Deep Reinforcement Learning amidst Lifelong Non-Stationarity

Anonymous Authors[1]

Abstract

As humans, our goals and our environment are persistently changing throughout our lifetime based on our experiences, actions, and internal and external drives. In contrast, typical reinforcement learning problem set-ups consider decision processes that are stationary across episodes. Can we develop reinforcement learning algorithms that can cope with the persistent change in the former, more realistic problem settings? While on-policy algorithms such as policy gradients in principle can be extended to non-stationary settings, the same cannot be said for more efficient off-policy algorithms that replay past experiences when learning. In this work, we formalize this problem setting, and draw upon ideas from the online learning and probabilistic inference literature to derive an off-policy RL algorithm that can reason about and tackle such lifelong non-stationarity. Our method leverages latent variable models to learn a representation of the environment from current and past experiences, and performs off-policy RL with this representation. We further introduce several simulation environments that exhibit lifelong non-stationarity, and empirically find that our approach substantially outperforms approaches that do not reason about environment shift.

1. Introduction

In the standard reinforcement learning (RL) set-up, the agent is assumed to operate in a stationary environment with access to episodic resets. However, this assumption rarely holds in more realistic settings, especially in the context of lifelong learning systems (Thrun, 1998). That is, over the course of an agent's lifetime, it may be subjected to environment dynamics and rewards that vary with time. In robotics applications for example, this non-stationarity can manifest itself in changing terrains as the robot navigates to new, previously unexplored regions. In other situations, not even the objective is necessarily fixed: consider an assistive robot helping a human whose preferences gradually change over time. Since stationarity is a core assumption in many existing RL algorithms, they are unlikely to perform well in these environments.

Crucially, in each of the above scenarios, the environment is specified by unknown, time-varying parameters. These latent parameters are also not i.i.d., e.g., if the sky is clear at this very moment, it likely will not suddenly start raining in the next; in other words, these parameters have associated but unobserved dynamics. In this paper, we formalize this problem setting with the dynamic parameter Markov decision process (DP-MDP). The DP-MDP corresponds to a sequence of stationary MDPs, related through a set of latent parameters governed by an autonomous dynamical system. While all non-stationary MDPs are special instances of the partially observable Markov decision process (POMDP) (Kaelbling et al., 1998), in this setting, we can leverage structure available in the dynamics of the hidden parameters and avoid solving general POMDPs.

On-policy RL algorithms can in principle cope with such non-stationarity (Sutton et al., 2007). However, in highly dynamic environments, only a limited amount of interaction is permitted before the environment shifts, and on-policy methods may fail to adapt rapidly enough in this low-shot setting (Al-Shedivat et al., 2017). Instead, we desire an off-policy RL algorithm that can use past experience both to improve sample efficiency and to reason about the environment dynamics. For the agent to adapt with the environment, it needs the ability to predict how the MDP parameters will shift. We thus desire a representation of the MDP as well as a model of how the parameters evolve in this space, both of which can be learned from off-policy experience.

To this end, our core contribution is an off-policy RL algorithm that can operate under non-stationarity by jointly learning (1) a latent variable model, which lends a compact representation of the MDP, and (2) a maximum entropy policy with this representation. We validate our approach, which we call **Li**felong **L**atent **A**ctor-**C**ritic (LILAC), on a set of simulated environments that demonstrate persis-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

tent non-stationarity, including a navigation task in an unbounded, non-episodic environment. In our experimental evaluation, we find that our method far outperforms RL algorithms that do not account for environment dynamics.

2. Dynamic Parameter Markov Decision Processes

The standard RL setting assumes episodic interaction with a fixed MDP (Sutton & Barto, 2018). While this setting enables learning in highly structured environments, it is limited in expressivity due to the core assumption of fully observed, Markovian dynamics. In the real world, the assumption of episodic interaction with identical MDPs is limiting as it does not capture the wide variety of exogenous factors that may effect the decision-making problem. A common model to avoid the strict assumption of Markovian observations is the partially observed MDP (POMDP) formulation (Kaelbling et al., 1998). While the POMDP is highly general, we focus in this work on leveraging known structure of the non-stationary MDP to improve performance. In particular, we consider an episodic environment, which we call the *dynamic parameter MDP* (DP-MDP), where a new MDP (we also refer to MDPs as tasks) is presented in each episode. In reflection of the regularity of real-world non-stationarity, the tasks are sequentially related through a set of continuous parameters.

Formally, the DP-MDP is equipped with state space $\mathcal{S}$, action space $\mathcal{A}$, and initial state distribution $\rho_{\mathbf{s}}(\mathbf{s}_1)$. Following the formulation of the Hidden Parameter MDP (HiP-MDP) (Doshi-Velez & Konidaris, 2016), a set of *unobserved* task parameters $\mathbf{z} \in \mathcal{Z}$ defines the dynamics $p_{\mathbf{s}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t; \mathbf{z})$ and reward function $r(\mathbf{s}_t, \mathbf{a}_t; \mathbf{z})$ for each task. In contrast to the HiP-MDP, the task parameters $\mathbf{z}$ in the DP-MDP are not sampled i.i.d. but instead shift stochastically according to $p_{\mathbf{z}}(\mathbf{z}^{i+1}|\mathbf{z}^i)$, with initial distribution $\rho_{\mathbf{z}}(\mathbf{z}^1)$. In other words, the DP-MDP is a sequence of tasks with parameters determined by the transition function $p_{\mathbf{z}}$. The DP-MDP can be also viewed as a hidden Markov model wherein the state represents the MDP in each episode and observations of the hidden Markov model are given in the form of trajectories collected by the agent. If the task parameters $\mathbf{z}$ for each episode were known, the augmented state space $\mathcal{S} \times \mathcal{Z}$ would define a fully observable MDP for which we can use standard RL algorithms. Hence, in our approach, we aim to infer the hidden task parameters and learn their transition function, allowing us to leverage existing RL algorithms by augmenting the observations with the inferred task parameters.

3. Preliminaries: RL as Inference

We first discuss an established connection between probabilistic inference and reinforcement learning (Toussaint, 2009; Levine, 2018) to provide some context for our approach. At a high level, this framework casts sequential decision-making as a probabilistic graphical model, and from this perspective, the maximum-entropy RL objective can be derived as an inference procedure in this model.

3.1. A Probabilistic Graphical Model for RL

As depicted in Figure 1, the proposed model consists of states $\mathbf{s}_t$, actions $\mathbf{a}_t$, and per-timestep optimality variables $\mathcal{O}_t$, which are related to rewards by $p(\mathcal{O}_t = 1|\mathbf{s}_t, \mathbf{a}_t) = \exp(r(\mathbf{s}_t, \mathbf{a}_t))$ and denote whether the action $\mathbf{a}_t$ taken from state $\mathbf{s}_t$ is optimal. While rewards are required to be non-positive through this relation, so long the rewards are bounded, they can be scaled and centered to be no greater than 0. A trajectory is the sequence of states and actions, $(\mathbf{s}_1, \mathbf{a}_1, \mathbf{s}_2, \ldots, \mathbf{s}_T, \mathbf{a}_T)$, and we aim to infer the posterior distribution $p(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}|\mathcal{O}_{1:T} = 1)$, i.e., the trajectory distribution that is optimal for all timesteps.

3.2. Variational Inference

Among existing inference tools, structured variational inference is particularly appealing for its scalability and efficiency to approximate the distribution of interest. In the variational inference framework, a variational distribution q is optimized through the variational lower bound to approximate another distribution p. Assuming a uniform prior over actions, the optimal trajectory distribution is:

$$\begin{aligned} p(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}|\mathcal{O}_{1:T} = 1) &\propto p(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}, \mathcal{O}_{1:T} = 1) \\ &= p(\mathbf{s}_1) \prod_{t=1}^{T} \exp(r(\mathbf{s}_t, \mathbf{a}_t)) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t). \end{aligned}$$

For the approximating distribution q, we choose the form $q(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}) = p(\mathbf{s}_1) \prod_{t=1}^{T} p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) q(\mathbf{a}_t|\mathbf{s}_t)$, where $p(\mathbf{s}_1)$ and $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ are fixed and given by the environment. We now rename $q(\mathbf{a}_t|\mathbf{s}_t)$ to $\pi(\mathbf{a}_t|\mathbf{s}_t)$ since this represents the desired policy. By Jensen's inequality, the variational lower bound for the evidence $\mathcal{O}_{1:T} = 1$ is

$$\begin{aligned} \log p(\mathcal{O}_{1:T} = 1) &= \log \mathbb{E}_q \left[\frac{p(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}, \mathcal{O}_{1:T} = 1)}{q(\mathbf{s}_{1:T}, \mathbf{a}_{1:T})} \right] \\ &\geq \mathbb{E}_\pi \left[\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t) - \log \pi(\mathbf{a}_t|\mathbf{s}_t) \right], \end{aligned}$$

which is the maximum entropy RL objective (Ziebart et al., 2008; Toussaint, 2009; Rawlik et al., 2013; Fox et al., 2015; Haarnoja et al., 2017). This objective adds a conditional entropy term and thus maximizes both returns and the entropy

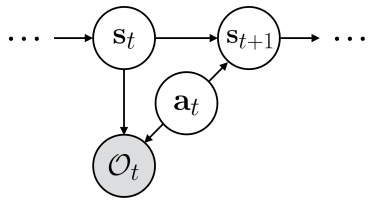

Figure 1. The graphical model for the RL-as-Inference framework consists of states $\mathbf{s}_t$, actions $\mathbf{a}_t$, and optimality variables $\mathcal{O}_t$. By incorporating rewards through the optimality variables, learning an RL policy amounts to performing inference in this model.

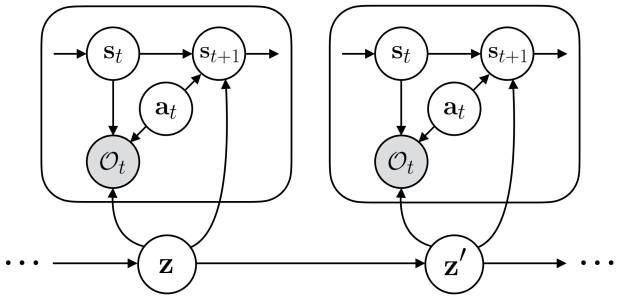

Figure 2. The graphical model for the DP-MDP. Each episode presents a new task, or MDP, determined by latent variables $\mathbf{z}$. The MDPs are further sequentially related through a transition function $p_{\mathbf{z}}(\mathbf{z}'|\mathbf{z})$.

of the policy. This formulation is known for its improvements in exploration, robustness, and stability over other RL algorithms, thus we build upon it in our method to inherit these qualities. We capture non-stationarity by augmenting the RL-as-inference model with latent variables $\mathbf{z}^i$ for each task i. As we will see in the next section, by viewing non-stationarity from this probabilistic perspective, our algorithm can be derived as an inference procedure in a unified model.

4. Off-Policy Reinforcement Learning in Non-Stationary Environments

Building upon the RL-as-inference framework, in this section, we offer a probabilistic graphical model that underlies the dynamic parameter MDP setting introduced in Section 2. Then, using tools from variational inference, we derive a variational lower bound that performs joint RL and representation learning. Finally, we present our RL algorithm, which we call **L**ifelong **L**atent **A**ctor-**C**ritic (LILAC), that optimizes this objective and builds upon on soft actor-critic (Haarnoja et al., 2018), an off-policy maximum entropy RL algorithm.

4.1. Non-stationarity as a Probabilistic Model

We can cast the dynamic parameter MDP as a probabilistic hierarchical model, where non-stationarity occurs at the episodic level, and within each episode is an instance of a stationary MDP. To do so, we construct a two-tiered model: on the first level, we have the sequence of latent variables $\mathbf{z}^i$ as a Markov chain, and on the second level, a Markov decision process corresponding to each $\mathbf{z}^i$. The graphical model formulation of the DP-MDP is illustrated in Figure 2.

Within this formulation, the trajectories gathered from each episode are modeled individually, rather than amortized as in Subsection 3.2, and the joint probability distribution is defined as follows:

$$p(\mathbf{z}^{1:N}, \tau^{1:N}) = p(\mathbf{z}^1)p(\tau^1|\mathbf{z}^1)\prod_{i=1}^{N} p(\mathbf{z}^i|\mathbf{z}^{i-1})p(\tau^i|\mathbf{z}^i)$$

where the probability of each trajectory τ given $\mathbf{z}$, assuming a uniform prior over actions, is

$$\begin{aligned} p(\tau|\mathbf{z}) &= p(\mathbf{s}_1)\prod_{t=1}^{T} p(\mathcal{O}_t = 1|\mathbf{s}_t, \mathbf{a}_t; \mathbf{z})p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t; \mathbf{z}) \\ &= p(\mathbf{s}_1)\prod_{t=1}^{T} \exp(r(\mathbf{s}_t, \mathbf{a}_t; \mathbf{z}))p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t; \mathbf{z}). \end{aligned}$$

With this factorization, the non-stationary elements of the environment are captured by the latent variables $\mathbf{z}$, and within a task, the dynamics and reward functions are necessarily stationary. This suggests that learning to infer $\mathbf{z}$, which amounts to representing the non-stationarity elements of the environment with $\mathbf{z}$, will reduce this RL setting to a stationary one. Taking this type of approach is appealing since there already exists a rich body of algorithms for the standard RL setting. In the next subsection, we describe how we can approximate the posterior over $\mathbf{z}$, by deriving the evidence lower bound for this model under the variational inference framework.

4.2. Joint Representation and Reinforcement Learning via Variational Inference

Recall the agent is operating in an online learning setting. That is, it must continuously adapt to a stream of tasks and leverage experience gathered from previous tasks for learning. Thus, at any episode $i > 1$, the agent has observed all of the trajectories collected from episodes 1 through $i-1$, $\tau^{1:i-1} = \{\tau^1, \cdots, \tau^{i-1}\}$, where $\tau = \{\mathbf{s}_1, \mathbf{a}_1, r_1, \ldots, \mathbf{s}_T, \mathbf{a}_T, r_T\}$.

We aim to infer, at every episode i, the posterior distribution over actions, given the evidence $\mathcal{O}^i_{1:T} = 1$ and the

experience from the previous episodes $\tau^{1:i-1}$. Following Subsection 3.2, we can leverage variational inference to optimize a variational lower bound to the log-probability of this set of evidence, $\log p(\tau^{1:i-1}, \mathcal{O}^i_{1:T} = 1)$. Since $p(\tau^{1:i-1}, \mathcal{O}^i_{1:T} = 1)$ factorizes as $p(\tau^{1:i-1})p(\mathcal{O}^i_{1:T} = 1|\tau^{1:i-1})$, the log-probability of the evidence can be decomposed into $\log p(\tau^{1:i-1}) + \log p(\mathcal{O}^i_{1:T} = 1|\tau^{1:i-1})$. These two terms can be separately lower bounded and summed to form a single objective.

The variational lower bound of the first term follows from that of a variational auto-encoder (Kingma & Welling, 2014) with evidence $\tau^{1:i-1}$ and latent variables $\mathbf{z}^{1:i-1}$:

$$\log p(\tau^{1:i-1}) = \log \mathbb{E}_q \left[\frac{p(\tau^{1:i-1}, \mathbf{z}^{1:i-1})}{q(\mathbf{z}^{1:i-1})} \right].$$

We choose our approximating distribution over the latent variables $\mathbf{z}^i$ to be conditioned on the trajectory from episode i, i.e. $q(\mathbf{z}^i|\tau^i)$. Then, the variational lower bound can be expressed as:

$$\log p(\tau^{1:i-1}) \geq \mathbb{E}_q \Bigg[\sum_{i'=1}^{i} \sum_{t=1}^{T} \log p(\mathbf{s}_{t+1}, r_t|\mathbf{s}_t, \mathbf{a}_t; \mathbf{z}^{i'}) - D_{\mathrm{KL}}(q(\mathbf{z}^{i'}|\tau^{i'}) \,||\, p(\mathbf{z}^{i'}|\mathbf{z}^{i'-1})) \Bigg] = \mathcal{L}_{\text{rep}}.$$

The lower bound $\mathcal{L}_{\text{rep}}$ corresponds to an objective for unsupervised representation learning in a sequential latent variable model. By optimizing the reconstruction loss of the transitions and rewards for each episode, the learned latent variables should encode the varying parameters of the MDP. Further, by imposing the prior $p(\mathbf{z}^i|\mathbf{z}^{i-1})$ on the approximated distribution q through the KL divergence, the latent variables are encouraged to be sequentially consistent across time. This prior corresponds to a model of the environment's latent dynamics and gives the agent a predictive estimate of future conditions of the environment (to the extent to which the DP-MDP is predictable). For the second term,

$$\begin{aligned}
\log p(\mathcal{O}^i_{1:T} = 1|\tau^{1:i-1}) &= \log \int p(\mathcal{O}^i_{1:T} = 1, \mathbf{z}^i|\tau^{1:i-1}) d\mathbf{z}^i \\
&= \log \int p(\mathcal{O}^i_{1:T} = 1|\mathbf{z}^i) p(\mathbf{z}^i|\tau^{1:i-1}) d\mathbf{z}^i \\
&\geq \mathbb{E}_{p(\mathbf{z}^i|\tau^{1:i-1})} \left[\log p(\mathcal{O}^i_{1:T} = 1|\mathbf{z}^i) \right] \\
&\geq \mathop{\mathbb{E}}_{\substack{p(\mathbf{z}^i|\tau^{1:i-1}) \\ \pi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{z}^i)}} \left[\sum_{i=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t; \mathbf{z}^i) - \log \pi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{z}^i) \right] \\
&= \mathcal{L}_{\text{RL}}.
\end{aligned}$$

The final inequality is given by steps from Subsection 3.2. The bound $\mathcal{L}_{\text{RL}}$ optimizes for both policy returns and policy entropy, as in the maximum entropy RL objective, but here the policy is also conditioned on the inferred latent embeddings of the MDP. This objective essentially performs task-conditioned reinforcement learning where the task variables at episode i are given by $p(\mathbf{z}^i|\tau^{1:i-1})$. Learning a multi-task RL policy is appealing, especially over a policy that adapts between episodes. That is, if the shifts in the environment are similar to those seen previously, we do not expect its performance to degrade even if the environment is shifting quickly, whereas a single-task policy would likely struggle to adapt quickly enough.

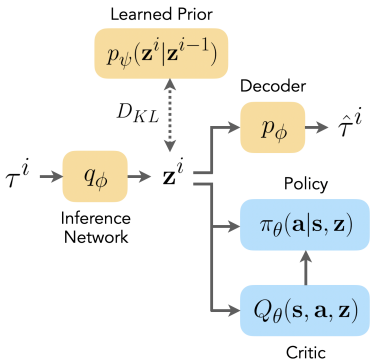

Figure 3. An overview of our network architecture. Our method consists of the actor π, the critic Q, an inference network q, a decoder network, and a learned prior over latent embeddings. Each component is implemented with a neural network.

Our proposed objective is the sum of the above two terms $\mathcal{L} = \mathcal{L}_{\text{rep}} + \mathcal{L}_{\text{RL}}$, which is also a variational lower bound for our entire model. Hence, while our objective was derived from and can be understood as an inference procedure in our probabilistic model, it also decomposes into two very intuitive objectives, with the first corresponding to unsupervised representation learning and the second corresponding to reinforcement learning.

4.3. Implementation Details

To optimize the above objective, we extend soft actor-critic (SAC) (Haarnoja et al., 2018), which implements maximum entropy off-policy RL. We introduce an inference network that outputs a distribution over latent variables for the i-th episode, $q(\mathbf{z}^i|\tau^i)$, conditioned on the trajectory from the i-th episode. The inference network, parameterized as a feedforward neural network, outputs parameters of a Gaussian distribution, and we use the reparameterization trick (Kingma & Welling, 2014) to sample $\mathbf{z}$. A decoder neural network reconstructs transitions and rewards given the latent embedding $\mathbf{z}^i$, current state $\mathbf{s}_t$, and action taken $\mathbf{a}_t$, i.e. $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t; \mathbf{z}^i)$ and $p(r_t|\mathbf{s}_t, \mathbf{a}_t; \mathbf{z}^i)$. Finally, $p(\mathbf{z}^i|\mathbf{z}^{i-1})$ and $p(\mathbf{z}^i|\tau^{1:i-1})$ are approximated with a shared long short-term memory (LSTM) network (Hochreiter & Schmidhuber, 1997), which, at each episode i, receives $\mathbf{z}^{i-1}$

from $q(\mathbf{z}^{i-1}|\tau^{i-1})$ and hidden state h_{i-1}, and produces $\mathbf{z}^i$ and the next hidden state h_i.

We visualize the entire network at a high level and how the different components interact in Figure 3. As depicted, the policy and critic are both conditioned on the environment state and the latent variables $\mathbf{z}$. During training, $\mathbf{z}$ is sampled from $q(\mathbf{z}^i|\tau^i)$ outputted by the inference network. At execution time, the latent variables $\mathbf{z}$ the policy receives are given by the LSTM network, based on the inferred latent variables from the previous episode. Following SAC (Haarnoja et al., 2018), the actor loss $\mathcal{J}_\pi$ and critic loss $\mathcal{J}_Q$ are

$$\mathcal{J}_\pi = \mathop{\mathbb{E}}_{\substack{\tau\sim\mathcal{D}\\ \mathbf{z}\sim q(\cdot|\tau)}} \left[D_{\text{KL}} \left(\pi(\mathbf{a}|\mathbf{s}) \middle|\middle| \frac{\exp(Q(\mathbf{s},\mathbf{a},\mathbf{z}))}{Z(\mathbf{s}_t)} \right) \right]$$

$$\mathcal{J}_Q = \mathop{\mathbb{E}}_{\substack{\tau\sim\mathcal{D}\\ \mathbf{z}\sim q(\cdot|\tau)}} [(Q(\mathbf{s},\mathbf{a},\mathbf{z}) - (r + V(\mathbf{s}',\mathbf{z})))^2],$$

where V denotes the target network. Our complete algorithm, Lifelong Latent Actor-Critic (LILAC), is summarized in Algorithm 1.

Algorithm 1 Lifelong Latent Actor-Critic (LILAC)

Input: env, α_Q, α_π, α_ϕ, α_{dec}, α_p
Randomly initialize θ_Q, θ_π, ϕ, θ_{dec}, and θ_p
Initialize empty replay buffer $\mathcal{D}$
Assign $\mathbf{z}^0 \leftarrow \vec{0}$
for i = 1, 2, ... **do**
  Sample $\mathbf{z}^i \sim p_\psi(\mathbf{z}^i|\mathbf{z}^{i-1})$
  Collect trajectory τ^i with $\pi_\theta(\mathbf{a}|\mathbf{s},\mathbf{z})$
  Update replay buffer $\mathcal{D}[i] \leftarrow \tau^i$
  for j = 1, 2, ..., N **do**
    Sample a batch of episodes E from $\mathcal{D}$
    ▷ Update actor and critic
    $\theta_Q \leftarrow \theta_Q - \alpha_Q \nabla_{\theta_Q} \mathcal{J}_Q$
    $\theta_\pi \leftarrow \theta_\pi - \alpha_\pi \nabla_{\theta_\pi} \mathcal{J}_\pi$
    ▷ Update inference network
    $\phi \leftarrow \phi - \alpha_\phi \nabla_\phi (\mathcal{J}_{\text{dec}} + \mathcal{J}_{\text{KL}} + \mathcal{J}_Q)$
    ▷ Update model
    $\theta_{\text{dec}} \leftarrow \theta_{\text{dec}} - \alpha_{\text{dec}} \nabla_{\theta_{\text{dec}}} \mathcal{J}_{\text{dec}}$
    $\theta_p \leftarrow \theta_p - \alpha_p \nabla_{\theta_p} \mathcal{J}_{\text{KL}}$
  end for
end for

5. Related Work

While the POMDP formulation can capture non-stationarity and partial observability in sequential decision-making problems, exact solution methods are tractable only for tiny state and actions spaces (Kaelbling et al., 1998). Representation learning, and especially deep learning paired with amortized variational inference, has enabled scaling of the POMDP formulation to a larger class of problems, including continuous state and action spaces (Igl et al., 2018; Han et al., 2020; Hafner et al., 2019) and image observations (Lee et al., 2019a; Kapturowski et al., 2019). However, the generality of the POMDP ignores performance improvements that may be realized by exploiting the structure of the DP-MDP problem, and does not explicitly consider between-episode non-stationarity.

A variety of intermediate problem statements between episodic MDPs and POMDPs have been proposed. The Bayes-adaptive MDP formulation (BAMDP) (Duff, 2002; Ross et al., 2008), as well as the closely related hidden parameter MDP (HiP-MDP) (Doshi-Velez & Konidaris, 2016) consider an MDP with unknown parameters governing the reward and dynamics, which we aim to infer online over the course of one episode. In this formulation, the exploration-exploitation dilemma is resolved by augmenting the state space with a representation of posterior belief over the latent parameters. As noted by Duff (2002) in the RL literature and Feldbaum (1960); Bar-Shalom & Tse (1974) in control theory, this representation rapidly becomes intractable due to exploding state dimensionality. As with the POMDP setting, recent work has developed effective methods for policy optimization in BAMDPs via, primarily, amortized inference (Zintgraf et al., 2020; Rakelly et al., 2019; Lee et al., 2019b). However, the BAMDP framework does not address the dynamics of the latent parameter between episodes, assuming a temporally-fixed structure. In contrast to the BAMDP formulation, we are capable of modeling the evolution of the latent variable over the course of episodes, leading to better priors for online inference.

A strongly related setting to the DP-MDP is the hidden-mode MDP (Choi et al., 2000). The HM-MDP proposes to augment an MDP with a latent parameter that evolves via a hidden Markov model with a discrete number of states. In both the HM-MDP and the DP-MDP, the latent variable evolves infrequently, as opposed to at every time step as in a POMDP setting. While the HM-MDP does not connect the non-stationarity of the latent parameter to the episodic RL problem, it is limited to a fixed number of latent variable states due to the use of standard HMM inference algorithms. In contrast, our approach allows continuous latent variables, thus widely extending the range of applicability.

Non-stationarity in learning. LILAC shares conceptual similarities with methods from online learning and lifelong learning (Shalev-Shwartz, 2012; Gama et al., 2014), which aim to capture non-stationarity in supervised learning, as well as meta-learning and meta-reinforcement learning algorithms, which aim to rapidly adapt to new settings. Meta-learning (Schmidhuber, 1987) algorithms learn an efficient adaptation procedure via meta-training on a variety of tasks, such that learning in a new task can be performed as efficiently as possible (Finn et al., 2017). Within meta-reinforcement learning, two dominant techniques ex-

ist: optimization-based (Finn et al., 2017; Rothfuss et al., 2019; Zintgraf et al., 2019; Stadie et al., 2018) and context-based, which includes both recurrent architectures (Duan et al., 2016; Wang et al., 2016; Mishra et al., 2018) and architectures based on latent variable inference (Rakelly et al., 2019; Lee et al., 2019a; Zintgraf et al., 2020). LILAC fits into this last category within this taxonomy, but extends previous methods by considering inter-episode latent variable dynamics. Previous embedding-based meta-RL algorithms—while able to perform online inference of latent variables and incorporate this posterior belief into action selection—do not consider how these latent variables evolve over the lifetime of the agent, as in the DP-MDP setting.

The inner latent variable inference component of LILAC possesses strong similarities to the continual and lifelong learning setting (Gama et al., 2014). Indeed, whereas context-based meta-RL approaches do not consider the correlation of latent factors between episodes (beyond assuming iid draws from a shared prior) and thus are not able to share knowledge between subsequent episodes, LILAC may be interpreted as an approach towards leaning-to-lifelong-learn.

Many continual and lifelong learning aim to learn a variety of tasks without forgetting previous tasks (Kirkpatrick et al., 2017; Zenke et al., 2017; Lopez-Paz et al., 2017; Aljundi et al., 2019; Parisi et al., 2019; Rusu et al., 2016; Shmelkov et al., 2017; Rebuffi et al., 2017; Shin et al., 2017). We consider a setting where it is practical to store past experiences in a replay buffer (Rolnick et al., 2019; Finn et al., 2019). Unlike these prior works, LILAC aims to learn the dynamics associated with latent factors, and perform online inference.

6. Experimental Evaluation

In our experiments, we aim to address our central hypothesis: by leveraging our latent variable model, our approach can make learning under persistent non-stationarity both effective and efficient.

Environments. We construct four continuous control environments with persistent varying sources of change in the reward and/or dynamics. These environments are designed such that the policy needs to change in order to achieve good performance. The first is derived from the simulated Sawyer reaching task in the Meta-World benchmark (Yu et al., 2019), in which the target position is not observed and moves between episodes. In the second environment based on Half-Cheetah from OpenAI Gym (Brockman et al., 2016), we consider changes in the direction and magnitude of wind forces on the agent, and changes in the target velocity. We next consider the 8-DoF minitaur environment (Tan et al., 2018) and vary the mass of the agent between episodes, representative of a varying payload. Finally, we construct a 2D navigation task in an *infinite, non-episodic* environment with non-stationary dynamics which we call 2D Open World. The agent's goal is to collect food pellets and to avoid other objects and obstacles, whilst subjected to unknown perturbations that vary on an episodic schedule. These environments are illustrated in Figure 4. For full environment details, see Appendix A.

Comparisons. We compare our approach to standard soft-actor critic (SAC) (Haarnoja et al., 2018), which corresponds to our method without any latent variables, allowing us to evaluate the performance of off-policy algorithms amid non-stationarity. We also compare to stochastic latent actor-critic (SLAC) (Lee et al., 2019a), which learns to model partially observed environments with a latent variable model but does not address inter-episode non-stationarity. This comparison allows us to evaluate the importance of modeling non-stationarity between episodes. Finally, we include proximal policy optimization (PPO) (Schulman et al., 2017) as a comparison to on-policy RL. Since the tasks in the Sawyer and Half-Cheetah domains involve goal reaching, we can obtain an oracle by training a goal-conditioned SAC policy, i.e. with the true goal concatenated to the observation. We provide this comparison to help contextualize the performance of our method against other algorithms. For all hyperparameter details, see Appendix B.

Results. Our experimental results are shown in Figure 5. Since on-policy algorithms tend to have worse sample complexity, we run PPO for 10 million environment steps and plot only the asymptotic returns. In all domains, LILAC attains higher and more stable returns compared to SAC, SLAC, and PPO. Since SAC amortizes experience collected across episodes into a single replay buffer, we observe that the algorithm converges to an averaged behavior. Meanwhile, SLAC does not have the mechanism to model non-stationarity across episodes, and has to infer the unknown dynamics and reward from the initial steps taken during each episode, which the algorithm is not very successful at. Due to the cyclical nature of the tasks, the learned behavior of SLAC results in oscillating returns across tasks. Similarly, PPO cannot adapt to per-episode changes in the environment and ultimately converges to learning an average policy. In contrast to these methods, LILAC infers how the environment changes in future episodes and steadily maintains high rewards over the training procedure, despite experiencing persistent shifts in the environment in each episode. Further, LILAC can learn under simultaneous shifts in *both* dynamics and rewards, verified by the HC WindVel results. Notably, LILAC also adeptly handles shifts in the 2D Open World environment without episodic resets. A partial snapshot of the agent's lifetime from this task is visualized in Figure 4.

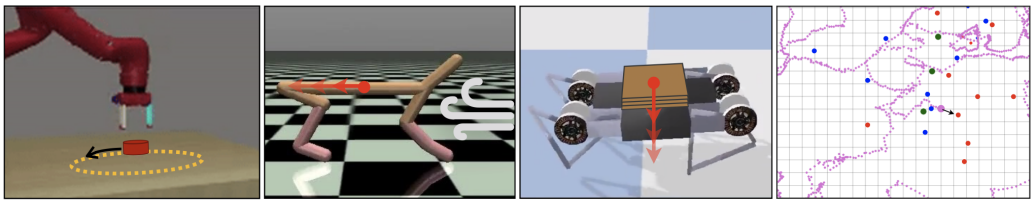

Figure 4. The environments in our evaluation. Each environment changes over the course of learning, including a changing target reaching position (left), variable wind and goal velocities (middle left), and variable payloads (middle right). We also introduce a 2D open world environment with non-stationary dynamics and visualize a partial snapshot of the LILAC agent's lifetime in purple (right).

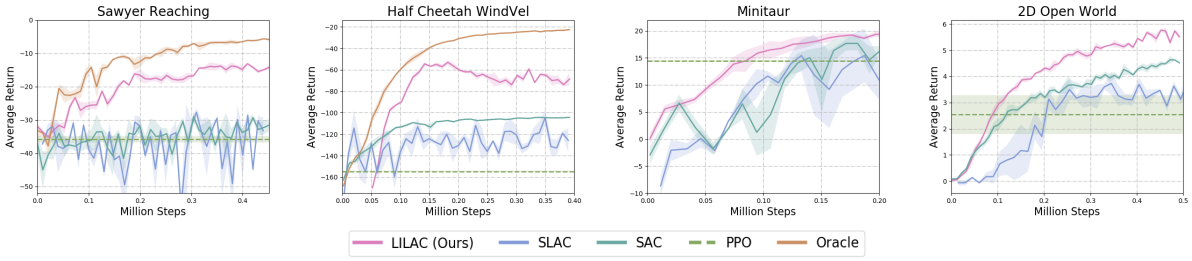

Figure 5. Learning curves across our experimental domains. For PPO, we plot the asymptotic returns achieved by the algorithm after 10 million environment steps. In all settings, our approach is substantially more stable and successful than SAC, SLAC, and PPO.

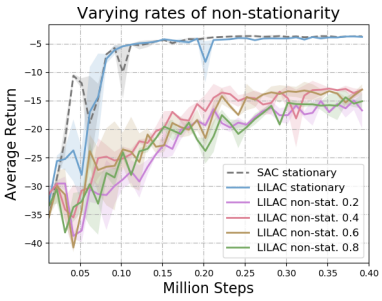

Figure 6. LILAC evaluated on the Sawyer task with varying rates of non-stationarity by moving the goal 0.2, 0.4, 0.6, and 0.8 radians along a circle between episodes. We also plot the performance of LILAC under stationary conditions (with the goal fixed).

Rate of environment shift. We next evaluate whether LILAC can handle varying rates of non-stationarity. To do so, we use the Sawyer reaching domain, where the goal moves along a fixed-radius circle, and vary the step size along the circle to generate environments that shift at different speeds. As depicted in Figure 6, LILAC's performance is largely independent of the environment's rate of change. We also evaluate LILAC under stationary conditions, i.e. with a fixed goal, and LILAC achieves the same performance as SAC, thus retaining the ability to learn as effectively as SAC in a fixed environment. These results demonstrate LILAC's efficacy under a range of rates of non-stationarity, including the stationary case. The gap in LILAC's performance between the non-stationary and stationary cases is likely due to imprecise estimates of future environment conditions given by the prior $p_\phi(\mathbf{z}'|\mathbf{z})$. Currently, the executed policy uses fixed $\mathbf{z}$ given by the prior for the entire episode, but a natural extension that may improve performance is updating $\mathbf{z}$ *during* each episode by encoding partial trajectories with the inference network.

7. Conclusion

We considered the problem of reinforcement learning with lifelong non-stationarity, a problem which we believe is critical to reinforcement learning systems operating in the real world. This problem is at the intersection of reinforcement learning under partial observability (i.e. POMDPs) and online learning; hence we formalized the problem as a special case of a POMDP that is also significantly more tractable. We derive a graphical model underlying this problem setting, and utilize it to derive our approach under the formalism of reinforcement learning as probabilistic inference (Levine, 2018). Our method leverages this latent variable model to model the change in the environment, and conditions the policy and critic on the inferred values of these latent variables. On a variety of challenging continuous control tasks with significant non-stationarity, we observe that our approach leads to substantial improvement compared to state-of-the-art reinforcement learning methods.

While the DP-MDP formulation represents a strict generalization of the commonly-considered meta-reinforcement learning settings (typically, a BAMDP (Zintgraf et al.,

2020)), it is still somewhat limited in its generality. In particular, the assumption of task parameters shifting *between* episodes, but never during them, presents a possibly unrealistic limitation. While relaxing this assumption leads, in the worst case, to a POMDP, there is potentially additional structure that may be exploited under the HM-MDP (Choi et al., 2000) assumption of infrequent, discrete, unobserved shifts in the task parameters. In particular, this notion of infrequent, discrete shifts underlies the changepoint detection literature (Adams & MacKay, 2007; Fearnhead & Liu, 2007). Previous work both within sequential decision making in changing environments (Da Silva et al., 2006; Hadoux et al., 2014; Banerjee et al., 2017) and meta-learning within changing data streams (Harrison et al., 2019) may enable a version of LILAC capable of handling unobserved changepoints, and this setting is likely a fruitful direction for future research.

References

Adams, R. P. and MacKay, D. J. Bayesian online changepoint detection. *arXiv:0710.3742*, 2007.

Al-Shedivat, M., Bansal, T., Burda, Y., Sutskever, I., Mordatch, I., and Abbeel, P. Continuous adaptation via meta-learning in nonstationary and competitive environments. *International Conference on Learning Representations (ICLR)*, 2017.

Aljundi, R., Kelchtermans, K., and Tuytelaars, T. Task-free continual learning. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Banerjee, T., Liu, M., and How, J. P. Quickest change detection approach to optimal control in markov decision processes with model changes. *American Control Conference (ACC)*, 2017.

Bar-Shalom, Y. and Tse, E. Dual effect, certainty equivalence, and separation in stochastic control. *IEEE Transactions on Automatic Control*, 19(5):494–500, 1974.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Choi, S. P., Yeung, D.-Y., and Zhang, N. L. Hidden-mode markov decision processes for nonstationary sequential decision making. In *Sequence Learning*, pp. 264–287. Springer, 2000.

Da Silva, B. C., Basso, E. W., Bazzan, A. L., and Engel, P. M. Dealing with non-stationary environments using context detection. *International Conference on Machine Learning (ICML)*, 2006.

Doshi-Velez, F. and Konidaris, G. Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. RL2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Duff, M. O. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. PhD thesis, University of Massachusetts at Amherst, 2002.

Fearnhead, P. and Liu, Z. On-line inference for multiple changepoint problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2007.

Feldbaum, A. Dual control theory. I. *Avtomatika i Telemekhanika*, 1960.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. *International Conference on Machine Learning (ICML)*, 2017.

Finn, C., Rajeswaran, A., Kakade, S., and Levine, S. Online meta-learning. *International Conference on Machine Learning (ICML)*, 2019.

Fox, R., Pakman, A., and Tishby, N. Taming the noise in reinforcement learning via soft updates. *arXiv preprint arXiv:1512.08562*, 2015.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 2014.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. *International Conference on Machine Learning (ICML)*, 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International Conference on Machine Learning (ICML)*, 2018.

Hadoux, E., Beynier, A., and Weng, P. Sequential decision-making under non-stationary environments via sequential change-point detection. 2014.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. *International Conference on Machine Learning (ICML)*, 2019.

Han, D., Doya, K., and Tani, J. Variational recurrent models for solving partially observable control tasks. *International Conference on Learning Representations (ICLR)*, 2020.

Harrison, J., Sharma, A., Finn, C., and Pavone, M. Continuous meta-learning without tasks. *arXiv preprint arXiv:1912.08866*, 2019.

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

Igl, M., Zintgraf, L., Le, T. A., Wood, F., and Whiteson, S. Deep variational reinforcement learning for pomdps. *International Conference on Machine Learning (ICML)*, 2018.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 1998.

Kapturowski, S., Ostrovski, G., Dabney, W., Quan, J., and Munos, R. Recurrent experience replay in distributed reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2019.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. *International Conference on Learning Representations (ICLR)*, 2014.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 2017.

Lee, A. X., Nagabandi, A., Abbeel, P., and Levine, S. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arXiv:1907.00953*, 2019a.

Lee, G., Hou, B., Mandalika, A., Lee, J., Choudhury, S., and Srinivasa, S. S. Bayesian policy optimization for model uncertainty. *International Conference on Learning Representations (ICLR)*, 2019b.

Levine, S. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Lopez-Paz, D. et al. Gradient episodic memory for continual learning. *Neural Information Processing Systems (NeurIPS)*, 2017.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. A simple neural attentive meta-learner. *International Conference on Learning Representations (ICLR)*, 2018.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

Rakelly, K., Zhou, A., Quillen, D., Finn, C., and Levine, S. Efficient off-policy meta-reinforcement learning via probabilistic context variables. *International Conference on Machine Learning (ICML)*, 2019.

Rawlik, K., Toussaint, M., and Vijayakumar, S. On stochastic optimal control and reinforcement learning by approximate inference. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

Rebuffi, S.-A., Kolesnikov, A., and Lampert, C. H. icarl: Incremental classifier and representation learning. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., and Wayne, G. Experience replay for continual learning. *Neural Information Processing Systems (NeurIPS)*, 2019.

Ross, S., Chaib-draa, B., and Pineau, J. Bayes-adaptive pomdps. *Neural Information Processing Systems (NeurIPS)*, 2008.

Rothfuss, J., Lee, D., Clavera, I., Asfour, T., and Abbeel, P. Promp: Proximal meta-policy search. *International Conference on Learning Representations (ICLR)*, 2019.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *arXiv:1606.04671*, 2016.

Schmidhuber, J. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shalev-Shwartz, S. Online learning and online convex optimization. *"Foundations and Trends in Machine Learning"*, 2012.

Shin, H., Lee, J. K., Kim, J., and Kim, J. Continual learning with deep generative replay. *Neural Information Processing Systems (NeurIPS)*, 2017.

Shmelkov, K., Schmid, C., and Alahari, K. Incremental learning of object detectors without catastrophic forgetting. *arXiv:1708.06977*, 2017.

Stadie, B. C., Yang, G., Houthooft, R., Chen, X., Duan, Y., Wu, Y., Abbeel, P., and Sutskever, I. Some considerations on learning to explore via meta-reinforcement learning. *arXiv:1803.01118*, 2018.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Sutton, R. S., Koop, A., and Silver, D. On the role of tracking in stationary environments. *International Conference on Machine Learning (ICML)*, 2007.

Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., Bohez, S., and Vanhoucke, V. Sim-to-real: Learning agile locomotion for quadruped robots. *Robotics: Science and Systems (RSS)*, 2018.

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

Toussaint, M. Robot trajectory optimization using approximate inference. *International Conference on Machine Learning (ICML)*, 2009.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *Conference on Robot Learning (CoRL)*, 2019.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. *International Conference on Machine Learning (ICML)*, 2017.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. 2008.

Zintgraf, L., Shiarlis, K., Igl, M., Schulze, S., Gal, Y., Hofmann, K., and Whiteson, S. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. *International Conference on Learning Representations (ICLR)*, 2020.

Zintgraf, L. M., Shiarlis, K., Kurin, V., Hofmann, K., and Whiteson, S. Fast context adaptation via meta-learning. *International Conference on Machine Learning (ICML)*, 2019.

A. Environment Details

Below, we provide environment details for each of the four experimental domains.

A.1. Sawyer Reaching

In this environment, which is based on the simulated Sawyer reaching task in the Meta-World suite (Yu et al., 2019), the goal is to reach a particular position. The target position, which is unobserved throughout, moves after each episode.

The episodes are 150 timesteps long, and the state is the position of the end-effector in (x, y, z) coordinate space. The actions correspond to changes in end-effector positions. The reward is defined as

$$r(\mathbf{s}, \mathbf{a}) = -\|\mathbf{s} - \mathbf{s}_g\|_2,$$

where $\mathbf{s}_g$ at episode i is defined as

$$\mathbf{s}_g = \begin{bmatrix} 0.1 \cdot \cos(0.5 \cdot i) \\ 0.1 \cdot \sin(0.5 \cdot i) \\ 0.2 \end{bmatrix}.$$

In other words, the sequence of goals is defined by a circle in the xy-plane. For the oracle comparison, the sequence of goals and the reward function are the same, except the state observation here is the concatenation of the end-effector position and the goal position $\mathbf{s}_g$.

A.2. Half-Cheetah Vel

This environment builds off of the Half-Cheetah environment from OpenAI Gym (Brockman et al., 2016). In this domain, the agent must reach a target velocity in the x-direction, which varies across episodes, i.e., the reward is

$$r(\mathbf{s}, \mathbf{a}) = -\|v_s - v_g\|_2 - 0.05 \cdot \|\mathbf{a}\|_2,$$

where v_s is the observed velocity of the agent. The state consists of the position and velocity of the agent's center of mass and the angular position and angular velocity of each of its six joints, and actions correspond to torques applied to each of the six joints.

The target velocity v_g varies according to a sine function, i.e., the target velocity for episode i is

$$v_g = 1.5 + 1.5 \sin(0.5 \cdot i).$$

For the oracle comparison, the target velocity v_g is appended to the state observation. Each episode, across all comparisons, is 50 timesteps long.

A.3. Half-Cheetah Wind+Vel

In this variant of Half-Cheetah Vel, the agent is additionally subjected to varying wind forces. The force for each episode is defined by

$$f_w = 10 + 10 \sin(0.2 \cdot i)$$

and is applied constantly along the x-direction throughout the episode.

A.4. Minitaur Mass

For this last domain, we use the simulated Minitaur environment developed by Tan et al. (2018). We induce non-stationarity by varying the mass of the agent between episodes akin to increasing and decreasing payloads. Specifically, the mass at each episode is

$$m = 1.0 + 0.75 \sin(0.3 \cdot i).$$

The reward is defined by

$$r(\mathbf{s}_t, \mathbf{a}_t) = 0.5 - |0.5 - \mathbf{s}_{t,v}| - 0.01 \cdot \|\mathbf{a}_t - 2\mathbf{a}_{t-1} + \mathbf{a}_{t-2}\|_1,$$

where the first two terms correspond to the velocity reward, which encourages the agent to run close to a target velocity of 0.5 m/s, and the last term corresponds to an acceleration penalty defined by the last three actions taken by the agent. The state includes the angles, velocities, and torques of all eight motors, and the action is the target motor angle for each motor. Each episode is 100 timesteps long.

B. Experimental and Hyperparameter Details

In this section, we provide details of the hyperparameters used for each method.

B.1. LILAC (Ours)

Latent space. For our method, we use a latent space size of 8 in Sawyer Reaching, and size of 40 in the other experiments: Half-Cheetah Vel, Half-Cheetah Wind+Vel, and Minitaur Mass.

Inference and decoder networks. The inference and decoder networks are MLPs with 2 fully-connected layers of size 64 in Sawyer Reaching; 1 fully-connected layer of size 512 in Half-Cheetah Vel and Half-Cheetah Wind+Vel; and 2 fully-connected layers of size 512 in Minitaur Mass.

Policy and critic networks. The policy and critic networks are MLPs with 3 fully-connected layers of size 256 in the Sawyer Reaching experiment; and 2 fully-connected layers of size 256 in the other experiments.

Training. For each training iteration, we sample a batch of 32 episodes and from each episode, we sample 8 tuples of transitions and rewards $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$. The training objective of the inference network is a linear combination of the reconstruction loss, critic loss, and KL divergence from the learned prior:

$$\mathcal{L}_{\text{enc}} = \mathcal{L}_{\text{dec}} + \beta_1 \mathcal{L}_Q + \beta_2 \mathcal{L}_{\text{KL}}.$$

For the Sawyer Reaching experiment, β_1 and β_2 are

$$\beta_1 = \begin{cases} 0, & \text{iter} < 10000 \\ 1, & \text{iter} \geq 10000 \end{cases}$$

$$\beta_2 = \begin{cases} 0, & \text{iter} < 10000 \\ \min(\text{1e-6} \cdot (\text{iter} - 10000), 1), & \text{iter} \geq 10000 \end{cases}$$

For Half-Cheetah Vel and Half-Cheetah Wind+Vel, β_1 and β_2 are

$$\beta_1 = \begin{cases} 0, & \text{iter} < 50000 \\ 1, & \text{iter} \geq 50000 \end{cases}$$

$$\beta_2 = \begin{cases} 0, & \text{iter} < 10000 \\ \min(\text{1e-6} \cdot (\text{iter} - 10000), 1), & \text{iter} \geq 10000 \end{cases}$$

Finally, for the Minitaur Mass experiment, β_1 and β_2 are

$$\beta_1 = \begin{cases} 0, & \text{iter} < 10000 \\ 1, & \text{iter} \geq 10000 \end{cases}$$

$$\beta_2 = \begin{cases} 0, & \text{iter} < 10000 \\ \text{1e-6}, & \text{iter} \geq 10000 \end{cases}$$

B.2. Stochastic Latent Actor-Critic (Lee et al., 2019a)

Latent space. SLAC factorizes its per-timestep latent variable $\mathbf{z}_t$ into two stochastic layers $\mathbf{z}_t^1$ and $\mathbf{z}_t^2$, i.e. $p(\mathbf{z}_t) = p(\mathbf{z}_t^2|\mathbf{z}_t^1)p(\mathbf{z}_t^1)$. In the Sawyer Reaching experiment, the size of $\mathbf{z}_t^1$ is 16 and the size of $\mathbf{z}_t^2$ is 8. In all other experiments, the size of $\mathbf{z}_t^1$ is 64 and the size of $\mathbf{z}_t^2$ is 32.

Inference and decoder networks. The inference and decoder networks are MLPs with 2 fully-connected layers of size 64 in Sawyer Reaching; 1 fully-connected layer of size 512 in Half-Cheetah Vel and Half-Cheetah Wind+Vel; and 2 fully-connected layers of size 512 in Minitaur Mass.

Policy and critic networks. The policy and critic networks are MLPs with 3 fully-connected layers of size 256 in the Sawyer Reaching experiment; and 2 fully-connected layers of size 256 in the other experiments.

B.3. Soft Actor-Critic (Haarnoja et al., 2018)

Policy and critic networks. The policy and critic networks are MLPs with 3 fully-connected layers of size 256 in the Sawyer Reaching experiment; and 2 fully-connected layers of size 256 in the other experiments.