BLOCK CONTEXTUAL MDPS FOR CONTINUAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In reinforcement learning (RL), when defining a Markov Decision Process (MDP), the environment dynamics is implicitly assumed to be stationary. This assumption of stationarity, while simplifying, can be unrealistic in many scenarios. In the continual reinforcement learning scenario, the sequence of tasks is another source of nonstationarity. In this work, we propose to examine this continual reinforcement learning setting through the *block contextual MDP* (BC-MDP) framework, which enables us to relax the assumption of stationarity. This framework challenges RL algorithms to handle both nonstationarity and rich observation settings and, by additionally leveraging smoothness properties, enables us to study generalization bounds for this setting. Finally, we take inspiration from adaptive control to propose a novel algorithm that addresses the challenges introduced by this more realistic BC-MDP setting, allows for zero-shot adaptation at evaluation time, and achieves strong performance on several nonstationary environments.

1 INTRODUCTION

In the standard reinforcement learning (RL) regime, many limiting assumptions are made to make the problem setting tractable. A typical assumption is that the environment is stationary, i.e., the dynamics and reward do not change over time. However, most real-world settings – from fluctuating traffic patterns to evolving user behaviors in digital marketing to robots operating in the real world – do not conform to this assumption. In the more extreme cases, even the observation and action space can change over time. These setups are commonly grouped under the continual learning paradigm (Ring et al., 1994; Thrun, 1998; Hadsell et al., 2020) and non-stationarity is incorporated as a change in the task or environment distribution (that the agent is operating in). The ability to handle non-stationarity is a fundamental building block for developing continual learning agents (Khetarpal et al., 2020).

Real life settings present an additional challenge: we can not rely on access to (or knowledge of) an interpretable and compact (if not minimal) state space. Often, we only have access to a rich and high-dimensional observation space. For example, when driving a car on a wet street, we only have access to the "view" around us and not the friction coefficient between the car and the street. Hence, we must assume that observation contains irrelevant information (that could hinder generalization) and account for it when designing agents that could successfully operate in the nonstationary environments.

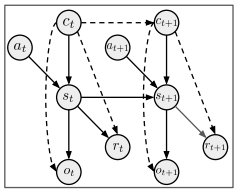


Figure 1: Graphical model of the BC-MDP setting.

We propose to model this more realistic, rich observation, nonstationary setting as a Block Contextual MDP (BC-MDP) (shown in Fig. 1) by combining two common assumptions: (i) the *block assumption* (Du et al., 2019) that addresses rich observations with irrelevant features and (ii) the *contextual MDP* (Hallak et al., 2015) assumption - MDPs with different dynamics and rewards share a common structure and a context that can describe the variation across tasks. We introduce the Lipschitz Block Contextual MDP framework that leverages results connecting Lipschitz functions to generalization (Xu & Mannor, 2012) and enables us to frame nonstationarity as a changing context over a family of stationary MDPs (thus modelling it as a contextual MDP). We propose a representation learning algorithm to enable the use of the current RL algorithms (that rely on the prototypical MDP setting) in nonstationary environments. It works by constructing a context space that is Lipschitz with respect to the changes in dynamics and reward of the nonstationary environment. We show, both theoretically

and empirically, that the trained agent generalizes well to unseen contexts. We also provide value bounds based on this approximate abstraction which depend on some basic assumptions.

This works is inspired from adaptive control (Slotine & Li, 1991), a control method that continuously performs parameter identification to adapt to nonstationary dynamics of the system. Adaptive control generally considers the "known unknowns," where the environment properties are known, but their values are unknown. We focus on the "unknown unknowns" setting, where the agent neither knows the environment property nor does it know its value in any task. Our setup is similar to meta-learning methods that "learn to learn," but meta-learning techniques generally require finetuning or updates on the novel tasks (Finn et al., 2017; Rakelly et al., 2019). *Our method does not need to perform parameter updates on the new tasks and can adapt in a zero-shot manner.* Since there are no parameter updates, our model does not suffer from catastrophic forgetting (McCloskey & Cohen, 1989). This property is especially critical when designing continual learning agents that operate in the real world. Further, our model can be *verified* after training and is guaranteed to stay true to that verification while also being capable of adapting zero-shot to new environments at test time. Intuitively, this follows from the observation that the agent's parameters are not updated when adapted to the unseen environments. We do not perform this type of formal verification, as current verification methods on neural networks only work for very small models, and are expensive to run (Katz et al., 2019). We refer to our proposed method as **Ze**ro-shot adaptation to **U**nknown **S**ystems (ZeUS).

Contributions. We 1) introduce the Lipschitz Block Contextual MDP framework for the continual RL setting, 2) provide theoretical bounds on adaptation and generalization ability to unseen tasks within this framework utilizing Lipschitz properties, 3) propose an algorithm (ZeUS) to perform online inference of "unknown unknowns" to solve a family of tasks (without performing learning updates at test time) and ensure the prior Lipschitz properties hold, and 4) empirically verify the effectiveness of ZeUS on environments with nonstationary dynamics or reward functions.

2 Related Work

Our work is related to four broad areas: (i) System Identification and Adaptive Control, (ii) Continual RL, (iii) Context Modeling and (iv) Meta RL and Multitask RL.

System Identification and Adaptive Control (Zadeh, 1956; Åström & Bohlin, 1965; Swevers et al., 1997; Bhat et al., 2002; Gevers et al., 2006; Ljung, 2010; Van Overschee & De Moor, 2012; Chiuso & Pillonetto, 2019; Ajay et al., 2019; Yu et al., 2017; Zhu et al., 2017). In this setup, the goal is to perform system identification of "known unknowns," where the environment properties are known, but their values are unknown. Continuing with the previous example of driving a car on a wet street, in this setup, the agent knows that friction coefficient varies across tasks but does not know its value. By inferring the value from observed data, the agent can condition its policy (on the inferred value) to solve a given task. The conventional approaches alternate between system identification and control policy optimization. One limitation of this approach is that a good initial policy and system identifier are required for efficient training. We extend this setup to the "unknown unknowns" setting, where the agent neither knows the environment property nor does it know its value in any task.

Our work is related to the **Continual (or Lifelong) RL** (Ring et al., 1994; Gama et al., 2014; Abel et al., 2018; Kaplanis et al., 2018; Xu & Zhu, 2018; Aljundi et al., 2019; Javed & White, 2019; Hadsell et al., 2020). In this setup, nonstationarity can manifest in two ways: (i) *Active nonstationarity* where the agent's actions may change the environment dynamics or action space (e.g., a cleaning robot tripping over the carpet), (ii) *Passive nonstationarity* where the environment dynamics may change irrespective of the agent's behavior (e.g., driving a car on a snow-covered road) (Khetarpal et al., 2020). Our work relates to the passive nonstationarity setup. Unlike Lopez-Paz & Ranzato (2017); Chaudhry et al. (2019); Aljundi et al. (2019); Sodhani et al. (2020) which focus on challenges like catastrophic forgetting (McCloskey & Cohen, 1989) [1], we focus on the ability to continually adapt (the policy) to unseen tasks (Hadsell et al., 2020). Xie et al. (2020) proposed LILAC that uses a probabilistic hierarchical latent variable model to learn a representation of the environment from current and past experiences and perform off-policy learning. A key distinction of our work is that we use task metrics to learn a context space and focus on generalization to unseen contexts.

[1] Since our model does not perform parameter updates when transferring to unseen tasks, it does not suffer from catastrophic forgetting.

Several works have focused on **modeling the environment context** from high-level pixel observations (Pathak et al., 2017; Ebert et al., 2018; Chen et al., 2018; Xu et al., 2019). This context (along with the observation) is fed as input to the policy to enable it to adapt to unseen dynamics (by implicitly capturing the dynamics parameters). T environment's context is encoded using a *context encoder* using the history of interactions. These approaches learn a single, global dynamics model conditioned on the output of the context encoder. Similar to these approaches, we also use a context encoder but introduce an additional loss to learn a context space with Lipschitz properties with respect to reward, and dynamics. Xian et al. (2021) proposed using HyperNetworks (Ha et al., 2016; Chang et al., 2019; Klocek et al., 2019; Meyerson & Miikkulainen, 2019) that use the context to generate the weights of the *expert* dynamics model.

Other works on structured MDPs, that leverage the Lipschitz properties, include Modi et al. (2018) that assumes that the given contextual MDP is smooth and that the distance metric and Lipschitz constants are known. In contrast, we propose a method that constructs a new smooth contextual MDP, with bounds on downstream behavior based on the approximate-ness of the new contextual MDP. Modi & Tewari (2020) propose RL algorithms with lower bounds on regret but assume that the context is known and linear with respect to the MDP parameters. In contrast, we do not assume access to the context at train or test time or linearity with respect to MDP parameters.

Meta-reinforcement learning aims to "meta-learn" how to solve new tasks efficiently (Finn et al., 2017; Clavera et al., 2019; Rakelly et al., 2019; Zhao et al., 2020). Optimization-based meta-RL methods (Finn et al., 2017; Mishra et al., 2017; Zintgraf et al., 2019) require updating model parameters for each task and therefore suffer from catastrophic forgetting in the continual learning setting. Context-based meta-RL methods perform online adaptation given a context representation (generally modeled as the hidden representations of a RNN (Nagabandi et al., 2019)). The hope is that the model would (i) adapt to the given context and (ii) correctly infer the next state. However, follow-up work (Lee et al., 2020) suggests that it is better to disentangle the two tasks by learning a context encoder (for adaption) and a context-conditioned dynamics model (for inferring the next state). Lee et al. (2020) also introduced additional loss terms when training the agent. However, their objective is to encourage the context encoding to be useful for predicting both forward (next state) and backward (previous state) dynamics while being temporally consistent, while our objective is to learn a context space with Lipschitz properties with respect to reward and dynamics. Other works have proposed modeling meta-RL as task inference Humplik et al. (2019); Kamienny et al. (2020) but these works generally assume access to some *privileged information* (like *task-id*) during training.

Our work is also related to the general problem of training a policy on Partially Observable Markov Decision Processes (POMDPs) (Kaelbling et al., 1998; Igl et al., 2018; Zhang et al., 2019; Han et al., 2019; Hafner et al., 2019) that capture both nonstationarity and rich observation settings. Our experiments are performed in the POMDP setup where we train the agent using pixel observations and do not have access to a compact state-space representation. However, we focus on a specific class of POMDPs — the contextual MDP with hidden context, which enables us to obtain strong generalization performance to new environments. Finally, we discuss additional related works in multi-task RL, transfer learning, and MDP metrics in Appendix A.

3 Background & Notation

A **Markov Decision Process** (MDP) (Bellman, 1957; Puterman, 1995) is defined by a tuple $\langle \mathcal{S}, \mathcal{A}, R, T, \gamma \rangle$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, $T : \mathcal{S} \times \mathcal{A} \rightarrow Dist(\mathcal{S})$ is the environment transition probability function, and $\gamma \in [0, 1)$ is the discount factor. At each time step, the learning agent perceives a state $s_t \in \mathcal{S}$, takes an action $a_t \in \mathcal{A}$ drawn from a policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, and with probability $T(s_{t+1}|s_t, a_t)$ enters next state s_{t+1}, receiving a numerical reward R_{t+1} from the environment. The value function of policy π is defined as: $V_\pi(s) = E_\pi[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | S_0 = s]$. The optimal value function V^* is the maximum value function over the class of stationary policies.

Contextual Markov Decision Processes were first proposed by Hallak et al. (2015) as an augmented form of Markov Decision Processes that utilize *side information* as a form of context, similar to contextual bandits. For example, the friction coefficient of a surface for a robot sliding objects across a table is a form of context variable that affects the environment dynamics, or user information like age and gender are context variables that influence their movie preferences.

Definition 1 (Contextual Markov Decision Process)**.** *A contextual Markov decision process (CMDP) is defined by tuple $\langle \mathcal{C}, \mathcal{S}, \mathcal{A}, \mathcal{M} \rangle$ where $\mathcal{C}$ is the context space, $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space. $\mathcal{M}$ is a function which maps a context $c \in \mathcal{C}$ to MDP parameters $\mathcal{M}(c) = \{R^c, T^c\}$.*

However, in the real world, we typically operate in a "rich observation" setting where we do not have access to a compressed state representation and the learning agent has to learn a mapping from the observation to the state. This additional relaxation of the original CMDP definition as a form of Block MDP (Du et al., 2019) was previously introduced in Sodhani et al. (2021b) for the multi-task setting where the agent focuses on a subset of the whole space for a specific task, which we again present here for clarity:

Definition 2 (Block Contextual Markov Decision Process (Sodhani et al., 2021b))**.** *A block contextual Markov decision process (BC-MDP) (Fig. 1) is defined by tuple $\langle \mathcal{C}, \mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{M} \rangle$ where $\mathcal{C}$ is the context space, $\mathcal{S}$ is the state space, $\mathcal{O}$ is the observation space, $\mathcal{A}$ is the action space. $\mathcal{M}$ is a function which maps a context $c \in \mathcal{C}$ to MDP parameters and observation space $\mathcal{M}(c) = \{R^c, T^c, \mathcal{O}^c\}$.*

Consider a robot moving around in a warehouse and performing different tasks. Rather than specifying an observation space that covers the robot's lifelong trajectory, it is much more practical to have the observation space change as its location and attention change, as it would with an attached camera. We can still keep the assumption of full observability because the robot can have full information required to solve the current task, e.g. with frame stacking to capture velocity and acceleration information. The continual learning setting differs from sequential multi-task learning as there is no delineation of tasks when c changes, causing nonstationarity in the environment. We make an additional assumption that the change in c is smooth over time and the BC-MDP itself is smooth, as shown in Definition 3. We now define a Lipschitz MDP for the MDP family we are concerned with.

Definition 3 (Lipschitz Block Contextual MDP)**.** *Given a BC-MDP $\langle \mathcal{C}, \mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{M} \rangle$ and a distance metric $d(\cdot, \cdot)$ over context space, if for any two contexts $c_1, c_2 \in \mathcal{C}$, we have the following constraints,*

$$\forall (s,a), W(T^{c_1}(s,a), T^{c_2}(s,a)) \leq L_p d(c_1, c_2),$$
$$\forall (s,a), \|R^{c_1}(s,a) - R^{c_2}(s,a)\| \leq L_r d(c_1, c_2),$$

then the BC-MDP is referred to as a Lipschitz BC-MDP with smoothness parameters L_p and L_r.

Here W denotes the Wasserstein distance. Note that Definition 3 is not a limiting assumption because we do not assume access to the context variables c_1 and c_2, and they can therefore be chosen so that the Lipschitz condition is always satisfied. In this work, we focus on a method for learning a context space that satisfies the above property.

4 Generalization Properties of Lipschitz BC-MDPs

The key idea behind the proposed method (presented in full in Section 5) is to construct a context space $\mathcal{C}$ with Lipschitz properties with respect to dynamics and reward, and therefore, optimal value functions across tasks. In this section, we show how this Lipschitz property aids generalization. The following results hold true for any given observation (or state) space and are not unique to Block MDPs, so we use notation with respect to states $s \in \mathcal{S}$ without loss of generality. Since we do not have access to the true context space, in Section 5, we describe how to learn a context space with the desired characteristics.

In order to construct a context space that is Lipschitz with respect to tasks, notably the optimal value functions across tasks, we turn to metrics based on state abstractions. Based on established results on distance metrics over states (see Appendix D), we can define a task distance metric for the continual RL setting.

Definition 4 (Task Metric)**.** *Given two tasks sampled from a BC-MDP, identified by contexts c_i & c_j,*

$$d_{task}(c_i, c_j) := \max_{s,a \in \{S,A\}} \Big[\big| R^{c_i}(s,a) - R^{c_j}(s,a) \big| + W(d_{task})\big(T^{c_i}(s,a), T^{c_j}(s,a)\big) \Big], \quad (1)$$

where $W(d_{task})$ is the Wasserstein distance between transition probability distributions.

We can now show that the dynamics, reward, and optimal value function are all also Lipschitz with respect to d_{task}. The first two are clear results from Definition 4.

Theorem 1 (V_c^* is Lipschitz with respect to d_{task})**.** *Let V^* be the optimal, universal value function for a given discount factor γ and context space $\mathcal{C}$. Then V^* is Lipschitz continuous with respect to d_{task} with Lipschitz constant $\frac{1}{1-\gamma}$ for any $s \in \mathcal{S}$,*

$$|V^*(s,c) - V^*(s,c')| \leq \frac{1}{1-\gamma} d_{task}(c,c').$$

The proof can be found in Appendix E. Applying Theorem 1 to a continual RL setting requires the context be identifiable from a limited number of interactions with the environment. This warrants the following assumption:

Assumption 1 (Identifiability)**.** *Let k be some constant number of steps the agent takes in a new environment with context c. There exists an $\epsilon_c > 0$ such that a context encoder ψ can take those transition tuples $(s_i, a_i, s_i', r_i), i \in \{1, ..., k\}$ and output a predicted context $\hat{c}$ that is ϵ_c-close to c.*

There are two key assumptions wrapped up in Assumption 1. The first is that the new environment is uniquely identifiable from k transitions, and the second is that we have a context encoder that can approximately infer that context. In practice, we use neural networks for modeling ψ and verify that neural networks can indeed learn to infer the context, as shown in Table 1 in Appendix B.2.

Why do we care about the Lipschitz property? Xu & Mannor (2012) established that Lipschitz continuous functions are robust, i.e. the gap between test and training error is bounded. This result is only useful when the problem space is Lipschitz, which is often not the case in RL. However, we have shown that any BC-MDP is Lipschitz continuous with respect to metric d_{task}. We now define a general supervised learning setup to bound the error of learning dynamics and reward models. The following result requires that the data-collecting policy is ergodic, i.e. a Doeblin Markovian chain (Doob, 1953; Meyn & Tweedie, 1993), defined as follows.

Definition 5 (Doeblin chain)**.** *A Markov chain $\{s_i\}_{i=1}^{\infty}$ on a state space $\mathcal{S}$ is a Doeblin chain (with α and t) if there exists a probability measure ρ on $\mathcal{S}$, $\alpha > 0$, an integer $t \geq 1$ such that*

$$P(s_t \in H | s_0 = s) \geq \alpha \rho(H); \quad \forall \text{ measurable } H \subseteq \mathcal{S}; \forall s \in \mathcal{S}.$$

Let $\hat{\mathcal{L}}(\cdot)$ denote expected error and $\mathcal{L}_{\text{emp}}(\cdot)$ denote training error of an algorithm $\mathcal{A}$ on training data $\mathbf{s} = \{s_1, ..., s_n\}$ and evaluated on points $z \in \mathcal{Z}$ sampled from distribution μ:

$$\hat{\mathcal{L}}(\cdot) := \mathbb{E}_{z \sim \mu} \mathcal{L}(\mathcal{A}_{\mathbf{s}}, z); \quad \mathcal{L}_{\text{emp}}(\cdot) := \frac{1}{n} \sum_{s_i \in \mathbf{s}} \mathcal{L}(\mathcal{A}_{\mathbf{s}}, s_i).$$

Here, $\mathcal{A}_{\mathbf{s}}$ denotes the instantiation of the learned algorithm trained on data $\mathbf{s}$ whereas $\mathcal{A}$ refers to the general learning algorithm. We can now bound the generalization gap, the difference between expected and training error using a result from Xu & Mannor (2012).

Theorem 2 (Generalization via Lipschitz Continuity (Xu & Mannor, 2012))**.** *If a learning algorithm $\mathcal{A}$ is $\frac{1}{1-\gamma}$-Lipschitz and the training data $\mathbf{s} = \{s_1, ..., s_n\}$ are the first n outputs of a Doeblin chain with constants α, T, then for any $\delta > 0$ with probability at least $1 - \delta$,*

$$\left|\hat{\mathcal{L}}(\mathcal{A}_{\mathbf{s}}) - \mathcal{L}_{emp}(\mathcal{A}_{\mathbf{s}})\right| \leq \frac{\epsilon}{1-\gamma} + M\left(\frac{8t^2(K \ln 2 + \ln(1/\delta))}{\alpha^2 n}\right)^{1/4}.$$

K denotes the ϵ-covering number of the state space. ϵ controls the granularity at which we discretize, or partition, that space. If ϵ is larger, K is smaller. M is a scalar that uniformly upper-bounds the loss $\mathcal{L}$. Once we learn a smooth context space, this result bounds the generalization error of supervised learning problems like learned dynamics and reward models. These learned models allow us to construct a new MDP that is $\epsilon_R, \epsilon_T, \epsilon_c$-close to the original. We can now show how this error propagates when learning a policy.

Theorem 3 (Generalization Bound)**.** *Without loss of generality we assume all tasks in a given BC-MDP family have reward bounded in $[0, 1]$. Given two tasks $\mathcal{M}_{c_i}$ and $\mathcal{M}_{c_j}$, we can bound the difference in Q^π between the two MDPs for a given policy π learned under an $\epsilon_R, \epsilon_T, \epsilon_{c_i}$-approximate abstraction of $\mathcal{M}_{c_i}$ and applied to $\mathcal{M}_{c_j}$,*

$$\left\|Q^*_{\mathcal{M}_{c_j}} - [Q^*_{\bar{\mathcal{M}}_{\hat{c}_i}}]_{\mathcal{M}_{c_j}}\right\|_\infty \leq \epsilon_R + \gamma\big(\epsilon_T + \epsilon_{c_i} + \|c_i - c_j\|_1\big)\frac{1}{2(1-\gamma)}.$$

Proof in Appendix E. Theorem 3 shows that if we learn an ϵ-optimal context-conditioned policy for task with context c_i and encounter a new context c_j at evaluation time where c_j is close to c_i, then the context-conditioned policy will be ϵ-optimal for the new task by leveraging the Lipschitz property. While these results clearly do not scale well with the dimensionality of the state space and discount factor γ, it shows that representation learning is a viable approach to developing robust world models (Theorem 2), which translates to tighter bounds on the suboptimality of learned Q functions (Theorem 3).

5 Zero-shot Adaptation to Unknown Systems

Based on the findings in Section 4, we can improve generalization by constructing a context space that is Lipschitz with respect to the changes in dynamics and reward of the nonstationary environment. In practice, computing the maximum Wasserstein distance over the entire state-action space is computationally infeasible. We relax this requirement by taking the expectation over Wasserstein distance with respect to the marginal state distribution of the behavior policy. This leads us to a representation learning objective that leverages this relaxed version of the task metric in Definition 4:

$$\mathcal{L}(\phi, \psi, T, R) = MSE\underbrace{\left(\left|\left|\psi(H_1) - \psi(H_2)\right|\right|_2, d(c_1, c_2)\right)}_{\text{context loss}} + MSE\underbrace{\left(T(\phi(o_t^{c_1}), a_t^{c_1}, \psi(H_1)), \phi(o_{t+1}^{c_1})\right)}_{\text{Dynamics loss}}$$
$$+ MSE\underbrace{\left(R(\phi(o_t^{c_1}), a_t^{c_1}, \psi(H_1)), r_{t+1}^{c_1}\right)}_{\text{Reward loss}}. \tag{2}$$

where red indicates stopped gradients. $H_1 := \{o_t^{c_1}, a_t, r_t, o_{t+1}^{c_1}, ...\}$ and $H_2 := \{o_t^{c_2}, a_t, r_t, o_{t+1}^{c_2}, ...\}$ are transition sequences from two environments with contexts c_1 and c_2 respectively. During training, the transitions are uniformly sampled from a replay buffer. We do not require access to the true context for computing $d(c_1, c_2)$ (in Equation (2)) as we can approximate $d(c_1, c_2)$ using Definition 3. Specifically, we train a transition dynamics model and a reward model (via supervised learning) and use their output to approximate $d(c_1, c_2)$. In practice, we scale the context learning error, our task metric loss, using a scalar value denoted as α_ψ.

We describe the architecture of ZeUS in Figure 2. We have an observation encoder ϕ that encodes the pixel-observations into real-valued vectors. A buffer of interaction-history is maintained for computing the context. The context encoder first encodes the individual state-action transition pairs and then aggregates the representations using standard operations: *sum*, *mean*, *concate*, *product*, *min* and *max* [2]. All the components are instantiated using feedforward networks. During inference, assume that the agent is operating in some environment denoted by (latent) context c_1. At time t, the agent gets an observation $o_t^{c_1}$ which is encoded into $s_t^{c_1} := \phi(o_t^{c_1})$[3]. The context encoder ψ encodes the last k interactions (denoted as H_1) into a context encoding $c_1 := \psi(H_1)$[4]. The observation and context encodings are concatenated and fed to the policy to get the action.

During training, we sample a batch of interaction sequences from the buffer. For sake of exposition, we assume that we sample only 2 sequences H_1 and H_2. Similar to the inference pipeline, we compute $\phi(o_t^{c_1}), \psi(H_1), \phi(o_t^{c_2})$ and $\psi(H_2)$ and the loss (Equation (2)). We highlight that the algorithm does not know if the two (sampled) interactions correspond to the same context or not. Hence, in a small percentage of cases, H_1 and H_2 could correspond to the same context and the context loss will be equal to 0. For implementing the loss in equation Equation (2), we do not need access to the true context as the distance between the contexts can be approximated using the learned transition and reward models using Definition 3. The pseudo-code is provided in Algorithm 1 (Appendix B). Since ZeUS is a representation learning algorithm, it must be paired with a policy optimization algorithm for end-to-end training. In the scope of this work, we use Soft Actor-Critic with auto-encoder loss (SAC-AE, Yarats et al. (2019)), though ZeUS can be used with any policy optimization algorithm.

[2] We experiment with these aggregation operators for all the baselines and not just ZeUS.

[3] We overload notation here since the true state space is latent.

[4] We again overload notation here since the true context space is also latent.

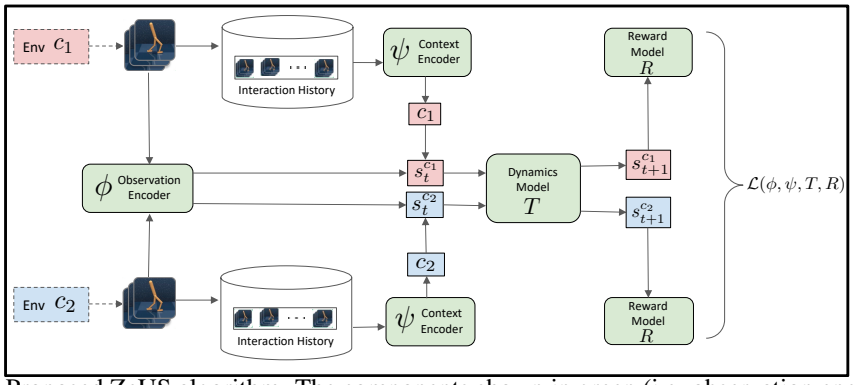


Figure 2: Proposed ZeUS algorithm. The components shown in green (i.e. observation encoder, context encoder, dynamics model and reward model) are shared across tasks. Components/representations in red or blue belong to separate tasks.

6 Experiments

We design our experiments to answer the following questions:

1. How well does ZeUS perform when training over a family of tasks with varying dynamics? (see Figure 8 in Appendix)
2. Can ZeUS adapt and generalize to unseen environments (with novel dynamics or reward) without performing any gradient updates? (see Figure 3 and Figure 4)
3. Can ZeUS learning meaningful context representations when training over a family of tasks with varying dynamics? (see Figure 5)

6.1 Setup

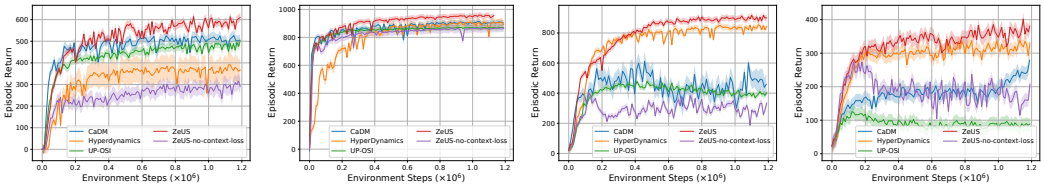
Similar to the setups from Zhou et al. (2019); Lee et al. (2020); Zhang et al. (2021b), we start with standard RL environments and extend them by modifying parameters that affect the dynamics (e.g. the friction coefficient between the agent and the ground) or the reward (e.g. target velocity) such that they exhibit the challenging nonstationarity and rich-observation conditions of our BC-MDP setting. For varying the transition dynamics, we use the following Mujoco (Todorov et al., 2012)[5] based environments from the DM Control Suite (Tassa et al., 2018): Cheetah-Run-v0 (vary the length of the torso of the cheetah), Walker-Walk-v0 (vary the friction coefficient between the walker and the ground), Walker-Walk-v1 (vary the length of the foot of the walker) and Finger-Spin-v0 task (vary the size of the finger). For environments with varying reward function, we use the Cheetah-Run-v1 environment (vary the target velocity that the agent has to reach) and Sawyer-Peg-v0 environment (vary the goal position for inserting the per) from Zhao et al. (2020). For environments with varying reward function, we assume access to the reward function, as done in Zhao et al. (2020).

For all environments, we pre-define a range of parameters to train and evaluate on. For environments with nonstationary transition dynamics, we create two set of parameters for evaluation *interpolation* (and *extrapolation*) where the parameters are sampled from a range that lies within (and outside) the range of parameters used for training. For the environments with varying reward function, we sample the parameters for the test environments from the same range as the training environments. For additional details refer to Appendix B.2. We report the evaluation performance of the best performing hyper-parameters for all algorithms (measured in terms of the training performance). All the experiments are run with 10 seeds and we report both the mean and the standard error (denoted by the shaded area on the plots). For additional implementation details refer to Appendix B.

6.2 Baselines

We select representative baselines from different areas of related work (Section 2): *UP-OSI* (Yu et al., 2017) is a system identification approach that infers the true parameters (of the system) and conditioning the policy on the inferred parameters. *Context-aware Dynamics Model* , *CaDM* (Lee

[5]License related information available at: https://www.roboti.us/license.html



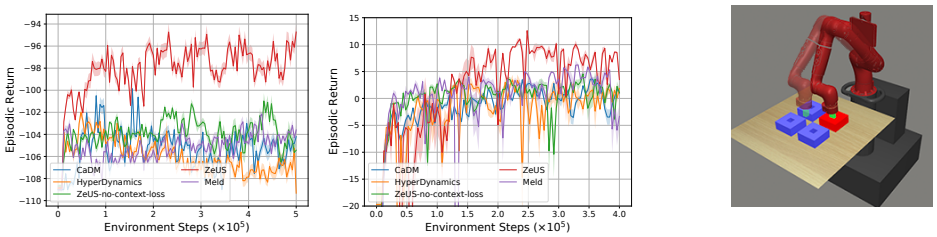
(a) Cheetah-Run-v0 (b) Finger-Spin-v0 (c) Walker-Walk-v0 (d) Walker-Walk-v1

Figure 3: We compare the performance of the proposed ZeUS algorithm with *CaDM*, *UP-OSI*, *HyperDynamics* and *ZeUS-no-context-loss* algorithms on the heldout evaluation environments (extrapolation) for four families of tasks with different dynamics parameters.

et al., 2020) is a context modelling based approach that is shown to outperform Gradient and Recurrence-based meta learning approaches (Nagabandi et al., 2018). *HyperDynamics*(Xian et al., 2021) generates the weights of the dynamics model (for each environment) by conditioning on a context vector and is shown to outperform both ensemble of experts and meta-learning based approaches (Nagabandi et al., 2018). We also consider a *Context-conditioned Policy* where the context encoder is trained using the one-step forward dynamics loss. This approach can be seen as an ablation of the ZeUS algorithm without the context learning error (from Equation (2)). We refer to it as *Zeus-no-context-loss*.

6.3 Adapting and Generalizing to Unseen Environments

In Figure 3, we compare ZeUS's performance on the heldout *extrapolation* evaluation environments which the agent has not seen during training. The transition dynamics varies across these tasks. *HyperDynamics* performs well on some environments but requires more resources to train (given that it generates the weights of dynamics models for each transition in the training batch). *UP-OSI* uses privileged information (in terms of the extra supervision). Both *CaDM* and ZeUS are reasonably straightforward to implement (and train) though ZeUS outperforms the other baselines. The context loss (Equation (2)) is an important ingredient for the generalization performance as observed by the performance of *Zeus-no-context-loss*. The corresponding plots for performance on the training environments and heldout *interpolation* evaluation environments are given in Figure 8 and Figure 9 (in Appendix) respectively. For additional ablation results for these environments, refer to C.3.



(a) Cheetah-Run-v1 (b) Sawyer-Peg-v0 (c) Sawyer-Peg-v0

Figure 4: (a), (b): We compare the performance of the proposed ZeUS algorithm with *CaDM*, *HyperDynamics*, *ZeUS-no-bisim* and *Meld* algorithms on environments with different reward functions. (c): Illustration of the Sawyer-Peg-V0 task.

In Figure 4, we compare ZeUS's performance with the baselines when the reward function varies across tasks. Since all the models have access to the reward (i.e. reward is concatenated as part of history), we do not compare with UP-OSI which is trained to infer the reward. Instead we include an additional baseline,

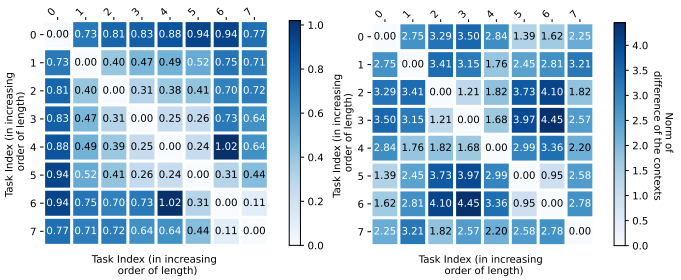


Figure 5: Norm of pairwise differences of contexts for different tasks for Cheetah-Run-v0 setup when trained with context loss (left) and without context loss (right).

Meld (Zhu et al., 2020), a meta-RL approach that performs inference in a latent state model to adapt to a new task. Like before, ZeUS outperforms the other baselines.

6.4 Learning a Meaningful Context Representation

We want to evaluate if the context representation constructed by ZeUS contains meaningful information about the true context of the BC-MDP. We compute the norm of pairwise difference of the learned contexts (corresponding to different tasks) for the Cheetah-Run-v0 setup (where the torso of the cheetah varies across the tasks) when it is trained with and without the context loss (Equation (2)). As shown in Figure 5 (left), when training with the context loss, tasks that are closer in terms of torso length are also *consistently* closer in the context space and pairs of tasks with larger differences of torso length are also consistently further apart. We also compute the Spearman's rank correlation coefficient between the ranking (of distance) between the learned contexts and the ground truth context. The coefficient is much higher (0.60) when trained with the context loss than training without the context loss (0.23), showing that the context loss is useful for training representations that capture the relationship across tasks of the true underlying context variable without *privileged information* of task ids or the true context.

7 Limitations

A theoretical limitation of this work is the inability to provide guarantees based on the likelihood of the model learning the correct causal dynamics. By structuring the context space to be Lipschitz, we can give guarantees only for those dynamics and reward where the context is close to the contexts seen at training time. While this result flows directly from Theorem 3, it is important to be aware of this limitation when using ZeUS in practice, namely that it may have poor performance when the distance between the contexts (corresponding to the training and the evaluation tasks) is high. We demonstrate an example in Figure 6, where we plot the performance of the agent for different values of target velocities (for Cheetah-Run-v1). While ZeUS outperforms the other methods, its performance also degrades as we move away from the training distribution.

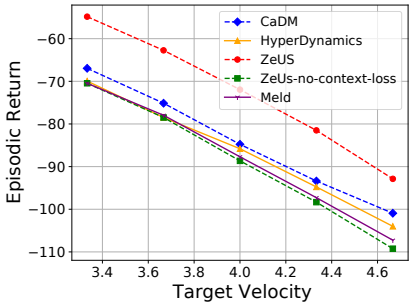


Figure 6: The performance of all the algorithms (on Cheetah-Run-v1) degrades as we move away from the training distribution.

8 Discussion

In this work, we propose to use the Block Contextual MDP framework (Sodhani et al., 2021b) to model the nonstationary, rich observation, RL setting. This allows classical RL methods that typically rely on a stationarity assumption to be used in continual learning, and we show that our adaptive approach can zero-shot generalize to new, unseen environments. We provide theoretical bounds on adaptation and generalization ability to unseen tasks within this framework and propose a representation learning algorithm (ZeUS) for performing online inference of "unknown unknowns". We empirically verify the effectiveness of the proposed algorithm on environments with nonstationary dynamics and reward functions and show that a crucial component of ZeUS is the context loss that ensures smoothness in the context space. This context loss successfully captures meaningful information about the true context, as verified in Section 6.4.

There are a few interesting directions for further research. One way to improve the generalization performance and tighten the generalization bounds is by constraining the neural networks used in ZeUS to have smaller Lipschitz constants. This is known to be able to improve generalization bounds (Neyshabur et al., 2015). We can also consider improving the algorithm to infer the underlying causal structure of the dynamics, as discussed in Section 7. This is a much harder problem than constructing a context space and inferring context in new environments. A second direction is to extend ZeUS to account for *active nonstationarity*, where the agent's actions can affect the environment. ZeUS would work for this setting, but there is clearly additional structure that can be leveraged for improved performance. There are connections to existing work in multi-agent RL.

REFERENCES

David Abel, Yuu Jinnai, Sophie Yue Guo, George Konidaris, and Michael Littman. Policy and value transfer in lifelong reinforcement learning. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 20–29. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/abel18b.html.

Anurag Ajay, Maria Bauza, Jiajun Wu, Nima Fazeli, Joshua B Tenenbaum, Alberto Rodriguez, and Leslie P Kaelbling. Combining physical simulators and object-based networks for control. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 3217–3223. IEEE, 2019.

Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Min Lin, Laurent Charlin, and Tinne Tuytelaars. Online continual learning with maximally interfered retrieval. *arXiv preprint arXiv:1908.04742*, 2019.

Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning*, pp. 166–175. PMLR, 2017.

Kavosh Asadi, Dipendra Misra, and Michael Littman. Lipschitz continuity in model-based reinforcement learning. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 264–273, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/asadi18a.html.

Ershad Banijamali, Rui Shu, Hung Bui, Ali Ghodsi, et al. Robust locally-linear controllable embedding. In *International Conference on Artificial Intelligence and Statistics*, pp. 1751–1759. PMLR, 2018.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado Van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *arXiv preprint arXiv:1606.05312*, 2016.

Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

Kiran S Bhat, Steven M Seitz, Jovan Popović, and Pradeep K Khosla. Computing the physical parameters of rigid-body motion from video. In *European Conference on Computer Vision*, pp. 551–565. Springer, 2002.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Oscar Chang, Lampros Flokas, and Hod Lipson. Principled weight initialization for hypernetworks. In *International Conference on Learning Representations*, 2019.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

Tao Chen, Adithyavairavan Murali, and Abhinav Gupta. Hardware conditioned policies for multi-robot transfer learning. *arXiv preprint arXiv:1811.09864*, 2018.

Alessandro Chiuso and Gianluigi Pillonetto. System identification: A machine learning perspective. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:281–304, 2019.

Ignasi Clavera, Anusha Nagabandi, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyztsoC5Y7.

Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 2169–2176. IEEE, 2017.

J. L. Doob. *Stochastic processes*. John Wiley & Sons, New York, 1953. MR 15,445b. Zbl 0053.26802.

Finale Doshi-Velez and George Konidaris. Hidden Parameter Markov Decision Processes: A Semi-parametric Regression Approach for Discovering Latent Task Parametrizations. *arXiv:1308.3513 [cs]*, August 2013. arXiv: 1308.3513.

Kenji Doya, Kazuyuki Samejima, Ken-ichi Katagiri, and Mitsuo Kawato. Multiple model-based reinforcement learning. *Neural computation*, 14(6):1347–1369, 2002.

Simon Du, Akshay Krishnamurthy, Nan Jiang, Alekh Agarwal, Miroslav Dudik, and John Langford. Provably efficient RL with rich observations via latent state decoding. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1665–1674. PMLR, 09–15 Jun 2019.

Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

Fernando Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pp. 720–727, 2006.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, pp. 162–169, Arlington, Virginia, United States, 2004. AUAI Press. ISBN 0-9749039-0-6.

Norm Ferns, Prakash Panangaden, and Doina Precup. Bisimulation metrics for continuous markov decision processes. *SIAM J. Comput.*, 40(6):1662–1714, December 2011. ISSN 0097-5397. doi: 10.1137/10080484X.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017.

Katerina Fragkiadaki, Pulkit Agrawal, Sergey Levine, and Jitendra Malik. Learning visual predictive models of physics for playing billiards. *arXiv preprint arXiv:1511.07404*, 2015.

João Gama, Indrundefined Žliobaitundefined, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4), March 2014. ISSN 0360-0300. doi: 10.1145/2523813. URL `https://doi.org/10.1145/2523813`.

Michel Gevers et al. System identification without lennart ljung: what would have been different? *Forever Ljung in System Identification, Studentlitteratur AB, Norrtalje*, 2, 2006.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 2020.

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, 2019.

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

Dongqi Han, Kenji Doya, and Jun Tani. Variational recurrent models for solving partially observable control tasks. *arXiv preprint arXiv:1912.10703*, 2019.

Jan Humplik, Alexandre Galashov, Leonard Hasenclever, Pedro A Ortega, Yee Whye Teh, and Nicolas Heess. Meta reinforcement learning as task inference. *arXiv preprint arXiv:1905.06424*, 2019.

Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for pomdps. In *International Conference on Machine Learning*, pp. 2117–2126. PMLR, 2018.

Maximilian Igl, Andrew Gambardella, Jinke He, Nantas Nardelli, N. Siddharth, Wendelin Böhmer, and Shimon Whiteson. Multitask soft option learning, 2020. URL https://openreview.net/forum?id=BkeDGJBKvB.

Khurram Javed and Martha White. Meta-learning representations for continual learning. *arXiv preprint arXiv:1905.12588*, 2019.

Nan Jiang, Alex Kulesza, and Satinder Singh. Abstraction selection in model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 179–188, 2015.

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.

Pierre-Alexandre Kamienny, Matteo Pirotta, Alessandro Lazaric, Thibault Lavril, Nicolas Usunier, and Ludovic Denoyer. Learning adaptive exploration strategies in dynamic environments through informed policy regularization. *arXiv preprint arXiv:2005.02934*, 2020.

Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Continual reinforcement learning with complex synapses. In *International Conference on Machine Learning*, pp. 2497–2506. PMLR, 2018.

Guy Katz, Derek A Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, et al. The marabou framework for verification and analysis of deep neural networks. In *International Conference on Computer Aided Verification*, pp. 443–452. Springer, 2019.

Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *arXiv preprint arXiv:2012.13490*, 2020.

Sylwester Klocek, Łukasz Maziarka, Maciej Wołczyk, Jacek Tabor, Jakub Nowak, and Marek Śmieja. Hypernetwork functional image representation. In *International Conference on Artificial Neural Networks*, pp. 496–510. Springer, 2019.

Iasonas Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6129–6138, 2017.

Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5757–5766. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/lee20g.html.

Lennart Ljung. Perspectives on system identification. *Annual Reviews in Control*, 34(1):1–12, 2010. ISSN 1367-5788. doi: https://doi.org/10.1016/j.arcontrol.2009.12.001. URL https://www.sciencedirect.com/science/article/pii/S1367578810000027.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *arXiv preprint arXiv:1706.08840*, 2017.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Elliot Meyerson and Risto Miikkulainen. Modular universal reparameterization: Deep multi-task learning across diverse domains. *arXiv preprint arXiv:1906.00097*, 2019.

S.P. Meyn and R.L. Tweedie. *Markov Chains and Stochastic Stability*. Springer-Verlag, London, 1993. URL /brokenurl#probability.ca/MT.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

Aditya Modi and Ambuj Tewari. No-regret exploration in contextual reinforcement learning. In *Conference on Uncertainty in Artificial Intelligence*, pp. 829–838. PMLR, 2020.

Aditya Modi, Nan Jiang, Satinder Singh, and Ambuj Tewari. Markov decision processes with continuous side information. In *Algorithmic Learning Theory*, pp. 597–618. PMLR, 2018.

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.

Anusha Nagabandi, Chelsea Finn, and Sergey Levine. Deep online learning via meta-learning: Continual adaptation for model-based RL. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HyxAfnA5tm`.

Gerhard Neumann, Wolfgang Maass, and Jan Peters. Learning complex motions by sequencing simpler motion templates. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 753–760, 2009.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In Peter Grünwald, Elad Hazan, and Satyen Kale (eds.), *Proceedings of The 28th Conference on Learning Theory*, volume 40 of *Proceedings of Machine Learning Research*, pp. 1376–1401, Paris, France, 03–06 Jul 2015. PMLR. URL `https://proceedings.mlr.press/v40/Neyshabur15.html`.

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.

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.

Xue Bin Peng, Glen Berseth, and Michiel Van de Panne. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 35(4):1–12, 2016.

Christian Perez, Felipe Petroski Such, and Theofanis Karaletsos. Generalized Hidden Parameter MDPs:Transferable Model-Based RL in a Handful of Trials. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):5403–5411, April 2020. doi: 10.1609/aaai.v34i04.5989. URL `https://ojs.aaai.org/index.php/AAAI/article/view/5989`.

Martin L Puterman. Markov decision processes: Discrete stochastic dynamic programming. *Journal of the Operational Research Society*, 1995.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

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

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 5331–5340, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Mark Bishop Ring et al. *Continual learning in reinforcement environments*. PhD thesis, University of Texas at Austin Austin, Texas 78712, 1994.

Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

J.J.E. Slotine and W. Li. *Applied Nonlinear Control*. Prentice Hall, 1991. ISBN 978-0-13-040890-7. URL https://books.google.com/books?id=cwpRAAAAMAAJ.

Shagun Sodhani, Sarath Chandar, and Yoshua Bengio. Toward Training Recurrent Neural Networks for Lifelong Learning. *Neural Computation*, 32(1):1–35, 01 2020. ISSN 0899-7667. doi: 10.1162/neco_a_01246. URL https://doi.org/10.1162/neco_a_01246.

Shagun Sodhani, Ludovic Denoyer, Pierre-Alexandre Kamienny, and Olivier Delalleau. MTEnv - environment interface for multi-task reinforcement learning. Github, 2021a. URL https://github.com/facebookresearch/mtenv.

Shagun Sodhani, Amy Zhang, and Joelle Pineau. Multi-task reinforcement learning with context-based representations. In *International Conference on Machine Learning (ICML)*, 2021b.

Jan Swevers, Chris Ganseman, D Bilgin Tukel, Joris De Schutter, and Hendrik Van Brussel. Optimal robot excitation and identification. *IEEE transactions on robotics and automation*, 13(5):730–740, 1997.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind control suite. Technical report, DeepMind, January 2018.

Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 4496–4506, 2017.

Sebastian Thrun. Lifelong learning algorithms. In *Learning to learn*, pp. 181–209. Springer, 1998.

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.

Peter Van Overschee and BL De Moor. *Subspace identification for linear systems: Theory—Implementation—Applications*. Springer Science & Business Media, 2012.

Manuel Watter, Jost Tobias Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. *arXiv preprint arXiv:1506.07365*, 2015.

Zhou Xian, Shamit Lal, Hsiao-Yu Tung, Emmanouil Antonios Platanios, and Katerina Fragkiadaki. Hyperdynamics: Generating expert dynamics models by observation. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=pHXfe1cOmA.

Annie Xie, James Harrison, and Chelsea Finn. Deep reinforcement learning amidst lifelong non-stationarity, 2020.

Huan Xu and Shie Mannor. Robustness and generalization. *Mach. Learn.*, 86(3):391–423, March 2012. ISSN 0885-6125. doi: 10.1007/s10994-011-5268-1. URL https://doi.org/10.1007/s10994-011-5268-1.

Ju Xu and Zhanxing Zhu. Reinforced continual learning. *arXiv preprint arXiv:1805.12369*, 2018.

Zhenjia Xu, Jiajun Wu, Andy Zeng, Joshua B Tenenbaum, and Shuran Song. Densephysnet: Learning dense physical object representations via multi-step dynamic interactions. *arXiv preprint arXiv:1906.03853*, 2019.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. 2019.

Haiyan Yin and Sinno Pan. Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Wenhao Yu, Jie Tan, C. Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. In *Robotics: Science and Systems*, 2017.

L Zadeh. On the identification problem. *IRE Transactions on Circuit Theory*, 3(4):277–281, 1956.

Amy Zhang, Zachary C. Lipton, Luis Pineda, Kamyar Azizzadenesheli, Anima Anandkumar, Laurent Itti, Joelle Pineau, and Tommaso Furlanello. Learning causal state representations of partially observable environments. *The Multi-disciplinary Conference on Reinforcement Learning and Decision Making*, 2019.

Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2021a. URL `https://openreview.net/forum?id=-2FCwDKRREu`.

Amy Zhang, Shagun Sodhani, Khimya Khetarpal, and Joelle Pineau. Learning robust state abstractions for hidden-parameter block MDPs. In *International Conference on Learning Representations*, 2021b. URL `https://openreview.net/forum?id=fmOOI2a3tQP`.

Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *European conference on computer vision*, pp. 94–108. Springer, 2014.

Tony Z. Zhao, Anusha Nagabandi, Kate Rakelly, Chelsea Finn, and Sergey Levine. Latent state models for meta-reinforcement learning from images. In *4th Annual Conference on Robot Learning, CoRL 2020, Proceedings*, Proceedings of Machine Learning Research. PMLR, 2020.

Wenxuan Zhou, Lerrel Pinto, and Abhinav Gupta. Environment probing interaction policies. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ryl8-3AcFX`.

Shaojun Zhu, Andrew Kimmel, Kostas E Bekris, and Abdeslam Boularias. Fast model identification via physics engines for data-efficient policy search. *arXiv preprint arXiv:1710.08893*, 2017.

Zhuangdi Zhu, Kaixiang Lin, and Jiayu Zhou. Transfer learning in deep reinforcement learning: A survey. *arXiv preprint arXiv:2009.07888*, 2020.

Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *International Conference on Machine Learning*, pp. 7693–7702. PMLR, 2019.

Karl Johan Åström and Torsten Bohlin. Numerical identification of linear dynamic systems from normal operating records. In *Proc. IFAC Conference on Self-Adaptive Control Systems*, 1965.

A Additional Related Work

This section extends the discussion on the related works from Section 2.

Some prior works train a **single fixed dynamics model** but introduce additional constraints that ensure the latent dynamics are locally linear (Watter et al., 2015; Banijamali et al., 2018) or that the learned dynamics model is invariant to translation dynamics (Fragkiadaki et al., 2015). However, such approaches fail when the parameters of the underlying dynamics model change. In theory, one could learn a new *expert* dynamics model per task, but that is computationally expensive and requires access to the unseen environments. Xian et al. (2021) generates the weights of an *expert* dynamics model by using the environment context as an input. Specifically, they proposed an algorithm called HyperDynamics where they use a HyperNetwork (Ha et al., 2016; Chang et al., 2019; Klocek et al., 2019; Meyerson & Miikkulainen, 2019) to generate the weights of the dynamics model.Similar to our work, Lee et al. (2020) also introduced additional loss terms when training the agent. However, their objective is to encourage the context encoding to be useful for predicting both forward (next state) and backward (previous state) dynamics while being temporally consistent. Asadi et al. (2018) present results when bounding the model class to Lipschitz functions but assume that the given MDP is Lipschitz. We instead show that this constraint can be placed on the representation learning objective for better results in the continual learning setting, even when the original BC-MDP is not Lipschitz.

Our work is related to the broad area of **multitask RL and transfer learning** (Caruana, 1997; Zhang et al., 2014; Kokkinos, 2017; Radford et al., 2019; Rajeswaran et al., 2016). Previous works have looked at multi-task and transfer learning from the perspective of policy transfer (Rusu et al., 2015; Yin & Pan, 2017), policy reuse (Fernández & Veloso, 2006; Barreto et al., 2016), representation transfer (Rusu et al., 2016; Devin et al., 2017; Andreas et al., 2017; Sodhani et al., 2021b) etc. One unifying theme in these works is that the agent is finetuned on the new environment while we focus on setup where there are no gradient updates when evaluating on the unseen environment. Zhang et al. (2021b) also uses task metrics to learn a context space, but focus on the rich observation version of the Hidden-Parameter MDP (HiP-MDP) setting (Doshi-Velez & Konidaris, 2013) with access to task ids. Similarly, Perez et al. (2020) also treats the multi-task setting as a HiP-MDP by explicitly designing latent variable models to model the latent parameters, but require knowledge of the structure upfront, whereas our approach does not make any such assumptions.

Several works proposed a *mixture-of-experts* based approach to adaptation where an expert model is learned for each task (Doya et al., 2002; Neumann et al., 2009; Peng et al., 2016) or experts are shared across tasks (Igl et al., 2020; Sodhani et al., 2021b). Teh et al. (2017) additionally proposed to distill these experts in a single model that should work well across multiple tasks. While these systems perform reasonably well on the training tasks, they do not generalize well to unseen tasks.

B Implementation Details

We provide additional details of the main algorithm, environment setup, compute used, baselines, and hyperparameter information.

B.1 Algorithm

We provide pseudocode for the main algorithm in Algorithm 1 and the context loss update in Algorithm 2.

Algorithm 2 Update Using Context Loss

Require: Batches of data sampled from the Replay Buffer $\mathcal{D}$, Dynamics Model T, Reward Model R, Observation Encoder ϕ, Context Encoder ψ, Learning rates α_ψ, α_T and α_R.

1: **for** each batch **do**
2: Compute $\mathcal{L}(\phi, \psi, T, R))$ using Equation (2)
3: $\psi \leftarrow \psi - \alpha_\psi \nabla_\theta \sum_i \mathcal{L}$
4: $T \leftarrow T - \alpha_T \nabla_\theta \sum_i \mathcal{L}$
5: $R \leftarrow R - \alpha_R \nabla_\theta \sum_i \mathcal{L}$
6: **end for**

Algorithm 1 Training ZeUS algorithm.

Require: Actor, Critic, Dynamics Model T, Reward Model R, Observation Encoder ϕ, Context Encoder ψ, Replay Buffer $\mathcal{D}$.

for each timestep $t = 1..T$ **do**
  for each $\mathcal{E}_i$ **do**
    $a_t^i \sim \pi^i(\cdot|s_t^i)$
    $s_t'^i \sim p^i(\cdot|s_t^i, a_t^i)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup (s_t^i, a_t^i, r(s_t^i, a_t^i), s_t'^i)$
    UPDATE_CRITIC($\mathcal{D}$)
    UPDATE_ACTOR($\mathcal{D}$)
    UPDATE_USING_CONTEXT_LOSS($\mathcal{D}$)
  end for
end for

B.2 SETUP

Environments For tasks with varying dynamics, we use the Finger, Cheetah and Walker environments. These are Mujoco (Todorov et al., 2012)[6] based tasks that are available as part of the DMControl Suite Tassa et al. (2018)[7]. We use MTEnv Sodhani et al. (2021a)[8] to interface with the environments. In the Finger Spin task, the agent has to manipulate (continually rotate) a planar *finger*. We vary the size across different tasks and the specific values (for each mode) are listed in Table 2. In the Cheetah Run task, a planar bipedal cheetah has to run as fast as possible (up to a maximum velocity). In the Walker Walk tasks, a planar walker has to learn to walk. We consider two cases here: (i) varying the friction coefficient between the walker and the ground, and (ii) the length of the foot of the walker.

For tasks with varying reward function, we use the Cheetah and the Sawyer Peg environments from Zhao et al. (2020)[9]. We highlight some challenges in the evaluation: In the cheetah environment, the reward depends on the difference in the magnitude of the agent's velocity and the target velocity. In each run of the algorithm, the target velocities are randomly sampled. We noticed that the cheetah can easily match the target velocity when the target velocity is small and makes a larger error when the target velocity is large. In practice, the performance of the cheetah (as measured in terms of episodic rewards) can vary a lot depending on the sampled target velocities. To make the comparison fair across the different baselines, we fix the values of the target velocities (for train and for eval) instead of sampling them. In the Peg-Insertion task, the reward has an extra "offset" term as shown in the implementation but this does not match the description in Zhao et al. (2020). We use the version of reward function without the offset, as described in Zhao et al. (2020).

All the algorithms are implemented using PyTorch (Paszke et al., 2017)[10]. We use the MTRL codebase[11] as the starting codebase.

Compute Resources Unless specified otherwise, all the experiments are run using Volta GPUs with 16 GB memory. Each experiment uses 1 GPU and 10 CPUs (to parallelize the execution in the environments). The experiments with HyperDynamics model are run with Volta GPUs with 32 GB Memory. For the Cheetah-Run-v0, Finger-Spin-v0, Walker-Walk-v0 and Walker-Walk-v1, training CADM and ZeUS takes about 44 hours, training HyperDynamics takes about 62 hours and training UP-OSI takes 24 hours (all for 1.2 M steps). For Cheetah-Run-v1 and Sawyer-Peg-v1, training CADM and ZeUS takes about 25 hours while HyperDynamics takes about 32.5 hours (all for 300K steps). These times include the time for training and evaluation.

[6] https://www.roboti.us
[7] https://github.com/deepmind/dm_control
[8] https://github.com/facebookresearch/mtenv/commit/7fdec15f7e842bce4c17f4f3328d9d6fdc79d7fc
[9] https://github.com/tonyzhaozh/meld
[10] https://pytorch.org/
[11] https://github.com/facebookresearch/mtrl/commit/eea3c99cc116e0fadc41815d0e7823349fcc0bf4

In Table 1, we show that a two-layer feedforward network can be trained to infer the context value from last 5 observation-action tuples. The setup is modeled as a regression problem where the observation-action transition tuples are obtained from a random policy.

Table 1: Loss when training the model to infer the context

Environment Name	Loss
Cheetah-Run-v0	5.12×10^{-5}
Finger-Spin-v0	7.45×10^{-5}
Walker-Walk-v0	7.22×10^{-5}
Walker-Walk-v1	6.41×10^{-5}

B.2.1 BASELINES

We provide additional implementation related details for the baselines. For a summary of the baselines, refer Section 6.2.

1. *Context-aware Dynamics Model (CaDM)*: Lee et al. (2020) proposed a two stage pipeline: (a) use a context encoder to obtain a context vector given the recent interactions and (ii) perform online adaption by conditioning the forward dynamics model on the context vector. CaDM is shown to outperform vanilla dynamics models which do not use the interaction history, stacked dynamics models which use the interaction history as an input, and Gradient and Recurrence-based meta learning approaches (Nagabandi et al., 2018).
2. *UP-OSI*: Yu et al. (2017) proposed learning two components: (i) Universal Policy (UP) and (ii) On-line System Identification model (OSI). The universal policy is trained over a wide range of dynamics parameters so that it can operate in a previously unseen environment (given access to the parameters of the dynamics model). It is modeled as a function of the dynamic parameters μ, such that $a_t = \pi_\mu(s_t)$ and is trained offline, in simulation, without requiring access to the real-world samples. The goal of the universal policy is to generalize to the space of the dynamics models. The on-line system identification model is trained to identify the parameters of the dynamics model conditioned on the last k state-action transition tuples i.e. $\mu = \phi((s_i, a_i, s'_i, r_i), \forall i \in \{1, ..., k\})$, where ϕ is the OSI module. ϕ is trained using a supervised learning loss by assuming access to true parameters of the dynamics model. During evaluation (in an unseen environment), the OSI module predicts the dynamic parameters at every timestep. The universal policy uses these inferred dynamics parameters to predicts the next action.The system identification module is trained to identify a model that is good enough to predict a policy's value function that is similar to the current optimal policy. Following the suggestion by (Yu et al., 2017), we initially train the OSI using the data collected by UP (when using true model parameters) and then switch to the data collected using inferred parameters. Specifically, the paper suggested using first 3-5 iterations (out of 500 iterations) for training with the ground truth parameters. We scaled the number of these warmup iterations to match the number of updates in our implementation and report the results using the same. During evaluation, we report the performance of UP using the inferred parameters (UP-*inferred*), as recommended in the paper.
3. *HyperDynamics*: Xian et al. (2021) proposed learning three components: (i) an encoding module that encodes the agent-environment interactions, (ii) a hypernetwork that generates the weight for a dynamics model at the current timestep, and (iii) a (target) dynamics model that uses the weights generated by the hypernetwork. All the components (and the policy) are trained jointly in an end-to-end manner. HyperDynamics is shown to ourperform both ensemble of experts and meta-learning based approaches (Nagabandi et al., 2018).
4. *Context-conditioned Policy*: We train a context encoder that encoders the interaction history into latent context that is passed to the policy and the dynamics. This approach can be thought of as an ablation of the ZeUS algorithm with $\alpha_\psi = 0$, or without the context learning error (from Equation (2)). We label this baseline as *Zeus-no-context-loss*.

We note that when training on the environments with varying reward dynamics, we concatenate the reward along with the environment observation (as done in Zhao et al. (2020)).

B.3 Environment Details

Table 2: Parameter values for different modes for the Finger environments (Finger-Spin-v0) when varying the size of the finger across the tasks.

Mode	Values
Train / Eval	[0.05, 0.0625, 0.075, 0.0875, 0.15, 0.1625, 0.175, 0.1875]
Eval Interpolation	[0.1, 0.1125, 0.125, 0.1375]
Eval Extrapolation	[0.01, 0.0125, 0.025, 0.0375, 0.2, 0.2125, 0.35, 0.5]

Table 3: Parameter values for different modes for the Cheetah environments (Cheetah-Run-v0) when varying the length of cheetah's torso across the tasks.

Mode	Values
Train / Eval	[0.6, 0.65, 0.7, 0.75, 1, 1.05, 1.1, 1.15, 1.25, 1.4, 1.5]
Eval Interpolation	[0.8, 0.85, 0.9, 0.95]
Eval Extrapolation	[0.3, 0.4, 0.5, 0.55, 1.2, 1.25, 1.4, 1.5]

Table 4: Parameter values for different modes for the Walker environments (Walker-Walk-v0) when varying the friction coefficient between the walker and the ground.

Mode	Values
Train / Eval	[0.8, 0.85, 0.9, 0.95, 1.2, 1.25, 1.3, 1.35]
Eval Interpolation	[1.0, 1.05, 1.1, 1.15]
Eval Extrapolation	[0.3, 0.5, 0.7, 0.75, 1.4, 1.45, 1.7, 1.9]

Table 5: Parameter values for different modes for the Walker environments (Walker-Walk-v1) when varying the length of walker's foot.

Mode	Values
Train / Eval	[0.09, 0.095, 0.1, 0.105, 0.13, 0.135, 0.14, 0.145]
Eval Interpolation	[0.11, 0.115, 0.12, 0.125]
Eval Extrapolation	[0.03, 0.06, 0.08, 0.085, 0.15, 0.155, 0.18, 0.21]

Table 6: Values for the target velocity for the Cheetah environments (Cheetah-Run-v1).

Mode	Values
Train	[0., 0.43, 0.86, 1.29, 1.71, 2.14, 2.57, 3.0]
Eval	[0.4, 0.78, 1.11, 1.47, 1.83, 2.18, 2.55, 2.9]

Table 7: Range for sampling the values for the Sawyer-Peg environments (Sawyer-Peg-v0)

Mode	Values
x_range_1	(0.44, 0.45)
x_range_2	(0.6, 0.61)
y_range_1	(-0.08, -0.07)
y_range_2	(0.07, 0.08)

B.4 License

1. Mujoco: Commercial (with Trial Version) https://www.roboti.us/license.html

2. DeepMind Suite: Apache https://github.com/deepmind/dm_control/blob/master/LICENSE
3. MTEnv: MIT License https://github.com/facebookresearch/mtenv/blob/main/LICENSE
4. Meld Codebase: https://github.com/tonyzhaozh/meld
5. PyTorch: https://github.com/pytorch/pytorch/blob/master/LICENSE
6. MTRL: MIT License https://github.com/facebookresearch/mtrl/blob/main/LICENSE
7. Hydra: MIT License https://github.com/facebookresearch/hydra/blob/master/LICENSE

B.5 Hyperparameter Details

In this section, we provide hyper-parameter values for each of the methods in our experimental evaluation. We also describe the search space for each hyperparameter. In Table 9 and Table 8 , we provide the hyperparameter values that are common across all the methods.

Figure 7: Variation in `Cheetah-Run-v0` tasks.

Table 8: Hyperparameter values that are common across all the methods for Cheetah-Run-v0, Finger-Spin-v0, Walker-Walk-v0 and Walker-Walk-v1 (envs with varying dynamics)

Hyperparameter	Hyperparameter values
batch size (per task)	128
network architecture	feedforward network
non-linearity	ReLU
policy initialization	standard Gaussian
exploration parameters	run a uniform exploration policy 1500 steps
# of samples / # of train steps per iteration	1 env step / 1 training step
policy learning rate	3e-4
Q function learning rate	3e-4
Critic update frequency	2
optimizer	Adam
beta for Adam optimizer for policy	(0.9, 0.999)
Q function learning rate	3e-4
beta for Adam optimizer for Q function	(0.9, 0.999)
Discount	.99
Episode length (horizon)	500
Reward scale	1.0
actor update frequency	2
actor log stddev bounds	$[-10, 2]$
number of layers in actor/critic	2
actor/critic hidden dimension	1024
number layers in dynamics model	1
dynamics hidden dimension	512
number of encoder layers	4
number of filters in encoder	32
Replay buffer capacity	400000
Temperature Adam's β_1	0.9
Init temperature	0.1
Context Length	5

Table 9: Hyperparameter values that are common across all the methods for Cheetah-Run-v1 and Sawyer-Peg-v0 (envs with varying reward functions)

Hyperparameter	Hyperparameter values
batch size (per task)	128
network architecture	feedforward network
non-linearity	ReLU
policy initialization	standard Gaussian
exploration parameters	run a uniform exploration policy 10000 steps
# of samples / # of train steps per iteration	1 env step / 1 training step
policy learning rate	3e-4
Q function learning rate	3e-4
Critic update frequency	2
optimizer	Adam
beta for Adam optimizer for policy	(0.5, 0.999)
Q function learning rate	3e-4
beta for Adam optimizer for Q function	(0.5, 0.999)
Discount	.99
Episode length (horizon)	40
Reward scale	1.0
actor update frequency	2
actor log stddev bounds	$[-10, 2]$
number of layers in actor/critic	2
actor/critic hidden dimension	1024
number layers in dynamics model	1
dynamics hidden dimension	512
number of encoder layers	4
number of filters in encoder	32
Replay buffer capacity	400000
Temperature Adam's β_1	0.5
Init temperature	0.1
Context Length	5

Table 10: Hyperparameter values for Context Aware Dynamics Model

Hyperparameter	Hyperparameter values	Environment
β	0.5	Cheetah-Run-v0
β	0.5	Finger-Spin-v0
β	0.5	Walker-Walk-v0
β	0.5	Walker-Walk-v1
β	0.5	Cheetah-Run-v1
β	0.5	Sawyer-Peg-v0

Table 11: Hyperparameter values for ZeUS

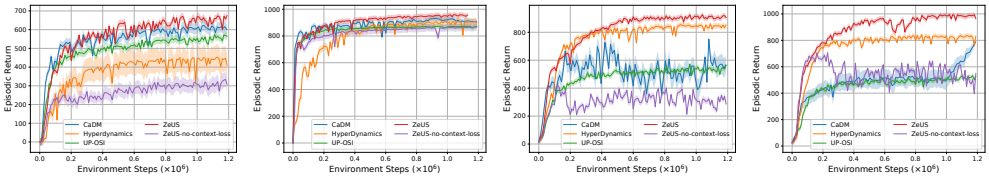
Hyperparameter	Hyperparameter values	Environment
α_ψ	1.0	Cheetah-Run-v0
α_ψ	0.5	Finger-Spin-v0
α_ψ	2.0	Walker-Walk-v0
α_ψ	0.1	Walker-Walk-v1
α_ψ	1.0	Cheetah-Run-v1
α_ψ	1.0	Sawyer-Peg-v0

C ADDITIONAL RESULTS

We present additional results not in the main paper.

C.1 HANDLING NONSTATIONARITY IN THE TRAINING ENVIRONMENTS

In Figure 8 we show performance on ZeUS and baselines on the training environments.

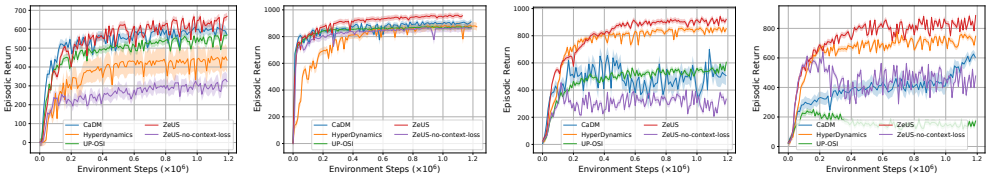


(a) Cheetah-Run-v0 (b) Finger-Spin-v0 (c) Walker-Walk-v0 (d) Walker-Walk-v1

Figure 8: We compare the performance of the proposed ZeUS algorithm with *CaDM*, *UP-OSI*, *HyperDynamics* and *ZeUS-no-context-loss* algorithms on the training environments for four families of tasks with different dynamics parameters.

C.2 HANDLING NONSTATIONARITY IN THE INTERPOLATION ENVIRONMENTS

In Figure 9 we show performance on ZeUS and baselines on evaluation environments that are interpolated from the train environments.

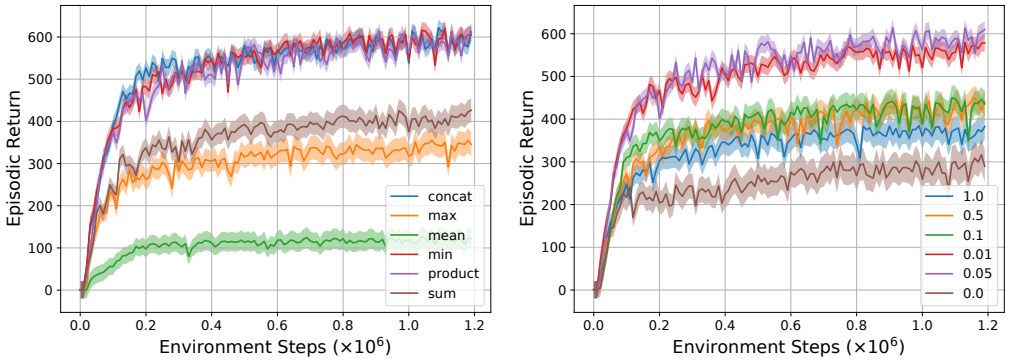


(a) Cheetah-Run-v0 (b) Finger-Spin-v0 (c) Walker-Walk-v0 (d) Walker-Walk-v1

Figure 9: We compare the performance of the proposed ZeUS algorithm with *CaDM*, *UP-OSI*, *HyperDynamics* and *ZeUS-no-context-loss* algorithms on the evaluation environments (interpolation) for four families of tasks with different dynamics parameters.

C.3 ADAPTING AND GENERALIZING TO ENVIRONMENTS WITH VARYING REWARD FUNCTIONS

In Figure 10 we show performance of ZeUS over a hyperparameter sweep for different values of aggregation operator and values of α_ψ for the Cheetah-run-v0 setup.



(a) Varying the aggregation operator in the context encoder

(b) Varying values of α_ψ

Figure 10: Performance of ZeUS on Cheetah-run-v0 task with varying values of the aggregation operator in the context enocder (in (a)) and varying values of α_ψ in (b).

D Additional Theoretical Background

Bisimulation is a strict form of state abstraction, where two states are bisimilar if they are behaviorally equivalent. **Bisimulation metrics** (Ferns et al., 2011) define a distance between states as follows:

Definition 6 (Bisimulation Metric (Theorem 2.6 in Ferns et al. (2011))). *Let $(\mathcal{S}, \mathcal{A}, T, r)$ be a finite MDP and $\mathfrak{met}$ the space of bounded pseudometrics on $\mathcal{S}$ equipped with the metric induced by the uniform norm. Define $F : \mathfrak{met} \mapsto \mathfrak{met}$ by*

$$F(d)(s, s') = \max_{a \in \mathcal{A}} (|r(s, a) - r(s', a)| + \gamma W(d)(T_s^a, T_{s'}^a)),$$

where $W(d)$ is the Wasserstein distance between transition probability distributions. Then F has a unique fixed point $\tilde{d}$ which is the bisimulation metric.

A nice property of this metric $\tilde{d}$ is that difference in optimal value between two states is bounded by their distance as defined by this metric. Zhang et al. (2021a) use bisimulation metrics to learn a representation space that is Lipschitz with respect to the MDP dynamics in the single task setting.

Why is the bisimulation metric useful? It turns out that the optimal value function has the nice property of being smooth with respect to this metric.

Lemma 1 (V^* is Lipschitz with respect to $\tilde{d}$ (Ferns et al., 2004)). *Let V^* be the optimal value function for a given discount factor γ. Then V^* is Lipschitz continuous with respect to $\tilde{d}$ with Lipschitz constant $\frac{1}{1-\gamma}$,*

$$|V^*(s) - V^*(s')| \leq \frac{1}{1-\gamma} \tilde{d}(s, s').$$

Proof in (Ferns et al., 2004). Therefore, we see that bisimulation metrics give us a Lipschitz value function with respect to $\tilde{d}$ with a Lipschitz constant $\frac{1}{1-\gamma}$.

E Additional Theoretical Results and Proofs

Theorem 1. *Let V^* be the optimal, universal value function for a given discount factor γ and context space $\mathcal{C}$. Then V^* is Lipschitz continuous with respect to d_{task} with Lipschitz constant $\frac{1}{1-\gamma}$ for any $s \in \mathcal{S}$,*

$$|V^*(s, c) - V^*(s, c')| \leq \frac{1}{1-\gamma} d_{task}(c, c').$$

Proof. We construct a super-MDP $\mathcal{M}'$ by concatenating context and state spaces into a new state space $\mathcal{S}' : \mathcal{C} \times \mathcal{S}$. We can apply Lemma 1 to $\mathcal{M}'$ for any $s \in \mathcal{S}, c, c' \in \mathcal{C}$:

$$|V^*_{\mathcal{M}'}((s,c)) - V^*_{\mathcal{M}'}((s,c'))| \leq \frac{1}{1-\gamma}\tilde{d}_{\mathcal{M}'}((s,c),(s,c')). \tag{3}$$

By Definition 6 and Definition 4 we know that

$$\tilde{d}_{\mathcal{M}'}((s,c),(s,c')) \leq d_{\text{task}}(c,c').$$

So we can substitute the right hand side into Equation (3),

$$|V^*_{\mathcal{M}'}((s,c)) - V^*_{\mathcal{M}'}((s,c'))| \leq \frac{1}{1-\gamma}d_{\text{task}}(c,c').$$

Which is equivalent to the following statement:

$$|V^*(s,c) - V^*(s,c')| \leq \frac{1}{1-\gamma}d_{\text{task}}(c,c').$$

□

E.1 Value and Transfer Bounds

In this section, we provide value bounds and sample complexity analysis of the ZeUS approach. This analysis is similar to the one done in Zhang et al. (2021b), which focused on a multi-environment setting with different dynamics but same task. We first define three additional error terms associated with learning a $\epsilon_R, \epsilon_T, \epsilon_c$-bisimulation abstraction,

$$\begin{aligned}
\epsilon_R &:= \sup_{\substack{a\in\mathcal{A},\\ o_1,o_2\in\mathcal{O},\phi(o_1)=\phi(o_2)}} \big|R(o_1,a) - R(o_2,a)\big|,\\
\epsilon_T &:= \sup_{\substack{a\in\mathcal{A},\\ o_1,o_2\in\mathcal{O},\phi(o_1)=\phi(o_2)}} \big\|\Phi T(o_1,a) - \Phi T(o_2,a)\big\|_1,\\
\epsilon_c &:= \|\hat{c} - c\|_1.
\end{aligned}$$

ϵ_R and ϵ_T are intra-context constants and ϵ_c is an inter-context constant. ΦT denotes the *lifted* version of T, where we take the next-step transition distribution from observation space $\mathcal{O}$ and lift it to latent space $\mathcal{S}$. We can think of ϵ_R, ϵ_T as describing a new MDP which is close – but not necessarily the same, if $\epsilon_R, \epsilon_T > 0$ – to the original MDP. These two error terms can be computed empirically over all training environments and are therefore not task-specific. ϵ_c, on the other hand, is measured as a per-task error. Similar methods are used in Jiang et al. (2015) to bound the loss of a single abstraction, which we extend to the BC-MDP setting with a family of tasks.

Value Bounds. We first look at the single, fixed context setting, which can be thought of as the single-task version of the BC-MDP. We can compute approximate error bounds in this setting by denoting ϕ an (ϵ_R, ϵ_T)-approximate bisimulation abstraction, where

$$\begin{aligned}
\epsilon_R &:= \sup_{\substack{a\in\mathcal{A},\\ o_1,o_2\in\mathcal{O},\phi(o_1)=\phi(o_2)}} \big|R(o_1,a) - R(o_2,a)\big|,\\
\epsilon_T &:= \sup_{\substack{a\in\mathcal{A},\\ o_1,o_2\in\mathcal{O},\phi(o_1)=\phi(o_2)}} \big\|\Phi T(o_1,a) - \Phi T(o_2,a)\big\|_1,\\
\epsilon_c &:= \|\hat{c} - c\|_1.
\end{aligned}$$

ΦT denotes the *lifted* version of T, where we take the next-step transition distribution from observation space $\mathcal{O}$ and lift it to latent space $\mathcal{S}$.

Lemma 2. *Given an MDP $\bar{\mathcal{M}}$ built on a (ϵ_R, ϵ_T)-approximate bisimulation abstraction of Block MDP $\mathcal{M}$, we denote the evaluation of the optimal Q function of $\bar{\mathcal{M}}$ on $\mathcal{M}$ as $[Q^*_{\bar{\mathcal{M}}}]_{\mathcal{M}}$. The value difference with respect to the optimal $Q^*_{\mathcal{M}}$ is upper bounded by*

$$\big\|Q^*_{\mathcal{M}} - [Q^*_{\bar{\mathcal{M}}}]_{\mathcal{M}}\big\|_\infty \leq \epsilon_R + \gamma\epsilon_T\frac{R_{max}}{2(1-\gamma)}.$$

Proof. From Lemma 3 in Jiang et al. (2015). □

We now evaluate how the error in c prediction and the learned bisimulation representation affect the optimal $Q^*_{\bar{\mathcal{M}}_{\hat{c}}}$ of the learned MDP, by first bounding its distance from the optimal Q^* of the true MDP for a single-task.

Lemma 3 (Q error)**.** *Given an MDP $\bar{\mathcal{M}}_{\hat{c}}$ built on a $(\epsilon_R, \epsilon_T, \epsilon_c)$-approximate bisimulation abstraction of an instance of a HiP-BMDP $\mathcal{M}_c$, we denote the evaluation of the optimal Q function of $\bar{\mathcal{M}}_{\hat{c}}$ on $\mathcal{M}$ as $[Q^*_{\bar{\mathcal{M}}_{\hat{c}}}]_{\mathcal{M}_c}$. The value difference with respect to the optimal $Q^*_{\mathcal{M}}$ is upper bounded by*

$$\left\|Q^*_{\mathcal{M}_c} - [Q^*_{\bar{\mathcal{M}}_{\hat{c}}}]_{\mathcal{M}_c}\right\|_\infty \leq \epsilon_R + \gamma(\epsilon_T + \epsilon_c)\frac{R_{max}}{2(1-\gamma)}.$$

Proof. In the BC-MDP setting, we have a global encoder ϕ over all tasks, but the different transition distributions and reward functions condition on the context c. We now must incorporate difference in dynamics in ϵ_T and reward in ϵ_R. Assuming we have two environments with hidden parameters c_i, c_j, we can compute $\epsilon_T^{c_i,c_j}$ and $\epsilon_R^{c_i,c_j}$ across those two environments by joining them into a super-MDP.: For $\epsilon_T^{c_i,c_j}$:

$$\begin{aligned}
\epsilon_T^{c_i,c_j} &= \sup_{\substack{a\in\mathcal{A},\\ o_1,o_2\in\mathcal{O},\phi(o_1)=\phi(o_2)}} \left\|\Phi T_{c_i}(o_1,a) - \Phi T_{c_j}(o_2,a)\right\|_1 \\
&\leq \sup_{\substack{a\in\mathcal{A},\\ o_1,o_2\in\mathcal{O},\phi(o_1)=\phi(o_2)}} \left(\left\|\Phi T_{c_i}(o_1,a) - \Phi T_{c_i}(o_2,a)\right\|_1 + \left\|\Phi T_{c_i}(o_2,a) - \Phi T_{c_j}(o_2,a)\right\|_1\right) \\
&\leq \sup_{\substack{a\in\mathcal{A},\\ o_1,o_2\in\mathcal{O},\phi(o_1)=\phi(o_2)}} \left\|\Phi T_{c_i}(o_1,a) - \Phi T_{c_i}(o_2,a)\right\|_1 + \sup_{\substack{a\in\mathcal{A},\\ o_1,o_2\in\mathcal{O},\phi(o_1)=\phi(o_2)}} \left\|\Phi T_{c_i}(o_2,a) - \Phi T_{c_j}(o_2,a)\right\|_1
\end{aligned}$$

For $\epsilon_R^{c_i,c_j}$ it is much the same:

$$\begin{aligned}
\epsilon_R^{c_i,c_j} &= \sup_{\substack{a\in\mathcal{A},\\ o_1,o_2\in\mathcal{O},\phi(o_1)=\phi(o_2)}} \left|R_{c_i}(o_1,a) - R_{c_j}(o_2,a)\right| \\
&\leq \sup_{\substack{a\in\mathcal{A},\\ o_1,o_2\in\mathcal{O},\phi(o_1)=\phi(o_2)}} \left(\left|R_{c_i}(o_1,a) - R_{c_i}(o_2,a)\right| + \left|R_{c_i}(o_2,a) - R_{c_j}(o_2,a)\right|\right) \\
&\leq \sup_{\substack{a\in\mathcal{A},\\ o_1,o_2\in\mathcal{O},\phi(o_1)=\phi(o_2)}} \left|R_{c_i}(o_1,a) - R_{c_i}(o_2,a)\right| + \sup_{\substack{a\in\mathcal{A},\\ o_1,o_2\in\mathcal{O},\phi(o_1)=\phi(o_2)}} \left|R_{c_i}(o_2,a) - R_{c_j}(o_2,a)\right|
\end{aligned}$$

Putting these together we get:

$$\epsilon_T^{c_i,c_j} + \epsilon_R^{c_i,c_j} \leq \epsilon_T + \epsilon_R + \|c_i - c_j\|_1$$

This result is intuitive in that with a shared encoder learning a per-task bisimulation relation, the distance between bisimilar states from another task depends on the change in transition distribution between those two tasks. We can now extend the single-task bisimulation bound (Lemma 2) to the BC-BMDP setting by denoting approximation error of c as $\|c - \hat{c}\|_1 < \epsilon_c$. □

We can measure the generalization capability of a specific policy π learned on one task to another, now taking into account error from the learned representation.

Theorem 3. *Given two MDPs $\mathcal{M}_{c_i}$ and $\mathcal{M}_{c_j}$, we can bound the difference in Q^π between the two MDPs for a given policy π learned under an $\epsilon_R, \epsilon_T, \epsilon_{c_i}$-approximate abstraction of $\mathcal{M}_{c_i}$ and applied to*

$$\left\|Q^*_{\mathcal{M}_{c_j}} - [Q^*_{\bar{\mathcal{M}}_{\hat{c}_i}}]_{\mathcal{M}_{c_j}}\right\|_\infty \leq \epsilon_R + \gamma\big(\epsilon_T + \epsilon_{c_i} + \|c_i - c_j\|_1\big)\frac{R_{max}}{2(1-\gamma)}.$$

This result clearly follows directly from Lemma 3. Given a policy learned for task i, Theorem 3 gives a bound on how far from optimal that policy is when applied to task j. Intuitively, the more similar in behavior tasks i and j are, as denoted by $\|c_i - c_j\|_1$, the better π performs on task j.