# Tackling Non-Stationarity in Reinforcement Learning via Causal-Origin Representation

**Wanpeng Zhang** [1] **Yilin Li** [2] **Boyu Yang** [3] **Zongqing Lu** [1 4]

## Abstract

In real-world scenarios, the application of reinforcement learning is significantly challenged by complex non-stationarity. Most existing methods attempt to model changes in the environment explicitly, often requiring impractical prior knowledge of environments. In this paper, we propose a new perspective, positing that non-stationarity can propagate and accumulate through complex causal relationships during state transitions, thereby compounding its sophistication and affecting policy learning. We believe that this challenge can be more effectively addressed by implicitly tracing the causal origin of non-stationarity. To this end, we introduce the **C**ausal-**O**rigin **REP**resentation (**COREP**) algorithm. COREP primarily employs a guided updating mechanism to learn a stable graph representation for the state, termed as causal-origin representation. By leveraging this representation, the learned policy exhibits impressive resilience to non-stationarity. We supplement our approach with a theoretical analysis grounded in the causal interpretation for non-stationary reinforcement learning, advocating for the validity of the causal-origin representation. Experimental results further demonstrate the superior performance of COREP over existing methods in tackling non-stationarity problems. The code is available at https://github.com/PKU-RL/COREP.

## 1. Introduction

Rapid advancements in reinforcement learning (RL) (Kaelbling et al., 1996; Sutton & Barto, 2018) have led to significant performance gains in various domains (Silver et al., 2018; Mirhoseini et al., 2020). However, a common assumption in many RL algorithms is the stationarity of the environment, which can limit the applicability in real-world scenarios characterized by varying dynamics (Padakandla et al., 2020; Padakandla, 2021). Recent efforts in the meta-RL approaches (Finn et al., 2017) have attempted to tackle this by enabling algorithms to adapt to changes (Poiani et al., 2021). However, these methods struggle in the face of more complex and unpredictable environmental dynamics (Sodhani et al., 2022; Feng et al., 2022). Approaches like FN-VAE (Feng et al., 2022) and LILAC (Xie et al., 2020) have made strides towards improving RL algorithms in non-stationary environments by explicitly modeling the change factors of the environment. Nevertheless, they may not comprehensively capture the complexity of real-world non-stationarity. This gap highlights the need for a more robust approach to handle the intricacies of non-stationary environments in RL.

In this paper, we propose a novel setting for efficiently tackling non-stationarity in RL from a new perspective inspired by the causality literature (Zhang et al., 2020; Huang et al., 2020). We contend that minor changes in dynamics can cause significant shifts in observations due to their propagation through intricate causal relationships in the dynamics. Thus, we need to trace the "causal origin" of these changes. However, directly constructing a causal graph that captures such information is a significant challenge due to the inherent instability in non-stationary environments (Strobl, 2019). To address this, we introduce the Causal-Origin Representation (COREP) algorithm. COREP employs a guided update mechanism, which enables the learning of a stable graph representation of state, termed as *causal-origin representation*. This representation aims to capture the underlying causal structure in a way that is resilient to the unpredictable changes characteristic of non-stationary environments, aiding RL algorithms to adapt and learn policies in such settings.

Specifically, we first propose a novel formulation of non-stationarity in RL as the mixture of decomposed sub-environments. In this formulation, we rewrite dynamics functions using masks to represent causal relationships, as-

[1]School of Computer Science, Peking University [2]Center for Statistical Science, Peking University [3]School of Data Science, Fudan University [4]Beijing Academy of Artificial Intelligence. Correspondence to: Zongqing Lu <zongqing.lu@pku.edu.cn>.

*Proceedings of the 41st International Conference on Machine Learning*, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

sumed invariant within each sub-environment. However, identifying these causal relationships without prior knowledge of the sub-environments poses a significant challenge. To overcome this, we propose the environment-shared union graph that captures causal relationship information among state elements. This is achieved by combining Maximal Ancestral Graphs (MAGs) from each sub-environment. We also provide theoretical support for the feasibility of recovering this environment-shared union graph.

To effectively learn the proposed graph representation, we design a dual graph structure comprising a core-graph and a general-graph. The core graph focuses on learning a graph representation that is stable to dynamic changes, guided by a TD (Temporal Difference) error based updating mechanism. While it concentrates on learning the most essential parts of the graph representation, some edge information might be overlooked. To address this, we employ a continuously updating general-graph to compensate for potential information loss and enhance the algorithm's adaptability. By integrating the core-graph and general-graph, we can finally construct the causal-origin representation, providing a comprehensive understanding of the environment's dynamics and significantly mitigating the impact of non-stationarity problems in RL.

Our main contributions can be summarized as follows:

- We provide a causal interpretation for non-stationary RL and propose a novel setting that focuses on the causal relationships within states;
- Based on the proposed formulation and setting, we design a modular algorithm that can be readily integrated into existing RL algorithms;
- We provide a theoretical analysis that offers both inspiration and theoretical support for our algorithm. Experimental results further demonstrate the effectiveness of our algorithm.

## 2. Preliminaries

**Problem Formulation.** Reinforcement learning problems are typically modeled as Markov Decision Processes (MDPs), defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ represents the transition probability, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. We may also use the form of a dynamics function: $\boldsymbol{s}' = f(\boldsymbol{s}, \boldsymbol{a}, \varepsilon)$ when the environment is deterministic. Here, $\varepsilon$ represents random noise. The goal of an agent in RL is to find a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected cumulative discounted reward, defined as the value function $V^{\pi}(\boldsymbol{s}) = \mathbb{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t r_t | \boldsymbol{s}_0 = \boldsymbol{s}]$, where $r_t$ is the reward at time step $t$.

In non-stationary environments, the dynamics of the environment can change over time. Our goal is to learn a policy $\pi$ that can adapt to the non-stationary environment and still achieve high performance. For simplicity of notations, we provide theoretical analysis in the form of the deterministic dynamics function $f$.

**Causal Structure Discovery.** Causal structure discovery usually aims at inferring causation from data, modeled with a directed acyclic graph (DAG) $\mathcal{D} = (V, E)$, where the set of nodes $V$ includes the variables of interest, and the set of directed edges $E$ contains direct causal effects between these variables (Pearl et al., 2000). The causal graph is a practical tool that relates the conditional independence relations in the generating distribution to separation statements in the DAG ($d$-separation) through the Markov property (Lauritzen, 1996). If there exist unobserved confounders in the dynamics, maximal ancestral graphs (MAGs) $\mathcal{M} = (V, D, B)$ are often used to represent observed variables by generalizing DAGs with bidirected edges which depicts the presence of latent confounders (Richardson & Spirtes, 2002). The sets $D, B$ stand for directed and bidirected edges, respectively.

We also provide the definition of the *partial order*, a basic concept necessary for the theoretical analysis in the manuscript. A partial order, $\pi$, on a DAG is defined to represent a relationship between nodes where their order is not strictly defined but still satisfies three properties as follows: (1) Reflexivity: each node is related to itself *i.e.*, $A <_{\pi} A$; (2) Antisymmetry: if $A <_{\pi} B$ and $B <_{\pi} A$, then $A =_{\pi} B$; (3) Transitivity: if $A <_{\pi} B$ and $B <_{\pi} C$, then $A <_{\pi} C$. When this relationship between $A$ and $B$ is not defined, *i.e.*, neither $A <_{\pi} B$ nor $B <_{\pi} A$, we refer to this case as $A \lessgtr B$ with $\pi$, or $A \lessgtr_{\pi} B$.

**Graph Neural Networks (GNNs).** GNNs are a class of neural networks designed for graph-structured data (Scarselli et al., 2008; Zhou et al., 2020). Given a graph $\mathcal{G} = (V, E)$, GNNs aim to learn a vector representation for each node $v \in V$ or the entire graph $\mathcal{G}$, leveraging the information of both graph structure and node features. GNNs generally follow the message passing framework (Balcilar et al., 2021). Layers in GNN can be formulated as $\boldsymbol{H}^{(l)} = \sigma\big(\sum_s \boldsymbol{L}_s \boldsymbol{H}^{(l-1)} \boldsymbol{W}_s^{(l)}\big)$, where $\boldsymbol{H}^{(l)}$ is the node representation outputted by the $l$-th layer, $\boldsymbol{L}_s$ is the $s$-th convolution support which defines how the node features are propagated, $\boldsymbol{W}_s^{(l)}$ is learnable parameters for the $s$-th convolution support in the $l$-th layer, and $\sigma(\cdot)$ is the activation function. Graph Attention Network (GAT) (Veličković et al., 2017) is a special type of GNN following the message passing framework. Instead of handcrafting, the self-attention mechanism is used to compute the support convolutions in each GAT layer, where the adjacency matrix plays the role of a mask matrix for computing the attention.

## 3. Methodology

### 3.1. Motivation and Key Idea

In the COREP algorithm, the primary goal is to tackle non-stationarity problems in RL by learning the underlying graph structure of the environment, which we term *causal-origin representation*. This representation strives to be both causal, meaning it can reflect the cause-effect relationships among state elements, and stable, meaning it is robust to the environment's changes. To achieve this, we design a dual GAT structure consisting of a core-GAT and a general-GAT. The core-GAT focuses on learning the stable part of the environment's causal graph, with its learning guided by a specific updating mechanism. The general-GAT, on the other hand, is continually updated to capture any additional information that the core-GAT might overlook. Together, these two GATs form a comprehensive understanding of the environment.

Specifically, in constructing the causal-origin representation, we first transform the states of the environment into node features and create a weighted adjacency matrix to represent the connections between these features. We then apply a self-attention mechanism, which helps the algorithm focus on the most relevant parts of each node. The update of the core-GAT is guided by a TD error detection mechanism, which helps identify the most significant changes in the environment. To further enhance learning efficiency, we integrate the causal-origin representation into a Variational Autoencoder (VAE) framework (Kingma & Welling, 2013). Additionally, we introduce regularization terms to improve the identifiability of causal relationships and to ensure the structural integrity of the causal-origin representation. The overall framework is shown in Figure 2.

### 3.2. Causal Interpretation of Non-Stationary RL

In this part, we will propose a causal interpretation of non-stationarity in RL, which provides us with inspiration and theoretical support for the algorithm design. First, for the standard dynamics function $\boldsymbol{s}' = f(\boldsymbol{s}, \boldsymbol{a}, \varepsilon)$ in RL, in order to better discuss the relationship between elements, we rewrite it as

$$\begin{aligned} s_i' &= f\left(\boldsymbol{c}_i^{s\to s} \odot \boldsymbol{s}, \boldsymbol{c}_i^{a\to s} \odot \boldsymbol{a}, \varepsilon\right), \\ \boldsymbol{s}' &= \left(s_1', \ldots, s_{d_s}'\right), \end{aligned} \tag{3.1}$$

where $\boldsymbol{s}$ denotes the state with dimension $d_s$, $s_i'$ is the $i$-th element of next state $\boldsymbol{s}'$, $\varepsilon$ is random noise, and $\odot$ denotes the Hadamard product (element-wise product). The masks $\boldsymbol{c}^{\cdot\to s}$ represent the structural dependence among elements in the following way, *e.g.*, the $j$-th element of $\boldsymbol{c}_i^{s\to s} \in \{0,1\}^{d_s}$ equals 1 if and only if $s_j$ causally affects $s_i$. Similarly, $\boldsymbol{c}_i^{a\to s}$ is also defined in the same way. In particular, if only the $i$-th element in $\boldsymbol{c}_i{}^{\cdot\to s}$ equals 1 and all others are 0, it represents that each element only causally affects itself. In this case, Equation (3.1) simplifies to a standard dynamics function.

As mentioned before, when the environment becomes non-stationary, the causal relationships between elements become more intricate. In addition to the relationships between states and actions, there may also be external factors causing environmental changes. These factors can have causal relationships with both state and action, yet are not explicitly considered in Equation (3.1). Intuitively, we assume the presence of hidden states $\boldsymbol{h}$ in the dynamics. Similarly, we define the underlying dynamics function:

$$\begin{aligned} h_i' &= g\left(\boldsymbol{c}_i^{h\to h} \odot \boldsymbol{h}, \boldsymbol{c}_i^{s\to h} \odot \boldsymbol{s}, \boldsymbol{c}_i^{a\to h} \odot \boldsymbol{a}, \varepsilon\right), \\ \boldsymbol{h}' &= \left(h_1', \ldots, h_{d_h}'\right). \end{aligned} \tag{3.2}$$

Considering the influence of the hidden state, the reward function can be similarly rewritten as:

$$r = \mathcal{R}\left(\boldsymbol{c}^{s\to r} \odot \boldsymbol{s}, \boldsymbol{c}^{h\to r} \odot \boldsymbol{h}, \boldsymbol{c}^{a\to r} \odot \boldsymbol{a}, \varepsilon\right). \tag{3.3}$$

The masks can be combined into matrix form as $\boldsymbol{C}^{\cdot\to s} := [\boldsymbol{c}_i^{\cdot\to s}]_{i=1}^{d_s}$, $\boldsymbol{C}^{\cdot\to h} := [\boldsymbol{c}_i^{\cdot\to h}]_{i=1}^{d_h}$. Some previous work assumed that similar masks are invariant over time, and simply encoded them into some change factors (Huang et al., 2022). Instead, we allow such $\boldsymbol{C}^{\cdot\to\cdot}$ to be time-varying, and propose a novel causal interpretation based on an environment-shared union graph representation to capture the transition information in the non-stationary environment.

With the defined underlying dynamics, we can represent the non-stationarity as being governed by specific causal relationships. Specifically, we regard a non-stationary environment as a mixture of various stationary sub-environments. The non-stationarity is thus interpreted as changes over time in the mixture distribution of these sub-environments. *It's important to emphasize that the interpretation of non-stationarity serves primarily as theoretical insight. In practical algorithm designs, we leverage this property in an approximate manner and do not require the environment to be explicitly decomposed into sub-environments.*

When considering a sample from the $k$-th sub-environment, the defined masks $\boldsymbol{C}_{(k)}^{\cdot\to\cdot}$ are used to represent invariant relationships within that sub-environment, meaning the current mask $\boldsymbol{C}_{\text{current}}^{\cdot\to\cdot} = \boldsymbol{C}_{(k)}^{\cdot\to\cdot}$. The relationships among variables such as states, actions, and hidden states, are characterized by DAGs $\mathcal{D}_{(k)} = (V, E_{(k)})$ over the same node set $V = \{\boldsymbol{s}, \boldsymbol{h}, \boldsymbol{a}, \boldsymbol{s}', \boldsymbol{h}', \boldsymbol{a}', r, e\}$. Here, $e$ represents the environment label with an in-degree of 0, and $E_{(k)}$ is the set of directed edges in each DAG. Edges in $E_{(k)}$ are determined by corresponding masks, *i.e.*, an edge from node $v_i$ to $v_j (v_i, v_j \in V)$ exists in $E_{(k)}$ if and only if $\boldsymbol{c}_{(k)}^{v_i\to v_j} = 1$. Additionally, $E_{(k)}$ also includes edges from $e$ to $(\boldsymbol{s}, \boldsymbol{h})$, which are subject to variation across different sub-environments.

In this setting, non-stationarity is reflected by the changes in the underlying graph structure of these DAGs.

To better understand the causal interpretation, we provide an example case involving two sub-environments, as shown in Figure 1. This illustrates how the proposed union graph structure effectively captures the non-stationarity in the environment. For more detailed explanations, please refer to Appendix A. Additionally, a toy example is provided in Appendix B to further explain this concept.

## 3.3. Union Graph of the Causal Structure

In the context of the above causal interpretation, a significant challenge arises when learning the structure of a causal graph without access to the environment label $e$, which can be considered a latent confounder that leads to spurious correlations. In the presence of unobservable nodes, maximal ancestral graph (MAG) is a useful tool to generalize DAGs (Richardson & Spirtes, 2002). For each environment-specific DAG $\mathcal{D}_{(k)}$, we can construct a corresponding MAG $\mathcal{M}_{(k)}$ (Sadeghi, 2013), as outlined in Algorithm A.1. In these MAGs, bidirected edges ($\leftrightarrow$) are used to characterize the change of marginal distribution of $\boldsymbol{s}, \boldsymbol{h}$ over different sub-environments. To model structural relationships in non-stationary RL with a unified approach, we further encode the relations among all actions, states, hidden states, and rewards with an environment-shared union graph $\mathcal{M}_\cup$, as defined by Definition 3.1 below.

**Definition 3.1** (Environment-shared union graph)**.** The environment-shared union graph $\mathcal{M}_\cup := (V, D, B)$ has the set of nodes $V$, the set of directed edges

$$D = \{u \to v : u, v \in V, \exists k \text{ such that } u \to v \text{ in } \mathcal{M}_{(k)}\},$$

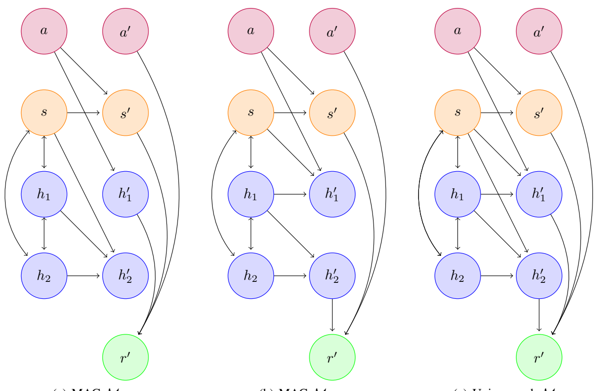
and the set of bidirected edges

$$B = \{u \leftrightarrow v : u, v \in V, \exists k \text{ such that } u \leftrightarrow v \text{ in } \mathcal{M}_{(k)}\}.$$

The above defined union graph $\mathcal{M}_\cup$ contains no cycle because Equation (3.1-3.3) implies that there exists a common topological ordering for $\{\mathcal{D}_{(k)}\}$, see Appendix A for details. Without knowing the label of the $k$-th sub-environment, we cannot generally identify the structure of $\mathcal{M}_{(k)}$ for each $k$ from the observed data. However, we can show that $\mathcal{M}_\cup$ is still a MAG, hence any non-adjacent pair of nodes is $d$-separated given some subset of nodes.

**Proposition 3.2.** *Suppose that the dynamics follows Equation (3.1-3.3), then there exists a partial order $\pi$ on $V$ such that (a) $u$ is an ancestor of $v \Rightarrow u <_\pi v$ in $\mathcal{M}_{(k)}$; and (b) $u \leftrightarrow v \Rightarrow u \nleq_\pi v$ in $\mathcal{M}_{(k)}$. As a consequence, the environment-shared union graph $\mathcal{M}_\cup$ is a MAG.*

We provide the complete proofs and a detailed explanation of conditions in Appendix A. Proposition 3.2 provides theoretical support for recovering the structure of the



(a) MAG $\mathcal{M}_{(1)}$. (b) MAG $\mathcal{M}_{(2)}$. (c) Union graph $\mathcal{M}_\cup$

*Figure 1.* MAG representations for two sub-environments and their union graph. In this example, the union graph is capable of representing all possible kinds of causal relationships within the changing dynamics. More explanations can be found in Appendix A.

environment-shared union graph $\mathcal{M}_\cup$. In the following sections, we will describe how the COREP algorithm learns a policy that is stable under non-stationarity by utilizing the environment-shared representation union graph $\mathcal{M}_\cup$.

## 3.4. Dual Graph Attention Network Structure

In this section, we will focus on the structure design. The detailed update mechanism will be discussed in Section 3.6.

In line with Proposition 3.2, our objective is to efficiently learn the causal-origin representation, which encapsulates the environment-shared union graph $\mathcal{M}_\cup$. To achieve this, we design a dual GAT structure comprising a core-GAT and a general-GAT. The core-GAT is specifically designed to learn a stable graph representation that aligns with the environment-shared union graph. For this purpose, we use TD error as a simple yet effective detector for significant changes in the environment's underlying graph structure. This approach enables selective updates to the core-GAT, ensuring it responds only to significant environmental changes, thereby eliminating the need for explicit recognition of such changes as often required by existing work (Sutton et al., 2007). Additionally, it facilitates the approximate learning of the environment-shared union graph.

While the core-GAT focuses on learning the stable part of the graph representation, some edges may be overlooked or lost in the process. To compensate for this potential loss of information and to enhance the algorithm's adaptation capabilities, we introduce a continuously updating general-GAT. By integrating both the core-GAT and the general-GAT, we can construct the causal-origin representation to provide a comprehensive understanding of the environment's dynamics for the policy and mitigating the impact of non-stationarity in RL.

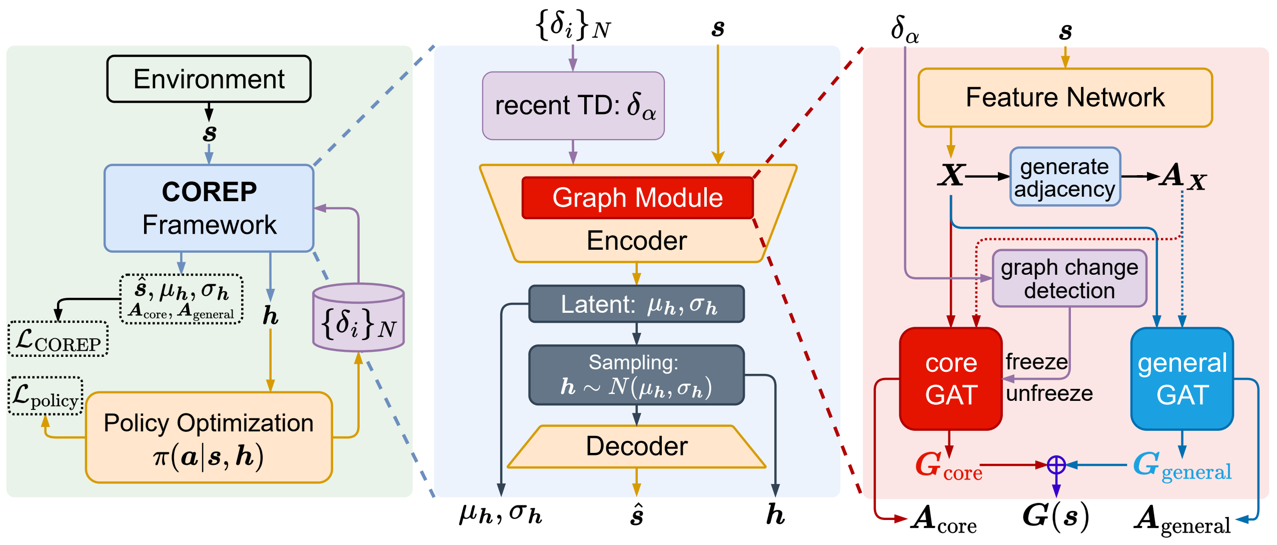


*Figure 2.* Overview of the COREP framework. (1) The left part illustrates that the COREP framework can be seamlessly incorporated into any RL algorithm. It takes the state as input and outputs the causal-origin representation for policy optimization. (2) The middle part shows the VAE structure employed by the COREP framework, which is utilized to enhance the learning efficiency. (3) The right part highlights the key components of COREP. The dual GAT structure is designed in line with the concept of causal-origin representation to retain the essential parts of the graph. The TD error detection can guide the core-GAT to learn the environment-shared union graph based on our theory. The general-GAT is continuously updated to compensate for the potential loss of information.

Specifically, we first transform states into node features using an MLP network $f_{\text{MLP}} : \mathbb{R}^{d_s} \to \mathbb{R}^{N \cdot d_f}$ and reshape the output into node feature matrix $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\} \in \mathbb{R}^{N \times d_f}$, where $N$ is the number of nodes, and $d_f$ is the number of features in each node. We then compute the weighted adjacency matrix which represents the probabilities of edges by using Softmax on the similarity matrix of nodes:

$$\boldsymbol{A}_{\boldsymbol{X}} = \text{Softmax}\left(\boldsymbol{X}\boldsymbol{X}^{\text{T}} \odot (\mathbf{1}_N - \boldsymbol{I}_N)\right), \qquad (3.4)$$

where $\mathbf{1}_N \in \mathbb{R}^{N \times N}$ represents the matrix with all elements equal to 1, $\boldsymbol{I}_N \in \mathbb{R}^{N \times N}$ represents the identity matrix, and $\odot$ denotes the Hadamard product. Multiplying $(\mathbf{1}_N - \boldsymbol{I}_N)$ is for removing the self-loop similarity when computing the weighted adjacency matrix.

Then a learnable weight matrix $\boldsymbol{W} \in \mathbb{R}^{d_f \times d_g}$ is applied to the nodes for transforming $\boldsymbol{X}$ into graph features $\boldsymbol{X}\boldsymbol{W} \in \mathbb{R}^{N \times d_g}$ where $d_g$ denotes the dimension of the graph feature. We then perform the self-attention mechanism on the nodes:

$$\alpha_{ij} = \text{attention}(\boldsymbol{x}_i\boldsymbol{W}, \boldsymbol{x}_j\boldsymbol{W} | \boldsymbol{A}_{\boldsymbol{X}}). \qquad (3.5)$$

The conditioned $\boldsymbol{A}_{\boldsymbol{X}}$ allows us to perform the masked attention, *i.e.*, we only compute $\alpha_{ij}$ for node $j \in \mathbf{N}_i(\boldsymbol{A}_{\boldsymbol{X}})$ where $\mathbf{N}_i(\boldsymbol{A}_{\boldsymbol{X}})$ is the neighbor set of node $i$ computed by the weighted adjacency matrix $\boldsymbol{A}_{\boldsymbol{X}}$. This helps us consider deeper-depth neighbors of each node by combining multiple $\text{attention}(\cdot)$ into a multi-layer network. For the $n$-th graph attention layer, the coefficients computed by the self-attention mechanism can be specifically expressed as:

$$\alpha_{ij} = \frac{\delta_{\mathbf{N}_i(\boldsymbol{A}_{\boldsymbol{X}})}(j) \cdot \exp\left(\sigma\left(\boldsymbol{l}_n \left[\boldsymbol{x}_i\boldsymbol{W} \oplus \boldsymbol{x}_j\boldsymbol{W}\right]^{\text{T}}\right)\right)}{\sum_{k \in \mathbf{N}_i(\boldsymbol{A}_{\boldsymbol{X}})} \exp\left(\sigma\left(\boldsymbol{l}_n \left[\boldsymbol{x}_i\boldsymbol{W} \oplus \boldsymbol{x}_k\boldsymbol{W}\right]^{\text{T}}\right)\right)}, \qquad (3.6)$$

where $\oplus$ is the concatenation operator, $\sigma$ is the activation function, $\boldsymbol{l}_n \in \mathbb{R}^{2d_g}$ is the learnable weight for the $n$-th graph attention layer, and $\delta_{\mathbf{N}_i(\boldsymbol{A}_{\boldsymbol{X}})}(j)$ is the indicator function, *i.e.*, $\delta_{\mathbf{N}_i(\boldsymbol{A}_{\boldsymbol{X}})}(j) = 1$ if $j \in \mathbf{N}_i(\boldsymbol{A}_{\boldsymbol{X}})$ otherwise $0$. Subsequently, the resulting features are concatenated to form the graph node:

$$\boldsymbol{g}_i = \sigma\Big(\sum_{j \in \mathbf{N}_i(\boldsymbol{A}_{\boldsymbol{X}})} \alpha_{ij}\boldsymbol{x}_j\boldsymbol{W}\Big). \qquad (3.7)$$

The respective outputs of the core-GAT and the general-GAT, *i.e.*, $\boldsymbol{G}_{\text{core}} \doteq \{\boldsymbol{g}_1, \boldsymbol{g}_2, \dots, \boldsymbol{g}_N\}^{\text{T}}_{\text{core}}$, $\boldsymbol{G}_{\text{general}} \doteq \{\boldsymbol{g}_1, \boldsymbol{g}_2, \dots, \boldsymbol{g}_N\}^{\text{T}}_{\text{general}}$, are concatenated to form the final causal-origin representation. Specifically, we denote the entire process of obtaining the causal-origin representation from $\boldsymbol{s}$ as a function $\boldsymbol{G} : \mathbb{R}^{d_s} \to \mathbb{R}^{N \times 2d_g}$, such that $\boldsymbol{G}(\boldsymbol{s}) \doteq \boldsymbol{G}_{\text{core}} \oplus \boldsymbol{G}_{\text{general}}$.

We finally feed $\boldsymbol{G}(\boldsymbol{s})$ into a VAE inference process to derive the latent representation $\boldsymbol{h}$. More details about this process are discussed in Section 3.5. The latent $\boldsymbol{h}$ is then provided to the policy $\pi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{h})$ for policy optimization. COREP does not restrict the choice of policy optimization algorithms. In our implementation, we choose the standard PPO algorithm (Schulman et al., 2017) for policy optimization.

## 3.5. Incorporation with VAE

To improve the efficiency of learning the causal-origin representation, we incorporate the causal-origin representation into the Variational AutoEncoder (VAE) framework (Kingma & Welling, 2013). Specifically, we feed the output $\boldsymbol{G}(\boldsymbol{s})$ into the VAE inference process to derive the mean and variance $(\mu_{\boldsymbol{h}}, \sigma_{\boldsymbol{h}})$ of the latent representation $\boldsymbol{h}$. Subsequently, we can sample $\boldsymbol{h} \sim \mathcal{N}(\mu_{\boldsymbol{h}}, \sigma_{\boldsymbol{h}}), \boldsymbol{h} \in \mathbb{R}^{d_h}$. The loss function for VAE is defined as

$$\begin{aligned}
&\mathcal{L}_{\text{VAE}}(\boldsymbol{s};\theta,\phi) \\
&= \mathbb{E}_{q_\phi(\boldsymbol{h}|\boldsymbol{G}(\boldsymbol{s}))}\left[\log p_\theta(\boldsymbol{G}(\boldsymbol{s})|\boldsymbol{h})\right] - \mathrm{KL}\left[q_\phi(\boldsymbol{h}|\boldsymbol{G}(\boldsymbol{s}))||p(\boldsymbol{h})\right] \\
&\approx \mathrm{MSE}(\boldsymbol{s},\hat{\boldsymbol{s}}) - \mathrm{KL}\left[q_\phi\left(\boldsymbol{h}|\boldsymbol{G}(\boldsymbol{s})\right)||\mathcal{N}(0,\boldsymbol{I})\right],
\end{aligned} \tag{3.8}$$

where $p_\theta, q_\phi$ represent the parameterized decoder and encoder respectively, $\mathrm{KL}(\cdot)$ denotes the Kullback-Leibler divergence, and $\mathrm{MSE}(\boldsymbol{s},\hat{\boldsymbol{s}})$ is an estimation of $\mathbb{E}_{q_\phi(\boldsymbol{h}|\boldsymbol{G}(\boldsymbol{s}))}\left[\log p_\theta(\boldsymbol{G}(\boldsymbol{s})|\boldsymbol{h})\right]$ which measures the mean square error between the original state and the reconstructed state with the causal-origin representation. *It is noteworthy that the VAE structure serves solely as a tool to enhance learning efficiency, therefore it is not a strictly necessary component of our method.* The latent $\boldsymbol{h}$ is then provided to the policy $\pi(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{h})$ for policy optimization.

## 3.6. Guided Updating for Core-GAT

As discussed in Section 3.4, to learn the causal-origin representation encapsulating the environment-shared union graph $\mathcal{M}_\cup$, we design a TD error-based detection mechanism to guide the update of core-GAT. Specifically, we store the TD errors of policy optimization into a buffer and compute the mean of recent TD errors, denoted as $\delta_\alpha$. Here, $\alpha$ controls the proportion of recent TD errors for detection. We then check whether $\delta_\alpha$ lies within the confidence interval $(\mu_\delta - \eta\sigma_\delta, \mu_\delta + \eta\sigma_\delta)$, where $\mu_\delta, \sigma_\delta$ are the mean and standard deviation of the TD buffer, and $\eta$ represents the confidence level. If the recent TD error $\delta_\alpha$ lies within this interval, we freeze the weights of the core-GAT and halt its updates; otherwise, we unfreeze its weights and proceed with updating the core-GAT.

We further introduce a regularization that penalizes the difference between the output adjacency matrices of the two GATs to guide the learning of the core-GAT:

$$\mathcal{L}_{\text{guide}} = \|\boldsymbol{A}_{\text{core}} - \boldsymbol{A}_{\text{general}}\|_2. \tag{3.9}$$

To enhance the identifiability, we introduce the regularization for the MAG structure and sparsity:

$$\begin{aligned}
\mathcal{L}_{\text{MAG}} &= \|\boldsymbol{A}_{\text{core}} - \boldsymbol{A}_{\text{core}}^{\mathrm{T}}\|_2 + \|\boldsymbol{A}_{\text{general}} - \boldsymbol{A}_{\text{general}}^{\mathrm{T}}\|_2, \\
\mathcal{L}_{\text{sparsity}} &= \|\boldsymbol{A}_{\text{core}}\|_1 + \|\boldsymbol{A}_{\text{general}}\|_1.
\end{aligned} \tag{3.10}$$

The $\mathcal{L}_{\text{MAG}}$ can penalize asymmetry in the adjacency matrices of both the core and general graphs, ensuring alignment with the design of MAGs which utilize bidirected edges to characterize the change of marginal distribution of $\boldsymbol{s}, \boldsymbol{h}$ across different sub-environments. The $\mathcal{L}_{\text{sparsity}}$ can encourage a sparse representation, which is crucial for maintaining a manageable and interpretable graph structure.

We finally compute the total loss function $\mathcal{L}_{\text{total}}$ as shown in Equation (3.11), combining the policy optimization objective $\mathcal{L}_{\text{policy}}$ with the regularization loss functions:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{policy}} + \lambda_1\mathcal{L}_{\text{guide}} + \lambda_2(\mathcal{L}_{\text{MAG}} + \mathcal{L}_{\text{sparsity}} + \mathcal{L}_{\text{VAE}}), \tag{3.11}$$

where $\mathcal{L}_{\text{VAE}}$ is calculated according to Equation (3.8). It is important to note that, despite the presence of multiple regularization terms, we simplify the objective by aggregating these terms based on their magnitudes, setting only two hyperparameters. Empirical results also demonstrate that COREP does not rely on complex parameter tuning.

By computing and backpropagating the gradient of $\mathcal{L}_{\text{total}}$, along with the policy optimization in an end-to-end manner, COREP prevents the distinct loss functions from leading to irregular causal structures and ensures that the policy learned can effectively tackle non-stationarity problems in RL. The detailed steps of COREP are outlined in Algorithm C.1. Implementation details are shown in Appendix C.

## 4. Experiments

In this section, our objective is to thoroughly evaluate the COREP algorithm by addressing three key questions: (1) How effective is COREP in tackling non-stationarity? (2) What specific contribution does each component of COREP make to its overall performance? (3) Can COREP maintain consistent performance across various degrees and settings of non-stationarity? To answer these questions, we conduct various experiments and provide corresponding analyses. Due to the page limitation, only parts of these experiments are presented in the main manuscript. For complete results, further analysis, and implementation details, please refer to Appendix C and D. To ensure reproducibility, we include our code in the supplementary material.

**Baselines.** We compare COREP with the following baselines: FN-VAE (Feng et al., 2022), VariBAD (Zintgraf et al., 2019), and PPO (Schulman et al., 2017). FN-VAE is the SOTA method for tackling non-stationarity, VariBAD is one of the SOTA algorithms in meta-RL that also has certain capabilities in handling non-stationarity, and PPO is a classical algorithm known for its strong stability. Furthermore, to examine the performance degradation caused by non-stationarity, we include an Oracle that has full access to non-stationarity information.

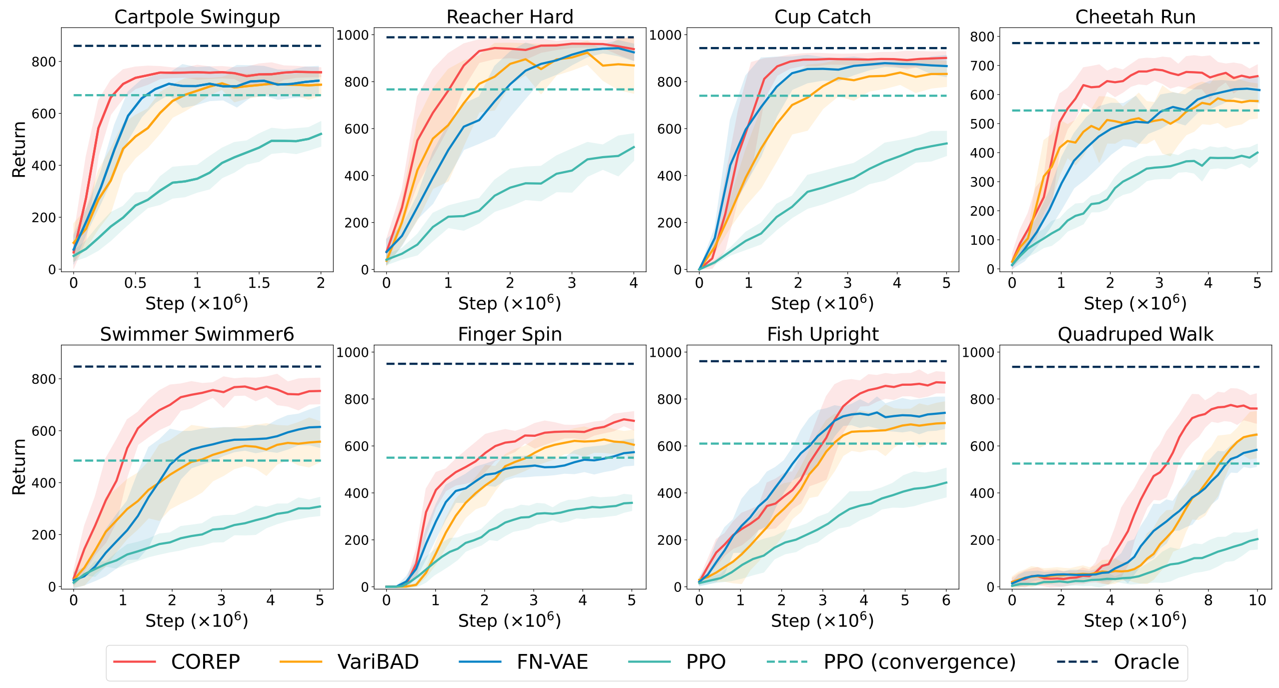


*Figure 3.* Learning curves of our COREP algorithm and other baselines in different tasks. Solid curves indicate the mean of all trials with 5 different seeds. Shaded regions correspond to the standard deviation among trials. Dashed lines represent the asymptotic performance of PPO and Oracle.

**Environment settings.** The experiments are conducted on various environments from the DeepMind Control Suite (Tassa et al., 2018), which is a widely used benchmark for RL algorithms. We modify the environments to introduce non-stationarity, enabling a comprehensive evaluation of COREP. In our settings, similar to prior work FN-VAE, we introduce periodic noises to the environmental dynamics to represent non-stationarity. To support our claim that COREP can handle more complex non-stationarity, we design a more intricate setting, *i.e.*, we randomly sample the coefficients for both within-episode and across-episode non-stationarity at every time step. Specifically, our modification can be expressed as:

$$s' = f(s,a) + f(s,a) \cdot \alpha_d \left[ c_1^t \cos(c_2^t \cdot t) + c_3^i \sin(c_4^i \cdot i) \right]. \tag{4.1}$$

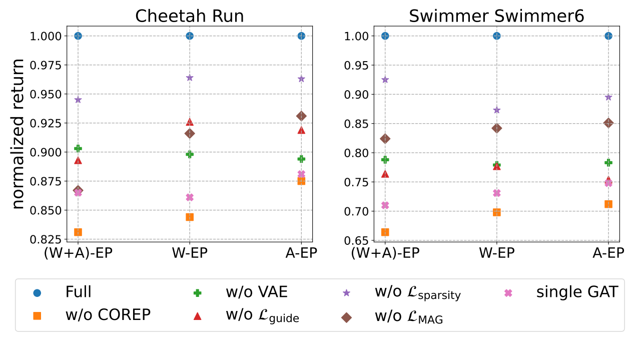
Here, $\alpha_d$ controls the overall degree of non-stationarity, and $c_k^t, c_k^i \sim \mathcal{N}(0.5, 0.5)$ represent the changing coefficients of *within-episode* and *across-episode* non-stationarity, respectively. This design generates various combinations of non-stationarity for each time step and episode, posing more significant challenges to our COREP algorithm and the compared baselines.

**Performance.** As illustrated in Figure 3, COREP outperforms all baselines across various environments, showcasing consistent performances in the face of non-stationarity. Notably, in complex environments such as *Swimmer Swimmer6, Fish Upright, and Quadruped Walk*, COREP not only achieves higher performance but also exhibits smaller variances. This indicates its strong resilience and adaptability to intricate non-stationary environments.

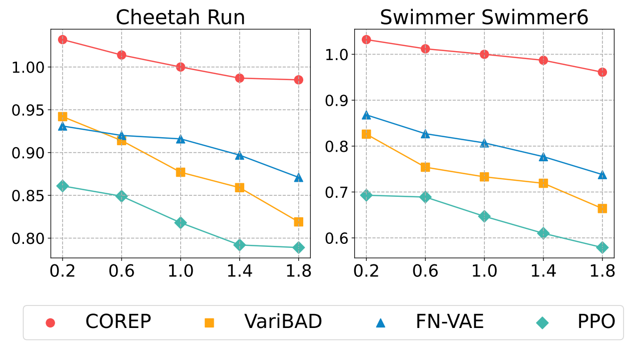
Comparatively, VariBAD demonstrates certain resistance to non-stationarity due to its adaptive capabilities. However, the large variances indicate a lack of stability in non-stationary settings. The FN-VAE method, which explicitly models the change factors, shows competitive performance in simpler environments but struggles to maintain consistency in more complex scenarios, underscoring its limitations in handling more challenging non-stationarity. Furthermore, the narrow performance gap between COREP and Oracle indicates the effectiveness of COREP in reducing the impact of non-stationarity on performance.

**Ablation study.** We conduct ablation studies to analyze the contribution of each component in COREP. To ensure consistency in our conclusion, the experiments are conducted under various non-stationarity settings: *within-episode & across-episode* (labeled as W+A-EP), *within-episode* (W-EP), and *across-episode* (A-EP) non-stationarities.

As depicted in Figure 4(a), removing all COREP-specific designs and retaining only the VAE process (the 'w/o COREP' variant) results in substantial decreases in performance. This highlights the overall effectiveness of our designs.



(a) Ablation study.



(b) Different degrees of non-stationarity.

*Figure 4.* Mean returns of 3 different trials with: (a) different components and non-stationarity settings. Returns are normalized to the full version of COREP in each environment; (b) different degrees of non-stationarity. Returns are normalized to the COREP algorithm with standard degree 1.0.

In the single GAT variant, which maintains the same network structure without introducing a secondary GAT and corresponding update mechanisms, we observe a significant performance gap compared to the full COREP. This result indicates that merely incorporating a graph representation is insufficient for capturing non-stationarity information effectively.

The removal of the TD error detection mechanism (the 'w/o $\mathcal{L}_{\text{guide}}$' variant) lead to considerable performance drops, further substantiating the importance of the guided update mechanism in COREP.

These results demonstrate the effectiveness and necessity of the two key designs in our method (*i.e.*, the dual GAT structure and the guided update mechanism) in tackling non-stationarity. Additionally, the results also show that the regularization terms $\mathcal{L}_{\text{MAG}}$ and $\mathcal{L}_{\text{sparse}}$ play important roles in enhancing COREP's ability to handle non-stationarity. Furthermore, the inclusion of the VAE process in COREP is found to be effective in improving the performance as expected.

**Different degrees of non-stationarity.**

We further investigate the impact of varying degrees of non-stationarity in the environment, as depicted in Figure 4(b). The results suggest that the compared baselines are more affected by the degree of non-stationarity. Contrastively, COREP exhibits consistent performance when encountering different degrees of non-stationarity, further demonstrating our claim that COREP can effectively tackle more complex non-stationarity.

**Visualization.** To visualize the graph structure learned by COREP, we respectively show the weighted adjacency matrix of core-GAT and general-GAT in *Cheetah Run* and *Walker Walk*. Please refer to Section D.6 for more details. It can be seen that core-GAT focuses more on a few core nodes in its learned graph structure, while general-GAT compensates for some overlooked detailed information by core-GAT. The results align well with our claim made in Section 3.1.

**Hyperparameter study.** We conduct additional experiments to study COREP's sensitivity to the hyperparameters $\lambda_1, \lambda_2$ in the objective function (3.11). Results are shown in Appendix D.7, indicating that the performance of COREP is not sensitive to the hyperparameters. Even with extensive adjustments to $\lambda_1, \lambda_2$, COREP consistently surpasses the SOTA baseline. Notably, we observe instances in certain environments with higher performance than default. This suggests that with more precise tuning, there is potential to further enhance COREP's effectiveness. These findings suggest that COREP's superior performance is largely attributed to its designs rather than relying on hyperparameter tuning.

## 5. Related Work

**Non-stationary RL.** Pioneering research in non-stationary RL primarily focused on directly detecting changes that had already occurred (Da Silva et al., 2006), rather than anticipating them. Various methods have been developed to anticipate changes in non-stationary RL settings. For example, Prognosticator (Chandak et al., 2020) tried to maximize future rewards without explicitly modeling non-stationary environments, while MBCD (Alegre et al., 2021) employed change-point detection to determine whether an agent should learn a new policy or reuse existing ones. However, these methods with change-point detection may not work well in complex non-stationary environments and often requires providing priors.

In cases where the evolution of non-stationary environments can be represented as a Semi-Markov chain, Hidden Markov-MDPs or Hierarchical Semi-Markov Decision Processes can be employed to address non-stationarity problems (Choi et al., 1999; Hadoux et al., 2014). Some later work attempts to resist non-stationarity by leveraging the generalization of meta-learning (Finn et al., 2017). For example, Adaption via Meta-learning (Al-Shedivat et al., 2017) integrated continuous adaptation into the learning-to-learn framework to solve the non-stationarity problems. TRIO (Poiani et al., 2021) tracked non-stationarity by inferring the evolution of latent parameters, capturing the temporal change factors during the meta-testing phase. GrBAL (Nagabandi et al., 2018) meta-trained dynamic priors, enabling efficient adaptation to local contexts.

However, these methods require the pre-definition of non-stationary tasks and subsequent meta-training on them. In real-world scenarios, though, we cannot access such information about the non-stationarity. An alternative line of research directly tries to learn latent representations to capture non-stationary components, leveraging latent variable models to directly model change factors in environments or estimating latent vectors describing the non-stationary aspects of dynamics.

Specifically, LILAC (Xie et al., 2020) regarded the change factor as a latent variable and explicitly modeled the latent MDP. FN-VAE (Feng et al., 2022) modeled multiple latent variables of non-stationarities to achieve better performance. However, in real-world scenarios, non-stationarity itself is often more complex. Simply modeling the latent dynamics may not solve such complex scenarios well. Interpreting nonstationarity from a causal perspective is another novel approach.

In addition, some work learns controllers on a collection of pre-defined stationary environments (Provan et al., 2022; Deng et al., 2022; Zhang & Li, 2019), which can get a guaranteed controller for any mixture of these stationary environments, thereby improving performance in more complex environments. Although our method theoretically treats non-stationary environments as mixtures of sub-environments in a similar way, we use a more elegant update mechanism in practical design to ensure that our method is applicable to continuously changing environments.

Some other research (Saengkyongam et al., 2023) seeks to identify an invariant causal structure to mitigate the impact of non-stationarity, presenting similarities to our approach. However, their methodology relies on offline data and has been tested solely in simple contextual Bandits environments. In contrast, our COREP algorithm is capable of online learning and addresses non-stationarity in more complex environments.

**Causal structure learning.** Various approaches for learning causal structure from observed data have been proposed, see (Vowels et al., 2022) for a review. These approaches mainly fall into two broad categories: constraint-based methods and score-based methods. The constraint-based methods check the existence of edges by performing conditional independence tests between each pair of variables, *e.g.* PC (Spirtes et al., 2000), IC (Pearl et al., 2000), and FCI (Spirtes et al., 1995; Zhang, 2008). In contrast, score-based methods generally view causal structure learning as a combinatorial optimization problem, and measure the goodness of fit of graphs over the data with a score, then optimize such score to find an optimal graph or equivalent classes (Chickering, 2002; Koivisto & Sood, 2004; Silander & Myllymäki, 2006; Cussens et al., 2017; Huang et al., 2018).

Recently, some gradient-based methods that transform the discrete search into a continuous optimization by relaxing the space over DAGs have been proposed. These methods allow for applying continuous optimizations such as gradient descent to causal structure learning. For example, NOTEARS (Zheng et al., 2018) reformulated the structure learning problem as a continuous optimization problem, and ensured acyclicity with a weighted adjacency matrix. DAG-GNN (Yu et al., 2019) proposed a generative model parameterized by a GNN and applied a variant of the structural constraint to learn the DAG. Some researchers (Saeed et al., 2020) considered the distribution arising from a mixture of causal DAGs, used MAGs to represent DAGs with unobserved nodes, and showed the identifiability of the union of component MAGs.

## 6. Conclusions, Limitations and Future Work

In this work, we first offer a novel interpretation of non-stationarity in RL, characterizing it through the union of MAGs. This new perspective has inspired us to design the COREP algorithm, which features a dual GAT structure and an update mechanism guided by TD-error detection. Focusing on the causal relationships whthin the dynamics, COREP learns a causal-origin representation that remains stable amidst changes in the environment, effectively addressing non-stationarity problems in RL. Our theoretical analysis offers both inspiration and foundational support for COREP. Furthermore, experimental results from various non-stationary environments demonstrate the efficacy of our algorithm.

However, the COREP algorithm does face certain limitations, particularly scalability issues in high-dimensional state spaces due to the computationally intensive nature of its graph-based representation. In our future work, we aim to overcome these challenges by integrating the causal-origin representation with other types of latent variable models, such as normalizing flows and probabilistic graphical models. This is expected to enhance both the scalability and the performance, making COREP more applicable in real-world scenarios.

## Acknowledgements

This work was supported in part by NSF China under grant 62250068. The authors would like to thank the anonymous reviewers for their valuable feedback and suggestions.

## Impact Statement

This research in machine learning, especially through the COREP algorithm, offers potential benefits for enhancing the decision-making abilities of agents in complex and unpredictable environments, such as robot control and autonomous driving. However, it is crucial to be aware of ethical implications, including data privacy, algorithmic bias, and the security of these systems. Future research should focus on developing transparent and safe algorithms for real-world scenarios.

## References

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

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

Balcilar, M., Renton, G., Héroux, P., Gaüzère, B., Adam, S., and Honeine, P. Analyzing the expressive power of graph neural networks in a spectral perspective. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

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

Chickering, D. M. Optimal structure identification with greedy search. *Journal of machine learning research*, 3 (Nov):507–554, 2002.

Choi, S., Yeung, D.-Y., and Zhang, N. An environment model for nonstationary reinforcement learning. *Advances in neural information processing systems*, 12, 1999.

Cussens, J., Haws, D., and Studenỳ, M. Polyhedral aspects of score equivalence in bayesian network structure learning. *Mathematical Programming*, 164:285–324, 2017.

Da Silva, B. C., Basso, E. W., Bazzan, A. L., and Engel, P. M. Dealing with non-stationary environments using context detection. In *Proceedings of the 23rd international conference on Machine learning*, pp. 217–224, 2006.

Deng, X., Zhang, Y., and Qi, H. Towards optimal hvac control in non-stationary building environments combining active change detection and deep reinforcement learning. *Building and environment*, 211:108680, 2022.

Feng, F., Huang, B., Zhang, K., and Magliacane, S. Factored adaptation for non-stationary reinforcement learning. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Hadoux, E., Beynier, A., and Weng, P. Solving hidden-semi-markov-mode markov decision problems. In *Scalable Uncertainty Management: 8th International Conference, SUM 2014, Oxford, UK, September 15-17, 2014. Proceedings 8*, pp. 176–189. Springer, 2014.

Huang, B., Zhang, K., Lin, Y., Schölkopf, B., and Glymour, C. Generalized score functions for causal discovery. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1551–1560, 2018.

Huang, B., Zhang, K., Zhang, J., Ramsey, J., Sanchez-Romero, R., Glymour, C., and Schölkopf, B. Causal discovery from heterogeneous/nonstationary data. *The Journal of Machine Learning Research*, 21(1):3482–3534, 2020.

Huang, B., Feng, F., Lu, C., Magliacane, S., and Zhang, K. AdaRL: What, where, and how to adapt in transfer reinforcement learning. In *International Conference on Learning Representations*, 2022.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Koivisto, M. and Sood, K. Exact bayesian structure discovery in bayesian networks. *The Journal of Machine Learning Research*, 5:549–573, 2004.

Lauritzen, S. L. *Graphical models*, volume 17. Clarendon Press, 1996.

Mirhoseini, A., Goldie, A., Yazgan, M., Jiang, J. W., Songhori, E., Wang, S., Lee, Y.-J., Johnson, E., Pathak, O., Nazi, A., et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2020.

Nagabandi, A., Clavera, I., Liu, S., Fearing, R. S., Abbeel, P., Levine, S., and Finn, C. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

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

Padakandla, S., KJ, P., and Bhatnagar, S. Reinforcement learning algorithm for non-stationary environments. *Applied Intelligence*, 50:3590–3606, 2020.

Pearl, J. et al. Models, reasoning and inference. *Cambridge, UK: Cambridge University Press*, 19(2), 2000.

Poiani, R., Tirinzoni, A., and Restelli, M. Meta-reinforcement learning by tracking task non-stationarity. In Zhou, Z.-H. (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 2899–2905. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/399. Main Track.

Provan, G., Quinones-Grueiro, M., and Sohége, Y. Towards real-time robust adaptive control for non-stationary environments. *IFAC-PapersOnLine*, 55(6):73–78, 2022.

Richardson, T. and Spirtes, P. Ancestral graph markov models. *The Annals of Statistics*, 30(4):962–1030, 2002.

Sadeghi, K. Stable mixed graphs. *Bernoulli*, 19(5B):2330–2358, 2013.

Saeed, B., Panigrahi, S., and Uhler, C. Causal structure discovery from distributions arising from mixtures of dags. In *International Conference on Machine Learning*, pp. 8336–8345. PMLR, 2020.

Saengkyongam, S., Thams, N., Peters, J., and Pfister, N. Invariant policy learning: A causal perspective. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

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

Silander, T. and Myllymäki, P. A simple approach for finding the globally optimal bayesian network structure. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, pp. 445–452, 2006.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Sodhani, S., Meier, F., Pineau, J., and Zhang, A. Block contextual mdps for continual learning. In *Learning for Dynamics and Control Conference*, pp. 608–623. PMLR, 2022.

Spirtes, P., Meek, C., and Richardson, T. Causal inference in the presence of latent variables and selection bias. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pp. 499–506, 1995.

Spirtes, P., Glymour, C. N., and Scheines, R. *Causation, prediction, and search*. MIT press, 2000.

Strobl, E. V. Improved causal discovery from longitudinal data using a mixture of dags. In *The 2019 ACM SIGKDD Workshop on Causal Discovery*, pp. 100–133. PMLR, 2019.

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

Sutton, R. S., Koop, A., and Silver, D. On the role of tracking in stationary environments. In *Proceedings of the 24th international conference on Machine learning*, pp. 871–878, 2007.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Vowels, M. J., Camgoz, N. C., and Bowden, R. D'ya like dags? a survey on structure learning and causal discovery. *ACM Computing Surveys*, 55(4):1–36, 2022.

Xie, A., Harrison, J., and Finn, C. Deep reinforcement learning amidst lifelong non-stationarity. *arXiv preprint arXiv:2006.10701*, 2020.

Yu, Y., Chen, J., Gao, T., and Yu, M. Dag-gnn: Dag structure learning with graph neural networks. In *International Conference on Machine Learning*, pp. 7154–7163. PMLR, 2019.

Zhang, J. On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artificial Intelligence*, 172(16-17): 1873–1896, 2008.

Zhang, K., Gong, M., Stojanov, P., Huang, B., Liu, Q., and Glymour, C. Domain adaptation as a problem of inference on graphical models. *Advances in neural information processing systems*, 33:4965–4976, 2020.

Zhang, W. and Li, J. Stable weighted multiple model adaptive control of continuous-time plant with large parameter uncertainties. *IEEE Access*, 7:144125–144133, 2019.

Zheng, X., Aragam, B., Ravikumar, P. K., and Xing, E. P. Dags with no tears: Continuous optimization for structure learning. *Advances in neural information processing systems*, 31, 2018.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

Zintgraf, L., Shiarlis, K., Igl, M., Schulze, S., Gal, Y., Hofmann, K., and Whiteson, S. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. *arXiv preprint arXiv:1910.08348*, 2019.

## A. Causality Background and Proofs

We first review the definition of the Markov condition, the faithfulness assumption and some graphical concepts shown in the condition of Theorem 3.2. We use $\mathrm{pa}_{\mathcal{D}}(v)$, $\mathrm{ch}_{\mathcal{D}}(v)$, and $\mathrm{an}_{\mathcal{D}}(v)$ to denote the parents, children and ancestor of node $v$, respectively; for the detailed definitions, see *e.g.* (Lauritzen, 1996).

**Definition A.1** (Global Markov Condition (Pearl et al., 2000))**.** A distribution $P$ over $V$ satisfies the global Markov condition on graph $\mathcal{D}$ if for any partition $(X, Y, Z)$ such that $X$ is d-separated from $Y$ given $Z$, then $X$ and $Y$ are conditionally independent given $Z$.

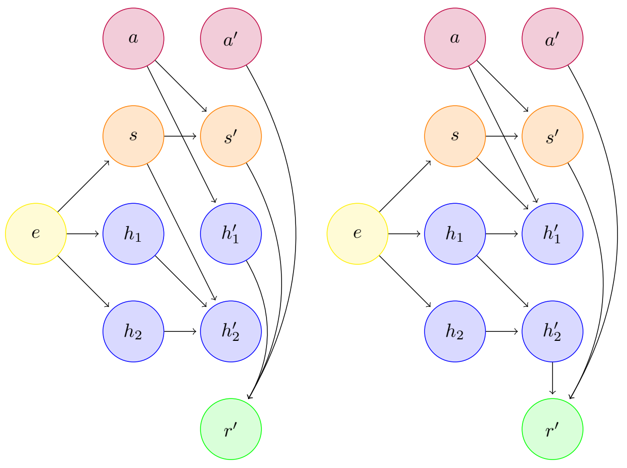
**Definition A.2** (Faithfulness (Pearl et al., 2000))**.** There are no independencies between variables that are not entailed by the Markov Condition.

Under the above assumptions, we can tell the conditional independences using the d-separation criterion from a given DAG $\mathcal{D}$ (Pearl et al., 2000). Similarly, the MAGs are ancestral graphs where any non-adjacent pair of nodes is d-separated (Richardson & Spirtes, 2002). The following algorithm shows how to construct a MAG from DAG (Saeed et al., 2020):

**Algorithm A.1** Construction of the maximal ancestral graph

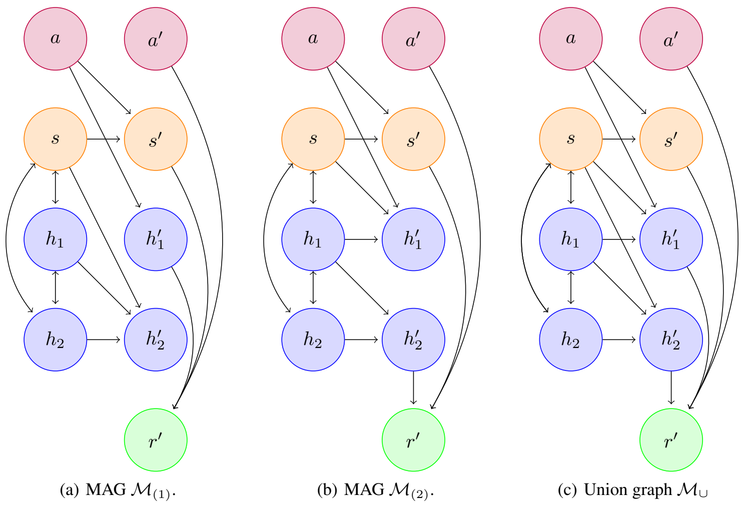
```
 1: Input: DAG $\mathcal{D} = (V, E)$
 2: Initialize $D = \emptyset, B = \emptyset$
 3: for $u, v \in \mathrm{ch}_{\mathcal{D}}(y)$ do
 4:     add $u \leftrightarrow v$ to $B$.
 5: end for
 6: for $t, u, v$ such that $(t \to u) \in E$ and $(u \leftrightarrow v) \in B$ do
 7:     if $u \in \mathrm{an}_{\mathcal{D}}(v)$ then
 8:         add $t \to v$ to $D$
 9:     end if
10: end for
11: for $u, v$ such that $(u \leftrightarrow v) \in B$ do
12:     if $u \in \mathrm{an}_{\mathcal{D}}(v)$ then
13:         remove $u \leftrightarrow v$ from $B$ and add $u \to v$ to $D$
14:     end if
15: end for
```

To illustrate the above algorithm, we provide two figures. Figure 5 shows the underlying causal DAGs for the two environments, and Figure 6 depicts the output of Algorithm A.1 as well as the corresponding environment-shared union graph.



(a) DAG $\mathcal{D}_{(1)}$. (b) DAG $\mathcal{D}_{(2)}$.

*Figure 5.* DAG representations for two sub-environments.



$a$
$a'$
$s$
$s'$
$h_1$
$h'_1$
$h_2$
$h'_2$
$r'$
(a) MAG $\mathcal{M}_{(1)}$.
(b) MAG $\mathcal{M}_{(2)}$.
(c) Union graph $\mathcal{M}_{\cup}$

*Figure 6.* MAG representations for two sub-environments and their union graph.

In this paper, we characterize the nonstationarity as a mixture of stationary distributions. Formally, we take the following definition.

**Definition A.3** (Mixture of stationary distributions)**.** The marginal distribution of $\{\boldsymbol{s}, \boldsymbol{h}, \boldsymbol{a}, \boldsymbol{s}', \boldsymbol{h}', \boldsymbol{a}'\}$ is a mixture of stationary distributions across environments, i.e.,

$$P(\boldsymbol{s}, \boldsymbol{h}, \boldsymbol{a}, \boldsymbol{s}', \boldsymbol{h}', \boldsymbol{a}') = \sum_k \pi_k P(\boldsymbol{s}, \boldsymbol{h}, \boldsymbol{a}, \boldsymbol{s}', \boldsymbol{h}', \boldsymbol{a}' \mid e = k),$$

where $\pi_k$ denotes the probability that the sample is from the $k$-th environment varying over time, and $P(\boldsymbol{s}, \boldsymbol{h}, \boldsymbol{a}, \boldsymbol{s}', \boldsymbol{h}', \boldsymbol{a}' \mid e = k)$ is invariant over time.

*Proof of Proposition 3.2.* The outline of the proof are as follows. We first construct a strict partial order $\pi$ on $V$. Then, we induce the MAGs $\mathcal{M}_{(1)}, \ldots, \mathcal{M}_{(k)}$ from the DAGs $\mathcal{D}_{(1)}, \ldots, \mathcal{D}_{(k)}$ by applying the rules defined in Algorithm A.1. We show the constructed partial order $\pi$ is *compatible*, that $\forall k$, it holds that (a) $u \in \mathrm{an}(v) \Rightarrow u <_\pi v$ in $\mathcal{M}^{(k)}$; and (b) $u \leftrightarrow v \Rightarrow u \not\lessgtr_\pi v$ in $\mathcal{M}^{(k)}$. Finally we leverage the existing results in (Saeed et al., 2020) to conclude that $\mathcal{M}_\cup$ is a MAG.

We define a relation $\pi$ on $V$ as following: for any variable $u \in \{\boldsymbol{s}, \boldsymbol{h}, \boldsymbol{a}\}$ and any variable $v \in \{\boldsymbol{s}', \boldsymbol{h}', \boldsymbol{a}'\}$, we have (i) $u <_\pi v$; (ii) $v <_\pi r'$. To show the above defined $\pi$ is a strict partial order, we first notice that $\pi$ is irreflexive, because $u \not<_\pi u, v \not<_\pi v$ and $r' \not<_\pi r'$. The transitivity and asymmetry also hold by definition of $\pi$. Therefore, $\pi$ is a partial order on $V$.

The Algorithm A.1 constructs an MAG from DAG with three steps. The first step is to add bidirected edges among the nodes in $\mathrm{ch}(e)$. Different values of $e$ leads different marginal distribution of $\boldsymbol{s}, \boldsymbol{h}$, hence $\mathrm{ch}(y) \subseteq \{\boldsymbol{s}, \boldsymbol{h}\}$. Therefore, the bidirected edges are added with both nodes belonging to $\{\boldsymbol{s}, \boldsymbol{h}\}$. For the second step, there is no such node $t$, with $(t \to u) \in E$ and $(u \leftrightarrow v) \in B$, because the nodes in $\{\boldsymbol{s}, \boldsymbol{h}\}$ have no ancestor other than itself. So the second step adds the directed edges when $u = v$. The third step in our case is redundant. Equation (3.1) shows that there is no instantaneous causal effects in the system, so there is no $u, v$ such that $(u \leftrightarrow v) \in B$ while $u \in \mathrm{an}_{\mathcal{D}}(v)$. From all above, if the input of Algorithm A.1 is $\mathcal{D}_{(k)}$, then it outputs a MAG $\mathcal{M}_{(k)} = (V, D_{(k)}, B_{(k)})$ with the set of nodes $V$, the set directed edges $D$ equals to the set of directed edges $E_{(k)}$ after removing the node $e$, and the set of bidirected edges $B_{(k)}$ consists edges among nodes in $\{\boldsymbol{s}, \boldsymbol{h}\}$.

Then, we check the condition (a) and (b) to show $\mathcal{M}_{(1)}, \ldots, \mathcal{M}_{(k)}$ are compatible with the above defined $\pi$. For (a), if $u$ is the ancestor node of $v$, then the structure of $\mathcal{M}_{(k)}$ implies that either $u \in \{\boldsymbol{s}, \boldsymbol{h}\}$ and $v \in \{\boldsymbol{s}', \boldsymbol{h}'\}$ are nodes in $\{\boldsymbol{s}', \boldsymbol{h}'\}$, or $u$ is a node from $\{\boldsymbol{s}', \boldsymbol{h}'\}$ and $v = r'$. For (b), if $u \leftrightarrow v$, then $u, v$ are nodes in $\{\boldsymbol{s}, \boldsymbol{h}\}$, hence $u \not\lessgtr_\pi v$ in $\mathcal{M}^{(k)}$. These means that $\pi$ is a common strict partial order on $V$ for all MAGs. In this setup, we can leverage existing results from Lemma 4.3 (Saeed et al., 2020) to show that the environment-shared union graph $\mathcal{M}_\cup$ is also a maximal ancestral graph.

□

## B. Toy Example under the Causal Interpretation

To better understand the causal interpretation for non-stationary RL, let's consider a simple toy example. Given a stationary environment with a state space represented as $(s_1, s_2)$. In this example, to maintain simplicity, we focus only on the state's mask, omitting the action mask and noise term. We define the original dynamics function as $f(\boldsymbol{c}_s \odot \boldsymbol{s}, a) = \boldsymbol{c}_s \odot \boldsymbol{s} + a$. For the toy environment, we consider a basic causal model wherein $s'_i$ is only influenced by $s_i$.

Consequently, the original mask is

$$\boldsymbol{c}_s = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{B.1}$$

Given this, we can derive that

$$\begin{aligned} s'_1 &= s_1 + a \\ s'_2 &= s_2 + a. \end{aligned} \tag{B.2}$$

We denote the non-stationarity in our experiments (Eq 4.1) simplistically as $\boldsymbol{s}' = f(\boldsymbol{s}, a)[1 + n(t)]$, where $n(t)$ represents the introduced non-stationarity, which makes the dynamics becoming time-varying. In this scenario, the non-stationary environment's dynamics function becomes

$$\begin{aligned} s'_1 &= s_1 + a + (s_1 + a)n(t) \\ s'_2 &= s_2 + a + (s_2 + a)n(t). \end{aligned} \tag{B.3}$$

It is obvious that the dynamics introduces a time-varying term. In this context, we can define $h_i \doteq (s_i + a)n(t)$, leading to $s'_i = s_i + h_i + a$. This allows us to deduce the dynamics of $\boldsymbol{h}$ as

$$\begin{aligned} h'_i &= (s'_i + a) \cdot n(t+1) \\ &= (s_i + h_i + 2a) \cdot n(t+1) \\ &\doteq g_i(\boldsymbol{c}^t_s \odot \boldsymbol{s}, \boldsymbol{c}^t_h \odot \boldsymbol{h}, a), \end{aligned} \tag{B.4}$$

where $\boldsymbol{c}^t_s, \boldsymbol{c}^t_h$ symbolize the time-varying masks caused by non-stationarity $n(t)$.

More specifically, we derive

$$\boldsymbol{c}^t_s = \boldsymbol{c}^t_h = \begin{pmatrix} n(t+1) & 0 \\ 0 & n(t+1) \end{pmatrix} \tag{B.5}$$

The masks represent the causal effects between variables after non-stationary changes occur in this example. The values represent the degree of causal effects, which can be treated as edge weights after normalization. Based on our theory, COREP's goal is to learn the union graph shared by these graph structures and edge weights during the change process. That is, through a common graph with edge weights, it includes all possible non-stationary changes. Therefore, the information of non-stationarity in the final learned graph is reflected both in the topology of the union graph and in the edge weights representing probabilities.

By introducing the time-varying masks and $\boldsymbol{h}$, we can make the dynamics function remains stationary, transferring non-stationarity to the causal model. Thus, we have provided a walk-through under the simple toy example.

In fact, as illustrated in Figures 5 and 6, there are more intricate causal relationships in complex environments. As depicted above, $\boldsymbol{h}$ can encapsulate not only the inherent environmental information but also the complex causal relationships with non-stationarity. Our proposed union MAG (Proposition 3.2) and the correspondingly designed dual-GAT architecture aim to learn such intricate causal models, enabling generic RL algorithms to handle non-stationarity under this causal representation.

## C. Implementation and Training Details

### C.1. Pseudo code for COREP

In Algorithm C.1, we summarize the steps of COREP. For more specific details, please refer to the code provided in our supplementary material.

**Algorithm C.1** **C**ausal-**O**rigin **REP**resentation (**COREP**)

```
1: **Init:** env; VAE parameters $\theta, \phi$; policy parameters: $\psi$; replay buffer $\mathcal{B}$; TD buffer $\mathcal{B}_\delta$.
2: **for** $i = 0, 1, \ldots$ **do**
3:     Collect trajectory $\tau_i$ with $\pi_\psi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{h})$.
4:     Update replay buffer $\mathcal{B}[i] \leftarrow \tau_i$.
5:     **for** $j = 0, 1, \ldots$ **do**
6:         Sample a batch of episodes $E_j$ from $\mathcal{B}$ and TD errors $\{\delta_k\}$ from $\mathcal{B}_\delta$.
7:         Transform states into $\boldsymbol{X}$ through MLPs.
8:         Compute $\boldsymbol{A}_{\boldsymbol{X}} = \mathrm{Softmax}\left(\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}} \odot (\mathbf{1}_N - \boldsymbol{I}_N)\right)$.
9:         Compute $\delta_\alpha = \left(\sum_{|\mathcal{B}_\delta| - \alpha|\mathcal{B}_\delta| < k < |\mathcal{B}_\delta|} \delta_k\right) / \alpha|\mathcal{B}_\delta|$.
10:        **if** $\delta_\alpha \notin (\mu_\delta - \eta\sigma_\delta, \mu_\delta + \eta\sigma_\delta)$ **then**
11:            unfreeze weights of core-GAT.
12:        **else**
13:            freeze weights of core-GAT.
14:        **end if**
15:        Get graph representation $\boldsymbol{G}_{\text{core}}, \boldsymbol{G}_{\text{general}}$ from core-GAT and general-GAT.
16:        Compute $\mathcal{L}_{\text{guide}}, \mathcal{L}_{\text{MAG}}, \mathcal{L}_{\text{sparsity}}$ according to Equation (3.9, 3.10).
17:        Input $\boldsymbol{G}(\boldsymbol{s}) = \boldsymbol{G}_{\text{core}} \oplus \boldsymbol{G}_{\text{general}}$ into VAE encoder $q_\phi$ and infer $\mu_{\boldsymbol{h}}, \sigma_{\boldsymbol{h}}$.
18:        Sample $\boldsymbol{h} \sim \mathcal{N}(\mu_{\boldsymbol{h}}, \sigma_{\boldsymbol{h}})$
19:        Decode $\hat{\boldsymbol{s}}$ from $\boldsymbol{h}$ using decoder $p_\theta$, then compute $\mathcal{L}_{\text{VAE}}$ according to Equation (3.8).
20:        Do policy optimization for $\pi_\psi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{h})$, then compute $\mathcal{L}_{\text{policy}}$ and TD error $\delta$.
21:        Compute $\mathcal{L}_{\text{total}}$ according to Equation (3.11) and use it for gradient-updating $\theta, \phi, \psi$.
22:        Push $\delta$ into TD buffer $\mathcal{B}_\delta$.
23:    **end for**
24: **end for**
```

## C.2. Hyperparameters

We list the hyperparameters for MLP, GAT, and VAE structures in Table C.1, and the hyperparameters for policy optimization and training in Table C.2.

*Table C.1.* Hyperparameters for the structure of MLP, GAT, and VAE.

| **Hyperparameter** | **Value** |
|---|---|
| MLP activation | ReLU |
| MLP hidden dim | 512 |
| MLP learning rate | 1e-3 |
| GAT activation | ELU |
| GAT hidden dim | 32 (Cartpole Swingup, Reacher Easy/Hard, Cup Catch, Cheetah Run)<br>64 (Otherwise) |
| GAT node numbers | 4 (Cartpole Swingup, Reacher Easy/Hard, Cup Catch)<br>8 (Cheetah Run, Hopper Stand)<br>16 (Otherwise) |
| node feature dim | 16 (Cartpole Swingup, Reacher Easy/Hard, Cup Catch, Cheetah Run)<br>32 (Otherwise) |
| GAT head numbers | 2 (Quadruped Walk, Fish Upright, Walker Walk, Swimmer Swimmer6/15)<br>1 (Otherwise) |
| VAE encoder hidden dim | 128 |
| VAE decoder hidden dim | 64 |
| latent representation dim | 4 (Cartpole Swingup, Reacher Easy/Hard, Cup Catch)<br>8 (Cheetah Run, Hopper Stand)<br>16 (Otherwise) |

*Table C.2.* Hyperparameters for policy optimization and training.

| **Hyperparameter** | **Value** |
|---|---|
| Policy hidden dim | 256 (Swimmer Swimmer6/15, Walker Walk, Fish Upright, Quadruped Walk)<br>128 (Otherwise) |
| Policy learning rate | 7e-4 |
| $\lambda_1$ (for $\mathcal{L}_{\text{guide}}$) | 0.1 |
| $\lambda_2$ (for $\mathcal{L}_{\text{MAG}}/\mathcal{L}_{\text{sparsity}}/\mathcal{L}_{\text{VAE}}$) | 1e-3 |
| PPO update epoch | 16 |
| PPO $\gamma$ | 0.97 |
| PPO $\varepsilon$ clip | 0.1 |
| TD buffer size | 2000 |
| Confidence level $\eta$ | 1.96 |

## D. Full Experiment Details

### D.1. Details about Environment Settings.

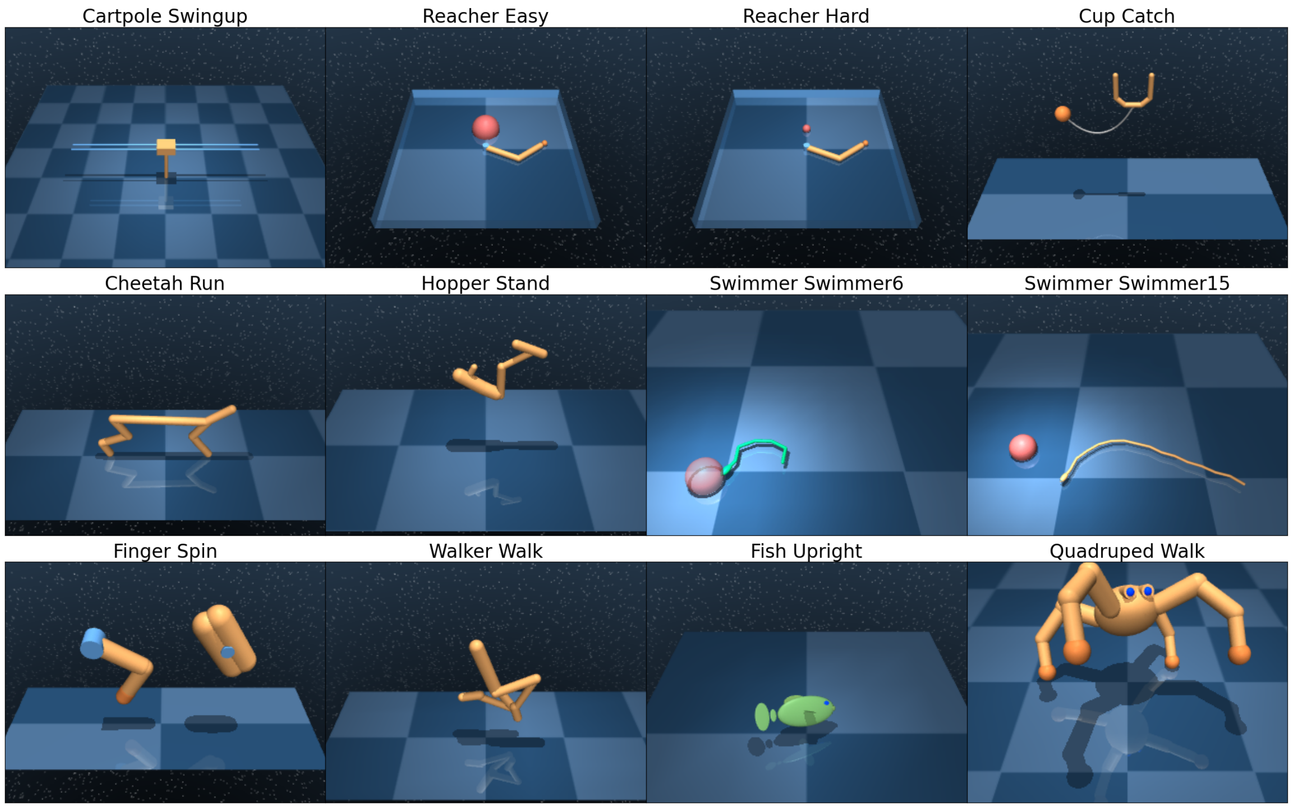


*Figure 7.* The environment we use in our experiment. We add non-stationary noise to the observations of these environments according to Equation (4.1).

Figure 7 shows the environments we use in the experiment. We add non-stationary noise to the observations of these environments according to Equation (4.1). These environments vary in terms of complexity, from low-dimensional problems like “Reacher Easy” to high-dimensional ones like “Quadruped Walk”. All these tasks require the agent to understand and control its physical embodiment in order to achieve the desired goals. The specific descriptions of these environments and goals are as follows.

**Cartpole Swingup.** The cart can move along a one-dimensional track. The pole is attached to the cart with a joint allowing it to rotate freely. The initial state has the pole hanging down, and the goal is to apply forces to the cart such that the pole swings up and is balanced upright. Actions typically involve applying a horizontal force to the cart.

**Reacher Easy.** The agent is a two-joint robotic arm. The arm must move in a two-dimensional plane to touch a target position. The arm’s state includes its joint angles and velocities. The action is the torque applied to each of the joints. The target’s position is fixed in this version.

**Reacher Hard.** The task is the same as ”Reacher Easy,” but the target position is randomly placed in each episode, making the task more difficult as the agent has to learn to reach various positions.

**Cup Catch.** The agent is a robotic arm holding a cup, and there’s a ball attached to the cup with a string. The arm needs to move in a way to swing the ball and catch it in the cup. The arm’s state includes the position and velocity of the arm joints and the position and velocity of the ball. The actions are the torques applied at the arm’s joints.

**Cheetah Run.** The agent is a model of a cheetah-like robot with 9 DoF(Degrees of Freedom): the agent can flex and extend its ”spine,” and each leg has two joints for flexing and extending. The agent’s state includes the joint angles and velocities, and the actions are the torques applied to each of the joints. The goal is to move forward as fast as possible.

**Hopper Stand.** The agent is a one-legged robot, and its goal is to balance upright from a resting position. The agent's state includes the angle and angular velocity of the torso, as well as the joint angles and velocities. The actions are the torques applied to the joints.

**Swimmer Swimmer6.** The agent is a snake-like robot swimming in a two-dimensional plane. The robot has 6 joints, and the goal is to swim forward as fast as possible. The agent's state includes the joint angles and velocities, and the actions are the torques applied to the joints.

**Swimmer Swimmer15.** This is a more complex version of the Swimmer environment, with the agent being a 15-joint snake-like robot. Like the simpler version, the goal is to swim forward as fast as possible.

**Finger Spin.** The agent is a robot with two fingers, and there's a freely spinning object. The goal is to keep the object spinning and balanced on the fingertips. The state includes the positions and velocities of the fingers and the object, and the actions are the forces applied by the fingers.

**Walker Walk.** The agent is a bipedal robot, and the goal is to walk forward as fast as possible. The agent's state includes the angle and angular velocity of the torso, and the joint angles and velocities. The actions are the torques applied to the joints.

**Fish Upright.** The agent is a fish-like robot swimming in a three-dimensional fluid. The goal is to swim forward while maintaining an upright orientation. The agent's state includes the orientation and velocity of the fish, and the actions are the torques and forces applied to move the fish.

**Quadruped Walk.** The agent is a quadrupedal (four-legged) robot. Like the bipedal walker, the goal is to walk forward as fast as possible. The agent's state includes the angle and angular velocity of the torso, and the joint angles and velocities. The actions are the torques applied to the joints.

To ensure consistency in our conclusion, the experiments are conducted under various non-stationarity settings, which include 'within-episode & across-episode', 'within-episode', and 'across-episode' non-stationarities. These settings are respectively denoted as (W+A)-EP, W-EP, and A-EP. Specifically, these non-stationarities can be expressed as

$$s' = f(s,a) + f(s,a) \cdot \alpha_d \left[ c_1^t \cos(c_2^t \cdot t) + c_3^i \sin(c_4^i \cdot i) \right] \tag{D.1}$$

$$s' = f(s,a) + f(s,a) \cdot \alpha_d \left[ c_1^t \cos(c_2^t \cdot t) \right] \tag{D.2}$$

$$s' = f(s,a) + f(s,a) \cdot \alpha_d \left[ c_3^i \sin(c_4^i \cdot i) \right] \tag{D.3}$$

We experimented only under (W+A)-EP when looking at the performance of the algorithm, while in a more detailed ablation study we experimented with all three different setings.

### D.2. Settings of Baselines

For VariBAD, we meta-train the models (5000 batch size, 2 epochs for all experiments) and show the learning curves of meta-testing. The tasks parameters for meta-training are uniformly sampled from a Gaussian distribution $\mathcal{N}(0,1)$.

For all approaches, we use the same backbone algorithm for policy optimization, *i.e.*, PPO with the same hyperparameters, as shown in Table C.2.

## D.3. Full Results of Performance

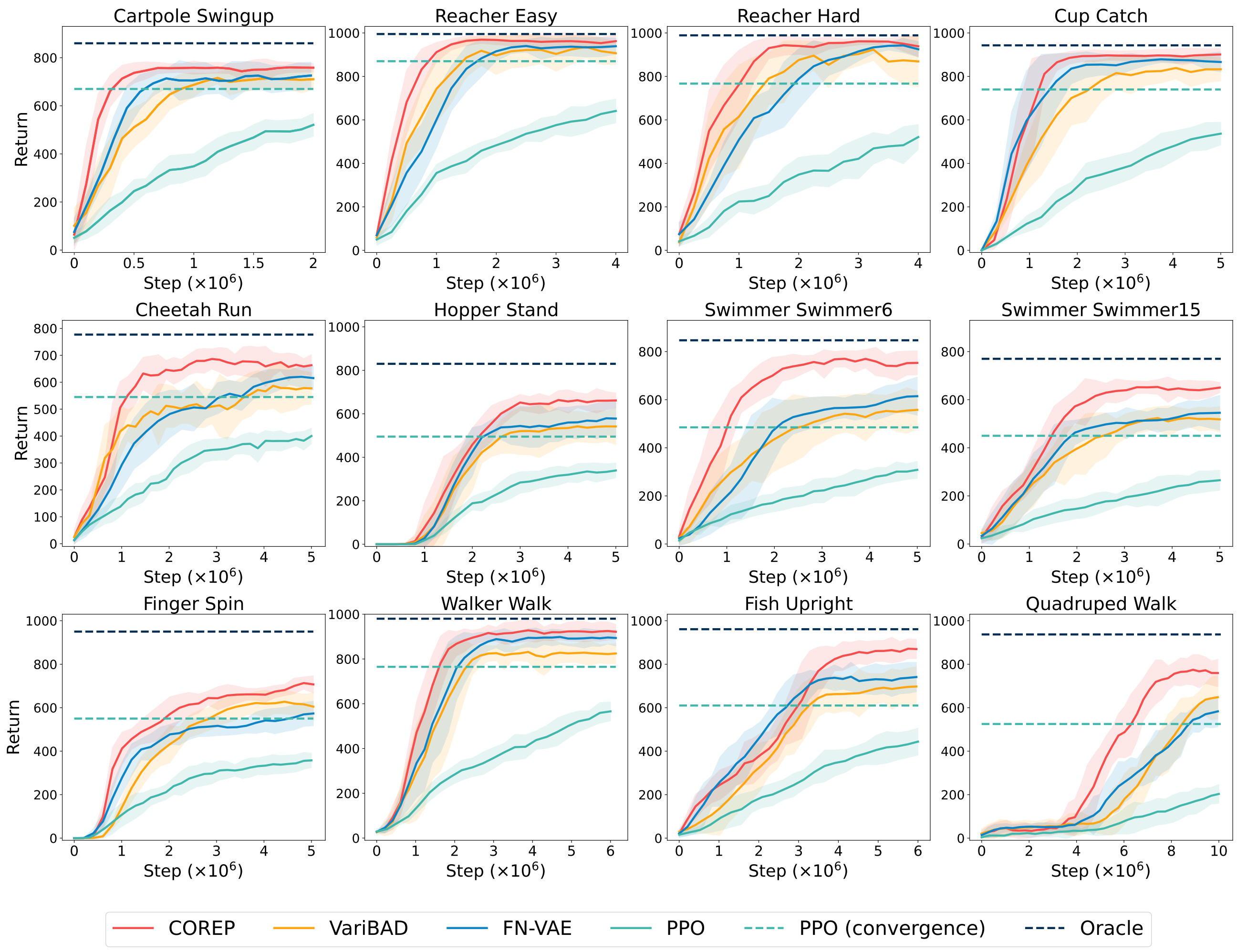


*Figure 8.* Learning curves of COREP and baselines in different environments. Solid curves indicate the mean of all trials with 5 different seeds. Shaded regions correspond to standard deviation among trials. The dashed lines represent the asymptotic performance of PPO and Oracle.

Figure 8 shows the full learning curves. We add non-stationary noise as Equation (D.1) to all environments. According to the results, COREP consistently performs well in environments of different complexities, proving the effectiveness of the algorithm. Especially in *Hopper Stand, Swimmer Swimmer6, Swimmer Swimmer15, Finger Spin, Fish Upright, and Quadruped Walk*, COREP demonstrates a larger performance gap, highlighting its superiority over baselines.

FN-VAE has the ability to approach our COREP in some simple environments (*Cartpole Swingup, Reacher Easy, Reacher Hard, and Cup Catch*), but still exhibits significant variance, reflecting its instability, especially in more complex environments where it performs even worse than VariBAD (*Finger Spin, Quadruped Walk*). VariBAD shows a large performance gap and variance in all environments, indicating poor stability to non-stationarity. PPO's performance is consistently the worst across all environments due to the lack of any optimization for non-stationarity.

## D.4. Full Results of Ablation Study

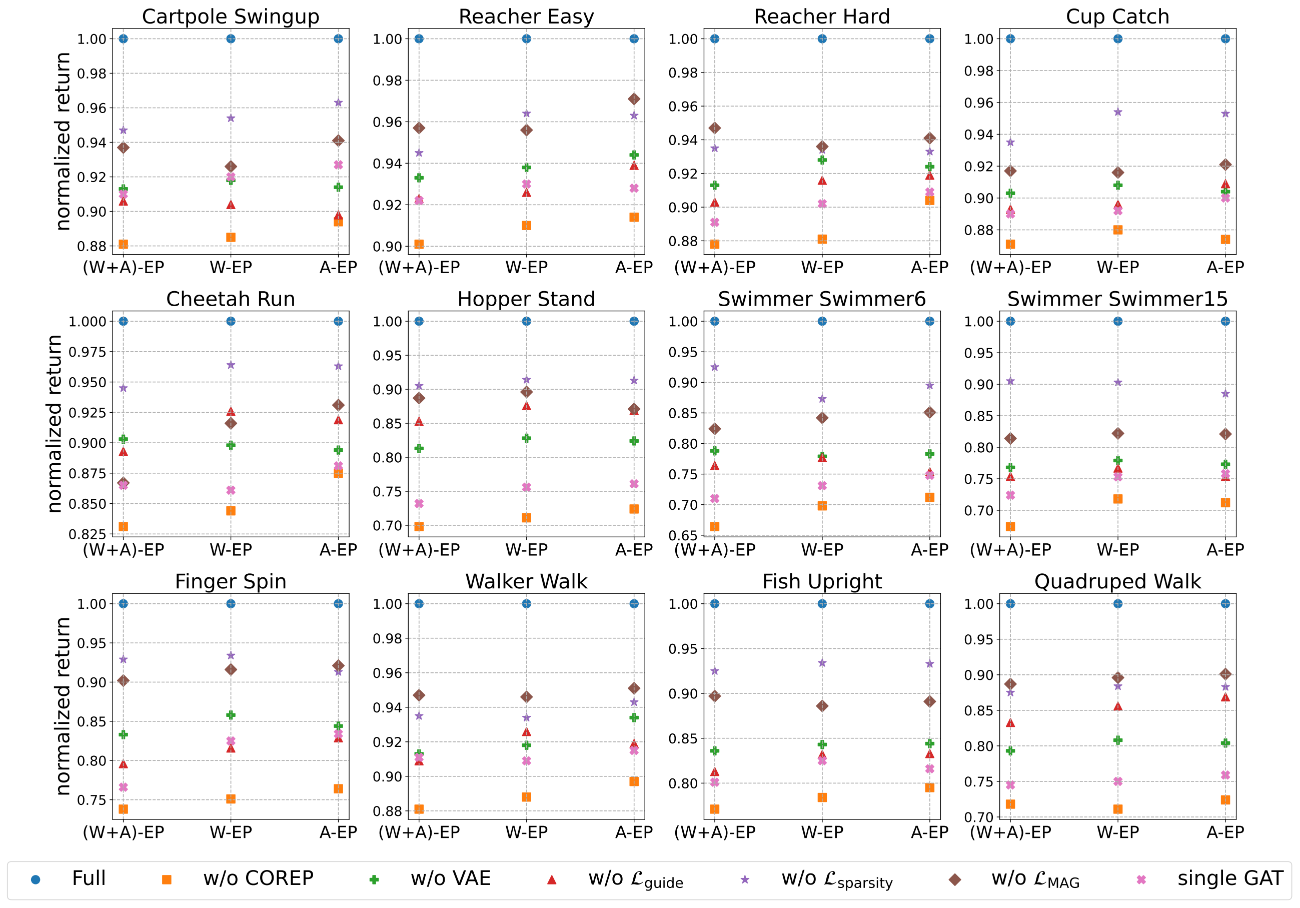


*Figure 9.* Final mean returns of 3 different trials on all environments with different components and non-stationarity settings. Returns are normalized to the full version of COREP in each environment.

Figure 9 shows the performance after removing different components in COREP. All (W+A)-EP, W-EP, and A-EP non-stationary noises, *i.e*. Equation (D.1, D.2, D.3), are separately added to the environments. Each point of Figure 9 represents the normalized return, which is used to observe the contribution of removed components to the overall algorithm. Specifically,

- ‘w/o COREP’ remove all COREP-specific designs and retaining only the VAE process;
- ‘w/o VAE’ is a version without the VAE process;
- ‘w/o $\mathcal{L}_{\text{guide}}$’ removes the guided update mechanism containing TD detection;
- ‘w/o $\mathcal{L}_{\text{sparsity}}$’ removes the corresponding loss $\mathcal{L}_{\text{sparsity}}$ in Equation (3.11);
- ‘w/o $\mathcal{L}_{\text{MAG}}$’ removes the corresponding loss $\mathcal{L}_{\text{MAG}}$ in Equation (3.11);
- ‘single GAT’ maintains the same network structure without introducing a secondary GAT and corresponding update mechanism.

## D.5. Full Results on Non-stationarity Degrees

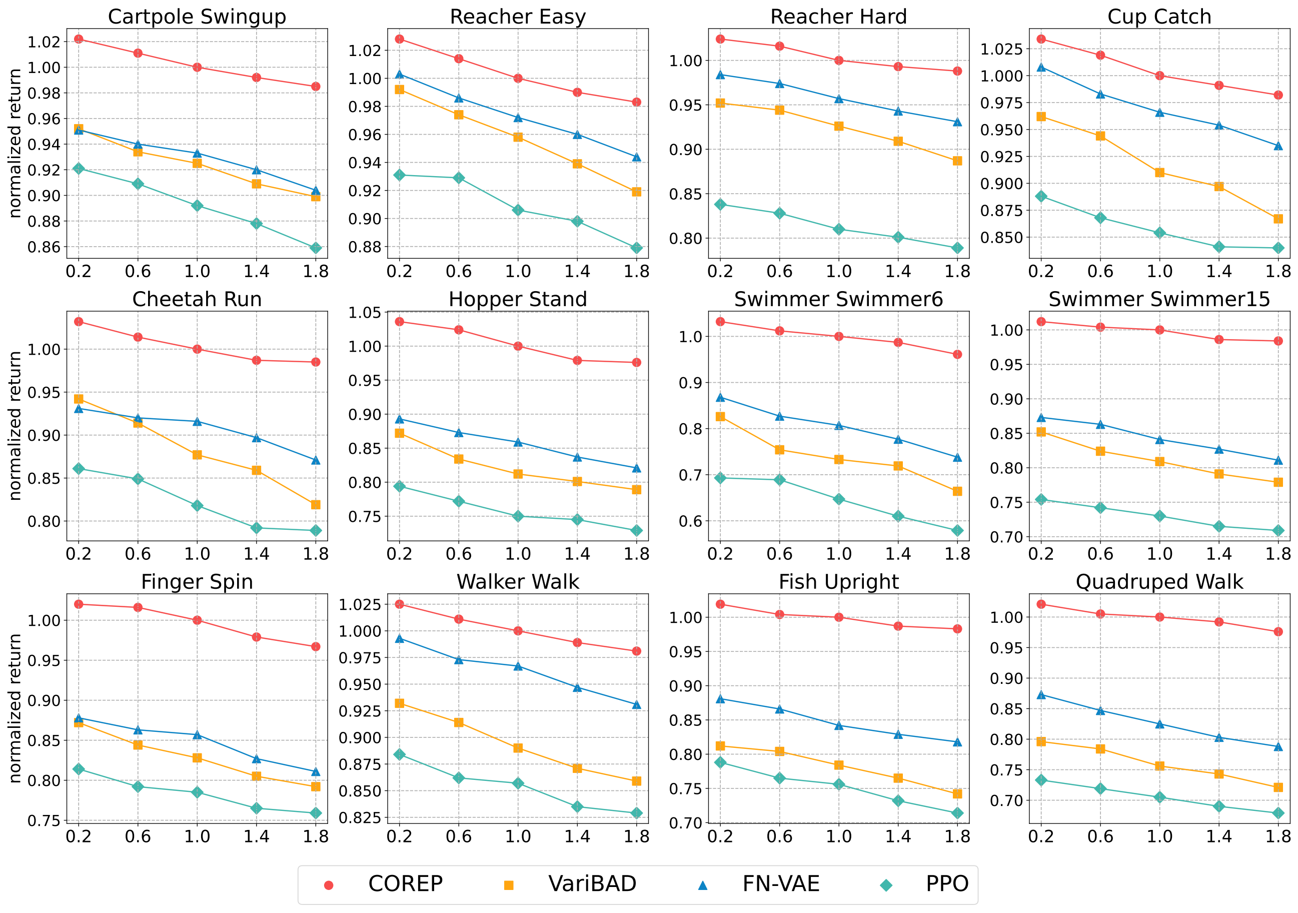


*Figure 10.* Final mean returns of 3 different trials on Cheetah Run and Swimmer Swimmer6 environments with different degrees of non-stationarity. Returns are normalized to the COREP algorithm with standard degree 1.0.

To analyze the impact of varying degrees of non-stationarity in the environment, we change the values of $\alpha_d$ in Equation (D.1). As depicted in Figure 10, the results suggest that the performance of the compared baselines is more affected by the degree of non-stationarity. Conversely, COREP exhibits consistent performance when encountering different degrees of non-stationarity, further affirming our claim that COREP can effectively tackle more complex non-stationarity.

## D.6. Visualization of Learned Graph

To visualize the graph learned by the COREP algorithm, we respectively show the weighted adjacency matrices of core-GAT and general-GAT after 5M steps in the *Cartpole Swingup*, *Reacher Hard*, and *Cup Catch* environment in Figure 11, 12, 13. It is noteworthy that the number of graph nodes is set to be the same as the observation dimension of each environment to bring better empirical insights into how the dual graph actually functions. For further information about the actual meaning of each dimension in different environments, please refer to the DeepMind Control Suite technical report (Tassa et al., 2018).

In these heatmaps, each value on a grid represents the weight of an edge from a node on the y-axis to a node on the x-axis. A higher value indicates a greater causal influence.

Based on the results, it can be seen that core-GAT indeed focuses more on a few core nodes in its learned graph structure, while general-GAT compensates for some overlooked detailed information by core-GAT. The results align well with our claim made in the manuscript.

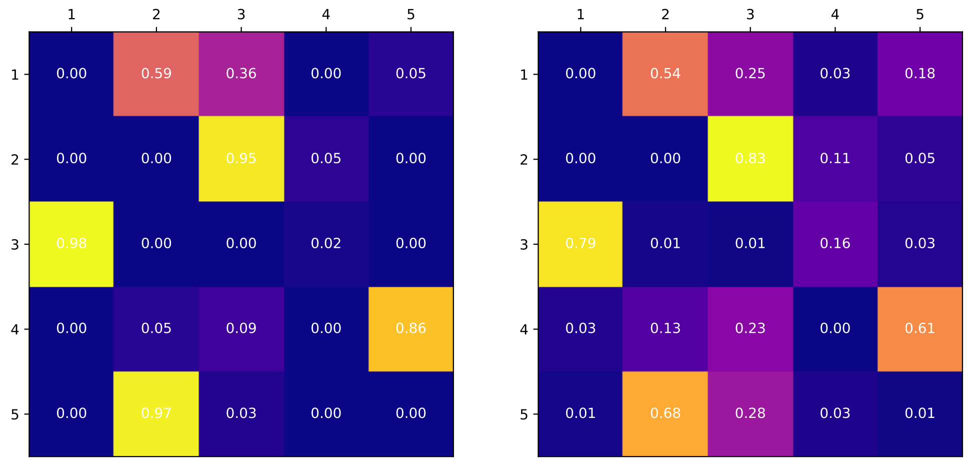


*Figure 11.* Weighted adjacency matrix of core-GAT (*left*) and general-GAT (*right*) in Cartpole Swingup after 5M steps.

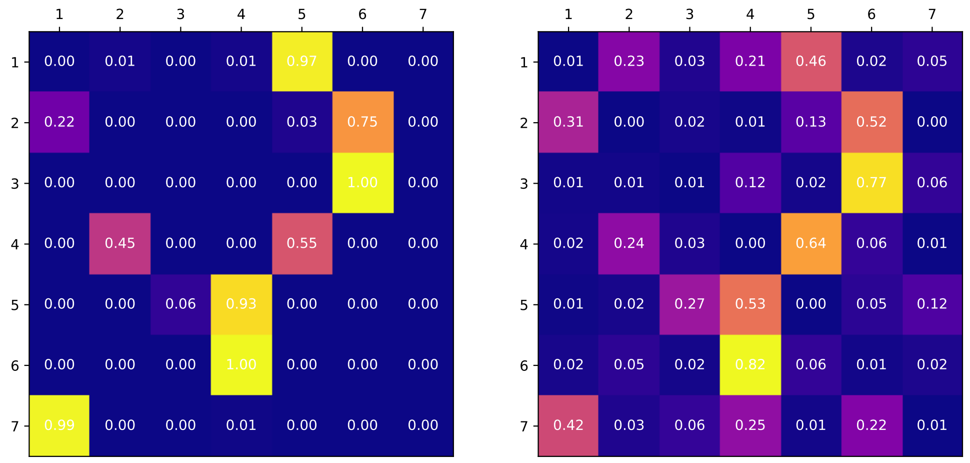


*Figure 12.* Weighted adjacency matrix of core-GAT (*left*) and general-GAT (*right*) in Reacher Hard after 5M steps.

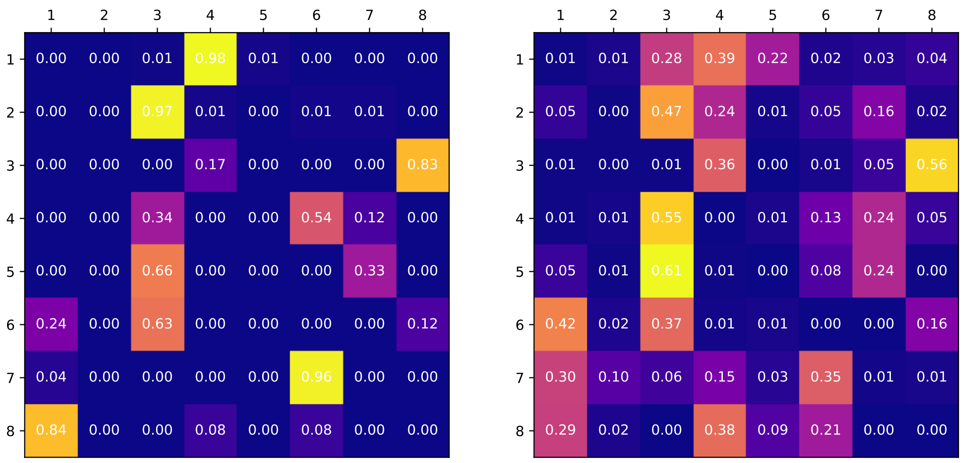


*Figure 13.* Weighted adjacency matrix of core-GAT (*left*) and general-GAT (*right*) in Cup Catch after 5M steps.

## D.7. Study on Tunning Parameters of Loss Terms

We conduct additional experiments to study COREP's sensitivity to the hyperparameters $\lambda_1, \lambda_2$ in the objective function (3.11). Results are shown in Table D.1 and Table D.2.

The results indicate that the performance is not sensitive to the values of these hyperparameters. Even with extensive adjustments to these parameters, COREP consistently surpasses the performance of SOTA baseline. Notably, we observe instances in certain environments where adjustments lead to even higher performance. This suggests that with more precise tuning, there is potential to further enhance COREP's effectiveness.

These findings are significant as they demonstrate that COREP's superior performance is largely attributed to its designs in structure and process, rather than relying on hyperparameter tuning. The low sensitivity to hyperparameter values also implies ease of use and adaptability in diverse settings.

*Table D.1.* Sensitivity results of parameter $\lambda_1$. The results indicate that the performance is not sensitive to $\lambda_1$, even with significant adjustments to its value, COREP still outperforms FN-VAE.

| $\lambda_1$ | Cartpole Swingup | Reacher Easy | Reacher Hard | Cup Catch | Cheetah Run | Hopper Stand |
|---|---|---|---|---|---|---|
| 0.5 | $732.3 \pm 30.6$ | $947.6 \pm 24.6$ | $938.4 \pm 29.2$ | $868.5 \pm 24.7$ | $634.8 \pm 41.6$ | $651.4 \pm 29.3$ |
| 0.1 (original) | $\mathbf{743.4 \pm 21.2}$ | $\mathbf{964.6 \pm 17.3}$ | $947.2 \pm 23.1$ | $\mathbf{877.5 \pm 19.2}$ | $\mathbf{651.1 \pm 44.3}$ | $645.5 \pm 25.8$ |
| 0.05 | $732.8 \pm 26.6$ | $939.3 \pm 16.4$ | $\mathbf{949.4 \pm 25.7}$ | $870.1 \pm 19.8$ | $642.7 \pm 52.8$ | $\mathbf{656.7 \pm 32.3}$ |
| 0.01 | $722.3 \pm 28.4$ | $945.2 \pm 23.3$ | $934.3 \pm 19.5$ | $864.6 \pm 25.5$ | $629.4 \pm 57.2$ | $641.3 \pm 33.8$ |
| 0.001 | $717.5 \pm 31.9$ | $928.8 \pm 25.8$ | $936.9 \pm 25.7$ | $858.3 \pm 18.6$ | $622.8 \pm 49.2$ | $627.5 \pm 27.5$ |
| FN-VAE | $710.3 \pm 64.5$ | $913.3 \pm 38.7$ | $928.1 \pm 21.9$ | $851.3 \pm 31.6$ | $606.5 \pm 75.3$ | $580.9 \pm 47.3$ |

*Table D.2.* Sensitivity analysis of parameters $\lambda_2$. The results indicate that the performance is not sensitive to $\lambda_2$, even with significant adjustments to its value, COREP still outperforms FN-VAE.

| $\lambda_2$ | Cartpole Swingup | Reacher Easy | Reacher Hard | Cup Catch | Cheetah Run | Hopper Stand |
|---|---|---|---|---|---|---|
| 0.01 | $728.7 \pm 36.1$ | $936.3 \pm 23.7$ | $931.5 \pm 24.4$ | $864.9 \pm 25.8$ | $632.2 \pm 37.5$ | $623.5 \pm 32.1$ |
| 0.005 | $\mathbf{745.6 \pm 34.8}$ | $955.3 \pm 25.3$ | $\mathbf{953.6 \pm 22.1}$ | $872.5 \pm 21.5$ | $644.5 \pm 36.1$ | $638.1 \pm 24.6$ |
| 0.001 (original) | $743.4 \pm 21.2$ | $\mathbf{964.6 \pm 17.3}$ | $947.2 \pm 23.1$ | $\mathbf{877.5 \pm 19.2}$ | $\mathbf{651.1 \pm 44.3}$ | $\mathbf{645.5 \pm 25.8}$ |
| 0.0005 | $718.6 \pm 31.5$ | $946.6 \pm 19.6$ | $945.5 \pm 19.4$ | $856.3 \pm 18.7$ | $621.6 \pm 48.4$ | $641.1 \pm 29.6$ |
| 0.0001 | $721.1 \pm 35.9$ | $949.5 \pm 27.8$ | $940.6 \pm 19.7$ | $859.2 \pm 26.4$ | $618.1 \pm 55.7$ | $619.5 \pm 31.7$ |
| FN-VAE | $710.3 \pm 64.5$ | $913.3 \pm 38.7$ | $928.1 \pm 21.9$ | $851.3 \pm 31.6$ | $606.5 \pm 75.3$ | $580.9 \pm 47.3$ |

## E. Compute Resource Details

We list the hardware resources used in Table E.1, and list the training time required for a single trial in each environment in Table E.2.

*Table E.1.* Computational resources for our experiments.

| **CPU** | **GPU** | **RAM** |
|---|---|---|
| Intel I9-12900K@3.2GHz (24 Cores) | Nvidia RTX 3090 (24GB) $\times$ 2 | 256GB |

*Table E.2.* Computing time of each single trial in different environments.

| **Environment** | **Training Time** |
|---|---|
| Cartpole Swingup | 12 hours |
| Reacher Easy | 16 hours |
| Reacher Hard | 20 hours |
| Cup Catch | 16 hours |
| Cheetah Run | 24 hours |
| Hopper Stand | 28 hours |
| Swimmer Swimmer6 | 28 hours |
| Swimmer Swimmer15 | 36 hours |
| Finger Spin | 28 hours |
| Walker Walk | 28 hours |
| Fish Upright | 30 hours |
| Quadruped Walk | 42 hours |

## F. Licenses

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

- Numpy (BSD-3-Clause license)
- PyTorch (BSD-3-Clause license)
- PyTorch Geometric (MIT license)
- DeepMind Control (Apache-2.0 license)
- OpenAI Gym (MIT License)