An Adaptive Deep RL Method for Non-Stationary Environments with Piecewise Stable Context

Anonymous Author(s)
Affiliation
Address
email

Abstract

One of the key challenges in deploying RL to real-world applications is to adapt to variations of unknown environment contexts, such as changing terrains in robotic tasks and fluctuated bandwidth in congestion control. Existing works on adaptation to unknown environment contexts either assume the contexts are the same for the whole episode or assume the context variables are Markovian. However, in many real-world applications, the environment context usually stays stable for a stochastic period and then changes in an abrupt and unpredictable manner within an episode, resulting in a segment structure, which existing works fail to address. To leverage the segment structure of piecewise stable context in real-world applications, in this paper, we propose a ***Se**gmented **C**ontext **B**elief **A**ugmented **D**eep (SeCBAD)* RL method. Our method can jointly infer the belief distribution over latent context with the posterior over segment length and perform more accurate belief context inference with observed data within the current context segment. The inferred belief context can be leveraged to augment the state, leading to a policy that can adapt to abrupt variations in context. We demonstrate empirically that SeCBAD can infer context segment length accurately and outperform existing methods on a toy grid world environment and Mujoco tasks with piecewise-stable context.

1 Introduction

Deep reinforcement learning has achieved great success in a wide range of challenging environments such as Atari games (Mnih et al., 2013; Bellemare et al., 2012; Hessel et al., 2017) or continuous control tasks (Schulman et al., 2015, 2017a). However, in stark contrast with this trend, applying RL to real-world applications remains a great challenge. In most real-world settings, there could be variations in environmental factors, such as changing terrains in robotic tasks, fluctuated bandwidth in congestion control, and dynamic traffic patterns in autonomous driving. We refer to such environment factors as *situation* or *context*. The changes in context are not neglectable since context usually has a substantial impact on transition and reward functions. When the context is fixed and known to us, the problem is stationary and easier to solve. However, in most realistic settings, the context is usually dynamic within an episode and unknown to us at test time. Therefore, detecting and adapting to variations in context is very important for RL agents to make a real impact in a wide range of real-world applications.

In real-world environments with varied unknown contexts, we find a typical evolving context pattern particularly interesting. The context c usually stays the same for a stochastic period until it changes abruptly and unpredictably into another context value c' which is sampled i.i.d. from some prior context distribution. What's more, we usually do not have access to context c directly but instead have access to some noisy observation x_t sampled from some distribution $p(x_t|c)$ only at training time (e.g., as auxiliary information from the simulator). We take fluctuated bandwidth in congestion control as an example. In Figure 1, we show a typical trace of network available bandwidth. The

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

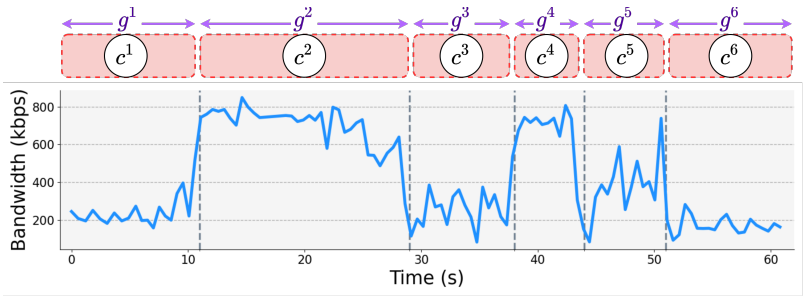


Figure 1: A typical trace of network available bandwidth. The trace can be approximately divided into several segments of different length g^i, for $i \in \{1, 2, \cdots, 6\}$. The network condition changes abruptly at the end of each segment, while in each segment the network condition stays stable.

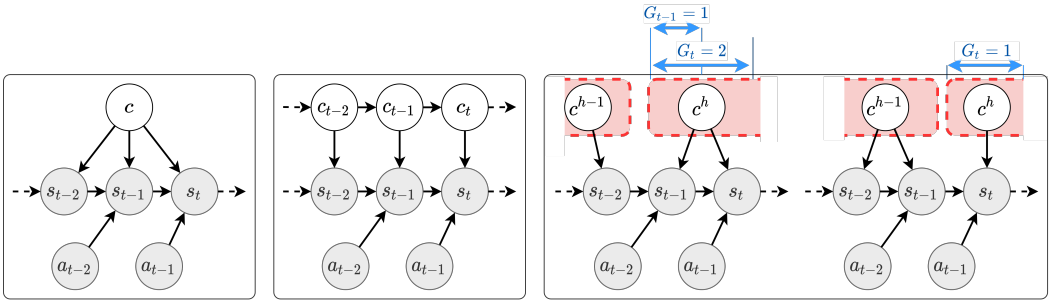
fluctuated bandwidth is usually modeled by multiple non-overlapping segments (Akhtar et al., 2018; Zhang and Duffield, 2001). Within each segment, the network condition is stationary, and thus the bandwidth approximately follows the same distribution. At the end of a segment, the network condition changes, and the bandwidth changes abruptly into another distribution. Here the context c represents the parameters that determine the distribution of fluctuated bandwidth (network condition), while the observations x_t represents the observed bandwidth which follows the above distribution $p(x_t|c)$. While the contexts c are piecewise-stable, there could be slight variations in x_t within each segment. This piecewise-stable pattern of context dynamics is of particular interest to us since it can capture a wide range of stochastic context processes in real-world applications, such as changing terrains in robotic tasks and fluctuating bandwidth in congestion control. Since the change in context is abrupt and unpredictable, we cannot predict the future context in advance. The best we can do is detect and adapt to changes when they happen.

Although adaptation to varied unknown contexts has been studied under non-stationary RL and meta RL, few existing works look into piecewise stable context with abrupt changes within an episode. Most works in meta RL as task inference (Rakelly et al., 2019; Zintgraf et al., 2020; Zhao et al., 2020; Poiani et al., 2021) and some works in non-stationary RL (Chandak et al., 2020a,b; Xie et al., 2021) assume the context stays the same for the whole episode and infer the context based on the entire episode (c.f. Figure 2(a)), therefore cannot quickly adapt to context changes within an episode. Other works on non-stationary RL assume intra-episode context changes and model c_t at each time step, but few study the piecewise-stable context as we do. Nagabandi et al. (2018) directly predict the context and thus cannot capture the prior that the context tends to stay the same for a stochastic period. Feng et al. (2022); Ren et al. (2022) model Markovian discrete context for each time step (c.f. Figure 2(b)), therefore failing to model the non-markov property of context and the prior over context segment length in our setting. Compared with existing works, our setting is distinctive since the segment structure is latent to us (c.f. Figure 2(c)). Therefore, the unique challenge in our setting is that we need to infer the segment structure, which can be further leveraged to infer belief context by only incorporating the relevant observed data in the current segment.

This paper studies how to infer segment structure to detect abrupt context changes and infer belief context accordingly to adapt to the piecewise-stable context in RL environments. We first introduce latent situational MDP which models RL environments with the stochastic *situation/context* process (Section 2). Then, we introduce how to infer the belief context from observed data. To address the challenge above, we propose to infer context segment structure and belief context jointly from observed data (Section 3.1). Then we augment the state with inferred belief context so that the RL agent can automatically trade-off between acting optimally conditioned on inferred context and gathering more information about the current context (Section 3.2). Finally, we combine the training objectives for RL and inference and present all the details for our proposed deep RL algorithm (Section 3.3). We evaluate our algorithm on a gridworld environment with dynamic goals and MuJoCo tasks (Todorov et al., 2012) with varied contexts. Experiments demonstrate that our algorithm can quickly detect and adapt to abrupt changes in piece-wise stable contexts and outperform existing methods(Section 4).

Our contributions can be listed as follows:

- We introduce latent situational MDP with piecewise-stable context, which can capture a wide range of real-world applications (Section 2).
- We propose SeCBAD, an adaptive deep RL method for non-stationary environments with piecewise-stable context. Our method can infer the context segment structure and the belief context accordingly from observed data, which can be leveraged to detect and adapt to context changes (Section 3).



(a) One Context (b) Markovian Contexts (c) Our setting

Figure 2: Probabilistic Graphical Models (PGMs) for different problem settings, with shaded circles for observable variables and white circles for latent variables. (a) One context: assume the context remains unchanged for the whole episode. (b) Markovian context: assuming the context is Markovian for each time step. (c) Our setting: the context remains unchanged in each segment, but the segment structure, illustrated by the red segment is unknown and needs to be inferred. We show two possible examples of PGM corresponding to $G_t = 2$ ((c), left) and $G_t = 1$ ((c), right), where G_t measures the length of the current segment up to time step t.

- Experiments on a gridworld environment and Mujoco tasks with piecewise-stable context demonstrate that our method can quickly detect and adapt to abrupt context changes and outperform existing methods (Section 4).

2 Problem Formulation

In this section, we define a *latent situational Markov Decision Process (LS-MDP)* as a tuple $M = (\mathcal{S}, \mathcal{A}, \mathcal{C}, \mathcal{X}, G, T, R, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, and $\gamma \in (0, 1]$ is the discount factor. To formulate the variations in environment factors, we introduce $\mathcal{C}$, the set of latent contexts, and $\mathcal{X}$, the set of observable contexts. $\mathcal{C}$ refers to the set of underlying hidden contexts that contains all necessary information but are left unobservable to the agent, while $\mathcal{X}$ refers to the set of contexts with only partial information. G refers to the segment length which will be described in detail in the next paragraph. $\mathcal{X}$ are observable only during training and unobservable during deployment. In an example environment setting where a robot walking over changing terrain, $\mathcal{C}$ represents the terrain features containing perfect information, and $\mathcal{X}$ represents noisy and imperfect information like mechanical metrics of the current step from virtual sensors in a simulator and thus is only accessible during training.

In our setting, we focus on the case where the environmental changes are abrupt and irregular, in contrast to the smoothly changing assumption on context/task in existing works in non-stationary RL (Chandak et al., 2020a,b). To better model the generative process of the contexts, we introduce the segment length G. Each episode is composed of several stationary segments with different segment lengths. For the ease of notation, we also introduce G_t, which measures the length of the current segment up to time step t. Then, the generative process can be described as follows: at the beginning of the h-th segment, the environment samples $G^h \sim p_G(G)$ and $c^h \sim p_c(c)$ from prior distributions p_G and p_c. Then, in the next G^h steps, the latent context c^h remains unchanged. For each time step t in this segment, the current segment length accumulates as $G_t = G_{t-1} + 1$ (we define $G_t = 1$ for the first time step in this segment), and the current latent context satisfies that $c_t = c^h$. The agent can observe $x_t \sim p(x_t|c_t)$ for each time step if during training. The stationarity lasts until the end of the current segment, then the environment resamples G^{h+1} and c^{h+1}, and the process repeats. We show two examples of graphical models of our setting given different G_t in Figure 2(c).

The transition function $T : p(s_{t+1}|s_t, a_t, c^h)$ and the reward function $R : p(r_t|s_t, a_t, c^h)$ are all conditioned on current latent context c^h. Therefore the changes in $\mathcal{C}$ lead to the changes in transition and reward functions. At test time, $\mathcal{X}$ is no longer accessible, so we need to infer context changes from observed transitions and rewards. Since the latent context $\mathcal{C}$ remains unobservable and changes silently, the environment is no longer stationary for the agents. To act optimally, it is important to keep track of the environment to recognize changes in time, and rapidly adapt to those changes.

3 Methodology

In this section, we present our *Segmented Context Belief Augmented Deep (SeCBAD)* RL method and elaborate on how SeCBAD solves the challenges discussed above. Our method consists of two main components:

- Joint inference of the belief distribution over the latent context and the segment structure from observed data.
- Policy optimization with inferred belief context under the belief MDP framework.

We first introduce the latent context inference part in Section 3.1, especially how to infer the segment structure jointly with belief context and leverage the segment structure to remove irrelevant observed data. After the belief context is approximated, it is then incorporated into the state as the input of the policy. We detail the policy optimization part under the belief MDP framework in Section 3.2. And finally, in Section 3.3, we describe how these parts constitute a practical algorithm.

3.1 Joint Inference for Belief Context and Segment Structure

In this part, we perform joint inference over latent context c_t and current segment length G_t from observed trajectory $\tau_{1:t}$, so as to remove irrelevant data in $\tau_{1:t}$ for belief context inference. This can be formally expressed by the following equation:

$$p(c_t, G_t|\tau_{1:t}) = p(c^{t-G_t+1:t}|G_t, \tau_{t-G_t:t})p(G_t|\tau_{1:t}) \tag{1}$$

where $\tau_{t_0:t} = (s_{t_0}, a_{t_0}, r_{t_0} \cdots, s_{t-1}, a_{t-1}, r_{t-1}, s_t)$[1], and $c^{t-G_t+1:t}$ refers to the latent context for the whole segment from $t - G_t + 1$ to t. In Equation 1, we separately estimate the posterior $p(c^{t-G_t+1:t}|G_t, \tau_{t-G_t:t})$ of latent context given known segment structure G_t, and the posterior of segment length $p(G_t|\tau_{1:t})$.

3.1.1 Approximate Inference for Latent Context under Known Segment Structure

In this part, we focus on estimating $p(c^{t-G_t+1:t}|G_t, \tau_{t-G_t:t})$, which is the posterior of the latent context under known segment structure G_t. We use the variational inference framework to approximate the true posterior. To be specific, we use an posterior inference network q_ϕ to infer the belief context in segment $[t - G_t + 1 : t]$: $q_\phi(c^{t-G_t+1:t}|G_t, \tau_{t-G_t:t})$. The variational lower bound for the log-likelihood of the current segment is given by:

$$\begin{aligned}
&\log p(\tau^X_{t-G_t:t}|a_{t-G_t:t-1}, s_{t-G_t}, G_t) \\
\geq\ &\mathbb{E}_{q_\phi(c^{t-G_t+1:t}|G_t,\tau_{t-G_t:t})}\left[\log p_\theta(\tau^X_{t-G_t:t}|G_t, c^{t-G_t+1:t}, a_{t-G_t:t-1}, s_{t-G_t})\right] \\
&- \mathbf{D}_{KL}(q_\phi(c^{t-G_t+1:t}|G_t, \tau_{t-G_t:t})\|p(c^{t-G_t+1:t})) \coloneqq \mathcal{J}^t_{Model}(G_t)
\end{aligned} \tag{2}$$

where p_θ denotes the decoder and $\tau^X_{t-G_t:t} = (\tau_{t-G_t:t}, x_{t-G_t+1:t})$. For the detailed derivation, please see Appendix A.1.

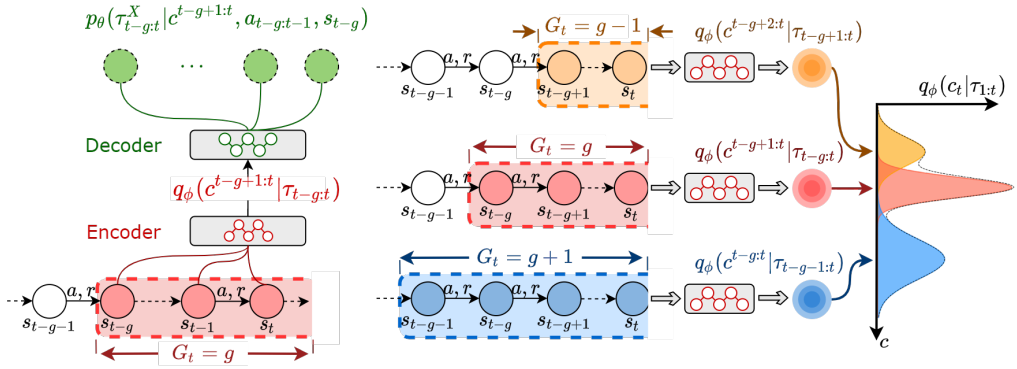
The reconstruction term of Equation 2 can be factorized as:

$$\begin{aligned}
&\log p_\theta(\tau^X_{t-G_t:t}|G_t, c^{t-G_t+1:t}, a_{t-G_t:t-1}, s_{t-G_t}) \\
=\ &\sum_{i=t-G_t+1}^{t} \log p_\theta(x_i, s_i, r_{i-1}|G_t, c^{t-G_t+1:t}, s_{i-1}, a_{i-1})
\end{aligned} \tag{3}$$

which is the sum of log-probablity of transitions under context sampled from q_ϕ. The term $\mathbf{D}_{KL}(q_\phi||p)$ is the KL-divergence between our variational posterior q_ϕ and the prior over the belief context. For the prior $p(c^{t-G_t+1:t})$, we use previous posterior at timestep $t-1$ if the context remains unchanged at timestep t, or $\mathcal{N}(0, I)$ otherwise.

Unlike most previous methods which assume invariant context within an episode or markovian context at each time step, our method 1) takes only observed data within current segment $\tau_{t-G_t:t}$ as input for encoder q_ϕ, 2) reconstructs only data within current segment $\tau^X_{t-G_t:t}$ at the output for decoder $\log p_\theta$. Our method is naturally motivated by the piecewise-stable assumption on context dynamics. By removing irrelevant data outside the current segment that are generated by past unassociated context, our method can estimate the current latent context more accurately.

[1]This definition makes $\tau_{t-G_t:t}$ only contain observed data that belong to the current segment.



(a) Encoder-Decoder Architecture (b) Belief Context as a Mixture Distribution

Figure 3: An overview of SeCBAD. (a) Encoder-decoder architecture for belief context inference given current segment length $G_t = g$: the encoder q_ϕ takes the recent g steps in the current red segment as input, and infer the belief context $q_\phi(c^{t-g+1:t}|\tau_{t-g:t})$, and the decoder takes in sampled context c from q_ϕ and decode the trajectory segment $\tau^X_{t-g:t}$ (the green nodes). (b) Belief context $q_\phi(c_t|\tau_{1:t})$ as mixture distribution by considering belief context $q_\phi(c^{t-g+1:t}|\tau_{t-g:t})$ for all possible structures according to the posterior $p(G_t|\tau_{1:t})$.

3.1.2 Iterative Inference for the Segment Length

In this part, we focus on estimating $p(G_t|\tau_{1:t})$, which is the posterior for current segment length, given $p(c^h|\tau_{t-G_t:t})$, the posterior of context under given segment structure. To compute this posterior distribution, we first compute the joint distribution $p(G_t, \tau_{1:t})$ recursively based on $p(G_{t-1}, \tau_{1:t-1})$ as follows[2]:

$$p(G_t = i, \tau_{1:t}) = \sum_{k=1}^{t-1} p(G_{t-1} = k, \tau_{1:t-1}) \cdot p(G_t = i|G_{t-1} = k) \cdot p(s_t, a_{t-1}, r_{t-1}|\tau_{t-i:t-1}) \quad (4)$$

The three major components in Equation 4 in turn are the previous joint distribution, the evolution prior, and the observation probability. The previous joint distribution is iteratively provided at the beginning of each time step. The evolution prior $p(G_t = i|G_{t-1} = k)$ measures the prior knowledge on the segment length G_t given the segment length of the previous time step $G_{t-1} = k$, where either $G_t = G_{t-1} + 1$ or $G_t = 1$ holds.

As for the observation probability which is the third term of Equation 4, it measures how likely the observations show up given the history of the segment. To be specific, we have [3]

$$p(s_t, a_{t-1}, r_{t-1}|\tau_{t-i:t-1}) = K\mathbb{E}_{p(c^{t-i+1:t}|G_t=i,\tau_{t-i:t-1})}\left[p(s_t, r_{t-1}|s_{t-1}, a_{t-1}, c^{t-i+1:t})\right] \quad (5)$$

In Equation 5, the term $p(s_t, r_{t-1}|s_{t-1}, a_{t-1}, c^{t-i+1:t})$ estimates the data likelihood for the next state-reward pair, where c is drawn from the posterior distribution given the data in the segment before timestep t. To compute the RHS of Equation 5, we sample from $q_\phi(c^{t-i+1:t-1}|G_t = i, \tau_{t-i:t-1})$[4] which is the belief context inferred from the whole history of the segment before timestep t, and use the sampled c the decoder to compute the data likelihood.

Given the joint distribution, the posterior distribution of G_t can be derived as

$$p(G_t = i|\tau_{1:t}) = \frac{p(G_t = i, \tau_{1:t})}{\sum_l p(G_t = l, \tau_{1:t})} \quad (6)$$

[2]For notation simplicity, we omit the condition $G_t = i$ in the term $p(s_t, a_{t-1}, r_{t-1}|\tau_{t-i:t-1})$, i.e. $t - i$ is the start of the segment.

[3]In Equation 5, $K = p(a_{t-1}|s_{t-1})$ is a constant with respect to i and has no impact on the posterior $p(G_t = i|\tau_{1:t})$.

[4]When $i \geq 2$, we can use the belief $q_\phi(c^{t-i+1:t-1}|G_{t-1} = i - 1, \tau_{t-i:t-1})$ to approximate the posterior $p(c^{t-i+1:t}|G_t = i, \tau_{t-i:t-1})$. For the case where $i = 1$, the posterior distribution $p(c^{t-i+1:t}|G_t = i, \tau_{t-i:t-1})$ is actually the prior distribution $p_c(c) = \mathcal{N}(0, I)$, and we can sample c from the prior distribution.

We can also incorporate observable contexts x in the observation probability to improve accuracy during training. See Appendix A.3 for more details.

3.1.3 Belief Context as a Mixture Distribution

Given inferred posterior of G_t in Section 3.1.2, the belief of the latent context at time step t can be derived as

$$b_t(c) = q_\phi(c_t|\tau_{1:t}) = \sum_{g_t} q_\phi(c^{t-g_t+1:t}|G_t = g_t, \tau_{t-g_t:t})p(G_t = g_t|\tau_{1:t}) \tag{7}$$

This mixed probability of $c^{t-g_t+1:t}$ which has taken all possible segment structures into consideration, can now represent the current belief b_t of the latent context c.

3.2 Policy Optimization with Belief Context

Inspired by the belief MDP (Kaelbling et al., 1998), Bayes Adaptive MDP (BAMDP) (Duff, 2002) and recent works on meta RL as task inference (Zintgraf et al., 2020), we incorporate the inferred belief context into the augmented state. At each time step t, the belief latent context is approximated via q_ϕ using Equation 7. Therefore, we define the augmented state as $(s, b) \in \mathcal{S} \times \mathcal{B}$, where $\mathcal{S}$ is the same state space as in LS-MDP and $\mathcal{B}$ is the set of belief latent contexts. Accordingly, we have transition $T^b(s_{t+1}, r_t, b_{t+1}|s_t, a_t, b_t) = p(s_{t+1}, r_t|s_t, a_t, b_t)p(b_{t+1}|s_t, a_t, r_t, s_{t+1}, b_t)$ and reward $R^b(r_t|s_t, b_t, a_t)$ This definition brings advantages in the sense that the information gathering and exploitation tradeoff is no longer a problem under such augmented states, since the transition and reward functions are no longer conditioned on exact c. Now, the policy is defined as $\pi(a|s, b)$ a mapping from the augmented state space to the action space, and the agent's objective is to maximize

$$J_{RL} = \mathbb{E}_{s_0,b_0,\pi,T^b}\left[\sum_{t=0}^{H} \gamma^t R^b(r_t|s_t, b_t, a_t)\right]. \tag{8}$$

3.3 Algorithm and Implementations of SeCBAD

In this section, we describe the overall algorithm and implementation details of SeCBAD. See Figure 3 for an overview of our framework. As shown in Figure 3(a), we use a GRU (Cho et al., 2014) parameterized by ϕ as the recurrent encoder q_ϕ, and distributions in latent context space is assumed to be diagonal Gaussians with mean and variance parameterized by q_ϕ. The decoder includes transition model $p_\theta(s_{t+1}|s_t, a_t, c)$, reward model $p_\theta(r_t|s_t, a_t, c)$ and observable context model $p_\theta(x_t|c)$. The output of all the decoders are Gaussian distributions with mean parameterized by feed-forward neural networks and fixed identity covariance. Then, we estimate $p(G_t|\tau_{1:t})$ using q_ϕ and p_θ as described in Section 3.1.2. As shown in Figure 3(b), we combine the belief context based on different segment according to $p(G_t|\tau_{1:t})$ to get the belief $b_t(c) = q_\phi(c_t|\tau_{1:t})$, where one approach is to provide a total of t mean, covariance and weights as policy input. For simplicity, we choose G_t^* with highest probability in $p(G_t|\tau_{1:t})$ and use the corresponding $q_\phi(c^{t-G_t^*+1:t}|\tau_{t-G_t^*:t})$ as the belief. Empirically, we find this approximation leads to little performance loss. We build our RL algorithm on the top of PPO (Schulman et al., 2017b) to learn the policy $\pi_\psi(a_t|s_t, b_t(c))$, where ψ denotes the parameters in the actor and the critic. We use the objective described in Section 3.2 to optimize the policy.

During deployment, q_ϕ and p_θ are fixed. We first compute the segment posterior $p(G_t|\tau_{1:t})$ using encoder q_ϕ and the transition and reward model p_θ. Then, we estimate the belief $b_t(c)$ and feed it into the policy as input. For more implementation details, please refer to the Appendix A.6.

4 Experiments

In this section, we empirically evaluate SeCBAD on two tasks. We first demonstrate our algorithm in a grid world environment in Section 4.1 to illustrate the significance of incorporating segment structure during inference. For large-scale experiments, we test the proposed algorithm in Section 4.2. The results of multiple challenging tasks show that SeCBAD outperforms baselines in terms of performance and sample efficiency. In Section 4.3, we further provide case studies of agent behaviors and learned latent to gain more insights into the results.

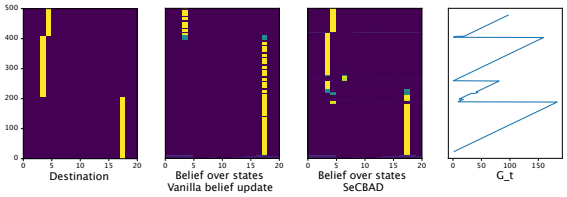


Figure 4: Experiment on grid world with dynamic goal state. We show a case study on the inferred belief. The leftmost figure depicts the ground truth goal position, and the next two figure depicts the belief estimated by vanilla inference and SeCBAD. where the color of one grid measures how the agent believes s^* locates at that grid. The rightmost figure is G_t^* estimated by SeCBAD. The result shows that our belief matches the ground truth more closely.

4.1 Motivating Example

In this experiment, we use a motivating example to show that it is vital for the agent to consider the segment structure to adapt to variations of unknown environment contexts. The agent is in a 4×5 grid world containing 1 goal state s^* and 19 other states. In s^*, the agent can gain $+1$ reward with probability p_0, and 0 reward with probability $1 - p_0$. For other states, the agent can gain $+1$ reward with probability p_1 and 0 reward with probability $1 - p_1$, where $p_0 > p_1$. The actions include up, down, left, right, and none. The specific location of s^* is unobservable to the agent, and may change after a stochastic period.

We use the analytical belief update rule instead of variational inference in this experiment. To be specific, we assume the q_ϕ and p_θ is known and fixed and then infer $p(G_t|\tau_{1:t})$ and the belief $b_t(c)$, which represents the belief distribution of the location of s^*. The belief update rule is detailed in Appendix A.5. With the analytical belief update rule, a theoretically optimal belief b_t is computed for each environment step t. We concatenate (s_t, b_t) together as the augmented state. We may expect that a policy trained upon this augmented state will achieve better performance if b_t is an accurate enough guess to the true location of the goal state. In such a way, we aim to focus on the effectiveness of inferring segment structure, in terms of how inferring segment structure can arrive at a meaningful belief state. We compare our algorithm with vanilla inference, where the agent updates belief without considering segment structure, like in Zintgraf et al. (2020); Rakelly et al. (2019). For both of two routines of updating belief state, we let the agent move along a same path. By moving along a same path, we mean the agent starts from the same location and then take the same action in each environment step for two belief update rules.

Figure 4 illustrates how SeCBAD and the vanilla inference baseline update their belief over states. It is shown that the belief $b_t(c)$ of SeCBAD closely matches with the actual goal state s^*. Also, compared with SeCBAD, the vanilla inference baseline needs more steps to detect the changes since it needs to correct the deviated prior belief. This result supports that incorporating segment structure into inference like SeCBAD is significant for estimating accurate belief in fast varying environments.

4.2 Locomotion Control Tasks with Varying Contexts

In this section, we show that SecBAD is able to tackle more complex tasks by testing the algorithm on several challenging control tasks with varying contexts. We conduct the experiments on modified MuJoCo (Todorov et al., 2012) tasks. These environments are commonly used in previous works (Zintgraf et al., 2020; Rakelly et al., 2019), and we further modify the contexts to provide challenges corresponding to LS-MDP as mentioned in Section 2. As for Ant Direction and Cheetah Direction, the policy needs to move towards the target direction as fast as possible. As for Ant Velocity and Cheetah Velocity, the policy then needs to additionally move at the speed of the target velocity besides the target direction. For Cheetah Goal, the policy needs to navigate to varying goals and stay there. For Ant environments, the direction and velocity are specified on a 2-dimensional plane, and for half cheetah environments, the direction and velocity are 1-dimensional. In these environments, the contexts refer to the target velocity, target direction, or location of the goal. To make it more challenging, we further add noises to contexts within each segment. Please refer to Appendix A.6 for more details. As for baselines, we select one representative algorithm for each type in Figure 2 to compare since to the best of our knowledge, few existing works study the same setting as we do. We use VariBAD (Zintgraf et al., 2020) to represent those methods assuming one context in the episode, and use FANS-RL (Feng et al., 2022) to represent those methods assuming Markovian

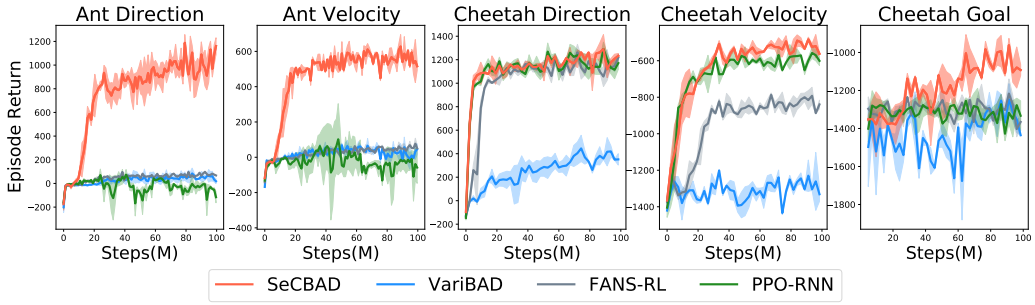


Figure 5: Performance curves on 5 MuJoCo environments with variations in contexts. SeCBAD achieves better performance and sample efficiency in various challenging control tasks.

contexts. Since LS-MDP is a special case of POMDP, we include a POMDP baseline PPO-RNN (Hausknecht and Stone, 2015) for comparison. The results are provided in Figure 5.

The experiment results empirically show that SeCBAD achieves superior performance and sample efficiency on challenging control tasks. As illustrated in Figure 5, SeCBAD achieves higher scores than baselines. We provide some insights into the results. For the method assuming that contexts stay the same within an episode, (Zintgraf et al., 2020) uses the learned latent contexts to decode the whole trajectory including transitions and rewards in other segments. This reconstruction mismatch may lead to averaged latent contexts so that the policy cannot act correspondingly (see detailed analysis and the ablation study in Appendix A.7.3.) For Markovian contexts baseline (Feng et al., 2022) and POMDP baseline (Hausknecht and Stone, 2015), they can perform relatively well on tasks where the contexts can be inferred from only one step transition (i.e., Cheetah Direction and Cheetah Velocity). However, for more complex tasks where the contexts need more steps to infer, SeCBAD significantly outperforms these two baselines (Feng et al., 2022; Hausknecht and Stone, 2015), which proves that the specially designed joint inference component in SeCBAD is effective and can help improve the performance.

To better understand the proposed algorithm, we conduct a series of ablation studies. In Appendix A.7.2, we study the effects of inaccurate prior on SeCBAD. In Appendix A.7.4, we study the implementation choices on how to use $p(G_t|\tau_{1:t})$. In Appendix A.7.5, we further test SeCBAD against different levels of noises within each segment.

4.3 Case Studies on the Segment Structure

In order to provide more insights into the results, we present a case study on Ant Direction in this section. We show the behavior and learned latent along with the inferred segment of a randomly selected episode during testing of the SeCBAD algorithm on Ant Direction in Figure 6.

In the first row, we exhibit the task direction in orange as well as the agent's actual direction in blue. It shows that SeCBAD can rapidly detect and adapt to context changes. We also show the learned latent contexts in the second row and the inferred segment structure in the third row. From the third row, it can be seen that the agent can detect the segment in time as the inferred segment structure closely matches the ground truth segment structure. As for the segment starting from the 430th environment step, the change in goal direction is negligible so that the agent automatically merges the two segments. Since the latent contexts are calculated according to the inferred G_t^*, the latent contexts in the second-row change as the segment changes while staying stable within each segment. This proves that the agent can be aware of the changes in unknown contexts and quickly adapt to the changes. We provide case studies of baselines and detailed analysis in Appendix A.7.1.

5 Related works

Our work is closely related to **non-stationary RL**, where the transition and reward functions may change over time. Most existing works on non-stationary RL focus on inter-episode non-stationarity (Xie et al., 2021; Chandak et al., 2020a,b; Xie et al., 2022; Sodhani et al., 2021; Alegre et al., 2021; Poiani et al., 2021; Al-Shedivat et al., 2017), some of which adopt contextual MDP as formulation (Hallak et al., 2015). Recently there are some works considering intra-episode non-stationarity (Ren et al., 2022; Kamienny et al., 2020; Kumar et al., 2021; Nagabandi et al., 2018;

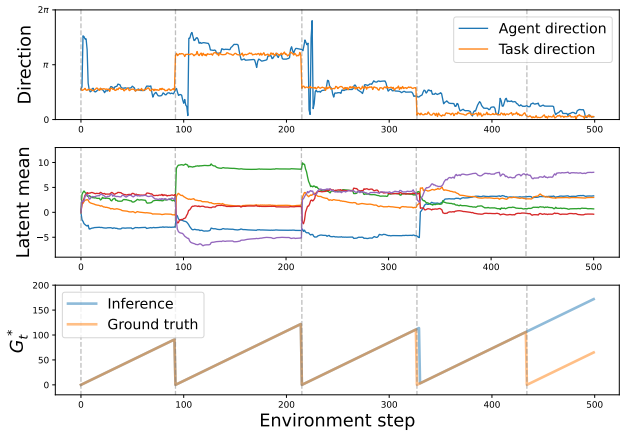


Figure 6: A case study on Ant Direction of SeCBAD. In the first row, we show the behavior of the agent with the goal direction in orange and the actual agent direction in blue. In the second row, we show the learned latent contexts with different colors corresponding to different dimensions. In the last row, we plot the G_t^* inferred by SeCBAD.

Feng et al., 2022). Ren et al. (2022) assume the latent context to be finite and Markovian, while Feng et al. (2022) assume the latent context is Makovian and the environment can be modeled as a factored MDP. In contrast to existing works on non-stationary MDP, we assume piecewise-stable context with abrupt changes within an episode, which is more realistic and can capture a wide range of real-world applications. To rapidly adapt to dynamic context, the agent needs to continuously perform information gathering behavior. **Bayesian RL** (Duff, 2002; Zintgraf et al., 2020; Fellows et al., 2021) is an elegant framework to optimally tradeoff the exploration and exploitation in an unknown and stationary MDP. As a special type of **Belief MDP** (Kaelbling et al., 1998), Bayes Adaptive MDP (BAMDP) (Duff, 2002) maintains a belief over the environment and uses this belief to augment the state. Our model can be viewed as a special case of belief MDP, where we only maintain belief over latent context to trade off between information gathering and exploitation. To accurately infer belief context, we adopt the **variational inference**, which has been adopted by many existing works in RL for task inference (Rakelly et al., 2019; Zhao et al., 2020; Humplik et al., 2019; Poiani et al., 2021; Zintgraf et al., 2020) or context inference (Xie et al., 2021; Feng et al., 2022; Ren et al., 2022). However, none of these methods suit our setting. We perform joint inference over latent context and segment structure from observed data, so as to remove irrelevant data for more accurate belief context inference. LS-MDP can also be viewed as a special case of **POMDP**. Recently, progress has been made in learning the latent dynamics model (Krishnan et al., 2015; Karl et al., 2016; Doerr et al., 2018; Buesing et al., 2018; Ha and Schmidhuber, 2018; Han et al., 2019; Hafner et al., 2019b,a). Theoretically, it is possible to perform optimally only using recurrent neural networks (RNNs) like Hausknecht and Stone (2015) since the whole history has been taken into consideration. However, it has been shown that (Hafner et al., 2019b) introducing more structured information will significantly enhance the performance. In this paper, we exploit the addtional assumption of LS-MDP to infer the context based on segment structure. The experiments show that SeCBAD can achieve better performance compared with existing methods.

6 Conclusion

In this paper, we propose SeCBAD, a **Se**gmented **C**ontext **B**elief **A**ugmented **D**eep RL method to deal with piecewise-stable context in non-stationary environments. Piecewise-stable context is quite common in a wide range of real-world applications. Compared with existing methods, our method can automatically detect the segment structure, which reflects when the context changes abruptly. The detected segment structure can be further used to compute context belief with only relevant observed data. To the best of our knowledge, this is the first method that can model and leverage piecewise-stable context in reinforcement learning to help the agent adapt to environment change. With inferred belief context, our RL agents can quickly detect and adapt to abrupt changes in a gridwold environment and mujoco tasks with piecewise-stable context. For future work, we plan to leverage various deep learning techniques to improve SeCBAD, which includes replacing the GRU encoder by Transformer to better capture the long-term dependency in the input trajectory segment.

References

Akhtar, Z., Nam, Y. S., Govindan, R., Rao, S., Chen, J., Katz-Bassett, E., Ribeiro, B., Zhan, J., and Zhang, H. (2018). Oboe: Auto-tuning video abr algorithms to network conditions. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 44–58.

Al-Shedivat, M., Bansal, T., Burda, Y., Sutskever, I., Mordatch, I., and Abbeel, P. (2017). Continuous adaptation via meta-learning in nonstationary and competitive environments. *arXiv preprint arXiv:1710.03641*.

Alegre, L. N., Bazzan, A. L., and da Silva, B. C. (2021). Minimum-delay adaptation in non-stationary reinforcement learning via online high-confidence change-point detection. *arXiv preprint arXiv:2105.09452*.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2012). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, Vol. 47:253–279. cite arxiv:1207.4708.

Buesing, L., Weber, T., Racaniere, S., Eslami, S., Rezende, D., Reichert, D. P., Viola, F., Besse, F., Gregor, K., Hassabis, D., et al. (2018). Learning and querying fast generative models for reinforcement learning. *arXiv preprint arXiv:1802.03006*.

Chandak, Y., Jordan, S., Theocharous, G., White, M., and Thomas, P. S. (2020a). Towards safe policy improvement for non-stationary mdps. *Advances in Neural Information Processing Systems*, 33:9156–9168.

Chandak, Y., Theocharous, G., Shankar, S., White, M., Mahadevan, S., and Thomas, P. (2020b). Optimizing for the future in non-stationary mdps. In *International Conference on Machine Learning*, pages 1414–1425. PMLR.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Doerr, A., Daniel, C., Schiegg, M., Duy, N.-T., Schaal, S., Toussaint, M., and Sebastian, T. (2018). Probabilistic recurrent state-space models. In *International Conference on Machine Learning*, pages 1280–1289. PMLR.

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

Fellows, M., Hartikainen, K., and Whiteson, S. (2021). Bayesian bellman operators. *Advances in Neural Information Processing Systems*, 34.

Feng, F., Huang, B., Zhang, K., and Magliacane, S. (2022). Factored adaptation for non-stationary reinforcement learning. *arXiv preprint arXiv:2203.16582*.

Ha, D. and Schmidhuber, J. (2018). World models. *arXiv preprint arXiv:1803.10122*.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. (2019a). Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. (2019b). Learning latent dynamics for planning from pixels.

Hallak, A., Di Castro, D., and Mannor, S. (2015). Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*.

Han, D., Doya, K., and Tani, J. (2019). Variational recurrent models for solving partially observable control tasks.

Hausknecht, M. and Stone, P. (2015). Deep recurrent q-learning for partially observable mdps. In *2015 aaai fall symposium series*.

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. (2017). Rainbow: Combining improvements in deep reinforcement learning. cite arxiv:1710.02298Comment: Under review as a conference paper at AAAI 2018.

Humplik, J., Galashov, A., Hasenclever, L., Ortega, P. A., Teh, Y. W., and Heess, N. (2019). Meta reinforcement learning as task inference. *arXiv preprint arXiv:1905.06424*.

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

Kamienny, P.-A., Pirotta, M., Lazaric, A., Lavril, T., Usunier, N., and Denoyer, L. (2020). Learning adaptive exploration strategies in dynamic environments through informed policy regularization. *arXiv preprint arXiv:2005.02934*.

Karl, M., Soelch, M., Bayer, J., and Van der Smagt, P. (2016). Deep variational bayes filters: Unsupervised learning of state space models from raw data. *arXiv preprint arXiv:1605.06432*.

Krishnan, R. G., Shalit, U., and Sontag, D. (2015). Deep kalman filters. *arXiv preprint arXiv:1511.05121*.

Kumar, A., Fu, Z., Pathak, D., and Malik, J. (2021). Rma: Rapid motor adaptation for legged robots.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. cite arxiv:1312.5602Comment: NIPS Deep Learning Workshop 2013.

Nagabandi, A., Finn, C., and Levine, S. (2018). Deep online learning via meta-learning: Continual adaptation for model-based rl. *arXiv preprint arXiv:1812.07671*.

OpenNetLab (2021). Alphartc. In *https://github.com/OpenNetLab/AlphaRTC/*.

Poiani, R., Tirinzoni, A., and Restelli, M. (2021). Meta-reinforcement learning by tracking task non-stationarity. *arXiv preprint arXiv:2105.08834*.

Rakelly, K., Zhou, A., Finn, C., Levine, S., and Quillen, D. (2019). Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pages 5331–5340. PMLR.

Ren, H., Sootla, A., Jafferjee, T., Shen, J., Wang, J., and Bou-Ammar, H. (2022). Reinforcement learning in presence of discrete markovian context evolution. *arXiv preprint arXiv:2202.06557*.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., and Moritz, P. (2015). Trust region policy optimization. In Bach, F. R. and Blei, D. M., editors, *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1889–1897. JMLR.org.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017a). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

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

Sodhani, S., Meier, F., Pineau, J., and Zhang, A. (2021). Block contextual mdps for continual learning. *arXiv preprint arXiv:2110.06972*.

Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE.

Xie, A., Harrison, J., and Finn, C. (2021). Deep reinforcement learning amidst continual structured non-stationarity. In *International Conference on Machine Learning*, pages 11393–11403. PMLR.

Xie, A., Sodhani, S., Finn, C., Pineau, J., and Zhang, A. (2022). Robust policy learning over multiple uncertainty sets. *arXiv preprint arXiv:2202.07013*.

Zhang, Y. and Duffield, N. (2001). On the constancy of internet path properties. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pages 197–211.

Zhao, T. Z., Nagabandi, A., Rakelly, K., Finn, C., and Levine, S. (2020). Meld: Meta-reinforcement learning from images via latent state models. *arXiv preprint arXiv:2010.13957*.

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

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Bound Derivation

The variational bound for our model can be derived using importance weighting and Jensen's inequality. We use $\tilde{c} \coloneqq c^{t-G_t+1:t}$ for simplicity:

$$
\begin{aligned}
&\log p(\tau^X_{t-G_t:t} | a_{t-G_t:t-1}, s_{t-G_t}, G_t) \\
&= \log \int p(\tau^X_{t-G_t:t}, \tilde{c} | a_{t-G_t:t-1}, s_{t-G_t}, G_t) \frac{q_\phi(\tilde{c}|G_t, \tau_{t-G_t:t})}{q_\phi(\tilde{c}|G_t, \tau_{t-G_t:t})} d\tilde{c} \\
&= \log \mathbb{E}_{q_\phi} \left[\frac{p(\tau^X_{t-G_t:t}, \tilde{c} | a_{t-G_t:t-1}, s_{t-G_t}, G_t)}{q_\phi(\tilde{c}|G_t, \tau_{t-G_t:t})} \right] \\
&= \log \mathbb{E}_{q_\phi} \left[\frac{p(\tau^X_{t-G_t:t} | \tilde{c}, a_{t-G_t:t-1}, s_{t-G_t}, G_t) p(\tilde{c}|G_t)}{q_\phi(\tilde{c}|G_t, \tau_{t-G_t:t})} \right] \\
&\geq \mathbb{E}_{q_\phi} \left[\log p(\tau^X_{t-G_t:t} | \tilde{c}, a_{t-G_t:t-1}, s_{t-G_t}, G_t) \right] - \mathbf{D}_{KL}(q_\phi(\tilde{c}|G_t, \tau_{t-G_t:t}) \| p(\tilde{c}|G_t)) \\
&= \sum_{i=t-G_t+1}^{t} \mathbb{E}_{q_\phi} \left[\log p(x_i | G_t, \tilde{c}) + \log p(s_i, r_{i-1} | G_t, \tilde{c}, s_{i-1}, a_{i-1}) \right] - \mathbf{D}_{KL}\left(q_\phi \| p(\tilde{c}|G_t)\right)
\end{aligned}
$$

A.2 Algorithm Framework

In this Section, we provide the detailed training and evaluation algorithms of SeCBAD. The training algorithm is detailed in Algorithm 1. The evaluation algorithm is detailed in Algorithm 2.

Algorithm 1 SeCBAD Training Algorithm

Input: buffer $\mathcal{B}$, imagination horizon H, interacting step T, batch size B, batch length L, number of trajetories N.
Initialize buffer $\mathcal{B}$ with S random seed episodes.
while not converged **do** ▷ *Parameter Optimization*
  for $c = 1, \dots, C$ **do**
    Draw B data sequences $\{(s_t, a_t, r_t, x_t)\}_{t=k}^{k+L}$ from $\mathcal{B}$
    Calculate $p(G_t|\tau_{1:t})$ using Equation (6).
    Sample $g \sim p(G_t|\tau_{1:t})$.
    Infer belief state $q_\phi(c^{t-G_t+1:t}|G_t = g, \tau_{t-G_t:t})$.
    Sample $k \in \mathbb{N}$ from $[t-g+1, t]$. ▷ *Calculate the ELBO: sample 1 reconstruction step*
    Predict observable context: $p_\theta(x_k|c, G_t = g)$
    Predict reward: $p_\theta(r_{k-1}|c, s_{k-1}, a_{k-1}, G_t = g)$, and state $p_\theta(s_k|c, s_{k-1}, a_{k-1}, G_t = g)$.
    Update ϕ, θ using Equation (2).
    Update ψ using Equation (8).
  end for
  Reset environment and get s_1, x_1.
  for $t = 1, \dots, T$ **do** ▷ *Data Collection*
    Calculate $p(G_t|\tau_{1:t})$ using Equation (6).
    Calculate belief $b_t(c)$ using Equation (7).
    Compute $a_t \sim \pi_\psi(a_t|s_t, b_t)$ with action model.
    Add exploration noise to action.
    Execute a_t and get x_{t+1}, s_{t+1}, r_t.
  end for
  Add experience to buffer $\mathcal{B} = \mathcal{B} \cup \{(s_t, a_t, r_t, x_t)_{t=1}^{T}\}$
end while

Algorithm 2 SeCBAD Evaluation Algorithm

Input: interacting step T, encoder q_ϕ, decoder p_θ, policy π_ψ.
Reset environment and get s_1.
for $t = 1, \dots, T$ **do**
  Calculate $p(G_t|\tau_{1:t})$ using Equation (6).
  Choose $g = \arg\max p(G_t|\tau_{1:t})$.
  Compute $b_t(c) = q_\phi(c^{t-G_t+1:t}|G_t = g, \tau_{t-G_t:t})$.
  Compute $a_t \sim \pi_\psi(a_t|s_t, b_t)$ with action model.
  Execute a_t and get x_{t+1}, s_{t+1}, r_t.
end for

A.3 Include x in observation probability

In Section 3.1.2, we recursively estimate the posterior of current segment length $p(G_t|\tau_{1:t})$, which conditions the segment length on the observation $\tau_{1:t}$ as conditions to keep consistent with the case in evaluation. However, During training, we can use more information to help us get a more accurate estimation. For instance, we can include the observable contexts x into $\tau_{1:t}$, like $\tau^X_{1:t}$ and update the corresponding equations as follows:

$$p(G_t = i, \tau^X_{1:t}) = \sum_{k=1}^{t-1} p(G_{t-1} = k, \tau^X_{1:t-1}) \cdot p(G_t = i|G_{t-1} = k) \cdot p(s_t, x_t, a_{t-1}, r_{t-1}|\tau^X_{t-i:t-1})$$

The difference mainly lies in the observation probability term:

$$\begin{aligned} &p(s_t, x_t, a_{t-1}, r_{t-1}|\tau^X_{t-i:t-1}) \\ &= K\mathbb{E}_p \left[p(s_t, r_{t-1}|s_{t-1}, a_{t-1}, c^{t-i+1:t})p(x_t|, c^{t-i+1:t})\right] \end{aligned}$$

The observation probability term now estimates the data likelihood not only for the next state-reward pair, but also for the next observable context x_t.

There are many other choices here for how to choose the information during training. It is also possible to only incorporate the observable contexts x instead of using $\tau^X_{1:t}$ for simplicity. We can still use the online inference method like we stated above, or we can use some more simple and heurstic methods to derive the posterior. Since we can directly access x from the simulator, it is possible for us to calculate the posterior in advance, which can help to save a lot of computation during training time.

A.4 Discussion with Related Works

Recently, some works (Ren et al., 2022; Feng et al., 2022) assume intra-episode context changes and model context c_t at each time step, more specifically with Markovian transition on latent contexts. Since Markovian assumptions are made in latent context space, these methods are theoretically able to capture the non-Markovian pattern in observable context space. However, as discussed in our paper, one of the key challenges in deploying RL to real-world applications is to adapt to variations of unknown environment contexts that stay stable for a stochastic period. In contrast to general non-stationary environments, our problem setting assumes piece-wise stable context as special problem structure, which can be further exploited to improve performance against methods for general non-stationary environments. By contrast, we propose to jointly infer the segment structure as well as the segment latent context. By doing so, we can avoid solving non-stationary environments in general case. In our paper, we implement a one-step VariBAD to represent the methods that model context c_t at each time step. For each time step, we use the whole trajectory as input just like VariBAD. However, instead of decoding the whole trajectory, we only make the agent to decode the current time step.

A.5 Bayesian Update Rule

In Section 4.1, we use the Bayesian belief update rule to calculate the belief distribution. Let K denotes the number of grids, then $b(\theta) \in \mathbb{R}^K$ is a simplex which refers to the probability location of the goal state and therefore a categorical distribution. As for the belief update rule, we have

$$b'(\theta) = b(\theta|s, a, s', r) \propto b(\theta)p(r|s, a, \theta)$$

In above equation, we omit the transition term since the transition keeps stationary across the episode. As for the reward term $p(r|s, a, \theta)$, it can be derived through the definition. For the k-th grid, suppose the observed state is the k'-th grid. Then, if $k = k'$, we can get the reward term as $p(r = 1|\theta = k, s = k', a) = p_1$ and $p(r = 0|\theta = k, s = k', a) = 1 - p_1$. If $k \neq k'$, we can get the reward term as $p(r = 1|\theta = k, s = k', a) = p_0$ and $p(r = 0|\theta = k, s = k', a) = 1 - p_0$. In this way, we can obtain the accurate belief without approximation.

A.6 Hyper parameters and Implementation Details

Network Architecture For Half-Cheetah environments, we use a GRU(Cho et al., 2014) with 128 units as the dynamics model of the encoder q_ϕ. For each time step, q_ϕ receives an encoded state, action and reward as input. The encoded state, action and reward are the output three single layer fully connected networks of respective size $[16, 32, 16]$ with ReLU as activation function. We assume the latent distribution are 5-dimensional Gaussians with predicted mean and standard deviation. As for the transition model $p_\theta(s_{t+1}|s_t, a_t, c)$, reward model $p_\theta(r_t|s_t, a_t, c)$ and observable context model $p_\theta(x_t|c)$, the output of all the decoders are Gaussian distributions with mean parameterized by fully connected network (with hidden size $[64, 32]$ and ReLU activation) and fixed identity covariance. As for the Ant-Cheetah environments, we use 10-dimensional Gaussians as the latent distribution. The other settings are kept the same as in Half-Cheetah environments.

As for the policy training part, we adopt PPO (Schulman et al., 2017b) to learn the policy $\pi_\psi(a_t|s_t, b_t(c))$. The actor and critic are parameterized by fully connected network of size $[128, 128]$ with Tanh as activation function.

Training Details The inputs of both actor and critic is s_t and b_t. To parameterize the distribution b_t, we use its mean and standard deviation. For simplicity, we only incorporate one possible segment length g into consideration and this approximation will not cause obvious performance drop according to our test. During training, we randomly sample g from the posterior, while during evaluation, we use the most possible g. Then, we calculate the corresponding belief b_t using g, and use the mean and standard deviation as the inputs. During training, we use the posterior $p(G_t|x_{1:t})$ which conditions only on the observable contexts to simplify the computation. During evaluation, we then use the posterior $p(G_t|x_{1:t})$ like we stated in Section 3.1.2 since we can no longer access the training only information x.

Hyper parameters We train PPO with Adam optimizer with learning rate 7e-4, max gradient norm 0.5, clip parameters 0.1, value loss coefficient 0.5, and entropy coefficient 0.01. We use different training schedule between the VAE part (encoder & decoder) and the policy part. We use two Adam optimizers with different learning rates. The VAE optimizer uses a learning rate of 1e-3, and the policy optimizer uses a learning rate of 7e-4. For the full details of hyper-parameter settings, please refer to the code repository that we will publish soon.

Environment Details We give detailed descriptions of the environments we used. **Ant Direction** environment is built upon the well-known Ant Mujoco environment. For each environment step, we set a target direction in the 2D horizontal plane. The reward function is given as $v_f - 0.2v_v$, where v_f denotes the agent velocity along the target direction, v_v denotes the agent velocity that is vertical to the target direction and is always non-negative. Such reward function pushes the agent moves faster along the target direction while penalizing the agent for the velocity component that is vertical to the target direction. In experiment the target direction is chosen uniformly from $[0, 2\pi]$.

For **Ant Velocity**, we set a target velocity $\boldsymbol{v_t} = (v_{t,x}, v_{t,y})$ in the 2D horizontal plane. Denote the agent velocity as $\boldsymbol{v_a} = (v_{a,x}, v_{a,y})$. We project the agent velocity $\boldsymbol{v_a}$ to $\boldsymbol{v_t}$ to get the forward velocity component $v_f = \boldsymbol{v_t} \cdot \boldsymbol{v_a}/\|\boldsymbol{v_t}\|_2$. The velocity component perpendicular to the target velocity is given by $v_v = \sqrt{\|\boldsymbol{v_a}\|_2^2 - v_f^2}$. If forward velocity $v_f < \|v_t\|_2$, we compute the reward as $v_f - 0.3v_v$, meaning that we expect the agent moves faster along the direction of task velocity, but not along the direction vertical to task velocity. Otherwise when $v_f \geq \|v_t\|_2$, we compute reward as $-v_f + 2\|\boldsymbol{v_t}\|_2 - 0.3v_v$. Observe that if ignore the term $-0.3v_v$ in the reward function, we assign the largest reward when $v_f = \|\boldsymbol{v_t}\|_2$, and reward decreases as v_f deviating from $\|\boldsymbol{v_t}\|_2$, thus promoting the agent follows the target velocity. In experiment, v_{t_x} and v_{t_y} are independently chosen from Uniform$(-3, 3)$ for each segment.

Cheetah Direction, **Cheetah Goal** and **Cheetah Velocity** are modified from Half Cheetah Mujoco environment. The agent can only move along x-axis. In the original version of Half Cheetah, the reward is given by the agent velocity along the positive x-axis. Denote agent velocity as v_a and agent location as x_a. In Cheetah Direction, we set the target direction to positive x-axis or negative x-axis, and the reward is v_a if the target direction is positive x-axis or $-v_a$ otherwise. We choose the positive and negative directions with equal probability. In Cheetah Goal, we set a target location x_t on the x-axis in each environment step, and the reward function is $-|x_a - x_t|$, with x_t sampled from

Uniform$(-5, 5)$. The agent location x_a is also included into the observation space, which is not for the original Half Cheetah, Cheetah Dir and Cheetah Velocity. For Cheetah Velocity, we set a target velocity v_t on the x-axis, the reward is $-|v_t - v_a|$, and v_t is chosen from Uniform$(-5, 5)$.

A.7 Extended Experiment Results

A.7.1 A Case Study on Ant Direction

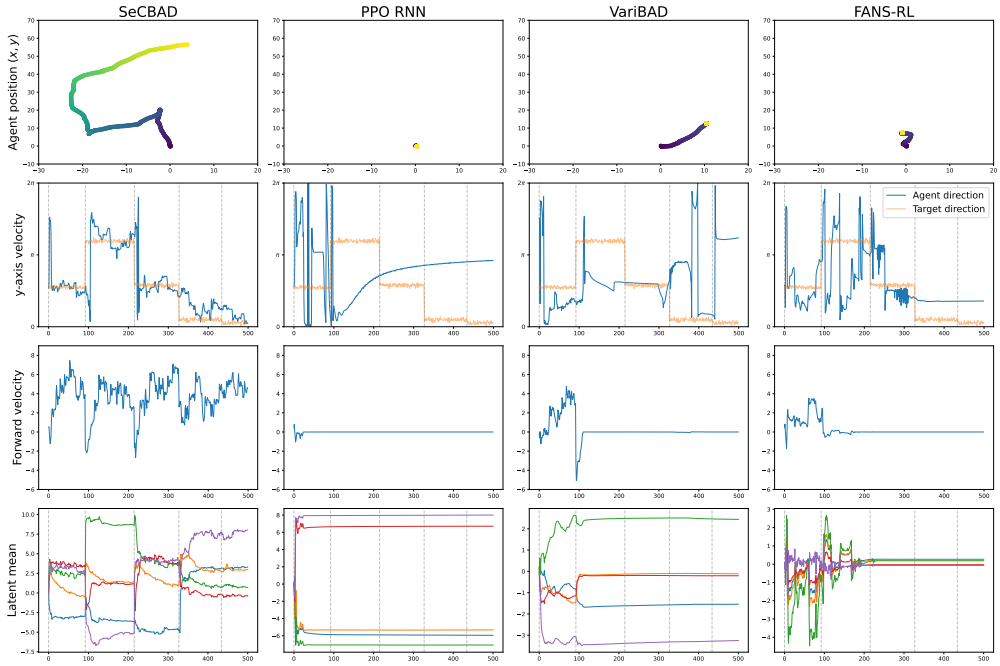


Figure 7: Behavior and the learned latent contexts of different algorithms on Ant Direction

To better understand the outperformance of SeCBAD, we show the behavior of SeCBAD and other baselines and the learned latent contexts in Figure 7.

The first row shows the agent's location through time. It can be seen that the agent of PPO RNN, VariBAD, and Factor MDP choose to stay around a fixed position shortly after starting, while SeCBAD can successfully guide the agent moving around.

In the second row, we show the agent's direction along with the target (task) direction. The orange curve shows the time-varying task direction, and the blue curve shows the agent's actual direction. Since the goal is to maximize the velocity along the task direction, we plot the velocity along the task direction in the third row. It can be concluded that SeCBAD is able to detect and adapt to the variations rapidly, and the agent is indeed moving along the task direction, while other methods fail to detect and adapt to the non-stationary task in time. After detecting the teak direction change, SeCBAD's velocity drops and then grows up as it tries to change the direction, while the velocities of other baselines are close to zero and the learned behaviors are undesirable.

We provide some insights into the results. We plot the mean of the latent in the fourth row of Figure 7. VariBAD (Zintgraf et al., 2020) tries to learn a single context for the whole episode and train the decoder by reconstructing the whole trajectory. This may make the agent learn the same contexts across the episode. Therefore the learned latent mean is smooth and thus uninformative to the changing environment. Please refer to Appendix A.7.3 for detailed analysis. FANS-RL (Feng et al., 2022) assumes that the contexts evolve in a Markovian pattern. PPO-RNN (Hausknecht and Stone, 2015) treats the problem in a general form as a POMDP and leverages little information about the context pattern. It is hard for these two methods to learn a meaning for contexts, especially in complex environments like Ant Direction. However, as for SeCBAD, thanks to the joint inference framework and segment decoder, the learned latent mean can change abruptly when the inferred segment structure changes, which informs the policy that the contexts have changed. The behavior in Figure 7 shows that the agent can learn to adapt to the variations rapidly with this accurate enough belief. We provide more case studies on other tasks in Section 4.2 in Appendix A.7.6.

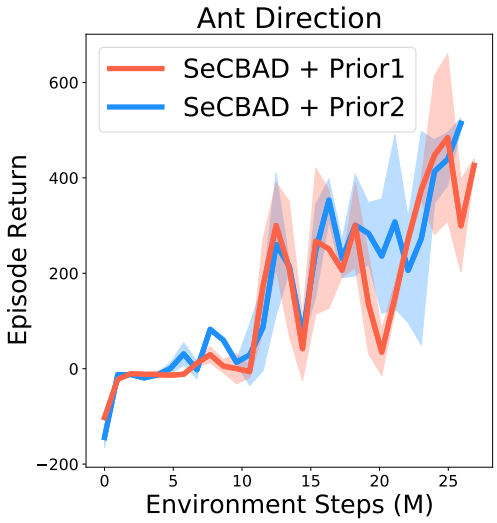


Figure 8: Analysis on the choice of prior

A.7.2 Prior Analysis

In this Section, we provide the analysis on the choice of prior $p(G_t = k|G_{t-1} = i)$ on Ant Direction. The results are provided in Figure 8. We choose two different prior functions. For prior 1, we let $p(G_t = 1|G_{t-1} = i) = \frac{1}{80}$ for any i and $p(G_t = i + 1|G_{t-1} = i) = \frac{79}{80}$. For prior 2, we use a more accurate prior by rolling out the generative process several times and approximate the $p(G_t = k|G_{t-1} = i)$ from the simulated data. The performance using these two priors are shown above.

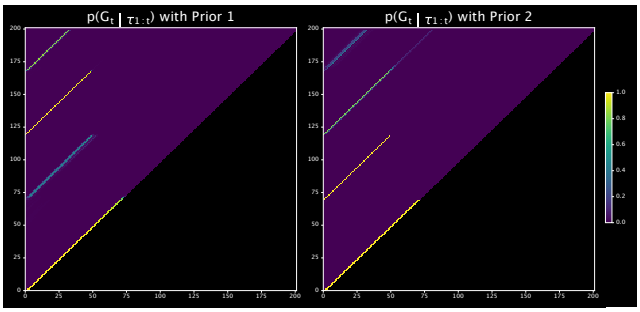


Figure 9: The segment length posterior $p(G_t|\tau_{1:t})$ calculated with two different prior functions.

It can be concluded that, two prior funciton have little performance gap. To gain more insights, we draw the probablity of $p(G_t|\tau_{1:t})$ in Figure 9. The results show that the inferred posterior is very deterministic, and the one with approximate prior is a little bit more blurry, but still deterministic enough. This result proves that, the observation term dominates the probability term and our method is robust with respect to the choice of prior.

A.7.3 Ablation study on the number of segments

In this section, we study how the number of segments affects the performance of SeCBAD and VariBAD (Zintgraf et al., 2020). As analyzed in Section 4.2, we hypothesize that methods assuming that contexts stay the same within an episode like (Zintgraf et al., 2020) may end up learning an averaged latent contexts. We argue that this is not related to the length of the segment but is due to the number of segments in an episode. During reconstruction, (Zintgraf et al., 2020) uses the learned latent contexts at timestep t to decode the transitions and rewards in the whole trajectory. Suppose there are n segments in the episode, then the probability of reconstructing the correct transition and

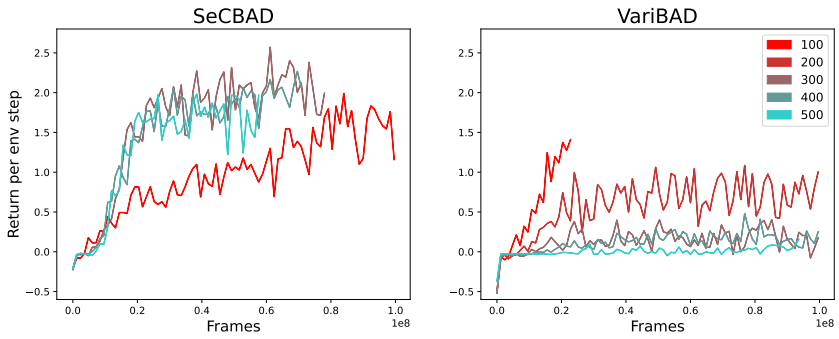


Figure 10: Ablation study on the number of segments. We keep the segment length fixed and adjust the episode length $L = \{100, 200, 300, 400, 500\}$ to adjust the number of segments.

reward (i.e., those in the same segment) is only $\frac{1}{n}$. The noisy even erroneous training signal may exacerbate as n grows and end up in learning averaged latent contexts for the whole episode.

To look further into this hypothesis, we conduct an ablation study on the number of segments. We choose the Ant Direction environment and change the max episode length $L \in \{100, 200, 300, 400, 500\}$ while letting the average segment length be fixed. Then the expected number of segments n may vary. We plot the averaged reward per step instead of the accumulated rewards to cancel out the effect of L and study the effects on n.

As illustrated in Figure 10, for SeCBAD, n has little effect on performance. However, for VariBAD, the performance deteriorates as the grows. The results fit the above analysis and show the significance of joint inference on the segment structure and context belief.

A.7.4 Ablation study on using $p(G_t|\tau_{1:t})$

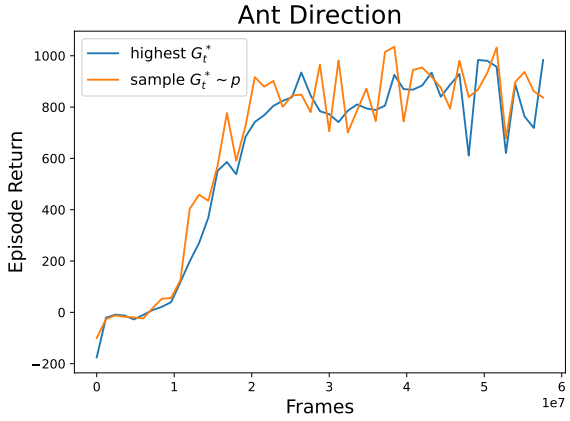


Figure 11: Ablation study on how to use the inferred $p(G_t|\tau_{1:t})$.

During the implementation, we may face a choice on how to use the inferred segment structure $p(G_t^*|\tau_{1:t})$. Since there are t possible choices for $p(G_t|\tau_{1:t})$ in timestep t, it's hard to directly use the full distribution. One option is to use the G_t with the highest probability. Another option is to sample $G_t \sim p$ to approximate the full distribution. In this section, we perform an ablation study on Ant Direction to show the performance of the two options. We keep other parameters fixed and only modify the way using G_t^*. As shown in Figure 11, the performances of the two options are very close. Therefore, for stability, we choose the G_t^* with the highest probability in experiments.

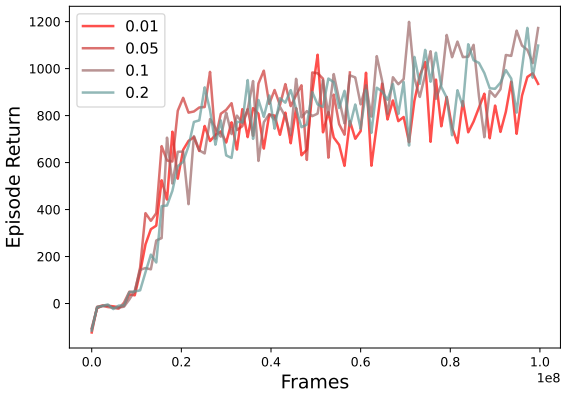


Figure 12: Ablation study on the robustness of SeCBAD to the noise within each segment. We change the standard deviation $\sigma \in \{0.01, 0.05, 0.1, 0.2\}$.

A.7.5 Ablation study on context robustness

In many real-world applications, the contexts are generally piecewise stable. However, within each segment, the contexts might be slightly noisy. Therefore, it is important to test the robustness of the proposed algorithm against the noise within each segment. In this section, we conduct an ablation study on SeCBAD on Ant Direction by adjusting the standard deviation of the noises.

For each segment, we uniformly sample the mean of the contexts. Then, for each timestep within each segment, we assume the contexts follow a Gaussian distribution. We adjust the standard deviation of the Gaussian distribution to model different levels of robustness by setting $\sigma = \{0.01, 0.05, 0.1, 0.2\}$ and keep other parameters fixed. As illustrated in Figure 12, SeCBAD is able to handle noises within each segment since the performances of different σ are very close. This further exhibit the ability of SeCBAD to solve many real-world applications.

A.7.6 More visualizations

In this section, we provide case studies of SeCBAD and other baselines for the rest environments in Section 4.2. For the case study on Ant Direction, we refer to A.7.1. Given each environment, we visualize the model behavior after training for the same number of frames. We use the same random seed for SeCBAD and other three baselines, thus these algorithms are tested on environments with the same trajectory context.

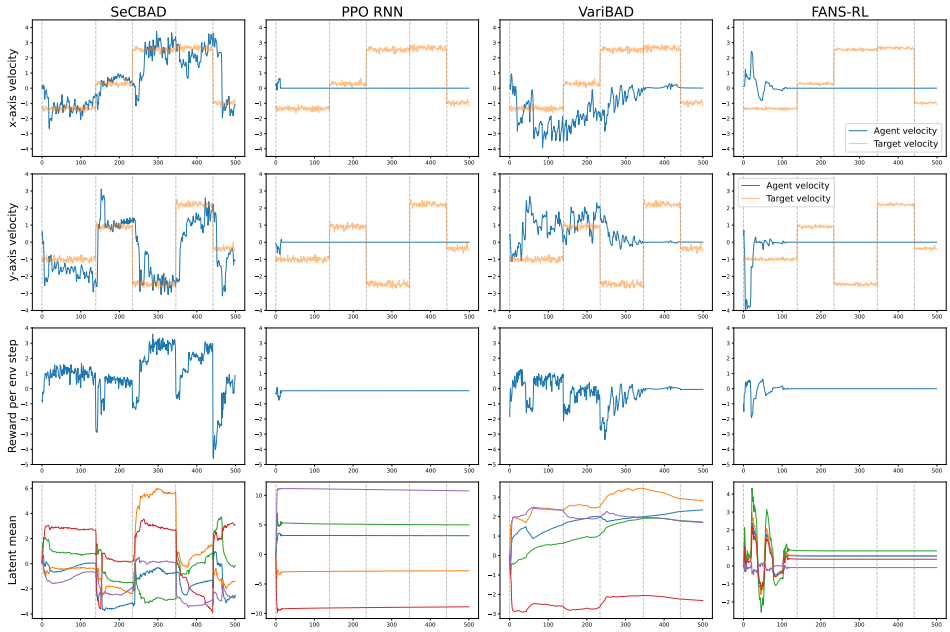


Figure 13: A case study on Ant Velocity.

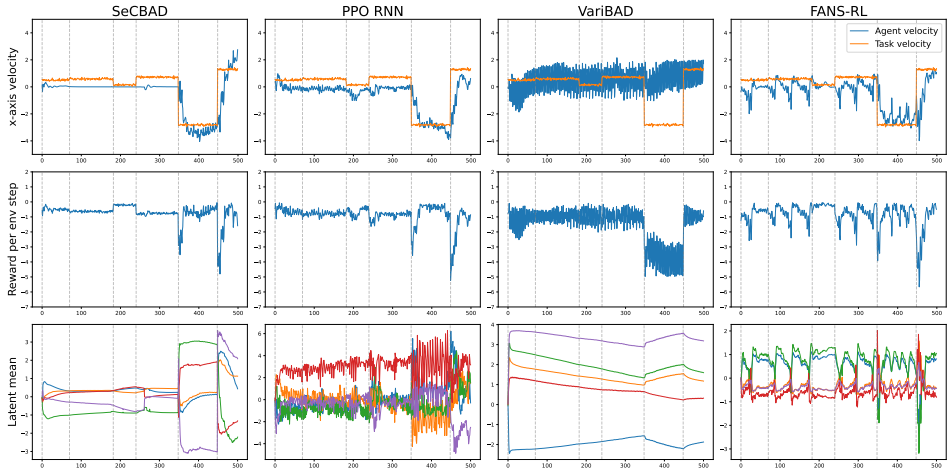


Figure 14: A case study on Half Cheetah Velocity.

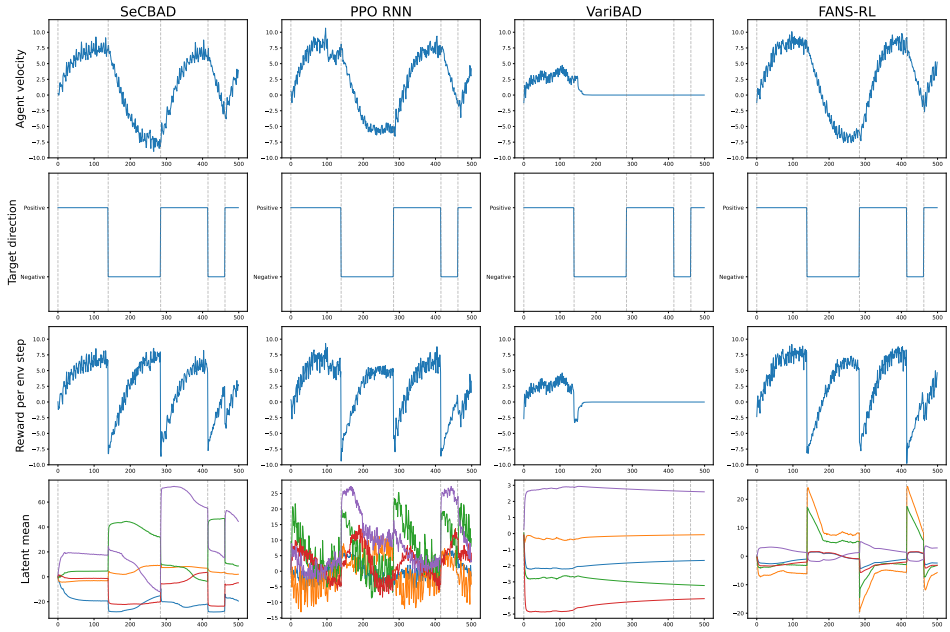


Figure 15: A case study on Half Cheetah Direction.

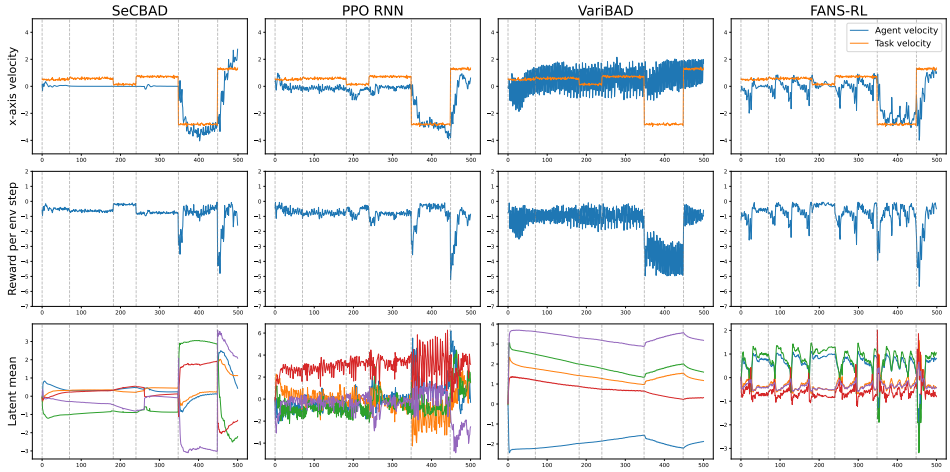


Figure 16: A case study on Half Cheetah Velocity.

A.8 Bandwith Control Tasks for Real Time Communication

To further illustrate the proposed LS-MDP setting can boost the deployment of RL in many real-world applications, we test SeCBAD on a real-world bandwidth control task for real-time communications (RTC) (OpenNetLab, 2021) in this section.

RTC applications, e.g., online audio/video calls and conferences, have been greatly increasing since the global pandemic. The most critical goal in RTC is to provide high Quality of Experience (QoE) for users, including high audio/video bitrate, low end-to-end latency, no video freeze, and few bitrate switches. To achieve this, a bandwidth control module is needed, i.e., the RTC sender needs to decide the bitrate of outstreaming audio/video based on the network status towards the receiver. For example, it would decrease the bitrate when observing a high end-to-end delay, otherwise, the bitrate would be increased. However, the ground truth network conditions are always changing in multiple items, such as available capacity, Round-Trip Time (RTT), and so on. The simple rule *"decrease the bitrate when observed a high delay"* becomes unreasonable when the ground truth RTT is increased. This reveals

the system should rapidly detect and react to the network condition otherwise the users' QoE may suffer.

To test SeCBAD for the bandwidth control problem in RTC, we use AlphaRTC (OpenNetLab, 2021) as our simulator environment. In our reinforcement learning formulation, we use a 7-tuple of current network statistics that is visible to the agent as states s_t, consisting of sending rate, short-term and long-term receiving rate, loss, and delay. The action a_t is the estimated bandwidth. The reward function is formulated as $2R/C - (D - RTT/2) - L - 1$, where R is the receiving rate in the time step, D is the average delay in the time step, L is the packet loss rate, C and RTT are the ground truth capacity and average RTT of the network. The latent context c_t here refers to the fluctuated network condition, in this section, we consider x_t as the ground truth bandwidth capacity C, and the RTT. The agents are trained in AlphaRTC (OpenNetLab, 2021) that simulates real-time communication processes by specifying C and RTT. All the network statistics are normalized to the $(0, 1)$ range.

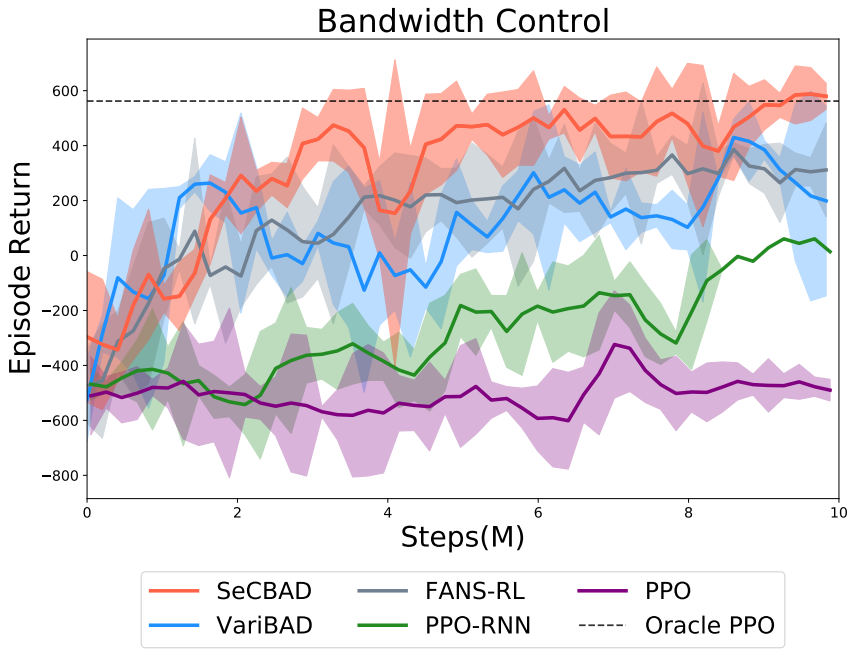


Figure 17: Experiment results on bandwidth control for RTC.

In this experiment, we compare our method with VariBAD (Zintgraf et al., 2020), FANS-RL (Feng et al., 2022), vanilla PPO (Schulman et al., 2017a) and PPO-RNN (Hausknecht and Stone, 2015). We also incorporate oracle PPO scores by incorporating the unobservable contexts into the observable states. All the methods are trained for 10 million steps and the shaded area is across 3 random seeds.

As illustrated in Figure 17, SeCBAD achieves better performance than other baselines and is very close to the oracle PPO baseline score. The performances of VariBAD (Zintgraf et al., 2020) and FANS-RL (Feng et al., 2022) are better than PPO-RNN (Hausknecht and Stone, 2015), but SeCBAD outperforms both of these methods. The results suggest that SeCBAD is able to detect and adapt to the varying contexts more rapidly, which allows the policy to precisely control. This indicates the importance of the joint inference structure.

Figure 18 shows the detailed behavior of different methods under one representative case.

The first row shows the agent's action (in blue) against the true available bandwidth (in orange). The second row shows the latency observed by the agent, which partially represents the mixed effect of another context, RTT. It can be observed that for SeCBAD, the agent is able to detect the change in bandwidth and adapt to the changes in time and produce a stable policy within each segment. The drops in actions are timed to coincide with the observed increasing latency. After realizing that the actual capacity is not changed, the agent can then increase the actions back to the optimal value. However, for other baselines, the actions oscillate a lot, especially for FANS-RL (Feng et al., 2022)

which is not practical since this may lead to frequent bitrate switches. VariBAD (Zintgraf et al., 2020) learns a quite smooth and conservative policy that leaves a large margin between the action and the actual capacity, which may lead to a waste in the network capacity. For PPO-RNN (Hausknecht and Stone, 2015), it produces a rather low action

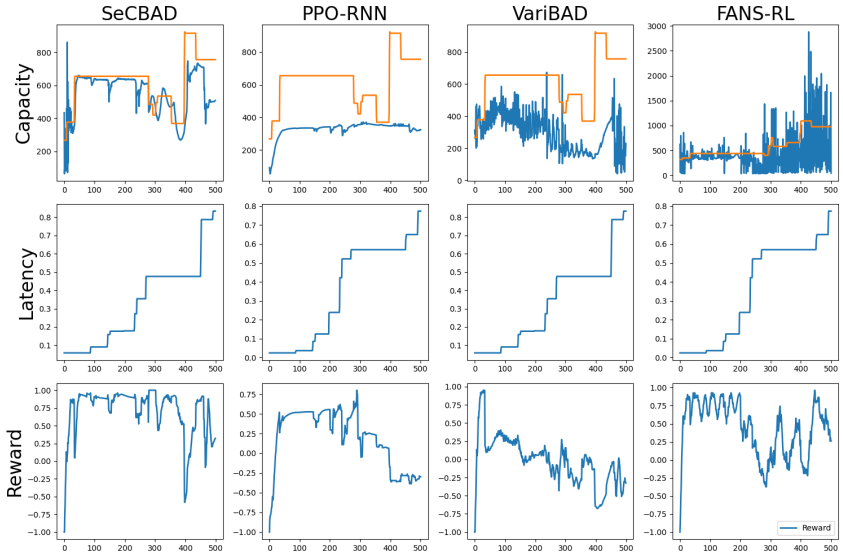


Figure 18: Case study on bandwidth control for RTC.