FOUNDATION POLICIES WITH MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

A generalist agent should perform well on novel tasks in unfamiliar environments. While Foundation Policies (FPs) enable generalization across new tasks, they lack mechanisms for handling novel dynamics. Conversely, agents equipped with memory models can adapt to new dynamics, but struggle with unseen tasks. In this work, we bridge this gap by integrating memory models into the FP architecture, allowing policies to condition on both task and environment dynamics. We evaluate FPs enhanced with attention, state-space, and RNN-based memory models on POPGym, a memory benchmark, and ExORL, an unsupervised RL benchmark. Our results show that GRUs achieve the best generalization to unseen tasks and dynamics for a given recurrent state size, approaching the performance of a supervised baseline that has access to task information during training and significantly outperforming memory-free FPs. Additionally, our approach improves FP performance on entirely new environments not encountered during training. Our anonymized code is available at https://anonymous.4open.science/r/zero-shot-96A1, and our datasets are open-sourced at REDACTED.

1 INTRODUCTION

Reinforcement Learning (RL) agents [92] exhibit superhuman decision-making skill when tasked with a *single* objective in a *single* environment [89, 67, 90, 91]. A new line of work focuses on producing *generalist agents* that replicate such results across *many* tasks and environments [83, 53, 105]. Foundation policies (FPs) [96, 97, 76, 45] are a promising approach for building generalist agents, providing a principled mechanism for generalising to *any* downstream task in an environment after an offline reward-free pre-training phase. However, as yet, FPs are not equipped to deal with a change in dynamics between pre-training and deployment.

A concurrent line of work on *in-context* RL attempts to build generalist agents by using *memory models* to condition policies on reward-labelled trajectories [14, 43, 56, 54, 62, 23] or to reach arbitrary goal states [31]. In principal, these models can perform dynamics generalisation by inferring changes between training and testing from the trajectory used to condition the policy. However, they lack the task generalisation ability of FPs for two reasons. They are either 1) trained with reward supervision and so cannot reliably generalise to new tasks with different reward functions, or are 2) trained without reward supervision to reach any goal-state in an environment and so cannot reliably generalise to new tasks not codified by a goal state.

Here, we reconcile these lines of work and propose *foundation policies with memory*, an architecture that, like in-context RL agents, infers the current dynamics context using powerful memory models and passes it to an FP for solving unseen tasks. We evaluate FPs with attention [98], state-space [32, 33], and RNN-based [24, 17] memory models across a range of experiments testing their ability to infer the dynamics context, and generalise to unseen tasks in unseen dynamics. We find that GRUs achieve the best generalisation to unseen tasks and dynamics for a given recurrent state size, approaching the performance of a supervised baseline that has access to task information during training and significantly outperforming memory-free FPs (Figure 1). Finally, we find that FPs with memory improve FP performance on entirely new environments not seen during training.

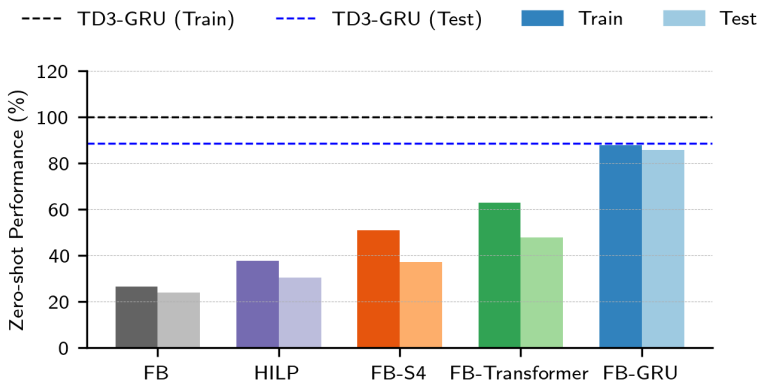

Figure 1: **Zero-shot task and dynamics generalisation.** FPs with memory models generalise to test tasks and dynamics not seen during training on the ExORL benchmark. FB-GRU approaches the performance of a supervised baseline, TD3-GRU, despite being trained without rewards. A full discussion is provided in Section 4.3.

2 PRELIMINARIES

Contextual markov decision processes. A Contextual Markov Decision Process (CMDP) is defined by $(C, S, O, \phi, A, R, \rho, \gamma, M(c))$. C is the set of contexts, S is the underlying state space, O and A are sets of observations and actions, $\phi : S \to O$ is the observation function, $R : S \to \mathbb{R}$ is a reward-function specifying a *task*, γ is a discount factor, and ρ is the initial state distribution [35]. M is a function that maps a context $c \in C$ to a Partially Observable Markov Decision Process (POMDP) [2] $M(c) = (S, O, A, R, \rho, \gamma, P^c)$ with a context-dependent transition function $P^c : S \times A \times C \to \Delta(S)$. A Markov policy $\pi : S \to \Delta(A)$ is optimal in context c for reward function R if it maximises the expected discounted future reward *i.e.* $\pi^*_{c,R} = \arg\max_\pi \mathbb{E}[\gamma^t R(s_{t+1}) | s_0, a_0, \pi, c]$, where $\mathbb{E}[\cdot | s_0, a_0, \pi, c]$ is the expectation under state-action sequence $(s_t, a_t)_{t \geq 0}$ starting at (s_0, a_0) with $s_t \sim P^c(\cdot | s_{t-1}, a_{t-1})$ and $a_t \sim \pi(\cdot | s_t)$. Note that the context $c \in C$ cannot be observed directly.

Problem setting. We split the CMDP into a set of training contexts C_{train} and testing contexts C_{test}. We assume access to a dataset $\mathcal{D}_{\text{train}}$ of *unlabelled* observation-action trajectories $\tau = (o_0, a_0, o_1, \ldots, o_T)$ collected from the training contexts by a highly exploratory behaviour policy. Our goal is to pre-train an adaptive policy $\pi(a | h, z)$, where $h \in \mathbb{R}^m$ is a hidden state summarising both the context c and inferred Markov state s, and $z \in \mathbb{R}^d$ denotes a compact representation of the task. We will pre-train this policy solely from offline data $\mathcal{D}_{\text{train}}$, without online interactions.

We will evaluate the policy on an unseen test task R_{test} in an unseen test context $c_{\text{test}} \in C_{\text{test}}$. The test task is revealed either via $\mathcal{D}_{\text{test}}$, a small dataset of *labelled* observation trajectories $((o_{t-L}, \ldots, o_t), R_{\text{test}}(s_t))$ of length L, or as an explicit function $o \mapsto R_{\text{test}}(s)$ (like 1 at a goal state and 0 elsewhere)[1]. Unless, the agent can infer c_{test} from $\mathcal{D}_{\text{test}}$, it will need to infer it from the observation-action history it observes during evaluation. This problem setting is directly equivalent to [97]'s zero-shot RL setting with a change in the environment dynamics between training and testing. As a result, we call it **zero-shot RL under changed dynamics**.

Foundation policies. Foundation policies (FPs) approximate the (universal) successor features [6, 11] of near-optimal policies for any task in an environment. They require access to a feature map $\varphi : S \mapsto \mathbb{R}^d$ that embeds states into a representation 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. The USFs $\psi : S \times A \times \mathbb{R}^d \to \mathbb{R}^d$

[1] Note that the agent only sees the observation, but the reward is a function of the underlying state.

are defined as the discounted sum of future features subject to a task-conditioned policy $\pi(s, z)$:

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

where the policy is trained in an actor-critic formulation [51] such that

$$\pi(s, z) \approx \arg\max_a \psi(s, a, z)^\top z,\ \forall\ s \in S, a \in A, z \in \mathbb{R}^d, \tag{2}$$

where $\psi(s, a, z)^\top z$ is the Q function (critic) formed by ψ. During training, candidate task weights are sampled from $\mathcal{Z}$, a prior over the task space[2]. During evaluation, the test task weights are found by regressing labelled states onto the features: $z_{\text{test}} := \arg\min_z \mathbb{E}_{s\sim d}[(R_{\text{test}}(s) - \varphi(s)^\top z)^2]$, before being passed to the policy. The features can be learned with Hilbert representations [76], laplacian eigenfunctions [97], contrastive methods [97], or in service of *successor measure* prediction [10], as is the case for the forward Backward (FB) foundation policy [96] used in this work.

3 METHOD

Recall that our goal is to pre-train an adaptive policy $\pi(a|h, z)$ that is conditioned on h, a hidden state summarising both the context c and inferred Markov state s, and task z. As we outlined in Section 2, the FP framework provides a principled way of pre-training $\pi(a|s, z)$ *i.e.* a policy conditioned solely on the task and Markov state. In this section, we will discuss amendments to the FP framework that allow policies to be conditioned on h rather than s.

3.1 MEMORY MODELS

Following past work on RL in CMDPs, we assume that we can produce an estimate of the dynamics context c and Markov state s from a *trajectory* of observation-action pairs $\tau = x_0, \ldots, x_L$, where $x_n = \epsilon(o_n, a_n)$ is some encoding of an observation-action pair and L is the *context length* [31, 65]. We seek a model of the form

$$y_j, h_j = f(x_j, h_{j-1}),\ \ j \in [1, \ldots, L], \tag{3}$$

where x_j, y_j are the inputs and outputs at time j, and f updates a hidden state $h \in \mathbb{R}^m$ summarising the current Markov state and dynamics context prediction. This is the standard setup of a *memory* model in RL [4, 68, 69, 70, 82, 66, 73, 40, 86, 100, 8, 109], because the asymptotic inference time complexity is $\mathcal{O}(1)$ which is helpful for fast data collection, or high-frequency motor control [62]. Until recently, only Recurrent Neural Networks (RNNs) [24, 41, 17] have had this property, but newly proposed structured state-space models (S4) [32, 33, 34] and fast Transformers [98, 19, 48] have runtime complexity approaching that of RNNs, and model histories with large L more accurately. We explore all of these memory models in Section 4.

3.2 FOUNDATION POLICIES WITH MEMORY

Equipped with memory model f, we now condition the FP's actor and critic on the hidden state it produces. We define *contextual* USFs as the discounted sum of future features extracted from the hidden state, subject to a policy conditioned on the inferred Markov state and dynamics context $\pi(h, z)$

$$\psi(h, z) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t \varphi(h_{t+1}) | h_0, \pi(h, z)\right] \quad h_0 = \mathbf{0}^m, \forall\ z\ \in \mathbb{R}^d, \tag{4}$$

where $h_t = f(x_t, h_{t-1})$ from Equation 3, x_t is zero-padded for all $t < L$, and $h_0 = \mathbf{0}^m$ is an initial hidden-state of zeroes. The policy is trained such that

$$\pi(h, z) \approx \arg\max_a \psi(h, z)^\top z,\ \ \forall\ h \in \mathbb{R}^m,\ z\ \in \mathbb{R}^d, \tag{5}$$

[2]See Appendix B.1.1 for more detail on $\mathcal{Z}$.

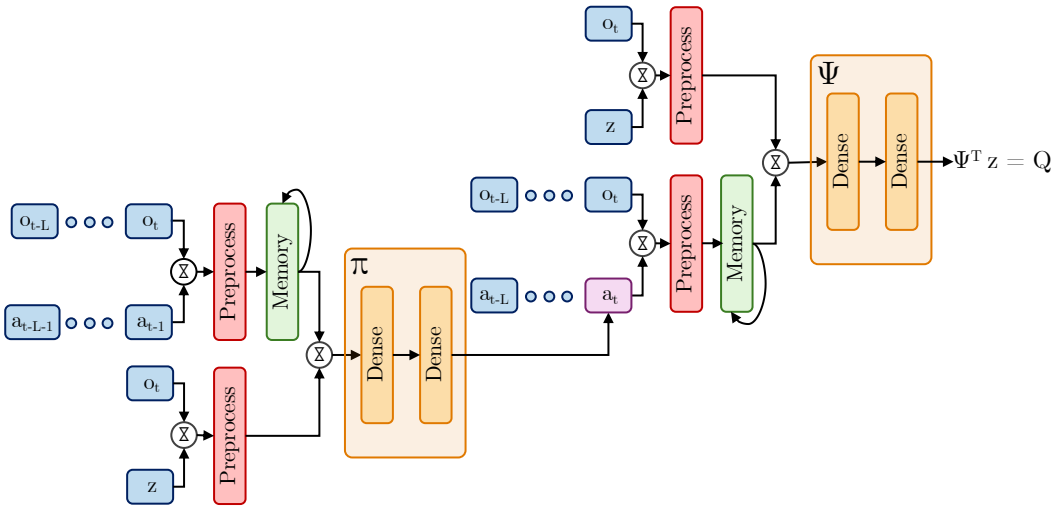

Figure 2: **Foundation policies with memory**. FPs are optimised in a standard actor critic setup [51]. The policy π selects an action a_t conditioned on a *history* of observations and actions $o_{t-L}, a_{t-L-1}, \ldots, o_t, a_{t-1}$ of length L encoded by the actor's memory model, and the task vector z. The Q function formed by the USF ψ evaluates the sequence of observations and actions $o_{t-L}, a_{t-L}, \ldots, o_t, a_t$ encoded by the critic's memory model for task z. The architecture of an FP *without* memory is illustrated in Figure 6 in Appendix B for comparison.

where $\psi(h, z)^\top z$ is the Q function (critic) formed by ψ. Training proceeds exactly as with conventional USFs, and the test-time task weights are found by regressing labelled states onto the hidden-state features: $z_{\text{test}} := \arg\min_z \mathbb{E}_{s_t,(o_{t-L:t},a_{t-L:t})\sim d}[(R_{\text{test}}(s_t) - \varphi(f((o_{t-L:t}, a_{t-L:t}), h_{0:L})^\top z)^2]$, before being passed to the policy. The full architecture and optimisation procedure is summarised in Figure 2. We found that using a shared memory model for the actor and critic led to model collapse, so use separate memory models for each. This corroborates the findings of [73]. Full implementation details are provided in Appendix B. In the experiments discussed in Section 4 we use FB representations as our FP which follow a slightly different training procedure. We discuss these details in Appendix B.

4 Experiments

In this section we perform an empirical study to evaluate our proposed method. We seek answers to three questions: (**Q1**) Can our method encode trajectories into a Markov state for use in solving one task in an environment? (**Q2**) Can our method generalise to unseen tasks in an environment with different dynamics to those seen in training? I.e. can our method perform *zero-shot RL under changed dynamics*? And (**Q3**) Can our method generalise to unseen tasks in a completely different environment to those seen in training? I.e. can our method perform *zero-shot environment generalisation*?

4.1 Setup

Environments. We respond to **Q1** using the POPGym benchmark [68], a set of tests that evaluate an agent's ability to infer Markov states from trajectories of observations and actions. We only evaluate on the "Hard" versions of CartPole, Pendulum, Noisy CartPole, Noisy Pendulum and Repeat Previous environments following [62]. For these experiments, we allow the agent to recondition its policy on the previous L observation-action pairs every step so we can disentangle the memory model's ability to accurately model the Markov state from its ability to carry forward an accurate hidden state. For all other experiments we do not allow such re-conditioning and require the policy to condition on only the previous hidden state, current observation-action pair, and task. Note for

these experiments $C_{\text{train}} = C_{\text{test}}$, so we are not yet testing whether our method can generalise across contexts.

We respond to **Q2** using the ExORL benchmark [108], a set of tests that evaluate an agent's ability to generalise to unseen tasks on the DeepMind Control Suite [93]. We evaluate on the same environments as [97]: Walker, Maze, Cheetah and Quadruped, removing the velocities from each of the state spaces to ensure the observations are not Markov, and call these variants *occluded*. To evaluate dynamics generalisation we train on datasets collected from environment instances where the robot's mass and damping coefficient are scaled to $\{0.5x, 1.5x\}$ of their usual values. We then evaluate on environment instances where the robot's mass and damping coefficient are scaled by $\{1.0x, 2.0x\}$ of their usual values, where $1.0x$ requires the agent to generalise via interpolation, and $2.0x$ requires the agent to generalise via extrapolation [74]. We evaluate on all tasks provided by the DeepMind control suite, and increase the number of goals in Maze from 4 to 20 for a total of 32 tasks across 4 environments.

We also respond to **Q3** using the ExORL benchmark [108]. This time we train on Walker-Occluded and Quadruped-Occluded and test on Cheetah-Occluded. The dynamics are unscaled (i.e. $1.0x$) and, as before, we evaluate on all tasks provided by the DeepMind control suite.

Aggregation across tasks or environments is always summarised by the Interquartile Mean (IQM) and standard deviation following the recommendations of [1]. On POPGym we report the mean-max epoch reward (MMER) metric used in the original paper. On ExORL, we report scores from the learning step for which the all-task IQM is maximised across seeds. Full experimental details are provided Appendix A, and a full description of our evaluation protocol is provided in Appendix A.3.

Baselines. We use FB [96] and HILP [76] as our FP baselines. FB is the most performant FP utilising successor measures, and HILP is the most performant FP utilising successor features. Both methods assume access to the Markov state for training as discussed in Section 2. So, instead of conditioning their predictions on a single observation, we provide them a stack of the 4 most recent observations i.e. $s_t = (o_{t-3}, o_{t-2}, o_{t-1}, o_t)$, also known as *frame-stacking* [67]. Frame-stacking is a naive method for inferring a Markov state from a short trajectory of observations, and is the first solution one would use when faced with our problem. We also baseline against a single-task, memory-based, supervised RL method. For this we use Offline TD3 [28] with a GRU memory model [17], which we refer to as **TD3-GRU**. Offline TD3 is the most performant single-task method on the ExORL benchmark [108]; TD3-GRU is the most performant method in [73], and TD3 with an LSTM memory model was shown to be particularly performant in [66]. TD3-GRU should indicate how well an agent optimising for one task performs if provided reward supervision.

Datasets. Though FPs can be deployed online they require an exploration policy for data collection. To disentangle test-time performance from an agent's data collection ability, we collect datasets on their behalf in advance using RND [11], an unsupervised RL algorithm. Agents trained on datasets collected with RND exhibit better performance than comparable methods like APS [60], APT [61], Proto [107] and DIAYN [25] in [108, 97, 76]. RND is run for 5 million learning steps in each of our environments and every transition is cached. For the supervised baseline TD3-GRU, transitions are relabelled with the appropriate rewards for a given task following [108]. All other methods are trained on these datasets reward-free.

Memory models. We test the performance of FPs equipped with three memory models. We use the most performant versions from each of the categories discussed in Section 3: attention-based, state-space based, and RNN-based. For our attention-based memory model we use a Transformer [98] with *FlashAttention* [19] for faster inference than a conventional Transformer. For our state-space-based memory model we use Diagonalized S4 [33], which uses a diagonal update matrix to perform faster training and inference than the popular S4 model [32]. For our RNN-based memory model we use a GRU [17] as it is the most performant RNN on the POPGym benchmark. Hereafter we refer to the FB models we augment with these as **FB-TF**, **FB-S4** and **FB-GRU** respectively. To ensure a fair comparison across memory models, we follow [68] and restrict each model to a fixed hidden state size, rather than a fixed parameter count. Concretely, we allow each model a hidden state size of $32^2 = 1024$ dimensions. In Section 4.3 we condition the models on trajectories of length 32, so a hidden state size of 32^2 allows the attention-based, and state-space models to perform their tensor products across the full input trajectory, and gives the RNN two 512-dimension

layers in which to summarise the trajectory. Full implementation details are provided in Appendix B.

4.2 POPGYM

We report the aggregate performance of all FB-based algorithms on the POPGym environments in Figure 3. Our supervised baseline, TD3-GRU, performs similarly to the PPO-GRU approach in the original POPGym paper. FB with frame-stacking performs poorly, reaching only 30% of TD3-GRU's aggregate score. Our three memory-based methods perform comparitively better, with FB-TF reaching 80% of TD3-GRU's performance, and FB-S4 and FB-GRU matching TD3-GRU's performance. We find that all methods fail on the RepeatPreviousHard environment, where other in-context RL agents have shown strong performance [62, 31]. This task requires the agent to remember the suit of a card dealt 64 timesteps ago (Appendix A), and our models are trained with context length $L = 64$. This suggests that the memory models are not accurately recalling information from the start of their context. The implications of our choice of length L are discussed in Section 5.

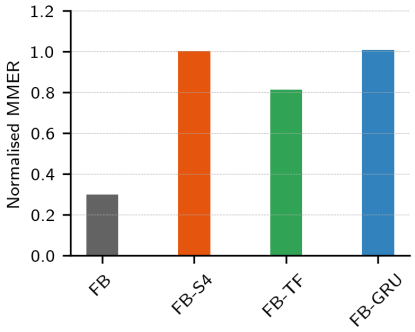

Figure 3: **POPGym results.** Aggregate mean-maximum epoch reward (MMER) across all POPGym environments, normalised w.r.t. TD3-GRU performance.

4.3 ZERO-SHOT RL UNDER CHANGED DYNAMICS

We report the aggregate performance of all algorithms on our *zero-shot RL under changed dynamics* experiments in Figure 4 *(left)*, and the ratios of interpolation/extrapolation performance to train performance in Figure 4 *(right)*. As with our POPGym experiments, FB performs poorly, reaching $\sim 25\%$ of the performance of our supervised baseline on the training environments. HILP performs slightly better, as we would expect given its results on ExORL in [76], but still much poorer than TD3-GRU. Of the three FPs with memory, FB-GRU performs best on train, interpolation and extrapolation evaluations, with results relative to the supervised baseline similar to FB trained on Markov states in [97]. Aggregate test performance approximately matches TD3-GRU aggregate test performance despite not seeing rewards during training. FB-TF exhibits the best interpolation-to-train ratio, and FB-GRU the best extrapolation-to-train ratio.

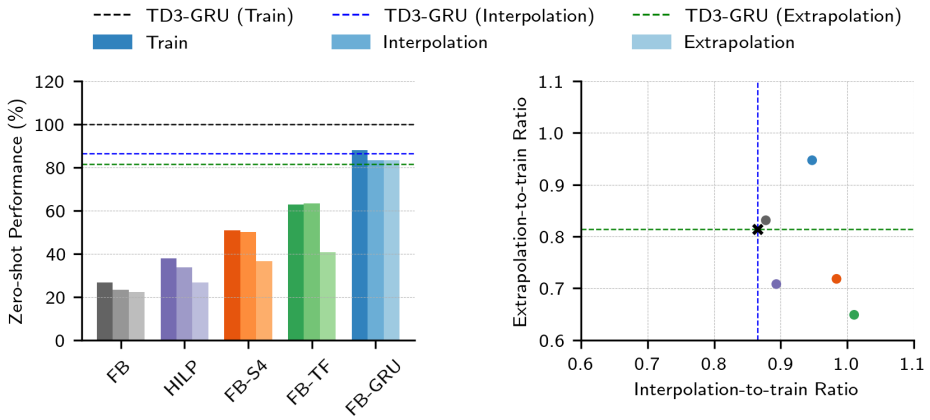

Figure 4: **Zero-shot dynamics generalisation on ExORL.** *(Left)* Aggregate performance across all ExORL tasks and domains normalised w.r.t. TD3-GRU performance, averaged over 5 seeds. We train on dynamics $\{0.5x, 1.5x\}$ their typical values and evaluate on $1.0x$ (interpolation) and $2.0x$ (extrapolation). *(Right)* The ratios of interpolation and extrapolation performance to train performance.

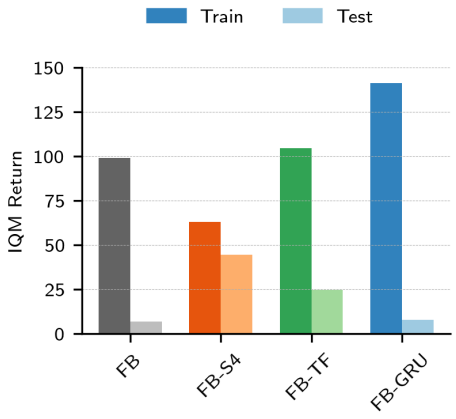

Figure 5: **Zero-shot environment generalisation on ExORL.** Aggregate performance across all tasks in the train environments (Walker, Quadruped), and the test environment (Cheetah), averaged across 5 seeds.

4.4 Zero-shot Environment Generalisation

We report the aggregate performance of all FB-based algorithms on our *zero-shot environment generalisation experiments* in Figure 5. Here, FB-GRU performs best on the training environments, but poorest on the testing environment, with FB performing similarly poorly on the testing environment. FB-S4 improves performance on the testing environment by $\sim$ 4x over FB, but at the cost of reducing training environment performance by half. FB-TF improves both training performance by 5% over FB and triples test performance. We emphasise that, although FPs with memory do improve zero-shot environment generalisation performance in some cases, the absolute returns remain low (a max of 33 for FB-S4 out of a possible 1000) suggesting their is significant room for improvement.

5 Discussion and Limitations

Context length. In Section 4 we train agents with a context length $L = 64$ timesteps, which is the maximum context length we could afford with our computational budget[3]. We see two limitations with this. First, we have assumed the dynamics context and task for all of our experiments can be inferred from this context, but it is not clear that this is the case. Indeed, TD3-GRU with reward supervision and a maximally exploratory dataset does not match its performance with Markov states from [97]. Second, successful episodes run for a minimum of 200 timesteps (as in PendulumHard) and a maximum of 1000 timesteps (as in ExORL), meaning we never train the memory model over full episodes, nor can we reliably maintain an episode's full trajectory in context at test-time. This introduces situations where the hidden state will be erroneously initialised mid-episode during training, creating well-established theoretical issues for memory models in RL [68], though these are yet to prove critical empirically [73, 66].

The obvious solution to these problems, were it available to us, would be to increase L until it is the maximum episode length, and train for longer as in [31, 62, 68]. However, even if we were to do this, any real-world deployment may induce episodes longer than this assumed max length, or indeed we may wish to operate in the non-episodic, continual setting. The existing literature implicitly assumes that if L is very large such issues will resolve themselves, but this is not clear to us. Exploring how to deal with such situations is an important future research direction.

Datasets. As outlined in Section 4.1, we train all methods on datasets pre-collected with RND [11] 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

[3]Our shared resource limits us to a maximum run length of 24 hours per GPU, and the ExORL runs took approximately 20 hours on one A100. See Appendix A.4 for more detail.

homogeneous. It is not clear how our specific proposals will interact with such datasets. If, for example, the dataset only exhibits parts of the state space from which the dynamics cannot be well-inferred, like a robot stuck stationary, 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 [52, 59]. That said, the proposals of [45] for conducting zero-shot RL from real-world datasets could be integrated into our proposals easily, and may help.

6 Related Work

Generalist policy pre-training FPs build upon successor representations [20], universal value function approximators [85], successor features [6] and successor measures [10]. The state-of-the-art methods instantiate these ideas as either universal successor features (USFs) [11] or forward-backward (FB) representations [96, 97], with recent work showing they can be trained on low quality datasets [45], or used to perform a range of imitation learning techniques efficiently [80]. A representation learning method is required to learn the features for USFs, with past works using inverse curiosity modules [79], diversity methods [60, 38], Laplacian eigenfunctions [101], or contrastive learning [16]. No works have yet explored the generalisation capacity of FPs to unseen dynamics. Two concurrent lines of work on *goal-conditioned* RL and *in-context* RL also seek to build generalist policies. Goal-conditioned RL methods train policies to reach any goal state from any other goal state [75, 63, 104, 26, 99], but lack the ability to generalise to tasks with "dense" reward functions, like those on the locomotion tasks in ExORL. In-context RL methods train policies using sequence models conditioned on reward-labelled histories [14, 43, 58, 83, 110, 13, 30, 88, 103, 102, 31, 62, 94, 23], but, unlike FPs, do not have a robust mechanism for training without access to rewards.

Dynamics Generalisation Dynamics generalisation in RL is a well-established problem [50, 65, 74]. Common remedies include: data augmentation [81, 5, 106, 37, 36, 55], domain randomisation [95, 21, 46, 47, 77], learning context-aware policies [87, 57, 9, 44], and meta-learning [15, 83, 27, 71, 72]. Our work is most similar to those that tackle dynamics generalisation by conditioning policies on dynamics inferred with a memory model [73, 18]. Where these methods are concerned with generalising to one unseen task in unseen dynamics contexts, our method can generalise to more than one unseen tasks in unseen dynamics contexts.

7 Conclusion

In this paper, we explored augmenting Foundation Policies (FPs) with memory models to allow them to condition policies on a dynamics context inferred from a history of observations and actions. We evaluated our proposals with attention, state-space, and RNN-based memory models on POPGym, a memory benchmark, and ExORL, an unsupervised RL benchmark. Our results show that GRUs achieve the best generalisation to unseen tasks and dynamics for a given recurrent state size, approaching the performance of a supervised baseline that has access to task information during training and significantly outperforming memory-free FPs. We believe our proposals represent a further step toward the development of generalist, adaptive agents.

REFERENCES

[1] 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.

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

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

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

[5] 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.

[6] 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.

[7] Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

[8] Jacob Beck, Kamil Ciosek, Sam Devlin, Sebastian Tschiatschek, Cheng Zhang, and Katja Hofmann. Amrl: Aggregated memory for reinforcement learning. In *International Conference on Learning Representations*, 2020.

[9] Michael Beukman, Devon Jarvis, Richard Klein, Steven James, and Benjamin Rosman. Dynamics generalisation in reinforcement learning via adaptive context-aware policies. *Advances in Neural Information Processing Systems*, 36, 2024.

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

[11] 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.

[12] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[13] Yevgen Chebotar, Quan Vuong, Alex Irpan, Karol Hausman, Fei Xia, Yao Lu, Aviral Kumar, Tianhe Yu, Alexander Herzog, Karl Pertsch, et al. Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. *arXiv preprint arXiv:2309.10150*, 2023.

[14] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[15] Tao Chen, Adithyavairavan Murali, and Abhinav Gupta. Hardware conditioned policies for multi-robot transfer learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[16] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

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

[18] 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.

[19] 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.

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

[21] Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33:13049–13061, 2020.

[22] Kenji Doya. Temporal difference learning in continuous time and space. *Advances in neural information processing systems*, 8, 1995.

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

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

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

[26] 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.

[27] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

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

[29] 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.

[30] Hiroki Furuta, Yutaka Matsuo, and Shixiang Shane Gu. Generalized decision transformer for offline hindsight information matching. *arXiv preprint arXiv:2111.10364*, 2021.

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

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

[33] 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.

[34] Ankit Gupta, Albert Gu, and Jonathan Berant. Diagonal state spaces are as effective as structured state spaces. *Advances in Neural Information Processing Systems*, 35:22982–22994, 2022.

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

[36] Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13611–13617. IEEE, 2021.

[37] Nicklas Hansen, Hao Su, and Xiaolong Wang. Stabilizing deep q-learning with convnets and vision transformers under data augmentation. *Advances in neural information processing systems*, 34:3680–3693, 2021.

[38] Steven Hansen, Will Dabney, Andre Barreto, Tom Van de Wiele, David Warde-Farley, and Volodymyr Mnih. Fast task inference with variational intrinsic successor features. *arXiv preprint arXiv:1906.05030*, 2019.

[39] 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.

[40] Natalia Hernandez-Gardiol and Sridhar Mahadevan. Hierarchical memory-based reinforcement learning. *Advances in Neural Information Processing Systems*, 13, 2000.

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

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

[43] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

[44] Scott Jeen and Jonathan M. Cullen. Dynamics generalisation with behaviour foundation models. *RL Conference Workshop on Training Agents with Foundation Models*, 2024.

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

[46] Minqi Jiang, Michael Dennis, Jack Parker-Holder, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Replay-guided adversarial environment design. *Advances in Neural Information Processing Systems*, 34:1884–1897, 2021.

[47] Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pp. 4940–4950. PMLR, 2021.

[48] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

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

[50] Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76: 201–264, 2023.

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

[52] 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., 2019.

[53] Aviral Kumar, Rishabh Agarwal, Xinyang Geng, George Tucker, and Sergey Levine. Offline q-learning on diverse multi-task data both scales and generalizes. *arXiv preprint arXiv:2211.15144*, 2022.

[54] Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, et al. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*, 2022.

[55] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33:19884–19895, 2020.

[56] Jonathan Lee, Annie Xie, Aldo Pacchiano, Yash Chandak, Chelsea Finn, Ofir Nachum, and Emma Brunskill. Supervised pretraining can learn in-context reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2023.

[57] 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.

[58] Kuang-Huei Lee, Ofir Nachum, Mengjiao Yang, Lisa Lee, Daniel Freeman, Winnie Xu, Sergio Guadarrama, Ian Fischer, Eric Jang, Henryk Michalewski, et al. Multi-game decision transformers. *Advances in neural information processing systems*, 35, 2022.

[59] 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.

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

[61] Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. *Advances in Neural Information Processing Systems*, 34:18459–18473, 2021.

[62] 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.

[63] 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.

[64] 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.

[65] Ishita Mediratta, Qingfei You, Minqi Jiang, and Roberta Raileanu. A study of generalization in offline reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

[66] 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.

[67] 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.

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

[69] Steven Morad, Ryan Kortvelesy, Stephan Liwicki, and Amanda Prorok. Reinforcement learning with fast and forgetful memory. *Advances in Neural Information Processing Systems*, 36, 2024.

[70] Steven Morad, Chris Lu, Ryan Kortvelesy, Stephan Liwicki, Jakob Foerster, and Amanda Prorok. Revisiting recurrent reinforcement learning with memory monoids. *arXiv preprint arXiv:2402.09900*, 2024.

[71] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

[72] Anusha Nagabandi, Chelsea Finn, and Sergey Levine. Deep online learning via meta-learning: Continual adaptation for model-based rl. *arXiv preprint arXiv:1812.07671*, 2018.

[73] 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.

[74] 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.

[75] 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, 2024.

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

[77] Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In *International Conference on Machine Learning*, pp. 17473–17498. PMLR, 2022.

[78] 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.

[79] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

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

[81] 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.

[82] Dhruv Ramani. A short survey on memory based reinforcement learning. *arXiv preprint arXiv:1904.06736*, 2019.

[83] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *Transactions of Machine Learning Research*, 2022.

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

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

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

[87] 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.

[88] Max Siebenborn, Boris Belousov, Junning Huang, and Jan Peters. How crucial is transformer in decision transformer? *arXiv preprint arXiv:2211.14655*, 2022.

[89] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529 (7587):484–489, 2016.

[90] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[91] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[92] Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[93] 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.

[94] Adaptive Agent Team, Jakob Bauer, Kate Baumli, Satinder Baveja, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, et al. Human-timescale adaptation in an open-ended task space. *arXiv preprint arXiv:2301.07608*, 2023.

[95] 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.

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

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

[98] 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.

[99] Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. In *International Conference on Machine Learning*, pp. 36411–36430. PMLR, 2023.

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

[101] Yifan Wu, George Tucker, and Ofir Nachum. The laplacian in rl: Learning representations with efficient approximations. *arXiv preprint arXiv:1810.04586*, 2018.

[102] Mengdi Xu, Yikang Shen, Shun Zhang, Yuchen Lu, Ding Zhao, Joshua Tenenbaum, and Chuang Gan. Prompting decision transformer for few-shot policy generalization. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 24631–24645, 17–23 Jul 2022.

[103] Taku Yamagata, Ahmed Khalil, and Raul Santos-Rodriguez. Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline RL. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 38989–39007, 23–29 Jul 2023.

[104] Rui Yang, Lin Yong, Xiaoteng Ma, Hao Hu, Chongjie Zhang, and Tong Zhang. What is essential for unseen goal generalization of offline goal-conditioned rl? In *International Conference on Machine Learning*, pp. 39543–39571. PMLR, 2023.

[105] Sherry Yang, Ofir Nachum, Yilun Du, Jason Wei, Pieter Abbeel, and Dale Schuurmans. Foundation models for decision making: Problems, methods, and opportunities. *arXiv preprint arXiv:2303.04129*, 2023.

[106] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

[107] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pp. 11920–11931. PMLR, 2021.

[108] 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.

[109] 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.

[110] Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In *international conference on machine learning*, pp. 27042–27059. PMLR, 2022.

APPENDICES

A Experimental Details

A.1 POPGym

We consider 5 environments from the POPGym benchmark [68] which is built atop the OpenAI gym [12]. Each tests the agents ability to summarise a trajectory of observations and actions into a Markov state for use in solving one downstream task. Following [62] we only consider the *hard* variants, because the other variants are considered too straightforward.

Stateless CartPole Hard. The cartpole environment from [7], but with the angular and linear positions removed from the observation. The agent must integrate to compute positions from velocity and balance the pole atop the cart to receive reward. The *hard* variant requires the pole to be balanced for 600 timesteps (the *easy* and *medium* variants require the pole to be balanced for fewer timesteps).

Noisy Stateless CartPole Hard. The same as Stateless CartPole Hard but with Gaussian noise added to observations. The *hard* variant sets the standard deviation of the noise $\sigma = 0.3$ (the *easy* and *medium* set $\sigma = 0.1$ and $\sigma = 0.2$ respectively).

Stateless Pendulum Hard. The swing-up pendulum [22] with the angular position information removed. The agent must integrate to compute positions from velocity and swing the pendulum up to receive reward. The *hard* variant requires the pendulum to be balanced for 200 timesteps (the *easy* and *medium* variants require the pole to be balanced for fewer timesteps).

Noisy Stateless Pendulum Hard. The same as Stateless Pendulum Hard but with Gaussian noise added to observations. The *hard* variant sets the standard deviation of the noise $\sigma = 0.3$ (the *easy* and *medium* set $\sigma = 0.1$ and $\sigma = 0.2$ respectively).

Repeat Previous Hard. Observations contain one of four values. The agent is rewarded for outputting the observation from some constant k timesteps ago, i.e. observation o_{t-k} at time t. The *hard* variant sets $k = 64$ (the *easy* and *medium* variants set $k < 64$).

A.2 ExORL

We consider 4 environments (three locomotion and one goal-directed) from the ExORL benchmark [108] which is built atop the DeepMind Control Suite [93]. We occlude their states by removing all velocity components, similar to [73, 66]. Environments are visualised here: `https://www.youtube.com/watch?v=rAai4QzcYbs`. The domains are summarised in Table 1.

Walker-Occluded. A two-legged robot required to perform locomotion starting from bent-kneed position. The observation and action spaces are 17 and 6-dimensional respectively (after occlusion), 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-Occluded. A four-legged robot required to perform locomotion inside a 3D maze. The observation and action spaces are 67 and 12-dimensional respectively (after occlusion), 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.

Maze-Occluded. A 2D maze with four rooms where the task is to move a point-mass to one of the rooms. The observation and action spaces are both 2-dimensional (after occlusion); the observation space consists of x, y positions of the mass, the action space is the x, y tilt angle. ExORL provides four reaching tasks `top left, top right, bottom left` and `bottom right` corresponding to each room. We add four other goals in each room following [97] to provide a total of 20 goal reaching tasks. The mass is always initialised in the top left and the reward is proportional to the distance from the goal, though is sparse i.e. it only registers once the agent is reasonably close to the goal.

Cheetah-Occluded. A running two-legged robot. The observation and action spaces are 10 and 6-dimensional respectively (after occlusion), 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.

A.3 Evaluation Protocol

We evaluate the cumulative reward (hereafter called score) achieved by all methods across three seeds in POPGym and 5 seeds in ExORL. We report task scores as per the best practice recommendations of [1]. 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.

A.4 Computational Resources

We train our models on NVIDIA A100 GPUs. Training TD3-GRU to solve one task on one GPU takes approximately 6 hours for POPGym and 8 hours for ExORL. One run of FB-stack and SF-stack on one domain (for all tasks) takes approximately 3 hours for POPGym and 5 hours for ExORL on one GPU. One run of the memory-based FPs on one domain (for all tasks) on one GPU in approximately 20 hours. Note the POPGym experiments run for 500k learning steps, whereas the ExORL experiments run for 1m learning steps. As a result, our core experiments on the ExORL benchmark used approximately 65 GPU days of compute.

B Implementation Details

B.1 Foundation Policies

FB and HILP follow the implementations by [76] which follow [97], 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 [45]. Hyperparameters are reported in Table 2. An illustration of a standard FP architecture is provided in Figure 6, for comparison with the FP with memory architecture in Figure 2.

Forward Representation $F(o, a, z)$ **/ USF** $\psi(o, a, z)$. Inputs $(o_{t-L:t}, a)$ and state-task pairs (o, z) are preprocessed by feedforward MLPs that embed their inputs into a 512-dimensional space. These embeddings are concatenated and passed through a third feedforward MLP F / ψ which outputs a d-dimensional embedding vector. The Transformer memory model with Flash Attention follows the exact implementation in [31]; the S4d memory model follows the exact implementation in [68], and the GRU memory model follows the exact implementation provided by Torch.

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-Occluded	Low	Locomotion	Dense
Quadruped-Occluded	High	Locomotion	Dense
Maze-Occluded	Low	Goal-reaching	Sparse
Cheetah-Occluded	Low	Locomotion	Dense

Table 2: **FP Hyperparameters.**

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 (Section 4.3), 64 (Sections 4.2 and 4.4)
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 [49]
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

Backward Representation $B(o_{t-L:t})$ (for FB). Inputs are preprocessed by feedforward MLPs that embed their inputs into a 512-dimensional space then passed to the backward representation B which is a feedforward MLP that outputs a d-dimensional embedding vector.

Actor $\pi(o_{t-L:t}, z)$. Inputs $(o_{t-L:t}, a)$ and state-task pairs (o, z) are preprocessed by feedforward MLPs that embed their inputs into a 512-dimensional space. These embeddings are concatenated and passed through a third 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 [29]'s recommendations, the policy is smoothed by adding Gaussian noise σ to the actions during training.

Misc. Layer normalisation [3] and `Tanh` activations are used in the first layer of all MLPs to standardise the inputs.

B.1.1 z SAMPLING

FPs require a method for sampling the task vector z at each learning step. [97] 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 representation $z = B(s)$. 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.

B.2 TD3-GRU

We adopt the same implementation and hyperparameters as is used on the ExORL benchmark. Hyperparameters are reported in Table 3. The memory module follows the implementation from [73] and uses a seperate encoder for the actor and critic.

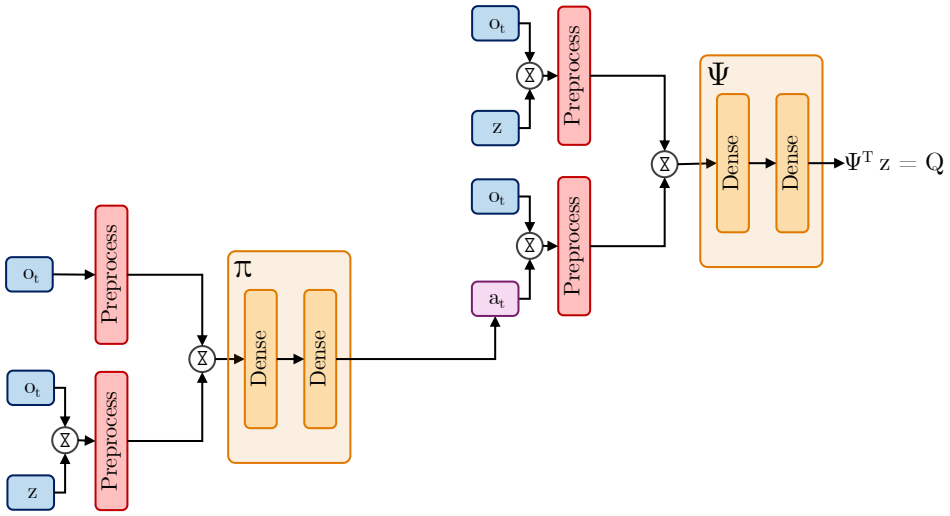

Figure 6: **Foundation policies *without* memory**. FPs are optimised in a standard actor critic setup [51]. The policy π selects an action a_t conditioned on a the current observation o_t, and the task vector z. The Q function formed by the USF ψ evaluates the action a_t given the current observation o_t and task z.

Critic(s). TD3 employs double Q networks, where the target network is updated with Polyak averaging via a momentum coefficient. The critics are feedforward MLPs that take a state-action pair (s, a) as input and output a value $\in \mathbb{R}^1$.

Actor. The actor is a standard feedforward MLP taking the state s as input and outputting an a-dimensional vector, where a is the action-space dimensionality. The policy is smoothed by adding Gaussian noise σ to the actions during training.

Misc. As is usual with TD3, layer normalisation [3] is applied to the inputs of all networks.

Table 3: **TD3-GRU hyperparameters.**

Hyperparameter	Value
Critic dimensions	(1024, 1024)
Actor dimensions	(1024, 1024)
GRU dimensions	(512, 512)
Preprocessor dimensions	(512, 512)
Learning steps	1,000,000 (ExORL), 500,000 (POPGym)
Batch size	512
Optimiser	Adam
Learning rate	0.0001
Discount γ	0.98
Activations	ReLU
Target network Polyak smoothing coefficient	0.01
Std. deviation for policy smoothing σ	0.2
Truncation level for policy smoothing	0.3

B.3 Code References

This work was enabled by: Python [84], NumPy [39], PyTorch [78], Pandas [64] and Matplotlib [42].

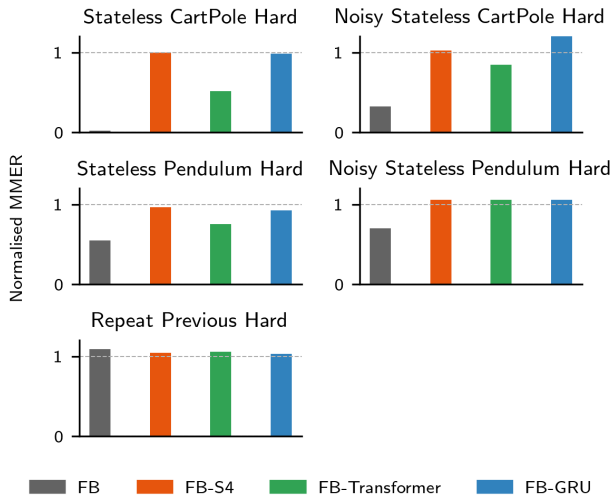

Figure 7: **Per-environment POPGym results.** The results are aggregated over 3 seeds, visualised by environment, and report the normalised MMER as with Table 4.

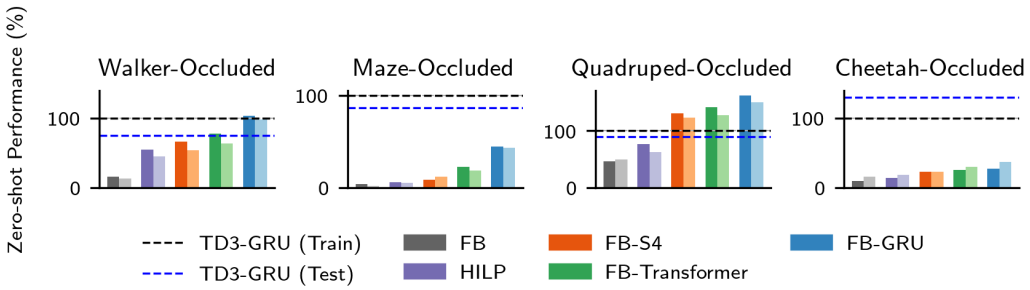

Figure 8: **Per-environment zero-shot dynamics generalisation results.** The results are aggregated over 5 seeds and all tasks in each environment, and show the normalised IQM w.r.t. TD3-GRU.

C Extended Results

We report a full breakdown of our results summarised in Sections 4.2, 4.3, 4.4. Table 4 reports results on POPGym from Section 4.2, Table 5 reports results on the zero-shot dynamics generalisation experiments from Section 4.3, and Table 6. Additionally, Figures 7 and 8 show plots where the results are aggregated by environment, and Figure 9 show plots where the zero-shot dynamics generalisation results are aggregated by task.

Table 4: **Full results on the POPGym environments (3 seeds).** We report the *unnormalised* mean-max epoch return (MMER) return $\pm$ the standard deviation averaged over 3 seeds.

Environment	TD3-gru	FB-stack	**FB-TF (ours)**	**FB-S4 (ours)**	**FB-GRU (ours)**
NoisyStatelessCartPoleHard	0.156 ± 0.011	0.05 ± 0.0	0.132 ± 0.012	0.16 ± 0.022	0.196 ± 0.021
NoisyStatelessPendulumHard	0.543 ± 0.004	0.381 ± 0.033	0.572 ± 0.005	0.572 ± 0.007	0.572 ± 0.009
RepeatPreviousHard	-0.418 ± 0.012	-0.455 ± 0.002	-0.441 ± 0.002	-0.436 ± 0.016	-0.431 ± 0.005
StatelessCartPoleHard	1.0 ± 0.0	0.017 ± 0.0	0.515 ± 0.259	1.0 ± 0.0	0.983 ± 0.025
StatelessPendulumHard	0.8 ± 0.032	0.436 ± 0.033	0.601 ± 0.015	0.77 ± 0.032	0.742 ± 0.01

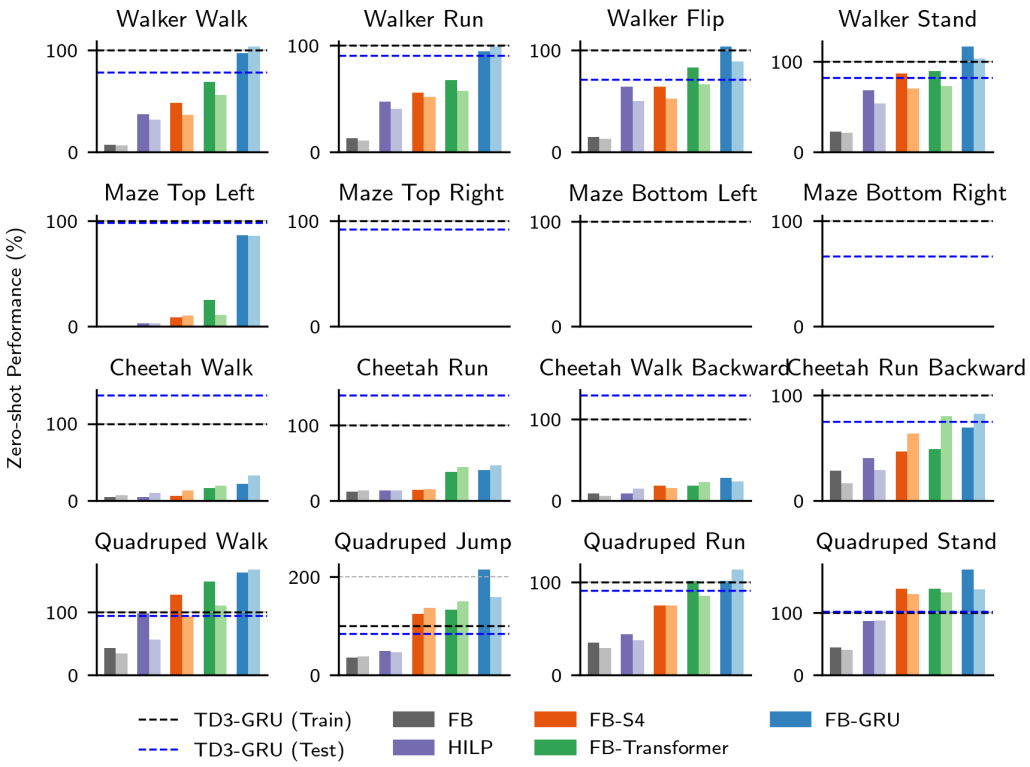

Figure 9: **Per-task zero-shot dynamics generalisation results.** The results are aggregated over 5 seeds, and show the normalised IQM w.r.t. TD3-GRU.

Table 5: **Full results on the ExORL dynamics generalisation 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.

Dynamics	Environment	Task	TD3-gru	HILP-stack	FB-stack	**FB-TF (ours)**	**FB-S4 (ours)**	**FB-GRU (ours)**
0.5x	Cheetah	All tasks	111 ± 98	21 ± 6	32 ± 7	25 ± 11	28 ± 3	43 ± 18
		Run	26 ± 6	7 ± 3	13 ± 2	5 ± 3	10 ± 2	2 ± 14
		Run Backward	16 ± 9	5 ± 9	5 ± 6	10 ± 8	5 ± 4	12 ± 16
		Walk	233 ± 76	26 ± 33	77 ± 13	16 ± 7	53 ± 13	14 ± 67
		Walk Backward	196 ± 122	36 ± 18	27 ± 18	64 ± 48	41 ± 18	89 ± 63
	Maze	Multi goal	413 ± 346	26 ± 20	80 ± 45	18 ± 25	20 ± 18	153 ± 40
	Quadruped	All tasks	279 ± 32	203 ± 31	123 ± 25	327 ± 21	541 ± 71	330 ± 179
		Jump	290 ± 115	104 ± 76	85 ± 25	315 ± 86	698 ± 60	311 ± 273
		Run	268 ± 88	104 ± 46	96 ± 73	210 ± 74	319 ± 100	178 ± 112
		Stand	322 ± 83	311 ± 90	151 ± 55	469 ± 140	824 ± 108	415 ± 337
		Walk	234 ± 113	270 ± 58	123 ± 69	289 ± 40	367 ± 110	362 ± 84
	Walker	All tasks	646 ± 224	446 ± 114	89 ± 13	451 ± 34	321 ± 24	533 ± 46
		Flip	570 ± 16	409 ± 182	74 ± 25	425 ± 46	330 ± 28	489 ± 123
		Run	249 ± 11	169 ± 26	33 ± 3	167 ± 10	117 ± 8	193 ± 10
		Stand	847 ± 30	827 ± 98	182 ± 29	778 ± 15	594 ± 49	934 ± 17
		Walk	723 ± 30	372 ± 196	43 ± 26	425 ± 74	243 ± 16	504 ± 66
1x	Cheetah	All tasks	146 ± 206	36 ± 36	72 ± 24	40 ± 11	47 ± 14	54 ± 38
		Run	37 ± 21	4 ± 11	25 ± 21	5 ± 3	16 ± 7	3 ± 24
		Run Backward	11 ± 6	0 ± 17	6 ± 8	16 ± 14	11 ± 6	20 ± 19
		Walk	524 ± 254	102 ± 116	163 ± 66	31 ± 30	92 ± 45	49 ± 152
		Walk Backward	255 ± 192	18 ± 16	59 ± 59	92 ± 42	53 ± 19	69 ± 69
	Maze	Multi goal	354 ± 354	11 ± 18	58 ± 32	35 ± 33	21 ± 17	154 ± 47
	Quadruped	All tasks	232 ± 77	179 ± 19	133 ± 29	360 ± 23	445 ± 56	403 ± 160
		Jump	218 ± 95	108 ± 34	135 ± 69	359 ± 58	557 ± 101	409 ± 210
		Run	246 ± 87	76 ± 82	106 ± 77	277 ± 44	296 ± 93	203 ± 83
		Stand	390 ± 130	313 ± 48	135 ± 28	565 ± 140	677 ± 125	532 ± 299
		Walk	192 ± 103	168 ± 49	88 ± 59	248 ± 47	286 ± 54	362 ± 135
	Walker	All tasks	519 ± 192	385 ± 109	74 ± 5	459 ± 42	301 ± 45	565 ± 54
		Flip	432 ± 55	315 ± 155	64 ± 7	409 ± 36	299 ± 51	480 ± 49
		Run	267 ± 29	168 ± 60	26 ± 2	181 ± 21	113 ± 16	218 ± 24
		Stand	781 ± 80	671 ± 113	168 ± 20	732 ± 41	554 ± 66	871 ± 38
		Walk	606 ± 46	348 ± 186	47 ± 9	475 ± 96	234 ± 57	654 ± 127
1.5x	Cheetah	All tasks	240 ± 286	13 ± 42	56 ± 25	23 ± 4	52 ± 17	52 ± 55
		Run	52 ± 29	2 ± 41	17 ± 22	5 ± 3	18 ± 11	7 ± 41
		Run Backward	24 ± 36	6 ± 9	11 ± 9	8 ± 5	14 ± 5	15 ± 13
		Walk	715 ± 84	14 ± 134	127 ± 92	29 ± 11	100 ± 31	45 ± 204
		Walk Backward	428 ± 290	18 ± 10	25 ± 37	48 ± 18	74 ± 27	84 ± 82
	Maze	Multi goal	250 ± 367	0 ± 17	67 ± 41	39 ± 37	16 ± 14	141 ± 50
	Quadruped	All tasks	217 ± 79	177 ± 27	108 ± 39	320 ± 26	264 ± 70	371 ± 135
		Jump	168 ± 69	120 ± 78	74 ± 75	255 ± 76	285 ± 96	297 ± 165
		Run	245 ± 139	119 ± 55	81 ± 94	310 ± 101	200 ± 107	204 ± 94
		Stand	371 ± 175	291 ± 46	155 ± 64	489 ± 128	348 ± 140	546 ± 288
		Walk	190 ± 86	147 ± 103	60 ± 37	252 ± 45	265 ± 61	329 ± 103
	Walker	All tasks	364 ± 166	222 ± 27	65 ± 4	336 ± 26	232 ± 50	514 ± 17
		Flip	273 ± 27	130 ± 56	49 ± 7	272 ± 27	208 ± 44	384 ± 19
		Run	204 ± 40	82 ± 19	23 ± 3	136 ± 24	96 ± 15	232 ± 24
		Stand	630 ± 88	461 ± 27	148 ± 14	547 ± 17	419 ± 102	790 ± 31
		Walk	454 ± 82	198 ± 63	38 ± 11	387 ± 55	191 ± 53	641 ± 46
2x	Cheetah	All tasks	312 ± 320	19 ± 14	58 ± 35	23 ± 7	56 ± 19	24 ± 22
		Run	71 ± 55	6 ± 30	9 ± 3	5 ± 1	20 ± 14	8 ± 27
		Run Backward	19 ± 37	6 ± 14	5 ± 3	9 ± 5	20 ± 7	13 ± 11
		Walk	775 ± 83	21 ± 31	147 ± 107	32 ± 11	91 ± 30	43 ± 21
		Walk Backward	552 ± 394	17 ± 19	35 ± 64	48 ± 27	93 ± 33	20 ± 79
	Maze	Multi goal	218 ± 355	0 ± 11	65 ± 50	44 ± 37	14 ± 12	131 ± 47
	Quadruped	All tasks	215 ± 53	132 ± 22	112 ± 27	270 ± 36	166 ± 85	341 ± 116
		Jump	169 ± 140	62 ± 38	74 ± 88	268 ± 79	170 ± 118	275 ± 159
		Run	221 ± 0	70 ± 66	86 ± 100	304 ± 102	140 ± 96	179 ± 117
		Stand	315 ± 193	294 ± 90	142 ± 30	353 ± 68	223 ± 197	421 ± 210
		Walk	209 ± 122	70 ± 61	57 ± 37	222 ± 78	111 ± 97	351 ± 139
	Walker	All tasks	243 ± 118	157 ± 28	60 ± 5	186 ± 39	151 ± 32	432 ± 24
		Flip	168 ± 40	105 ± 35	44 ± 6	149 ± 41	140 ± 28	268 ± 32
		Run	142 ± 31	65 ± 13	22 ± 4	78 ± 20	69 ± 16	229 ± 27
		Stand	434 ± 66	364 ± 53	143 ± 16	341 ± 62	237 ± 51	656 ± 12
		Walk	319 ± 79	79 ± 43	29 ± 6	183 ± 57	139 ± 49	567 ± 48

Table 6: **Full results on the ExORL environment generalisation 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. The Baseline FB-GRU represents the scores of an FB-GRU model trained solely on Cheetah-Occluded.

Environment	Task	FB-stack	**FB-TF (ours)**	**FB-S4d (ours)**	**FB-GRU (ours)**	Baseline FB-GRU
Walker	Walk	25 ± 7	34 ± 8	21 ± 2	22 ± 9	-
	Stand	126 ± 28	144 ± 23	95 ± 32	82 ± 18	-
	Run	17 ± 7	26 ± 5	18 ± 2	15 ± 12	-
	Flip	23 ± 8	27 ± 6	21 ± 2	23 ± 10	-
Quadruped	Walk	60 ± 64	99 ± 39	28 ± 30	110 ± 116	-
	Stand	150 ± 79	148 ± 84	80 ± 29	257 ± 128	-
	Run	74 ± 48	71 ± 26	25 ± 75	63 ± 56	-
	Jump	71 ± 35	58 ± 73	88 ± 72	190 ± 92	-
Cheetah	Walk	2 ± 3	24 ± 12	89 ± 24	10 ± 4	38 ± 216
	Walk Backward	12 ± 6	38 ± 21	22 ± 7	10 ± 12	3 ± 7
	Run	-	3 ± 2	16 ± 4	2 ± 1	8 ± 46
	Run Backward	2 ± 1	8 ± 4	4 ± 1	1 ± 2	0 ± 1