Factored Adaptation for Non-stationary Reinforcement Learning

Fan Feng[1], **Biwei Huang**[2], **Kun Zhang**[2,3], **Sara Magliacane**[4,5]
[1]City University of Hong Kong [2]Carnegie Mellon University
[3]Mohamed bin Zayed University of Artificial Intelligence
[4]University of Amsterdam [5]MIT-IBM Watson AI Lab
{ffeng1017,sara.magliacane}@gmail.com, biweih@andrew.cmu.edu, kunz1@cmu.edu

Abstract

Dealing with non-stationarity in environments (e.g., in the transition dynamics) and objectives (e.g., in the reward functions) is a challenging problem that is crucial in real-world applications of reinforcement learning (RL). While most current approaches model the changes as a single shared embedding vector, we leverage insights from the recent causality literature to model non-stationarity in terms of individual latent change factors, and causal graphs across different environments. In particular, we propose Factored Adaptation for Non-Stationary RL (FANS-RL), a factored adaption approach that learns jointly both the causal structure in terms of a factored MDP, and a factored representation of the individual time-varying change factors. We prove that under standard assumptions, we can completely recover the causal graph representing the factored transition and reward function, as well as a partial structure between the individual change factors and the state components. Through our general framework, we can consider general non-stationary scenarios with different function types and changing frequency, including changes across episodes and within episodes. Experimental results demonstrate that FANS-RL outperforms existing approaches in terms of return, compactness of the latent state representation, and robustness to varying degrees of non-stationarity.

1 Introduction

Learning a stable policy under non-stationary environments is a long-standing challenge in Reinforcement learning (RL) [1, 2, 3]. While most RL approaches assume stationarity, in many real-world applications of RL there can be changes in the dynamics or the reward function, both *across* different episodes and *within* each episode. Recently, several works adapted Meta-RL methods to learn sequences of non-stationary tasks [4, 5]. However, the continuous MAML [6] adaptation for non-stationary RL [4] does not explicitly model temporal changing components, while TRIO [5] needs to meta-train the model on a set of non-stationary tasks. LILAC [7] and ZeUS [8] leverage latent variable models to directly model the change factors in the environment in a shared embedding space. In particular, they consider families of MDPs indexed by a single latent parameter. In this paper, we argue that disentangling the changes as separate latent parameters and modeling the process with a factored representation improves the efficiency of adapting to non-stationarity.

In particular, we leverage insights from the causality literature [9, 10] that model non-stationarity in terms of individual latent change factors and causal graphs across different environments. We propose Factored Adaptation for Non-Stationary RL (FANS-RL), a factored adaptation framework that jointly learns the causal structure of the MDP and a factored representation of the individual change factors, allowing for changes at discrete timepoints and continuously varying environments. While we provide a specific architecture (FN-VAE), the theoretical framework of FANS-RL can be implemented with

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

different architectures and combined with various RL algorithms. We formalize our setting as a *Factored Non-stationary MDP* (FN-MDP), which combines a Factored-MDP [11, 12, 13] with latent change factors that evolve in time following a Markov process.

We build upon the AdaRL framework [14], a recently proposed fast adaptation approach. AdaRL learns a factored representation that explicitly models changes (i.e., domain-specific components) in observation, dynamics and reward functions across a set of source domains. An optimal policy learnt on the source domains can then be adapted to a new target domain simply by identifying a low-dimensional change factor, without any additional finetuning. FANS-RL extends AdaRL from the stationary case with constant change factors to a general non-stationary framework. Specifically, FANS-RL learns the low-dimensional and time-evolving representations $\boldsymbol{\theta}_t^s$ and $\boldsymbol{\theta}_t^r$ that fully capture the non-stationarity of dynamics and rewards, allowing for continuous and discrete changing functions, both *within-episode* and *across-episode*. Our main contributions can be summarized as:

- We formalize FN-MDPs, a unified factored framework that can handle many non-stationary settings, including discrete and continuous changes, both within and across episodes. We prove that, under standard assumptions, the causal graph of the transition and reward function is identifiable, while we can recover a partial structure for the change factors.
- We introduce Factored Adaptation for Non-Stationary RL (FANS-RL), a general non-stationary RL approach that interleaves model estimation of an FN-MDP and policy optimization. We also describe FN-VAE, an example architecture for learning FN-MDPs.
- We evaluate FANS-RL on simulated benchmarks for continuous control and robotic manipulation tasks and show it outperforms the state of the art on the return, compactness of the latent space representation and robustness to varying degrees of non-stationarity.

2 Factored Non-stationary MDPs

To model different types of non-stationarity in a unified and factored way, we propose Factored Non-stationary Markov Decision Processes (FN-MDPs). FN-MDPs are an augmented form of a factored MDPs [11, 12, 13] with latent change factors that evolve over time following a Markov process. Since the change factors are latent, FN-MDPs are partially observed. We define them as:

Definition 1. *A Factored Non-stationary Markov Decision Process (FN-MDP) is a tuple* $(\mathcal{S}, \mathcal{A}, \Theta^s, \Theta^r, \gamma, \mathcal{G}, \mathbb{P}_s, \mathcal{R}, , \mathbb{P}_{\theta^r}, \mathbb{P}_{\theta^s})$*, where* $\mathcal{S}$ *is the state space,* $\mathcal{A}$ *the action space,* Θ^s *the space of the change factors for the dynamics,* Θ^r *the space of the reward change factors and* γ *the discount factor. We assume* $\mathcal{G}$ *is a Dynamic Bayesian Network over* $\{s_{1,t}, \dots, s_{d,t}, a_{1,t}, \dots, a_{m,t}, r_t, \theta^{\boldsymbol{s}}_{1,t}, \dots, \theta^{\boldsymbol{s}}_{p,t}, \theta^r_{1,t}, \dots, \theta^r_{q,t}\}$*, where* d*,* m*,* p*, and* q *are the dimensions of states, action, change factors on dynamics and reward, respectively. We define the factored state transition distribution* $\mathbb{P}_s$ *as:*

$$\mathbb{P}_s(\boldsymbol{s}_t|\boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1}, \boldsymbol{\theta}_t^s) = \prod_{i=1}^{d} \mathbb{P}_s(s_{i,t}|pa(s_{i,t}))$$

where $pa(s_{i,t})$ *denotes the causal parents of* $s_{i,t}$ *in* $\mathcal{G}$*, which are a subset of the dimensions of* $\boldsymbol{s}_{t-1}$*,* $\boldsymbol{a}_{t-1}$ *and* $\boldsymbol{\theta}^{\boldsymbol{s}}_{\boldsymbol{t}}$*. Note that the action* $\boldsymbol{a}_{t-1}$ *is a vector of* m *dimensions in our setting. We assume a given initial state distribution* $\mathbb{P}_s(\boldsymbol{s}_0)$*. Similarly, we define the reward function* $\mathcal{R}$ *as a function of the parents of* r_t *in* $\mathcal{G}$*, i.e.,* $\mathcal{R}(\mathbf{s}_t, \mathbf{a}_t, \boldsymbol{\theta}_t^{\boldsymbol{r}}) = \mathcal{R}(pa(r_t))$*, where* $pa(r_t)$ *are a subset of dimensions of* $\mathbf{s}_t, \mathbf{a}_t$*, and* $\boldsymbol{\theta}_t^{\boldsymbol{r}}$*. We define the factored latent change factors transition distributions* $\mathbb{P}_{\theta^s}$ *and* $\mathbb{P}_{\theta^r}$ *as:*

$$\mathbb{P}_{\theta^s}(\boldsymbol{\theta}_t^{\boldsymbol{s}}|\boldsymbol{\theta}_{t-1}^{\boldsymbol{s}}) = \prod_{j=1}^{p} \mathbb{P}_{\theta^s}(\theta^s_{j,t}|pa(\theta^s_{j,t})), \qquad \mathbb{P}_{\theta^r}(\boldsymbol{\theta}_t^{\boldsymbol{r}}|\boldsymbol{\theta}_{t-1}^{\boldsymbol{r}}) = \prod_{k=1}^{q} \mathbb{P}_{\theta^r}(\theta^r_{k,t}|pa(\theta^r_{k,t}))$$

where $pa(\theta^s_{j,t})$ *are a subset of the dimensions of* $\boldsymbol{\theta}^{\boldsymbol{s}}_{t-1}$*, while* $pa(\theta^r_{k,t})$ *are a subset of dimensions of* $\boldsymbol{\theta}^{\boldsymbol{r}}_{t-1}$*. We assume the initial distributions* $\mathbb{P}_{\theta^s}(\boldsymbol{\theta}^{\boldsymbol{s}}_0)$ *and* $\mathbb{P}_{\theta^r}(\boldsymbol{\theta}^{\boldsymbol{r}}_0)$ *are given.*

We show an example DBN representing the graph $\mathcal{G}$ of an FN-MDP in Fig. 1(a). Since we are interested in learning the graphical structure of the FN-MDP, as well as identifying the values of the latent change factors from data, we describe a generative process of an FN-MDP environment. In particular, we assume that the graph $\mathcal{G}$ is time-invariant throughout the non-stationarity, and there are no unobserved confounders and instantaneous causal effects in the system. We will learn a set of binary masks $\boldsymbol{c}^{\cdot\to\cdot}$ and $\boldsymbol{C}^{\cdot\to\cdot}$ that are the indicators for edges in $\mathcal{G}$.

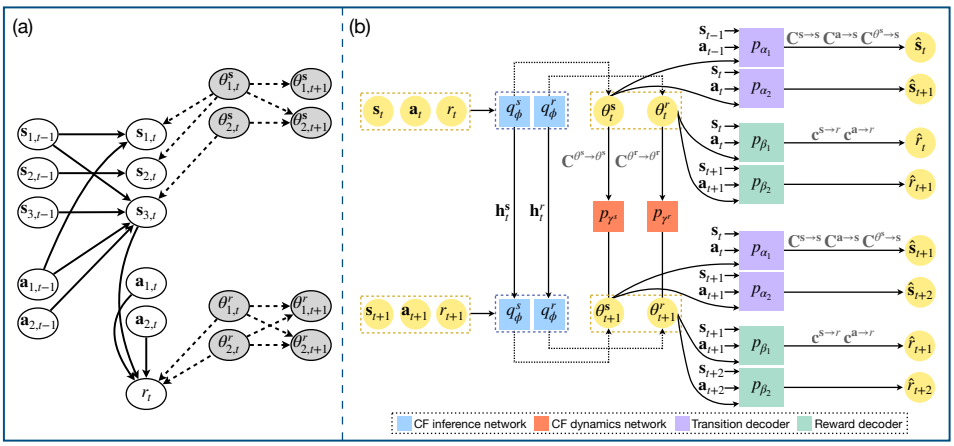

Figure 1: (a). A graphical representation of an FN-MDP. For readability, we only illustrate a subsection of dimensions of states, actions, and latent change factors. The shaded variables are unobserved; (b). The architecture of FN-VAE, which learns the generative model, explained in Sec. 3.

Generative environment model. We adapt the generative model in AdaRL [14] across k different domains to a time-varying setting on a single domain. We assume the generative process of the environment at timestep t in terms of the transition function for each dimension $i = 1, \ldots, d$ of $\mathbf{s}_t$ is:

$$s_{i,t} = f_i(\boldsymbol{c}_i^{\boldsymbol{s}\to\boldsymbol{s}} \odot \boldsymbol{s}_{t-1}, \boldsymbol{c}_i^{\boldsymbol{a}\to\boldsymbol{s}} \odot \boldsymbol{a}_{t-1}, \boldsymbol{c}_i^{\boldsymbol{\theta}_t\to\boldsymbol{s}} \odot \boldsymbol{\theta}_t^{\boldsymbol{s}}, \epsilon_{i,t}^s) \tag{1}$$

where $\odot$ is the element-wise product, f_i are non-linear functions and $\epsilon_{i,t}^s$ is an i.i.d. random noise. The binary mask $\boldsymbol{c}_i^{\boldsymbol{s}\to\boldsymbol{s}} \in \{0,1\}^d$ represents which state components $s_{j,t-1}$ are used in the transition function of $s_{i,t}$. Similarly, $\boldsymbol{c}_i^{\boldsymbol{a}\to\boldsymbol{s}} \in \{0,1\}^m$ indicates whether the action directly affects $s_{i,t}$. The change factor $\boldsymbol{\theta}_t^{\boldsymbol{s}} \in \mathbb{R}^p$ encodes any change in the dynamics. The binary mask $\boldsymbol{c}_{\boldsymbol{i}}^{\boldsymbol{\theta}_t\to\boldsymbol{s}} \in \{0,1\}^p$ represents which of the components of $\boldsymbol{\theta}_t^{\boldsymbol{s}}$ influence $s_{i,t}$. We model the reward function as:

$$r_t = h(\boldsymbol{c}^{\boldsymbol{s}\to r} \odot \boldsymbol{s}_t, \boldsymbol{c}^{\boldsymbol{a}\to r} \odot \boldsymbol{a}_t, \boldsymbol{\theta}_t^r, \epsilon_t^r) \tag{2}$$

where $\boldsymbol{c}^{\boldsymbol{s}\to\boldsymbol{r}} \in \{0,1\}^d$, $\boldsymbol{c}^{\boldsymbol{a}\to\boldsymbol{s}} \in \{0,1\}^m$, and ϵ_t^r is an i.i.d. random noise. The change factor $\boldsymbol{\theta}_t^r \in \mathbb{R}^q$ encodes any change in the reward function. The binary masks $\boldsymbol{c}^{\cdot\to\cdot}$ can be seen as indicators of edges in the DBN $\mathcal{G}$. In AdaRL, all change factors are assumed to be constant in each domain. Since in this paper we allow the change parameters to evolve in time, we introduce two additional equations:

$$\begin{aligned} \theta_{j,t}^s &= g^s(\boldsymbol{c}_j^{\boldsymbol{\theta}^s\to\boldsymbol{\theta}^s} \odot \boldsymbol{\theta}_{t-1}^{\boldsymbol{s}}, \epsilon_t^{\theta^s}) \\ \theta_{k,t}^r &= g^r(\boldsymbol{c}_k^{\boldsymbol{\theta}^r\to\boldsymbol{\theta}^r} \odot \boldsymbol{\theta}_{t-1}^r, \epsilon_t^{\theta^r}) \end{aligned} \tag{3}$$

for $i = 1, \ldots, d$, $j = 1, \ldots, p$, $k = 1, \ldots, q$, and g^s, and g^r are non-linear functions. We assume the binary masks $\boldsymbol{c}^{\cdot\to\cdot}$ are stationary across timesteps and so are the $\epsilon_{i,t}^s$, ϵ_t^r, $\epsilon_t^{\theta^s}$ and $\epsilon_t^{\theta^r}$, the i.i.d. random noises. Although $\boldsymbol{c}^{\cdot\to\cdot}$ and ϵ are stationary, we model the changes in the functions and some changes in the graph structure through $\boldsymbol{\theta}$. For example a certain value of $\boldsymbol{\theta}_t^r$ can switch off the contribution of some of the state or action dimensions in the reward function, or in other words nullify the effect of some edges in $\mathcal{G}$. Similarly the contribution of the noise distribution to each function can be modulated via the change factors. On the other hand, this setup does not allow adding edges that are not captured by the binary masks $\boldsymbol{c}^{\cdot\to\cdot}$. We group the binary masks in the matrices $\boldsymbol{C}^{\boldsymbol{s}\to\boldsymbol{s}} := [\boldsymbol{c}_i^{\boldsymbol{s}\to\boldsymbol{s}}]_{i=1}^d$, $\boldsymbol{C}^{\theta^s\to\boldsymbol{s}} := [\boldsymbol{c}_i^{\theta^s\to\boldsymbol{s}}]_{i=1}^d$, and $\boldsymbol{C}^{\boldsymbol{a}\to\boldsymbol{s}} := [\boldsymbol{c}_i^{\boldsymbol{a}\to\boldsymbol{s}}]_{i=1}^d$. Similarly, we also group the binary vectors in the dynamics of the latent change factors in matrices $\boldsymbol{C}^{\boldsymbol{\theta}^s\to\boldsymbol{\theta}^s} := [\boldsymbol{c}_j^{\boldsymbol{\theta}^s\to\boldsymbol{\theta}^s}]_{j=1}^p$ and $\boldsymbol{C}^{\boldsymbol{\theta}^r\to\boldsymbol{\theta}^r} := [\boldsymbol{c}_k^{\boldsymbol{\theta}^r\to\boldsymbol{\theta}^r}]_{k=1}^q$. Since latent change factors $\boldsymbol{\theta}^{\boldsymbol{s}}$ and $\boldsymbol{\theta}^r$ follow a Markov process based on g^s and g^r, we can consider different types of changes by varying the form of g^s and g^r, generalizing the approaches in literature. We can also model concurrent changes in dynamics and reward, including different types of changes, e.g. a continuous g^s and a piecewise-constant g^r.

Compact representations. Huang et al. [14] show that the only dimensions of the state and change factors useful for policy learning are those that eventually affect the reward. These dimensions are called *compact representations* and are defined as the dimensions of the state and change factors with a path (i.e. a sequence of edges $\rightarrow$) to the present or future reward $r_{t+\tau}$ for $\tau \geq 0$ in the DBN $\mathcal{G}$:

$$s_{i,t} \in \boldsymbol{s}^{min} \iff s_{i,t} \rightarrow \ldots \rightarrow r_{t+\tau} \text{ for } \tau \geq 0, \text{ and } \theta_i \in \boldsymbol{\theta}^{min} \iff \theta_i \rightarrow \ldots \rightarrow r_{t+\tau} \text{ for } \tau \geq 0$$

Continuous changes. If g^s and g^r are continuous, then they can model smooth changes in the environment, including across episodes. While the functions in Eq. 1-3 allow us to model *within-episode* changes, i.e. changes that can happen only before $t = H$ where H is the horizon, we also want to model *across-episode* changes. We use a separate time index $\tilde{t}$ that models the agent's lifetime. Initially $\tilde{t} = t$ for $t \leq H$, but while we reset $t = 0$ afterwards, $\tilde{t}$ continues to grow indefinitely.

Discrete changes. We assume discrete changes happen at specific timesteps and can be represented with a piecewise constant function. In particular we denote change timesteps as $\tilde{\boldsymbol{t}} = (\tilde{t}_1, \ldots, \tilde{t}_M)$ where $\tilde{t}_i$ describes a specific timepoint in the agent's lifetime time index $\tilde{t}$. This allows us to model *within-episode changes*. In this case, we assume that the change happens always at the same steps $(t_1, \ldots, t_m)$ in each episode, i.e., we assume that $\tilde{\boldsymbol{t}} = (t_1, \ldots, t_m, H + t_1, \ldots, H + t_m, 2H + t_1, \ldots, 2H + t_m, \ldots)$, where H is the horizon. We can also model *across-episode changes*, when we assume the change points only occur at the end of each episode, i.e., $\tilde{\boldsymbol{t}} = (H, 2H, 3H, \ldots)$.

We extend the result on identifiability of factored MDPs in AdaRL [14] to non-stationary environments. We first assume that we observe the change factors and show we can identify the true causal graph $\mathcal{G}$:

Proposition 1 (Full identifiability with observed change factors)**.** *Suppose the generative process follows Eq. 1-3 and all change factors $\boldsymbol{\theta}_t^{\boldsymbol{s}}$ and $\boldsymbol{\theta}_t^{\boldsymbol{r}}$ are observed, i.e., Eq. 1-3 is an MDP. Under the Markov and faithfulness assumptions, i.e. conditional independences correspond exactly to d-separations, all binary masks $\boldsymbol{C}^{\cdot \rightarrow \cdot}$ are identifiable, i.e., we can fully recover the causal graph $\mathcal{G}$.*

We provide all proofs and a detailed explanation in Appendix B. If we do not observe the change factors $\boldsymbol{\theta}_t^{\boldsymbol{s}}$ and $\boldsymbol{\theta}_t^{\boldsymbol{r}}$, we cannot identify their dimensions, and we cannot fully recover the causal graph $\mathcal{G}$. On the other hand, we can still identify the partial causal graph over the state variables $\mathbf{s}_t$, reward variable r_t, and action variable $\mathbf{a}_t$. We can also identify which dimensions in $s_{i,t}$ have changes, i.e., we can identify $\boldsymbol{C}^{\boldsymbol{\theta}^s \rightarrow \boldsymbol{s}}$. We formalize this idea in the following (proof in Appendix B):

Proposition 2 (Partial Identifiability with latent change factors)**.** *Suppose the generative process follows Eq. 1-3 and the change factors $\boldsymbol{\theta}_t^{\boldsymbol{s}}$ and $\boldsymbol{\theta}_t^{\boldsymbol{r}}$ are unobserved. Under the Markov and faithfulness assumptions, the binary masks $\boldsymbol{C}^{\boldsymbol{s} \rightarrow \boldsymbol{s}}$, $\boldsymbol{C}^{\boldsymbol{a} \rightarrow \boldsymbol{s}}$, $\boldsymbol{c}^{\boldsymbol{s} \rightarrow r}$ and $\boldsymbol{c}^{\boldsymbol{a} \rightarrow r}$ are identifiable. Moreover, we can identify which state dimensions are affected by $\boldsymbol{\theta}_t^{\boldsymbol{s}}$ and whether the reward function changes.*

This means that even in the most general case, we can learn most of the true causal graph G in an FN-MDP, with the exception of the transition structure of the latent change factors. In the following, we show a variational autoencoder setup to learn the generative process in FN-MDPs.

3 Learning the Generative Process in FN-MDPs

There are many possible architectures to learn FN-MDPs through Eq. 1-3. We propose *FN-VAE*, a variational autoencoder architecture described in Fig. 1(b). In FN-VAE, we jointly learn the structural relationships, state transition function, reward function, and transition function of the latent change factors, as described in detail in Appendix Alg. A1. An FN-VAE has four types of components: *change factor (CF) inference networks* that reconstruct the latent change factors, *change factor (CF) dynamics networks* that model their dynamics with an LSTM [15], *transition decoders* that reconstruct the state dynamics at the time t and predict one step further at $t + 1$, and *reward decoders* that reconstruct the reward at t and predict the future reward at $t + 1$. We now describe them in detail.

CF inference networks (blue boxes in Fig. 1(b)). The two inference models for latent change factors $q_{\phi^s}(\boldsymbol{\theta}_t^{\boldsymbol{s}} \mid \boldsymbol{s}_t, \boldsymbol{a}_t)$ and $q_{\phi^r}(\boldsymbol{\theta}_t^r \mid \boldsymbol{s}_t, \boldsymbol{a}_t, r_t)$ are parameterised by ϕ^s and ϕ^r, respectively. To model the time-dependence of $\boldsymbol{\theta}_t^{\boldsymbol{s}}$ and $\boldsymbol{\theta}_t^r$, we use LSTMs [15] as inference networks. At timestep t, the dynamics change factor LSTM infers $q_{\phi^s}\left(\boldsymbol{\theta}_t^s \mid \boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{h}_{t-1}^s\right)$, where $\boldsymbol{h}_{t-1}^s \in \mathbb{R}^L$ is the hidden state in the LSTM. Thus we can obtain $\mu_{\phi^s}(\boldsymbol{\tau}_{0:t})$ and $\sigma^2_{\phi^s}(\boldsymbol{\tau}_{0:t})$ using q_{ϕ^s}, and sample the latent changing factor $\boldsymbol{\theta}_t^{\boldsymbol{s}} \sim \mathcal{N}(\mu_{\phi^s}(\boldsymbol{\tau}_{0:t}), \sigma^2_{\phi^s}(\boldsymbol{\tau}_{0:t}))$, where $\boldsymbol{\tau}_{0:t} = (\boldsymbol{s}_0, \boldsymbol{a}_0, r_0, \boldsymbol{s}_1, \boldsymbol{a}_1, r_2, \ldots, \boldsymbol{s}_t, \boldsymbol{a}_t, r_t)$.

Similarly, the reward change factor LSTM infers $q_{\phi^r}(\boldsymbol{\theta}_t^r \mid \boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{h}_{t-1}^r)$, where $\boldsymbol{h}_{t-1}^r \in \mathbb{R}^L$ is the hidden state, such that we can sample $\boldsymbol{\theta}_t^r \sim \mathcal{N}(\mu_{\phi^r}(\boldsymbol{\tau}_{0:t}), \sigma^2_{\phi^r}(\boldsymbol{\tau}_{0:t}))$.

CF dynamics network (orange boxes in Fig. 1(b)). We model the dynamics of latent change factors with $p_{\gamma^s}\left(\boldsymbol{\theta}_{t+1}^{\boldsymbol{s}} \mid \boldsymbol{\theta}_t^{\boldsymbol{s}}, \boldsymbol{C}^{\boldsymbol{\theta}^s \to \boldsymbol{\theta}^s}\right)$ and $p_{\gamma^r}\left(\boldsymbol{\theta}_{t+1}^{\boldsymbol{r}} \mid \boldsymbol{\theta}_t^{\boldsymbol{r}}, \boldsymbol{C}^{\boldsymbol{\theta}^r \to \boldsymbol{\theta}^r}\right)$. To ensure the Markovianity of $\boldsymbol{\theta}_t^s$ and $\boldsymbol{\theta}_t^r$, we define a loss $\mathcal{L}_{\text{KL}}$ that helps minimize the KL-divergence between q_ϕ and p_γ.

$$\begin{aligned}\mathcal{L}_{\text{KL}} = \sum_{t=2}^{T} &\text{KL}\big(q_{\phi^s}\left(\boldsymbol{\theta}_t^s \mid \boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{h}_{t-1}^s\right)\big) \| p_{\gamma^s}(\boldsymbol{\theta}_t^{\boldsymbol{s}} | \boldsymbol{\theta}_{t-1}^{\boldsymbol{s}}; \boldsymbol{C}^{\boldsymbol{\theta}^s \to \boldsymbol{\theta}^s})\big) \\ &+ \text{KL}\big(q_{\phi^r}(\boldsymbol{\theta}_t^r \mid \boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{h}_{t-1}^r)\big) \| p_{\gamma^r}(\boldsymbol{\theta}_t^r | \boldsymbol{\theta}_{t-1}^r; \boldsymbol{C}^{\boldsymbol{\theta}^r \to \boldsymbol{\theta}^r})\big)\end{aligned} \tag{4}$$

If we assume that the change between $\boldsymbol{\theta}_t^{\boldsymbol{s}}$ and $\boldsymbol{\theta}_{t+1}^{\boldsymbol{s}}$, and similarly $\boldsymbol{\theta}_t^{\boldsymbol{r}}$, is smooth, we can add a smoothness loss $\mathcal{L}_{\text{smooth}}$. We provide a smooth loss for discrete changes in Appendix D.2.

$$\mathcal{L}_{\text{smooth}} = \sum_{t=2}^{T} \left(||\boldsymbol{\theta}_t^{\boldsymbol{s}} - \boldsymbol{\theta}_{t-1}^{\boldsymbol{s}}||_1 + ||\boldsymbol{\theta}_t^{\boldsymbol{r}} - \boldsymbol{\theta}_{t-1}^{\boldsymbol{r}}||_1\right) \tag{5}$$

Transition decoders (purple boxes in Fig. 1(b)). We learn an approximation of the transition dynamics in Eq. 1 by learning a reconstruction, parameterized by α_1, and a prediction encoder, parametrized by α_2. To simplify the formulas, we define $\boldsymbol{C}^{\cdot \to \boldsymbol{s}} := (\boldsymbol{C}^{\boldsymbol{s} \to \boldsymbol{s}}, \boldsymbol{C}^{\boldsymbol{a} \to \boldsymbol{s}}, \boldsymbol{C}^{\boldsymbol{\theta}^s \to \boldsymbol{s}})$. At timestep t, the reconstruction encoder $p_{\alpha_1}\left(\boldsymbol{s}_t \mid \boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1}, \boldsymbol{\theta}_t^s; \boldsymbol{C}^{\cdot \to \boldsymbol{s}}\right)$ reconstructs the state from current state $\boldsymbol{s}_t$ with sampled $\boldsymbol{\theta}_t^{\boldsymbol{s}}$. The one-step prediction encoder $p_{\alpha_2}\left(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{\theta}_t^{\boldsymbol{s}}\right)$ instead tries to approximate the next state $\boldsymbol{s}_{t+1}$. We do not use the prediction loss, when the one-step prediction is not smooth. In particular, we do not use it for the last time-step in episode i if there is a change happening at the first step in episode $(i+1)$, since the states in new episodes will be randomly initiated. We also do not use it in the case of discrete changes at the timesteps $(\tilde{t}_1 - 1, \ldots, \tilde{t}_M - 1)$. The loss functions are:

$$\begin{aligned}\mathcal{L}_{\text{rec-dyn}} &= \sum_{t=1}^{T-2} \mathbb{E}_{\theta_t^s \sim q_\phi} \log p_{\alpha_1}(\boldsymbol{s}_t | \boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1}, \boldsymbol{\theta}_t^s; \boldsymbol{C}^{\cdot \to \boldsymbol{s}}) \\ \mathcal{L}_{\text{pred-dyn}} &= \sum_{t=1}^{T-2} \mathbb{E}_{\theta_t^s \sim q_\phi} \log p_{\alpha_2}(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{\theta}_t^s)\end{aligned} \tag{6}$$

Reward decoders (green boxes in Fig. 1(b)). Similarly, we use a reconstruction encoder $p_{\beta_1}\left(r_t \mid \boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{\theta}_t^r, \boldsymbol{c}^{\boldsymbol{s} \to r}, \boldsymbol{c}^{\boldsymbol{a} \to r}\right)$, parameterized by β_1, and a one-step prediction encoder $p_{\beta_2}\left(r_{t+1} \mid \boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}, \boldsymbol{\theta}_t^r\right)$, parametrized by β_2, to approximate the reward function. Similarly to transition decoders, we do not use the one-step prediction loss, if it is not smooth. The losses are:

$$\begin{aligned}\mathcal{L}_{\text{rec-rw}} &= \sum_{t=1}^{T-2} \mathbb{E}_{\theta_t^r \sim q_\phi} \log p_{\beta_1}(r_t | \boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{\theta}_t^r; \boldsymbol{c}^{\boldsymbol{s} \to r}, \boldsymbol{c}^{\boldsymbol{a} \to r}) \\ \mathcal{L}_{\text{pred-rw}} &= \sum_{t=1}^{T-2} \mathbb{E}_{\theta_t^r \sim q_\phi} \log p_{\beta_2}(r_{t+1} | \boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}, \boldsymbol{\theta}_t^r)\end{aligned} \tag{7}$$

Sparsity loss. We encourage sparsity in the binary masks $\boldsymbol{C}^{\cdot \to \cdot}$ to improve identifiability, by using following loss with adjustable hyperparameters $(w_1, \ldots, w_7)$, which we learn through grid search.

$$\begin{aligned}\mathcal{L}_{\text{sparse}} = w_1 \|\boldsymbol{C}^{\boldsymbol{s} \to \boldsymbol{s}}\|_1 + w_2 \|\boldsymbol{C}^{\boldsymbol{a} \to \boldsymbol{s}}\|_1 + w_3 \|\boldsymbol{C}^{\boldsymbol{\theta}^s \to \boldsymbol{s}}\|_1 + w_4 \|\boldsymbol{c}^{\boldsymbol{s} \to r}\|_1 + w_5 \|\boldsymbol{c}^{\boldsymbol{a} \to r}\|_1 \\ + w_6 \|\boldsymbol{C}^{\boldsymbol{\theta}^s \to \boldsymbol{\theta}^s}\|_1 + w_7 \|\boldsymbol{C}^{\boldsymbol{\theta}^r \to \boldsymbol{\theta}^r}\|_1\end{aligned} \tag{8}$$

The total loss is $\mathcal{L}_{\text{vae}} = k_1(\mathcal{L}_{\text{rec-dyn}} + \mathcal{L}_{\text{rec-rw}}) + k_2(\mathcal{L}_{\text{pred-dyn}} + \mathcal{L}_{\text{pred-rw}}) - k_3 \mathcal{L}_{\text{KL}} - k_4 \mathcal{L}_{\text{sparse}} - k_5 \mathcal{L}_{\text{smooth}}$, where $(k_1, \ldots, k_5)$: hyper-parameters, which we learn with an automatic weighting method [16].

Learning from raw pixels. Our framework can be easily extended to image inputs by adding an encoder ϕ^o to learn the latent state variables from pixels, similar to other works [17, 18, 14]. In this case, our identifiability results do not hold anymore, since we cannot guarantee that we identify the true causal variables. We describe this component in Appendix D.3.

Algorithm 1: Factored Adaptation for non-stationary RL

1: **Init:** Env; VAE parameters: $\phi = (\phi^s, \phi^r)$, $\alpha = (\alpha_1, \alpha_2)$, $\beta = (\beta_1, \beta_2)$, γ; Binary masks: $\boldsymbol{C}^{\cdot\to\cdot}$; Policy parameters: ψ; replay buffer: $\mathcal{D}$; Number of episodes: N; Episode horizon: H; Initial $\boldsymbol{\theta}$: $\boldsymbol{\theta}^s_{\text{old}}$ and $\boldsymbol{\theta}^r_{\text{old}}$; Length of collected trajectory: k.
2: **Output:** VAE parameters: ϕ, α, β, γ; Policy parameters: ψ
3: Collect multiple trajectories of length k : $\boldsymbol{\tau} = \{\boldsymbol{\tau}^1_{0:k}, \boldsymbol{\tau}^2_{0:k}, \ldots\}$ with policy π_ψ from Env;
4: Learn FN-VAE from $\boldsymbol{\tau}$, including masks $\boldsymbol{C}^{\cdot\to\cdot}$ that represent the graph $\mathcal{G}$ (Appendix Alg. A1)
5: Identify the compact representations $\boldsymbol{s}^{min}$ and change factors $\boldsymbol{\theta}^{min}$ based on $\boldsymbol{C}^{\cdot\to\cdot}$
6: **for** $n = 0, \ldots, N-1$ **do**
7: **for** $t = 0, \ldots, H-1$ **do**
8: Observe $\boldsymbol{s}_t$ from Env;
9: **if** $t = 0$ **then**
10: $\boldsymbol{\theta}^s \leftarrow \boldsymbol{\theta}^s_{\text{old}}$ and $\boldsymbol{\theta}^r \leftarrow \boldsymbol{\theta}^r_{\text{old}}$
11: **else**
12: $\boldsymbol{\theta}^s \leftarrow \boldsymbol{\theta}^s_{t-1}$ and $\boldsymbol{\theta}^r \leftarrow \boldsymbol{\theta}^r_{t-1}$
13: **end if**
14: **for** j = s, r **do**
15: Infer mean $\mu_{\gamma^j}(\boldsymbol{\theta}^j)$ and variance $\sigma^2_{\gamma^j}(\boldsymbol{\theta}^j)$ of the change parameter $\boldsymbol{\theta}^j_t$ via p_{γ^j}
16: Sample $\boldsymbol{\theta}^j_t \sim \mathcal{N}\left(\mu_{\gamma^j}(\boldsymbol{\theta}^j), \sigma^2_{\gamma^j}(\boldsymbol{\theta}^j)\right)$
17: **end for**
18: **if** $t = H-1$ **then**
19: $\boldsymbol{\theta}^s_{\text{old}} \leftarrow \boldsymbol{\theta}^s_t$ and $\boldsymbol{\theta}^r_{\text{old}} \leftarrow \boldsymbol{\theta}^s_t$;
20: **end if**
21: Generate $\boldsymbol{a}_t \sim \pi_\psi(\boldsymbol{a}_t \mid \boldsymbol{s}^{min}_t, \boldsymbol{\theta}^{min}_t)$ and receive reward $r_{n,t}$ from Env;
22: Add $(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{\theta}^s_t, \boldsymbol{\theta}^r_t)$ to replay buffer $\mathcal{D}$;
23: Extract a trajectory with length k from replay buffer $\mathcal{D}$;
24: Learn FN-VAE (Appendix Alg. A1) with updateG=False (i.e. with fixed masks $\boldsymbol{C}^{\cdot\to\cdot}$);
25: Sample a batch of data from replay buffer $\mathcal{D}$ and update policy network parameters ψ;
26: **end for**
27: **end for**

4 FANS-RL: Online Model Estimation and Policy Optimization

We propose Factored Adaptation for Non-Stationary RL (FANS-RL), a general algorithm that interleaves model estimation and policy optimization, as shown in Alg. 1. After we estimate the initial FN-MDP with the FN-VAE, we can identify compact representations $\boldsymbol{s}^{min}$ and $\boldsymbol{\theta}^{min}$ following AdaRL [14]. In particular, the only dimensions of the state and change factors that are useful for policy learning are those that have a directed path to the reward in the graph $\mathcal{G}$. The online policy $\pi_\psi\left(\boldsymbol{a}_t \mid \boldsymbol{s}^{min}_t, q_\phi(\boldsymbol{\theta}^{min}_t \mid \boldsymbol{\tau}_{0:t})\right)$ can be learned end-to-end, including learning the FN-VAE, as shown in Alg. 1. We use SAC [19] as our policy learning model, so the policy parameters are $\psi = (\pi, Q)$.

Continuous changes. In Alg. 1 we describe our framework in case of continuous changes that can span across episodes. We start by collecting a few trajectories τ and then learn our initial FN-VAE (Lines 3-4). We can use the graphical structure of the initial FN-VAE to identify the compact representations (Line 5). During the online model estimation and policy learning stage, we estimate the latent change factors $\boldsymbol{\theta}^s_t$ and $\boldsymbol{\theta}^r_t$ using the CF dynamics networks γ^s and γ^r (Lines 8-20). Since in this case, we assume the dynamics of the change factors are smooth across episodes, at time $t = 0$ we will use the last timestep $(H-1)$ of the previous episode as a prior on the change factors (Line 10). Otherwise, we will estimate the change factors using their values in the previous timestep $t-1$. We use the estimated latent factors $\boldsymbol{\theta}_t$ and observed state $\boldsymbol{s}_t$ to generate $\boldsymbol{a}_t$ using π_ψ and receive a reward r_t (Line 21). We add $(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{\theta}^s_t, \boldsymbol{\theta}^r_t)$ to the replay buffer (Line 22). We now update our estimation of the FN-VAE, but we keep the graph G fixed (Lines 23-24). Finally, we sample a batch of trajectories in the replay buffer and update the policy network ψ (Line 25).

Discrete changes. Since we assume discrete changes happen at specific timestep $\tilde{\boldsymbol{t}} = (\tilde{t}_1, \ldots, \tilde{t}_M)$, we can easily modify Alg. 1 for discrete changes, both within-episode and across-episode, by changing Lines 9-20 to only update the change parameters at the timesteps in $\tilde{\mathbf{t}}$, as shown in Appendix Alg. A2.

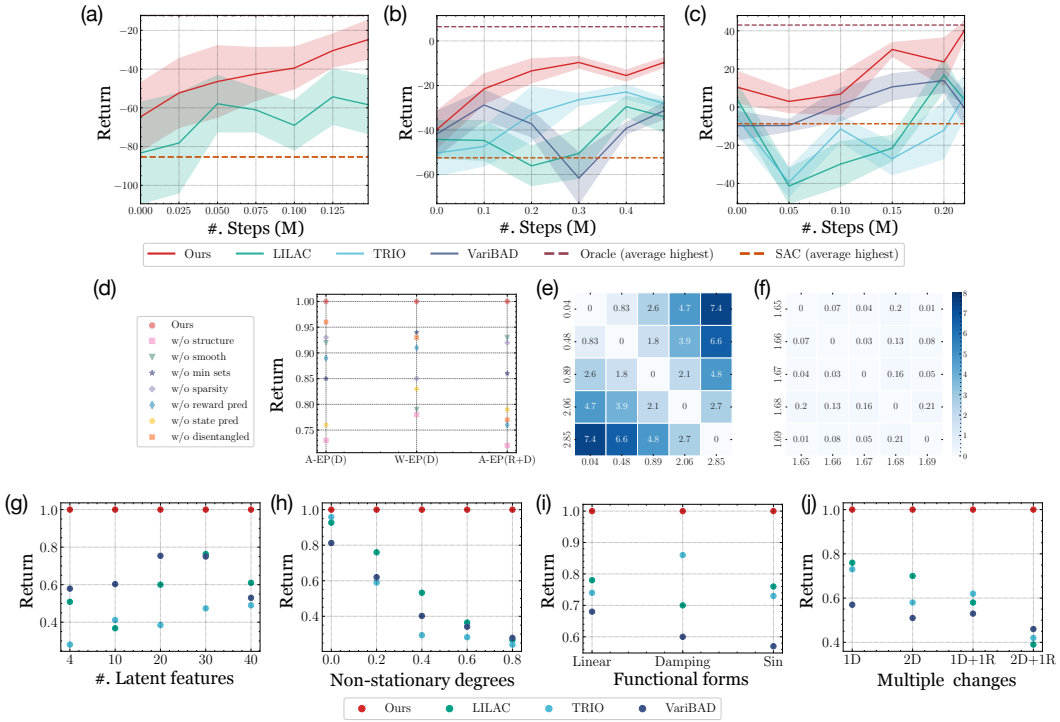

Figure 2: Summary of experimental results. (a)-(c). Average return (smoothed) across 10 runs. We only indicate the average of the highest result of all times for oracle and SAC. The shaded region is $(\mu - \sigma, \mu + \sigma)$, where μ is the mean and σ is the standard deviation. (a) Half-Cheetah-V3 with continuous (sine) changes on f_w; (b) Sawyer-Reaching with discrete across-episode changes on $\boldsymbol{s}^g$; and (c) Minitaur with discrete across-episode changes on m and $s_{t,v}$ concurrently. (d) Ablation studies on Half-Cheetah with across & within episode changes on dynamics and across episode changes on both dynamics and rewards. (e)-(f): Pairwise distance on learned $\boldsymbol{\theta}$ between different time steps in Half-Cheetah experiment with across-episode changes on rewards. (g)-(j): Average and normalized final return on 10 runs on Half-Cheetah (g) with within-in episode changes on wind forces using a different number of dimensions in latent representation space; (h) with different non-stationary degrees on across-episode and multi-factor changes; (i) with different functional forms (across episode changes on dynamics); and (j) with different combinations of across-episode changes.

5 Evaluation

We evaluate our approach on four well-established benchmarks, including Half-Cheetah-V3 from MuJoCo [20, 21], Sawyer-Reaching and Sawyer-Peg from Sawyer [22, 18], and Minitaur [23]. We modified these tasks to test several non-stationary RL scenarios with continuous and discrete changes. The results suggest that FANS-RL can (1) obtain high rewards, (2) learn meaningful mappings to the change factors with compact latent vectors, and (3) be robust to different non-stationary levels, different functional forms (e.g. piecewise linear, sine, damped sine) and multiple concurrent changes. For space limits, we only highlight a subset of results, and report the full results in Appendix C.4. The implementation will be open-sourced at `https://bit.ly/3erKoWm`.

Half-Cheetah-v3 in MuJoCo. In this task, the agent is moving forward using the joint legs and the objective is to achieve the target velocity v^g. We consider both the changes in the dynamics (change of the wind forces f^w, change of gravity) and reward functions (change of target velocity v^g). We also consider changes on the agent's mechanism, where one random joint is disabled. The reward function is $r_t = -\|v_t^o - v_t^g\|_2 - 0.05\|a_t\|_2$, where v^o and a_t are the agent's velocity and action, respectively, at timestep t. The number of time steps in each episode is 50. For dynamics, we change the wind forces f^w in the environment. Moreover, in terms of the reward functions, we change the target velocity v^g to be a time-dependent variable. The change function in the dynamics f^w can be either continuous or discrete, and discrete changes can happen both within and across episodes.

Methods	ZeUS	Meld	CaDM	Hyper-Dynamics	Ours
Best Avg. Return	12.45	6.38	4.04	3.91	**18.01**

Table 1: Average highest return on Sawyer-Peg. The number of random trails is 10.

Similarly to LILAC [7], we choose different functions (piecewise linear, sine and damped sine), besides allowing the change at specified intervals, we also allow it to change smoothly. The change in the reward function v^g is not generally stable in the continuous case, so we only consider the discrete and across episode change functions v^g for the reward. We also design a scenario where dynamic and reward functions change concurrently. We report all equations for g^s and g^r in Appendix C.1.

Sawyer. We consider two robotic manipulation tasks, Sawyer-Reaching and Sawyer-Peg. We describe the non-stationary settings in Appendix C.2.

In Sawyer-Reaching, the sawyer arm is trained to reach a target position $\boldsymbol{s}^g$. The reward r_t is the difference between the current position $\boldsymbol{s}_t$ and the target position $r_t = -\left\|\boldsymbol{s}_t - \boldsymbol{s}^g\right\|_2$. In this task, we cannot directly modify the dynamics in the simulator, so consider a reward-varying scenario, where the target location changes across each episode following a periodic function.

In Sawyer-Peg, the robot arm is trained to insert a peg into a designed target location $\boldsymbol{s}^g$. In this task, following [8], we consider a reward-varying scenario, where the target location changes across each episode following a periodic function. In order to compare with similar approaches, e.g., ZeUS [8], CADM [24], Hyperdynamics [25]and Meld [18], we evaluate our method on raw pixels. Following [8], we consider discrete across-episode changes, where the target location can change in each episode, and is randomly sampled from a small interval.

Minitaur. A minitaur robot is a simulated quadruped robot with eight direct-drive actuators. The minitaur is trained to move at the target speed $s_{t,v}$. The reward is $r_t = 0.3 - |0.3 - \boldsymbol{s}_{t,v}| - 0.01 \cdot \|\boldsymbol{a}_t - 2\boldsymbol{a}_{t-1} + \boldsymbol{a}_{t-2}\|_1$. We modify (1) the mass m (dynamics) and (2) target speed $s_{t,v}$ (reward) of the minitaur. We consider continuous, discrete across-episode and discrete within-episode changes for the dynamics, and across-episode changes for the reward. We describe the settings in Appendix C.3.

Baselines. We compare our approach with a meta-RL approach for stationary RL, **VariBAD** [26], a meta-RL approach for non-stationary RL, **TRIO** [5], as well as with two representative task embedding approaches, **LILAC** [7] and **ZeUS** [8]. The details on the meta-learning setups are given in Appendix D.4. We also compare with stationary RL method, **SAC** [19], which will be our lower-bound, and compare with an **oracle** agent that has access to the full information of non-stationarity (e.g., the wind forces) and can use it to learn a policy, which will be our upper-bound. For all baselines, we use SAC for policy learning. We compare with the baselines on *average return* and *the compactness of the latent space* in varying degrees and functional forms of non-stationarity.

Experimental results. Fig. 2(a)-(c) shows the smoothed curves of average return across timesteps in a subsection of Half-Cheetah, Sawyer-Reaching, and Minitaur experiments. Smoothed curves of other experiments are given in Appendix Fig. A2 and A3. For continuous changes, we only compare with LILAC [7] since other approaches are not applicable. We smooth the learning curves by uniformly selecting a few data points for readability. Table 1 shows the results on Sawyer-Peg experiments using raw pixels as input, based on the reported results in [8]. The learning curve is given in Appendix Fig. A9. For a fair comparison, we indicate the best and final average return for each baseline in Fig. A9. Full quantitative results of all experiemnts are given in Appendix C.4, including significance tests for Wilcoxon signed-rank test at $\alpha = 0.05$ showing that FANS-RL is significantly better than baselines. Finally, we conduct ablation studies on each component of FANS-RL and report some key results in Fig. 2(d). Full results are in Appendix C.4.

Ablation studies. The ablation studies verify the effectiveness of all components, including binary masks/structure, smoothness loss, sparsity loss, reward prediction or state prediction. The results show that *the largest gain is provided by the factored representation*, validating our original hypothesis, followed by state prediction. As expected, reward prediction is also important when there is a nonstationary reward, while smoothness is important for within-episode changes. The disentangled design of CF inference networks is valuable when there are changes on both dynamics and reward functions. Full results are in Appendix C.5.

Visualization on the learned $\boldsymbol{\theta}$. To verify that the learned $\boldsymbol{\theta}$ can capture the true change factors, we compute the pairwise distance between learned $\boldsymbol{\theta}^r$ at different time steps. We randomly sample 10 time steps from the Half-Cheetah experiment with across-episode changes on reward functions. Fig. 2(e) gives the pairwise distance of $\boldsymbol{\theta}^r$ among 5 time steps (from episode $148, 279, 155, 159, 230$) with different target speed values ($0.04, 0.48, 0.89, 2.06, 2.85$), respectively. We can find that there is a positive correlation between the distance of learned $\boldsymbol{\theta}^r$ and values of change factors. Meanwhile, we also sample 5 time steps from episodes $31, 188, 234, 408$ with target values around 1.67. Fig. 2(f) shows that the distance of $\boldsymbol{\theta}^r$ among these 5 time steps is very small, indicating that the learned $\boldsymbol{\theta}^r$ are almost the same for the similar values of change factors at different time steps. The visualization suggests that the learned $\boldsymbol{\theta}$ can capture meaningful mappings from the time-varying factors.

Varying latent dimensions, non-stationary levels and functional forms. Fig. 2(g) shows the normalized averaged return versus the number of two latent features on Half-Cheetah. Our framework can learn a better policy with relatively smaller feature dimensions in the latent space than other approaches. As we show in Appendix C.4, Saywer and Minitaur have the similar trend, where we learn a better policy than the baselines with fewer latent features. We also vary the non-stationary levels in Half-Cheetah with discrete across-episode changes on both dynamics and rewards. A higher non-stationary degree indicates a faster change rate. Fig. 2(h) shows that FANS-RL achieves the highest return across all tested non-stationary degrees and that the gap increases with the non-stationarity. We also test FANS-RL together with all baselines on different non-stationary functions, including piecewise linear, damping-like and sinusoid waves. Fig. 2(i) displays the results, which indicate that FANS-RL can generally outperform the baselines on diverse non-stationary function forms. Detailed function equations and experimental setups can be referred to Appendix C.

Multiple change factors We consider different numbers and types of changes to verify the benefits from the factored structure in FANS-RL. We conduct experiments with 1) only change wind forces (1D); 2) change wind forces and gravity concurrently (2D); 3) change wind force and target speed (1D+1R); and 4) change wind force, gravity, and target speed together (2D+1R) in an across-episode way in Half-Cheetah. From Fig. 2(j), we find that, thanks to the factored representation, FANS-RL performs better in those more complicated scenarios with multiple numbers and types of changes.

6 Related Work

Non-stationary RL. Early works in non-stationary RL [27, 28] only detect changes that have already happened instead of anticipating them. If the evolution of non-stationary environments is a (Semi)-Markov chain, one can deal with non-stationarity with HM-MDPs [29] or HS3MDPs [30]. Several methods learn to anticipate changes in non-stationary deep RL. Chandak et al. [31] propose to maximize future rewards without explicitly modeling non-stationary environments. MBCD [32] uses change-point detection to decide if the agent should learn a novel policy or reuse previously trained policies. Al-Shedivat et al. [4] extend MAML [6] for the non-stationary setting, but do not explicitly model the temporal changes. TRIO [5] tracks the non-stationarity by inferring the evolution of latent parameters, which captures the temporal change factors during the meta-testing phase. ReBAL and GrBAL [33] meta-train the dynamic prior, which adapts to the local contexts efficiently. However, these methods have to meta-train the model on a set of non-stationary tasks, which may not be accessible in real-world applications. Another line of research directly learns the latent representation to capture the non-stationary components. In particular, LILAC [7] and ZeUS [8] leverage latent variable models to directly model the change factors in environments, and Guo et al. [34] estimate latent vectors that describe the non-stationary or variable part of the dynamics. Our approach fits in this line of work; however, as opposed to these methods, which model all temporal changes using a shared embedding space or a mixture models of dynamics, we use a factored representation.

Factored MDPs. Among works on factored representations in stationary settings, Hallak et al. [35] learn factored MDPs [11, 12, 13] to improve sample efficiency in model-based off-policy RL. Balaji et al. [36] employ known factored MDP to improve both model-free and model-based RL algorithms. Working memory graphs [37] learn the factored observation space using Transformers. NeverNet [38] factorizes the state-action space through graph neural networks. Zholus et al. [39] factorize the visual states into actor-related, object of manipulation, and the latent influence factor between these two states to achieve sparse interaction in robotic control tasks. Differently from methods modeling the factored dynamics and rewards only, Zhou et al. [40] and Tang et al. [41] explore factored entities and actions, respectively. Zhou et al. [40] extend the factored MDP to the multi-entity environments,

learning the compositional structure in tasks with multiple entities involved. Tang et al. [41] leverage the factored action space to improve the sample efficiency in healthcare applications. However, the factored structures in these two works are derived from inductive bias or domain experts.

Factored MDPs, causality and multiple environments. Several works leverage factored MDPs to improve the sample efficiency and generalization of RL. Most of these works focus on learning an invariant (e.g. causal) representation that fits all environments and do not support learning latent change factors. For example, AFaR [42] learns factored value functions via attention modules to improve sample efficiency and generalization across tasks. Mutti et al. [43] learn a causal structure that can generalize across a family of MDPs under different environments, assuming that there are no latent causal factors. Similarly, Wang et al. [44] propose to learn the factored and causal dynamics in model-based RL, in which the learned causal structure is assumed to generalize to unseen states. By deriving the state abstraction based on the causal graph, it can improve both the sample efficiency and generalizability of policy learning of MBRL. While most of the previous works focus on learning a causal structure that is time-invariant, Pitis et al. [45] learn a locally causal dynamics that can vary at each timestep and use it to generate counterfactual dynamics transitions in RL. While the previously described methods focus on a domain or time-invariant representations, in our work we also focus on modelling domain or time-specific factors in the form of latent change factors. A related work, AdaRL [14] learns the factored representation and the model change factors under heterogeneous domains with varying dynamics or reward functions. However, AdaRL is designed only for the domain adaptation setting and constant change factors without considering non-stationarity.

Independent causal mechanisms. Another related line of work is based on independent causal mechanisms [46, 47]. Recurrent independent mechanisms (RIMs) [48] learn the independent transition dynamics in RL with sparse communication among the latent states. Meta-RIMs [49] leverage meta-learning and soft attention to learn a set of RIMs with competition and communication. As opposed to these works, we do not assume that the mechanisms are independent and we learn the factored structure among all components in MDPs with a DBN.

7 Conclusions, Limitations and Future Work

We describe Factored Adaptation for Non-Stationary RL (FANS-RL), a framework that learns a factored representation for non-stationarity that can be combined with any RL algorithm. We formalize our problem as a Factored Non-stationary MDP (FN-MDP), augmenting a factored MDP with latent change factors evolving as a Markov process. FN-MDPs do not model a family of MDPs, but instead include the dynamics of change factors analogously to the dynamics of the states. This allows us to capture different non-stationarities, e.g., continuous and discrete changes, both within and across different episodes. To learn FN-MDPs we propose FN-VAEs, which we integrated in FANS-RL, an online model estimation and policy evaluation approach. We evaluate FANS-RL on benchmarks for continuous control and robotic manipulation, also with pixel inputs, and show it outperforms the state of the art on rewards and robustness to varying degrees of non-stationarity. Learning the graph in model estimation is computationally expensive, which limits the scalability of our approach. In future work, we plan to meta-learn the graphs among different tasks to improve the scalability of our approach and its applicability to complex RL problems, e.g., multi-agent RL.

Acknowledgments

FF would like to acknowledge the CityU High-Performance Computing (HPC) resources in Hong Kong SAR and LISA HPC from the SURF.nl. SM was supported by the MIT-IBM Watson AI Lab and the Air Force Office of Scientific Research under award number FA8655-22-1-7155. BH would like to acknowledge the support of Apple Scholarship. KZ was partially supported by the National Institutes of Health (NIH) under Contract R01HL159805, by the NSF-Convergence Accelerator Track-D award #2134901, by a grant from Apple Inc., and by a grant from KDDI Research Inc.

References

[1] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[2] Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, pages 1–50, 2021.

[3] Sindhu Padakandla. A survey of reinforcement learning algorithms for dynamically varying environments. *ACM Computing Surveys (CSUR)*, 54(6):1–25, 2021.

[4] Maruan Al-Shedivat, Trapit Bansal, Yura Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *International Conference on Learning Representations (ICLR)*, 2018.

[5] Riccardo Poiani, Andrea Tirinzoni, and Marcello Restelli. Meta-reinforcement learning by tracking task non-stationarity. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2899–2905, 2021.

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, pages 1126–1135. PMLR, 2017.

[7] Annie Xie, James Harrison, and Chelsea Finn. Deep reinforcement learning amidst lifelong non-stationarity. *International Conference on Machine Learning (ICML)*, 2021.

[8] Shagun Sodhani, Franziska Meier, Joelle Pineau, and Amy Zhang. Block contextual mdps for continual learning. In *Learning for Dynamics and Control Conference (L4DC)*, pages 608–623. PMLR, 2022.

[9] Biwei Huang, Kun Zhang, Jiji Zhang, Joseph D Ramsey, Ruben Sanchez-Romero, Clark Glymour, and Bernhard Schölkopf. Causal discovery from heterogeneous/nonstationary data. *Journal of Machine Learning Research*, 21(89):612–634, 2020.

[10] Kun Zhang, Mingming Gong, Petar Stojanov, Biwei Huang, Qingsong Liu, and Clark Glymour. Domain adaptation as a problem of inference on graphical models. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:4965–4976, 2020.

[11] Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial intelligence*, 121(1-2):49–107, 2000.

[12] Michael Kearns and Daphne Koller. Efficient reinforcement learning in factored mdps. In *International Joint Conference on Artificial Intelligence (IJCAI)*, volume 16, pages 740–747, 1999.

[13] Ian Osband and Benjamin Van Roy. Near-optimal reinforcement learning in factored mdps. *Advances in Neural Information Processing Systems (NeurIPS)*, 27:604–612, 2014.

[14] Biwei Huang, Fan Feng, Chaochao Lu, Sara Magliacane, and Kun Zhang. Adarl: What, where, and how to adapt in transfer reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2022.

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

[16] Lukas Liebel and Marco Körner. Auxiliary tasks in multi-task learning. *arXiv preprint arXiv:1805.06334*, 2018.

[17] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Advances in Neural Information Processing Systems (NeurIPS)*, pages 2455–2467, 2018.

[18] Zihao Zhao, Anusha Nagabandi, Kate Rakelly, Chelsea Finn, and Sergey Levine. MELD: meta-reinforcement learning from images via latent state models. In *Conference on Robot Learning (CORL)*, pages 1246–1261. PMLR, 2020.

[19] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, pages 1861–1870. PMLR, 2018.

[20] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

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

[22] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CORL)*, pages 1094–1100. PMLR, 2020.

[23] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *Robotics: Science and Systems*, 2018.

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

[25] Zhou Xian, Shamit Lal, Hsiao-Yu Tung, Emmanouil Antonios Platanios, and Katerina Fragkiadaki. Hyperdynamics: Meta-learning object and agent dynamics with hypernetworks. In *International Conference on Learning Representations (ICLR)*, 2021.

[26] Luisa Zintgraf, Sebastian Schulze, Cong Lu, Leo Feng, Maximilian Igl, Kyriacos Shiarlis, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: Variational bayes-adaptive deep rl via meta-learning. *Journal of Machine Learning Research*, 22(289):1–39, 2021.

[27] Bruno C Da Silva, Eduardo W Basso, Ana LC Bazzan, and Paulo M Engel. Dealing with non-stationary environments using context detection. In *International Conference on Machine Learning (ICML)*, pages 217–224, 2006.

[28] Richard S Sutton, Anna Koop, and David Silver. On the role of tracking in stationary environments. In *International Conference on Machine Learning (ICML)*, pages 871–878, 2007.

[29] Samuel Choi, Dit-Yan Yeung, and Nevin Zhang. An environment model for nonstationary reinforcement learning. *Advances in neural information processing systems (NeurIPS)*, 12, 1999.

[30] Emmanuel Hadoux, Aurélie Beynier, and Paul Weng. Solving hidden-semi-markov-mode markov decision problems. In *International Conference on Scalable Uncertainty Management*, pages 176–189. Springer, 2014.

[31] Yash Chandak, Georgios Theocharous, Shiv Shankar, Martha White, Sridhar Mahadevan, and Philip Thomas. Optimizing for the future in non-stationary mdps. In *International Conference on Machine Learning (ICML)*, pages 1414–1425. PMLR, 2020.

[32] Lucas N. Alegre, Ana L. C. Bazzan, and Bruno C. da Silva. Minimum-delay adaptation in non-stationary reinforcement learning via online high-confidence change-point detection. In *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 97–105, 2021.

[33] Ignasi Clavera, Anusha Nagabandi, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2019.

[34] Jixian Guo, Mingming Gong, and Dacheng Tao. A relational intervention approach for unsupervised dynamics generalization in model-based reinforcement learnings. *International Conference on Learning Representations (ICLR)*, 2022.

[35] Assaf Hallak, François Schnitzler, Timothy Mann, and Shie Mannor. Off-policy model-based learning under unknown factored dynamics. In *International Conference on Machine Learning (ICML)*, pages 711–719. PMLR, 2015.

[36] Bharathan Balaji, Petros Christodoulou, Xiaoyu Lu, Byungsoo Jeon, and Jordan Bell-Masterson. Factoredrl: Leveraging factored graphs for deep reinforcement learning. *NeurIPS Deep Reinforcement Learning Workshop*, 2020.

[37] Ricky Loynd, Roland Fernandez, Asli Celikyilmaz, Adith Swaminathan, and Matthew Hausknecht. Working memory graphs. In *International Conference on Machine Learning (ICML)*, pages 6404–6414. PMLR, 2020.

[38] Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. In *International conference on learning representations (ICLR)*, 2018.

[39] Artem Zholus, Yaroslav Ivchenkov, and Aleksandr Panov. Factorized world models for learning causal relationships. In *ICLR Workshop on the Elements of Reasoning: Objects, Structure and Causality*, 2022.

[40] Allan Zhou, Vikash Kumar, Chelsea Finn, and Aravind Rajeswaran. Policy architectures for compositional generalization in control. In *ICML Workshop on Spurious Correlations, Invariance and Stability*, 2022.

[41] Shengpu Tang, Maggie Makar, Michael Sjoding, Finale Doshi-Velez, and Jenna Wiens. Leveraging factored action spaces for efficient offline reinforcement learning in healthcare. In *Decision Awareness in Reinforcement Learning Workshop at ICML*, 2022.

[42] Shagun Sodhani, Sergey Levine, and Amy Zhang. Improving generalization with approximate factored value functions. In *ICLR Workshop on the Elements of Reasoning: Objects, Structure and Causality*, 2022.

[43] Mirco Mutti, Riccardo De Santi, Emanuele Rossi, Juan Felipe Calderon, Michael Bronstein, and Marcello Restelli. Provably efficient causal model-based reinforcement learning for systematic generalization. *arXiv preprint arXiv:2202.06545*, 2022.

[44] Zizhao Wang, Xuesu Xiao, Zifan Xu, Yuke Zhu, and Peter Stone. Causal dynamics learning for task-independent state abstraction. In *International Conference on Machine Learning (ICML)*, pages 23151–23180. PMLR, 2022.

[45] Silviu Pitis, Elliot Creager, Ajay Mandlekar, and Animesh Garg. Mocoda: Model-based counterfactual data augmentation. In *Decision Awareness in Reinforcement Learning Workshop at ICML*, 2022.

[46] Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021.

[47] Giambattista Parascandolo, Niki Kilbertus, Mateo Rojas-Carulla, and Bernhard Schölkopf. Learning independent causal mechanisms. In *International Conference on Machine Learning (ICML)*, pages 4036–4044. PMLR, 2018.

[48] Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms. In *International Conference on Learning Representations (ICLR)*, 2021.

[49] Kanika Madan, Nan Rosemary Ke, Anirudh Goyal, Bernhard Schölkopf, and Yoshua Bengio. Fast and slow learning of recurrent independent mechanisms. In *International Conference on Learning Representations (ICLR)*, 2021.

[50] Peter Spirtes, Clark N Glymour, and Richard Scheines. *Causation, Prediction, and Search*. Spring-Verlag Lectures in Statistics, 1993.

[51] Kevin Patrick Murphy. *Dynamic bayesian networks: representation, inference and learning*. University of California, Berkeley, 2002.

[52] Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, Cambridge, 2000.

[53] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. Causal inference on time series using restricted structural equation models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 154–162, 2013.

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] There are theoretical claims backed by two theoretical results in Prop. 1 and Prop. 2. The empirical claims are backed by an extensive evaluation in Section 5.
   (b) Did you describe the limitations of your work? [Yes] We did provide a discussion on the limitations and potential future work in Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We discussed the broader societal impact in Appendix A.
   (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? [Yes] The assumptions are stated in Prop. 1 and Prop. 2.
   (b) Did you include complete proofs of all theoretical results? [Yes] The complete proofs are in the Appendix B
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] We will include the code and instructions in the supplementary files.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] The details are in the Appendix D.4.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] The learning curves (Fig. 2 (a-c), Appendix Fig. A2-A3) include error bars with 10 runs.
   (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] The information of computational resource is given in Appendix E.1.
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] Yes, we have cited the OpenAI Gym and Mujoco libraries in our paper.
   (b) Did you mention the license of the assets? [Yes] Yes, the licenses are given in Appendix E.2.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
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]

Appendix for Factored Adaptation for Non-Stationary Reinforcement Learning

A Broader Impact

Our work is a first exploration in leveraging factored representations for non-stationary RL in order to improve adaptation. One of the limitations of our current approach, and similarly to a few other non-stationary RL approaches, is that we do not provide theoretical guarantees in terms of adapting to non-stationarity. This limits the applicability of these approaches in safety-critical applications, e.g. self-driving cars, or in adversarial environment. One of the future directions of this work would be to provide theoretical guarantees under reasonable assumptions, similarly to generalization bounds for factored representations for fast domain adaptation [14].

Based on the application, there might be different assumptions or inductive biases that might be considered reasonable. Since our work leverages insights from recent causality literature [9], we also inherit the same inductive bias in terms of assuming there is an underlying causal structure that is time-invariant throughout the non-stationarity. In our current method, this causal structure is estimated in the model estimation phase as one of the first steps of the algorithm. After this estimation, there can still be changes in the functional dependencies between the various components, but we assume there are no new edges/causal relations between components that were previously disconnected, or new forms of non-stationarity, in terms of connections between the change factor components and the state dimensions or reward. For example, if we consider Halfcheetah-v3 with a change in gravity and estimate a model that can handle this type of non-stationarity, our method will not be able to perform well under a new type of non-stationarity (e.g. change of wind forces) that was not observed during model estimation. An interesting extension of our work would be designing ways to efficiently detect changes in the causal structure and adapt the model.

As is the case with other works in causal discovery, we also make some standard assumptions to recover the causal graph from time series observational data. In particular, we assume that there are no other unobserved confounders, except for the change factors, and that there are no instantaneous causal effects between the state components, which is implied by our definition of an FN-MDP and its Dynamic Bayesian Network. In practical applications, this means that we are able to measure all the relevant causal variables and we are measuring them at a rate that is faster than their interaction. Additionally, in our identifiability proofs, we assume the causal Markov and faithfulness assumptions [50], which provide a correspondence between conditional independences and d-separations in the graph. The faithfulness assumption can be violated for example in case of deterministic relations, thus requiring a careful modelling of the system. In general, if these assumptions are violated, the causal structure we learn might be incorrect, and therefore the factored representation might not be beneficial to adapt to non-stationarity. A future direction would be to relax some of the current assumptions to provide a more realistic and flexible framework for factored non-stationary RL.

B Proofs and Causality Background

B.1 Preliminaries

B.1.1 Dynamic Bayesian networks

Dynamic Bayesian networks (DBNs) [51] are the extensions of Bayesian networks (BN), which model the time-dependent relationship between nodes (See an example in Fig. A1(b)). The unfolded DBNs can be represented as BNs. The variables in DBNs are in discrete time slices and dependent on variables from the same and previous time slices. Hence, the DBNs can model the stationary process repeated over the discrete time slices.

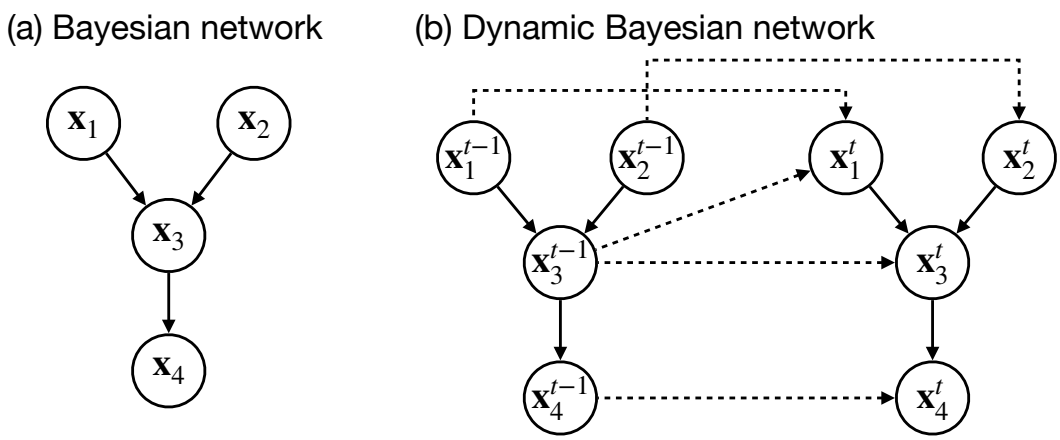

Figure A1: Examples on Bayesian and Dynamic Bayesian networks. The dashed edges indicate dependencies across time slices.

B.1.2 Markov and faithfulness assumptions

Given a directed acyclic graph $G = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ is the set of nodes and $\mathbf{E}$ is the set of directed edges, we can define a graphical criterion that expresses a set of conditions on the paths.

Definition 2 (d-separation [52])**.** *A path p is said to be blocked by a set of nodes $\mathbf{Z} \subseteq \boldsymbol{V}$ if and only if (1) p contains a chain $i \to m \to j$ or a fork $i \leftarrow m \to j$ such that the middle node m is in Z, or (2) p contains a collider $i \to m \leftarrow j$ such that the middle node m is not in $\mathbf{Z}$ and such that no descendant of m is in $\mathbf{Z}$. Let $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\mathbf{Z}$ be disjunct sets of nodes. $\mathbf{Z}$ is said to d-separate $\boldsymbol{X}$ from $\boldsymbol{Y}$ (denoted as $\boldsymbol{X} \perp_d \boldsymbol{Y}|\mathbf{Z}$) if and only if $\mathbf{Z}$ blocks every path from a node in $\boldsymbol{X}$ to a node in $\boldsymbol{Y}$.*

Definition 3 (Global Markov Condition [50, 52])**.** *A distribution P over $\mathbf{V}$ satisfies the global Markov condition on graph G if for any partition $(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{Y})$ such that $\boldsymbol{X}$ is d-separated from $\boldsymbol{Y}$ given $\boldsymbol{Z}$, i.e. $\boldsymbol{X} \perp_d \boldsymbol{Y}|\boldsymbol{Z}$ the distribution factorizes as:*

$$P(\boldsymbol{X}, \boldsymbol{Y}|\boldsymbol{Z}) = P(\boldsymbol{X}|\boldsymbol{Z})P(\boldsymbol{Y}|\boldsymbol{Z}).$$

In other words, $\boldsymbol{X}$ is conditionally independent of $\boldsymbol{Y}$ given $\boldsymbol{Z}$, which we denote as $\boldsymbol{X} \perp\!\!\!\perp \boldsymbol{Y}|\boldsymbol{Z}$.

Definition 4 (Faithfulness Assumption [50, 52])**.** *There are no independencies between variables that are not entailed by the Markov Condition.*

If we assume both of these assumptions, then we can use d-separation as a criterion to read all of the conditional independences from a given DAG G. In particular, for any disjoint subset of nodes $\mathbf{X},\mathbf{Y},\mathbf{Z} \subseteq \mathbf{V}$: $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z} \iff \mathbf{X} \perp_d \mathbf{Y}|\mathbf{Z}$.

B.2 AdaRL summary

Our work extends the factored representation for fast policy adaptation across domains introduced in AdaRL [14], which we summarize here. While Huang et al. [14] propose a general framework that can be applied to both MDPs and POMDPs, in this work we focus on MDPs, so we only present the simplified version of AdaRL for MDPs. The simplified AdaRL setting considers n source domains and n' target domains. The state at time t is represented as $\mathbf{s}_t = (s_{1,t}, \cdots, s_{d,t})^\top \in \mathcal{S}^d$, while $\mathbf{a}_t \in \mathcal{A}^m$ is the executed action and $r_t \in \mathcal{R}$ is the reward signal. The generative process of the environment in the k-th domain with $k = 1, \ldots, n + n'$ can be described in terms of the transition function for each dimension $i = 1, \ldots, d$ of $\mathbf{s}_t$ as:

$$s_{i,t} = f_i(\boldsymbol{c}_i^{\boldsymbol{s}\to\boldsymbol{s}} \odot \boldsymbol{s}_{t-1}, \boldsymbol{c}_i^{\boldsymbol{a}\to\boldsymbol{s}} \odot \boldsymbol{a}_{t-1}, \boldsymbol{c}_i^{\boldsymbol{\theta}_k\to\boldsymbol{s}} \odot \boldsymbol{\theta}_k^{\boldsymbol{s}}, \epsilon_{i,t}^s) \tag{A1}$$

where $\odot$ denotes the element-wise product. The binary mask $\boldsymbol{c}_i^{\boldsymbol{s}\to\boldsymbol{s}} \in \{0,1\}^d$ represents which of the state components $s_{j,t-1}$ are used in the transition function of $s_{i,t}$. Similarly, $\boldsymbol{c}_i^{\boldsymbol{a}\to\boldsymbol{s}} \in \{0,1\}^m$ is a mask that indicates whether the action directly affects $s_{i,t}$. The change factor $\boldsymbol{\theta}_k^{\boldsymbol{s}} \in \mathbb{R}^p$ is the only parameter that depends on the domain k in Eq. 1 and it encodes any change across domains in the

dynamics. The binary mask $\boldsymbol{c}_{\boldsymbol{i}}^{\boldsymbol{\theta}_k \to \boldsymbol{s}} \in \{0,1\}^p$ represents which of the $\boldsymbol{\theta}_k^{\boldsymbol{s}}$ components influence the $s_{i,t}$. Finally, $\epsilon_{i,t}^s$ is an i.i.d. random noise. Similarly the reward function is modeled as:

$$r_t = h(\boldsymbol{c}^{\boldsymbol{s}\to r} \odot \boldsymbol{s}_{t-1}, \boldsymbol{c}^{\boldsymbol{a}\to r} \odot \boldsymbol{a}_{t-1}, \boldsymbol{\theta}_k^r, \epsilon_t^r) \tag{A2}$$

where $\boldsymbol{c}_i^{\boldsymbol{s}\to\boldsymbol{r}} \in \{0,1\}^d$, $\boldsymbol{c}_i^{\boldsymbol{a}\to\boldsymbol{s}} \in \{0,1\}^m$, and ϵ_t^r is an i.i.d. random noise. The change factor $\boldsymbol{\theta}_k^r \in \mathbb{R}^q$ is the only parameter that depends on the domain k in Eq. A2 and it encodes any change in the reward function. In this simplified setting, the binary masks $\boldsymbol{c}^{\cdot\to\cdot}$ can be seen as indicators of edges in a Dynamic Bayesian Network (DBN). Under Markov and faithfulness assumptions, i.e., assuming the conditional independences in the data and d-separations in the true underlying graph coincide, the edges in the graph can be uniquely identified. This means one can learn the true causal graph representing jointly all of the environments, even if the change parameters are latent.

In the general AdaRL framework, the representation is learned via a combination of a state prediction network (to estimate the various f_i) and a reward prediction network (to estimate h). All binary masks $\boldsymbol{c}^{\cdot\to\cdot}$ and change factors $\theta_k^{\cdot}$ are trainable parameters. All change factors $\theta_k^{\cdot}$ are assumed to be constant in each domain k. If the inputs are pixels, another encoder is added to infer the symbolic states, forming a *Multi-model Structured Sequential Variational Auto-Encoder (MiSS-VAE)*. MiSS-VAE leverages the generative modeling to learn the data generation process in RL system with multiple domains. All change factors in MiSS-VAE are modeled by constants in each domain. This setting is not suitable for non-stationary RL where the change factors evolve over time.

In general, not all of the dimensions of the learned state and change factor vectors are useful in policy learning. Huang et al. [14] select a subsection of dimensions which are essential for policy optimization. Leveraging the learned representation as a DBN, we can select compact states and CFs as having a directed path to a reward:

$$s_{i,t} \in \boldsymbol{s}^{min} \iff s_{i,t} \to \ldots \to r_{t+\tau} \text{ for } \tau \geq 1$$
$$\theta_i \in \boldsymbol{\theta}^{min} \iff \theta_i \to \ldots \to r_{t+\tau} \text{ for } \tau \geq 1$$

B.3 Proofs

Proposition 3 (Identifiability with observed change factors)**.** *Suppose all the change factors $\boldsymbol{\theta}_t^{\boldsymbol{s}}$ and $\boldsymbol{\theta}_t^{\boldsymbol{r}}$ are observed, i.e., Eq. 1-3) is an MDP. Under the Markov and faithfulness assumptions, all the binary masks $\boldsymbol{C}^{\cdot\to\cdot}$ are identifiable.*

Proof. We construct the graph in the Factored Non-stationary MDP as a dynamic Bayesian network (DBN) G over the variables $\mathbf{V}_{\mathbf{MDP}} = \{s_{1,t-1}, \ldots, s_{d,t-1}, s_{1,t}, \ldots, s_{d,t}, a_{1,t-1}, \ldots, a_{m,t-1}, r_{t-1}\}$, and the change factors $\mathbf{V}_\theta = \{\theta_{1,t-1}^s, \ldots, \theta_{p,t-1}^s, \theta_{1,t-1}^r, \ldots, \theta_{q,t-1}^r, \theta_{1,t}^s, \ldots, \theta_{p,t}^s, \theta_{1,t}^r, \ldots, \theta_{q,t}^r\}$. In this setting, we can always the correct causal graph under the Markov and faithfulness assumptions. We rewrite the time index of r_{t-1} as r_t.

We could rewrite also the change factors time indices by shifting them back in time by one step, i.e. $t \to t-1$, and this would allow us to model the whole setup as a Markov DBN without any instantaneous effect. In this setup, we can leverage existing results to show that the true causal graph is asymptotically identifiable from conditional independences in a time-series without any unobserved confounders or instantaneous effects [53]. In order to make our proof clearer, we instead show step by step how we can recover the parts of the graph related to $\mathbf{V}_{\mathbf{MDP}}$, to $\mathbf{V}_\theta$ and finally the connections between them.

In this setting, there are no instantaneous effects except for the change factors, and the only causal parents for a variable that is not a change factor at time t can be in the previous time-step $t-1$. In particular, as in usual MDPs, these are the only allowed edges:

1. state dimension $s_{i,t-1}$ at time $t-1$ to state dimension $s_{j,t}$ at time t, for $i, j \in \{1, \ldots, d\}$ (this includes the case in which $i = j$);
2. action dimension $a_{k,t-1}$ at time $t-1$ to state dimension $a_{j,t}$ at time t, for $j \in \{1, \ldots, d\}, k \in \{1, \ldots, m\}$;
3. state dimension $s_{i,t-1}$ at time $t-1$ to reward r_t at time t, for $i \in \{1, \ldots, d\}, k \in \{1, \ldots, m\}$;

4. action dimension $a_{k,t-1}$ at time $t-1$ to reward r_t at time t, for $k \in \{1, \dots, m\}$;

and in addition we have some extra knowledge about the allowed edges to and from change factors, as expressed in our generative model in Equations (1-3):

1. transition change factor dimension $\theta^s_{i,t}$ at time t to state dimension $s_{j,t}$ at time t, for $i \in \{1, \dots, p\}, j \in \{1, \dots, d\}$;
2. transition change factor dimension $\theta^s_{i,t}$ at time t to transition change factor dimension $\theta^s_{j,t+1}$ at time $t+1$, for $i, j \in \{1, \dots, p\}$;
3. reward change factor dimension $\theta^r_{i,t}$ at time t to reward change factor dimension $\theta^r_{j,t+1}$ at time $t+1$, for $i, j \in \{1, \dots, q\}$;

For the reward change factors, we assume they are fully connected to the reward, as shown in Eq. (2). Using this background knowledge, we can learn the edges from any other variable $V_{i,t-1} \to V_{j,t}$, for $V_{i,t-1}, V_{j,t} \in \mathbf{V_{MDP}}$, just by checking if $V_{i,t-1} \not\perp\!\!\!\perp V_{i,t} | \theta^r_t, \theta^s_t, \mathbf{s}_{t-2}, \mathbf{a}_{t-2}$. This dependence implies that the variables are d-connected, under the Markov and faithfulness assumptions. Except for a direct edge, there is no other possible path through the graph unrolled in time, since we have blocked all influence of time-step $t-2$ and earlier, and the paths through future time-steps contain colliders. So this means that $V_{i,t-1}$ and $V_{j,t}$ are adjacent, and in particular the edge follows the arrow of time, from $t-1$ to t. In our setting, we do not model the actions at time t, since we assume they are not caused by any other variable. We also never need to condition on the reward to check if two variables are adjacent, since it's always on a collider path. This means we are able to learn the following binary masks:

1. state dimensions to state dimensions $\boldsymbol{C^{s \to s}}$;
2. action dimensions to state dimensions $\boldsymbol{C^{a \to s}}$;
3. state dimensions to reward $\boldsymbol{c^{s \to r}}$
4. action dimensions to reward $\boldsymbol{c^{a \to r}}$;

To learn the edges between any change factor component to another change factor component, i.e., $V_{i,t-1} \to V_{j,t}$ for $V_{i,t-1}, V_{j,t} \in \mathbf{V}_\theta$, we can just check if $V_{i,t-1} \not\perp\!\!\!\perp V_{i,t} | \theta^r_{t-2}, \theta^s_{t-2}$, since states, actions and rewards can never be parents of the change factors, so we do not need to condition on them to close any path through the earlier time-steps. We also assumed that change factor components follow a Markov process, so they do not have instantaneous effects towards each other. This means we are able to learn the following masks:

1. transition change factor dimensions to transition change factor dimensions $\boldsymbol{C^{\theta^s \to \theta^s}}$;
2. reward change factor dimensions to reward change factor dimensions $\boldsymbol{C^{\theta^r \to \theta^r}}$;

Finally to learn the edges between the change factors $V_{i,t} \in \mathbf{V}_\theta$ and the other variables $V_{j,t} \in \mathbf{V_{MDP}}$, we can just check if $V_{i,t} \not\perp\!\!\!\perp V_{j,t} | \mathbf{s}_{t-1}, \mathbf{a}_{t-1}, \theta^r_{t-1}, \theta^s_{t-1}$ and if this is true we can learn the edge $V_{i,t} \to V_{j,t}$. This means we are able to learn the mask:

1. transition change factor dimensions to state dimensions $\boldsymbol{C^{\theta^s \to s}}$;

All of these results together show that if we have the Markov and faithfulness assumption, i.e. the conditional independence tests return the true d-separations in the graph, and we observe the change factors, we are able to completely identify the graph of the FN-MDP represented by the binary masks. □

Proposition 4 (Partial Identifiability with latent change factors)**.** *Suppose the generative process follows Eq. 1-3 and the change factors $\boldsymbol{\theta^s_t}$ and $\boldsymbol{\theta^r_t}$ are unobserved. Under the Markov and faithfulness assumptions, the binary masks $\boldsymbol{C^{s \to s}}, \boldsymbol{C^{a \to s}}, \boldsymbol{c^{s \to r}}$ and $\boldsymbol{c^{a \to r}}$. Moreover, we can identify which state dimensions are affected by $\boldsymbol{\theta^s_t}$ and if the reward function changes.*

Proof. In this case, the MDP is non-stationary, since we cannot observe the latent change factors. We assume that we can represent the latent change factors as a smooth function of the observed time index t. This assumption is called *pseudo-causal sufficiency* in previous work [9]. We can then use the time index t as a surrogate variable to characterize the unobserved change factors, since at each time-step their value will be fixed.

We again consider a DBN $\mathcal{G}_{MDP \cup \{t\}}$ over the variables $\mathbf{V_{MDP}} = \{s_{1,t-1}, \dots, s_{d,t-1}, s_{1,t}, \dots, s_{d,t}, a_{1,t-1}, \dots, a_{m,t-1}, r_{t-1}\}$ and the time index t. We rewrite

the time index of r_{t-1} as r_t. Note that we do not represent the change factors in this DBN, but we can capture their effect through t since they are assumed to be deterministic smooth functions of the time index.

We can then reuse the results by Huang et al. [9] (Theorem 1) in which under the pseudo-causal sufficiency (Assumption 1 in that paper) and the Markov and faithfulness assumption (Assumption 2), one can asymptotically identify the true causal skeleton (i.e. the adjacencies) in the graph $\mathcal{G}_{MDP\cup\{t\}}$) through conditional independence tests. In particular for any $V_i, V_j \in \mathbf{V}_{MDP} \cup \{t\}$, V_i and V_j are not adjacent if there exists a subset of the unrolled graph $\mathbf{V}_k$ of $\mathbf{V}_{MDP\cup\{t\}} \setminus \{V_i, V_j\}$ in $\mathcal{G}_{MDP\cup\{t\}}$ such that $V_i \perp\!\!\!\perp V_j | \mathbf{V}_k$. In particular, we can also focus on the tests that were used for the proof of the previous Proposition, by just substituting the change factors with the time index t. The skeleton is the undirected version of the causal graph, so this result only tells us that can asymptotically get the correct undirected edges, but not their orientations.

Fortunately, in our setting we have some additional background knowledge that allows us to orient all the existing edges. In particular, as in usual MDPs, these are the only allowed edges:

1. state dimension $s_{i,t-1}$ at time $t-1$ to state dimension $s_{j,t}$ at time t, for $i, j \in \{1, \dots, d\}$ (this includes the case in which $i = j$);
2. action dimension $a_{k,t-1}$ at time $t-1$ to state dimension $a_{j,t}$ at time t, for $j \in \{1, \dots, d\}, k \in \{1, \dots, m\}$;
3. state dimension $s_{i,t-1}$ at time $t-1$ to reward r_t at time t, for $i \in \{1, \dots, d\}, k \in \{1, \dots, m\}$;
4. action dimension $a_{k,t-1}$ at time $t-1$ to reward r_t at time t, for $k \in \{1, \dots, m\}$;

This means that, for example, we cannot have a variable at time t causing a variable at time $t-1$. Therefore if two variables $V_{i,t-1}$ and $V_{j,t}$ are adjacent, we already know that the direction of that edge will be $V_{i,t-1} \to V_{j,t}$. This also implies that the following binary masks are identifiable (i.e. no edge remains unoriented):

1. state dimensions to state dimensions $\boldsymbol{C}^{\boldsymbol{s}\to\boldsymbol{s}}$;
2. action dimensions to state dimensions $\boldsymbol{C}^{\boldsymbol{a}\to\boldsymbol{s}}$;
3. state dimensions to reward $\boldsymbol{c}^{\boldsymbol{s}\to r}$
4. action dimensions to reward $\boldsymbol{c}^{\boldsymbol{a}\to r}$;

These represent all of the edges in $\mathcal{G}_{MDP}$. We can also learn the edges from t to V_{MDP} (by construction we assume the opposite direction is not possible), which will represent the effect of the change factors, as we show in the following.

Since t inherits all of the children of the latent change factors in G, we can further show that if $s_{i,t} \perp\!\!\!\perp t | \boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1}$ in G, then none of the latent change factor dimensions $\theta^s_{j,t}$ affect $s_{i,t}$, i.e., $s_{i,t} \perp\!\!\!\perp t | \boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1} \iff c^{\boldsymbol{\theta}^{\boldsymbol{s}}\to\boldsymbol{s}}_{i,j} = 0$. Intuitively, this means that the distribution of $s_{i,t}$ only depends on $\mathbf{s}_{t-1}$ and $\mathbf{a}_{t-1}$, and not on the timestep t, or in other words, this distribution is stationary. Under the same principle, if $r_t \perp\!\!\!\perp t | \boldsymbol{s}_t, \boldsymbol{a}_t$, then the reward is stationary.

□

C Details on Experimental Designs and Results

C.1 MuJoCo

We modify the Half-Cheetah environment into a variety of non-stationary settings. Details on the change factors are given as below.

Changes on dynamics. We change the wind forces f^w in the environment. We consider the changing functions can be both continuous and discrete.

• Continuous changes: $f^w_t = 10 + 10\sin(0.005 \cdot t)$, where t is the timestep index;
• Discrete changes: (1) Across-episode:
a. Sine function: $f_w = 10 + 10\sin(0.5 \cdot i)$
b. Damping-like function: $f_w = 10 + 3 \cdot (1.01)^{-\lceil i/10 \rceil} \sin(0.5 \cdot i)$

c. Piecewise linear function $f_w = 5 + 0.02 \cdot \|i - 1500\|$; , where i is the episode index.
(2) Within-episode: $f_w = 10 + 10\sin(0.4 \cdot \lfloor t/10 \rfloor)$, where t is the timestep index.

We also consider a special case where the agent's mechanism is changing over time. Specifically, the one random joint is disabled at the beginning of each episode.

Changes on reward functions. To introduce non-stationarity in the rewards, we change the target speed v_g in each episode. To make the learning process stable, we only consider the discrete changes and the change points are located at the beginning of each episode. The changing function is $v_g = 1.5 + 1.5\sin(0.2 \cdot i)$, where i denotes the episode index.

Changes on both dynamics and rewards. We consider a more general but challenging scenario, where the changes on dynamics and rewards can happen concurrently during the lifetime of the agents. We change the wind forces and target speed at the beginning of each episode. At episode i, the dynamics and reward functions are:

$$\begin{cases} f_w = 10 + 10\sin(w \cdot i) \\ v_g = 1.5 + 1.5\sin(w \cdot i) \end{cases}$$

Here, w is the non-stationary degree. We consider multiple values of w in our experiments. In Fig. 2(d), $w = 0.5$. In Fig. 2(h), w is the value of non-stationary degree.

C.2 Sawyer benchmarks

In Sawyer-Reaching, the sawyer arm is trained to reach a target position $\boldsymbol{s}_t^g$. The reward r_t is the difference between the current position $\boldsymbol{s}_t$ and the target position $r_t = -\left\|\boldsymbol{s}_t - \boldsymbol{s}^g\right\|_2$. In this task, we cannot directly modify the dynamics in the simulator, so consider a reward-varying scenario, where the target location changes across each episode following a periodic function. In Sawyer-Peg, the robot arm is trained to insert a peg into a designed target location $\boldsymbol{s}^g$. The reward function is $r_t = \mathbb{I}\left(\|\boldsymbol{s_t} - \boldsymbol{s}^g\|_2 \leq 0.05\right)$.

We change the target location in Sawyer reaching task. The target location $\boldsymbol{s}_t^g$ is given as below:

$$\mathbf{s}_t^g = \begin{bmatrix} 0.1 \cdot \|\cos(0.2 \cdot i)\| \\ 0.1 \cdot \sin(0.5 \cdot i) \\ 0.2 \end{bmatrix}$$

where i is the episode index. For Sawyer-Peg task, the target location $\boldsymbol{s}_g$ changes at each episode. The parameters in each dimension of $\boldsymbol{s}_g$ is randomly sampled at episode i as below:

- x_range_1: $(0.44, 0.45)$;
- x_range_2: $(0.6, 0.61)$;
- y_range_1: $(-0.08, -0.07)$;
- y_range_2: $(0.07, 0.08)$;

C.3 Minitaur benchmarks

We consider both the changes on dynamics and reward functions.

Changes on dynamics. We change the mass of taur m in the environment. Specifically, we consider both the continuous and discrete changes.

- Continuous changes: $m_t = 1.0 + 0.75\sin(0.005 \cdot t)$;
- Discrete and within-episode changes: $m_t = 1.0 + 0.75\sin(0.3 \cdot \lfloor t/20 \rfloor)$

Changes on both dynamics and reward functions. We also consider a case where both the dynamics and reward functions change at the beginning of each episode. We change the target speed of minitaur to introduce the non-stationarity of reward functions. The change functions are given below:

$$\begin{cases} m_i = 1.0 + 0.5\sin(0.5 \cdot i) \\ s_v = 0.3 + 0.2\sin(0.5 \cdot i) \end{cases}$$

C.4 Full results

Fig. A2 and A3 give the smoothed learning curves on average return over 10 runs versus timesteps in Half-Cheetah and Minitaur experiments. Table C.4 shows the average final return over 10 runs for all experiments. Fig. A4 demonstrates the return on Half-Cheetah with different non-stationary degrees on multi-factor changing scenario. Fig. A5 gives average return on different benchmarks with varying numbers of latent features with all evaluated approaches.

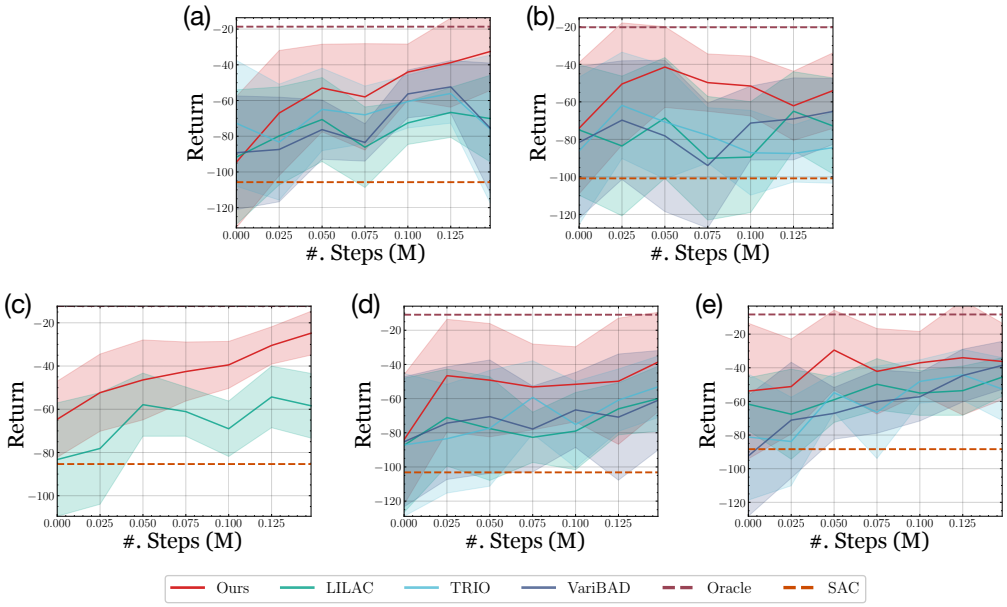

Figure A2: The average return (smoothed) across timesteps in Half-Cheetah experiments. (a) Discrete (across-episode) changes on wind forces; (b) Discrete (within-episode) changes on wind forces; (c) Continuous changes on wind forces; (d) Discrete (across-episode) changes on target speed; (e) Discrete (across-episode) changes on wind forces and target speed concurrently.

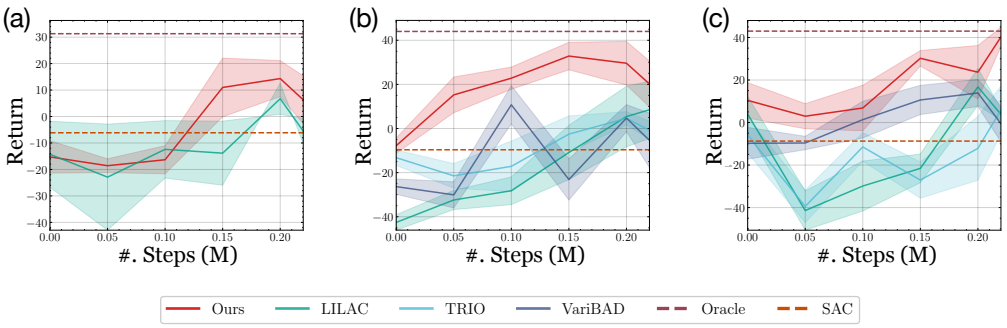

Figure A3: The average return (smoothed) across timesteps in Minitaur experiments. (a) Continuous changes on the mass; (b) Discrete (across-episode) changes on the target speed; (c) Discrete (across-episode) changes on mass and target speed concurrently.

	Oracle	SAC	LILAC	TRIO	VariBAD	Ours
Half-Cheetah: A-EP (D)	−24.4 (±16.2)	−113.4 • (±28.5)	−70.1 • (±27.7)	−76.0 • (±47.3)	−75.5 • (±41.6)	**-32.6** (±25.0)
Half-Cheetah: A-EP (A)	−9.6 (±5.7)	−30.5 • (±12.1)	−19.4 • (±11.4)	−21.9 • (±13.0)	−17.3 • (±10.2)	**-15.1** (±9.8)
Half-Cheetah: W-EP (D)	−48.2 (±41.6)	−107.5 • (±20.6)	−72.9 • (±29.3)	−84.4 • (±21.7)	−65.1 • (±20.1)	**-54.0** (±23.0)
Half-Cheetah: CONT (D)	−12.3 (±27.7)	−112.0 • (±16.9)	−58.4 • (±22.3)	-	-	**-24.8** (±21.1)
Half-Cheetah: A-EP (R)	−10.9 (±20.1)	−131.5 • (±16.9)	−60.1 • (±21.7)	−53.1 • (±20.6)	−61.0 • (±33.3)	**-38.7** (±33.3)
Half-Cheetah: A-EP (R+D)	−15.2 (±38.1)	−105.3 • (±38.1)	−45.6 (±13.1)	−52.6 • (±21.4)	−38.6 (±16.3)	**-36.2** (±26.0)
Sawyer-Reaching: A-EP (R)	6.4 (±3.9)	−52.5 • (±9.1)	−34.0 • (±8.2)	−28.1 • (±2.9)	−31.3 • (±4.3)	**-9.7** (±2.5)
Minitaur: CONT (D)	31.3 (±4.2)	−6.1 • (±3.9)	−5.5 • (±11.7)	-	-	**6.3** (±10.4)
Minitaur: W-EP (D)	44.9 (±5.8)	−9.6 • (±5.5)	8.5 • (±14.9)	−0.8 • (±4.7)	5.4 • (±14.1)	**20.2** (±11.9)
Minitaur: A-EP (R+D)	43.0 (±4.7)	−8.7 • (±5.4)	3.8 • (±3.0)	5.8 • (±12.9)	21.5 • (±9.7)	**40.2** (±5.3)

Table A1: Average final return of different methods on Half-Cheetah, Sawyer-Reaching, and minitaur benchmarks with a variety of non-stationary settings. The best non-oracle results w.r.t. the mean are marked in **bold**. "•" indicates the baseline for which the improvements of our approach are statistically significant (via Wilcoxon signed-rank test at 5% significance level). D, R, and A denote changes on dynamics, reward and agent's mechanism respectively. A-EP, W-EP, and CONT denote across-episode, within-episode and continuous changes, respectively.

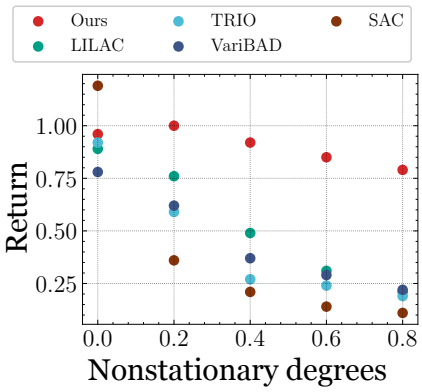

Figure A4: Average final return on 10 runs on Half-Cheetah with different non-stationary degrees on across-episode and multi-factor changes.

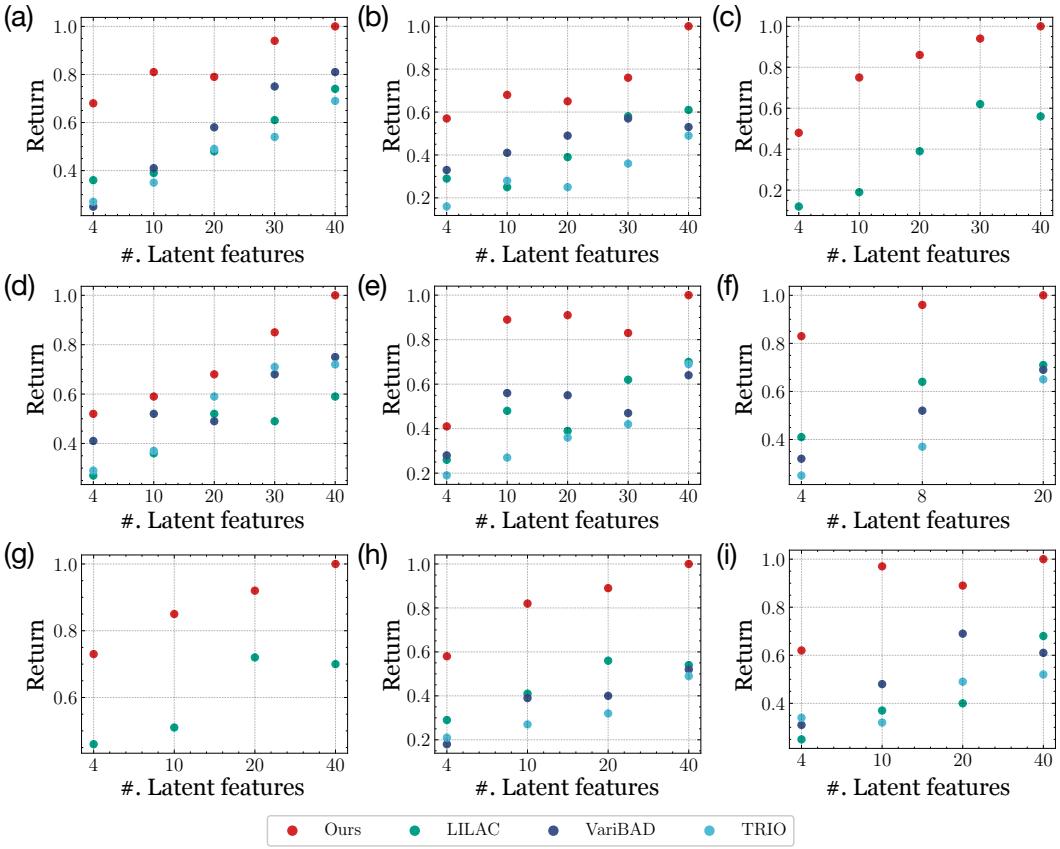

Figure A5: Average return on different benchmarks with different number of latent features. (a) Half-Cheetah experiments with discrete (across-episode) changes on wind forces; (b) Half-Cheetah experiments with discrete (within-episode) changes on wind forces; (c) Half-Cheetah experiments with continuous changes on wind forces; (d) Half-Cheetah experiments with discrete (across-episode) changes on target speed; (e) Half-Cheetah experiments with discrete (across-episode) changes on wind forces and target speed concurrently; (f) Sawyer-Reaching experiment with discrete (across-episode) changes on target locations; (g) Minitaur experiments with continuous changes on the mass; (h) Minitaur experiments with discrete (across-episode) changes on the target speed; (i) Minitaur experiments with discrete (across-episode) changes on mass and target speed concurrently.

C.5 Ablation studies on FANS-RL

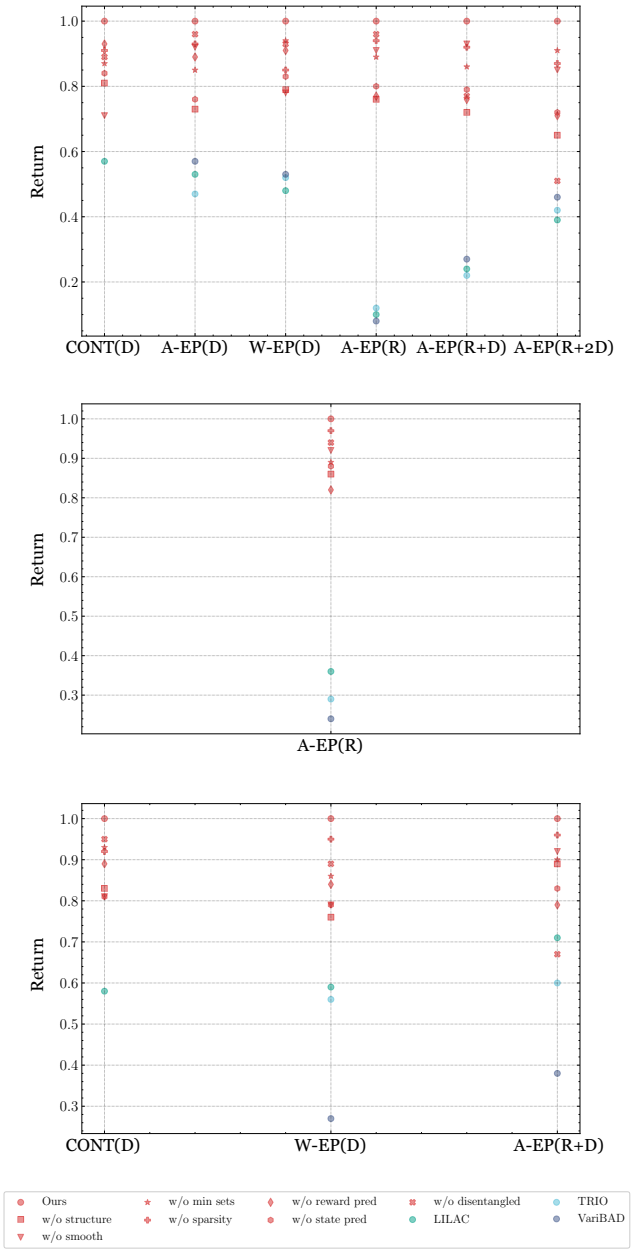

Figure A6: Ablation studies on different components in FANS-RL on (a) Half-Cheetah experiment; (b) Sawyer experiment; and (c) Minitaur experiments. CONT, A-, W-EP indicate continuous, across-episode, and within-episode changes, respectively. (D) and (R) represent changes on dynamics and reward functions, respectively. Best viewed in color.

To verify the effectiveness of each component in our proposed framework, we consider the following ablation studies:

- Without smoothness loss ($\mathcal{L}_{\text{smooth}}$);
- Without structural relationships ($\boldsymbol{C}^{\leftrightarrow}$);
- Without compact representations ($\boldsymbol{s}^{min}, \boldsymbol{\theta}^{min}$);

- Without sparsity losses ($\mathcal{L}_{\text{sparse}}$);
- Without reward or state prediction losses ($\mathcal{L}_{\text{pred-rw}}$, $\mathcal{L}_{\text{pred-dyn}}$);
- Without the disentangled design of CF inference networks for dynamics (q_ϕ^s) and rewards (q_ϕ^r). Specifically, we use one CF inference encoder and the mixed latent space of $\boldsymbol{\theta}^s$ and $\boldsymbol{\theta}^r$ in this setting.

As shown in Fig. A6, all the studied components benefit the performance. Furthermore, FANS-RL can still outperform the strong baselines even without some of the components.

We also test different smoothness losses, including the moving average (MA) $\mathcal{L}_{\text{smooth}} = \sum_{t=2}^{T} \left(\|\theta_t - (\theta_{t-1} + \theta_{t-2} + \ldots + \theta_{t-T})/T\|_1\right)$ and exponential moving average (EMA) $\mathcal{L}_{\text{smooth}} = \sum_{t=2}^{T} \left(\|\theta_t - (\beta\theta_{t-1} + (1-\beta)\mathbf{v}_{t-2})\|_1\right)$, where $\mathbf{v}_t = \beta\theta_t + (1-\beta)\mathbf{v}_{t-1}$ and $\mathbf{v}_0$ is a zero vector. Table C.5 shows the normalized final results of using different smoothness losses. We can find that different smoothness losses have comparable performances.

	Ours	MA ($T = 2$)	EMA ($\beta = 0.98$)
Half-Cheetah: A-EP (D_1)	1.00	1.02	0.89
Half-Cheetah: A-EP (D_2)	1.00	0.96	0.90
Half-Cheetah: W-EP (D)	1.00	0.88	1.05
Half-Cheetah: CONT (D)	1.00	1.04	0.95
Half-Cheetah: A-EP (R)	1.00	0.93	0.82
Half-Cheetah: A-EP (R+D)	1.00	1.09	1.02
Sawyer-Reaching: A-EP (R)	1.00	0.97	0.91
Minitaur: CONT (D)	1.00	1.08	0.96
Minitaur: W-EP (D)	1.00	0.86	1.03
Minitaur: A-EP (R+D)	1.00	0.97	0.94

Table A2: Average final return of using different smoothness losses.

C.6 Visualization on the learned change factors

Fig. A7 gives the visualization on the learned $\boldsymbol{\theta}$ in Half-Cheetah. Fig. A7(a-b) show the pairwise Euclidean distance between learned $\boldsymbol{\theta}^r$ and the axes denote the values of change factors on rewards. Similarly, Fig. A7(c-d) displays the Euclidean distance between learned $\boldsymbol{\theta}^s$ and the values of change factors on dynamics. The results suggest that there is a positive correlation between the distance of learned $\boldsymbol{\theta}$ versus the true change factors. This can verify that $\boldsymbol{\theta}$ can capture the change factors in the system.

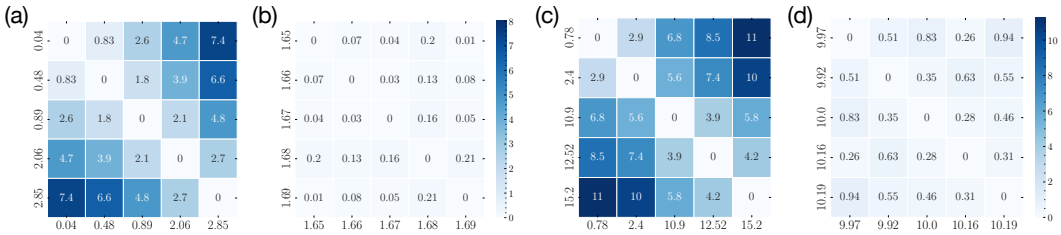

Figure A7: Visualization on the learned $\boldsymbol{\theta}^r$ and $\boldsymbol{\theta}^s$. Best viewed in color.

D Details on the Factored Adaptation Framework

D.1 Algorithm pipelines of FN-VAE

Alg. A1 gives the full pipeline of FN-VAE.

D.2 The framework dealing with discrete and across-episode changes

Algorithm A1: Learning FN-MDPs using FN-VAE.

Input: Trajectories $\boldsymbol{\tau}$, FN-VAE parameters $\phi = (\phi^s, \phi^r)$, $\alpha = (\alpha_1, \alpha_2)$, $\beta = (\beta_1, \beta_2)$, γ; Mask matrices $G = (\boldsymbol{C}^{\cdot\to\cdot})$, Boolean updateG, Learning rates $\lambda_\phi, \lambda_\gamma, \lambda_\alpha, \lambda_\beta, \lambda_G$, Length of collected rollouts k; Number of training epochs E.
Output: ϕ, α_1, α_2, β_1, β_2, γ
for $i = 1, 2, \ldots,$ E **do**
  Randomly sample a batch of trajectories $\boldsymbol{\tau}_{0:k}$ in $\boldsymbol{\tau}$
  `# Infer the latent change factors`
  for j = s, r **do**
    Infer $\mu_{\phi^j}(\boldsymbol{\tau}_{0:k})$ and $\sigma^2_{\phi^j}(\boldsymbol{\tau}_{0:k})$ using q_{ϕ^j}
    Infer $\mu_{\gamma^j}(\boldsymbol{\theta}^j_{0:k})$ and $\sigma^2_{\gamma^j}(\boldsymbol{\theta}^j_{0:k})$ using p_{γ^j}
    Sample $\boldsymbol{\theta}^j_{0:k} \sim \mathcal{N}\left(\mu_{\phi^j}(\boldsymbol{\tau}_{0:k}), \sigma^2_{\phi^j}(\boldsymbol{\tau}_{0:k})\right)$
  end for
  Reconstruct and predict $\hat{\boldsymbol{s}}_{0:k}$, $\hat{\boldsymbol{s}}_{1:k}$, $\hat{\boldsymbol{r}}_{0:k}$, $\hat{\boldsymbol{r}}_{1:k}$ using p_{α_1}, p_{α_2}, p_{β_1}, and p_{β_2}
  `# Update the FN-VAE model`
  $\phi \leftarrow \phi - \lambda_\phi \nabla_\phi \mathcal{L}_{\text{VAE}}$
  $\gamma \leftarrow \gamma - \lambda_\gamma \nabla_\gamma (\mathcal{L}_{\text{KL}} + \mathcal{L}_{\text{smooth}})$
  $\alpha \leftarrow \alpha - \lambda_\alpha \nabla_\alpha (\mathcal{L}_{\text{rec-dyn}} + \mathcal{L}_{\text{pred-dyn}})$
  $\beta \leftarrow \beta - \lambda_\beta \nabla_\beta (\mathcal{L}_{\text{rec-rw}} + \mathcal{L}_{\text{pred-rw}})$
  if updateG **then**
    $G \leftarrow G - \lambda_G \nabla_G (\mathcal{L}_{\text{rec-dyn}} + \mathcal{L}_{\text{rec-rw}} + \mathcal{L}_{\text{KL}} + \mathcal{L}_{\text{sparse}})$
  end if
end for

Alg. A2 gives the extended framework for handling both across- and within-episode changes in non-stationary RL, respectively. The major difference between Alg. A2 and Alg. 1 is that we only infer θ using via CF dynamics networks at change points. Furthermore, we also adjust the objective functions of FN-VAE to fit the discrete changes. At timestep t in episode n, where $t_m \leq \big((n-1)\cdot H + t\big) < t_{m+1}$, we have:

• Prediction and reconstruction losses:

$$\begin{aligned}\mathcal{L}_{\text{rec-dyn}} &= \sum_{t=1}^{T-2} \mathbb{E}_{\theta^s_{t_m}\sim q_\phi} \log p_{\alpha_1}(\boldsymbol{s}_t|\boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1}, \boldsymbol{\theta}^{\boldsymbol{s}}_{t_m}; \boldsymbol{C}^{\cdot\to\boldsymbol{s}}) \\ \mathcal{L}_{\text{pred-dyn}} &= \sum_{t=1}^{T-2} \mathbb{E}_{\theta^s_{t_m}\sim q_\phi} \log p_{\alpha_2}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{\theta}^{\boldsymbol{s}}_{t_m})\end{aligned} \tag{A3}$$

$$\begin{aligned}\mathcal{L}_{\text{rec-rw}} &= \sum_{t=1}^{T-2} \mathbb{E}_{\theta^r_{t_m}\sim q_\phi} \log p_{\beta_1}(r_t|\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{\theta}^{r}_{t_m}; \boldsymbol{c}^{\boldsymbol{s}\to r}, \boldsymbol{c}^{\boldsymbol{a}\to r}) \\ \mathcal{L}_{\text{pred-rw}} &= \sum_{t=1}^{T-2} \mathbb{E}_{\theta^r_{t_m}\sim q_\phi} \log p_{\beta_2}(r_{t+1}|\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}, \boldsymbol{\theta}^{r}_{t_m})\end{aligned} \tag{A4}$$

• KL loss:

$$\begin{aligned}\mathcal{L}_{\text{KL}} = &\sum_{t=2}^{T} \text{KL}\big(q_{\phi^s}(\boldsymbol{\theta}^{\boldsymbol{s}}_{t_m}|\boldsymbol{\theta}^{\boldsymbol{s}}_{t_{m-1}}, \boldsymbol{\tau}_{0:t}))\|p_{\gamma^s}(\boldsymbol{\theta}^{\boldsymbol{s}}_{t_m}|\boldsymbol{\theta}^{\boldsymbol{s}}_{t_{m-1}}; \boldsymbol{C}^{\boldsymbol{\theta}^{\boldsymbol{s}}\to\boldsymbol{\theta}^{\boldsymbol{s}}})\big) \\ &+\text{KL}\big(q_{\phi^r}(\boldsymbol{\theta}^{r}_{t_m}|\boldsymbol{\theta}^{r}_{t_{m-1}}, \boldsymbol{\tau}_{0:t}))\|p_{\gamma^r}(\boldsymbol{\theta}^{r}_{t_m}|\boldsymbol{\theta}^{r}_{t_{m-1}}; \boldsymbol{C}^{\boldsymbol{\theta}^{r}\to\boldsymbol{\theta}^{r}})\big)\end{aligned} \tag{A5}$$

• Sparsity loss:

$$\begin{aligned}\mathcal{L}_{\text{sparse}} = &w_1\|\boldsymbol{C}^{\boldsymbol{s}\to\boldsymbol{s}}\|_1 + w_2\|\boldsymbol{C}^{\boldsymbol{a}\to\boldsymbol{s}}\|_1 + w_3\|\boldsymbol{C}^{\boldsymbol{\theta}^{\boldsymbol{s}}\to\boldsymbol{s}}\|_1 \\ &+ w_6\|\boldsymbol{C}^{\boldsymbol{\theta}^{\boldsymbol{s}}\to\boldsymbol{\theta}^{\boldsymbol{s}}}\|_1 + w_7\|\boldsymbol{C}^{\boldsymbol{\theta}^{r}\to\boldsymbol{\theta}^{r}}\|_1 \\ &+ w_4\|\boldsymbol{c}^{\boldsymbol{s}\to r}\|_1 + w_5\|\boldsymbol{c}^{\boldsymbol{a}\to r}\|_1\end{aligned} \tag{A6}$$

Algorithm A2: Factored Adaptation for non-stationary RL (discrete changes.)

Init: Env; VAE parameters: $\phi = (\phi^s, \phi^r)$, $\alpha = (\alpha_1, \alpha_2)$, $\beta = (\beta_1, \beta_2)$, γ; Mask matrices: $\boldsymbol{C}^{\cdot\to\cdot}$; Policy parameters: ψ; replay buffer: $\mathcal{D}$; Number of episodes: N; Episode horizon: H; Change index $\tilde{\boldsymbol{t}} = \{t_1, \ldots, t_M\}$; $m = 0$.
Output: ϕ, α_1, α_2, β_1, β_2, γ, ψ
`# Model initialization`
Collect multiple trajectories $\boldsymbol{\tau} = \{\boldsymbol{\tau}^1_{0:k}, \boldsymbol{\tau}^2_{0:k}, \ldots\}$ with policy π_ψ from Env;
Learn an initial VAE model on $\boldsymbol{\tau}$ (Alg. A1)
Identify the compact representations $\boldsymbol{s}^{min}$ and change factors $\boldsymbol{\theta}^{min}$ based on $\boldsymbol{C}^{\cdot\to\cdot}$
`# Model estimation & policy learning`
for $n = 0, \ldots, N-1$ **do**
  for $t = 0, \ldots, H-1$ **do**
    Observe $\boldsymbol{s}_t$ from Env;
    `# Estimating latent change factors`
    if $n \cdot (H-1) + t \in \tilde{\boldsymbol{t}}$ **then**
      $m \leftarrow m + 1$
      for j = s, r **do**
        Infer $\mu_{\gamma^j}(\boldsymbol{\theta}^j_{t_{m-1}})$ and $\sigma^2_{\gamma^j}(\boldsymbol{\theta}^j_{t_{m-1}})$ via p_{γ^j}
        Sample $\boldsymbol{\theta}^j_{t_m} \sim \mathcal{N}\left(\mu_{\gamma^j}(\boldsymbol{\theta}^j_{t_{m-1}}), \sigma^2_{\gamma^j}(\boldsymbol{\theta}^j_{t_{m-1}})\right)$
      end for
    end if
    Generate $\boldsymbol{a}_t \sim \pi_\psi(\boldsymbol{a}_t \mid \boldsymbol{s}^{min}_t, \boldsymbol{\theta}^{min}_{t_m})$
    Receive $r_{n,t}$ from Env
    Add $(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{\theta}^s_{t_m}, \boldsymbol{\theta}^r_{t_m})$ to replay buffer $\mathcal{D}$;
    Extract a trajectory with length k from $\mathcal{D}$;
    Learn VAE (Alg. A1) with updateG=False;
    Sample a batch of data from $\mathcal{D}$
    Update policy network parameters ψ
  end for
end for

• Smoothness loss:

$$\mathcal{L}_{\text{smooth}} = \sum_{t=2}^{T} \left(||\boldsymbol{\theta}^{\boldsymbol{s}}_{t_m} - \boldsymbol{\theta}^{\boldsymbol{s}}_{t_{m-1}}||_1 + ||\boldsymbol{\theta}^{\boldsymbol{r}}_{t_m} - \boldsymbol{\theta}^{\boldsymbol{r}}_{t_{m-1}}||_1 \right) \tag{A7}$$

The total loss $\mathcal{L}_{\text{vae}} = k_1\mathcal{L}_{\text{rec}} + k_2\mathcal{L}_{\text{pred}} - k_3\mathcal{L}_{\text{KL}} - k_4\mathcal{L}_{\text{sparse}} - k_5\mathcal{L}_{\text{smooth}}$, where k_1, k_2, k_3, k_4, and k_5 are adjustable hyper-parameters to balance the objective functions.

D.3 The framework dealing with raw pixels

We augment the generative process in Eq. 1-3 with the generative process of observation.

$$o_t = u_i(c_i^{\boldsymbol{s}\to\boldsymbol{o}} \odot \boldsymbol{s}_t, \epsilon^o_t), \tag{A8}$$

where u is a non-linear function and $i = 1, \ldots, d$. $\boldsymbol{c}^{\boldsymbol{s}\to\boldsymbol{o}} := [c_i^{\boldsymbol{s}\to\boldsymbol{o}}]_{i=1}^d$. ϵ^o_t is an i.i.d. random noise. To learn the u_i, we model the states as the latent variables in FN-VAE. Fig. A8 gives the modified FN-VAE dealing with raw pixels, where the states are also in the latent space. Different from the original FN-VAE, we incorporate state inference networks and state dynamics networks. Moreover, we reconstruct and predict the current and future observations using the observation decoder. Detailed objective functions are given below.[1]

• Prediction and reconstruction losses

[1]Here we give the example of handling the discrete changes.

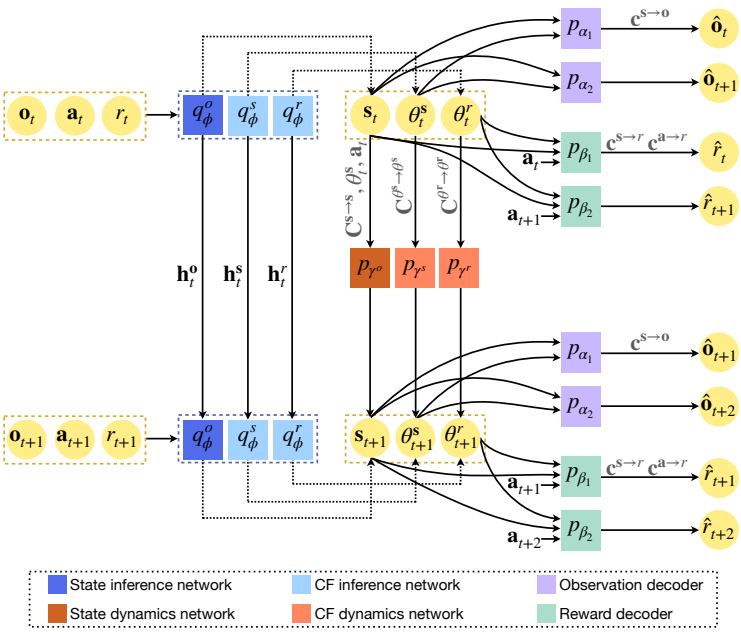

Figure A8: The architecture of FN-VAE using raw pixel as input.

$$
\begin{aligned}
\mathcal{L}_{\text{rec-obs}} &= \sum_{t=1}^{T-2} \mathbb{E}_{s_t \sim q_{\phi^o}} \log p_{\alpha_1}(\boldsymbol{o}_t | \boldsymbol{s}_t; \boldsymbol{c}^{\boldsymbol{s}\to\boldsymbol{o}}) \\
\mathcal{L}_{\text{pred-obs}} &= \sum_{t=1}^{T-2} \mathbb{E}_{s_t \sim q_{\phi^o}} \log p_{\alpha_2}(\boldsymbol{o}_{t+1} | \boldsymbol{s}_t, \boldsymbol{\theta}^{\boldsymbol{s}}_{t_m})
\end{aligned}
\tag{A9}
$$

$$
\begin{aligned}
\mathcal{L}_{\text{rec-rw}} &= \sum_{t=1}^{T-2} \mathbb{E}_{(\theta^r_{t_m} \sim q_\phi, s_t \sim q_{\phi^o})} \log p_{\beta_1}(r_t | \boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{\theta}^r_{t_m}; \boldsymbol{c}^{\boldsymbol{s}\to\boldsymbol{r}}, \boldsymbol{c}^{\boldsymbol{a}\to\boldsymbol{r}}) \\
\mathcal{L}_{\text{pred-rw}} &= \sum_{t=1}^{T-2} \mathbb{E}_{(\theta^r_{t_m} \sim q_\phi, s_t \sim q_{\phi^o})} \log p_{\beta_2}(r_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_{t+1}, \boldsymbol{\theta}^r_{t_m})
\end{aligned}
\tag{A10}
$$

• KL loss

$$
\begin{aligned}
\mathcal{L}_{\text{KL}} = \sum_{t=2}^{T} & \text{KL}\big(q_{\phi^s}(\boldsymbol{\theta}^{\boldsymbol{s}}_{t_m} | \boldsymbol{\theta}^{\boldsymbol{s}}_{t_{m-1}}, \boldsymbol{\tau}_{0:t})) \| p_{\gamma^s}(\boldsymbol{\theta}^{\boldsymbol{s}}_{t_m} | \boldsymbol{\theta}^{\boldsymbol{s}}_{t_{m-1}}; \boldsymbol{C}^{\boldsymbol{\theta}^{\boldsymbol{s}}\to\boldsymbol{\theta}^{\boldsymbol{s}}})\big) \\
& + \text{KL}\big(q_{\phi^r}(\boldsymbol{\theta}^{r}_{t_m} | \boldsymbol{\theta}^{r}_{t_{m-1}}, \boldsymbol{\tau}_{0:t})) \| p_{\gamma^r}(\boldsymbol{\theta}^{r}_{t_m} | \boldsymbol{C}^{\boldsymbol{o}_t\to\boldsymbol{\theta}^r})\big) \\
& + \text{KL}\big(q_{\phi^o}(\boldsymbol{s}_t | \boldsymbol{\tau}_{0:t}, \boldsymbol{\theta}^{s}_{t_m})) \| p_{\gamma^o}(\boldsymbol{s}_t | \boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1}, \boldsymbol{\theta}^{s}_{t_m}; \boldsymbol{C}^{\boldsymbol{s}\to\boldsymbol{s}}, \boldsymbol{C}^{\boldsymbol{a}\to\boldsymbol{s}}, \boldsymbol{C}^{\boldsymbol{\theta}^s\to\boldsymbol{s}})\big)
\end{aligned}
\tag{A11}
$$

where $\boldsymbol{\tau}_{0:t} = \{\boldsymbol{o}_0, r_0, \boldsymbol{o}_1, r_1, \ldots, \boldsymbol{o}_t, r_t\}$.

• Sparsity loss

$$
\begin{aligned}
\mathcal{L}_{\text{sparse}} = & w_1 \|\boldsymbol{C}^{\boldsymbol{s}\to\boldsymbol{s}}\|_1 + w_2 \|\boldsymbol{C}^{\boldsymbol{a}\to\boldsymbol{s}}\|_1 + w_3 \|\boldsymbol{C}^{\boldsymbol{\theta}^{\boldsymbol{s}}\to\boldsymbol{s}}\|_1 \\
& + w_4 \|\boldsymbol{c}^{\boldsymbol{s}\to\boldsymbol{r}}\|_1 + w_5 \|\boldsymbol{c}^{\boldsymbol{a}\to\boldsymbol{r}}\|_1 \\
& + w_6 \|\boldsymbol{C}^{\boldsymbol{\theta}^{\boldsymbol{s}}\to\boldsymbol{\theta}^{\boldsymbol{s}}}\|_1 + w_7 \|\boldsymbol{C}^{\boldsymbol{\theta}^{r}\to\boldsymbol{\theta}^{r}}\|_1 + w_8 \|\boldsymbol{C}^{\boldsymbol{s}\to\boldsymbol{o}}\|_1
\end{aligned}
\tag{A12}
$$

• Smooth loss

$$
\mathcal{L}_{\text{smooth}} = \sum_{t=2}^{T} \Big(||\boldsymbol{\theta}^{\boldsymbol{s}}_{t_m} - \boldsymbol{\theta}^{\boldsymbol{s}}_{t_{m-1}}||_1 + ||\boldsymbol{\theta}^{\boldsymbol{r}}_{t_m} - \boldsymbol{\theta}^{\boldsymbol{r}}_{t_{m-1}}||_1 \Big)
\tag{A13}
$$

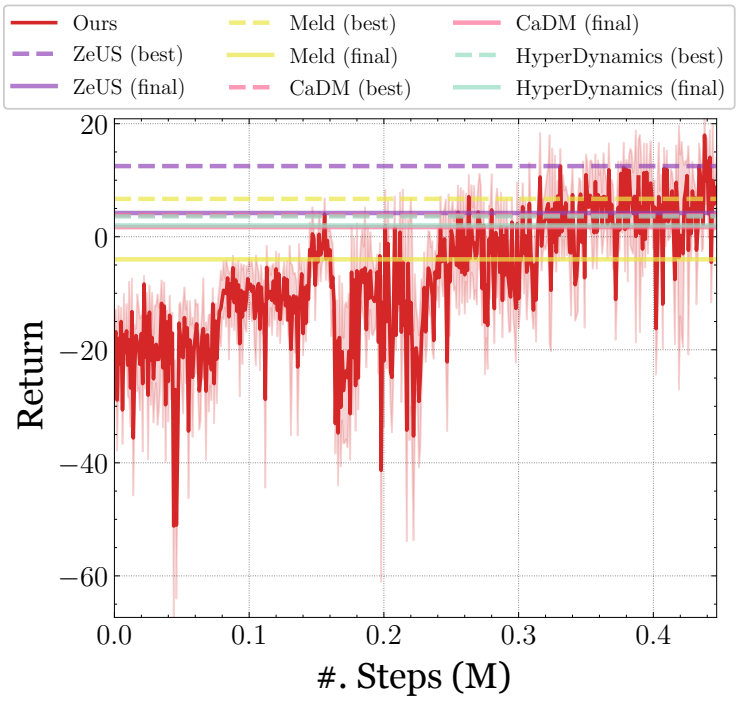

Figure A9: Average return across 10 runs on Sawyer-Peg (raw pixels) with across-episode changes.

D.4 Hyper-parameter selection

D.4.1 Factored model estimation

Input with symbolic states. In Half-Cheetah, Sawyer-Reaching, and Minitaur the symbolic states are observable. For the CF dynamic networks, we use 2-layer fully connected networks. The number of neurons is 512. For CF inference networks, we use 2-layer fully connected networks, where the number of neurons is 256, followed by the LSTM networks with 256 hidden units. The initial learning rates for all losses are set to be 0.1 with a decay rate 0.99. The batch size is 256 and the length of time steps is equal to the horizon in each task. The number of RNN cells is 256. The decoder networks are 2-layer fully connected networks. The number of neurons is 512.

Input with raw pixels. In Saywer-Peg, we directly learn and adapt in non-stationary environments with raw pixels observed. Different from other experiments, we use the architecture described in Fig. A8. At timestep t, we stack 4 frames as the input $\boldsymbol{o}_t$. A 5-layer convolutional networks is used to extract the features of the trajectories of observations and rewards. The layers have 32, 64, 128, 256, and 256 filters. And the corresponding filter sizes are 5, 3, 3, 3, 4. The observation decoders are the transpose of the convolutional networks. Then the extracted features are used as the input of LSTM networks in state inference networks. The state inference networks and state dynamic networks share the same architectures with the CF inference and dynamics networks, respectively. We use the same CF inference networks, CF dynamics networks, and reward decoders with those in cases with symbolic states as input. The number of latent features is 40.

Balancing parameters in losses For all experiments:

- All $w_\cdot$ are set to be 0.1;
- Weights of the reconstruction loss: $k_1 = 0.8$;
- Weights of the prediction loss: $k_2 = 0.8$;
- Weights of KL loss: $k_3 = 0.5$;
- Weights of sparsity loss: $k_4 = 0.1$;

- Weights of smooth loss: $k_5 = 0.02$.

We use the automatic weighting method in [16] to learn the weights for $k_1, \ldots, K_5$ and grid search for $w_1, \ldots, w_7$.

Model initialization. Table A3, A4, and A5 provide the settings of learning the model initialization.

	CONT (D)	A-EP (D)	W-EP (D)	A-EP (R)	A-EP (R+D)
# trajectories	500	20	20	20	100
# steps in each episode	50	50	50	50	50
# episodes	10	100	100	100	100

Table A3: The selected hyper-parameters for model estimation in Half-Cheetah experiment.

	Sawyer-Reaching	Sawyer-Peg
# trajectories	500	20
# steps in each episode	150	40
# episodes	10	100

Table A4: The selected hyper-parameters for model estimation in Saywer experiments.

	CONT (D)	W-EP (D)	A-EP (R+D)
# trajectories	500	50	80
# steps in each episode	100	100	100
# episodes	10	50	100

Table A5: The selected hyper-parameters for model estimation in Minitaur experiments.

D.4.2 Policy learning

In the Half-Cheetah, Sawyer-Reaching, and Minitaur experiments, we follow the learning rates selection for policy networks in [7]. In Sawyer-Peg, for both actor and critic networks, we use 2-layer fully-connected networks. The number of neurons is 256. For all experiments, we use standard Gaussian to initialize the parameters of policy networks. The learning rate is $3e-4$. The relay buffer capacity is $50,000$. The number of batch size is 256.

Details on TRIO and VariBAD. For TRIO and VariBAD, we meta-train the models (batch size: 5000, # epochs: 2 for all experiments) and show the learning curves of meta-testing. The tasks parameters for meta-training are uniformly sampled from a Gaussian distribution. For all approaches, we use the same set of hyper-parameters for policy optimization modules (i.e., SAC). For the latent parameters in TRIO, we follow the original paper where the latent space from the inference network is projected to a higher dimension. The number of latent parameters for TRIO is the same as those in other approaches (Half-Cheetah and Minitaur: 40, Sawyer-Reaching: 20). We compared with TS-TRIO, with the kernels set as in the original implementation.

E Experimental Platforms and Licenses

E.1 Platforms

All methods are implemented on 8 Intel Xeon Gold 5220R and 4 NVidia V100 GPUs.

E.2 Licenses

In our code, we have used the following libraries which are covered by the corresponding licenses:

- Tensorflow (Apache License 2.0),
- Pytorch (BSD 3-Clause "New" or "Revised" License),
- OpenAI Gym (MIT License),
- OpenCV (Apache 2 License),
- Numpy (BSD 3-Clause "New" or "Revised" License)
- Keras (Apache License).