Factored Adaptation for Non-Stationary Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Dealing with non-stationarity in environments (i.e., transition dynamics) and objectives (i.e., 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 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 and 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 changing function types and changing frequency, including changes across episodes and within episodes. Experimental results demonstrate that FANS-RL outperforms existing approaches in terms of rewards, 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 recent causality literature [9] 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 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 [10, 11, 12], for which we learn the structure, with latent change factors that evolve in time following a Markov process.

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

We build upon the AdaRL framework [13], 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 rewards, 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 MDP [10, 11, 12] 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* $\{s_{1,t}, \dots, s_{d,t}, a_{1,t}, \dots, a_{m,t}, r_t, \theta^{\boldsymbol{s}}_{1,t}, \dots, \theta^{\boldsymbol{s}}_{p,t}, \theta^r_{1,t}, \dots, \theta^r_{q,t}\}$*. 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* $\boldsymbol{\theta}^{\boldsymbol{s}}_{\boldsymbol{t}}$*. We also assume a given initial state distribution* $\mathbb{P}_s(\boldsymbol{s}_0)$*. Similarly, we can define the reward function* $\mathcal{R}$ *as a function of only the parents of* r_t *in* $\mathcal{G}$*, i.e.,* $\mathcal{R}(\mathbf{s}_t, \mathbf{a}_t, \boldsymbol{\theta}^{\boldsymbol{r}}_t) = \mathcal{R}(pa(r_t))$*, where* $pa(r_t)$ *are a subset of dimensions of* $\mathbf{s}_t, \mathbf{a}_t$*, and* $\boldsymbol{\theta}^{\boldsymbol{r}}_t$*. We define the factored latent change factors transition distributions* $\mathbb{P}_{\theta^s}$ *and* $\mathbb{P}_{\theta^r}$ *as:*

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

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

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 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 $\mathcal{G}$.

Generative environment model. We adapt the generative model in AdaRL [13] 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 $i = 1, \dots, d$ of $\mathbf{s}_t$ is:

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

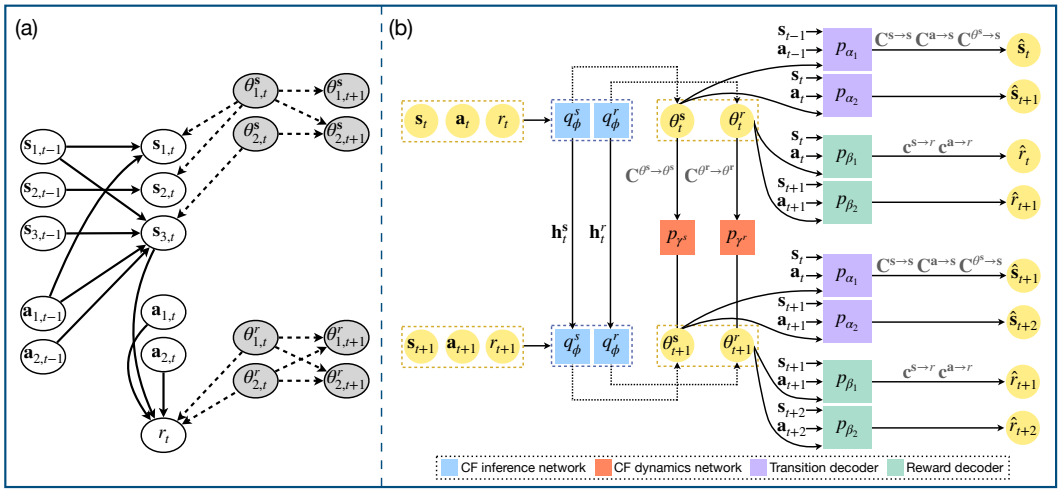

Figure 1: (a). A graphical representation of an FN-MDP. For readability, we only illustrate a subsection of dimensions of states, actions, and latent change factors. The shaded variables are unobserved; (b).The architecture of FN-VAE, which learns the generative model, explained in Sec. 3.

where $\odot$ is the element-wise product, f_i are non-linear functions and $\epsilon^s_{i,t}$ 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 $\boldsymbol{\theta}^{\boldsymbol{s}}_t \in \mathbb{R}^p$ 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 $\boldsymbol{\theta}^{\boldsymbol{s}}_t$ influence $s_{i,t}$. We model the reward function as:

$$r_t = h(\boldsymbol{c}^{\boldsymbol{s}\to r} \odot \boldsymbol{s}_{t-1}, \boldsymbol{c}^{\boldsymbol{a}\to r} \odot \boldsymbol{a}_{t-1}, \boldsymbol{\theta}^r_t, \epsilon^r_t) \tag{2}$$

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^r_t is an i.i.d. random noise. The change factor $\boldsymbol{\theta}^r_t \in \mathbb{R}^q$ 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:

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

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 $\epsilon^s_{i,t}$, ϵ^r_t, $\epsilon^{\theta^s}_t$ and $\epsilon^{\theta^r}_t$, the i.i.d. random noises. Although $\boldsymbol{c}^{\cdot\to\cdot}$ and ϵ are stationary, we allow the change of graph structure and noise distributions, whose changes are captured by $\boldsymbol{\theta}$ instead. We group the binary masks in transition distributions in the matrix $\boldsymbol{C}^{\boldsymbol{s}\to\boldsymbol{s}} := [\boldsymbol{c}_i^{\boldsymbol{s}\to\boldsymbol{s}}]_{i=1}^d$, $\boldsymbol{C}^{\theta_k\to\boldsymbol{s}} := [\boldsymbol{c}_i^{\theta_k\to\boldsymbol{s}}]_{i=1}^d$, 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 $\boldsymbol{C}^{\boldsymbol{\theta}^{\boldsymbol{s}}\to\boldsymbol{\theta}^{\boldsymbol{s}}} := [\boldsymbol{c}_j^{\boldsymbol{\theta}^{\boldsymbol{s}}\to\boldsymbol{\theta}^{\boldsymbol{s}}}]_{j=1}^p$ 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, e.g. a continuous g^s and a piecewise-constant g^r.

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 [13] 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 $\boldsymbol{C}^{\cdot \to \cdot}$ 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 $\boldsymbol{C}^{\boldsymbol{\theta}^{\boldsymbol{s}} \to \boldsymbol{s}}$. 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 $\boldsymbol{C}^{\boldsymbol{s} \to \boldsymbol{s}}$, $\boldsymbol{C}^{\boldsymbol{a} \to \boldsymbol{s}}$, $\boldsymbol{c}^{\boldsymbol{s} \to r}$ and $\boldsymbol{c}^{\boldsymbol{a} \to r}$ 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 [14], *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 $q_{\phi^s}\left(\boldsymbol{\theta}_{\boldsymbol{t}}^{\boldsymbol{s}} \mid \boldsymbol{s}_t, \boldsymbol{a}_t\right)$ and $q_{\phi^r}\left(\boldsymbol{\theta}_t^r \mid \boldsymbol{s}_t, \boldsymbol{a}_t, r_t\right)$ are parameterised by ϕ^s and ϕ^r, respectively. To model the time-dependence of $\boldsymbol{\theta}_{\boldsymbol{t}}^{\boldsymbol{s}}$ and $\boldsymbol{\theta}_t^r$, we use LSTMs [14] 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 $p_{\gamma^s}\left(\boldsymbol{\theta}_{t+1}^{\boldsymbol{s}} \mid \boldsymbol{\theta}_t^{\boldsymbol{s}}, \boldsymbol{C}^{\boldsymbol{\theta}^{\boldsymbol{s}} \to \boldsymbol{\theta}^{\boldsymbol{s}}}\right)$ and $p_{\gamma^r}\left(\boldsymbol{\theta}_{t+1}^{\boldsymbol{r}} \mid \boldsymbol{\theta}_t^{\boldsymbol{r}}, \boldsymbol{C}^{\boldsymbol{\theta}^{\boldsymbol{r}} \to \boldsymbol{\theta}^{\boldsymbol{r}}}\right)$. 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.

$$\begin{aligned}\mathcal{L}_{\text{KL}} = \sum_{t=2}^{T} & \text{KL}\big(q_{\phi^s}(\boldsymbol{\theta}_t^{\boldsymbol{s}} | \boldsymbol{\theta}_{t-1}^{\boldsymbol{s}}, \boldsymbol{\tau}_{0:t})) \| p_{\gamma^s}(\boldsymbol{\theta}_t^{\boldsymbol{s}} | \boldsymbol{\theta}_{t-1}^{\boldsymbol{s}}; \boldsymbol{C}^{\boldsymbol{\theta}^{\boldsymbol{s}} \to \boldsymbol{\theta}^{\boldsymbol{s}}})\big) \\ & + \text{KL}\big(q_{\phi^r}(\boldsymbol{\theta}_t^{r} | \boldsymbol{\theta}_{t-1}^{r}, \boldsymbol{\tau}_{0:t})) \| p_{\gamma^r}(\boldsymbol{\theta}_t^{r} | \boldsymbol{\theta}_{t-1}^{r}; \boldsymbol{C}^{\boldsymbol{\theta}^{r} \to \boldsymbol{\theta}^{r}})\big)\end{aligned} \tag{4}$$

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 $\boldsymbol{C}^{\cdot\to\boldsymbol{s}} := (\boldsymbol{C}^{\boldsymbol{s}\to\boldsymbol{s}}, \boldsymbol{C}^{\boldsymbol{a}\to\boldsymbol{s}}, \boldsymbol{C}^{\boldsymbol{\theta}^{\boldsymbol{s}}\to\boldsymbol{s}})$. At timestep t, the reconstruction encoder $p_{\alpha_1}(\boldsymbol{s}_t \mid \boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1}, \boldsymbol{\theta}_t^s; \boldsymbol{C}^{\cdot\to\boldsymbol{s}})$ reconstructs the state from current state $\boldsymbol{s}_t$ with sampled $\boldsymbol{\theta}_t^{\boldsymbol{s}}$. The one-step prediction encoder $p_{\alpha_2}(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{\theta}_t^{\boldsymbol{s}})$ instead tries to approximate the next state $\boldsymbol{s}_{t+1}$. The corresponding loss functions are given below:

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

Reward decoders (green boxes in Fig. 1(b)). Similarly, we use a reconstruction encoder $p_{\beta_1}(r_t \mid \boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{\theta}_t^r, \boldsymbol{c}^{\boldsymbol{s}\to r}, \boldsymbol{c}^{\boldsymbol{a}\to r})$, parameterized by β_1, and a one-step prediction encoder $p_{\beta_2}(r_{t+1} \mid \boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}, \boldsymbol{\theta}_t^r)$, parametrized by β_2, to approximate the reward function, with the losses:

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

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

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

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

Learning from raw pixels. In some applications, the state variables are not observed, but instead we observe the image pixels. Our framework can be easily extended to this case by adding an encoder ϕ^o to learn the latent state variables from images, similar to other works learning latent states from pixels [15, 16, 13]. We describe the reconstruction and prediction loss of this component in Appendix D.3.

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 shared representation $\boldsymbol{s}^{min}$ and compact domain-specific representation $\boldsymbol{\theta}^{min}$ following AdaRL [13]. In particular, the only dimensions of the state and change factors that are useful for policy learning are those that eventually affect the reward, i.e., have a directed path to the reward in the graph $\mathcal{G}$. The online policy $\pi_\psi\left(\boldsymbol{a}_t \mid \boldsymbol{s}_t^{min}, q_\phi(\boldsymbol{\theta}_t^{min} \mid \boldsymbol{\tau}_{0:t})\right)$ can be learned end-to-end, including learning the FN-VAE, as shown in Alg. 1. We use SAC [17] as our policy learning model, so the policy parameters $\psi = (\pi, Q)$ are the actor $\mathcal{L}_\pi$ and critic loss $\mathcal{L}_Q$.

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 $\boldsymbol{\theta}_t^s$ and $\boldsymbol{\theta}_t^r$ using the CF dynamics networks γ^s and γ^r (Lines 8-22). 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 (Lines 9-13). 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 (Line 23). We receive a reward r_t and add $(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{\theta}_t^s, \boldsymbol{\theta}_t^r)$ to the replay buffer (Lines 24-25). We now update our estimation of the FN-VAE, but we keep the graph G fixed (Lines 26-27). Finally, we sample a batch of trajectories in the replay buffer and update the policy network ψ (Lines 28-29).

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 9-22 to only update the change parameters at the timesteps in $\tilde{\mathbf{t}}$, as shown in Appendix Alg. A2.

Algorithm 1: Factored Adaptation for non-stationary RL

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.
Output: VAE parameters: ϕ, α, β, γ; Policy parameters: ψ
Collect multiple trajectories $\boldsymbol{\tau} = \{\boldsymbol{\tau}^1_{0:k}, \boldsymbol{\tau}^2_{0:k}, \ldots\}$ with policy π_ψ from Env;
Learn FN-VAE from $\boldsymbol{\tau}$, including masks $\boldsymbol{C}^{\cdot\to\cdot}$ that represent the graph $\mathcal{G}$ (Appendix Alg. A1)
Identify the compact representations $\boldsymbol{s}^{min}$ and change factors $\boldsymbol{\theta}^{min}$ based on $\boldsymbol{C}^{\cdot\to\cdot}$
for $n = 0, \ldots, N-1$ **do**
  for $t = 0, \ldots, H-1$ **do**
    Observe $\boldsymbol{s}_t$ from Env;
    if $t = 0$ **then**
      for j = s, r **do**
        Infer mean $\mu_{\gamma^j}(\boldsymbol{\theta}^j_{\text{old}})$ and variance $\sigma^2_{\gamma^j}(\boldsymbol{\theta}^j_{\text{old}})$ for $\boldsymbol{\theta}^j_0$ via p_{γ^j}
        Sample $\boldsymbol{\theta}^j_0 \sim \mathcal{N}\left(\mu_{\gamma^j}(\boldsymbol{\theta}^j_{\text{old}}), \sigma^2_{\gamma^j}(\boldsymbol{\theta}^j_{\text{old}})\right)$
      end for
    else
      for j = s, r **do**
        Infer mean $\mu_{\gamma^j}(\boldsymbol{\theta}^j_{t-1})$ and variance $\sigma^2_{\gamma^j}(\boldsymbol{\theta}_{t-1})$ of the change parameter $\boldsymbol{\theta}^j_t$ via p_{γ^j}
        Sample $\boldsymbol{\theta}^j_t \sim \mathcal{N}\left(\mu_{\gamma^j}(\boldsymbol{\theta}^j_{t-1}), \sigma^2_{\gamma^j}(\boldsymbol{\theta}^j_{t-1})\right)$
      end for
    end if
    if $t = H-1$ **then**
      $\boldsymbol{\theta}^s_{\text{old}} \leftarrow \boldsymbol{\theta}^s_t$ and $\boldsymbol{\theta}^r_{\text{old}} \leftarrow \boldsymbol{\theta}^s_t$;
    end if
    Generate $\boldsymbol{a}_t \sim \pi_\psi(\boldsymbol{a}_t \mid \boldsymbol{s}^{min}_t, \boldsymbol{\theta}^{min}_t)$
    Receive reward $r_{n,t}$ from Env
    Add $(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{\theta}^s_t, \boldsymbol{\theta}^r_t)$ to replay buffer $\mathcal{D}$;
    Extract a trajectory with length k from replay buffer $\mathcal{D}$;
    Learn FN-VAE (Appendix Alg. A1) with updateG=False (i.e. with fixed masks $\boldsymbol{C}^{\cdot\to\cdot}$);
    Sample a batch of data from replay buffer $\mathcal{D}$;
    Update policy network parameters ψ;
  end for
end for

5 Evaluation

We evaluate our approach on four well-established benchmarks, including Half-Cheetah-V3 from MuJoCo [18, 19], Sawyer-Reaching and Sawyer-Peg from Sawyer [20, 16], and Minitaur [21]. 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.

Halfcheetah-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). The reward function is $r_t = -\|v^o_t - v^g_t\|_2 - 0.05\|a_t\|_2$, where v^o and a_t are the agent's velocity and action, respectively, at timestep t. The horizon 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. 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

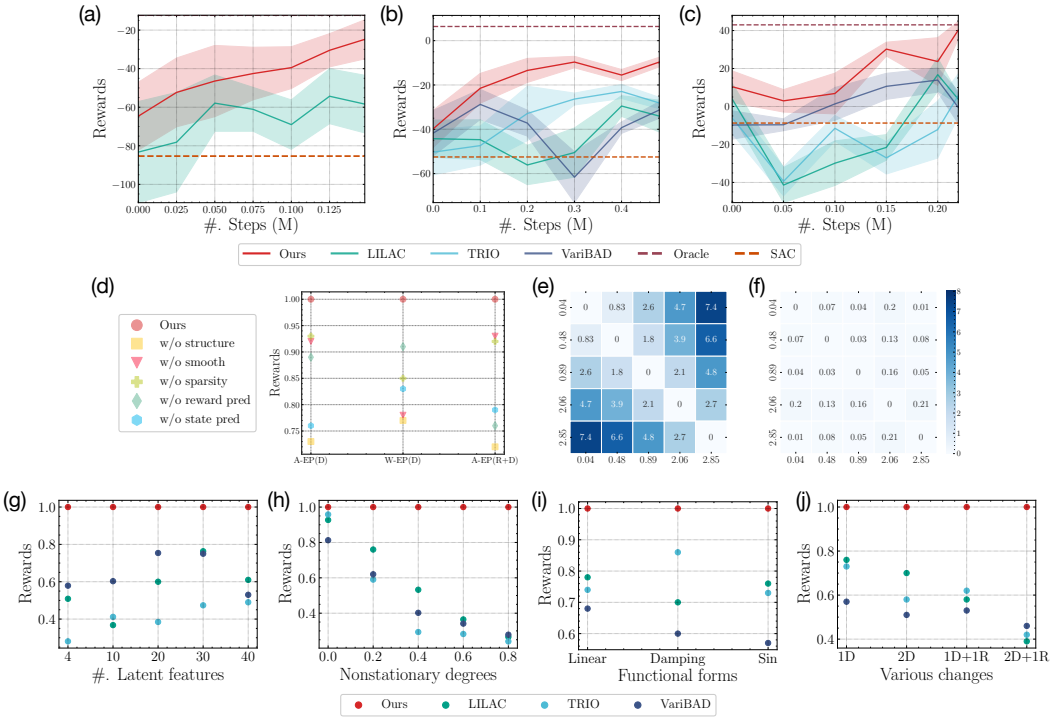

Figure 2: Summary on experimental results. (a)-(c). Average rewards (smoothed) across 10 runs. We only indicate the averaged highest result of all times for oracle and SAC. The shaded region is $(\mu - \sigma, \mu + \sigma)$, where μ is the mean and σ is the standard deviation. (a) Half-Cheetah-V3 with 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): Averaged and normalized final rewards 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.

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. 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 $\boldsymbol{s}^g$. The reward function is $r_t = \mathbb{I}\left(\left\|\boldsymbol{s_t} - \boldsymbol{s}^g\right\|_2 \leq 0.05\right)$. 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 order to compare with similar approaches, e.g., ZeUS [8], CADM [22], Hyperdynamics [23]and Meld [16], 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. We describe the non-stationary settings in Appendix C.2.

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 $r_t = 0.3 - |0.3 - \boldsymbol{s}_{t,v}| - 0.01 \cdot \left\|\boldsymbol{a}_t - 2\boldsymbol{a}_{t-1} + \boldsymbol{a}_{t-2}\right\|_1$. 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.

Methods	ZeUS	Meld	CaDM	Hyper-Dynamics	Ours
Best	12.45	6.38	4.04	3.91	**18.01**
Final	4.42	-4.62	1.70	1.94	**8.63**

Table 1: Average final and best rewards on Sawyer-Peg. The number of random trails is 10.

Baselines. We compare our approach with a meta-RL approach for stationary RL, **VariBAD** [24], 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** [17], 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 rewards* and *the compactness of the latent space* in varying degrees and functional forms of non-stationarity.

Experimental results and ablation studies. Fig. 2(a)-(c) shows the smoothed curves of average rewards across timesteps in a subsection of Half-Cheetah, Sawyer-Reaching, and Minitaur experiments. Smoothed curves of other experiments are given in Appendix Fig. A1 and A2. For continuous changes, we only compare with LILAC [7] since other approaches are not applicable. We smooth the learning curves by randomly selecting a few data points for readability. The original learning curves are in Appendix Fig. A3-A5. Table 1 shows the results on Sawyer-Peg experiments using raw pixels as input, based on the reported results in [8]. For a fair comparison, we indicate the best and final average rewards for each baseline. The learning curve is given in Appendix Fig. A10. Full results 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.5. 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.

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 rewards versus the number of 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 rewards 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 can find that, thanks to the factored modeling, FANS-RL can perform better in those more complicated scenarios with multiple numbers and types of changes.

6 Related work

Early works in non-stationary RL [25, 26] can only detect changes that have already happened instead of anticipating potential changes. If the evolution of non-stationary environments is a Markov or Semi-Markov chain, one can deal with non-stationarity with HM-MDPs [27] or HS3MDPs [28]. Recently, several methods have been developed that learn to anticipate changes in non-stationary deep RL. Chandak et al. [29] propose to maximize future rewards without explicitly modeling the non-stationary environments. Al-Shedivat et al. [4] extend the model-agnostic meta-learning (MAML) [6] framework into 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 can capture the temporal change factors during the meta-testing phase. ReBAL and GrBAL [30] meta-train the dynamic prior, which can adapt 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. Our approach fits in this line of work; however, as opposed to LILAC and ZeUS, which model all temporal changes using a shared embedding space, we use a more efficient, factored representation.

Factored MDPs [10, 11, 12] represent MDPs in which the state transition and reward functions only rely on a subset of state and action variables. Hallak et al. [31] learn factored MDPs greedily to improve sample efficiency in model-based off-policy RL. Balaji et al. [32] employ factored MDP to improve the sample efficiency in both model-free and model-based RL algorithms, where the factored graphs are given by expert knowledge. Working memory graphs [33] learn the factored observation space using Transformer. NeverNet [34] factorizes the state-action space via learning the graph neural networks and shows the performance gain on robotic control tasks. Zholus et al. [35] factorize the visual states into actor-related, the object of manipulation, and the latent influence factor between these two states to achieve sparse interaction and causal influence in robotic control tasks. AFaR [36] learns factored value functions via attention modules to improve sample efficiency and generalization across tasks. In an MDP setting, Mutti et al. [37] learn a causal structure that generalizes across a family of MDPs. Another related line of work is based on the principle of independent causal mechanisms [38, 39]. Recurrent independent mechanisms (RIMs) [40] learn the independent transition dynamics in RL with sparse communication among the latent states. Meta-RIMs [41] leverage meta-learning and soft attention to learn a set of RIMs with competition and communication. The model captures the dynamics from images, and improves the sample efficiency in RL tasks. 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 described Factored Adaptation for Non-Stationary RL (FANS-RL), a framework that learns a factored representation to adapt to non-stationarity that can be combined with any standard RL algorithm. We first formalized our problem as a Factored Non-stationary MDP (FN-MDP), augmenting a factored MDP with latent change factors evolving in time as a Markov process. FN-MDPs do not model a family of MDPs, but instead naturally include the dynamics of latent change factors analogously to the dynamics of the state components. This allows us to capture different types of non-stationarity, e.g., continuous and discrete changes, both within each episode and across different episodes. In order to learn FN-MDPs we proposed FN-VAE, a factored non-stationary variational autoencoder, which we then combined in FANS-RL, an online model estimation and policy evaluation approach. We evaluated FANS-RL on benchmarks for continuous control and robotic manipulation tasks, including with raw 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 stage is computationally expensive, which limits the scalability of our approach. In future work, we plan to meta-learn the causal graphs among different tasks to improve the scalability of our approach and its applicability to other complex RL problems, e.g., multi-agent reinforcement learning.

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. *arXiv preprint arXiv:2110.06972*, 2021.

[9] B. Huang, K. Zhang, J. Zhang, J. Ramsey, R. Sanchez-Romero, C. Glymour, and B. Schölkopf. Causal discovery from heterogeneous/nonstationary data. *Journal of Machine Learning Research*, 21(89):612–634, 2020.

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

[11] 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.

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

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

[21] 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.

[22] 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.

[23] Zhou Xian, Shamit Lal, Hsiao-Yu Tung, Emmanouil Antonios Platanios, and Katerina Fragkiadaki. Hyperdynamics: Meta-learning object and agent dynamics with hypernetworks. *arXiv preprint arXiv:2103.09439*, 2021.

[24] 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.

[25] 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.

[26] 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.

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

[28] 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.

[29] 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.

[30] 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.

[31] 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.

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

[33] 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.

[34] 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.

[35] Artem Zholus, Yaroslav Ivchenkov, and Aleksandr Panov. Factorized world models for learning causal relationships. In *ICLR2022 Workshop on the Elements of Reasoning: Objects, Structure and Causality*, 2022.

[36] Shagun Sodhani, Sergey Levine, and Amy Zhang. Improving generalization with approximate factored value functions. In *ICLR2022 Workshop on the Elements of Reasoning: Objects, Structure and Causality*, 2022.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. Causal inference on time series using restricted structural equation models. In *Advances in Neural Information Processing Systems (NIPS)*, 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. A1-A5) 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]