The first stage relies on an accurate approximation of $F(s, a, z)$ *i.e.* the long-run dynamics of the environment subject to a policy attempting to solve task z. The second stage relies on an accurate approximation of $B(s)$ *i.e.* the task associated with reaching state s. If the states in F are replaced by observations that only partially characterise the underlying state, then the BFM will struggle to predict the long-run dynamics, Q functions derived from F will be inaccurate, and the policy will not learn optimal sequences of actions. We call this failure mode **state misidentification** (Figure 2, middle). Likewise, if the states in B are replaced by partial observations, and the reward function depends on the underlying state (Section 3), then the BFM cannot reliably find the task z associated with the set of states that maximise the reward function. We call this failure mode **task misidentification** (Figure 2, left). The failure modes occur together when both models receive partial observations (Figure 2, right).

4.2 Addressing Partial Observability with Memory Models

In principle, all forms of partial observability can be resolved with *memory models* that compress trajectories into a hidden state that approximates the underlying Markov state (see Section 2 of Ni et al. (2021)). A memory model is a function f that outputs a new hidden state h_t given a past hidden state h_{t-L-1} and trajectory τ_t^L:

$$h_t = f(\tau_t^L, h_{t-L-1}). \quad (4)$$

Note that by setting $L = 0$, we recover the standard one-step formulation of a recurrent neural network (RNN) (Elman, 1990; Hochreiter & Schmidhuber, 1997; Cho, 2014). RNNs are common choice in past works (Wierstra & Schmidhuber, 2007; Zhang et al., 2016; Schmidhuber, 2019), but more recent works explore structured state space sequence models (S4) (Deng et al., 2023; Lu et al., 2024) and transformers (Parisotto et al., 2020; Grigsby et al., 2023; 2024). In *model-based* partially observable RL, dynamics misidentification is resolved with memory-based dynamics models, and task misidentification is resolved with a memory-based reward models (Hafner et al., 2019a;b; 2020; 2023). In *model-free* partially observable RL, the agent does not disentangle the dynamics from the task, so task and dynamics misidentification are resolved together by memory-based critics and policies (Ni et al., 2021; Meng et al., 2021).

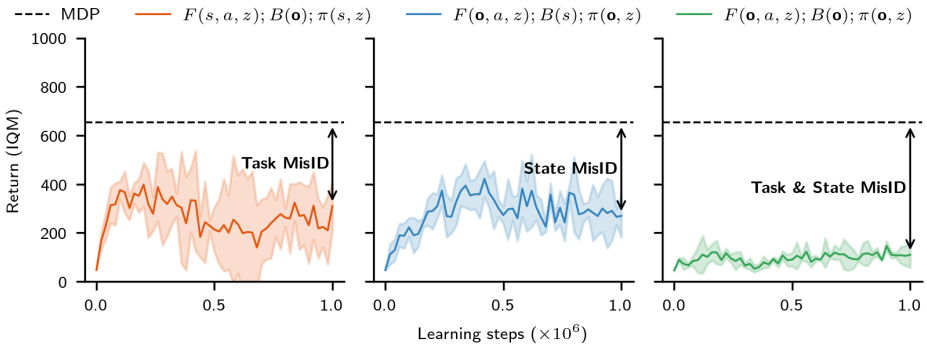

Figure 2: **The failure modes of BFMs under partial observability.** FB's average (IQM) all-task return on Walker when observations are passed to its respective components. Observations are created by adding Gaussian noise to the underlying states. *(Left)* Observations are passed as input to B causing FB to misidentify the task. *(Middle)* Observations are passed as input to F and π causing FB to misidentify the dynamics. *(Right)* Observations are passed as input to F, π and B causing FB to misidentify both the task and dynamics.

4.3 Behaviour Foundation Models with Memory

We now adapt methods from single-task partially observable RL for BFMs. Standard FB operates on states (Equation 1) that are inaccessible under partial observability, so we amend its formulation to operate on trajectories from which the underlying Markov state can be inferred with a memory model. The successor measure $M^\pi(\tau_0^L, \cdot)$ over $\mathcal{T}^L$ is the cumulative discounted time spent in each future trajectory τ_{t+1}^L starting from trajectory τ_0^L, and following policy π thereafter.[3] The architectures of the forward model F, backward model B, and policy π are unchanged, but now condition on the hidden states of memory models, rather than on states and actions. They are trained such that

$$M^{\pi_z}(\tau_0^L, X) \approx \int_X F(f_F(\tau_0^L), z)^\top B(f_B(\tau^L))\rho(\mathrm{d}\tau^L) \qquad \forall\, \tau_0^L \in \mathcal{T}^L, X \subset \mathcal{T}^L, z \in \mathbb{R}^d,$$
$$\pi(f_\pi(\tau^L), z) \approx \arg\max_a F(f_F(\tau^L), z)^\top z \qquad \forall\, \tau^L \in \mathcal{T}^L, z \in \mathbb{R}^d.$$

where f_F, f_B, f_π are separate memory models for F, B, and π respectively, and the previous hidden state h_{t-L-1} is dropped as an argument for brevity (*c.f.* Equation 4). At test time, task embeddings are found via Equation 3, but with reward-labelled trajectories rather than reward-labelled states:

$$z_{\text{test}} \approx \mathbb{E}_{(\tau^L, R(s)) \sim \mathcal{D}_{\text{labelled}}}[R_{\text{test}}(s) B(f_B(\tau^L))]. \tag{5}$$

We refer to the resulting model as *FB with memory* (FB-M). The full architecture is summarised in Figure 1, and implementation details are provided in Appendix E. Also note that our general proposal is BFM-agnostic; we derive the USF-based BFM formulation in Appendix D.

5 Experiments

5.1 Setup

We evaluate our proposals in two partially observed settings: 1) partially observed states (*i.e.* standard POMDPs), and partially observed changes in dynamics (*i.e.* generalisation (Packer et al., 2018)). The standard benchmarks for each of these settings only require the agent to solve one task, and so do not allow us to evaluate zero-shot RL capabilities out-of-the-box. As a result, we choose to amend the standard zero-shot RL benchmark, ExORL (Yarats et al., 2022), such that it tests zero-shot RL with partially observed states and dynamics changes.

[3] Note that the forward model and backward model can in principle have different context lengths. This is helpful if, for example, we know that the reward, as inferred via the backward model, depends on a shorter history length than would be required to infer the full Markov state via the forward model.

Partially observed states We amend two of Meng et al. (2021)'s partially observed state environments for the zero-shot RL setting: 1) `noisy` states, where isotropic zero-mean Gaussian noise with variance σ_{noise} is added to the Markov state, and 2) `flickering` states, where states are dropped (zeroed) with probability $p_{\text{flick.}}$. We set $\sigma_{\text{noise}} = 0.2$ and $p_{\text{flick.}} = 0.2$ following a hyperparameter study (Appendix B). We evaluate on all tasks in the Walker, Cheetah and Quadruped environments.

Partially observed changes in dynamics We amend Packer et al. (2018)'s dynamics generalisation tests for the zero-shot RL setting. Environment dynamics are modulated by scaling the mass and damping coefficients in the MuJoCu backend (Todorov et al., 2012). The agents are trained on datasets collected from environment instances with coefficients scaled to $\{0.5\times, 1.5\times\}$ their usual values, then evaluated on environment instances with coefficients scaled by $\{1.0\times, 2.0\times\}$. Scaling by $1.0\times$ tests the agent's ability to generalise via *interpolation* within the range seen during training, and scaling by $2.0\times$ tests the agent's ability to generalise via *extrapolation* (Packer et al., 2018).

Baselines We use two state-of-the-art zero-shot RL methods as baselines: FB (Touati & Ollivier, 2021) and HILP (Park et al., 2024b). We additionally implement a naïve baseline called FB-stack whose input is a *stack* of the 4 most recent observations and actions, following Mnih et al. (2015)'s canonical protocol. Finally, we use FB trained on the underlying MDP as an oracle policy to give us an upper-bound on expected performance.

Datasets We train all methods on datasets collected with an RND behaviour policy (Borsa et al., 2018) as these are the datasets that elicit best performance on ExORL. The RND datasets used in the partially observed states experiments are taken directly from ExORL. For the partially observed change in dynamics and partially observed rewards experiments, we collect these datasets ourselves by running RND in each of the environments for 5 million learning steps. Our implementation and training protocol exactly match ExORL's.

Memory model We use a GRU as our memory model (Cho, 2014). GRUs are the most performant memory model on POPGym (Morad et al., 2023) which tests partially observed single-task RL methods. We find these results hold for partially observed zero-shot RL too, as discussed in Section 6.1. We set the forward model's context length $L_F = 32$ and the backward model's context length $L_B = 8$. See Appendix A for a hyperparameter study and further discussion.

5.2 Partially Observed States

Figure 3 compares the zero-shot performance of all algorithms our `noisy` and `flickering` variants of the standard ExORL environments. Note that these results are aggregated across all tasks in each environment, and 5 random seeds. The performance of memory-free FB is always far below that of an oracle policy trained on the underlying MDP (dotted line), reaching less than 25% of the oracle value in 5 out of 6 cases, and HILP performs similarly. Augmenting FB by stacking recent observations mitigates the partial observability problem to some extent in each case, but our approach using memory models (FB-M) always outperforms this baseline. The benefit of FB-M is most pronounced for the Quadruped environment (where it achieves close to oracle performance), and generally larger for the `flickering` mode of partial observability than for the `noisy` mode.

5.3 Partially Observed Changes in Dynamics

Next, we consider the problem of partially observed dynamics changes in both the interpolation and extrapolation regimes. The trends of results in this context, shown in Figure 4, are somewhat more complex. Firstly, we find that the algorithms achieve lower performance than oracle policies given direct access to the underlying MDP, and struggle more with adapting to dynamics in the extrapolation regime (with the exception of FB-M on Cheetah), matching expectations. A second consistent trend is that HILP is far less performant than FB. More importantly, our FB-M proposal boosts or maintains the performance of memory-free FB in all cases, bringing a greater advantage in the more challenging dynamics extrapolation regime. While the same can be said for the observation stacking baseline in the Walker environment, this crucially does not hold for Cheetah and Quadruped, where

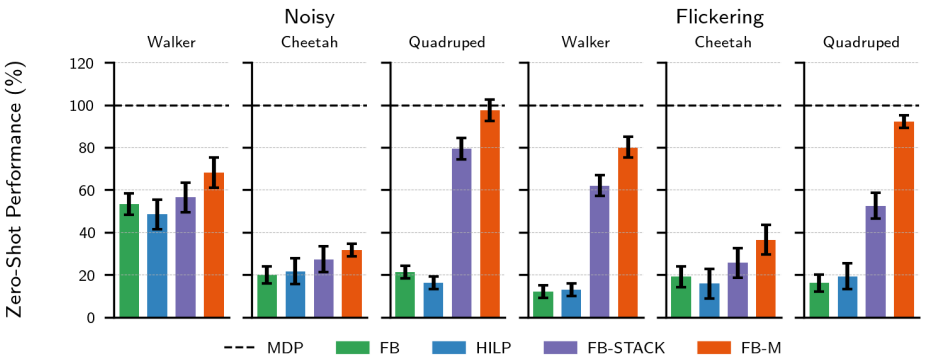

Figure 3: **Aggregate zero-shot task performance on ExORL with partially observed states.** IQM of task scores across all tasks on `noisy` and `flickering` variants of Walker, Cheetah and Quadruped, normalised against the performance of FB in the fully observed environment. 5 random seeds.

it actually *degrades* the performance relative to memory-free FB. Overall, these results suggest that FB-M brings a far more consistent benefit under changed dynamics than frame stacking, at the very least matching the memory-free performance, and often substantially improving it.

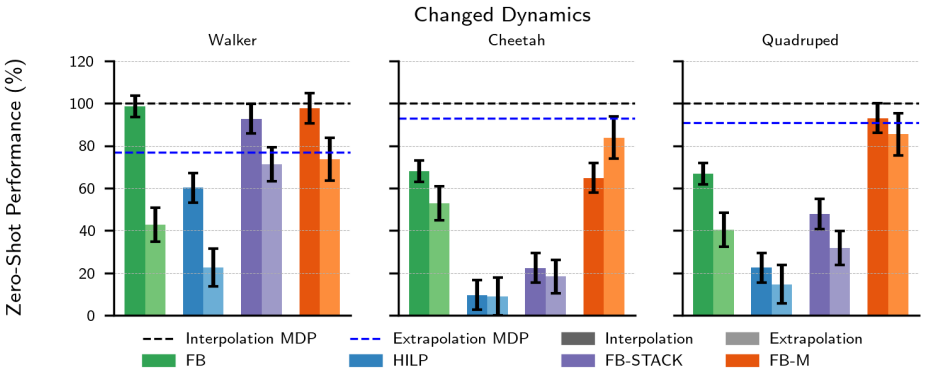

Figure 4: **Aggregate zero-shot task performance on ExORL with changed dynamics at test time.** IQM of task scores across all tasks when trained on dynamics where mass and damping coefficients are scaled to $0.5\times$, $1.5\times$ their usual values and evaled on $1.0\times$, $2.0\times$ their usual values. To solve the test dynamics with $1.0\times$ scaling the agent must interpolate within the training set; to solve the test dynamics with $2.0\times$ scaling the agent must extrapolate from the training set.

6 Discussion and Limitations

6.1 Memory Model Choice

Our method uses GRUs as memory models, but much recent work has shown that transformers (Vaswani et al., 2017) and structured state-space models (Gu et al., 2021) outperform GRUs in natural language processing (Brown et al., 2020), computer vision (Dosovitskiy et al., 2020), and model-based RL (Deng et al., 2023). In this section, we explore whether these findings hold for the zero-shot RL setting. We compare FB-M with GRU memory models to FB-M with transformer and diagonalised S4 (S4d) memory models (Gu et al., 2022). We follow (Morad et al., 2023) in restricting each model to a fixed hidden state size, rather than a fixed parameter count, to ensure a fair comparision. Concretely, we allow each model a hidden state size of $32^2 = 1024$ dimensions. Full implementation details are provided in Appendix E. We evaluate each method in the three

variants of Walker `flickering` used in Section 4.1 *i.e.* where only the inputs to F and π_z are observations, only inputs to B are observations, and where inputs to all models are observations.

Our results are reported in Figure 5. We find that the performance of FB-M is reduced in all cases when a transformer or S4 memory model is used instead of a GRU. This corroborates Morad et al. (2023)'s findings that the GRU is the most performant memory model for single-task partially observed RL. Perhaps most crucially, we find that training collapses when *both* F and B are non-GRU memory models, despite non-GRU memory models performing reasonably when added to *only* F or B, suggesting that the combined representation $M(\tau^L, \tau^L_+) \approx F(f_F(\tau^L))^\top B(f_B(\tau^L_+))$ is degenerate. Better understanding this failure mode is important future work.

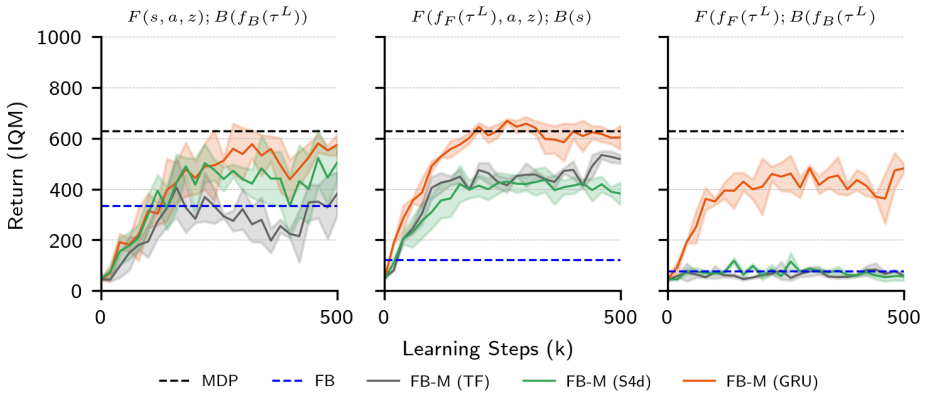

Figure 5: **Aggregate zero-shot task performance of FB-M with different memory models. IQM of task scores across all tasks on Walker `flickering`.** *(Left)* Observations are passed only to a memory-based backward model; the forward model and policy are memory-free. *(Middle)* Observations are passed only to the forward model and policy; the backward model is memory-free. *(Right)* Observations are passed to all models.

6.2 Datasets

As outlined in Section 5.1, we train all methods on datasets pre-collected with RND (Borsa et al., 2018) which is a highly exploratory algorithm designed for maximising data heterogeneity. However, deploying such an algorithm in any real setting may be costly, time-consuming or dangerous. As a result, our proposals are more likely to be trained on real-world datasets that are smaller and more homogeneous. It is not clear how our specific proposals will interact with such datasets. If, for example, the dataset only represents parts of the state space from which the dynamics cannot be well-inferred, like a robot with limited freedom of movement, then we would expect our proposals to struggle. Indeed, with poor coverage of the state-action space, we would expect to see the OOD pathologies seen in the single-task offline RL setting (Kumar et al., 2019a; Levine et al., 2020). That said, the proposals of (Jeen et al., 2024) for conducting zero-shot RL from less diverse datasets could be integrated into our proposals easily, and may help.

7 Conclusion

In this paper, we explored how the performance of BFMs degrades when subjected to certain types of partial observability. We introduce memory-based BFMs that condition F, B and π_z on trajectories of observation-action pairs, and show they go some way to remedying state and task misidentification. We evaluated our proposals on a suite of partially observed zero-shot RL problems, where the observations passed to the agent are noisy, dropped randomly or do not communicate a change in the underlying dynamics, and showed improved performance over memory-free baselines. We found the GRU to be the most performant memory model, and showed that transformers and s4 memory models cannot be trained stably at our scale. We believe our proposals represent a further step towards the real-world deployment of zero-shot RL methods.

A Model Hyperparameters

A.1 Context Lengths

The context length L of both the F/π_z and B is an important hyperparameter. When adding memory to actors or critics, it is standard practice to parallelise training across batched trajectories of fixed L (zero-padded for all $t < L$), yet condition the policy on the entire episode history during evaluation with recurrent hidden states. If L is chosen to be less than the maximum episode length, as is often required with limited compute, a shift between the training and evaluation distributions is inevitable. Though this does not tend to harm performance significantly (Hausknecht & Stone, 2015), the aim is generally to maximise L subject to available compute. The Markov states of different POMDPs will require different L, but longer L increasing training time and risks decreased training stability. In Figure 6 we sweep across $L \in \{2, 4, 8, 16, 32\}$ for both F/π_z and B. In general, we see small increases in performance for increased context length, and choose $L = 32$ for our main experiments.

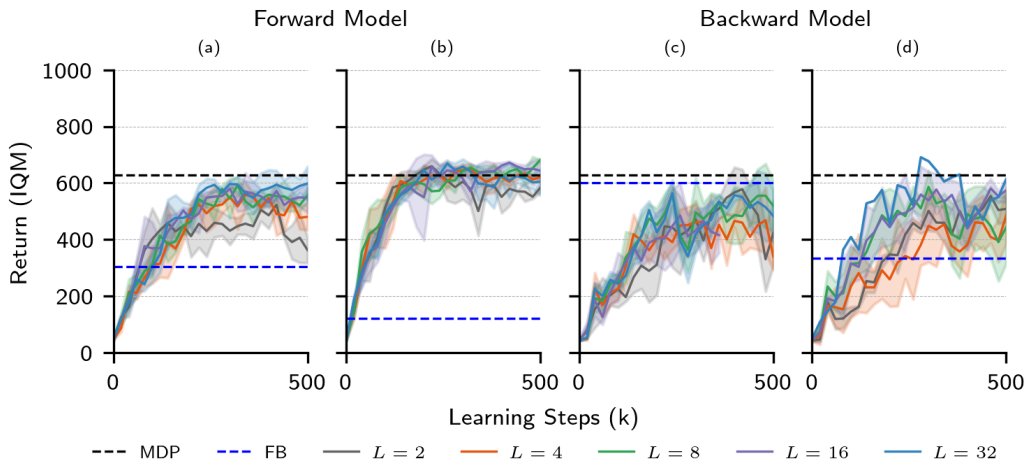

Figure 6: **Hyperparameter sweep over context length L.** We evaluate the performance of FB-M with GRU memory model on Walker `noisy` ((a) and (c) and Walker `flickering` ((b) and (d)). When we sweep over the forward model's context length, we pass states to the backward model and keep it memory-free; when we sweep over the backward model's context length we pass states to the forward model and policy and keep them memory-free.

B POMDP Hyperparameters

The `noisy` and `flickering` amendments to standard ExORL environments (Section 5) have associated hyperparameters σ and p_f. Hyperparameter σ is the variance of the 0-mean Gaussian from which noise is sampled before being added to the state, and p_f is the probability that state s is dropped (zeroed) at time t. In Figure 7 we sweep across three valued of each in $\{0.05, 0.1, 0.2\}$. From these findings we set $\sigma = 0.2$ and $p_f = 0.2$ in the main experiments

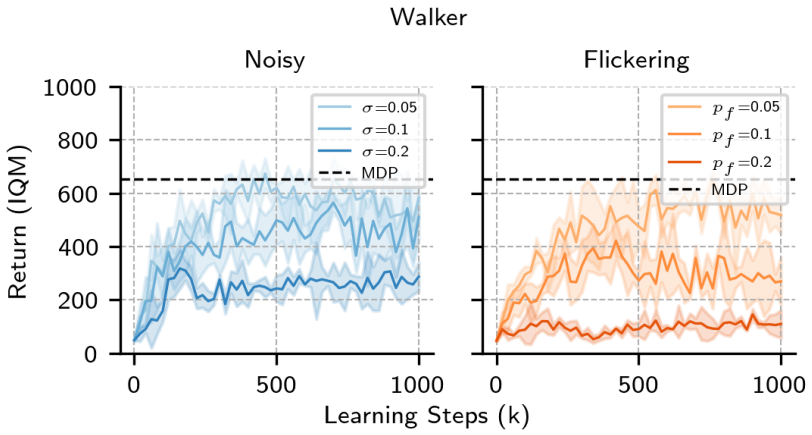

Figure 7: **POMDP hyperparameter sweep.** We evaluate the performance of standard FB on Walker when the states are noised according to $\sigma \in \{0.05, 0.1, 0.2\}$ and dropped according to $p_f \in \{0.05, 0.1, 0.2\}$.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

Karl Johan Åström. Optimal control of markov processes with incomplete state information. *Journal of mathematical analysis and applications*, 10(1):174–205, 1965.

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

Bram Bakker. Reinforcement learning with long short-term memory. *Advances in neural information processing systems*, 14, 2001.

Philip J Ball, Cong Lu, Jack Parker-Holder, and Stephen Roberts. Augmented world models facilitate zero-shot dynamics generalization from a single offline environment. In *International Conference on Machine Learning*, pp. 619–629. PMLR, 2021.

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

Léonard Blier, Corentin Tallec, and Yann Ollivier. Learning successor states and goal-dependent values: A mathematical viewpoint. *arXiv preprint arXiv:2101.07123*, 2021.

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Anthony R Cassandra, Leslie Pack Kaelbling, and Michael L Littman. Acting optimally in partially observable stochastic domains. In *AAAI*, volume 94, pp. 1023–1028, 1994.

Kyunghyun Cho. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International conference on machine learning*, pp. 1282–1289. PMLR, 2019.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural computation*, 5(4):613–624, 1993.

Fei Deng, Junyeong Park, and Sungjin Ahn. Facing off world model backbones: Rnns, transformers, and s4. *Advances in Neural Information Processing Systems*, 36, 2023.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620, 2022.

Jesse Farebrother, Marlos C Machado, and Michael Bowling. Generalization and regularization in dqn. *arXiv preprint arXiv:1810.00123*, 2018.

William Fedus, Carles Gelada, Yoshua Bengio, Marc G Bellemare, and Hugo Larochelle. Hyperbolic discounting and learning over multiple horizons. *arXiv preprint arXiv:1902.06865*, 2019.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Jake Grigsby, Linxi Fan, and Yuke Zhu. Amago: Scalable in-context reinforcement learning for adaptive agents. *International Conference on Learning Representations*, 2023.

Jake Grigsby, Justin Sasek, Samyak Parajuli, Daniel Adebi, Amy Zhang, and Yuke Zhu. Amago-2: Breaking the multi-task barrier in meta-reinforcement learning with transformers. *Advances in Neural Information Processing Systems*, 2024.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. *Advances in Neural Information Processing Systems*, 35:35971–35983, 2022.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019b.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

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

Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes, 2015.

Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 aaai fall symposium series*, 2015.

Nicolas Heess, Jonathan J Hunt, Timothy P Lillicrap, and David Silver. Memory-based control with recurrent neural networks. *arXiv preprint arXiv:1512.04455*, 2015.

Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9 (03):90–95, 2007.

Scott Jeen, Tom Bewley, and Jonathan M. Cullen. Zero-shot reinforcement learning from low quality data. *Advances in Neural Information Processing Systems 38*, 2024.

Yuankun Jiang, Chenglin Li, Wenrui Dai, Junni Zou, and Hongkai Xiong. Monotonic robust policy optimization with model discrepancy. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4951–4960. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/jiang21c.html.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33: 21810–21823, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019a.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019b.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning: State-of-the-art*, pp. 45–73. Springer, 2012.

Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 5757–5766. PMLR, 2020.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Off-policy policy gradient with state distribution correction. *arXiv preprint arXiv:1904.08473*, 2019.

Chris Lu, Yannick Schroecker, Albert Gu, Emilio Parisotto, Jakob Foerster, Satinder Singh, and Feryal Behbahani. Structured state space models for in-context reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Yecheng Jason Ma, Jason Yan, Dinesh Jayaraman, and Osbert Bastani. How far i'll go: Offline goal-conditioned reinforcement learning via f-advantage regression. *arXiv preprint arXiv:2206.03023*, 2022.

Daniel J Mankowitz, Nir Levine, Rae Jeong, Yuanyuan Shi, Jackie Kay, Abbas Abdolmaleki, Jost Tobias Springenberg, Timothy Mann, Todd Hester, and Martin Riedmiller. Robust reinforcement learning for continuous control with model misspecification. *arXiv preprint arXiv:1906.07516*, 2019.

Wes McKinney et al. pandas: a foundational python library for data analysis and statistics. *Python for high performance and scientific computing*, 14(9):1–9, 2011.

Lingheng Meng, Rob Gorbet, and Dana Kulić. Memory-based deep reinforcement learning for pomdps. In *2021 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 5619–5626. IEEE, 2021.

Marvin Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Steven Morad, Ryan Kortvelesy, Matteo Bettini, Stephan Liwicki, and Amanda Prorok. Popgym: Benchmarking partially observable reinforcement learning. *arXiv preprint arXiv:2303.01859*, 2023.

Jun Morimoto and Kenji Doya. Robust reinforcement learning. *Neural computation*, 17(2):335–359, 2005.

Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent model-free rl can be a strong baseline for many pomdps. *arXiv preprint arXiv:2110.05038*, 2021.

Arnab Nilim and Laurent El Ghaoui. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*, 2018.

Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *International conference on machine learning*, pp. 7487–7498. PMLR, 2020.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions. *Advances in Neural Information Processing Systems*, 36, 2024a.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with hilbert representations. *International Conference on Machine Learning*, 2024b.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Matteo Pirotta, Andrea Tirinzoni, Ahmed Touati, Alessandro Lazaric, and Yann Ollivier. Fast imitation via behavior foundation models. In *International Conference on Learning Representations*, 2024.

Roberta Raileanu, Max Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in deep reinforcement learning. *arXiv preprint arXiv:2006.12862*, 2020.

Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

David Raposo, Sam Ritter, Adam Santoro, Greg Wayne, Theophane Weber, Matt Botvinick, Hado van Hasselt, and Francis Song. Synthetic returns for long-term credit assignment. *arXiv preprint arXiv:2102.12425*, 2021.

Marc Rigter, Minqi Jiang, and Ingmar Posner. Reward-free curricula for training robust world models. *arXiv preprint arXiv:2306.09205*, 2023.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Michel F Sanner et al. Python: a programming language for software integration and development. *J Mol Graph Model*, 17(1):57–61, 1999.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pp. 1312–1320. PMLR, 2015.

Juergen Schmidhuber. Reinforcement learning upside down: Don't predict rewards–just map them to actions. *arXiv preprint arXiv:1912.02875*, 2019.

Jürgen Schmidhuber. Reinforcement learning in markovian and non-markovian environments. *Advances in neural information processing systems*, 3, 1990.

Younggyo Seo, Kimin Lee, Ignasi Clavera Gilaberte, Thanard Kurutach, Jinwoo Shin, and Pieter Abbeel. Trajectory-wise multiple choice learning for dynamics generalization in reinforcement learning. *Advances in Neural Information Processing Systems*, 33:12968–12979, 2020.

Richard Stuart Sutton. *Temporal credit assignment in reinforcement learning*. University of Massachusetts Amherst, 1984.

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

Dhruva Tirumala, Markus Wulfmeier, Ben Moran, Sandy Huang, Jan Humplik, Guy Lever, Tuomas Haarnoja, Leonard Hasenclever, Arunkumar Byravan, Nathan Batchelor, et al. Learning robot soccer from egocentric vision with deep reinforcement learning. *arXiv preprint arXiv:2405.02425*, 2024.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30. IEEE, 2017.

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.

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

Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist? In *The Eleventh International Conference on Learning Representations*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Steven D Whitehead and Dana H Ballard. Active perception and reinforcement learning. In *Machine Learning Proceedings 1990*, pp. 179–188. Elsevier, 1990.

Daan Wierstra and Jürgen Schmidhuber. Policy gradient critics. In *European Conference on Machine Learning*, pp. 466–477. Springer, 2007.

Zhongwen Xu, Hado P van Hasselt, Matteo Hessel, Junhyuk Oh, Satinder Singh, and David Silver. Meta-gradient reinforcement learning with an objective discovered online. *Advances in Neural Information Processing Systems*, 33:15254–15264, 2020.

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

Marvin Zhang, Zoe McCarthy, Chelsea Finn, Sergey Levine, and Pieter Abbeel. Learning deep neural network policies with continuous memory states. In *2016 IEEE international conference on robotics and automation (ICRA)*, pp. 520–527. IEEE, 2016.

Supplementary Materials

The following content was not necessarily subject to peer review.

C Experimental Details

C.1 ExORL

We consider 3 environments (three locomotion and one goal-directed) from the ExORL benchmark (Yarats et al., 2022) which is built atop the DeepMind Control Suite (Tassa et al., 2018). Environments are visualised here: https://www.youtube.com/watch?v=rAai4QzcYbs. The domains are summarised in Table 1.

Walker A two-legged robot required to perform locomotion starting from bent-kneed position. The observation and action spaces are 24 and 6-dimensional respectively, consisting of joint torques and positions. ExORL provides 4 tasks `stand`, `walk`, `run` and `flip`. The reward function for `stand` motivates straightened legs and an upright torso; `walk` and `run` are supersets of `stand` including reward for small and large degrees of forward velocity; and `flip` motivates angular velocity of the torso after standing. Rewards are dense.

Quadruped A four-legged robot required to perform locomotion inside a 3D maze. The observation and action spaces are 84 and 12-dimensional respectively, consisting of joint torques and positions. We evaluate on 4 tasks `stand`, `run`, `walk` and `jump`. The reward function for `stand` motivates a minimum torso height and straightened legs; `walk` and `run` are supersets of `stand` including reward for small and large degrees of forward velocity; and `jump` adds a term motivating vertical displacement to stand. Rewards are dense.

Cheetah A running two-legged robot. The observation and action spaces are 17 and 6-dimensional respectively, consisting of positions of robot joints. We evaluate on 4 tasks: `walk`, `walk backward`, `run` and `run backward`. Rewards are linearly proportional either a forward or backward velocity–2 m/s for walk and 10 m/s for run.

C.2 Evaluation Protocol

We evaluate the cumulative reward achieved by all methods across 5 seeds. We report task scores as per the best practice recommendations of (Agarwal et al., 2021). Concretely, we run each algorithm for 500k learning steps (1m for ExORL), evaluating task scores at checkpoints of 20,000 steps. At each checkpoint, we perform 10 rollouts, record the score of each, and find the interquartile mean (IQM). We average across seeds at each checkpoint. We extract task scores from the learning step for which the all-task IQM is maximised across seeds. Results are reported with their associated standard deviation. Aggregation across tasks, domains and datasets is always performed by evaluating the IQM.

C.3 Computational Resources

We train our models on NVIDIA A100 GPUs. One run of FB, FB-stack and HILP on one domain (for all tasks) takes approximately 6 hours on one GPU. One run of the FB-M on one domain (for

Table 1: **ExORL domain summary.** *Dimensionality* refers to the relative size of state and action spaces. *Type* is the task categorisation, either locomotion (satisfy a prescribed behaviour until the episode ends) or goal-reaching (achieve a specific task to terminate the episode). *Reward* is the frequency with which non-zero rewards are provided, where dense refers to non-zero rewards at every timestep and sparse refers to non-zero rewards only at positions close to the goal. Green and red colours reflect the relative difficulty of these settings.

Environment	Dimensionality	Type	Reward
Walker	Low	Locomotion	Dense
Quadruped	High	Locomotion	Dense
Cheetah	Low	Locomotion	Dense

all tasks) on one GPU in approximately 20 hours. As a result, our core experiments on the ExORL benchmark used approximately 65 GPU days of compute.

D Universal Successor Features with Memory

USFs require access to a feature map $\varphi : S \mapsto \mathbb{R}^d$ that maps states into an embedding space in which the reward is assumed to be linear *i.e.* $R(s) = \varphi(s)^\top z$ with *weights* $z \in \mathbb{R}^d$ representing a task (Barreto et al., 2017; Borsa et al., 2018). The USFs $\psi : S \times A \times \mathbb{R}^d \to \mathbb{R}^d$ are defined as the discounted sum of future features subject to a task-conditioned policy π_z, and are trained such that

$$\psi(s_0, a_0, z) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t \varphi(s_{t+1}) | s_0, a_0, \pi_z\right] \qquad \forall\ s_0 \in S, a_0 \in A, z \in \mathbb{R}^d \tag{6}$$

$$\pi(s, z) \approx \arg\max_a \psi(s, a, z)^\top z, \qquad \forall\ s \in S, a \in A, z \in \mathbb{R}^d. \tag{7}$$

During training candidate task weights are sampled from $\mathcal{Z}$; during evaluation, the test task weights are found by regressing labelled states onto the features:

$$z_{\text{test}} \approx \arg\min_z \mathbb{E}_{s\sim\mathcal{D}_{\text{test}}}[(R_{\text{test}}(s) - \varphi(s)^\top z)^2], \tag{8}$$

before being passed to the policy. The features can be learned with Hilbert representations (Park et al., 2024b), laplacian eigenfunctions, or contrastive methods (Touati et al., 2023).

We define *memory-based* USFs as the discounted sum of future features extracted from the memory model's hidden state, subject to a memory-based policy $\pi_z(f_\pi(\tau^L)$:

$$\psi(\tau_0^L, z) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t \varphi(f_\psi(\tau_{t+1}^L)) \mid \tau_0^L, \pi_z\right] \qquad \forall\ \tau_0^L \in \mathcal{T}, z \in \mathbb{R}^d \tag{9}$$

$$\pi(f_\pi(\tau^L), z) \approx \arg\max_a \psi(f_\psi(\tau^L), z)^\top z \qquad \forall\ \tau^L \in \mathcal{T}, z \in \mathbb{R}^d, \tag{10}$$

where f_ψ and f_π are seperate memory models for ψ and π, and the previous hidden state h_{t-L-1} is dropped as an argument for brevity (*c.f.* Equation 4). At test time, task embeddings are found via Equation 8, but this time with reward-labelled trajectories rather than reward-labelled states:

$$z_{\text{test}} \approx \arg\min_z \mathbb{E}_{(\tau^L, R(s))\sim\mathcal{D}_{\text{labelled}}}[(R_{\text{test}}(s) - \varphi(f_\psi(\tau^L)^\top z)^2], \tag{11}$$

before being passed to the policy.

E Implementation Details

E.1 FB-M

Memory Models $f_F(\tau^L)$, $f_B(\tau^L)$ and $f_\pi(\tau^L)$ FB-M has separate memory models for the forward model f_F, backward model f_B and policy f_π following the findings of (Ni et al., 2021), but their implementations are identical. Trajectories of observation-action pairs are preprocessed by one-layer feedforward MLPs that embed their inputs into a 512-dimensional space. The memory model is a GRU whose hidden state is initialised as zeros and updated sequentially by processing each embedding in the trajectory. For the experiments in Section 6.1 we additionally use transformer Vaswani et al. (2017) and s4 memory models Gu et al. (2021). Our transformer uses *FlashAttention* (Dao et al., 2022) for faster inference, and we use diagonalised s4 (s4d) (Gu et al., 2022) rather than standard s4 because of its improved empirical performance on sequence modelling tasks.

Forward Model $F(f_F(\tau^L), z)$ The forward model takes the final hidden state from f_F and concatenates it with a preprocessed embedding of the most recent observation-task pair (o, z), following

the standard FB convention Touati & Ollivier (2021). This vector is passed through a final feedforward MLP F which outputs a d-dimensional embedding vector.

Backward Model $B(f_B(\tau^L))$ The backward model takes the final hidden state from f_B passed it through a two-layer feedforward MLP that outputs a d-dimensional embedding vector.

Actor $\pi(f_\pi(\tau^L), z)$ The actor takes the final hidden state from f_π and concatenates it with a preprocessed embedding of the most recent observation-task pair (o, z), following the standard FB convention Touati & Ollivier (2021) This vector is passed through a final feedforward MLP which outputs an a-dimensional vector, where a is the action-space dimensionality. A `Tanh` activation is used on the last layer to normalise their scale. As per (Fujimoto et al., 2019)'s recommendations, the policy is smoothed by adding Gaussian noise σ to the actions during training.

E.2 FB and HILP

FB and HILP follow the implementations by (Park et al., 2024b) which follow (Touati et al., 2023), other than the batch size which we reduce from 1024 to 512 to reduce the computational expense of each run without limiting performance as per (Jeen et al., 2024). Hyperparameters are reported in Table 2. An illustration of a standard FP architecture is provided in Figure 8, for comparison with the FP with memory architecture in Figure 1.

Forward Model $F(o_t, a_t, z)$ **/ USF** $\psi(o_t, a_t, z)$ Observation-action pairs (o, a) and observation-task pairs (o, z) are preprocessed and concatenated before being passed through a final feedforward MLP F / ψ which outputs a d-dimensional embedding vector.

Backward Model $B(o_t)$ **/ Feature Embedder** $\varphi(o_t)$ Observations are preprocessed then passed to the backward model B / feature embedder φ which is a two-layer feedforward MLP that outputs a d-dimensional embedding vector.

Actor $\pi(o_t, z)$ Observations (o_t) and observation-task pairs (o_t, z) are preprocessed by one-layer and concatenated before being passed through a final feedforward MLP which outputs a a-dimensional vector, where a is the action-space dimensionality. A `Tanh` activation is used on the last layer to normalise their scale. As per (Fujimoto et al., 2019)'s recommendations, the policy is smoothed by adding Gaussian noise σ to the actions during training.

Misc. Layer normalisation (Ba et al., 2016) and `Tanh` activations are used in the first layer of all MLPs to standardise the inputs.

E.2.1 z Sampling

BFMs require a method for sampling the task vector z at each learning step. (Touati et al., 2023) employ a mix of two methods, which we replicate:

1. Uniform sampling of z on the hypersphere surface of radius $\sqrt{d}$ around the origin of $\mathbb{R}^d$,
2. Biased sampling of z by passing states $s \sim \mathcal{D}$ through the backward model $z = B(s)$. This also yields vectors on the hypersphere surface due to the $L2$ normalisation described above, but the distribution is non-uniform.

We sample z 50:50 from these methods at each learning step.

E.3 Code References

This work was enabled by: Python (Sanner et al., 1999), NumPy (Harris et al., 2020), PyTorch (Paszke et al., 2017), Pandas (McKinney et al., 2011) and Matplotlib (Hunter, 2007).

Table 2: **Hyperparameters for all BFMs.**

Hyperparameter	Value
Latent dimension d	50
F / ψ dimensions	(1024, 1024)
B dimensions	(512, 512)
Preprocessor dimensions	(512, 512)
Transformer heads	4
Transformer / S4d model dimension	32
GRU dimensions	(512, 512)
Context length L	32 (Sections 5.2 and 5.3), 64 (Section **??**)
Frame stacking (FB & HILP)	4
Std. deviation for policy smoothing σ	0.2
Truncation level for policy smoothing	0.3
Learning steps	1,000,000 (ExORL), 500,000 (POPGym)
Batch size	512
Optimiser	Adam (Kingma & Ba, 2014)
Learning rate	0.0001
Discount γ	0.98
Activations (unless otherwise stated)	ReLU
Target network Polyak smoothing coefficient	0.01
z-inference labels	10,000
z mixing ratio	0.5
HILP representation discount factor	0.98
HILP representation expectile	0.5
HILP representation target smoothing coefficient	0.005

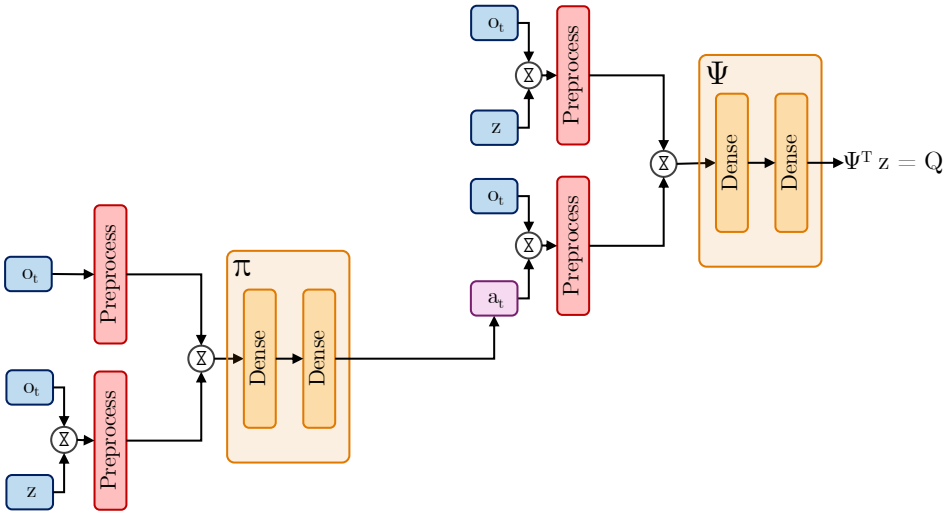

Figure 8: **FB *without* memory**. FPs are optimised in a standard actor critic setup (Konda & Tsitsiklis, 1999). The policy π selects an action a_t conditioned on a the current observation o_t, and the task vector z. The Q function formed by the USF ψ evaluates the action a_t given the current observation o_t and task z.

F Extended Results

We report a full breakdown of our results summarised in Sections 5.2 and 5.3. Table 3 reports results on our partially observed states experiments and Table 4 reports results on our changed dynamics experiments.

Table 3: **Full results on partially observed states (5 seeds).** For each dataset-domain pair, we report the score at the step for which the all-task IQM is maximised when averaging across 5 seeds ± the standard deviation.

			FB	**HILP**	**FB-stack**	**FB-M (ours)**	**MDP**
Cheetah	`flickering`	All tasks	121 ± 19	121 ± 19	121 ± 29	173 ± 51	434 ± 44
		Run	31 ± 13	31 ± 13	28 ± 18	75 ± 37	164 ± 34
		Run backward	8 ± 5	8 ± 5	35 ± 12	55 ± 15	174 ± 39
		Walk	184 ± 66	184 ± 66	206 ± 114	306 ± 175	663 ± 58
		Walk backward	61 ± 21	61 ± 21	198 ± 75	278 ± 70	743 ± 78
	`noisy`	All tasks	102 ± 59	102 ± 59	129 ± 29	150 ± 59	434 ± 44
		Run	28 ± 30	28 ± 30	38 ± 13	34 ± 17	164 ± 34
		Run backward	24 ± 23	24 ± 23	45 ± 15	57 ± 28	174 ± 39
		Walk	228 ± 98	228 ± 98	207 ± 50	199 ± 88	663 ± 58
		Walk backward	133 ± 99	133 ± 99	238 ± 78	283 ± 142	743 ± 78
Quadruped	`flickering`	All tasks	140 ± 75	140 ± 75	383 ± 133	673 ± 19	723 ± 10
		Jump	163 ± 142	163 ± 142	419 ± 124	771 ± 29	756 ± 8
		Run	95 ± 70	95 ± 70	315 ± 127	478 ± 14	507 ± 12
		Stand	98 ± 115	98 ± 115	511 ± 209	950 ± 14	965 ± 8
		Walk	181 ± 81	181 ± 81	323 ± 106	487 ± 51	743 ± 12
	`noisy`	All tasks	117 ± 68	117 ± 68	580 ± 124	711 ± 21	723 ± 10
		Jump	175 ± 92	175 ± 92	574 ± 126	712 ± 20	756 ± 8
		Run	63 ± 151	63 ± 151	447 ± 93	512 ± 31	507 ± 12
		Stand	65 ± 130	65 ± 130	841 ± 198	899 ± 31	965 ± 8
		Walk	35 ± 65	35 ± 65	428 ± 107	721 ± 43	743 ± 12
Walker	`flickering`	All tasks	82 ± 10	82 ± 10	577 ± 41	511 ± 85	628 ± 10
		Flip	57 ± 21	57 ± 21	500 ± 67	400 ± 79	602 ± 12
		Run	34 ± 6	34 ± 6	278 ± 24	237 ± 34	379 ± 7
		Stand	204 ± 35	204 ± 35	792 ± 136	761 ± 77	864 ± 17
		Walk	52 ± 9	52 ± 9	771 ± 58	646 ± 204	747 ± 18
	`noisy`	All tasks	309 ± 78	309 ± 78	475 ± 76	434 ± 23	628 ± 10
		Flip	165 ± 72	165 ± 72	378 ± 77	361 ± 45	602 ± 12
		Run	143 ± 56	143 ± 56	184 ± 49	183 ± 17	379 ± 7
		Stand	509 ± 137	509 ± 137	675 ± 81	731 ± 85	864 ± 17
		Walk	387 ± 96	387 ± 96	642 ± 125	486 ± 42	747 ± 18

Table 4: **Full results on ExORL changed dynamics experiments (5 seeds).** For each dataset-domain pair, we report the score at the step for which the all-task IQM is maximised when averaging across 5 seeds ± the standard deviation.

			FB	HILP	FB-stack	FB-M (ours)	MDP
1x	Cheetah	All tasks	476 ± 77	67 ± 37	156 ± 55	453 ± 120	434 ± 44
		Run	167 ± 59	17 ± 11	59 ± 18	150 ± 68	164 ± 34
		Run backward	166 ± 21	6 ± 21	36 ± 38	192 ± 66	174 ± 39
		Walk	816 ± 280	84 ± 43	312 ± 52	483 ± 242	663 ± 58
		Walk backward	777 ± 71	160 ± 83	186 ± 226	956 ± 167	743 ± 78
	Quadruped	All tasks	551 ± 82	186 ± 55	394 ± 76	566 ± 24	723 ± 10
		Jump	566 ± 128	291 ± 188	412 ± 69	787 ± 22	756 ± 8
		Run	360 ± 120	51 ± 27	251 ± 54	496 ± 17	507 ± 12
		Stand	842 ± 79	171 ± 186	521 ± 82	964 ± 9	965 ± 8
		Walk	434 ± 12	81 ± 68	358 ± 111	803 ± 84	743 ± 12
	Walker	All tasks	640 ± 4	391 ± 107	603 ± 8	635 ± 19	640 ± 10
		Flip	452 ± 20	340 ± 89	459 ± 15	452 ± 44	602 ± 12
		Run	387 ± 22	161 ± 47	236 ± 23	298 ± 16	379 ± 7
		Stand	876 ± 22	752 ± 290	856 ± 4	890 ± 30	864 ± 17
		Walk	845 ± 35	316 ± 139	853 ± 28	886 ± 40	747 ± 18
2x	Cheetah	All tasks	369 ± 140	62 ± 33	128 ± 83	586 ± 144	602 ± 10
		Run	146 ± 92	16 ± 12	18 ± 23	223 ± 73	333 ± 15
		Run backward	225 ± 83	1 ± 0	70 ± 75	320 ± 128	196 ± 10
		Walk	366 ± 400	86 ± 90	59 ± 45	814 ± 121	844 ± 21
		Walk backward	743 ± 230	144 ± 50	312 ± 275	976 ± 292	805 ± 18
	Quadruped	All tasks	333 ± 61	120 ± 47	263 ± 47	704 ± 31	731 ± 11
		Jump	309 ± 46	131 ± 81	272 ± 43	714 ± 79	749 ± 8
		Run	212 ± 42	42 ± 41	170 ± 33	474 ± 7	467 ± 20
		Stand	510 ± 121	156 ± 191	334 ± 39	957 ± 22	931 ± 10
		Walk	268 ± 60	62 ± 52	274 ± 78	723 ± 136	537 ± 15
	Walker	All tasks	278 ± 44	146 ± 74	463 ± 15	478 ± 19	500 ± 12
		Flip	100 ± 15	107 ± 29	320 ± 10	336 ± 86	351 ± 14
		Run	200 ± 27	81 ± 31	283 ± 19	297 ± 42	453 ± 120
		Stand	346 ± 141	290 ± 190	624 ± 34	691 ± 64	340 ± 11
		Walk	465 ± 58	98 ± 74	632 ± 57	574 ± 77	601 ± 18