UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Deep Reinforcement Learning in Non-stationary Environments

by Zihe Liu

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy in Computer Science

under the supervision of Prof. Jie Lu,
A/Prof. Guangquan Zhang and Dr. Junyu Xuan

University of Technology Sydney
Faculty of Engineering and Information Technology

October 2024

AUTHOR'S DECLARATION

I, *Zihe Liu*, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the *Australian Artificial Intelligence Institute*, *Faculty of Engineering and Information Technology* at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

SIGNATURE: Production Note:
Signature removed prior to publication.

Zihe Liu

DATE: Tuesday 8th October, 2024

PLACE: Sydney, Australia

Abstract

Deep Reinforcement Learning has demonstrated superior performance in various domains, such as recommender systems, health operations and autonomous driving. Most traditional deep reinforcement learning can be characterized as the search for a policy that obtains the highest cumulative reward in an unknown but stationary environment with fixed state transitions and reward functions. However, this assumption does not always hold in many practical scenarios; environments are non-stationary and have abrupt and unpredictable change points in many cases. For example, when a well-trained deep reinforcement learning policy is applied to an outdoor robot that may encounter different terrains and enter caves without lighting, the previous optimal policy may make mistakes or even fail. In these practical environments, mistakes will be made repeatedly if the algorithm does not identify the change and actively adapt to it. To address this problem, several methods are proposed in this thesis to focus on deep reinforcement learning in challenging environments with time-varying non-stationarity.

We first formalize the problem of deep reinforcement learning in non-stationary environments with unknown change points. Then, we investigate both model-free and model-based schemes, proposing several solutions to solve the defined problem. We have developed four adaptive algorithms that incorporate change detection mechanisms in the two representative deep reinforcement learning frameworks, proving superior performance over existing methods.

First, a formal model-free reinforcement learning framework that spans the entire learning pipeline and clearly identifies the similarities and differences between the proposed methods and existing schemes is proposed. Under this framework, three novel methods with environment change detection and rapid adaptation are proposed. To detect environmental changes, we consider variations in the joint distribution of states and actions, changes in policy behavior due to abrupt environmental shifts, and alterations in Bayesian uncertainty during the training process. The distinct methods for utilizing prior knowledge, drawing from aspects of gradients, policy behavior, and Gaussian Process posteriors, focus on preserving knowledge beneficial to the current setting. Each method is designed to adjust policy adaptation based on information from change detection, considering proper response to different change extents. The flexible adaptation mechanism ensures optimal performance in the current environment. Second, we investigate the model-based framework targets in high-dimensional environments. We propose a method for learning change dynamics within a latent space aimed at environments with high-dimensional inputs such as image data. Our method identifies change points in the non-stationary environment in the latent space, enabling online detection and adaptation.

Overall, this thesis presents novel methodologies for change point detection and online adaptation for deep reinforcement learning in non-stationary environments, focusing on the robust ability to ensure adequate performance in the face of sequential change in many realistic environments. The empirical results demonstrate the adequate performance of our methods over other baselines, offering practical solutions for real-world applications where the assumption of stationarity does not hold and ensuring continued optimal performance in the face of non-stationary environments.

Dedication

To myself and my parents . . .

Acknowledgments

It is a life-changing journey to complete my Ph.D. study at the Australian Artificial Intelligence Institute (AAII), UTS. It would not have been possible to do without the support and guidance that I received from many people.

First, I extend my heartfelt thanks to my esteemed principal supervisor, Distinguished Prof. Jie Lu, for her invaluable supervision, support, and mentorship in my Ph.D. research and navigating life in a foreign country. Her guidance and care have given me tremendous strength to complete my PhD studies. Her patience, motivation, and immense knowledge have guided me at every confusing moment. I have learned so much from her, which would benefit my life. I also thank my co-supervisors A/Prof. Guangquan Zhang for his kind mentorship, insightful comments, and encouragement. His strict academic attitude and respectful personality have benefited my Ph.D. study. My sincere gratitude goes to Dr. Junyu Xuan for his help and guidance throughout my Ph.D. journey. His patience and professional feedback gave me so much support in every research stage. Studying under Dr. Xuan's mentorship has been an invaluable experience, and I am genuinely thankful for all the opportunities and support he has provided me.

As a member of DeSI Lab, AAII, and FEIT, I would like to thank all staff, team members, and students for their efforts and support. Thanks to Prof. Yi Zhang, Prof. Shiping Wen, Dr. Feng Liu and Dr. Hua Zuo for organizing and taking panel positions on my candidate assessments. Thanks to Wei Duan, En Yu, Wenting Zhang, Mengjing Wu, Xinheng Wu, Guangzhi Ma, Ming Zhou, Xiaoyu Yang, Zhaoqing Liu, Nelson Ma, Zelia

Soo, Yue Yang, Oscar Kiyoshige Garces, Pham Minh Thu Do, Dr. Feng Liu, Dr. Yiliao Song, Dr. Kairui Guo, Dr. Keqiuyin Li, Dr. Zhen Fang, Dr. Bin Zhang and Dr. Tianyu Liu for their assistance in my research and life. Thanks to Jemima de Vries, Jim Howes, and Michele Mooney for proofreading my manuscripts. I also thank Robyn Barden, Esin Morcos, Camila Cremonese, Lily Qian, and Margot Kopel for supporting my school life.

Thank you to all my friends for their tremendous support. I would like to thank Wenting Zhang and Mengjing Wu for the time we spent together when arriving in Sydney. Thanks to En Yu and Yiqiao Li, who enrolled offshore with me and have been struggling together through every critical stage. I am grateful to Xinheng Wu for encouraging me and advising me in many difficult moments. Thanks to Wei Duan for standing with me during my Ph.D. study. Special thanks go to my friends in China - Lijun Wu, Dr. Qianyun Yin, Meixin Sun, Li Wang, and Xuemin Cao for their continuous caring and encouragement. Thanks to Dr. Jieru Zhao and Dr. Qimin Zhang; though separated by vast oceans, we are united in one journey. Thanks to Dr. Bin Zhang, who has always supported me in every moment we shared and every challenge we have overcome.

Finally, I would like to express my gratitude to my parents for their unwavering support and unconditional love throughout my life. You have given me the strength and courage to pursue my dreams.

Zihe Liu

Sydney, Australia, 2024

List of Publications

1. **Zihe Liu**, Jie Lu, Junyu Xuan and Guangquan Zhang, "Deep Reinforcement Learning in Nonstationary Environments with Unknown Change Points," in IEEE Transactions on Cybernetics, DOI: 10.1109/TCYB.2024.3356981.

2. **Zihe Liu**, Jie Lu, Guangquan Zhang and Junyu Xuan, "A Behavior-Aware Approach for Deep Reinforcement Learning in Non-stationary Environments without Known Change Points," in Proceedings of the thirty-third international joint conference on artificial intelligence, IJCAI-24 Main Track, DOI: 10.24963/ijcai.2024/512.

3. Junyu Xuan, Mengjing Wu, **Zihe Liu**, Jie Lu. "Functional Wasserstein Variational Policy Optimization." in The 40th Conference on Uncertainty in Artificial Intelligence (UAI) 2024.

4. **Zihe Liu**, Jie Lu, Junyu Xuan and Guangquan Zhang, "Learning Latent and Changing Dynamics in Real Non-stationary Environments," submitted to IEEE Transactions on Knowledge and Data Engineering. [under review]

5. **Zihe Liu**, Jie Lu, Junyu Xuan and Guangquan Zhang, "Functional Detection and Adaptation in Non-stationary Environments," submitted to IEEE Transactions on Neural Networks and Learning Systems. [under review]

Table of Contents

List of Figures

Figure **Page**

List of Tables

Table **Page**

Abbreviation and Notation

Abbreviations

RL:	Reinforcement Learning
DRL:	Deep Reinforcement Learning
MDP:	Markov Decision Process
POMDP:	Partially Observable Markov Decision Process
HMM:	Hidden Markov Models
RSSM:	Recurrent State-space Model
MPC:	Model Predictive Control
RNN:	Recurrent Neural Network
CEM:	Cross-Entropy Method
GP:	Gaussian Process
DNN2GP:	Deep Neural Networks to Gaussian Process
MAP estimation:	Maximum A Posteriori estimation
MMD:	Maximize Mean Discrepancy
RBF kernel:	Radial Basis Function kernel
NTK:	Neural Tangent Kernel
KL divergence:	Kullback-Leibler divergence
WD:	Wasserstein Distance
DNN:	Deep Neural Networks
TD error:	Temporal Difference error
VPG:	Vanilla Policy Gradient
TRPO:	Trust Region Policy Optimization
PPO:	Proximal Policy Optimization
DQN:	Deep Q Network
A3C:	Asynchronous Advantage Actor-Critic
DDPG:	Deep Deterministic Policy Gradient

SAC:	Soft Actor-Critic
TD3:	Twin Delayed Deep Deterministic Policy Gradient
BGPG:	Behavior Guided Policy Gradients
DARL:	Detection-Adaptation Reinforcement Learning
BADA:	Behavior-Aware Detection and Adaptation
LLCD:	Learn Latent and Changing Dynamics
FDA:	Functional Detection Adaptation
EWC:	Elastic Weight Consolidation
GEM:	Gradient Episodic Memory
ODCP:	Online parametric Dirichlet Change Point
PC:	Policy Consolidation

Notations

Chapter 3

g_k	The gradient of the policy associated to the k_{th} task.
g	The gradient trained from a standard policy gradient.
$\tilde{g}$	The optimal gradient vector of policy adaptation.
$\bar{g}$	The normalized vector of the current gradient g.
h	The normalized vector of the closest gradient $\tilde{g}$.
v	The norm of current gradient g and optimal gradient $\tilde{g}$.
g_l	The gradient of the l_{th} layer.
$d^{(c)}$	The distance between policies.
$d^{(m)}$	The distance between state distributions.
W_l	The l-th layer neural network parameters.
ΔW	The parameter perturbation of neural network.

Chapter 4

π_θ	The policy with parameter θ.
τ	Trajectories of state, action and rewards.
Φ	The behavioral embedding map function.
$\mathbb{P}_\theta$	The behavior embedding distribution corresponding to the current policy π_θ.
$\mathbb{P}_{t-1}$	The behavior embedding distribution corresponding to the previous policy π_{t-1}.
$\mathbb{P}_{ep}$	The behavior embedding distribution corresponding to the policy at epoch ep.
$\mathbb{P}_{pre}$	The behavior embedding distribution corresponding to the previous converged policy.

$W(\mu, \nu)$ The Wasserstein distance between distribution μ and ν.

Chapter 5

$\mathcal{GP}$ A Gaussian Process.

$\mathbf{w}$ The weight of a neural network.

π The reinforcement policy.

$\mathbf{m}$ The mean of a Gaussian Process.

$\mathbf{K}$ The kernel of a Gaussian Process.

Λ The Hessian matrix.

$\mathbf{J}$ The Jacobians.

$\mathbf{v}$ The diagonal of the covariance.

p The prior distribution.

q The posterior estimation distribution.

S The surprise between prior and posterior.

Chapter 6

o_t The raw observation at timestep t.

s_t The latent state at timestep t.

h_t The deterministic variables at timestep t.

p The prior distribution.

q The posterior estimation distribution.

l_t The run length at timestep t.

CHAPTER 1

INTRODUCTION

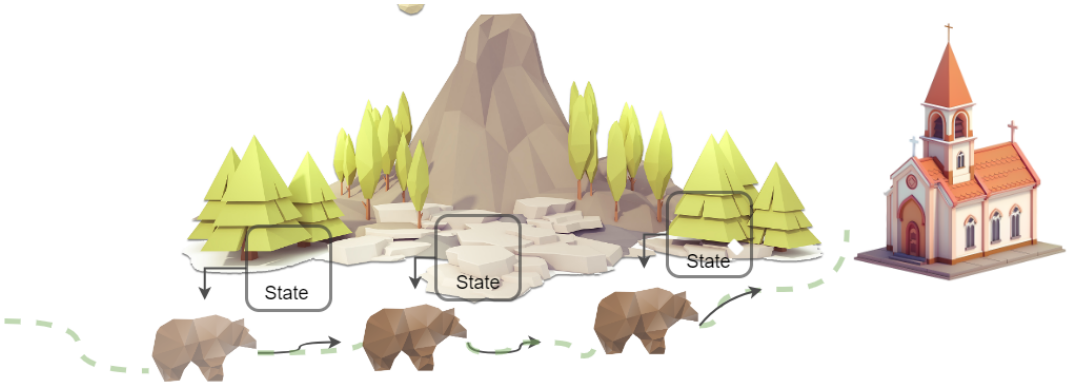
Machine learning has traditionally been centered around learning from static datasets, where algorithms infer patterns and make predictions based on historical data. However, the paradigm shifts when moving towards Reinforcement Learning (RL), an area of machine learning concerned with how intelligent agents should take actions in an environment to maximize the cumulative reward. Unlike its supervised and unsupervised counterparts, RL is distinguished by its focus on learning from interaction - agents are not merely passive recipients of data but active learners engaged in a sequence of decision-making that is both influenced by and influences the environment. With the development of deep neural networks, Deep Reinforcement Learning (DRL) integrates the power of deep neural networks, enabling the handling of unstructured and high-dimensional data spaces that were previously intractable. DRL stands at the forefront of machine learning's evolution, representing a cutting-edge blend of learning through interaction and deep neural network representation learning. This framework enables applications ranging from autonomous vehicles navigating through traffic [2] to algorithms mastering the game of Go [3], where the agent must learn to

make a series of decisions that lead to a long-term goal.

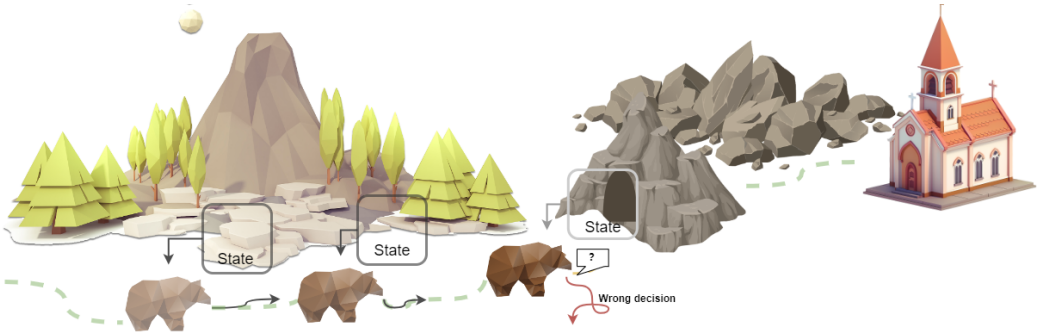
While DRL boasts remarkable successes across diverse fields, most schemes are designed to learn an optimal policy in an unknown but stationary environment with fixed state distributions, state transitions and reward functions [4]. Yet, many practical environments are inherently non-stationary, characterized by temporal dynamics and unpredictability. This disconnect posits a significant challenge: traditional DRL algorithms may struggle to maintain their performance when faced with evolving environmental factors. As Figure 1.1 shows, an agent with knowledge of a specific environmental condition may make wrong decisions when facing changes. This thesis aims to bridge this gap, extending the application of DRL to more realistic, dynamic environments with time-varying unknown non-stationarity.

1.1 Background and Motivation

Most machine learning can be characterized as the search for a solution that, once found, no longer needs to be changed [5], and so does deep reinforcement learning. This assumption allows for simplifying the learning problem and has led to many of the successes in DRL [6]. However, when this assumption does not hold, as is often the case that many real-world environments are non-stationary and sometimes change quite frequently [7], the learned policy may become obsolete or inappropriate for the new state of the environment, leading to a decline in performance. For example, in practical environments such as the fluctuating stock market, ever-changing e-commerce user preferences, unpredictable road conditions in autonomous driving, variable energy demands in smart grids, and the shifting landscapes of patient health in healthcare monitoring, traditional DRL algorithms can encounter significant challenges. As the environment evolves, errors are inevitable, and without timely and adaptive responses, the situation can deteriorate progressively [5]. The reason is that when a well-trained



(a) Stationary Environment



(b) Non-stationary Environment

Figure 1.1: Stationary and non-stationary environments. The state distribution, transitions and expected outcomes are consistent and predictable in a stationary environment. When an agent encounters non-stationary environments, such as a shift from daylight to a dark cave, it probably leads to a wrong decision due to the traditional DRL model's inability to adapt to the new conditions.

policy faces a new environment, traditional DRL algorithms often lack mechanisms for rapid adaptation and can be sample-inefficient, requiring large amounts of data to learn or relearn a suitable policy. This weakness can lead to suboptimal performance and escalating errors if the algorithms cannot adapt promptly and effectively to the continuous changes.

Figure 1.2 shows the abstract Markov Decision Process (MDP) setting of our research problem. The MDPs that agents interact with will come one after another, and agents may not see a specific MDP after it comes. There is no particular order of MDPs, and the steps that agents interact within each MDP are also not assumed. These MDPs

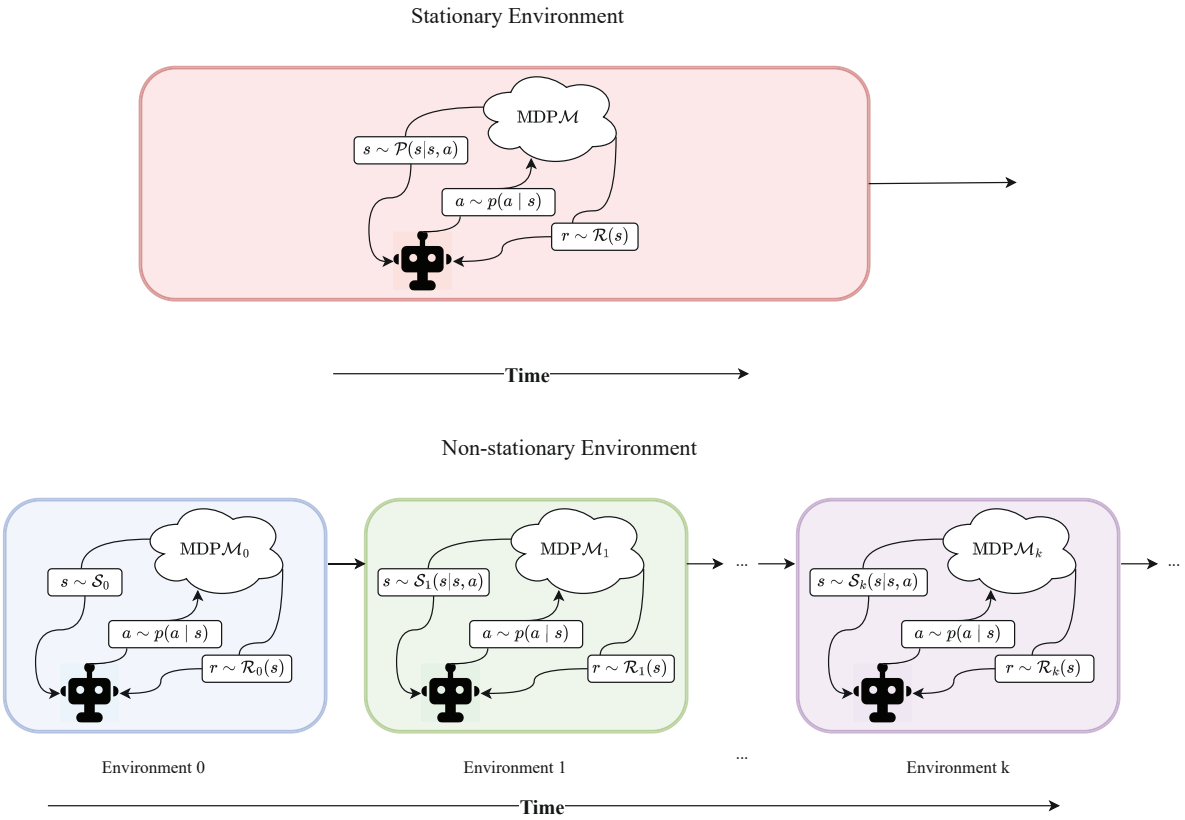


Figure 1.2: The upper figure illustrates a stationary environment where the underlying MDP is unknown but fixed. The bottom figure shows a non-stationary environment with the switching MDPs with different state distributions, transitions or reward functions.

may differ in any aspect, such as state distribution, transition probability and reward function. This thesis aims to address the critical need for DRL approaches that can not only recognize these shifts but also adjust to them in real-time, ensuring sustained and reliable functionality in the face of real-world unpredictability.

Several studies [8, 9, 10] have demonstrated the efficacy of DRL agents in dynamic environments with known change points by leveraging the transfer of knowledge from previously trained policies to adapt to new conditions quickly. However, in real-world scenarios, agents often encounter environments with unknown change points, such as a delivery drone navigating through varying weather conditions or a cleaning robot adapting to different floor types and unexpected obstacles. These environments are

characterized by unpredictable changes in state transitions and reward functions, making them challenging for traditional DRL approaches. Despite the growing importance of addressing this issue, there has been limited research focused on developing RL algorithms capable of handling such dynamic and uncertain environments. One solution is contextual detection for model-based RL [11], which detects environmental changes by maintaining several models and replacing the currently active model with the highest quality. Recently, Chen et al. [12] proposed an adaptive deep RL method for non-stationary environments with piecewise-stable context. They infer the context segment by Bayesian posterior and the belief context through observed data to adapt to the context changes. However, learning an environmental model is not easy for high-dimensional environments due to the limitations of sample efficiency. To address this, Lomonaco et al. proposed CRL-Unsup [13], which determines environmental changes using the difference between long-term and short-term reward averages and adapts to new environment conditions using continual learning methods [8]. This detection method requires a manually set threshold value. Alternatively, Padakandla et al. [14] proposed an approach that detects environmental changes by recognizing shifts in the distribution of the experience stream, albeit at the cost of higher computational complexity. Most current approaches do not consider different change extents, which is crucial to appropriately responding to policy learning.

This thesis aims to solve the long-existing problem by developing frameworks that can actively and accurately identify the changes in non-stationary environments and adapt to new environment conditions with the help of the change information. In non-stationary environments, recognizing environmental shifts allows agents to update their strategies promptly and maintain optimal performance with detected information. This adaptability is essential for real-world applications, where conditions can change rapidly and unpredictably. Effective change detection and adaptation mechanisms enable agents

to remain robust and effective in the face of evolving challenges, ultimately leading to more intelligent systems.

1.2 Research Questions and Objectives

We span the entire learning pipeline and primarily approach this problem using two inseparable processes: identifying the environmental changes through a set of methods and augmenting the policy training for the new environment by drawing on previously well-trained policies using the detected change point and information.

According to the two schemes, the research questions and research objectives are identified as:

QUESTION 1: *How to identify unknown change points in non-stationary reinforcement learning environments?*

Identifying change points in non-stationary reinforcement learning environments without predefined switch points presents a significant challenge. At the same time, ensuring the accuracy and speed of detection is crucial. Accurate detection allows the agent to adapt its strategy promptly and effectively, while fast detection minimizes the impact of changes on the agent's performance. For example, Lomonaco et al. [13] identify change points by observing the discrepancy between short-term and long-term rewards. This approach necessitates manual configuration of threshold values and window sizes. One aim is to employ various autonomous techniques that monitor environmental changes, ideally providing confidence intervals for decision-making. Additionally, we aim to solve the change detection problem without involving any extra parameters that must be tuned carefully.

QUESTION 2: *How to rapidly adapt to the new environment condition using detected change information?*

Once a change in the environment is detected, rapid adaptation is essential to maintain optimal performance. One practical approach is to leverage knowledge like [8, 15], where previous experiences are used to adjust to new conditions quickly. Another approach is to use additional regularization techniques to provide the information required by the new environment. For example, methods proposed in [16] and [17] introduce regularization terms that encourage the model to preserve valuable knowledge from previous tasks while learning new information. These approaches do not consider the response to different change levels. In contrast, we aim to achieve even more effective adaptation based on the information obtained from the detection process. By integrating the detected changes into the learning framework, we aim to develop algorithms that can dynamically adjust their learning in response to the nature and magnitude of the detected environmental shifts. For example, if a significant change is detected, the algorithm might increase exploration to gather information about the new environment quickly. Conversely, for minor changes, the algorithm could make subtle adjustments to the policy while maintaining its current knowledge base.

QUESTION 3: *How to enhance sample efficiency when adapting?*

When using knowledge from previous environments to adapt, a straightforward approach is to preserve experiences from well-learned environments. Still, the vast amount of data can lead to prohibitively high associated costs. Therefore, we focus on selecting *valuable* experiences from a well-learned environment to represent the knowledge acquired by the policy, ensuring efficient and effective knowledge retention. Then, we need to explore the strategy of retaining the most representative interactions between agents and environments. There are some works on improving sample efficiency based on reward [18], TD error [19] and global distribution [20] to benefit policy learning. For adaptation in new environments, we

need to explore a specific strategy to provide the most comprehensive information for our problem setting. The goal is to use the most negligible storage to avoid the extra cost of our framework.

QUESTION 4: *How to learn in non-stationary environments with high dimensionality?*

Model-free RL algorithms often need to take tens of millions of steps [21] to train a policy that is good enough, which is impractical in many open-world applications, especially health and safety-related scenarios. In contrast, model-based RL learns the transition and reward models to maintain high-quality environmental planning and prediction, known for higher sample efficiency. However, modeling high-dimensional RL environments is challenging because some data, like images, do not always constitute a Markovian space in practice [22, 23]. If the environment is non-stationary, it introduces additional computational costs. Therefore, the development of efficient mechanisms for the detection of environmental changes is imperative. These mechanisms must be swift and accurate to ensure timely adaptation, allowing the model to respond effectively to new conditions.

This research aims to achieve the following objectives, which are expected to answer the above research questions:

OBJECTIVE 1: *To develop methods that detect environment changes from data distribution analysis.*

This objective corresponds to Research Question 1. Our proposed algorithm analyzes the data collected during the interaction process between agents and the environment. By monitoring changes in these data, we aim to capture the underlying environmental variations. This approach allows us to detect shifts in the state space, changes in reward dynamics, or alterations in the transition probabilities, which are critical for adapting the agent's policy to maintain optimal performance.

These algorithms employ statistical and machine learning techniques to identify patterns and anomalies in the data that signify environmental changes, ensuring its continued effectiveness in non-stationary settings.

OBJECTIVE 2: *To develop methods that detect environment changes from deep learning properties.*

This objective corresponds to Research Question 1. Our proposed algorithm is grounded in deep learning theory, encompassing aspects such as the relationship between parameter changes and gradients during neural network training, as well as the connection between neural networks and Bayesian surprise to detect changes in the environment. This detection method, based on the models' response to environmental changes, offers a novel perspective to provide an innovative approach to change detection, enhancing the adaptability and responsiveness of deep learning models in dynamic settings.

OBJECTIVE 3: *To develop adaptation mechanisms using self-adjusted regularization.*

This objective corresponds to Research Questions 2. We aim to achieve knowledge reuse by adding additional constraints to the objective function while considering the degree of environmental changes. We employ behavior-guided and environment model-guided self-regulation constraints that adjust the learning process based on different environmental changes. This adaptive approach ensures that the retained knowledge is not only preserved but also effectively utilized to maintain optimal performance in the face of dynamic and evolving conditions. By integrating these mechanisms, our algorithm seeks to balance stability and plasticity, enabling the reinforcement learning agent to adapt efficiently to new situations without forgetting previously acquired skills.

OBJECTIVE 4: *To develop adaptation mechanisms according to deep learning proper-*

ties.

This objective corresponds to Research Questions 2. Our goal is to move beyond simply retaining and applying knowledge straightforwardly; instead, we aim to preserve knowledge that is specifically useful and relevant to the current environment. We aim to adapt to changes by utilizing the relationship between neural network parameter changes and gradients, as well as the connection between neural networks and Gaussian processes. By leveraging these theoretical foundations, our algorithm seeks to enhance the understanding and optimization of learning based on prior knowledge. This approach not only contributes to the advancement of reinforcement learning methodologies but also holds the potential to improve the performance and generalization of tasks across various applications.

OBJECTIVE 5: *To develop a trajectory selection method to improve the sample efficiency of previous environments.*

This objective corresponds to Research Question 3. We target selecting the most valuable trajectories that can represent a well-learned environment. With the selection, we can significantly improve the sample efficiency in model-free RL, which often needs millions of trials to search for an optimal policy. There are several reasonable selection strategies to pick memorable points for an RL task, such as choosing the highest rewarded experiences, picking the points near decision boundaries and selecting experiences with low TD error. We will investigate the selection method based on sequentially non-stationary environments, leading to better adaptation performance.

OBJECTIVE 6: *To develop a model-based RL framework that monitors the shift of complex changing environments with high-dimensional inputs.*

This objective corresponds to Research Question 4. We aim to design a model-based RL algorithm that can cope with potentially high-dimensional, non-stationary

environments with change point monitoring and rapid adaptation. Learning environment transitions and reward functions have some practical problems. Thus, we move model learning into a latent space, which is expected to demonstrate Markov transition properties more clearly and, at the same time, significantly reduce dimension compared to the raw observation space. At the same time, such a latent space is also beneficial for reducing the detection cost, making it a more efficient approach for managing dynamic environments in model-based RL.

1.3 Research Contributions

This thesis extends traditional deep reinforcement learning to more practical non-stationary environments with unknown change points. The main contributions of this study are concisely summarised as follows:

- A formal model-free deep reinforcement learning scheme in non-stationary environments with unknown change points. Our framework spans the entire learning pipeline and identifies the methods' similarities and differences with existing methods.

- A formal model-based deep reinforcement learning scheme in non-stationary environments with both detection and adaptation in the latent space.

- Two distribution-based change detection methods for reinforcement learning environments to accurately identify the change of RL states, transitions and reward functions, including high-dimensional and time-dependent data.

- An environment change detection method closely linked to Bayesian surprise. This method uses the GP predictive uncertainty that contains information about the distribution of these inputs to determine change points.

- An environment change detection method related to neural network updating. This method uses the relationship between neural network weights and gradients to identify sudden changes in environments.

- Two detection-boosted adaptation methods with regularization that quickly learn a new policy once a change is detected with an ability to reduce the effect from unrelated tasks.

- Two adaptation methods by using gradient constraints and optimizations to leverage the knowledge learned from the previous environment. We have considered the influence of change extents and avoided conflict gradient editing.

- A data selection strategy that enhances the sample efficiency of adaptation mechanisms and ensures high performance after environmental changes. This method reduces the cost of the adaptation scheme.

- A learning strategy to learn effectively in non-stationary environments with high-dimensional inputs. Our method detects the change points of non-stationary environments in the latent space online and can adapt to new environments rapidly.

1.4 Thesis Organization

The structure of the thesis is shown in Figure 1.3, and the chapters are organized as follows:

- Chapter 2 investigates the literature by categorizing studies based on whether the change points in the environment are known or unknown. Simultaneously, we also discuss the similarities and differences between our research question and other related areas in detail. This chapter allows us to systematically analyze

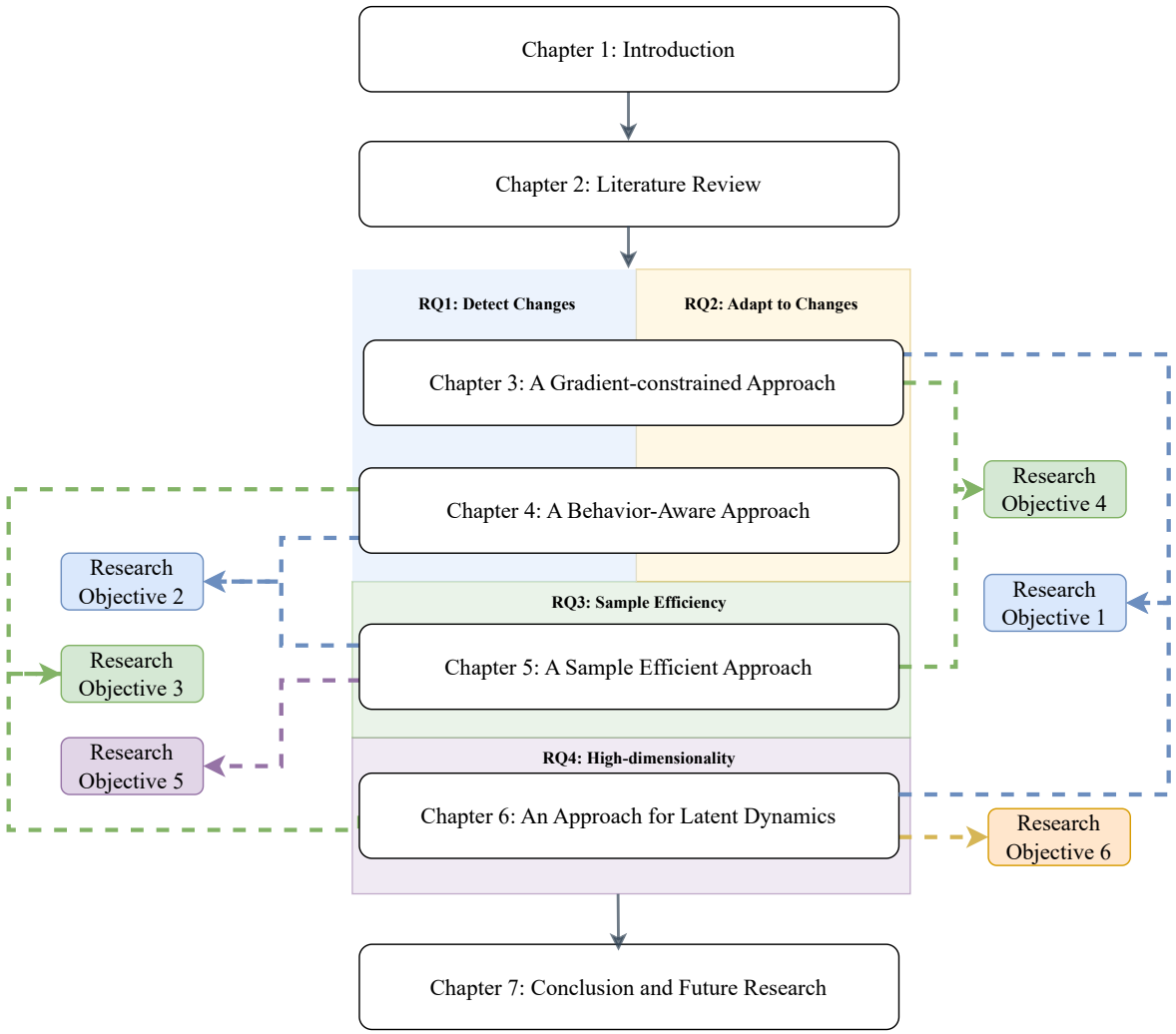


Figure 1.3: Thesis organization.

the methodologies, effectiveness, and limitations of RL algorithms in adapting to varying degrees of environmental non-stationarity.

- Chapter 3 presents a robust deep reinforcement learning algorithm for non-stationary environments with unknown change points. The algorithm actively detects change points by monitoring the joint distribution of states and actions. A detection-boosted, gradient- constrained optimization method then adapts the training of the current policy with the supporting knowledge of formerly well-trained

policies. Previous policies and experience help current policies adapt rapidly to environmental changes. Experiments show that the proposed algorithm accumulates the highest reward among several alternatives and is the fastest to adapt to new environments.

- Chapter 4 introduces Behavior-Aware Detection and Adaptation (BADA), an innovative framework that merges environmental change detection with behavior adaptation. The key inspiration behind our method is that policies exhibit different global behaviors in changing environments. Specifically, environmental changes are identified by analyzing variations between behaviors using Wasserstein distances without manually set thresholds. The model adapts to the new environment through behavior regularization based on the extent of changes. The results of a series of experiments demonstrate better performance relative to several current algorithms.

- Chapter 5 introduces Functional Detection and Adaptation (FDA) that incorporates change detection and adaptation to new environments. This method focuses on choosing the most representative trajectories of previous environments, addressing the problem of high storage cost and limited access to the past environments. Then, we employ Bayesian surprise to detect environmental changes. It also utilizes the Gaussian process posterior to provide knowledge for the new environment. Experimental results demonstrate the effectiveness of this approach in handling non-stationary and evolving environments.

- Chapter 6 proposes a new model-based reinforcement learning algorithm that proactively and dynamically detects possible changes and Learns these Latent and Changing Dynamics (LLCD) in a latent Markovian space for real non-stationary environments. To ensure the Markovian property of the RL model and improve computational efficiency, we employ a latent space model to learn the environment's

transition dynamics. Furthermore, we perform online change detection in the latent space to promptly identify change points in non-stationary environments. Then we utilize the detected information to help the agent adapt to new conditions. Experiments indicate that the rewards of the proposed algorithm accumulate for the most rapid adaptions to environmental change, among other benefits.

- Chapter 7 gives a brief summary of the thesis and its contributions. Potential future studies are summarized as well.

CHAPTER 2

LITERATURE REVIEW

The problem of Deep Reinforcement Learning (DRL) in non-stationary environments has been a focal point of research in several studies. In real-world scenarios, environments can exhibit a spectrum of complexity, ranging from relatively simple settings with predefined change points to highly dynamic and unpredictable situations where the temporal changes are unknown. To gain a comprehensive understanding of the existing approaches and challenges, we investigate the literature by categorizing studies based on whether the change points in the environment are known or unknown. Simultaneously, we will discuss the similarities and differences between our research question and other related areas in detail. This point of view allows us to systematically analyze the methodologies, effectiveness, and limitations of DRL algorithms in adapting to varying degrees of environmental non-stationarity.

In this chapter, we first summarize prior works similar to our research problem, explicitly focusing on Deep Reinforcement Learning that continuously adapts in dynamic environments with unknown change points. This review will provide a foundational understanding of the current state of the art and identify critical areas where our

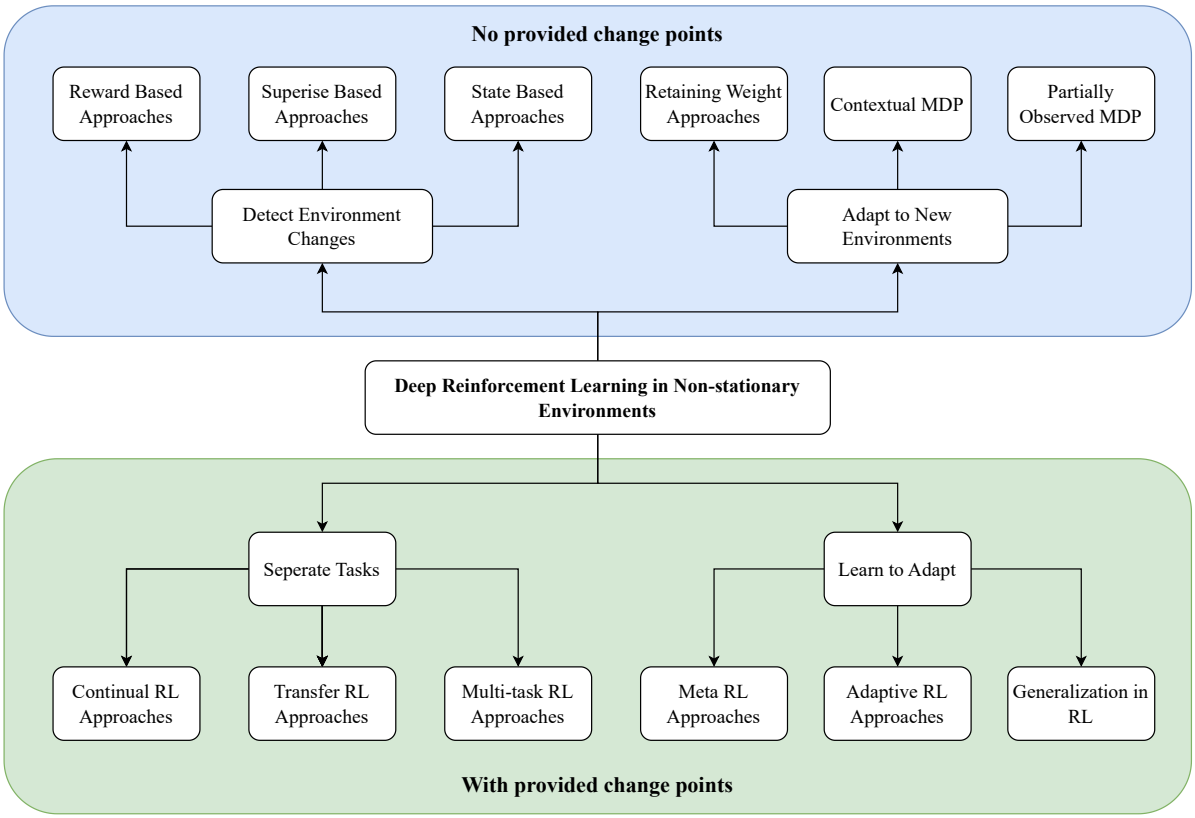


Figure 2.1: The structure of the literature review.

research can contribute to the field. Following that, we analyze works in related fields such as meta Reinforcement Learning, continual Reinforcement Learning, and transfer Reinforcement Learning. We elucidate the differences between these approaches and our focal problem from various perspectives, including problem settings and solution methodologies.

2.1 Deep Reinforcement Learning Concepts and Frameworks

DRL allows intelligent agents to learn an optimal policy by representing the complex relationships between states and actions in an interactive game-like environment [24].

Its powerful ability to model sequential decision-making has seen DRL widely used for various applications [25, 26, 27, 28, 29].

A wide of representative works have been proposed, including model-free and model-based approaches, both sharing the common goal of enabling agents to learn optimal policies in complex environments. Model-free RL directly learns a policy or value function from interactions with the environment, bypassing the need for an explicit model of the environment's dynamics. This approach does not have access to the transition probability distribution. Model-free approaches are generally more straightforward, requiring much data to achieve satisfactory performance. Representative algorithms include Vanilla Policy Gradient (VPG) [30], Trust Region Policy Optimization (TRPO) [31] Proximal Policy Optimization (PPO) [32], Deep Q Network (DQN) [24], Asynchronous Advantage Actor-Critic (A3C) [33], Deep Deterministic Policy Gradient (DDPG) [34], Soft Actor-Critic (SAC) [35], Twin Delayed Deep Deterministic Policy Gradient (TD3) [36] etc.

Among these state-of-the-art methods, VPG [30], TRPO [31] and PPO [32] are on-policy algorithms whose function is learned from actions we took by the current policy. There is a sampling policy and an update policy in the update scheme of model-free RL. The sampling policy is the policy an agent follows when choosing which action to take in the environment at each time step, and the update policy is how the agent updates the Q-function. On-policy algorithms attempt to improve upon the current behavior policy that is used to make decisions. Therefore, these algorithms learn the value of the policy carried out by the agent. Off-policy algorithms learn the value of the optimal policy and can improve upon a policy different from the behavior policy. Off-policy DRL maintains a behavior policy and a target policy. While the behavior policy generates actions for observed states, the target policy is trained iteratively using the subsequent outcome of the action. The vanilla off-policy algorithms include DQN [24], TD3 [36], DDPG [34] and SAC [35].

Reinforcement learning can be applied in many domains, and the most famous application is AlphaGo [37]. Furthermore, operations research [38], robotics [39], traffic systems [40], recommender systems [41] and video games [42] are all making use of DRL. For example, the application of DRL in robotics usually aims to create intelligent machines that can assist humans in a variety of tasks and execute tasks that are beyond human capabilities. Robotics can achieve the same task more safely and efficiently than human beings. Robots are also used in space missions, emergency surgery, and hospital meal preparation [43], among other things. DRL is applied successfully in robotics with tasks like navigation [44, 45], control [46], target search [47, 48], multi-agent coordination [49, 50], manipulation [51, 52, 53] and transport [54]. This tripartite division underlines DRL's adaptability across different robotic terrains and objectives. The environment in which robotics work is relatively complex and changes over time. However, agents usually have difficulty learning in these conditions because the rewards are sparse; that is, the reward will not be obtained at each time step, and only a few actions will be rewarded.

Another closely related application area of DRL is recommendation systems. The problem of recommending the best items to a user is not only a prediction problem but a sequential decision problem [55]. This suggests that the recommendation problem could be modeled as an MDP and solved by DRL methods. A large amount of work [56, 57, 58] applied DQN in recommender systems, and others use Actor-Critic methods [59, 60]. In a typical DRL setting, an agent aims to maximize a numerical reward through interaction with an environment. This is analogous to the recommendation problem, where the recommend algorithm tries to recommend the best items to the user and to maximize the user's satisfaction.

However, the prior work on DRL in specific non-stationary environments is limited. Most of them hold the assumption that the data are coming from a set of fixed states and

take actions in a stationary environment. The reason why DRL does not work well in complex environments is that adapting DRL to dynamic problems is the first step for the DRL controller to more general applications, which has the central issue of scalability.

2.2 Deep Reinforcement Learning in Non-stationary Environments

Addressing the problem of Reinforcement learning in non-stationary environments with unknown change points typically involves two approaches. The first entails actively detecting changes and adapting upon identifying such alterations. In contrast, the second approach does not include change detection but focuses on continuous adaptation at each time step.

2.2.1 Detecting Environment Change Points

Methods involving detection can more accurately identify task information and improve algorithm efficiency, as adaptation typically begins only after changes have been detected.

Detect Dynamics Changes One approach is to model the environmental dynamics, i.e., the transition function of the environment to be interacted with. With the modeled dynamics, monitoring its change is straightforward. The classical approaches to this problem are based on context detection. RL-CD (Reinforcement Learning with Context Detection) [11] is the first presented algorithm for detecting RL environmental changes. The core idea is to create and simultaneously update multiple partial models of the environment dynamics. There are several partial models in the system, and each model is used in different contexts. The method of detecting changes is introducing a quality signal for the partial model. Thus, a confidence value that reflects the model performance is presented. The confidence value relates to the trial times of an agent in a given

state, so it is proportional to the quality of the models. RL-CD detects environmental changes by replacing the currently active model with the highest quality. If the best model performs poorer than a threshold, a new model is created to learn an optimal policy. Based on RL-CD, Hadoux et al. [61] considered the non-stationary environment a set of contexts (or modules). In the inside of one context, the environment is static. The authors proposed a novel algorithm that detects context changes by learning a group of unknown contexts, known as sequential analysis. Similarly, this algorithm estimates the transition and reward functions but has fewer parameters. Banerjee et al. [62] pointed out that what is optimal for optimizing rewards may not necessarily be optimal for the quickest detection of model changes. This work computed Shiryaev algorithm [63], CUmulative SUM statistic (CUSUM) [64] and Shiryaev-Roberts (SR) statistic [65] on a sequence of random variables, i.e., the state and action pair over time, and switch to the optimal policy for the model at the time when the change detection algorithm crosses its threshold. Further, Alegre et al. [66] constructs a mixture model that ensembled several probabilistic dynamics predictors. They also proposed the change detection for the underlying MDP via a multivariate variant of CUSUM [67, 68] statistics. However, sometimes, the drift of environments only permits limited interaction before the changes occur in environmental properties. These interaction data reflect the transition function and reward function of the environment while modeling the dynamics directly, which is laborious and impractical in many scenarios.

Data Distribution Another approach to address this problem is to track changes in the distribution of training data during the learning process. Azayev et al. [69] had trained an extra classifier to determine which terrain the robot encounters. The method is close to supervised learning because each terrain has a label. Padakandla et al. [14] used an online change detection method [70], which is designed for the compositional multivariate data modeled as Dirichlet distributions and divides the multiple change

point detection into a sequence of single change point detection. There is only one active window of a specific size, and the detection is carried inside. If a change point is detected, the beginning of a new window is set to this point. Chen et al. [12] proposed an adaptive deep RL method for non-stationary environments with piecewise-stable context. Their method can infer the context segment structure and the belief context accordingly from observed data, which can be leveraged to detect and adapt to context changes. The detection method is similar to Bayesian Online Change Detection (BOCD) [71], estimating the posterior for the current segment length.

Superise-based Methods Bayesian surprise [72] is a general concept derived from the first Bayes principles. For data $\mathscr{M}$, given a prior distribution $P(\mathscr{M})$ of beliefs, the fundamental effect of a new data observation D on the observer is to change the prior distribution into the posterior distribution $P(\mathscr{M} \mid \mathscr{D})$ via Bayes theorem. Thus, the surprise is defined by the average of the relative entropy or Kullback-Leibler divergence between the prior and posterior distributions. The concept can be used in outlier detection. Thus, Nagayoshi et al. [73] proposed a representation method based on environment entropy. The method tracks the entropy change in response to changing conditions. However, in reality, the simulated environments are complex and difficult to reuse. Similarly, in continual learning studies like [17, 16], they calculated the surprise when the model is updated using each minibatch and performed Welch's t-test [74] on the adjacent surprise.

Reward-based Methods Lomonaco et al. [13] proposed and open-sourced CRLMaze, which is a new benchmark for continuous reinforcement learning in a 3D changing environment based on ViZDoom [75] and designed a variety of environmental changes. The threshold for triggering change determination is the difference between the short-term and long-term rewards. When the difference goes under the threshold, changes in the reward function or non-stationary interrupts of the learning process can be mitigated

by consolidation. However, reward-based methods are limited in environments with sparse rewards and need manually set thresholds.

Learning Task Embeddings Lin et al. [76] learned the variance as a function in a dynamic sparse reward environment with continuous action space. A significant variance increases exploration in a state with a low chance of getting a high reward. Similarly, Sutton et al. [5] tracked environment changes by fine-tuning the policy continuously and underscored the importance of tracking in domains with temporal coherence for meta-learning. Xie et al. [77] leveraged latent variable models to learn a representation of the environment from current and past experiences and perform off-policy RL with this representation. Most of these methods closely link to meta-learning [78, 79]. They model some task-specific vectors, which can be used to predict the most transferable source task for a given target task via the similarity between task embeddings.

2.2.2 Adapting to New Reinforcement Learning Environments

Adaptation to new environments has been extensively researched, with considerable work done in various fields. In this section, we will focus only on methods relevant to this thesis's settings. Other approaches will be discussed and compared in detail in the following sections, providing a comprehensive overview of the current state-of-the-art and highlighting the unique contributions of our proposed methods within the broader context of reinforcement learning and environmental adaptation.

Model-based methods da Silva et al. [11], and Hadoux et al. [61] estimated the prediction quality of different models and instantiated new ones when none of the existing models performed well in discrete settings. Further, Alegre et al. [66] extend it to the continuous setting via a mixture model composed of a (possibly infinite) ensemble of probabilistic dynamics predictors that model the different modes of the distribution over underlying latent MDPs. Banerjee et al. [62] proposed a two-threshold switching policy

based on KL divergence between transition models to adapt to different environments, requiring prior knowledge of the dynamics.

Model-free methods Lomonaco et al. [13] used the gap between the short-term and long-term rewards to indicate environmental shifts. Their adaptation method is based on Elastic Weight Consolidation (EWC) [8], which aims to overcome catastrophic forgetting in continual RL scenarios. Padakandla et al. [14] update the Q-functions when a change point is detected. Another approach to adaptation is to adapt to the environment continuously rather than detect change points. One idea proposed by [80] is that when an environment changes, the learning records collected by the agent tend to increase entropy. Thus, they introduced the new concept of surprise. This solution involves a confidence model that describes the state most familiar to the agent and how it relates to the distribution of states they have experienced. Experiencing a more familiar state will result in a higher establishment function. The goal of the subsequent behavioral strategy will be to select the behavior that allows the subject to continue in the most familiar state. Kaplanis et al. [81] added an extra loss item into PPO to simultaneously remember the agent's policy at various timescales to learn without forgetting and adapting to new environments. This approach saves on detection costs but performs poorly when faced with more significant or more frequent environmental changes, especially when there are changes in observations.

2.3 Multi-task Deep Reinforcement Learning

Multi-task RL aims to learn a shared policy for a diverse set of tasks. Multi-task RL encompasses a wide array of transfer learning style methods. At its core, it is training a single model to solve multiple tasks. The main challenge of multi-task RL is the conflicting gradients among different tasks.

Some previous online RL works address this problem via gradient surgery [82], which

projects the conflict gradient onto the normal vector of the other gradient. Liu et al. [83] solved this problem via conflict-averse learning by finding the best update vector within a ball around the average gradient that maximizes the worse local improvement between task 1 and task 2. There are some prior works using parameter composition [84, 85]. Yang et al. [84] introduced an explicit modularization technique that uses a routing network to reconfigure a base policy network for each task through soft combinations of possible routes, improving sample efficiency and performance on various simulated robotics manipulation tasks. Sun et al. [85] proposed a parameter-compositional approach that learns a policy subspace represented by a set of parameters, allowing policies for individual tasks to be composed by interpolating in this subspace.

For the offline setting, Decision-Transformer-based methods [86, 87, 88] rely on expert trajectories and entail substantial training expenses. Yu et al. [88] proposed a method that conditions a robotic policy on task embeddings comprised of visual demonstrations and language instructions. This allows these two modalities to clarify ambiguities and improve generalization performance over using either alone for complex pick-and-place tasks. Lee et al. [87] demonstrated that scaling up transformer-based models trained on diverse datasets using an offline reinforcement learning approach, similar to methods used in vision and language domains, can produce highly capable generalist agents. Kumar et al. [89] demonstrated that with appropriate design choices like ResNets, cross-entropy distributional backups, and feature normalization, large-capacity offline Q-learning models trained on heterogeneous datasets can achieve strong performance that scales with model size. Yuan et al. [90] tackled the challenge of learning robust task representations in offline meta-reinforcement learning by proposing a contrastive learning framework with a bi-level encoder structure that maximizes mutual information between task representations and rewards, using negative sample approximations to make the representations invariant to the mismatch between training and test behavior

policy distributions. There are also some work based on gradient descents in the finetuning stage, such as Sun et al. [91], which presents a self-supervised multi-task pretraining framework for sequential decision-making tasks that uses a Control Transformer coupled with a carefully designed control-centric pretraining objective to learn transferable representations that capture essential information for both short-term and long-term control. Similarly, Taiga et al. [92] employed a method where an agent is pretrained on multiple variants of the same Atari 2600 game before being fine-tuned on previously unseen variants. Maurer et al. [93] extracted features for multiple tasks in a single low-dimensional shared representation. Eramo et al. [94] further highlight the benefits of learning a shared representation, as error propagation in approximate value iteration and policy iteration improves when learning multiple tasks jointly.

These works aim to solve multi-task problems, which maintain consistency among gradients in the subspace of tasks, thereby finding a compromise that satisfies multiple tasks simultaneously. In our setting, where the environment is constantly changing and unpredictable, the goal is to leverage knowledge gained from similar past experiences to perform better in the current environment rather than striving to maintain performance across all previously encountered environments.

2.4 Continual Deep Reinforcement Learning

Continual reinforcement learning algorithms aim to mitigate catastrophic forgetting, where learning new tasks causes the agent to forget previously learned knowledge. In a continual setting, the boundaries between tasks are often well-defined, and the agent is expected to learn each task in sequence without forgetting previous tasks. According to [95], the prior work can be divided into explicit parameter-based methods, replay-based methods and structure-shared methods.

Parameter-based methods The parameter-based approach, as exemplified by [9],

involved using the representations learned by networks from previous tasks as inputs for subsequent tasks. Another approach is to retain a prior concerning the historical usage levels of each parameter during learning, thus preserving significant prior knowledge [8]. Regrettably, this stability might constrain the possibility of backward transfer during the process. Similar approaches [96, 97] accomplish this objective by utilizing the principle of superposition, in which context information for each task is preserved, allowing the weights to be distinctly broken down into orthogonal sub-networks. There are some distillation-based researches [98, 99, 100, 101] involving using one neural network as a reference or soft target for another. This technique can enhance the training process by supplying an additional auxiliary target, which the trained network aims to replicate. Another widely used approach is to emphasize the significance of past experiences.

Replay-based methods Another approach of continual RL is leveraging experience replay. Replay methods like [102, 103, 104] can, therefore, assist in correcting the short-term bias present in their objective function, provided that past experiences are a reliable approximation for new events. To address the problem of significant storage cost, some methods proposed pseudo-rehearsals sampled from a generative model [105, 106]. Lopez-Paz et al. [15] added task labels as an extra input, storing the training data in episodic memory. When a new task appears, the gradient of the previous task is calculated to constrain the gradient of the current task. Instead of using shared replay, Kessler et al. [107] learn a factorized policy, using the same feature extraction layers but different heads, each specializing in a new task. This allows it to select the best policy for an unlabeled task.

Structure-shared methods Some approaches within the structure-shared methods domain emphasize modularity and composition [108, 109, 110, 111], where modules specialized for each task can be composed for related tasks. Other research [112, 113, 114] focuses on the skills learned by the network, which can be generalized to other

tasks. Similarly, work by [115, 116] concentrates on integrating skills and composition seamlessly, facilitating the explicit reuse of previously acquired knowledge in the form of skills. Some studies center on goals, which can be interpreted as states the agent aims to reach, a reward the agent must achieve, or a termination point of a skill. An ambitious approach is to discover general-purpose goals without relying on any reward signal, akin to unsupervised learning [117, 118, 119, 117].

Lifelong learning While continual learning focuses on sequential task learning within a specific domain, lifelong reinforcement learning [120] takes a broader perspective. It aims to enable an agent to continuously acquire and transfer knowledge across various domains throughout its lifetime, posing additional challenges beyond catastrophic forgetting. The core settings in most work are similar to continual reinforcement learning. Fu et al. [121] presented a model-based lifelong reinforcement learning strategy that enhances sample efficiency by distilling a hierarchical Bayesian posterior, facilitating forward and backward knowledge transfer. Lu et al. [116] proposed a method leveraging unsupervised skill learning and a dynamics model for planning, reducing the need for extensive real-world interaction. Aljundi et al. [122] introduced another gate to compare which new tasks are most similar to the previous training. The parameters of the new network are then initialized according to the most similar task. Xie et al. [123] measured the similarity between the past samples and the current task's transition dynamics to determine which samples to transfer in the online fine-tuning phase.

As a distinction between continual and lifelong reinforcement learning, our motivation is to enable the agent to adapt to changing environments and maintain stability during changes. The underlying reason is that agents rarely encounter identical and repeat environments in the practical scenario. Consequently, it is crucial to leverage prior knowledge for rapid adaptation, ensuring optimal performance within the ever-changing online environment. Furthermore, the change points of environments are unpredictable,

which makes it challenging to deploy continual learning methods to our setting directly.

2.5 Transfer Deep Reinforcement Learning

Transfer Learning (TL) refers to leveraging knowledge learned from one task, usually named as the source domain, to boost the performance of machine learning models on another task, usually named as the target domain. A comprehensive survey of transfer learning is given in [124].

In reinforcement learning, transfer learning techniques are faced with more challenges. Because the MDPs have complicated components, knowledge from the source domain can be transferred in different ways [125]. The knowledge to be transferred can be rewards, policies, demonstrations, or representations learned by deep neural networks. Amounts of works have studied how to transfer these types of knowledge to boost the performance of reinforcement learning models.

Share rewards The most intuitive way to share knowledge in reinforcement learning is to share rewards. Reward sharing is a type of method to construct the distribution of rewards in the target domain utilizing knowledge from the source domain. Potential based Reward Shaping (PBRS) [126] is the most classic one among these types of methods. In PBRS, a shaping function was proposed to measure the differences between two potential functions. The shaping function provides the rewards containing knowledge from the source domain to help agents make better decisions. Then Potential Based state-action Advice [127] was proposed, where potential functions include actions as well. Dynamic Value Function Advice [128] developed a framework to incorporate arbitrary knowledge into dynamic potential functions via reward sharing.

Learning from demonstrations Another common type of shared knowledge is demonstrations, which can lead to more efficient explorations. Among these methods, a common assumption is that the source and target Markov Decision Processes are the

same. Demonstrations are first introduced in [129]. Then Direct Policy Iteration with Demonstrations [130] was proposed. Two complete demonstrated rollouts are sampled. The one is from an expert policy. The other is from the self-generated. Then, the union of two rollouts is utilized to learn the estimation of values in the Q table. In Approximate Policy Iteration with Demonstration [131], only the rollout from the self-generated is used to estimate the values in the Q table. The rollout from expert policy is used to learn a value function. In [132], two separate replay buffers are used to cache the demonstrated and self-generated data. Thus, the expert demonstrations can continuously be sampled from the data in the buffer. In [133], a potential function represents the highest similarity between a state-action pair and the expert demonstrations. In Generative Adversarial Imitation Learning [134], the state-action distributions under a given policy are measured using an occupancy measure. Then, the new reward function is to maximize the accumulated rewards encouraged and to minimize the distribution divergence between the current policy and the expert policy. The RL can be transferred into an optimization problem. Further, Kang et al. [135] improved the policy optimization compared to [134].

Policy transfer The main idea of policy transfer is to use pre-trained policies from source domains in the target domain. Usually, the number of source domains is not less than one. A student policy is learned from multiple teacher policies by minimizing the divergence of the distributions of actions. In [136], the KL-Divergence is used to measure the divergence of the distributions of each teacher and student. In [137, 138],the trajectories of the teacher's policy are replaced with that of the student's policy during optimization. Besides policy distillation, another idea is to reuse policies. Policies from source domains are reused directly in the target domain. In [139], policies from source domains are weighted and the target policy is yielded from the weighted combination of these policies. Other similar methods include [140, 141].

Inter-task mapping These methods assume that there exists a one-to-one mapping

between source domains and the target domain. Then, the mapping can be used to transfer knowledge. In [142, 143, 144], mapping functions over the state space are learned from data. In [145, 146], a mapping function over the transition dynamics space is learned. These methods assume that there exists a similarity between the transition probability and the state representations between source and target domains.

Share the representations The last category of methods is to share the representations learned by deep neural networks. In [147], progressive neural networks share the representations. Progressive neural networks consist of multiple columns, each for one specific task. For the target task, the columns for source tasks are frozen and the representations from these columns are applied to the new column. Then Fernando et al. [148] adopted a similar idea but used a fixed-size network. Instead of reusing representations, other methods learn a disentangled representation across source and target domains. These methods include Successor Representations [149, 150, 151] and Universal Function Approximation [152, 153]. Eysenbachet al. [154] proposed an estimated modified reward function to transfer experience. Xie et al. [123] solved multi-task reinforcement learning by retaining and reusing prior experience. Chen et al. [155] introduced a one-test-time trial scenario, where an agent must complete a task within a single episode by learning to imitate fixed prior experiences.

It is noted that the common theme in these works is the presence of shifts between train and test settings, while our setting lays on constant online adaptation with active change detection simultaneously.

2.6 Meta Deep Reinforcement Learning.

Meta-reinforcement learning (meta-RL) is a paradigm in reinforcement learning that aims to learn how to quickly adapt to new tasks or environments rather than solving a single fixed task. The key idea is to leverage experience from a distribution of tasks to

learn a general strategy that can be efficiently fine-tuned or adapted to solve new tasks from the same distribution. More specifically, in meta-RL, there is a meta-training phase where the agent is exposed to a set of training tasks sampled from a task distribution. The goal is to learn a meta-policy or meta-learner that can quickly adapt to any new task from the same distribution during a meta-testing phase, using only a few examples or a small amount of experience from the new task. The meta-training process can be thought of as learning a good initialization or representation that captures common structure across tasks, which can then be fine-tuned or adapted efficiently for each new task. Meta-RL algorithms aim to learn this efficient adaptation strategy during meta-training.

Some critical approaches in meta-RL include Model-Agnostic meta-learning, recurrent models, gradient-based methods and context-based methods.

Duan et al. [156] extended this idea to introduce an on-policy meta RL algorithm corresponding to training an extra network with hidden states maintained across the whole training procedure to help the policy learn new tasks rapidly. Aghapour et al. [157] proposed a double meta RL algorithm, which adds an extra meta-model that learns the dynamics of the environment and generates data to meta-train the policy. Z Xu et al. [158] proposed a gradient-based meta-learner that improves the performance with only a few gradient update steps by making use of the task embedding. Furthermore, Xie et al. [77] leveraged latent variable models to learn a representation of the environment from current and past experiences and perform off-policy RL with this representation. However, this view may ignore the sequential nature of these tasks where learning should be greedy. To differentiate with our motivation, we emphasize that in meta RL, the training and testing tasks are separate, and the adaptation occurs while testing rather than learning. Also, most meta-RL methods assume that tasks come from the same distribution, while our motivation lies in handling more drastic environmental changes.

2.7 Contextual Markov Decision Process

There is also some fundamental research on MDP. Hallak et al. [159] introduce contextual MDP, which involves tuples with hidden parameters that are constant over time. The different tasks of the non-stationary environment are identified by clustering transition models. Modi and A. Tewari [160] studied the case where the transition kernel of each MDP is specified with a generalized linear model of the context. Modi et al. [161] used the context or side information to model multiple tasks in smoothly varying environments and linear structured MDPs. Besides, many prior works have studied Contextual MDP [162, 163, 164], where contexts are sampled once and are fixed throughout each episode. Building upon this, Dynamic Contextual MDPs are a generalization case, where contexts can change over time in episodes. Tennenholtz et al. [165] considered history-dependent dynamics of contexts and can capture slow changes. In contrast, Mao et al. [166] aimed at a non-stationary contextual MDP is considered without dependence on previous actions and states. Similarly, Ren et al. [167] propose a Bayesian approach to learning contextual MDPs where dynamics are not state-action-dependent or history-dependent. Sodhani et al. [6] proposed a block contextual MDPs, utilizing Lipschitz properties to ensure the generalization ability to unseen tasks.

2.8 Partially Observed Markonv Decision Process

A partially observable Markov decision process (POMDP) [168, 169, 170] is also a generalization of an MDP. A POMDP models an agent decision process in which it is assumed that an MDP determines the system dynamics, but the agent cannot directly observe the underlying state. One approach is to estimate the real POMDP model, which contains a sensor model to the probability distribution of different observations given the underlying state [171, 172], which build on the estimation of the parameters of hidden Markov

models (HMMs) using spectral methods [173]. Instead of estimating the real POMDP model, Jin et al. [174] utilized the observable operator model. Similarly, under completeness assumptions, Guo et al. [175] proposed an offline RL algorithm that constructs confidence regions for Bellman operators that characterize POMDPs. Jin et al. [176] added a bonus to the rewards that penalize states not well-covered by the observed ones. Zanette et al. [177] constructed an MDP model on which the performance of any policy lower bounds that of the natural environment and then learned a near-optimal policy on this model. Our idea is closest to the second approach. The POMDP framework is general enough to model various real-world sequential decision processes. The research problem in this thesis distinguishes POMDP from the assumption of the underlying MDP changes.

2.9 Concept Drift

Our research problem is closely linked to concept drift [178, 179], which involves a non-stationary data stream with changing labels and data. These data streams present novel challenges for machine learning models, particularly concept drift. Concept drift pertains to the phenomenon where the statistical properties of the target variable change unpredictably over time. It poses a significant challenge to real-world machine learning algorithms operating in dynamic and evolving environments. Concept drift is a prevalent issue across various fields, including computer and telecommunication systems, traffic monitoring, personalized recommendation systems, and medical decision support, among others [179].

The most direct approach to address concept drift is to retrain a new machine learning model with the incoming data. Upon detecting a drift signal, it's typical to train a new model to replace the old one. Alternatively, adaptive models possess the capability to update themselves partially when concept drift is identified. This method proves to be

more efficient, particularly when the drift occurs within a localized region. However, these adaptive methods are constrained to specific types of models, such as tree models [180, 181, 182], lazy learners [183, 184], and support vector machines [185].

In recent years, ensemble learning has garnered considerable attention in the field of machine learning [186]. Ensemble methods consist of a collection of base classifiers, which may vary in type or parameters. The predictions of each base classifier are combined using specific voting rules to make predictions on newly arrived data. Numerous adaptive ensemble methods have been devised to address concept drift, either by extending classical ensemble methods or by devising specialized adaptive voting rules. Ensemble methods encompass various techniques [187, 188, 189].

Deep learning has demonstrated remarkable success across a spectrum of applications. Deep neural networks are typically updated through gradient descent-based optimization techniques, a methodology that can be extended to data stream settings for online updating. In a recent study by Soleymani et al. [190], a pre-trained convolutional neural network is proposed, with its parameters continuously updated online to adapt to concept drift. Additionally, various deep model architectures have been explored to tackle concept drift, including recurrent neural networks [191] and Long Short-Term Memory (LSTM) networks [192]. However, a key challenge with deep learning lies in its data hunger; deep neural networks require substantial amounts of data for effective training. This poses a significant challenge when handling concept drift, as only limited data may be available for training post-drift detection.

CHAPTER 3

A Gradient-Constrained Approach

This chapter aims at the research objectives 1 and 4 mentioned in the Chapter. 1. In this chapter, we present a novel approach for detecting change points by monitoring the joint distribution of states and actions. Central to the method is leveraging the relationship between the distance metric in neural networks and the associated gradients. Specifically, a relaxed, constrained gradient optimization is employed as an adaptation mechanism. By monitoring shifts in the joint distribution over states and actions, the algorithm can identify points where the environments undergo a transition. The neural network distance metric provides a principled way to quantify these gradient changes through the geometry induced on the network's parameter space. The gradient-based adaptation allows the framework to adjust to detected changes stably and efficiently by leveraging knowledge from the previous operating environment.

This chapter is based on the paper "Deep Reinforcement Learning in Non-stationary Environments with Unknown Change Points" IEEE Transactions on Cybernetics, 2024 (DOI: 10.1109/TCYB.2024.3356981)

3.1 Background

Deep Reinforcement Learning (DRL) allows intelligent agents to learn an optimal policy by representing the complex relationships between states and actions in an interactive game-like environment [24]. Its powerful ability to model sequential decision-making has seen DRL widely used for various applications [25, 26, 27, 28, 29]. A wide of representative works have been proposed, including proximal policy optimization (PPO) [32], asynchronous advantage actor-critic (A3C) [33], etc. When the applications of RL extend from simulated environments to real-world settings, a challenge that needs to be addressed is that the environment is often non-stationary, where the reward distributions and state transitions are constantly changing. So, an attractive topic of DRL is to explore their ability to be robust in non-stationary environments during their lifetime. However, one common assumption of standard DRL schemes is that the environment where the agent operates is dynamic but stationary. This means the probability distribution of the state transitions and the reward functions remain unchanged during the interaction. Unfortunately, such an assumption does not always hold fast in practice. In the real world, many environments are non-stationary and sometimes change quite often [7]. For example, a rescue robot may potentially venture into a cave with changing lighting conditions or navigate through unexplored terrains during task execution. Similarly, the chatbot should seamlessly adapt to changes in conversation topics, including those previously unencountered. In these non-stationary environments, the policies learned from the previous environment settings will not always work well. Accurate detection and rapid adaptation to new environmental conditions in these scenarios are crucial.

Some research [8, 9, 10] help RL agents work successfully in environments where the change points are presented because a model can learn a new policy relatively quickly after a change by simply borrowing knowledge from former well-trained policies. In practice, however, the agent is not always aware of the change points of environments.

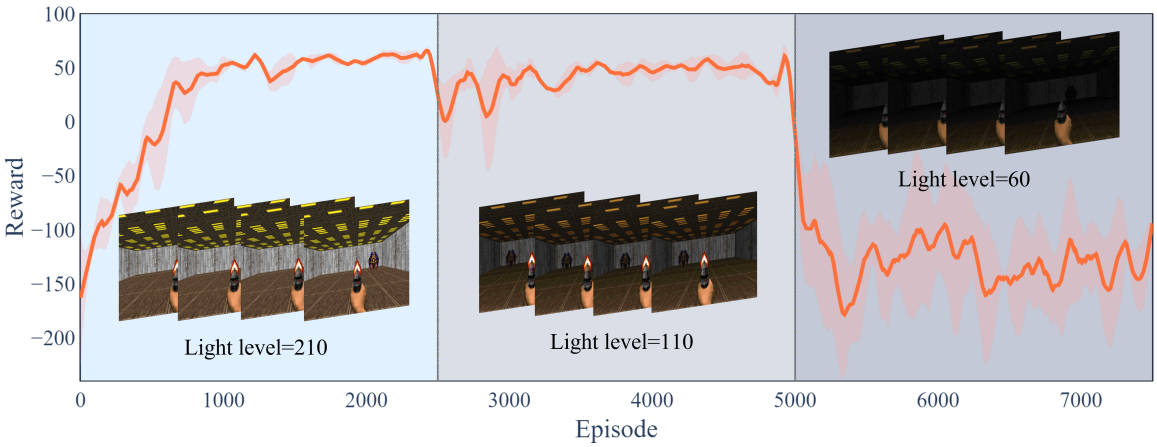


Figure 3.1: Illustration of non-stationary environments and their impact on reward changes. Three sequentially changed Vizdoom environments with different light levels will lead to a drop in reward. PPO is the baseline to obtain these rewards.

For example, a delivery drone needs to contend with changing weather and wind patterns, and a mopping robot trained on a wooden floor must be able to cope with tiled floors or the sudden appearance of a rug, a recently moved chair, or a discarded toy. These environments usually have unpredictable time-varying state transitions and reward functions. As depicted in Figure 3.1, an RL agent experiences a noticeable decline in rewards when confronted with unforeseen changes in lighting, leading to its failure in a dark environment. While solving this problem is becoming increasingly significant, only a few studies have attempted to tackle the issue. One solution is contextual detection for model-based RL [11, 62], which estimates the environmental models of different contexts but is challenging to accomplish in complex scenarios. Maintaining an environmental model is not easy for high-dimensional environments due to the limitations of sample efficiency in RL. Another method [13] for model-free RL determines environmental changes using the difference between long-term and short-term reward averages, which requires a manually set threshold value. Alternatively, Padakandla et al. [14] recognize the distribution shift of the experience stream to represent the environment change points. As shown later in experiments, these methods are vulnerable to significant and

sudden environmental changes.

This chapter presents an end-to-end model-free Reinforcement Learning algorithm with sound and stable performance in non-stationary environments with unknown change points. The key notion is that changes of the environment distribution $\mathscr{P}(a,s)$ may have two possible reasons: the change of the agent policy $\mathscr{P}(a|s)$, and the change of environment states distribution $\mathscr{P}(s)$. Our method first detects the environmental change points by examining a joint distribution that comprises the state marginal distribution, based on a Maximize Mean Discrepancy (MMD) [193], and the conditional distribution with a deep neural network distance [194]. While Padakandla et al. [14] devised a method using the marginal distribution to detect points of change in an environment, it is not always reliable. Although the marginal distribution $\mathscr{P}(s)$ change works as an excellent early-warning signal, underlying changes in the marginal distribution will lead to a gradual shift in policy focus, but those shifts might not result in a significant decrease in the performance of the policy. So, if the policy does not ultimately change, what would have been an excellent early warning signal becomes more of a false alarm. To address this problem, our idea adds the detection of policy $\mathscr{P}(a|s)$ changes using neural network distance [194].

With the change points detected, a distance-relaxed adaptation method then augments the training of a new policy for the new environment by drawing on previously well-trained policies. Our idea is to constrain the present gradient in terms of the former policy gradients with the consideration of distances from detection. It is worth noting that our goal is to stay robust in non-stationary environments by using previous policies, which differs from the aim of continual RL that seeks to resist catastrophic forgetting. With this objective, not all previously acquired policies prove to be with equal beneficial contribution; some may even have a detrimental impact on the agent's performance in the current environment. Our new gradient-constrained idea can reduce the effect of the 'bad'

previous policies and trust those sourced from the 'good' previous policies. Experiments comparing our algorithm with several alternative algorithms show that our solution accumulates the highest reward and is the fastest to adapt to new environments. This work holds strong potential for increasing the environmental applicability of intelligent agents like drones, autonomous vehicles, and underwater robots.

The contributions of this chapter include the following:

- A formal model-free Detection-Adaptation RL (DARL) framework that spans the entire pipeline of learning and clearly identifies the method's similarities and differences to existing DRL schemes. The framework demonstrates strong performance in more practical and challenging non-stationary environments.
- A change detection method to detect the change of environments accurately. Our approach takes into consideration the joint, marginal/conditional distribution, i.e., experience streams and policies. For DRL in non-stationary environments, the approach delivers not only good detection performance but also provides crucial information for policy adaptation.
- A detection-boosted, gradient-constrained adaptation method that quickly learns a new policy once a change is detected with an ability to reduce the effect of 'bad' policies. The proposed method enables DRL agents to converge quickly to the optimal reward, even in environments where the reward distributions and state transition change during interactions.

3.2 Problem Formulation

A Markov decision process (MDP) is defined by a tuple $\mathbf{M} = \{\mathscr{S}, p_0, \mathscr{A}, \mathscr{R}, \mathscr{P}\}$, where $\mathscr{S}$ is the state space, $p_0(s)$ is the starting distribution of the states, $\mathscr{A}$ is the action space, $\mathscr{R}(r|s,a)$ is the reward function $\mathscr{R} : \mathscr{S} \times \mathscr{A} \to \mathbb{R}$, and $\mathscr{P}(s_{t+1}|s_t,a)$ is the state

transition probability. A policy $\pi_\theta(a|s)$ is a distribution over actions given a state, with θ as the parameter set. When a deep neural network is used to model π_θ, θ contains the weights of the network. Let e_π denote an interaction trajectory with length T: $e_{\pi,T} = \{s_0, a_0, r_0, s_1, a_1, ..., s_T\}$ under policy π. With discount factor γ, the expected discounted reward is defined as

$$\eta(\pi) = \mathbb{E}_{e\sim\pi}\left[\sum_t \gamma^t r(s_t, a_t)\right]. \tag{3.1}$$

The optimal policy π^* is the one that maximizes the value of $\eta(\pi)$

$$\pi^* = \arg\max_\pi \eta(\pi). \tag{3.2}$$

RL employs algorithms to learn the optimal policy in the above setting. For example, a straightforward procedure for finding the optimal policy [195] is to collect e_π by interacting with the environment under policy π_θ and then to update θ using gradient information from Equation (4.1).

Standard RL assumes that the underlying MDP **M** is dynamic but fixed. When this assumption does not stand, we have the RL scheme for non-stationary environments instead. Further, when the change points are unknown, we have the following problem:

Problem 3.1. ***Reinforcement Learning in non-stationary environments with known change points.*** *Let $\{\mathcal{M}_{k=1:K}\}$ be a sequence of different MDPs with change points $\{T_1, ..., T_{K-1}\}$. An agent will sequentially interact with $\{\mathcal{M}_{k=1:K}\}$, where the change points are given. The goal of the agent is to find a sequence of corresponding policies to obtain the optimal expected discounted reward as*

$$\pi^*_{1:K} = \arg\max_{\pi_{1:K}} \eta(\pi) = \arg\max_{\pi_{1:K}} \mathbb{E}_{e\sim\pi}\left[\sum_{k=1:K}\sum_{t=T_{k-1}}^{T_k} \gamma^t r(s_t, a_t)\right], \tag{3.3}$$

where we set $T_0 = 0$.

Since we have detected the change point of the MDPs, we can quickly learn the current policy by borrowing the knowledge from former policies. However, the change

points are not always known or clear in practice. This problem is, therefore, formalized as Problem 2 below.

Problem 3.2. ***Reinforcement Learning in non-stationary environments with unknown change points***. *Let* $\{\mathbf{M}_{k=1:K}\}$ *be a sequence of different MDPs with change points* $\{T_1,...,T_{K-1}\}$. *An agent will sequentially interact with* $\{\mathbf{M}_{k=1:K}\}$ *with unknown change points, where the goal is to find a sequence of policies for obtaining the optimal expected discounted reward as*

$$\pi^*_{1:J} = \arg\max_{\pi_{1:J}} \eta(\pi) = \arg\max_{\pi_{1:J}} \mathbb{E}_{e\sim\pi}\left[\sum_{k=1:J}\sum_{t=T'_{k-1}}^{T'_k} \gamma^t r(s_t,a_t)\right], \tag{3.4}$$

where $\{T'_1,...,T'_{J-1}\}$ *are detected change points by the agent, so they might not exactly match with real change points* $\{T_1,...,T_{K-1}\}$. *Since we have detected the change point of the MDPs, we can quickly learn the current policy by borrowing the knowledge from former policies.*

Note that if the dimensions of $\mathscr{S}$ and $\mathscr{A}$ change between $\{\mathbf{M}_{k=1:K}\}$, it would be naive to detect the change points of change. Hence, with Problem 1, our focus is solely on the same state and action sets, noting that the underlying distributions and $\{\mathscr{R}_k,\mathscr{P}_k\}_{k=1:K}$ might change sequentially from $k=1$ to $k=K$. Unlike [196], we do not make an impractical assumption about which part of the MDP will change. Beyond this, our DARL scheme is model-free, which means we do not maintain an environment model like [11]. But, in keeping with other studies that have attempted to resolve Problem 1, we do make the assumption that there will be enough episodes in each environment $\mathbf{M}_k$ to ensure the policy being trained converges before the following change. If the change points $\{T_1,...,T_{K-1}\}$ are pre-defined, the problem will degenerate into a naive RL training problem with sequential switches in environments.

3.3 Methodology

Our proposed end-to-end framework is designed explicitly for non-stationary environments with unknown change points. DARL comprises two closely related components: change point detection and detection-boosted adaptation. Each is introduced here in turn.

3.3.1 Environment Change Detection

As a model-free RL agent, the scheme only includes the following elements for each time step t: a state, an action, a reward (observed from the environment), and a policy. Hence, these are the only data through which to detect a change. The collected data $\{s_t, a_t, r_t, s_{t+1}\}$ will be saved in the *experience memory* $\mathscr{M}$. Note that the a_t is determined by the policy $\pi(a_t|s_t)$.

All these data points can be understood as samples from a joint distribution $\mathscr{P}(s,a) = \mathscr{P}(s)\mathscr{P}(a|s)$, where s is an observed state and a the action taken, $\mathscr{P}(s)$ is the marginal distribution of the states, and $\mathscr{P}(a|s)$ (policy) is the conditional distribution of the actions given a state. The data is collected from an underlying and unknown MDP, and we assume the data collected from an environment that feeding back states $\mathscr{P}(s)$ with a well-trained policy $\mathscr{P}(a \mid s)$ represents an estimation of a certain joint distribution $\mathscr{P}(s,a)$, which corresponds to the current environment. Hence, a reliable and sensitive method for detecting the presence of change points in the environment should consider this joint distribution. Unfortunately, the literature contains little research of this nature. Specifically, the detection of DARL occurs over two steps. First, any changes in $\mathscr{P}(a|s)$ are detected from a distance function in the neural network. Subsequently, changes in $\mathscr{P}(s)$ are evaluated to ascertain whether a change has genuinely occurred.

Policy detection We detect the change of policy during the interaction with an environment by considering the deep neural network representation in the literature [194].

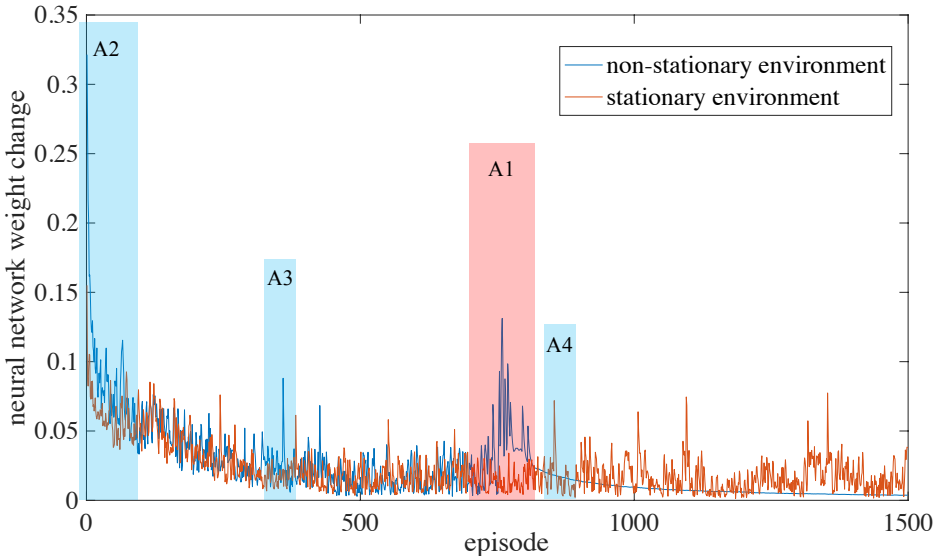


Figure 3.2: When the environment changes, a notable drift is observed in the distance between adjacent network weights. The results are derived from utilizing PPO in the Cartpole environment, with a change point at 750.

Consider a neural network with L layers, we denote the network parameters as $W = (W_1, W_2, \ldots, W_L)$. After one update step, W will have a perturbation $\Delta W = (\Delta W_1, \Delta W_2, \ldots, \Delta W_L)$.

The relative difference between two sequential adjacent deep neural networks' parameter W with the same structure is

$$d^{(c)} = \prod_{l=1}^{L} \left(1 + \frac{\|\Delta W_l\|_2}{\|W_l\|_2}\right), \tag{3.5}$$

where L is the network depth, $\|\cdot\|_2$ denotes the Euclidean norm, and $d^{(c)}$ denotes the distances within the conditional distribution (policy). It is worth noting that in neural networks, the parameters of well-trained convolutional kernels follow a Gaussian-like distribution, while fully connected layers and batch norm layers do not conform to a well-defined distribution [197]. However, similar distribution characteristics have not been observed for other learnable parameters. Consequently, we consider the policy as a whole and use Euclidean distance to measure its difference rather than relying on distribution-based metrics like KL divergence. As [194] outlines, this distance has a

useful property that it links with the network gradient change through

$$\frac{\|g_l(W+\Delta W)-g_l(W)\|_F}{\|g_l(W)\|_F} \leq d^{(c)}-1, \tag{3.6}$$

where g_l is the gradient of the loss function $\mathscr{L}$ with respect to the parameter matrix $W_l (l=1,\ldots,L)$.

The loss function used in this chapter follows PPO [32], which denotes as

$$\mathbf{L} = \min\left(\frac{\pi(a_t \mid s_t)}{\pi_{\text{old}}(a_t \mid s_t)}\hat{A}_t(s,a), g\left(\epsilon,\hat{A}_t(s,a)\right)\right), \tag{3.7}$$

where π is the current policy with parameters W, π_{old} is the policy before the update. $\hat{A}_t$ is an estimator of the advantage function at timestep t, and ϵ is a hyperparameter. In the implementation, the specialized clipping in the objective function removes incentives for the new policy to get far from the old policy.

$$g(\varepsilon,A) = \begin{cases} (1+\varepsilon)A & \text{if } A \geq 0 \\ (1-\varepsilon)A & \text{if } A < 0 \end{cases} \tag{3.8}$$

Equation (3.6) shows that the distance $d^{(c)}$ between the weights in Equation (3.5) is the upper bound of the change in gradients due to the disruption of weights. This means that a large $d^{(c)}$ might lead to greater gradient change of the neural networks (policies). To further confirm this significance, we want to find the outlying point of the change prior to the current time step t within a sliding first-in-first-out window of weights

$$\left\{W_{t-(S-1)},\ldots,W_{t-1}\right\} \Rightarrow \left\{d^{(c)}_{t-(S-1),t-(S-2)},\ldots,d^{(c)}_{t-2,t-1}\right\}, \tag{3.9}$$

where S is the size of window, $\{W\}_{t-(S-1)}^{t-1}$ is a window of policy weights, and $\{d^{(c)}\}_{t-(S-1),t-(S-2)}^{t-2,t-1}$ is a window of corresponding relative distances. As stated in Equation (3.5), we use the pair $\langle W_{t-(S-1)}, W_{t-(S-2)}\rangle$ to compute $d^{(c)}_{t-(S-1),t-(S-2)}$. According to the three-sigma rule [198], 99.7% of the values in the window should obey normal distribution and be concentrated in the $(\mu-3\sigma,\mu+3\sigma)$ interval, where μ and σ denote the mean and standard deviation in a window. Hence, values beyond the range in this window should be

considered as outliers. Even in the case of a non-normal distribution, there is also at least an 88.8% probability that it will be within $(\mu - 3\sigma, \mu + 3\sigma)$ according to Chebyshev's inequality [199]. Therefore, we consider this test method to be reliable.

To show the accuracy of this detection intuitively, Figure 3.2 gives a demonstration where we can see that there is a significant and immediate change on the variable $d^{(c)}$ when a sudden environmental change occurs at and only at episode 750 in area A1. We can also observe that $d^{(c)}$ quickly decreases after the change point. The reason is that as a deep neural network is trained with more incoming data, the weights will quickly converge to some point with a gradually limited relative change. This is a phenomenon consistent with a neural tangent kernel where we see small changes in the weights after several training steps [200, 201]. This phenomenon reassures us to use the above outlier judgment to detect policy change.

Episodic detection As also shown in Figure 3.2, significant changes in neural network weights can be seen before and after episode 750, like A2, A3, and A4. The reason behind the A2 change is that, when trained on incoming data at the beginning of the optimization process, the weights of the neural networks are normally randomly initialized and will change severely before converging to a relatively stable point. The mutation points are seen in A3 and A4, but they are the normal fluctuation of neural network training. They are not likely due to environmental change. Hence, the detection method will report many false change points if we rely only on detecting (policy) conditional distribution changes. Therefore, it is essential to confirm each change in the marginal distribution. Our idea is that the marginal distribution $\mathscr{P}(s)$ should have a significant drift before and after a real environment change point, so if it does, it can be used as additional confirmation. To this end, distribution drift detection is vital to determine the source of the disturbance on network weights, i.e., whether it comes from training or environmental change. The distribution drift is detected using the maximum mean

discrepancy (MMD) test [193] and based on the two collected interaction experience sets before and after the detected change point. With this test, one can judge whether two sample sets are from the same distribution with an associated p-value and output a distance between two distributions

$$\begin{aligned} d^{(m)} = & \frac{1}{n(n-1)} \sum_i \sum_{j \neq i} \phi(s_{0,i}, s_{0,j}) - \frac{2}{n^2} \sum_i \sum_{j \neq i} \phi(s_{0,i}, s_{1,j}) \\ & + \frac{1}{n(n-1)} \sum_i \sum_{j \neq i} \phi(s_{1,i}, s_{1,j}), \end{aligned} \tag{3.10}$$

where ϕ is a kernel (we use RBF kernel here), n is the size of two sets, (m) denotes the distance of the marginal distribution, and $s_{0,i}$ is i-th experience of $\{s_0\}$.

By jointly considering the policy and episodic experience changes, we can accurately detect changes in the environment.

3.3.2 Policy Adaptation

With the detected change points, we carry out an adaptation procedure to augment the training of a new policy for a new MDP with the help of former well-trained policies. Directly applying existing adaptation ideas of DRL with known change points is not good enough. The reasons are: 1) our detected change points might not be identical to the ground-truth change points. An inaccurate estimation, such as a delay, leads to inaccurate environment change point identification. If previous knowledge is leveraged directly and equally to adapt to the current environment, incorrect environment estimation can lead to confusion, such as mixing up two different environments that should be identified as distinct entities. The policy learned from such a mixed environment may lead to abnormal behavior and a 'bad/negative' impact on the current one. 2) Some former policies trained in past MDPs might differ greatly from the current MDP. Uniformly considering the contribution of different MDPs may lead to performance reduction in the current environment. So, equally aggregating the contributions from all past policies is not reasonable. We need a technique to reduce the effect of such a 'bad/negative' policy.

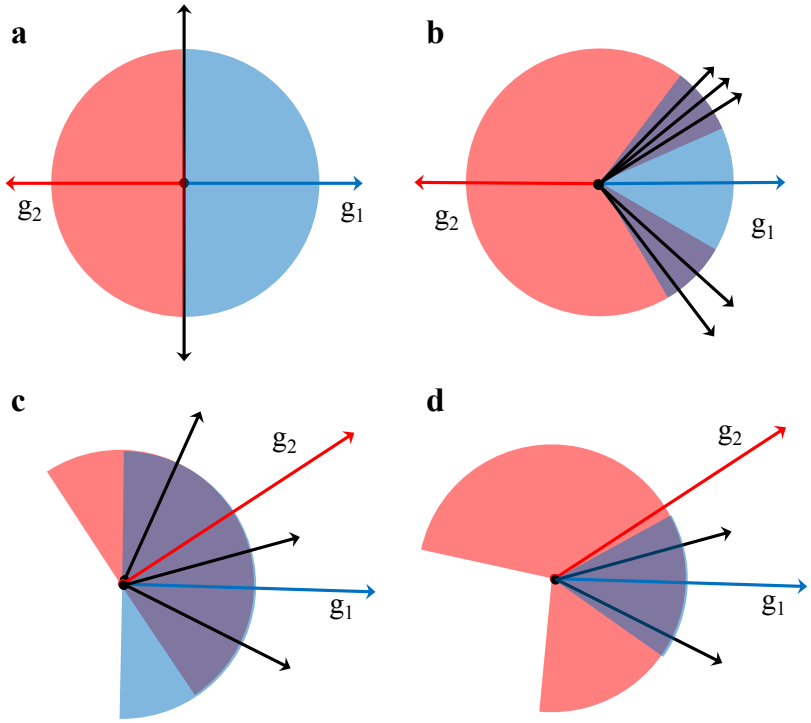


Figure 3.3: Illustration of adaptation control differences between GEM and our method in two g_k situations, where **a** and **c** illustrate GEM; **b** and **d** illustrate our method; g_1 (blue) is with small distance to current MDP and g_2 (red) is with large distance; the (purple) overlap is the solution space for $\tilde{g}$; black vectors are possible solutions.

Thus, we propose a distance-relaxed adaptation method that includes the distances defined in Equations (3.5) and (3.10). Inspired by Gradient Episodic Memory (GEM) [15], the distance-relaxed cosine similarity between the gradients of the former policies and the new policy is applied to constrain their angles within a feasible area. In this section, we use the term "task" k to refer to the learning process of RL agents in the corresponding environments, and in each environment, an optimal policy π_k with gradient g_k will be found. The advantage of this distance-relaxed cosine constraint is illustrated in Figure 3.3, where two former policies π_1, π_2 with gradient g_1, g_2 exist and with different distances to the current MDP ($d_1^{(c)}$ is small and $d_2^{(c)}$ is large). In the first row of Figure 3.3, we illustrate an extreme situation where g_1 and g_2 have exactly inverse directions. Under GEM, there are only two possible solutions (black

vectors) because the contributions from g_1 and g_2 are equally treated and neutralized, as illustrated in Figure 3.3(a). However, there is a larger solution space (purple area) under our method because we allow extending solution space constrained by g_2 (red area) because it has a larger distance from the current MDP. That is to say, the contribution from the 'bad' g_2 is reduced while the contribution from the 'good' g_1 is enhanced, as illustrated in Figure 3.3(b). The second row of Figure 3.3 shows a general case where g_1 and g_2 is neither inverse nor coincident but with an angle. The solution space (purple area in Figure 3.3(c)) of GEM is symmetric for both g_1 and g_2, but the one (purple area in Figure 3.3(d)) from our method has a preference to be closer to g_1.

Additionally, we apply a distance-relaxed upper bound using Equation (3.6) to constrain the Euclidean distance of the gradients between the former policies and the new policy within a feasible area. Thus, the complete optimization goal, with constraints, is

$$\begin{aligned} \min_{\tilde{g}} \quad & -\cos(g, \tilde{g}) \\ \text{s.t.} \quad & \cos\langle \tilde{g}, g_k \rangle \geq d_k^{(m)} \text{ for all } k < t \\ & \|\tilde{g} - g_k\|_2 \leq (d_k^{(c)} - 1)\|g_k\|_2 \text{ for all } k < t, \end{aligned} \tag{3.11}$$

where g is the current gradient vector trained from a standard policy gradient, $\tilde{g}$ is the closest gradient vector satisfying all the constraints, g_k is the gradient vector of task k (in the implementation, we calculate g_k using the k_{th} policy on the current data $\mathcal{M}_c$), and t is the number of observed tasks now. $d_k^{(m)}$ is a normalized scalar for each task k within $[-1, 0]$. Note that this scope ensures that the solution space constrained by cosine distance with all g_k is not empty. To ease the optimization, we further fix the norm of $v = \|\tilde{g}\| = \|g\|$ and assume $\tilde{g} = hv$, and then our problem becomes,

$$\begin{aligned} \min_{h} \quad & -\bar{g}^T h \\ \text{s.t.} \quad & h^T g_k \geq d_k^{(m)} \|g_k\| \text{ for all } k < t \\ & \|vh - g_k\|_2 \leq (d_k^{(c)} - 1)\|g_k\|_2 \text{ for all } k < t, \end{aligned} \tag{3.12}$$

where $\bar{g}$ is the normalized vector of the current gradient trained from a standard policy gradient, and h is the normalized vector of $\tilde{g}$, which is the closest gradient vector satisfying all the constraints in Equation (3.11). This is a standard Linear program with linear and quadratic constraints, and we can solve it to obtain optimal h^* and recover the optimal $\tilde{g}^* = h^* \nu$ easily. It is noted that the problem this chapter focuses on is different from traditional continual learning. We aim to continuously adapt to new environments quickly instead of combating catastrophic forgetting. To address this problem, we aim to figure out the similarities between current policies and previous policies by considering their behavior in the current environment. Therefore, instead of following the typical continual learning approach where the gradient vector g_k is computed using the current policy π_c on all previous data $\{\mathscr{M}_k\}_1^{t-1}$, we adopt a different strategy. We calculate the gradient vector g_k using the previous well-trained policy on the current data $\mathscr{M}_c$. Then we employ neural network distance $d^{(c)}$ and episodic experience distance $d^{(m)}$ as constraints. This approach enables us to determine how much the previous ones should impact the current gradient.

3.3.3 Detection-Adaptation RL

Our algorithm Detection-Adaptation RL (DARL) is illustrated in Algorithm. 7. First, we interact with the environment using the current policy π_c to collect data $\mathscr{M}_c$ and compute the gradient g. Then, we calculate the gradient distance $d^{(c)}$ between the current policy π_c and the previous policy, as well as the distance $d^{(m)}$ between the collected data. If a change point is detected, the gradient g will be optimized and overwritten according to Equation (3.12), and the current policy and data memory will be stored. Otherwise, the policy will maintain the original gradient. With the optimized gradient, the policy will adapt to the new environment with the enhancement of previous knowledge. The core of DARL adaptation is to leverage previously learned knowledge depending on the

Algorithm 1 Detection Adaptation RL (DARL)

Initialization: Current policy π_c, current experience memory $\mathscr{M}_c$, previous policies $\{\pi_k\}$, previous memories $\{\mathscr{M}_k\}$.
repeat
  while not done **do**
    Collect $\{s_t, a_t, s_{t+1}, r_t\}$ using π_c and save to $\mathscr{M}_c$.
  end while
  Compute the gradient g of π_c using $\mathscr{M}_c$.
  Calculate anomaly of previous neural networks' distance $d^{(c)}$ using Equation (3.5).
  Calculate the neighbouring experience distribution distance $d^{(m)}$ using Equation (3.10).
  if Anomaly detected in $d^{(c)}$ sequence and neighbouring distribution shifts **then**
    Optimize g according to Equation (3.12) to find an optimal solution $\tilde{g}^*$.
    Update $g \leftarrow \tilde{g}^*$.
    $\{\mathscr{M}_k\} \leftarrow \{\mathscr{M}_k\} \cup \mathscr{M}_c$, $\{\pi_k\} \leftarrow \{\pi_k\} \cup \pi_c$.
  end if
  Update π_c along gradient g.
until Training ends.

similarity between different environments and policies through DARL detection. DARL helps identify environmental change points and mitigate the adverse effects of potential sudden environmental changes on policy learning.

3.4 Experiments and Analysis

Our experiments aim to answer several questions: 1) Does DARL outperform other methods in cumulative reward? 2) Can DARL detect the change points accurately? 3) Is the DARL adaptation strategy better than following RL updates? 4) Does joint detection outperform sole detection? 5) Is there any factor that influences DARL performance?

3.4.1 Experiment Setups

Environments: We perform our experiment in the following environments with change factors: Gym [202] with dynamic transitions, MiniGrid [203] with time-varying obsta-

Table 3.1: Settings for the Gym environment changes

Stage Number	CartPole(gravity, mass_pole, force_mag)	LunarLander(wind_level)
0	(9.8,0.1,10)	(0)
1	(20,10,10)	(0.25)
2	(9.8,0.1,30)	(0.5)

cles, and ViZDoom [204] with changing light levels and ceiling textures. The detailed environment settings are listed in Table 3.1 and Figure 3.4.

For each environment, two change points are simulated. It is important to emphasize that all environment-changing variables are not observable to the agent, and the agent has no access to them when the environment changes.

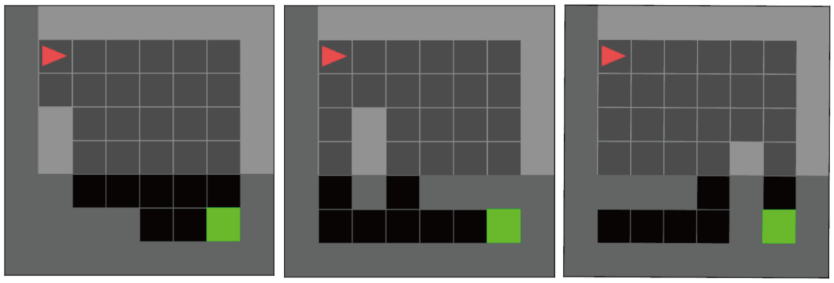
We used the following environments:

- MiniGrid [203] is a 2D maze environment with an NxM grid of tiles, and the agent can pick up and carry exactly one object. We put extra *obstacles* in rooms to simulate change.

- ViZDoom [204] is an image-based environment that allows the agent to play Doom using a screen buffer of the fps scenario. We simulate the environmental change by varying *light* and *texture*.

- Gym [202]: We simulate the environmental change by varying *gravity*, *mag force*, and *mass* in Cartpole. We involve an extra variable to simulate the wind in LunarLander.

Comparative methods: Our target of experimental evaluation is to 1) verify the effectiveness of our detection; 2) evaluate the effectiveness of our adaptation; 3) show the overall performance of our DARL algorithm. Proximal policy optimization (PPO) [32] was used as the base algorithm for all following comparative methods:



(a)

(b)

Figure 3.4: We change the light level and texture of the ceiling in the shooting game ViZDoom(upper) and the position of obstacles in the maze environment MiniGrid(bottom), respectively, during training.

- Standard PPO without any detection and adaptation;
- Standard PPO with Gradient Episodic Memory (GEM) [8] gradient-constrained adaptation and shown change points.
- CRL-Unsup [13] with both detection and adaptation.
- Online parametric Dirichlet change point (ODCP) algorithm [14, 70]; This algorithm does not have an adaptation component so we use standard restart procedure.
- Policy Consolidation (PC) [81], which simultaneously remembers the agent's policy at a range of timescales and regularises the current policy by its own history to

Table 3.2: The implementation parameters of our method, DARL.

	Cartpole	LunarLander	MiniGrid	VizDoom
Network Layer	2*Linear	2*Linear	3*Linear	4*Conv
Network Width	64	64	32	64
Episode	1500	1500	45000	7500
Batch Size	128	128	1024	1024
Learning Rate	0.001	0.001	0.001	0.001

improve the ability to learn. Hence, we evaluated the method as it is without detection and adaptation.

Note that all algorithms are model-free and with the same setting unless otherwise specified, like the policy neural network structure, learning rate, window size, etc.

Evaluation metrics: Any RL method's goal is to obtain a high accumulative reward and continuous interaction between agents and environments. In all experiments, the most important metric is how much/high rewards each method can get. We actively detect the possible changes because we believe we can obtain more rewards if we can quickly detect and then adapt to the changed MDP.

For detection, we use F1 Score $\mathrm{F}_1 = \frac{2*P*R}{P+R}$ as the metric, where P is precision and R is recall.

Implementation Details The hyper-parameters we used in our experiment implementation are listed in Table 3.2.

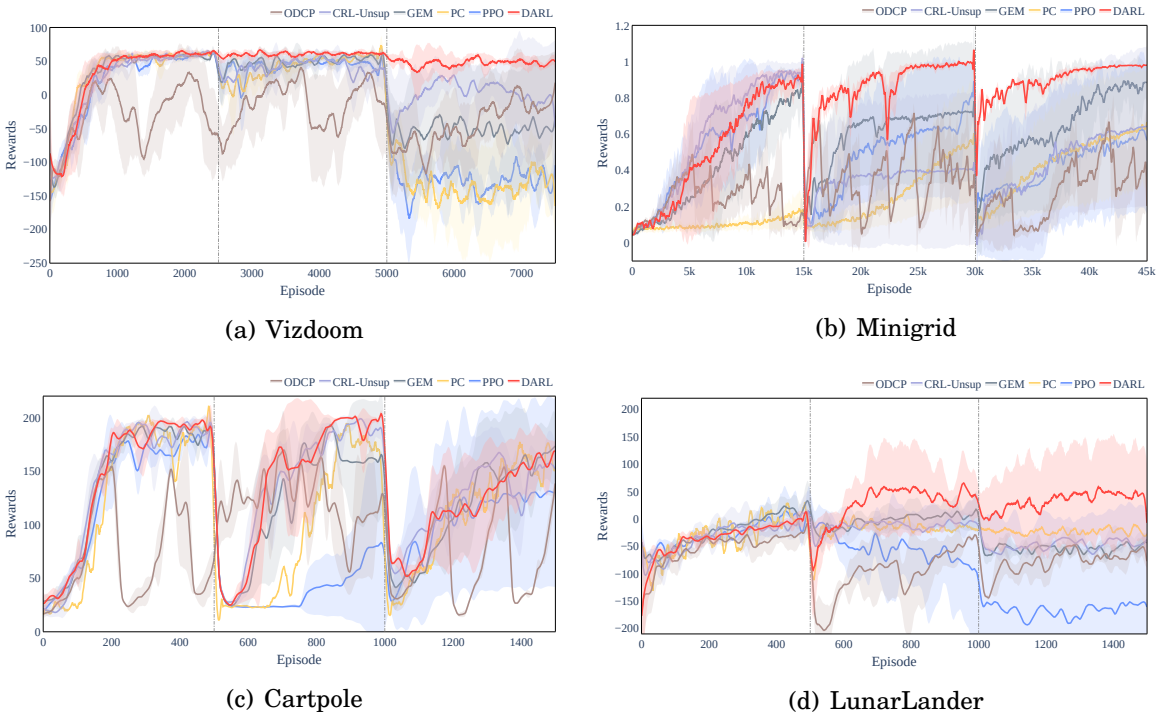
3.4.2 Results

We present a comprehensive comparison of reward curves and detection accuracy, providing insights into the relative performance of DARL. Figure 5.4 illustrates the results

in MiniGrid, VizDoom, and the relatively simpler Gym environments. We compare performance using the training reward curve with shadowed 95% confidence intervals.

In MiniGrid and VizDoom, as well as in simpler environments like Cartpole and LunarLander, the DARL method not only identifies change points with high accuracy but also maintains a consistently high reward curve post-change, suggesting a robust adaptation to the new environment dynamics. Notably, DARL's performance in Cartpole remains relatively stable across episodes, while in LunarLander, DARL demonstrates superior recovery after initial performance dips at change points compared to other methods. In high-dimensional, complex environments such as VizDoom, the adaptability of DARL is evident as it outperforms other methods, particularly after the environment changes, maintaining higher reward consistency. This is crucial, considering the complexity and the visual rendering that VizDoom presents. In the case of the simpler environment Cartpole, while the rewards for DARL are not as high as some methods initially, the performance after the change points shows DARL's robustness, suggesting that it doesn't overfit to the initial environment setup and can maintain good performance even when the environment dynamics are altered. The results in LunarLander further solidify DARL's position as a method that can quickly adapt to new environments. Its reward curve post-change is not only higher but also more stable compared to other methods, indicating a better understanding of the environment's dynamics. It's also worth noting that the confidence intervals of DARL are tighter in environments like Cartpole and LunarLander, which implies that DARL's performance is more consistent across different trials. This consistency is less pronounced in more complex environments like VizDoom, which is understandable given the increased complexity and variability in such environments.

Overall, DARL showcases an excellent balance between detection accuracy and reward optimization across a variety of environments, confirming its versatility and



(a) Vizdoom

(b) Minigrid

(c) Cartpole

(d) LunarLander

Figure 3.5: Evaluation on detection accuracy and average reward in four environments. The environment changes at episode {500, 1000} in Cartpole and LunarLander. In MiniGrid, the environment changes at episode {15000, 30000}, and in VizDoom, the environment changes at episode {2500, 5000}. Our method, DARL, achieves the highest performance among the baselines.

efficiency as a method for change point detection in reinforcement learning settings.

We detailed discuss the experiment results from two main aspects:

Detection accuracy The detection results are given in Table 3.3. Firstly, ODCP detects environmental change points by testing the Dirichlet likelihood of the experience tuples, where any discrete or continuous data will be transformed into compositional data. The results show that ODCP reports change points frequently soon after the reward converges. In contrast, our method can effectively filter out these false detections because our method detects the joint distribution of states and actions. Secondly, CRL-Unsup introduces the ratio between long-term and short-term rewards as the detection indicator. The method will report a change when the ratio is larger than a (manually) given thresh-

Table 3.3: The F1 score of detection methods.

	ODCP	CRL-Unsup	DARL
CartPole	0.67	1.0	1.0
Lunar-Lander	0.5	0.67	0.8
MiniGrid	0.4	1.0	1.0
VizDoom	0.4	1.0	1.0

Table 3.4: The detection performance of policy detection and episodic detection, respectively.

	Policy Detection	Episodic Detection	Full Detection
Vizdoom	0.67	0.57	1
Minigrid	0.4	0.57	1
LunarLander	0.46	0.4	0.8
Cartpole	0.36	0.57	1

old. The detection accuracy is close to ours, as shown in the tables. However, we want to highlight that 1) the threshold value of CRL-Unsup highly relies on prior knowledge and manual selection, but ours is without any manually adjusted hyperparameters; 2) CRL-Unsup is a reward-based method which is straightforward, but sometimes we cannot timely obtain the reward from the environments then these reward-based methods will be delayed as well, and the rewards from two MDPs are sometimes not comparable so the reward may remain unchanged but already not good enough in the new environment. Our method focuses on the joint distribution of states and actions, which is more robust and applicable than using reward only.

Adaptation performance At the first change point, the convergence speeds of the three methods are approximately close without considering the effect of detection delay, with CRL-Unsup achieving the fastest adaptation after episode 500 in CartPole and

Table 3.5: The correct detected change points. The point where the actual change points are $\{500, 1000\}$.

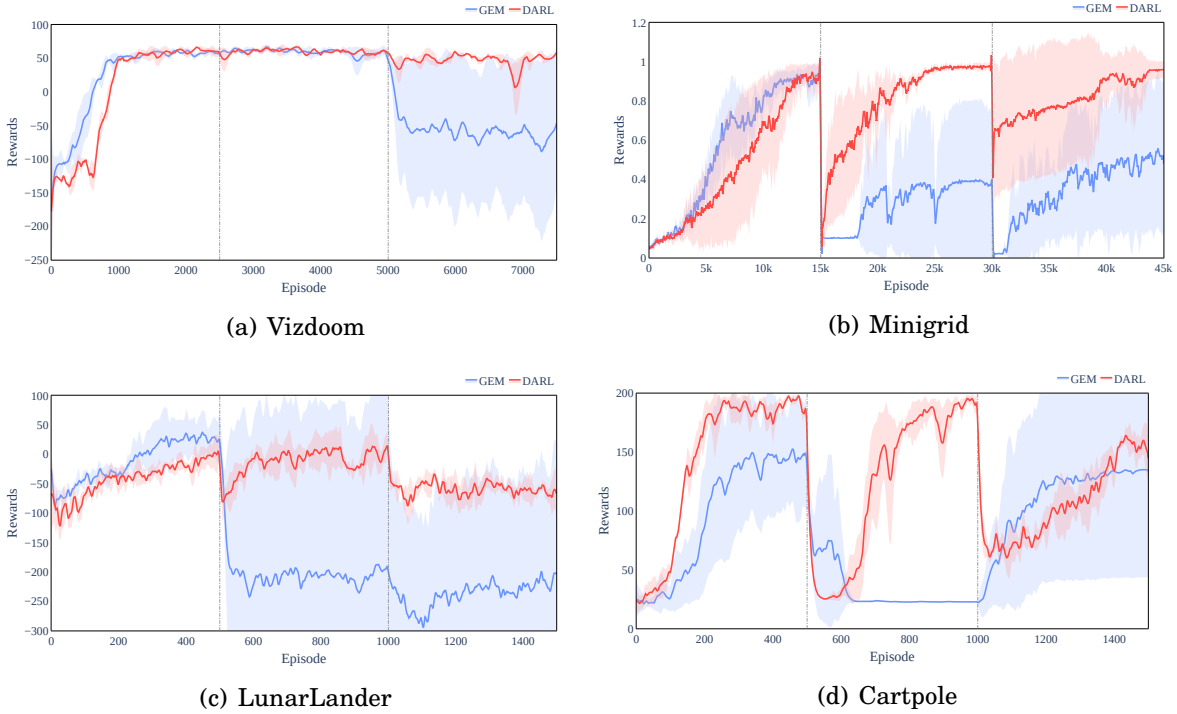
	CRL-Unsup	DARL
CartPole	{502,1003}	{508, 1021}
Lunar-Lander	{504, 1017}	{515,1022}

ours achieving the fastest adaptation after episode 1000 in CartPole. In the image-based environments VizDoom and MiniGrid, our method has the fastest convergence speed among all methods. Additionally, to study the influence of a 'bad' policy, the reward curves in Figure 3.6 show that the traditional GEM with relatively strict constraints is significantly affected by a precious 'bad' policy, while DARL does not. The impact will be discussed in detail following. In summary, our method has a faster convergence than other baselines in all environments.

3.4.3 Ablation Studies

In this section, we systematically dissect the individual components of our model to evaluate their respective contributions and understand their impact on the overall performance.

Joint detection. To evaluate the performance of change detection, we analyzed two detection schemes individually. Figure 3.8 shows the detected point of using policy detection and episodic detection separately. Table 3.4 shows the F1 score of each separate detection module and the full detection method. Upon individual assessment, it was observed that neither policy detection (represented by yellow marks) nor episodic detection (denoted by green marks) could, in isolation, accurately identify the actual change points (illustrated by vertical bars). It is noted that our method only makes the decision (marked as blue triangles in the figure) when both episodic changes and policy changes are detected



(a) Vizdoom

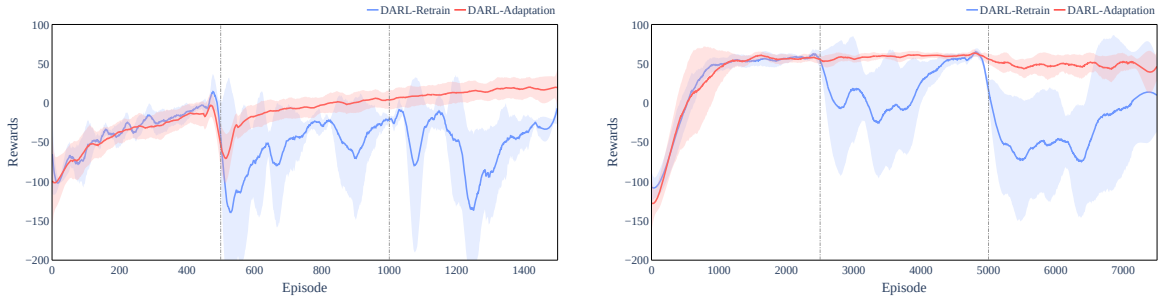
(b) Minigrid

(c) LunarLander

(d) Cartpole

Figure 3.6: Evaluation on adaptation method in four environments. The environment changes at episode {500, 1000} in Cartpole and LunarLnader. In MiniGrid, the environment changes at episode {15000, 30000}, and in VizDoom, the environment changes at episode {2500, 5000}. From stage 2, GEM is obviously affected by the 'bad' policy, while ours is affected less.

simultaneously within a very short timeframe (e.g., three episodes). These joint detections are visually encoded as blue triangles in Figure 3.8, signifying the points where both detection methods concur on the presence of a change point. Our joint detection approach yielded a notable increase in the fidelity of change point detection, which is quantitatively captured in Table 3.4. The table is anticipated to showcase enhanced F1 scores for the combined method relative to the individual detection modules. The F1 score, a harmonic mean of precision and recall, is a robust measure of accuracy. We can claim that using only one detection method is inaccurate, while the combined method can filter most incorrect estimations.



(a) LunarLander (b) Vizdoom

Figure 3.7: This figure depicts the training results using only detection without policy adaptation. The complete DARL with detection and adaptation serves as a comparison. Similarly, DARL also undergoes retraining after detecting change points.

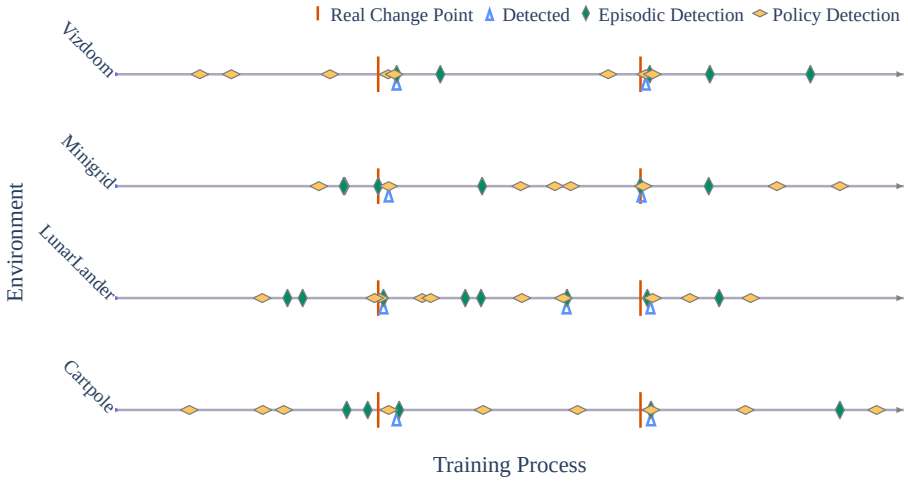


Figure 3.8: This figure illustrates the results using episodic detection and policy detection separately. The horizontal axis represents the entire training process, with annotations based on the timeline. Yellow markers indicate the change points detected by policy detection, while green markers represent those detected by episodic detection. DARL makes change decisions on the points marked by blue triangles.

Impact between tasks. Notably, DARL adapts faster because our method has the ability to filter out the 'bad' effect from previous policies. This effect of the 'bad' policy can be seen in Figure 3.6 more obviously, where we have added a random policy at the beginning phase of the interactions. This policy is considered a well-trained policy in a given MDP, which may differ from the following MDPs and contribute 'bad' information to any following policy training. The results show that our algorithm can converge faster

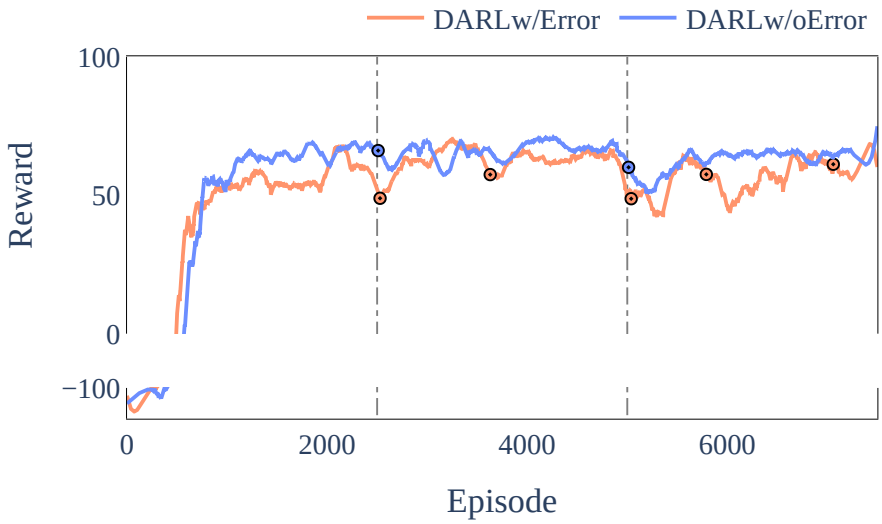


Figure 3.9: The figure shows the reward curve of DARL using different detection windows. The blue line indicates results with all change points detected correctly, and the orange line shows the results with some incorrect change detections. The circle with a dot indicates the change point detected.

than GEM, the performance of which is heavily affected by the initial policy, but ours is affected less. It should be highlighted that in stage 1, there is only one constraint, so the tighter constraint in GEM can help it find a better solution faster than ours because the gradient scope is relatively narrow, so it is faster to find a better one than ours within this (relatively) small scope. However, a challenge arises as the number of change stages increases: the gradient scope narrows exponentially, resulting in failure due to the absence of suitable solutions for the target policy. That is why the performance of GEM drops at stages 2 and 3. In contrast, the loose constraint causes slightly slower convergence at the beginning, but as the number of change points (i.e., constraints) increases, our method converges faster and better. This feature makes DARL more robust in significantly different environments as we do not always need the guidance of irrelevant previous environments and policies.

Impact of incorrect detect points. During the process of detecting environmental changes, a crucial aspect to consider is whether false detections could lead to significant repercussions. As Figure 3.9 shows, when an incorrect point is detected (the second point on the orange line), the reason may be the agent explored a new state space that was never seen, which is far from the well-trained one. When a false alarm is detected, an extra constraint would be added to policy adaptation, and the reward would drop slightly and soon go back to a converged level. The reason is that DARL calculates the policy and episodic distance at each update step while adapting. If the underlying environment remains stationary, the disparities in experiences encountered at each step will not be significant at each step for a long time. Likewise, any transient occurrences of local outliers during neural network updates will be short-lived. Subsequently, as the distance diminishes, the optimization process will converge toward the vicinity of the original gradient direction. Furthermore, our relaxed constraint can induce the influence of significantly different environments. If there were some missing points, for instance, due to a smaller detection window size, the impact would be to slow the adaptation down for the new environment. The reason is that when we optimize the gradient in Equation (3.12), not every g_k is identified, which would cause our optimization problem to search for the optimal solution in a larger feasible space. Missing change points, which lead to the use of fewer constraints, do not render the adaptation ineffective; they may decelerate the adaptation process.

Policy adaptation. To evaluate the performance of policy adaptation, we involved another two approaches as comparison methods: CAGrad [83] and PCGrad [82], which can also deal with gradient transfer and leverage as comparison methods. PCGrad projects the conflict gradient onto the normal vector of the other gradient. CAGrad finds the best update vector within a ball around the average gradient that maximizes the worse local improvement between task 1 and task 2. Figure 3.10 shows the ablation study

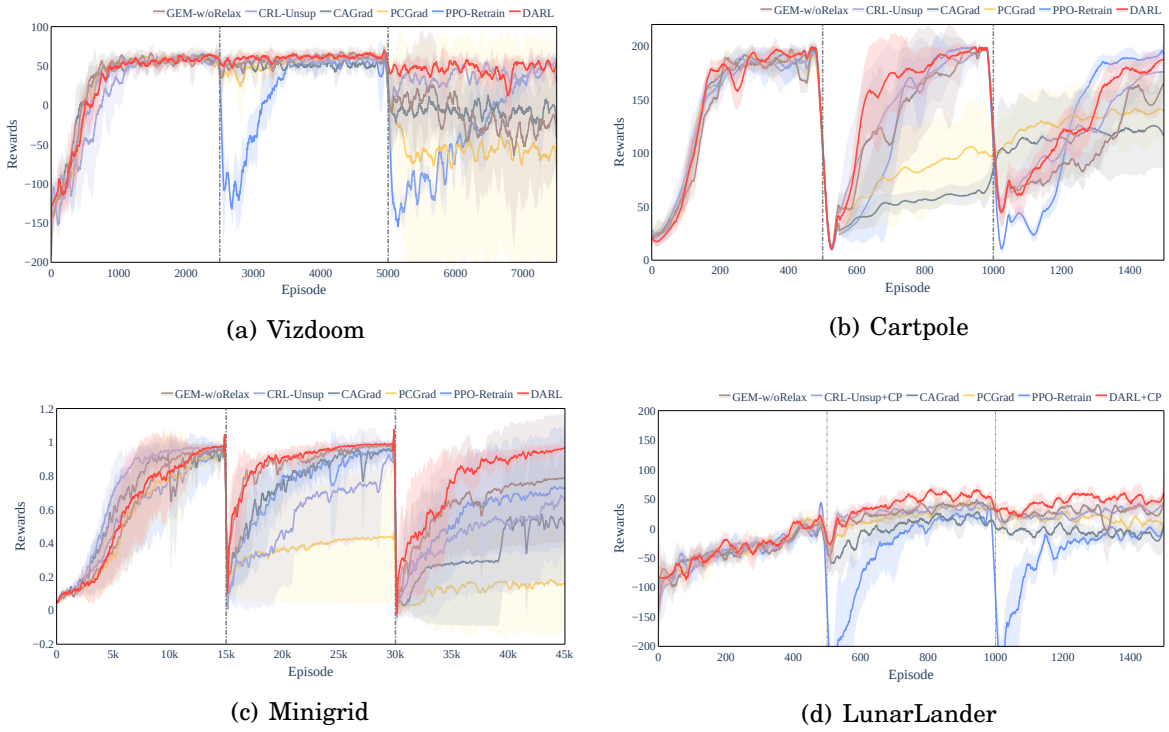


Figure 3.10: This figure shows the evaluation results on the policy adaptation method w/o change detection module in four environments, where the change points are given to all methods.

of the policy adaptation module. All of the methods are given change points in advance, and we can observe that our method, DARL got the highest reward and convergence speed overall comparison methods. Especially in Vizdoom, after the first change point, all the adaptation methods have close reward curves. However, following the change point, DARL demonstrates notably superior performance to other methods. The reason is that CAGrad and PCGrad aim to solve multi-task problems, which maintain consistency among gradients in the subspace of tasks, thereby finding a compromise that satisfies multiple tasks simultaneously. In our setting, where the environment is constantly changing and unpredictable, the goal is to leverage knowledge gained from similar past experiences to perform better in the current environment rather than striving to maintain performance across all previously encountered environments. Our adaptation

method can maintain the adaptability required to handle the dynamic and evolving nature of the environment, which may not always follow a consistent pattern. To ensure performance in the current environment, the policy is optimized based on the specific context and similarity to past experiences. Further, Figure 3.7 shows the result using detection with policy adaptation and retraining. We can observe that using the policy adaptation module obtains a faster convergence speed and higher average reward than not using it. This result further demonstrates the effectiveness of using DARL adaptation, which adapts to new environments faster than training directly with RL objectives.

3.4.4 Further Analysis

In this section, we have analyzed some special situations, the impact of parameters, and limitations.

Change frequency. We assume the change occurs after the policy converges, but what happens if the environment is dramatically unstable? We compare the following experiment, as shown in Figure 3.11. As Figure 3.4 shows, the environment of 3 change points follows the pattern of the stage {1, 2, 3}; the 5-change-point environment follows the pattern of the stage {1, 2, 1, 2, 1}; the 10-change-point environment follows the pattern of the stage of a randomly generated sequence {3, 2, 1, 3, 2, 3, 1, 3, 1, 2}. This result shows that the obtained reward will decrease for any method if the environment changes frequently. However, our method gets the highest reward in a volatile environment. Our method, DARL, exhibits superior adaptability to baselines, particularly in the face of high-frequency change scenarios, as represented by the 10-change-point environment. DARL's efficacy in these settings can be attributed to its agile policy update mechanism, which facilitates prompt adaptation to new environmental states. In environments with fewer change points, DARL maintains robust performance, underscoring its ability to leverage previous policies effectively. This adaptability is contrasted by the performance

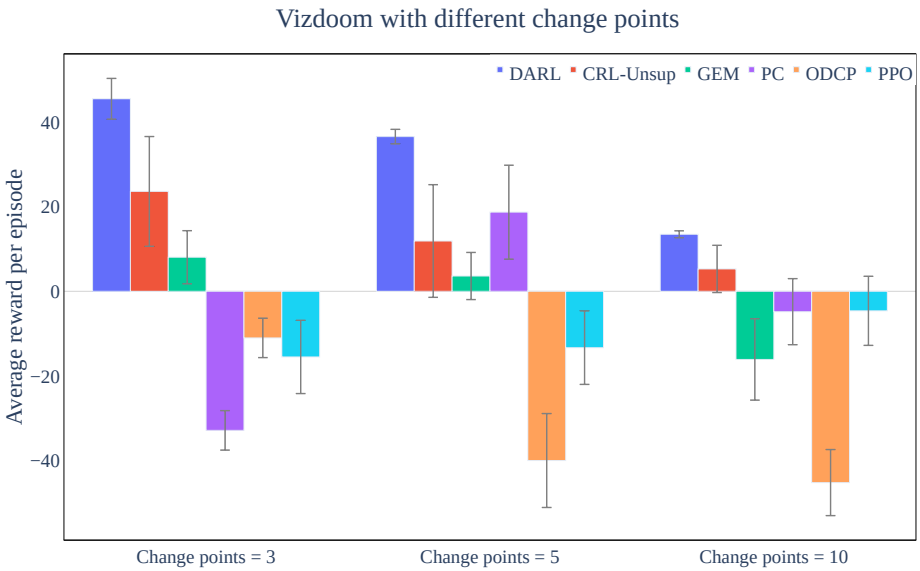


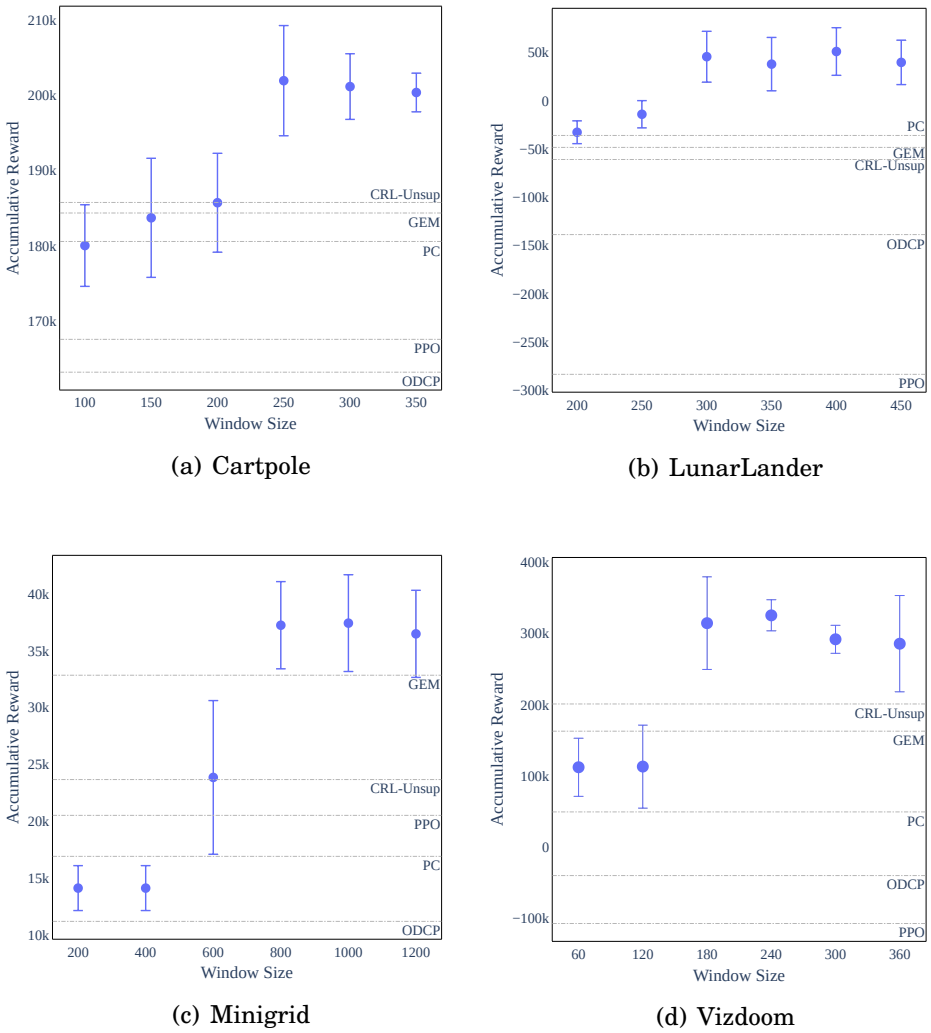
Figure 3.11: The average rewards in a changing ViZDoom environment with different amounts of change points.

of the GEM algorithm, which underperforms in the context of frequent changes due to its more stringent constraints. Policy consolidation performs better in slightly changing environments than in other environments. This is because the previous policy is recorded at each updating step, which helps to adapt quickly when the change is slight. Overall, the frequency of environmental changes is a pivotal factor influencing the performance of reinforcement learning algorithms. The results in Figure 3.11 substantiate that increased change frequency generally diminishes the average rewards across all tested algorithms. DARL, however, mitigates this effect by efficiently updating its policy, hence sustaining higher rewards even as the change frequency intensifies. This capability is crucial for applications in which the environment is subject to frequent and unpredictable variations.

Detection window. The role of the detection window size as a hyperparameter is pivotal in the performance of DARL, as discussed in section 3.3. The interplay between window size and detection accuracy is a delicate one; an appropriately sized window is imperative for the algorithm to capture enough information to discern changes in

the environment accurately. Therefore, the question of how window size, as an essential hyperparameter, affects the performance of our method is a worthwhile answered question. We conducted experiments with different window sizes in all four environments, as shown in Table 3.6 and Figure 3.12. It was found that a diminutive window size adversely affects the performance by failing to detect some of the change points, resulting from insufficient data within the window to make an accurate determination. This underscores that too narrow a window may lead to premature conclusions. However, we observed that with a certain amount of data, our adaptive method still performs best over all baselines. Conversely, an excessively large window size could bring extra computational costs. Thus, the optimal window size strikes a trade-off between accurate detection and computation cost. The window size in each environment is highlighted in bold in Table 3.6. Additionally, the detection results of different window sizes are as Table 3.7. The setting number of change points is two.

Limitations. As discussed before, one limitation is that the frequency of change points would affect the results. The reason is that when environmental changes occur too frequently, the policy might not converge consistently each time, potentially leading to the utilization of unwell-trained policies and missed change points, which can impact the effectiveness of adaptation. Although DARL can handle relatively 'bad' policies, excessively frequent changes can significantly impact the underlying RL baseline itself, consequently affecting the performance of DARL. Another limitation is that in relatively simple environments where input and output are simple enough, the improvements from policy adaptation might not be as great as in high-dimensional environments with complex inputs, due to simpler environments facilitating easier convergence of traditional algorithms. Finally, the current framework is designed for on-policy RL due to the episodic feature, while extending it to off-policy RL shows promising potential.



(a) Cartpole

(b) LunarLander

(c) Minigrid

(d) Vizdoom

Figure 3.12: The impact of the change detection window size is notable; we observe that detection performance affects the reward, stabilizing once the window size reaches a certain threshold.

3.5 Summary

In this chapter, we formalized the problem of reinforcement learning in non-stationary environments with unknown change points. To handle such problems, we developed an end-to-end algorithm involving two steps in the change detection and adaptation process. First, points of change are detected from the joint marginal/conditional distribution. Second, a distance-relaxed gradient-constrained adaptor quickly trains a new policy

Table 3.6: The detection results of different window sizes. The setting number of change points is two.

CartPole		**LunarLander**	
Window Size	F1 Score	Window Size	F1 Score
100	0.67	200	0.5
150	0.5	250	0.8
200	0.8	**300**	**1**
250	**1**	350	1
300	1	400	1
350	1	450	1
MiniGrid		**Vizdoom**	
Window Size	F1 Score	Window Size	F1 Score
200	0	60	0.57
400	0.5	120	0.67
600	0.5	180	0.8
800	0.8	240	0.8
1000	**1**	**300**	**1**
1200	1	360	1

with the help of former well-trained policies. We show that the joint distribution-based detection is more accurate than the marginal distribution alone. We also demonstrate that traditional multi-task/continual adaption is unsuitable for our sequential setting, where it is beneficial to treat former policies differently. In a series of experiments in different control environments, our approach accumulated the most rewards, adapted the fastest, and demonstrated near-optimal change detection. Note that the framework presented can be applied easily to other popular on-policy DRL algorithms.

In future work, we plan to investigate the theoretical guarantee of proposed ideas related to policy improvement. Another interesting work will investigate the mutual

CartPole			**LunarLander**		
Window Size	Detected	Error	Window Size	Detected	Error
100	1	0	200	2	1
150	2	1	250	3	1
200	3	1	**300**	2	0
250	2	0	350	2	0
300	2	0	400	2	0
350	2	0	450	2	0
MiniGrid			**Vizdoom**		
Window Size	Detected	Error	Window Size	Detected	Error
200	1	1	60	5	3
400	2	1	120	4	2
600	2	1	180	3	1
800	3	1	**240**	3	1
1000	2	0	300	2	0
1200	2	0	360	2	0

Table 3.7: Window Size used when detecting environment change points.

boosting of detection and adaptation. Extending this framework to off-policy algorithms would also be appealing. For adaptation, further endeavoring to develop corresponding strategies to make the most out of the samples is worth investigating.

CHAPTER 4

A Behavior-Aware Approach

This chapter aims at the research objectives 2 and 3 mentioned in the Chapter. 1. The key inspiration behind our method in this chapter stems from the observation that policies exhibit distinct global behaviors across changing environments. Our approach forgoes manually set thresholds for change detection. Instead, we identify environmental changes by analyzing variations between the behaviors induced by different environmental conditions. This analysis leverages Wasserstein distances, a powerful metric for quantifying the discrepancy between probability distributions over policy behaviors. By monitoring shifts in the Wasserstein distances between current and past policy behaviors, we can detect points where the environment has undergone a transition, triggering a change in the optimal policy.

This chapter is based on Z. Liu, J. Lu, J. Xuan and G. Zhang, "A Behavior-Aware Approach for Deep Reinforcement Learning in Non-stationary Environments without Known Change Points," in International Joint Conference on Artificial Intelligence (IJCAI) 2024, [Accepted].

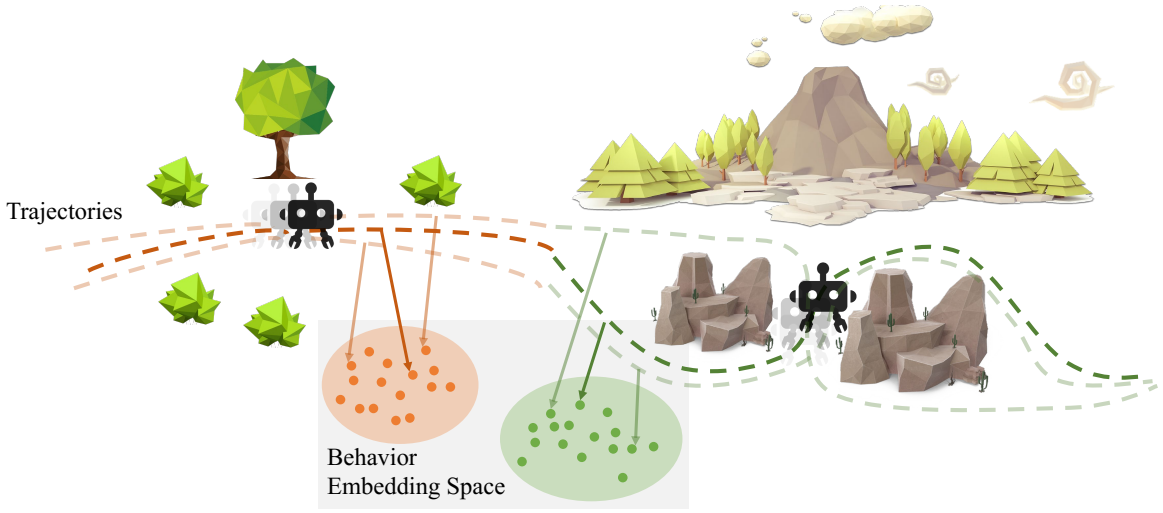


Figure 4.1: When an outdoor robot moves from flat terrain to mountains, its speed, direction, and acceleration control changes corresponding to the changing conditions. We believe these variations can be fully captured through behavior.

4.1 Background

Deep reinforcement learning has extensive applications in economics [205], energy engineering [206, 207], medical analysis [208, 209] and other domains, where policies are trained to make optimal sequential decisions in an assumed stationary environment. However, in practice, stationary environments are rare. Rather, the norm is non-stationary environments where the underlying environment can change in quite unpredictable and abrupt ways. For instance, outdoor robots must navigate constantly changing terrain and lighting levels, while financial markets should rapidly shift alongside breaking news and global events. Hence, ignoring the non-stationarity of underlying environments will frequently lead to poor performance even using a superior algorithm. There is no doubt that addressing this issue requires a dedicated strategy.

In prior work, several research teams have looked for solutions. Some have converted the problem into a continual multi-task reinforcement learning problem [8, 10], while others have transformed the issue into a meta reinforcement learning problem [210, 77]. Yet the common thread in all these studies is that the change points need to be known

in advance, as these change points are used to divide the non-stationary environment into 'multiple tasks.' However, there are often no ready-to-use indicators for unpredictable changes. Furthermore, a typical continual learning setting focuses on preventing catastrophic forgetting, while remembering the knowledge from previous tasks may not contribute to the current adaptation, especially in more practical environments without cyclically recurring tasks. Converting the problem into a continual problem can be troublesome because one of the primary goals of any continually observed task is to avoid catastrophic forgetting, but cyclically recurring tasks are not common in practical environments. Furthermore, such conversion is problematic because their primary goal is to resist catastrophic forgetting in continually observed tasks. As we know, there are often no ready-to-use change points for unpredictable changes, and cyclically recurring tasks are less common in more practical environments.

To address the absence of known change points, some research actively detects environmental changes using methods like reward-based detection [13, 8] or state-based detection [14]. However, the reward-based method generally requires timely rewards and manually set thresholds. In addition, the state information is not comprehensive and accurate enough for detection because different global behaviors may have the same final state or perform similar actions at a local level [211]. Therefore, changes in state alone do not serve as reliable indicators for determining environmental changes.

We posit that the agents in an environment can be better characterized through their behavior. In our research, behavior represents the embeddings mapped from the sequences of states, actions and rewards during a period. As demonstrated in Figure 4.1, when an outdoor robot encounters environmental conditions change, such as terrain, its speed and direction tend to demonstrate significant changes from those of the previous terrain. However, the separate variables like speed and direction at a few time steps can not describe the comprehensive trajectory change, making it tough to understand and

adapt to the new environment. In contrast, behavior can offer more comprehensive information from a global level. We believe that behavior distribution changes simultaneously reflect environmental changes and can help us adapt to new conditions, so our proposed method uses behavior as a core indicator and knowledge. We propose using these shifts in behavior distribution to detect environmental changes. Additionally, these changes indicate that departing from the behavior in the original environment is beneficial for optimal behaviors within the new conditions.

Inspired by this, we present a novel approach to detect environment changes by monitoring behavior distribution shifts based on the Wasserstein distance [212, 213]. The agent(s)' behavior is then regularized accordingly to help the policy steer away from the previous optimum and adapt to new environmental conditions. Experiments in benchmark environments prove our method to be effective and accurate compared to other methods. We propose a setting that enhances the applicability and effectiveness of reinforcement learning across diverse fields, from robotics navigating in dynamic landscapes to trading systems that can respond to volatile markets.

Our main contributions are summarized as follows,

- We propose an environmental change detection method, testing environmental change points through the Wasserstein distance between the global behavior information without manually setting thresholds.

- With detected change points, we introduce a policy adaptation method that facilitates faster deviation from the previous optimum and exploration of new behavioral regions. We adjust regularization based on the extent of change to ensure adaptability under various conditions.

- We provide an end-to-end framework called Behavior-Aware Detection and Adaptation (BADA), which addresses deep reinforcement learning adaptability in non-

stationary environments without known change points by analyzing and employing behavior.

4.2 Problem Formulation

A Markov decision process $\mathbf{M}$ is defined by a state space $\mathscr{S}$, a starting state distribution $p_0(s)$, an action space $\mathscr{A}$, and a reward function $\mathscr{R}: \mathscr{S} \times \mathscr{A} \to \mathbb{R}$. A policy π_θ is parameterized by θ. The interaction trajectory $\tau = \{s_0, a_0, r_0, s_1, a_1, r_1, ...\}$ is collected by a policy π_θ. With a discount factor of γ, the optimal policy π^* is the one that maximizes the expected discounted reward:

$$\pi^* = \arg\max_\pi \mathbb{E}_{\tau\sim\pi}\left[\sum_t \gamma^t \mathscr{R}(s_t, a_t)\right]. \tag{4.1}$$

Standard reinforcement learning assumes that the underlying $\mathbf{M}$ is dynamic but fixed. When this assumption does not stand, a reinforcement learning scheme for non-stationary environments must be implemented. Further, this chapter targets a specific problem within non-stationary environments, settings in which the change points are not known. Formally:

Problem 4.1. *Let* $\{\mathbf{M}_{k=1:K}\}$ *be a sequence of different MDPs with the change points* $\{C_1, ..., C_{K-1}\}$. *An agent will sequentially interact with* $\{\mathbf{M}_{k=1:K}\}$ *with unknown change points, where the goal is to find a sequence of policies for obtaining the optimal expected discounted cumulative reward:*

$$\pi^*_{1:J} = \arg\max_{\pi_{1:J}} \mathbb{E}_{\tau\sim\pi}\left[\sum_{k=1:J}\sum_{t=C'_{k-1}}^{C'_k} \gamma^t \mathscr{R}(s_t, a_t)\right], \tag{4.2}$$

where $\{C'_1, ..., C'_{J-1}\}$ *are the detected change points.*

The problem encompasses two sub-goals: detecting change points accurately and adapting to the new environment quickly detected changes.

4.3 Methodology

In this chapter, a fundamental assumption is that given a policy π, the trajectory has different behavior distributions in different environments. This is the basis of change point detection and new environment adaptation.

4.3.1 Behavior-based Change Detection

During the training process, the policy continuously interacts with the environment. Within each update epoch t, the trajectories collected by π_θ are denoted as

$$\tau = \{s_0, a_0, r_0, ..., s_H, a_H, r_H\}, \tag{4.3}$$

where H is the step taken in this epoch. A behavioral embedding map $\Phi : \Gamma \rightarrow \mathscr{E}$ is used to map the trajectories into a behavioral latent space. In our particular implementation, this map function is a multilayer perceptron. The embedding $\mathbb{P}_\theta$ represents the behavior embedding distribution corresponding to policy π_θ at epoch t.

As mentioned previously, environmental non-stationarity leads to a shift in the trajectory. Therefore, the behavior distributions from two adjacent epochs $\{\mathbb{P}_{t-1}, \mathbb{P}_\theta\}$ are used to quickly identify the change points promptly. Here, the Wasserstein distance [214, 213] is used as the measure for evaluating the difference between behavior trajectories. The Wasserstein distance originates from the optimal transport problem, which evaluates the cost required to transform one probability distribution into another. Given two distributions μ, ν, the Wasserstein distance is defined as

$$\mathrm{W}(\mu, \nu) = \inf_{\gamma \in \Gamma(\mu,\nu)} \int c(x, y) \mathrm{d}\gamma(x, y), \tag{4.4}$$

where $\Pi(\cdot, \cdot)$ denotes the joint distribution with marginal distributions, and $c(\cdot, \cdot)$ denotes the cost function quantifying the distance between two points. If the cost of a move is simply the distance between the two points, then the optimal cost is identical to the

definition of the Wasserstein 1-distance [215]. We calculate the distance by using the dual form of Equation (4.4), which is defined as:

$$\mathrm{W}(\mu, \nu) = \sup_{f_\mu, f_\nu} \int f_\mu d\mu(x) - \int f_\nu d\nu(y) \quad , \tag{4.5}$$

where $f_\mu, f_\nu : \mathbb{R}^d \to \mathbb{R}$ and $\mathrm{Lip}(f_\mu) \leq 1$. The $\mathrm{Lip}(f)$ denotes the minimal Lipschitz constant for the function f. To calculate the Wasserstein distance, the objective is to find the optimal f_μ^*, f_ν^* to maximize the integral.

Wasserstein distance is a metric that reflects the proximity between two distributions, even if no overlap components exist. This property is important for our problem because the agent may manifest completely different behavior before and after changes. Therefore, the support between these distributions on behavior spaces would be limited, and then a proper distribution distance definition for such a situation is crucial. Additionally, its symmetrical nature offers a more effective measure of the differences between distributions compared with other options, like KL divergence. For example, as Figure 4.2 shows, when a policy is sequentially trained from one environment to another - say where the textures and lighting change - the agent's behavior embedding distribution will show a distinct shift in distribution without overlapping. This observation can also help us identify these behavioral-level changes using the Wasserstein distance.

With the evaluated distance before and after a potential change point, we still need to decide on the change, which is normally based on a manually determined threshold. It is difficult because it depends on the environment and behavior distributions, and what is even worse is that different change points may need different thresholds. Here, we propose to perform the permutation test [216, 217], which infers the presence of any change points. The permutation test is an exact statistical hypothesis test based on proof by contradiction. This method involves permuting the order of samples, recalculating statistical test metrics, constructing an empirical distribution, and then determining the p-value based on this distribution to make inferences.

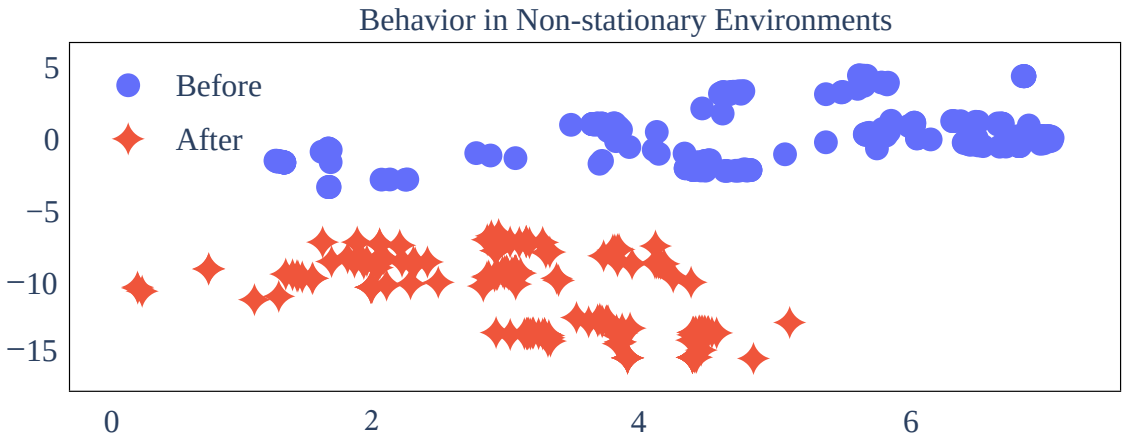


Figure 4.2: This figure presents a t-SNE plot of behavior. The distinct clusters demonstrate the significant impact of environmental changes on behavior and inspire us to use the behavior to adapt actively to coming changes.

To explain the permutation idea, given two samples from adjacent behavior embedding distributions $\mathbb{P}_\theta, \mathbb{P}_{t-1}$ and calculate the test statistic $T = \mathrm{W}(\mathbb{P}_\theta, \mathbb{P}_{t-1})$. The typical null hypothesis is given by:

$$H_0 : \mathbb{P}_\theta = \mathbb{P}_{t-1}, \tag{4.6}$$

i.e., all samples come from the same distribution. Then, for each permutation $e = 1, 2, ..., E$, randomly permute the components of $\mathbb{P}_\theta \cup \mathbb{P}_{t-1}$, and split the permuted data into $\mathbb{P}_\theta^{(e)}, \mathbb{P}_{t-1}^{(e)}$ with the original sizes, then calculate test statistics $T_e = \mathrm{W}(\mathbb{P}_\theta^{(e)}, \mathbb{P}_{t-1}^{(e)})$. By repeating the permutation and calculation, a p-value is given by

$$p = \frac{1}{E} \sum_{t=1}^{E} 1\{T_e \geq T\}, \tag{4.7}$$

where 1 is an indicator function. This test is guaranteed to control the type-I error [218] because we evaluate the p-value of the test via the permutation approach. In addition, the non-parametric nature, i.e., that it does not rely on assumptions about data distribution. As Figure 4.2 shows, the trajectory distribution usually does not conform to an easily computable and representable form of distribution. Therefore, using a permutation test is highly suitable for solving our problem. If the p-value is lower than the significance

Algorithm 2 Behavior Change Detection

Input: Behavior embedding $\mathbb{P}_{ep-1}, \mathbb{P}_{ep}$, number of permutation times N_p and the selected significance level α.

1: Calculate the Wasserstein Distance $T = W(\mathbb{P}_{ep-1}, \mathbb{P}_{ep}$ between the original data.
2: **for** each permutation $t = 1, 2, ..., N_p$ **do**
3: Shuffle the $\mathbb{P}_{ep-1} \cup \mathbb{P}_{ep}$ randomly, and split the permuted data into $\mathbb{P}^{(t)}_{ep-1}, \mathbb{P}^{(t)}_{ep}$ of the original sizes.
4: Calculate the Wasserstein Distance between the permuted data $T_t = W(\mathbb{P}^{(t)}_{ep-1}, \mathbb{P}^{(t)}_{ep})$.
5: **end for**
6: **return** the p-value $\frac{1}{N_p} \sum_{t=1}^{N_p} 1\{T_t \geq T\}$.

level, the current epoch t is noted as a change point $c = t$, and $\mathbb{P}_{pre} = \mathbb{P}_{c-1}$ is the optimal behavior distribution corresponding to the previous environment. The detailed behavior-based change detection method is presented in Algorithm 2. Then, the adaptation scheme is involved.

4.3.2 Behavior-Aware Adaptation

Although the vanilla DRL can adapt to the new environment, it normally requires many interactions that sample inefficient and generate great delay. With the detection signal from the above section, we aim to achieve fast adaption. Since our detection is based on Wasserstein distance, we follow Wasserstein-based policy gradient baseline - Behavior Guided Policy Gradients (BGPG) [211]. Its training objective (for stationary environment) is to maximize:

$$F(\theta) = \mathbb{E}_{\tau \sim \mathbb{P}_\theta}[\mathscr{R}(\tau)] - W(\mathbb{P}_\theta, \mathbb{P}_{t-1}), \tag{4.8}$$

where $\mathbb{P}_{t-1}$ is the behavior distribution of last update epoch.

When a policy converges in one environment, the behavior will enter a relatively stable distribution, and this provides a basis for us to detect changes in the environment. When a change point is detected at epoch c, indicating that a significant change in the environment has occurred, $\mathbb{P}_{c-1}$ is saved as a previous optimal behavior distribution $\mathbb{P}_{pre}$. To assist the policy in quickly deviating from the optimal behavior of the previous

Algorithm 3 Behavioral Aware Detection and Adaptation (BADA)

Initialize: Policy π_θ, behavioral embedding mapping function Φ, and significance level α.

for Epoch $t = 1, 2, ...,$ **do**
  Collect $\tau = \{s_0, a_0, r_0, ..., s_H, a_H, r_H\}$ from the current environment.
  Obtain behavior embedding $\mathbb{P}_\theta$ by behavioral embedding mapping function Φ.
  Compute the original statistics $T = W(\mathbb{P}_\theta, \mathbb{P}_{t-1})$.
  for Permute iteration $e = 1, 2, ...E$ **do**
    Shuffle $\mathbb{P}_\theta \cup \mathbb{P}_{t-1}$ and split the data into $\mathbb{P}_\theta^{(e)}, \mathbb{P}_{t-1}^{(e)}$ and compute statistics $T_e = W(\mathbb{P}_\theta^{(e)}, \mathbb{P}_{t-1}^{(e)})$.
  end for
  Obtain the p-value $\frac{1}{E}\sum_{t=1}^{E} 1\{T_e \geq T\}$
  if p-value$\leq \alpha$ at epoch c **then**
    Save $\mathbb{P}_{c-1}$ as previous behavior distribution $\mathbb{P}_{pre}$.
    Update policy parameter by $\theta_{t+1} \leftarrow \theta_t - \alpha \nabla_\theta F(\theta_t)$ following Equation (4.9)
  else
    Update policy parameter by $\theta_{t+1} \leftarrow \theta_t - \alpha \nabla_\theta F(\theta_t)$ following Equation (5.12).
  end if
  Save $\mathbb{P}_{t-1} \leftarrow \mathbb{P}_\theta$ for environment change detection.
end for

environment, we propose to add a regularizer that maximizes the difference between the current behavior distribution and the previously converged behavior distribution. This new objective function is designed as follows:

$$F(\theta) = \mathbb{E}_{\tau \sim \mathbb{P}_\theta}[\mathscr{R}(\tau)] - \mathrm{W}(\mathbb{P}_\theta, \mathbb{P}_{t-1}) + \delta \mathrm{W}(\mathbb{P}_\theta, \mathbb{P}_{pre}), \tag{4.9}$$

where $\mathscr{R} = \sum A^{\pi_{t-1}}(s_i, a_i) \frac{\pi_\theta(a_i|s_i)}{\pi_{t-1}(a_i|s_i)}$, $A^{\pi_{t-1}}(s_i, a_i)$ is the advantage function, and $\mathbb{P}_{pre}$ is the converged behavior distribution in the previous environment, and $\delta \in \mathbb{R}_{>0}$ is a hyper-parameter. Here, we use the adjacent behavior distance on the detected change point $W(\mathbb{P}_{c-1}, \mathbb{P}_c)$ as δ, depending on the extent of change. This self-adjusted coefficient ensures that the adaptation regularization has a greater impact as the level of environmental change increases.

If no change is detected, the adaptation term will not work, so δ will be set as zero. The first penalty constrains policy updates within a trust region, ensuring the validity of importance sampling. However, this constraint can lead to slow adaptation when

the environment undergoes abrupt changes, as the policy hesitates to deviate from its previous optimal behavior. At the change point c, $\mathbb{P}_{prev} = \mathbb{P}_{c-1}$. Only following the first penalty term at this point might trap the policy in a suboptimal area for an extended period. Therefore, our second adaptation regularization serves as an adversarial term, steering the policy away from previous behavior. As the policy gradually adapts to the current environment, i.e., $t \gg c$, the adaptation term $\mathrm{W}(\mathbb{P}_\theta, \mathbb{P}_{c-1})$ and the first term $\mathrm{W}(\mathbb{P}_\theta, \mathbb{P}_{t-1})$ no longer conflict. The penalty constraints ensure performance improvement in a stationary environment, and the role of the adaptation term weakens as the policy moves away from the previous optimum.

With the optimal f_μ^*, f_ν^* according to Equation (4.5), the regularization term in Equation (4.9) is:

$$\mathrm{W}(\mathbb{P}_\theta, \mathbb{P}_{pre}) \approx \mathbb{E}_{\tau\sim\mathbb{P}_\theta}[f_\mu^*(\tau)] - \mathbb{E}_{\phi\sim\mathbb{P}_{pre}}[f_\mu^*(\phi)]. \tag{4.10}$$

Maximizing this term can guide the optimization by favoring those trajectories that show more difference between old ones. When another change occurs, we consider only the preceding behavior distribution. We believe that excessive constraints may lead to a narrow area and result in local optima. Therefore, focusing on the immediate historical behavior ensures adaptability to changing environments without introducing unnecessary complexities. This training goal allows us to scale to scenarios with multiple changes easily.

Overall, maximizing the difference between behaviors serves a contrastive term compared to the RL constraint $W(\mathbb{P}_\theta, \mathbb{P}_{t-1})$, which prevents the policy from deviating excessively from the old one. As shown in Equation (4.9), our regularization $W(\mathbb{P}_\theta, \mathbb{P}_{pre})$ does not entirely dominate the search for optimal behaviors. These two constraints achieve a balance by allowing the policy to adapt to new environments while simultaneously retaining some knowledge from the previous optimum. Additionally, when the environment experiences minor changes in state distribution at point c, then $W(\mathbb{P}_{c-1}, \mathbb{P}_c)$

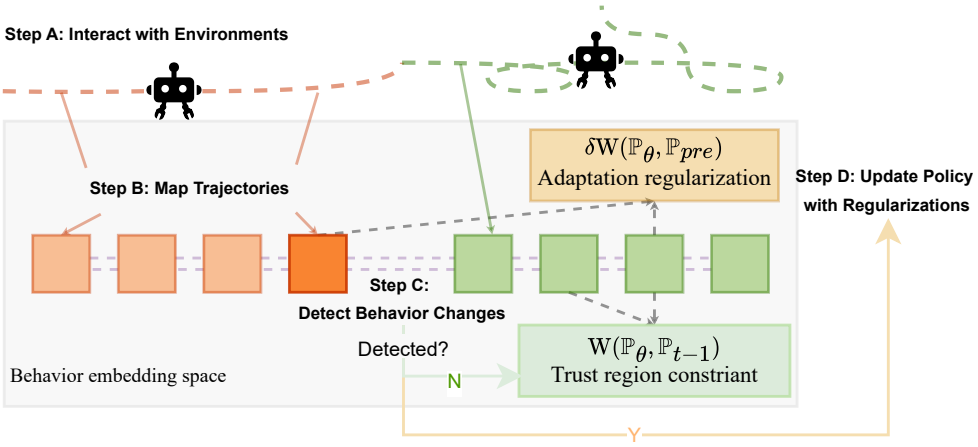


Figure 4.3: The BADA framework. When a change is detected through the behavior distribution permutation test, regularization will be added to deviate policy behavior from the previous optimum.

will be relatively small. Consequently, the regularization coefficient δ will be adjusted, ensuring a flexible and relaxed constraint that responds appropriately to the extent of change. Figure 5.1 illustrates the adaptation scheme, and Algorithm 4 sets out the complete BADA method in detail.

4.4 Experiments and Analysis

This section comprehensively evaluates our BADA method, addressing key questions: 1) Can BADA achieve higher rewards in environments without known change points? 2) Is behavior-based change detection superior to alternative methods? 3) Does BADA's adaptation method outperform retraining and other adaptation approaches? 4) Can BADA maintain performance with frequent environmental changes? These inquiries guide our experiments and analysis.

4.4.1 Settings

Environments. We conducted all experiments within ViZDoom [219], a first-person shooting game with various scenarios. This environment allows reinforcement learning agents to be developed using only visual information (the screen buffer). We chose four scenarios through which to evaluate our proposed method. We employ distinct challenges and modifications to simulate dynamic environments for training an agent. For example, as Figure 5.3 shows, the environment transit from high-contrast *simpler_basic* to dimly lit *basic* settings, shift from defending a line in a rectangle map to defending a point in a circular map against enemies in *defend_the_line*/*center*. In addition, we adjust the number of enemies in *deadly_corridor* and change the medikit textures in the *healt_gathering* scenario to represent new rooms. Agents need to respond to these changes.

- ***basic*/*simpler_basic*:** In *basic* scenario, a player faces off against a randomly spawned monster. The player can go left/right and shoot to kill the monster. The *simpler_basic* scenario works the same but uses different textures for better contrast to help the agent learn faster. We simulate the environment change by switching from *simpler_basic* to *basic* with dimmer lighting.

- ***deadly_corridor*:** The map is a corridor with shooting monsters on both sides. A green vest is placed at the opposite end of the corridor. The objective for the player is to get the vest and avoid being killed somewhere along the way. The player can choose from 7 available actions: move forward/backward/left/right, turn left/right, shoot. We simulate environment changes by reducing the number of enemies.

- ***health_gathering*:** The map is a rectangle with a green, acidic floor, which hurts the player periodically. There are some medkits spread uniformly over the map.

The player needs to pick up these medkits to survive. We changed the texture of the medkits and walls to simulate entering a new room.

- ***defend_the_line*/*defend_the_center*:** In these two scenarios, a player is spawned along the longer wall of a rectangular map, facing three melee-only and three shooting monsters on the opposite wall. These monsters will respawn after a certain period. In the altered environment, the agent's objective shifts to eliminating surrounding enemies at the center of a circular map.

Each training episode has a fixed tick depending on different scenarios, and the agent is allowed to make an action every 12 frames. The action chosen based on the first frame remains fixed and is consistently maintained throughout the subsequent frames. If the agent performs one action per frame, the difference between the states is so subtle that it significantly impedes the learning process. Therefore, we adopted a repeat action strategy to solve this issue.

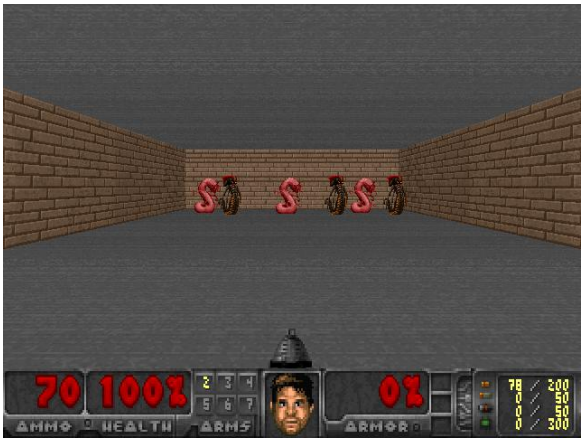
Comparison methods. In all experiments, we used the PPO/TRPO update, and once the environment changed, the model could not access any information on the changed environmental conditions. Further, we compared BADA to three baseline methods as follows.

- PPO [32] and TRPO [31] without detection and adaptation;
- Behavior-based BGPG [211] without detection and adaptation;
- CRL-Unsup [13] with both detection and adaptation.

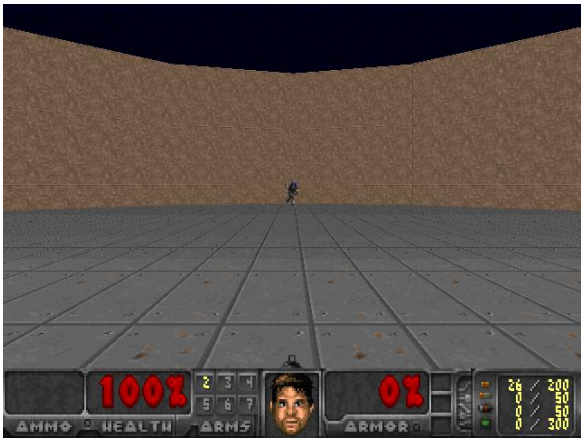
The agent architecture for all methods consisted of a 4-layer convolutional neural network (ConvNet) with 3x3 kernels featuring 16 maps, complemented by ReLU activation functions. This was followed by a fully connected layer that outputs a distribution of action sizes.

simpler_basic

basic



defend_the_line



defend_the_center

Figure 4.4: The simulated non-stationary environments. The upper setting is from high-contrast *simpler_basic* to dimly lit *basic* scenario, and the bottom one is from *defend_the_line* with a rectangular map to *defend_the_center* with a circular map.

To evaluate performance in terms of environmental change detection, we compared BADA to:

- A permutation test using KL divergence
- A two-sample test using weighted maximum mean discrepancy (WMMD) [220]
- The online parametric Dirichlet change point (ODCP) [70]

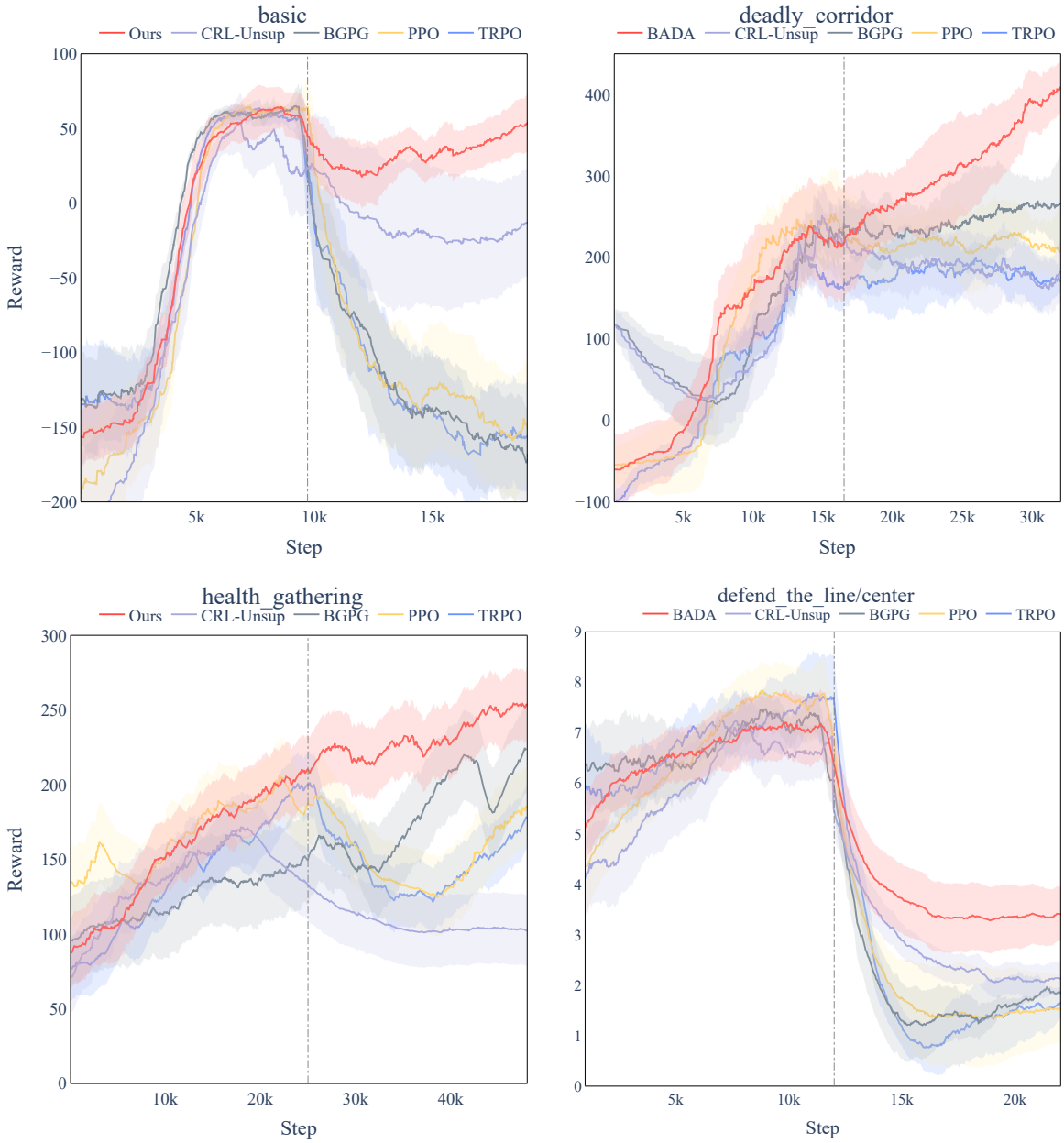


Figure 4.5: Performance comparison of different methods in non-stationary environments. The vertical dashed lines represent the points of environmental change, and the shaded areas around the reward lines indicate the standard deviation over different runs.

- CRL-Unsup [13], which is based on long and short-term rewards.

Metrics. One metric is the commonly used cumulative reward. The other metric is F1 Score $F_1 = \frac{2*P*R}{P+R}$ [221], indicating the detection accuracy.

Training details. The agent architecture for all methods consisted of a 4-layer con-

volutional neural network (ConvNet) with 3x3 kernels featuring 16 maps, complemented by ReLU activation functions. This was followed by a fully connected layer that outputs a distribution of action sizes.

4.4.2 Overall Performance

We evaluated the results from two aspects: cumulative reward and detection accuracy.

Cumulative rewards. As depicted in Figure 4.5, BADA (illustrated in red) demonstrates a notable acceleration in reward accumulation following the change point (indicated by the vertical dashed line in each graph). The rapid improvement in rewards post-change point can be attributed not only to the efficiency of the adaptation regularization, which enables a swift deviation from the previous optimal policy, but also to BADA's precise response to the environmental shifts.

In the *basic* scenario, where the lighting and wall texture undergo modifications, methods lacking adaptation mechanisms experience a considerable decline in performance. CRL-Unsup showcases a commendable adaptation capacity, with a steady increase in rewards after the initial detection of environmental changes, although slightly lagging behind BADA. This suggests that our behavior-based regularization term facilitates quicker adaptation to new settings.

In the *health_gathering* scenario, the alteration in the medkit's texture has a less pronounced effect than changes in lighting levels. The results reveal that even methods without adaptation are capable of regaining relatively high rewards after a few updates. Meanwhile, BADA's rewards continue to ascend, highlighting its adaptability even in environments with more subtle changes. However, CRL-Unsup appears susceptible to false alarms regarding environmental changes, leading to unnecessary adaptations and a less stable learning trajectory.

Interestingly, in the *deadly_corridor* scenario, a reduction in the number of enemies

Table 4.1: Comparative F1 scores of change detection methods in non-stationary environments. The results are based on ten runs of different seeds.

	basic	*health_gathering*
BADA(Ours)	**0.95±0.08**	**0.90±0.11**
Permutation(KL)	0.70±0.09	0.50±0.26
CRL-Unsup	0.80±0.12	0.35±0.16
WMMD	0.50±0.13	0.56±0.27
ODCP	0.55±0.36	0.37±0.07
	deadly_corridor	*defend_the_line*
BADA(Ours)	**0.78±0.16**	**0.86±0.07**
Permutation(KL)	0.69±0.09	0.50±0.26
CRL-Unsup	0.67±0.21	0.72±0.11
WMMD	0.47±0.19	0.62±0.15
ODCP	0.38±0.20	0.50±0.19

does not translate to additional rewards for well-trained agents that do not utilize adaptation strategies. This could be due to their adherence to the original behavior, resulting in a delayed response to the environmental alterations. In contrast, BADA quickly attains higher rewards and maintains an upward trend, showcasing its capacity to continuously learn from new environmental conditions.

In the *defend_the_line*/*center* scenarios, changes in the map shape and defended goals compel each method to learn an entirely new task. Here, BADA and CRL-Unsup surpass other baselines by swiftly attaining higher scores on the new task, with BADA exhibiting superior performance overall. This underscores the effectiveness of moving away from the previous optimal strategy to discover a new one.

Environment change detection accuracy. Table 4.1 provides a comparative analysis of the F1 scores for change detection methods in various non-stationary environments,

where BADA consistently surpasses other methods across all scenarios, indicating its robust adaptability and precision in change detection. In the *deadly_corridor* scenario, despite BADA's lower accuracy compared to other environments, it still outperforms other methods, which underscores the relative challenge of detecting changes in environments where the behavioral impact of changes is subtler, such as the reduced number of enemies.

The table also reflects on the relative weakness of the permutation test based on KL divergence, particularly in scenarios where environmental state distributions do not align well with the prerequisites of KL divergence. The challenges in measuring distributional shifts where data is sparse or supports do not overlap may lead to poorer performance, a limitation evidently overcome by the Wasserstein-based approach used by BADA. The Wasserstein distance, rooted in the concept of optimal transport, excels in scenarios that feature stark distributional changes, as it encapsulates the minimum cost to transform one distribution to another, proving particularly potent in non-stationary settings encountered in reinforcement learning.

Furthermore, while CRL-Unsup shows reasonable performance, its reliance on extensive testing for manual threshold selection is a significant drawback compared to the BADA method, which requires no manual tuning of hyperparameters and inherently provides a level of significance. This autonomy in BADA affords a more streamlined and user-friendly application, removing the often labor-intensive process of hyperparameter optimization. The table also indicates that methods like ODCP and WMMD fall short in image-based scenarios, suggesting an inefficiency in handling the complexities of high-dimensional data inherent to images.

These insights suggest several avenues for further research and development. Refining change detection mechanisms to improve accuracy and reduce the need for manual interventions remains a priority. Moreover, exploring ways to extend the efficiency of

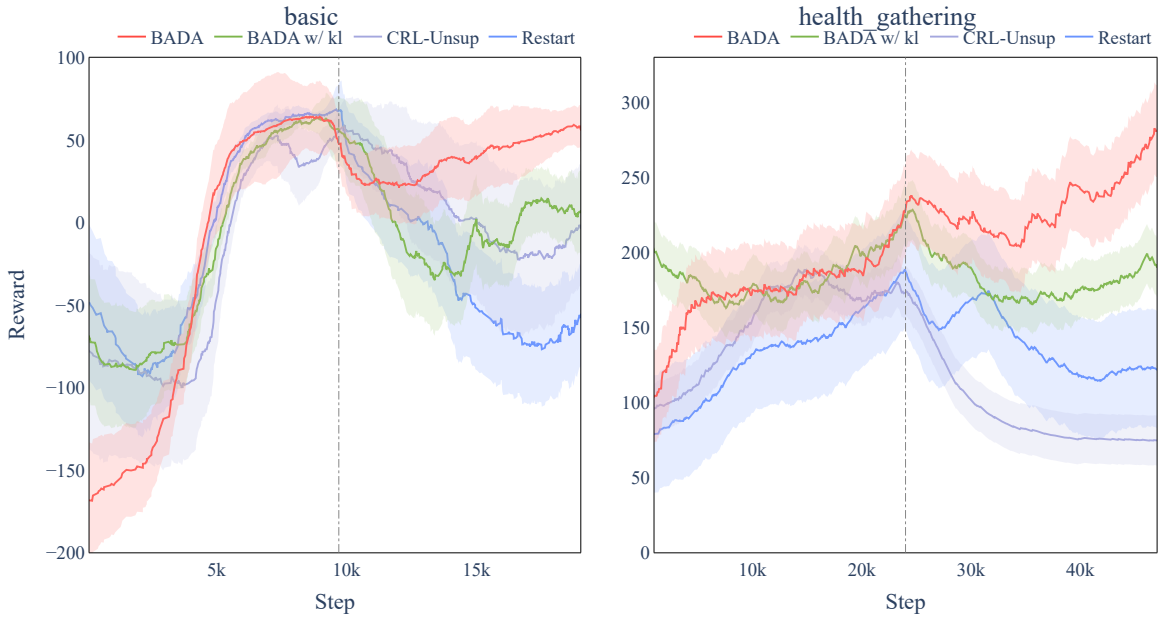


Figure 4.6: Cumulative rewards of adaptation strategies in non-stationary environments with known change points.

the Wasserstein distance in even more diverse settings could help adapt these methods to a broader range of practical applications. This could be especially beneficial in real-world environments where changes are subtle and not easily discernible through raw data analysis. By addressing these challenges, the potential to apply BADA and similar methods to a wider array of domains-including those outside traditional reinforcement learning applications-becomes increasingly feasible.

4.4.3 Ablation Study

Adaptation evaluation. To evaluate the performance of adaptation separately and confirm whether the adaptation scheme contributes to new training as opposed to simply retraining the agent following a reinforcement learning loss, we test the following comparison methods:

- BADA w/ KL that employs KL divergence instead of Wasserstein distance as the regularization term.

- CRL-Unsup with the EWC adaptation method.

- Restarting training following a traditional PPO scheme.

All methods are provided with the change points to initiate adaptation or retraining.

The efficacy of the BADA method is clearly delineated in Figure 4.6, particularly in scenarios where environmental change points are predetermined. Several key observations can be made regarding its performance:

Firstly, the advantage of BADA over the 'Restart' method is unequivocal. This is evidenced by the more rapid rebound and continuous upward trajectory of rewards following a change point. BADA's capability to swiftly shift from an erstwhile optimal behavior to a new one underscores the limitations of reinitializing the training process. Unlike restarting, which discards accumulated knowledge, BADA leverages past learning to expedite the discovery of new optimal policies, highlighting the inefficiencies of starting anew without retaining valuable insights.

Secondly, BADA's superiority extends over alternative adaptation strategies as well. The application of the Wasserstein Distance constraint, in particular, has demonstrated more effectiveness than the KL divergence approach in guiding the adaptation process. This suggests that the Wasserstein metric may provide a more nuanced and beneficial gradient for learning new behaviors in altered environments. Furthermore, BADA's outperformance of CRL-Unsup showcases the merit of behavior-based adaptation strategies. By focusing on the agent's observable behaviors rather than unsupervised learning indicators, BADA harnesses a more direct and possibly more robust signal for guiding adaptation.

Moreover, BADA's effectiveness implies a significant potential for applications in various domains where environments are susceptible to sudden changes. For instance, in robotics, where a machine may need to adapt to new terrains, or in finance, where

market conditions can shift abruptly, BADA's ability to utilize past learning while quickly adjusting to new circumstances can be particularly valuable.

Therefore, the results gleaned from the application of BADA affirm its advanced capability for adaptation in the face of environmental changes. By integrating past experiences and employing a more sensitive measure for divergence, BADA not only improves upon the recovery time from disruptions but also enhances the overall learning trajectory post-adaptation. This aligns with the ongoing pursuit in reinforcement learning to develop algorithms that not only learn efficiently but also possess the agility to adapt to new and unforeseen challenges seamlessly.

Frequently changing environments. The efficacy of BADA in environments with varying frequencies of change points underscores its robustness, but it also highlights the intrinsic challenges associated with maintaining an optimal policy in the face of frequent changes. In extremely non-stationary environments, where change is the only constant, the constraints placed upon the distribution may not reflect the previous optimal policy due to insufficient convergence time between changes.

As demonstrated in Figure 4.7, BADA's ability to achieve higher average rewards, as shown in red, even in environments peppered with frequent change points, is a testament to its robustness. The performance trends reveal that while an increase in the number of change points from 2 to 4 does not drastically affect performance, escalating the frequency to 9 change points introduces a significant challenge, as evidenced by the general performance dip across all methods. Nevertheless, BADA's relatively smaller decline in performance and its consistent outperformance of other methods speaks volumes about its adaptability, even when environmental changes occur at a more rapid clip.

Further insights gleaned from Table 5.3 show that detection accuracy for all methods suffers as the number of change points climbs. This raises critical considerations for the

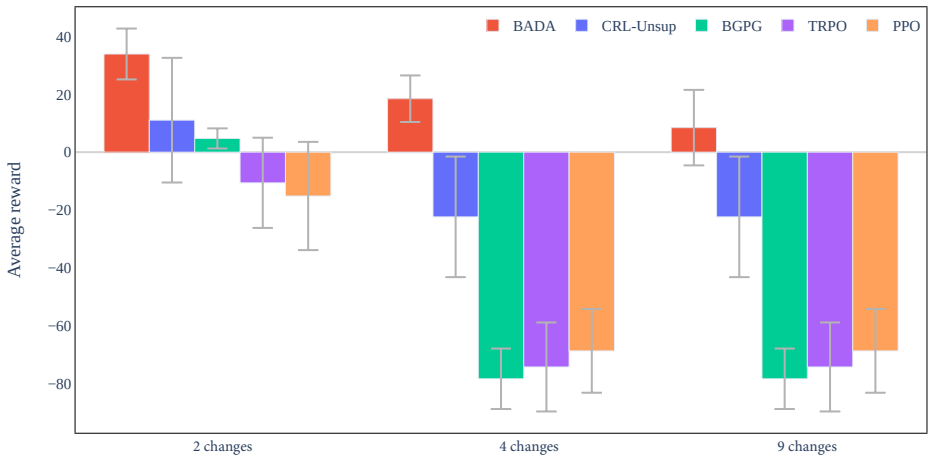


Figure 4.7: Average reward after the first change points in environments with increasing change points.

application of BADA and similar approaches in environments characterized by a high degree of volatility. As policies may not reach convergence within each environmental iteration, agent behaviors can persist in a state of flux, which in turn could diminish BADA's change detection and subsequent adaptation efficiency.

To enhance BADA's change detection and adaptation capabilities in such volatile conditions, future research could explore strategies like increasing the diversity of experience replay to buffer against frequent changes or incorporating predictive models that anticipate changes before they occur. Another promising direction could be the integration of online learning components that allow for continuous, real-time model updates, thereby reducing the dependency on convergence at each environmental stage.

Overall, the insights from BADA's application in highly dynamic environments illuminate the path forward for reinforcement learning in real-world scenarios, where adaptability and resilience are not just beneficial but essential for success. The pursuit of more agile, perceptive, and self-correcting reinforcement learning systems continues to drive innovation in the field, with the ultimate goal of creating agents that can navigate the unpredictability of real-life with grace and efficacy.

Parameter sensitivity. Figure 4.8 shows the parameter sensitivity analysis for

Table 4.2: F1 scores for change detection methods across environments with increasing number of change points. The results are based on ten runs with different seeds.

	2 changes	4 changes	9 changes
BADA(Ours)	**0.89±0.12**	**0.78±0.16**	**0.56±0.20**
Permutation (KL)	0.52±0.19	0.49±0.11	0.43±0.09
CRL-Unsup	0.71±0.13	0.60±0.17	0.37±0.08
WMMD	0.45±0.17	0.40±0.11	0.28±0.19
ODCP	0.25±0.15	0.21±0.11	0.20±0.13

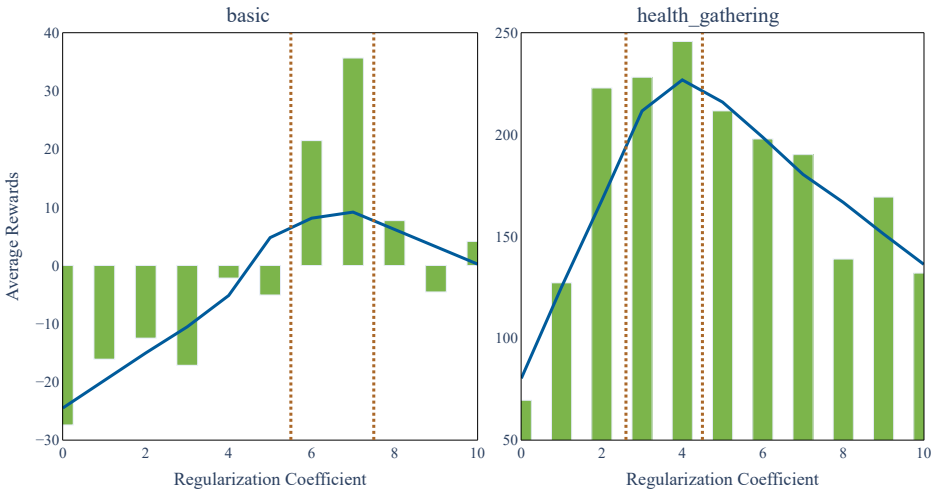


Figure 4.8: The parameter sensitivity analysis of the adaptation regularization. The orange lines represent the coefficient range we used.

the adaptation regularization term $W(\mathbb{P}_\theta, \mathbb{P}_{pre})$ in Equation (4.9). The emergence of a pronounced peak in average rewards for both test environments suggests that there is indeed an optimal coefficient value for the adaptation term, which aligns with the range of our adaptive coefficient, highlighted in orange. This optimal value is crucial as it correlates with the magnitude of environmental changes, essentially dictating the degree to which the model should adapt its behavior in response to detected changes.

The experiments validate the efficacy of our self-adjusting coefficient, which dynamically calibrates according to the environment's variability, thereby endorsing its

implementation in practice. The observed decline in performance beyond the peak signals the detrimental effects of an excessively zealous correction term, which could potentially introduce volatility in the learning trajectory or even lead to detrimental overfitting to the most recent changes, at the expense of previously learned knowledge.

This analysis underscores the necessity of a well-tuned balance in the adaptation regularization term. It is not simply a matter of responding to environmental changes but doing so with a nuanced approach that avoids overcorrection while still permitting sufficient flexibility for the agent to incorporate new information effectively. To enhance this balance further, future work could investigate more sophisticated, perhaps even context-aware, mechanisms for determining the adaptation term, potentially involving real-time assessments of environmental stability or the agent's performance variability. In sum, the adaptation regularization term is a pivotal component of the learning framework, and its precise calibration is instrumental in maintaining an equilibrium between stability and adaptability. The continued improvement of this calibration process is expected to significantly advance the capabilities of reinforcement learning agents, particularly in dynamic and unpredictable environments.

4.5 Summary

Our work addresses deep reinforcement learning in non-stationary environments without known change points by developing the Behavior-Aware Detection and Adaptation (BADA) framework. The behavior-based change detection method represents a novel approach to monitoring and responding to environmental shifts by closely analyzing policy behavior. This method has proven effective and accurate without any manually set threshold, allowing for timely adjustments to the learning strategy. Furthermore, the online adaptation mechanism integrates this behavioral information, providing a self-adjusted regularization term. The behavior-based regularization can help policy steer

from suboptimal areas and find potential behavior in new conditions. The experimental results show its superior performance in accurately detecting changes and quickly adapting to new environments compared to other methods. Future iterations of BADA could benefit from exploring mechanisms for off-policy adaptations, broadening BADA's applicability in various RL settings.

While our BADA algorithm demonstrates robust performance in non-stationary RL environments, it is not without limitations that merit further research and development. One of the primary constraints is the requirement for tuning the parameters associated with the adaptation loss term. Future iterations of BADA could benefit from exploring mechanisms for automatic, dynamic adjustment of these parameters. Integrating self-adaptive methods that can adjust the adaptation loss in response to the environment's dynamics would streamline the learning process, reducing the need for manual tuning and enhancing the algorithm's applicability to a broader range of scenarios. Another limitation lies in the framework's reliance on on-policy RL algorithms. The exploration of off-policy adaptations would thus represent a significant step forward, broadening the horizon of BADA's applicability and efficiency in various RL settings.

CHAPTER 5

A Sample Efficient Approach

This chapter aims to address research objectives 2, 4, and 5 mentioned in the Chapter 1. The method in this chapter contributes a novel approach in function space, converting the policy representation from a deep neural network to a Gaussian Process. Leveraging the converted Gaussian Process, we introduce a Wasserstein surprise-based method for active change detection and a functional regularization mechanism for rapid policy adaptation when the environment changes, incorporating the detected change information. Our approach also employs a unique trajectory selection strategy designed for adaptation regularization to reduce the computational cost significantly. Empirical results demonstrate that our method achieves accurate detection and effective adaptation across various environments with diverse change factors, ultimately leading to enhanced performance and efficiency in various applications.

This chapter is based on Z. Liu, J. Lu, J. Xuan and G. Zhang "Functional Detection and Adaptation in Non-stationary Environments," submitted to IEEE Transactions on Neural Networks and Learning Systems, [under review]

5.1 Background

When an agent interacts continually with the environment, updating based on the latest data, it provides adaptability to slowly evolving environments, particularly for on-policy methods. However, agents often suffer from performance degradation or failure when facing sudden and significant environmental changes. Therefore, to maximize long-term rewards, a promising approach for DRL in non-stationary environments is to develop a policy that *detect* environmental change points actively and *adapt* to new environments accordingly.

Previous research has explored some techniques for detecting changes in non-stationary environments. One method [14] fitted a Dirichlet distribution over multivariate data, which is computationally intensive. Another method [13] identified change points by comparing short-term and long-term rewards using a manually set threshold. Therefore, developing a computationally efficient method *without extra hyperparameter* is crucial. After the change points are detected, one adaptation strategy by retaining previous knowledge by leveraging the Fisher information matrix was proposed in previous work [13]. The challenge with the weight regularizations arises from the fact that the specific values of the weights are not significant, mainly due to parametric symmetries [222]. Making current weights closer to the previous ones may not always ensure using the previous optimal policy. In deep learning, the essential objective is to optimize the weight of neural networks to build the function that maps inputs to outputs. The parameter change does not provide a fully explained proxy for the change in the function [222], leading to insufficient knowledge leveraging. For deep reinforcement learning in non-stationary environments, the goal is to find a policy that achieves the highest cumulative rewards in long-term learning. Hence, a better approach is to *regularize the functions directly*.

We aim to address the critical need for DRL approaches to recognize and adjust to

changes effectively and accurately, ensuring sustained and reliable functionality amidst real-world unpredictability. By exploring the entire learning pipeline and identifying the similarities and differences among existing schemes, this chapter proposes *Functional Detection and Adaptation (FDA)*, which involves environmental change detection and policy adaptation. The framework of our proposed FDA is illustrated in Fig. 5.1. Firstly, we introduce an environmental change detection method based on Bayesian surprise [72]. The idea stems from a straightforward observation: posterior uncertainty increases as the model is queried far from the observed data. To measure the model uncertainty, we use a Gaussian Process (GP) approximation for a given deep neural network. The Bayesian surprise is then measured by the 2-Wasserstein distance [223], and a Welch's t-test is performed between adjacent epochs to identify change points. With detected changes, we performed functional regularization with the converted GP to enhance policy adaptation in new environments. We also designed a trajectory selection strategy to reduce the additional computational cost. The functional regularization of the FDA can be self-adjusted according to change extents, making the policy adapt to new environments accurately. The empirical results show that our proposed method outperforms other baselines in non-stationary environments.

The contributions of this chapter are:

- *Surprise-based Change Detection Method:* We developed a Wasserstein surprise-based detection method that accurately identifies environment change points without requiring extra parameter tuning.

- *Change-aware Functional Regularization:* We proposed a functional regularization method with a self-adjusted coefficient for on-policy RL to effectively adapt to different environmental changes based on representative environment-related data.

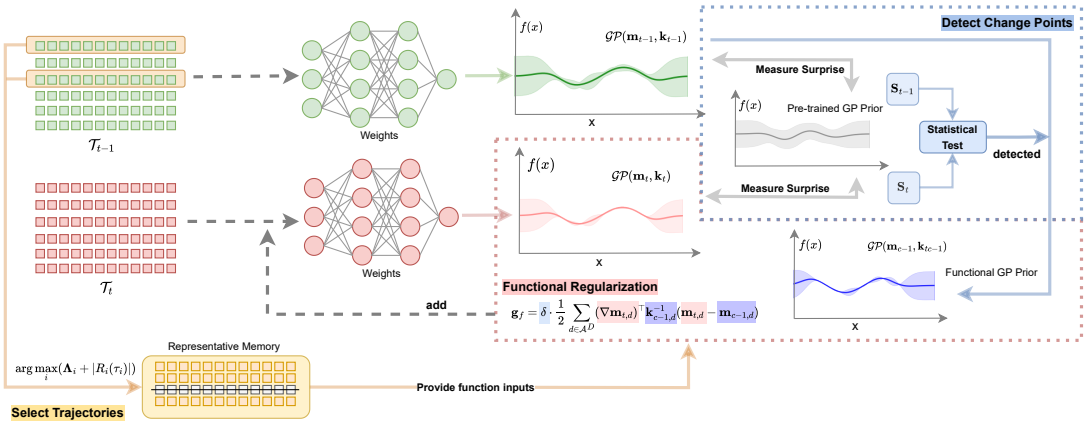


Figure 5.1: The framework of Functional Detection Adaptation (FDA) detects change points based on surprise, selects representative trajectories from interactions, and deploys functional regularization to adapt to new environments when changes are identified. If no change point is detected, the functional regularization will not activate, degenerating into a traditional DRL problem.

- *Representative Trajectories Selection Strategy:* We designed a trajectories selection method to reduce the computational cost of functional regularization.

5.2 Problem Formulation

A MDP is defined by a tuple $\mathbf{M} = \{\mathscr{S}, p_0, \mathscr{A}, \mathscr{R}, \mathscr{P}\}$, where $\mathscr{S}$ is the state space, $p_0(s)$ is the starting distribution of the states, $\mathscr{A}$ is the action space, $\mathscr{R}(r|s,a)$ is the reward function $\mathscr{R} : \mathscr{S} \times \mathscr{A} \to \mathbb{R}$, and $\mathscr{P}(s_{t+1}|s_t,a)$ is the state transition probability. A policy $\pi_w(a|s)$ is a distribution over actions given a state, with w as the neural network weights. These MDPs may differ in any aspect, such as state distribution, transition probability and reward function. Then we have the formal definition of a *stationary environment*:

Definition 5.1. Stationary Environment A stationary environment can be defined as a single MDP $\mathbf{M}$ characterized by a tuple $\mathbf{M} = \{\mathscr{A}, \mathscr{S}, p_0, \mathscr{R}, \mathscr{P}\}$. The action space $\mathscr{A}$, state space $\mathscr{S}$, state distribution p_0 and reward function $\mathscr{R}$ are deterministic and will not change during interactions.

Conversely, we define the *non-stationary environment* as:

Definition 5.2. Non-stationary Environment A non-stationary environment can be defined as a sequence of MDPs $\{\mathbf{M_k}\}_{k\in\mathbb{N}^+}$, where the number of MDPs is unknown. Each MDP $\mathbf{M_k}$ is defined by a tuple $\mathbf{M} = \{\mathscr{A}, \mathscr{S}, p_k, \mathscr{R}_k, \mathscr{P}_k\}$, arriving sequentially over time.

Remark 5.3. The steps of the agent interacting with an arbitrary MDP $\mathbf{M_i}$ are not predetermined, meaning the change points are unknown.

Remark 5.4. The MDPs have the same sizes of action and state space, while the state transition function p_k, reward function $\mathscr{R}_k$, and state distribution p_0 may vary arbitrarily.

Remark 5.5. For different MDPs $\mathbf{M_i}$ and $\mathbf{M_j}$ $(i \neq j)$, at least one component of them is different.

Remark 5.6. The MDP components change unpredictably and are not assumed to satisfy smooth or continuous conditions.

Remark 5.7. The MDPs are not assumed to increase in difficulty or complexity over time.

We identify the problem of learning in a non-stationary environment as follows:

Problem 5.8. *In non-stationary environments, the goal is to find a policy to obtain the maximum expected discounted cumulative reward for a sequence of MDPs* $\{\mathbf{M_k}\}_{k\in\mathbb{N}^+}$*, given as follows:*

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau\sim\pi} \sum_{k=0}^{K} \gamma \mathscr{R}_k(\tau), \tag{5.1}$$

where $\tau \sim \mathscr{T}, \mathscr{T} = \{s_0, a_0, r_0, \ldots\}$ *collected from the interactions with environments,* γ *is the discount factor, and* K *is the actual number of MDPs.*

When there is a detection scheme to identify the MDP change, we use MDPs $\{\mathbf{M'_k}\}_{k'\in\mathbb{N}^+}$ to represent the detected MDPs, which may be different from the detected MDPs $\{\mathbf{M'_k}\}_{k\in\mathbb{N}^+}$.

5.3 Methodology

In this paper, we solve the problem of DRL in non-stationary environments by actively detecting change points and adapting to new conditions using learned knowledge and change information. We develop a novel detection method without involving extra parameters that must be carefully tuned. If a change point is detected, a change-aware functional regularization with a self-adjusted coefficient is deployed to adapt to different extents of environmental changes.

5.3.1 Detecting Environment Changes

In non-stationary environments, a simple method to identify environmental changes is based on an assumption: the uncertainty in a model's outputs increases as it encounters data that deviates from the previously trained data. This concept is closely related to Bayesian surprise [72], which quantifies the divergence between posterior and prior distributions. However, a policy modeled by a deterministic deep neural network has limited capability in uncertainty modeling [224]. To measure the surprise by uncertainty, we first convert DNN to Gaussian Process, representing a distribution over functions by specifying a multivariate Gaussian distribution over all possible function values.

5.3.1.1 Modeling Uncertainty

According to Rasmussen and Williams [225], a linear model can be written as:

$$\begin{aligned} y_i = f_w(x_i) + \epsilon_i, \quad \text{where } f_w(x_i) := \boldsymbol{\phi}(x_i)^\top \mathbf{w}, \\ \mathbf{w} \sim \mathcal{N}(0, \delta^{-1}\mathbf{I}_P) \text{ and } \epsilon_i \sim \mathcal{N}(0, \Lambda^{-1}) \end{aligned} \tag{5.2}$$

with a feature map $\boldsymbol{\phi}(\mathbf{x})$. The parameter $\mathbf{w}$ is drawn from a Gaussian prior, where $\mathbf{I}_P$ is a $P \times P$ identity matrix and δ forms the covariance of the distribution. Then the posterior distribution $\mathcal{N}(\mathbf{w} \mid \mathbf{w}_{\text{lin}}, \Sigma_{\text{lin}})$ corresponds to a GP posterior on function $f_{w_{\text{lin}}}(\mathbf{x})$

is introduced with the following mean and covariance functions:

$$\begin{aligned} &m_{\text{lin}}(\mathbf{x}) := f_{w_{\text{lin}}}(\mathbf{x}),\ k_{\text{lin}}(\mathbf{x},\mathbf{x}') := \boldsymbol{\phi}(\mathbf{x})^\top \Sigma_{\text{lin}} \boldsymbol{\phi}(\mathbf{x}'), \\ &\text{where } \Sigma_{\text{lin}}^{-1} := \sum_i \boldsymbol{\phi}(x_i) \Lambda \boldsymbol{\phi}(x_i)^\top + \delta \mathbf{I}_p. \end{aligned} \tag{5.3}$$

However, the linear model has a limited ability of representation. In this paper, we mainly focus on deep reinforcement learning. Fortunately, similar to the standard weight-space to function-space conversion for linear basis-function models, Khan et al. [226] proposed an approach, DNN2GP, to convert deep neural networks to Gaussian processes.

Consider a policy $\pi_{\mathbf{w}}$ modeling by a DNN with weights $\mathbf{w}$. Within each update epoch t, the trajectories collected by $\pi_{\mathbf{w}}$ are denoted as $\mathscr{T}_t = \{s_0, a_0, r_0, ..., s_H, a_H, r_H\}$, where H is the step taken in this epoch. Then, a batch with size B is sampled from $\mathscr{T}_t$, denoted by $\boldsymbol{\tau} = \{\tau_i\}_{i=0}^{i=B}$. With the inputs $\boldsymbol{\tau}$, the policy outputs the predicted actions $\mathbf{a} = \{a_i\}_{i=0}^{i=B}$. Given a local minimum $\mathbf{w}_*$ of loss $\ell(\mathbf{w})$, which is assumed to be twice differentiable and strictly convex in π, a GP approximation can be conducted. Following DNN2GP [226], we employ a variant of the Laplace approximation with the mean $\mu_* = \mathbf{w}_*$ and covariance

$$\Sigma_*^{-1} = \sum_{i=1}^{N} \mathbf{J}_{w_*}(\tau_i)^\top \Lambda_{w_*}(\tau_i, a_i) \mathbf{J}_{w_*}(\tau_i) + \delta \mathbf{I}_p, \tag{5.4}$$

where $\Lambda_{w_*}(\boldsymbol{\tau}, \mathbf{a}) := \nabla^2 \ell(\mathbf{a}, \pi)$ is the scalar Hessian of the loss function, $\mathbf{J}_{w_*}(\boldsymbol{\tau}) := \nabla_w \pi_w(\boldsymbol{\tau})^\top$ is the $1 \times P$ Jacobian. Therefore, comparing Eqs. (5.3) and (5.4), Σ_* can be seen as the covariance of a linear model with a feature map $\boldsymbol{\phi}(\boldsymbol{\tau}) = \mathbf{J}_{w_*}(\boldsymbol{\tau})^\top$ and noise precision $\Lambda = \Lambda_{w_*}(\boldsymbol{\tau}, \mathbf{a})$. Then, DNN2GP develops an approximate GP posterior for neural networks for generic losses. For the deep reinforcement learning problem with multiple action spaces $\mathscr{A}^D$, we consider it as a function with multi-class outputs using softmax function $\mathscr{S}$. The number of categories is D, while the softmax function maps to a $D-1$ dimensional vector by ignoring the last category, ensuring identifiability. The mean and covariance

functions are:

$$\begin{aligned}
&\mathbf{m}_{w_*}(\boldsymbol{\tau}) := \mathscr{S}(\pi_{w_*}(\boldsymbol{\tau})),\\
&\mathbf{k}_{w_*}(\boldsymbol{\tau},\boldsymbol{\tau}') := \Lambda_{w_*}(\boldsymbol{\tau})\mathbf{J}_{w_*}(\boldsymbol{\tau})\Sigma_*\mathbf{J}_{w_*}(\boldsymbol{\tau}')^\top\Lambda_{w_*}(\boldsymbol{\tau}),\\
&\text{where } \Sigma_*^{-1} := \sum_i \mathbf{J}_{w_*}(\tau_i)^\top\Lambda_{w_*}(\tau_i)\mathbf{J}_{w_*}(\tau_i) + \delta\mathbf{I}_p
\end{aligned} \tag{5.5}$$

The $\Lambda_{w_*}(\boldsymbol{\tau}) = \mathscr{S}(\pi_{w_*}(\boldsymbol{\tau}))[1-\pi_{w_*}(\boldsymbol{\tau})]^\top$ is a $(D-1)\times(D-1)$ matrix and $\mathbf{J}_{w_*}(\boldsymbol{\tau})$ is the $(D-1)\times P$ Jacobian matrix. P is the number of parameters.

The GP posterior in Eq. (5.5) is expensive. To reduce the computation complexity, we model each class density by an independent GP and marginalize the latent GPs for predictions like [227]. Under this assumption, consider there are D separate GPs instead of $D-1$, the mean and covariance of GP posterior over each action $\mathbf{a}^{(d)}$ (i.e., the d-th item of $\mathbf{a}$) for the input trajectory $\boldsymbol{\tau}$ is:

$$\begin{aligned}
&\mathbf{m}_{w_*}(\boldsymbol{\tau}) := \mathscr{S}(\pi_{w_*}(\boldsymbol{\tau}))^{(d)},\\
&\mathbf{k}_{w_*}(\boldsymbol{\tau},\boldsymbol{\tau}') := \Lambda_{w_*}(\boldsymbol{\tau})^{(d)}\mathbf{J}_{w_*}(\boldsymbol{\tau})\Sigma_*\mathbf{J}_{w_*}(\boldsymbol{\tau})^\top\Lambda_{w_*}(\boldsymbol{\tau})^{(d)\top}\\
&\qquad\qquad + \Lambda_{w_*}(\boldsymbol{\tau})^{(d,d)},\\
&\text{where } \Sigma_*^{-1} := \sum_i \mathbf{J}_{w_*}(\tau_i)^\top\Lambda_{w_*}(\tau_i)\mathbf{J}_{w_*}(\tau_i) + \delta\mathbf{I}_p
\end{aligned} \tag{5.6}$$

where $\mathscr{S}(\pi_{w_*}(\boldsymbol{\tau}))^{(d)}$ is the d-th class of the output of the softmax function, $\Lambda_{w_*}(\boldsymbol{\tau})^{(d)\top}$ is the d-th row of the Hessian matrix, $\Lambda_{w_*}(\boldsymbol{\tau})^{(d,d)}$ is the d,d-th element of the Hessian matrix and $\mathbf{J}_{w_*}(\tau)$ is the Jacobians with size $D\times P$. Then, the kernel matrix of the multi-output network is initialized as a block diagonal matrix with size $B\times B$ for each class, allowing us only to compute the inverses of each diagonal block. This simplifies the computational complexity of the full matrix with size $D\times B\times B\times D$. Finally, the GP approximate of a policy modeled by a DNN is denoted as $\mathscr{GP}(\mathbf{m}_{w_*}(\boldsymbol{\tau}),\mathbf{k}_{w_*}(\boldsymbol{\tau},\boldsymbol{\tau}'))$.

5.3.1.2 Surprise-based Change Detection

With the converted GPs, previous work denoted the surprise by different metrics, such as Euclidean distance [16] and symmetrized KL divergence [17]. However, Euclidean

distance may become less informative due to the curse of dimensionality, and the KL divergence is ill-defined with non-overlapping support [228]. To overcome the weakness and limitations, our proposed method, FDA, utilizes the 2-Wasserstein Distance [223], which is symmetric and finite in all cases, allowing us to measure Bayesian surprise accurately.

During on-policy RL training, a policy π interacted with the environment and collected trajectories $\boldsymbol{\tau} \sim \mathcal{T}_t$ at epoch t. By updating using these trajectories, the converted GP at epoch t is $\mathcal{GP}(\mathbf{m}_t(\boldsymbol{\tau}), \mathbf{k}_t(\boldsymbol{\tau}, \boldsymbol{\tau}'))$, where $\boldsymbol{\tau} \in \mathcal{T}_t$. This GP contains the information of the newly collected trajectories $\boldsymbol{\tau} \in \mathcal{T}_t$. Given a model prior $\mathcal{GP}(\mathbf{m}_0, \mathbf{k}_0)$ pre-trained on random environments, we denote the surprise brought by the newly observed trajectories $\mathcal{T}_t$ is denoted using 2-Wasserstein Distance:

$$\begin{aligned} \mathbf{S}_t = \mathrm{WD}[\mathcal{GP}(\mathbf{m}_{0,\boldsymbol{\tau}}, \mathbf{k}_{0,\boldsymbol{\tau}}), \mathcal{GP}(\mathbf{m}_{t,\boldsymbol{\tau}}, \mathbf{k}_{t,\boldsymbol{\tau}})], \\ \text{where } \boldsymbol{\tau} \in \mathcal{T}_t. \end{aligned} \tag{5.7}$$

With multivariate Gaussian distribution, there is a closed form:

$$\begin{aligned} \mathbf{S}_t = \{S_t^{(i)}\}_{i=0}^{i=B} = \{\|\mathbf{m}_{t,i} - \mathbf{m}_{0,i}\|_2^2 + \\ \mathrm{trace}(\mathbf{k}_{t,i} + \mathbf{k}_{0,i} - 2(\mathbf{k}_{0,i}^{\frac{1}{2}} \mathbf{k}_{t,i} \mathbf{k}_{0,i}^{\frac{1}{2}})^{\frac{1}{2}})\}_{i=0}^{i=B}. \end{aligned} \tag{5.8}$$

The mean and variance are evaluated on a batch sampled from the newly collected $\tau_i \in \mathcal{T}_t$. It is noted that for multivariate problems with D classes, we use D individual separate GP to calculate the distance to get a $S_t^{(i)}$ with size $1 \times D$. Hence, $\mathbf{S}_t$ has the size of $B \times D$ with the input batch of size B.

As mentioned before, if the environment changes, there will be a large surprise between the predictions of the GP prior and posterior; conversely, the surprise will not increase significantly. Therefore, with a fixed prior, the surprise from trajectories collected in a stationary environment is predictive. In other words, if the environment changes at epoch t, the surprise will significantly differ from the previous surprises. Inspired by this, we perform a statistical test between the values $\mathbf{S}_t$ for the current

epoch and those from the previous epoch $\mathbf{S}_{t-1}$ before updating the current policy. Using a statistical test avoids manually selecting a threshold and provides a p-value indicating the decision-making. A reasonable choice is Welch's t-test, which does not assume equal variances between the two groups, making it more robust when the two samples have unequal variances. The null hypothesis is given by:

$$H_0 : \mathbf{S}_t = \mathbf{S}_{t-1}, \tag{5.9}$$

indicating that all samples come from the same distribution. For $\mathbf{S}_t$ and $\mathbf{S}_{t-1}$ both with sizes $B \times D$, Welch's t-test defines the statistic by the following formula:

$$\mathbf{t} = \frac{(\bar{\mathbf{S}}_t - \bar{\mathbf{S}}_{t-1})\sqrt{B}}{\sqrt{\sigma_t^2 + \sigma_{t-1}^2}} \tag{5.10}$$

where σ_t and σ_{t-1} denote the sample standard error. The degrees of freedom for the test can be approximated as:

$$\nu \approx \frac{\left(\sigma_t^2 + \sigma_{t-1}^2\right)^2 (B-1)}{\sigma_t^4 + \sigma_{t-1}^4} \tag{5.11}$$

The p-value can then be determined by comparing the calculated t-statistic to the t-distribution with ν degrees of freedom. Given a significance level α, when $p < \alpha$, we reject the null hypothesis, i.e., making the decision that there is a change point at epoch t.

5.3.2 Adapting with Functional Regularizations

Upon detecting environmental changes, a mechanism that adapts based on previously acquired knowledge is crucial for acquiring higher cumulative rewards. One popular approach is to keep some network weights to retain some general knowledge or to make the outputs close to the values for previous tasks [8, **?**]. However, when optimizing a neural network, weight change might serve as a poor proxy for the change in function [222]. *Weight regularization* may not always ensure the quality and effectiveness of previous

knowledge. In contrast, *functional regularization* prioritizes the neural network's learned function by managing changes in how it processes data rather than merely adjusting its internal settings. It provides a more consistent learning journey without depending too much on intricate parameter adjustments.

5.3.2.1 Change-aware Functional Regularization

We design a self-adjusted functional regularization term that will be employed to adapt to new environments by providing previously learned knowledge if a change point is detected.

Initially, we follow Behavior Guided Policy Gradients (BGPG) [211] for regular RL updates when no change point is detected. For a policy $\pi_{\mathbf{w}}$, the training objective at epoch t is to maximize:

$$F(\mathbf{w})_t = \mathbb{E}_{\tau \sim \mathbb{P}_t}[\mathscr{R}(\tau)] - \mathrm{WD}(\mathbb{P}_t, \mathbb{P}_{t-1}). \tag{5.12}$$

where $\mathbb{P}_t$ is the trajectory distribution mapped by $\Phi : \Gamma \to \mathscr{E}$ from the collected trajectories $\boldsymbol{\tau} \in \mathscr{T}_t$. Similarly, $\mathbb{P}_{t-1}$ corresponds to the trajectory distribution collected by π_{t-1} before the last update.

If a change point is identified at epoch c by our surprise-based change detection method, functional regularization will be involved. According to Eq. (5.6), the neural network of the policy before change is converted as $\mathscr{GP}(\pi_{c-1} \mid \mathbf{m}_{c-1}, \mathbf{k}_{c-1})$, providing knowledge as a *functional prior*, trained over all the previous environments. The current policy π_t can be represented as $\mathscr{GP}(\pi_t \mid \mathbf{m}_t, \mathbf{k}_t)$. Using $\mathbf{a}$ to denote the vector of function values defined at trajectories from the representative memory $\mathscr{M}_R$, with a sample from $\mathbf{w} \sim q(\mathbf{w})$, a GP posterior can be inferred by $\tilde{q}_w(\boldsymbol{\pi}) = \boldsymbol{N}(\boldsymbol{\pi} \mid \mathbf{m}_t(\mathbf{w}), \mathbf{k}_t(\mathbf{w}))$, where the mean and kernel can be deviated by evaluating $\mathscr{GP}(m_t(\boldsymbol{\tau}), k_t(\boldsymbol{\tau}, \boldsymbol{\tau}'))$ at the representative trajectories. Denoting a sample from $q_{c-1}(\mathbf{w})$ by $\mathbf{w}_{c-1}$, we can obtain another GP posterior, which is used as a functional prior $\tilde{q}_{\mathbf{w}_{c-1}}(\pi) = \mathscr{N}(\pi \mid \mathbf{m}_{c-1}, \mathbf{k}_{c-1})$. Following Pan et al. [16],

we add a functional regularization $\mathbb{E}_{q(\mathbf{w})}[\log q_{c-1}(\mathbf{w})] \approx \mathbb{E}_{\tilde{q}(\mathbf{a})}[\log \tilde{q}_{c-1}(\mathbf{a})]$, which has a closed-form expression:

$$\begin{aligned}\max_{\mathbf{w}} \Big[& F(\mathbf{w})_t - \delta \cdot \frac{1}{2} \big[\operatorname{Tr}\left(\mathbf{k}_{c-1}^{-1} \mathbf{k}_t(\mathbf{w})\right) \\ & + (\mathbf{m}_t(\mathbf{w}) - \mathbf{m}_{c-1})^\top \mathbf{k}_{c-1}^{-1} (\mathbf{m}_t(\mathbf{w}) - \mathbf{m}_{c-1}) \big] + \text{constant} \Big].\end{aligned} \tag{5.13}$$

The regularization plays the role of keeping the core knowledge useful for the following environments, and the coefficient δ is self-adjusted according to different change extents. This coefficient is used to achieve *change-aware functional regularization* that can adjust regularization according to different change extents. The reason is that employing function transfer directly to avoid catastrophic forgetting is not the core objective in non-stationary environments. Completely keeping previous knowledge may not always contribute to the current environmental conditions. Thus we involve a self-adjusted coefficient $\delta = |1/(\overline{\mathbf{S}_c} - \overline{\mathbf{S}_{c-1}})|$, where epoch c is the detected change point. This coefficient indicates the relative surprise brought by the new environment and is inversely proportional to the change level. The flexible coefficient ensures different responses to various conditions, making the regularization act as a *contrastive term* to the constraint $\mathrm{WD}(\mathbb{P}_t, \mathbb{P}_{t-1})$ in Eq. (5.12). If the difference of surprise is significant, i.e., the current environment is significantly different from the previous one, the functional regularization will be weakened. In contrast, more knowledge is retained when the new environment is similar to the previous one. This ensures our adaptation mechanism is sensitive to different change levels.

To reduce the computational complexity, there are some approximations: 1) following [16], we use the mean of $q_{c-1}(\mathbf{w})$ instead of a sample $\mathbf{w}_{t-1}$; 2) we ignore the derivative concerning $\mathbf{k}_t(\mathbf{w})$; 3) following [226], we set $\mathbf{w} = \boldsymbol{\mu}$; 4) we use a diagonal $\boldsymbol{\Sigma}$ which corresponds to a mean-field approximation, reducing the cost of inversion. Then, the objective

is:

$$\begin{aligned} &\max_{\mathbf{w}} F(\mathbf{w})_t \\ &-\delta \cdot \frac{1}{2} \sum_{d \in \mathscr{A}^D} (\mathbf{m}_{t,d} - \mathbf{m}_{c-1,d})^\top \mathbf{k}_{c-1,d}^{-1} (\mathbf{m}_{t,d} - \mathbf{m}_{c-1,d}), \end{aligned} \tag{5.14}$$

where $\mathscr{A}^D$ is the action space, $\mathbf{m}_{t,d}$ is the vector of $\mathbf{m}_t(\boldsymbol{\tau})$ for action d evaluated at $\{\tau_i\} \in \mathscr{M}_R$, $\mathbf{m}_{c-1,d}$ is the vector of $\mathbf{m}_{c-1}(\boldsymbol{\tau})$ for action d, and $\mathbf{k}_{c-1,d}$ is the kernel matrix from the prior for action d over the trajectories.

As $F(\mathbf{w})_t$ is a loss function for a deep neural network, performing backward propagation on the objective is unfeasible. To address this problem, we follow [16] to compute an additional gradient to the previous neural network gradient. We can approximate the gradient of the functional regularization in Eq. (5.14) as:

$$\mathbf{g}_f = \delta \cdot \frac{1}{2} \sum_{d \in \mathscr{A}^D} (\nabla \mathbf{m}_{t,d}) \mathbf{k}_{c-1,d}^{-1} (\mathbf{m}_{t,d} - \mathbf{m}_{c-1,d}), \tag{5.15}$$

where $\nabla \mathbf{m}_{t,d}(\mathbf{w})[i] = \nabla_{\mathbf{w}}[\sigma(\pi_t(\tau_i)] = \Lambda_w(\tau_i) \mathbf{J}_w(\tau_i)^\top$, which can be easily calculated by the Jacobian and Hessian.

5.3.2.2 Selecting Representative Trajectories

Functional regularization often directly regularizes network outputs [222], which can be computationally expensive when dealing with large input data points. In addition, preserving all the trajectories from environments can lead to high storage costs. Some previous research was working on reducing the computational complexity [16, 17] by prioritizing memorable and important input samples in classification problems. Therefore, designing a novel *selection strategy* for DRL in non-stationary environments is crucial to improve efficiency. Selecting *representative* trajectories from each environment to represent the knowledge learned by the policy ensures efficient and effective adaptation. In this section, given the trajectories over time: $\mathscr{T} : \{\tau_1, \tau_2, \ldots \tau_H\}$, we denote the selection metric of each trajectory as $\mathscr{C}(\tau_i)$, which is used to sort $\forall i$ and pick up the top ones.

There are several methods of selecting trajectories from interactions. One approach is based on temporal difference (TD) error, which describes the prediction error made by the network. It has been used as the foundation for a prioritized sampling strategy [19] to enhance the RL sample efficiency. The high TD error helps the agent prioritize points the model has not yet learned. Moreover, the points with high TD errors may be noisy or not learnable by the model [229]. Therefore, the most representative trajectories should be those with low TD errors, selecting by $\mathscr{C}(\tau_i) = -(r_i + \gamma \max_{s,a\sim\tau} Q(s_i', a') - Q(s_i, a_i))$. Another method is to select trajectories according to reward. Neuroscientific research indicates that replay is more commonly associated with events that yield rewards [230]. It has also been observed that in reinforcement learning, biased sampling towards rewarding transitions may be beneficial [18]. We employ the absolute value of the future discounted return from an individual trajectory, denoted as $\mathscr{C}(\tau_i) = |R_i(\tau_i)|$. The other method considers that if the selected trajectories match the global distribution in the learned environment, it is expected to get the closest performance. To ensure a random sample across the global distribution, reservoir sampling [20] is a suitable strategy that assigns a random value $\mathscr{C}(\tau_i) \sim \mathscr{N}(0,1)$ to each trajectory. The probability of every trajectory $\sqcup_i$ to add to the representative memory $\mathscr{M}$ is calculated as $\forall \tau_i, P(\tau_i \in \mathscr{M}) = \min\left(1, \frac{|M|}{t}\right)$, where t is the given moment and $|M|$ signifies the total count of representative trajectories. These methods are designed to enhance policy learning and sampling efficiency rather than adaptation when change occurs. Therefore, we design a novel selection strategy aimed at adapting to new environments based on previous knowledge.

Inspired by selecting memorable points in classification problems [231], we consider that trajectories close to the policy's decision boundary (between choosing one action over another) are highly influential. In other words, these trajectories can significantly vary in different decision regions. For the MAP estimation, the examples with a high

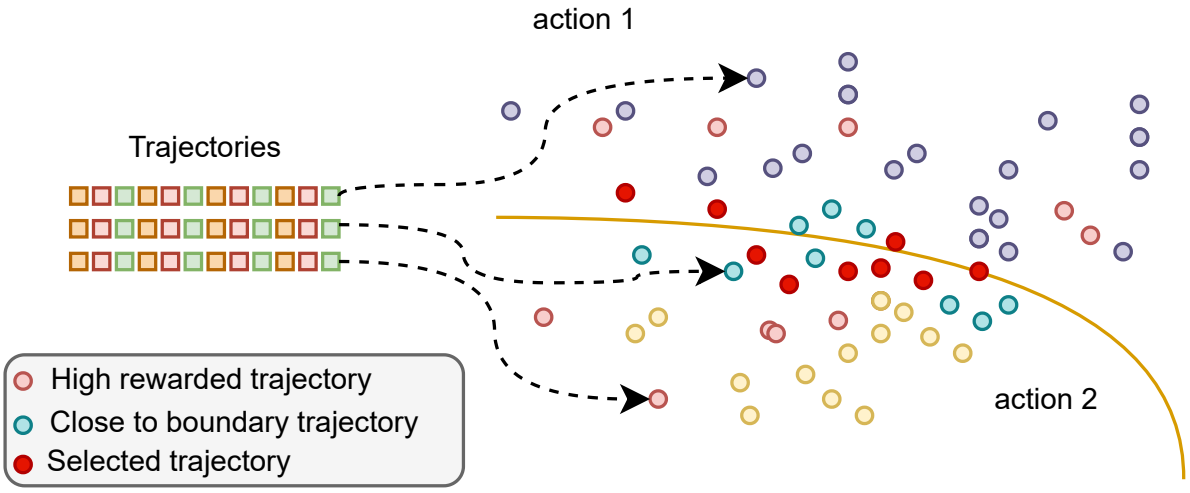


Figure 5.2: We select trajectories that are not only closest to the decision boundaries but also with the highest cumulative rewards. The orange curve denotes the decision boundaries of action 1 and action 2.

value of noise precision Λ_i contribute more to the decision-making according to the objective: $\mathbf{w}_{\text{MAP}} = \arg\max_w \Sigma_{i=1}^{N} \Lambda_i \ell(a, \pi)$. In other words, trajectories with large Λ_i are close to the decision boundary. These ideas are widely used in the theory of leverage-score sampling [232] to identify the most influential examples. Similarly, in Eq. (5.6), the quantity $\Lambda_{w_*}(\tau_i)$ plays the same role as the noise precision Λ [16]. Therefore, $\mathscr{C}(\tau_i) = \Lambda_{w_*}(\tau_i)$ can be used to select the trajectories close to the decision boundary.

Further, building upon the foundation of proximity to decision boundaries, we aim to identify more valuable trajectories. For RL problems, the reward directly denotes the policy's performance. Hence, prioritizing trajectories trajectory with higher rewards directly aligns with the goal of reinforcement learning: maximizing cumulative rewards. This can lead to the best representation of past environments. Therefore, we select sequences that are not only closest to the decision boundaries but also have the highest cumulative rewards (as Fig. 5.2 demonstrated), which is denoted as

$$\mathscr{C}(\tau_i) = \Lambda_i + |R_i(\tau_i)|. \tag{5.16}$$

We consider the selection method can pick the trajectory that carries massive information and higher value in a reinforcement learning environment. These trajectories can provide core function inputs from previous learning and guide adaptation that is sensitive to environmental change.

In the implementation, we retain a *representative memory* $\mathscr{M}_R$ of size M to store the selected trajectories before the change occurs. At epoch t with no change point detected, the top M trajectories with highest $\mathscr{C}(\tau_i)$ in Eq. (5.16) in $\mathscr{M}_R \cup \boldsymbol{\tau}$, where $\boldsymbol{\tau} \in \mathscr{T}_t$ will be selected as the most representative trajectories of this environment. With the selected trajectories, the regularization in Eq. (5.14) will be calculated only on the representative memory, significantly reducing the computation cost.

5.3.3 Computational Complexity Analysis

The proposed Functional Detection and Adaptation (FDA) algorithm is illustrated in Alg. (4). Upon traditional RL training, we add detection and adaptation procedures.

For the surprise measurement, according to Eq. (5.8), there is a complexity of $\mathscr{O}(DB^3)$ due to the matrix multiplication, where D is the action size and B is the batch size. This increases linearly in action space size and is feasible when the batch size is not too large. Welch's t-test involves $\mathscr{O}(B)$ calculation.

For the functional adaptation, as Eq. (5.15) shows, every iteration requires functional gradients, the cost of which is dominated by the computation of $\mathbf{J}_w(\tau)$ at all $\tau \in \mathscr{M}_R$. Assuming the size of the representative points is M, this adds an additional $\mathscr{O}(DMP)$ computation, where P is the number of parameters. Selecting the most representative trajectories requires a forward pass over $M+B$ trajectories, followed by picking the top M samples. In addition, for each adaptation procedure, the functional prior on representative trajectories needs to be calculated *only once* with a cost $\mathscr{O}(M^3)$.

Algorithm 4 Functional Detection and Adaptation (FDA)

Initialize: Policy $\pi_{\mathbf{w}}$, $\mathcal{GP}(\mathbf{m}_0, \mathbf{k}_0)$, Representative memory $\mathcal{M}_R$ and significance level α.

for Epoch $t = 1, 2, ...$ **do**
  Collect $\mathcal{T}_t = \{s_0, a_0, r_0, ..., s_H, a_H, r_H\}$ using the policy from the environment.
  Select representative trajectories τ_i according to Eq. (5.16) from $\mathcal{T}_t \cup \mathcal{M}_R$ and save to $\mathcal{M}_R$.
  // `Change detection (Alg. (5)).`
  $(\mathbf{m}_t, \mathbf{k}_t^{-1}), \mathbf{S}_t, p$ =`Detect`$(\pi_{\mathbf{w}}, \mathcal{T}_t, \mathbf{m}_0, \mathbf{k}_0, \mathbf{S}_{t-1})$
  if p$\leq \alpha$ **then**
    Mark epoch t as changed epoch c.
    Initialize $(\mathbf{m}_{c-1}, \mathbf{k}_{c-1})$ using $\mathbf{m}_{t-1}, \mathbf{k}_{t-1}^{-1}$.
    Compute functional regularize gradient $\mathbf{g}_f$ according to Eq. (5.15).
    Update policy parameter $\mathbf{w}$ following Eq. (5.13).
  else
    Update policy parameter $\mathbf{w}$ following Eq. (5.12).
  end if
  Update the surprise $\mathbf{S}_{t-1} \leftarrow \mathbf{S}_t$.
  Save GP $(\mathbf{m}_{t-1}, \mathbf{k}_{t-1}^{-1}) \leftarrow (\mathbf{m}_t, \mathbf{k}_t^{-1})$.
end for

Algorithm 5 Surprise based Change Detection

`Function Detect`$(\pi_{\mathbf{w}}, \mathcal{T}_t, \mathbf{m}_0, \mathbf{k}_0, \mathbf{S}_{t-1})$
  Get $\mathbf{m}_t, \mathbf{k}_t^{-1}$ for $\pi_{\mathbf{w}}$ according to Eq. (5.6).
  Compute surprise $\mathbf{S}_t$ according to Eq. (5.8).
  Perform Welch's t-test on $(\mathbf{S}_t, \mathbf{S}_{t-1})$ then get p-value p.
  return $(\mathbf{m}_t, \mathbf{k}_t^{-1}), \mathbf{S}_t, p$.

5.4 Experiment and Analysis

This section thoroughly examines our method, addressing critical questions: 1) Can our method obtain higher cumulative rewards in non-stationary environments with unknown change points? 2) Is Wasserstein surprise-based change detection more effective than other detection methods? 3) Does our functional regularization surpass retraining, weight regularization and other adaptation strategies? 4) Can our selection strategy outperform other selection strategies? 5) Can FDA maintain performance in frequently changing environments? These questions guide our experiments and analysis.

5.4.1 Experiment Settings

Environments We perform our experiment in ViZDoom [204] with changing conditions:

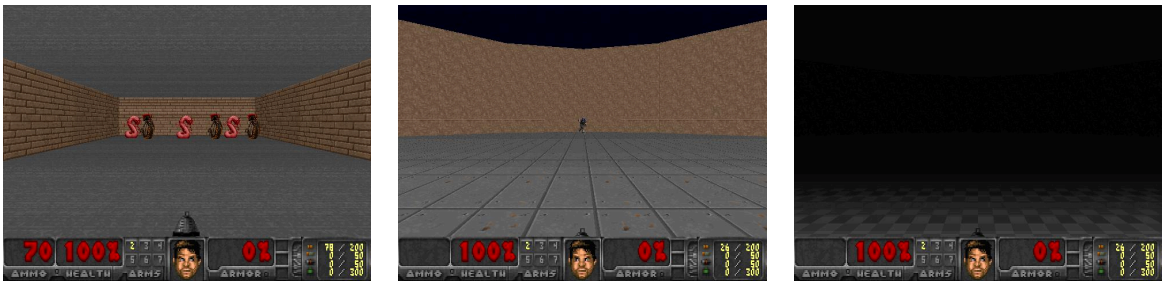
- ***basic*/*simpler_basic*:** In *basic* scenario, a player faces off against a randomly spawned monster. The player can go left/right and shoot to kill the monster. The *simpler_basic* scenario works the same but uses different textures for better contrast to help the agent learn faster. We simulate the environment change by changing from *simpler_basic* to *basic* with dimmer lighting, as Fig. 5.3 shows.

- ***defend_the_line*/*defend_the_center*:** In these two scenarios, a player is spawned along the longer wall of a rectangular map, facing three melee-only and three shooting monsters on the opposite wall. These monsters will respawn after a certain period. In the altered environment, the agent's objective shifts to eliminating surrounding enemies at the center of a circular map. As shown in Fig. 5.3, we first simulated the changed environment by changing *defend_the_line* to *defend_the_center*, then turning down the lighting.

- ***deadly_corridor*:** The map is a corridor with shooting monsters on both sides. A green vest is placed at the opposite end of the corridor. The objective for the player is to get the vest and avoid being killed somewhere along the way. The player can choose from 7 available actions: move forward/backward/left/right, turn left/right, shoot. We simulate environment changes by reducing the number of enemies from 6 to 5 and 4.

For each environment, two change points are simulated. It is important to emphasize that the way the environment changes is not provided to the agent, and the agent has no access to know when the environment changes.

basic



defend_the_line/center

Figure 5.3: The simulated non-stationary environments are based on VizDoom. The screenshots on the upper line depict the *basic* environment with different wall textures and lighting levels. The screenshots on the bottom line show the progression from *defend_the_line* to *defend_the_center* and finally to a darker *defend_the_center*.

Comparative methods To verify the efficiency of the FDA, we choose the following methods for comparison:

- Weight Detection Adaptation (WDA), replacing our objective function in Eq. (5.14) with a Wasserstein weight regularization term.
- Standard BGPG without any detection and adaptation;
- CRL-Unsup [13] with both detection and adaptation.

It is noted that BGPG [211] is used as the base algorithm for all methods. In the experiments, all algorithms are model-free and have the same setting unless otherwise specified, such as the policy neural network structure, learning rate, etc.

For detection methods, we choose:

Table 5.1: The F1-Score of detection methods. The average and standard deviation are based on ten runs with different seeds. For all methods, the points detected in no more than 5 epochs after the real change points are considered correct ones.

	basic	deadly_corridor	defend_the_line/center
WMMD	0.49 ± 0.28	0.25 ± 0.08	0.29 ± 0.05
ODCP	0.52 ± 0.07	0.47 ± 0.18	0.53 ± 0.17
CRL-Unsup	0.64± 0.17	0.59± 0.16	0.63 ± 0.25
FRCL	0.44± 0.29	0.38 ± 0.13	0.51 ± 0.24
FROMP	0.60 ± 0.09	0.61 ± 0.28	0.45 ± 0.25
FDA	**0.75 ± 0.30**	**0.69 ± 0.18**	**0.73 ± 0.13**

- CRL-Unsup [13], which detects environment change by monitoring reward.

- Online parametric Dirichlet change point (ODCP) algorithm [14, 70]; This algorithm does not have an adaptation component, so we use a standard restart procedure.

- FRCL [17], which uses KL divergence to measure Bayesian surprise.

- FROMP [16], which uses Euclidean distance to measure Bayesian surprise.

- A two-sample test using weighted maximum mean discrepancy (WMMD) [220].

Evaluation metrics Any RL method aims to obtain a high accumulative reward and continuous interaction between agents and environments. The most important metric in all experiments is how much/high rewards each method can get. We actively detect the possible changes because we believe we can obtain more rewards if we can quickly detect and then adapt to the changed MDP. For detection, we use F1 Score $\mathrm{F}_1 = \frac{2*P*R}{P+R}$ as the metric, where P is precision and R is recall.

5.4.2 Main Results

Change Detection Tab.5.1 lists the F1 scores for all the methods. As shown, our Wasserstein surprise-based method outperforms other methods in all scenarios. First, WMMD and ODCP show unsatisfactory results, especially in *deadly_corridor* and *defend_the_line*/*center*, where the distribution of trajectories might not differ significantly. CRL-Unsup performs better than the distribution-based methods; however, the size of long-term and short-term windows and the threshold of decision need to be carefully tuned. By contrast, the FDA does not require manually adjusting hyperparameters and simultaneously provides a significance level. Further, the relatively poorer performance of FRCL based on KL divergence compared to our Wasserstein-based approach can be attributed to the definition of KL divergence. In scenarios where the probability distributions have non-overlapping supports, KL divergence is infinite, so it is hard to measure the difference between distributions accurately. By contrast, Wasserstein distance is based on the optimal transport problem, denoting the minimum “cost" of turning one distribution into another. It is particularly beneficial in non-stationary reinforcement learning environments, which often feature abrupt and significant change. FROMP uses the Euclidean distance to denote the surprise, which may not perform well with high-dimensional data as it fails to capture more complex relationships between data points.

Cumulative Rewards Fig. 5.4 demonstrates the superior performance of our method, FDA (illustrated in red), especially notable after environmental change points (indicated by vertical dashed lines in each graph). This enhanced performance underscores the effectiveness of our adaptation regularization, which swiftly adjusts the policy away from its previous optimum in response to new environmental conditions.

In the *basic* scenario, where environmental changes involve adjustments to lighting and wall textures, methods lacking adaptation mechanisms exhibit a significant decline

in performance. BGPG, following the plain RL update, trajectories a sharp decline post-change, indicating poor adaptability to the new environment conditions. In contrast, the CRL-Unsup method showcases considerable adaptability with a consistent reward increase post-change, although it was still not as effective as FDA. This performance differential highlights the advantages of our behavior-based regularization term, which promotes quicker adaptation to novel environments. Moreover, the FDA employing functional regularization outperforms the FDA with weight regularization, suggesting that functional features offer a more holistic representation of environmental knowledge. In the *deadly_corridor* scenario, characterized by an increased number of enemies, with most competing methods struggling with local optima. WDA and CRL-Unsup fail to maintain performance like BGPG, indicating that regularizing by weight may lead to worse performance than following the RL objective. In contrast, the FDA excels by adapting to intensified challenges effectively, demonstrating its robustness against complex dynamic changes in the environment. The *defend_the_line*/*center* environment, marked by a substantial shift in objectives, also sees a notable dip in rewards for all methods except FDA. Our method not only minimizes performance degradation but also adjusts more rapidly to the new objectives, emphasizing its capability to handle drastic shifts in environmental demands effectively. WDA with a weight regularization shows a significant drop in performance after the change and does not recover, indicating that weight regularization is less effective in scenarios where quick adaptation to new objectives is crucial. CRL-Unsup mirrors the decline in other methods but shows a minor recovery later. Finally, BGPG performs poorly across the board in this scenario, with a sharp decline and no significant recovery, indicating a struggle with rapid adaptation to new game objectives.

Overall, these results affirm that FDA stands out in its ability to adapt to diverse and dynamically changing environments, significantly outpacing traditional methods

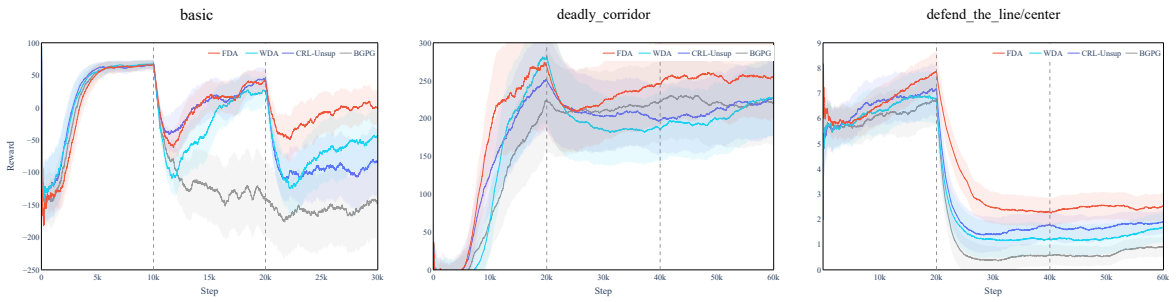


Figure 5.4: Reward curves in simulated non-stationary environments. The line is the average of different seeds for ten runs. The shaded area denotes the standard deviation. Our proposed method, FDA, is denoted by red curves.

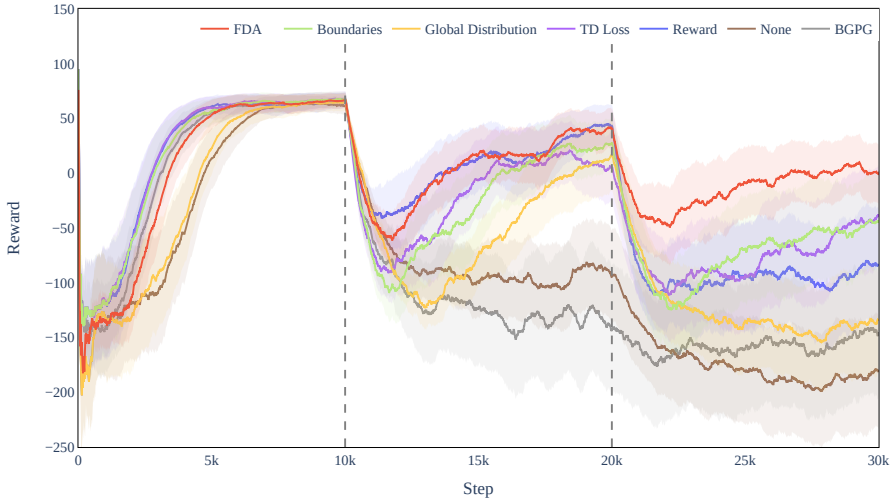


Figure 5.5: Cumulative rewards of different selecting strategies used for adapting using functional regularization. The line is the average of different seeds for ten runs. The shaded area denotes the standard deviation.

that fail to incorporate effective adaptation strategies.

5.4.3 Ablation Study

Selection Strategy Our experiments assessed the effectiveness of the novel selection strategy designed to prioritize trajectories close to decision boundaries and exhibit high cumulative rewards. Fig. 5.5 visually illustrates the impact of selecting high-rewarded boundaries as a criterion for trajectory selection in reinforcement learning environments.

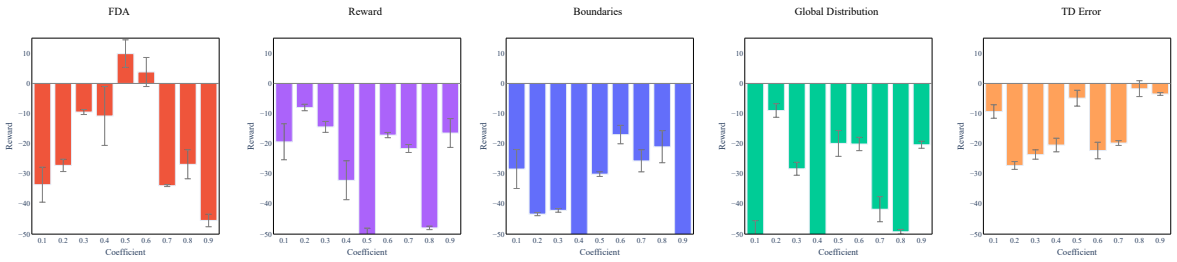


Figure 5.6: The average rewards of different regularization coefficient δ in Eq. (5.13). The error bar denotes the standard deviation over five runs.

The effectiveness of the FDA is evident, highlighting its potential to enhance the efficiency and efficacy of adaptation processes significantly. By concentrating on trajectories near decision boundaries with high rewards, our agent demonstrated a remarkable ability to adapt to environmental changes swiftly, yielding outcomes that surpassed those achieved by other selection methods. This strategic focus not only reduces the computational complexity of functional regularization but also ensures that the agent concentrates on trajectories that are the most informative and valuable for adaptation.

Adaptation Efficiency Tab. 5.2 compares average rewards obtained by different methods, illustrating the adaptation capabilities of each. Our method, FDA, consistently achieves higher average rewards than FDA-R (Restart) and BGPG. FDA-R (Restart), which resets the training to follow a basic objective denoted by Eq. (5.12) upon detecting an environmental change, registers lower average rewards. This method's approach of restarting training may not efficiently utilize previously acquired knowledge, leading to suboptimal performance in adapting to new conditions. In contrast, the plain BGPG method, which does not incorporate adaptive or restarting mechanisms, records the lowest average rewards in three tested environments. This indicates a significant disadvantage when the method lacks the flexibility to adjust to environmental changes. The superior performance of the FDA demonstrates its robust adaptation ability, significantly outperforming methods that rely solely on standard reinforcement learning

Table 5.2: The average rewards of our method and BGPG-Restart. Our method achieves higher reward than simply following plain RL update and restarting a new training using RL objective. The results are based on ten runs with different seeds.

	basic	deadly_corridor	defend_the_line
BGPG	-58.92±35.62	153.16±32.92	4.89±0.11
FDA-R(Restart)	-35.74±40.23	203.83±28.46	3.54±0. 41
FDA	**10.28±21.98**	**255.78±22.23**	**5.83±0.33**

updates or that restart training from scratch upon each environmental shift. FDA leverages accumulated knowledge from past environments, effectively adapting to new challenges and variations, which results in consistently higher average rewards across different scenarios. Overall, the data in Tab. 5.2 underscores the effectiveness of our FDA method in handling non-stationary environments. It achieves commendable adaptation performance without restarting the training process from the beginning or depending exclusively on basic DRL updates. This capability makes the FDA preferable for environments where conditions frequently change, highlighting its potential in dynamic and complex settings.

Parameter Sensitivity Fig. 5.6 shows the average rewards obtained by different regularization coefficients δ in Eq. (5.13). The y-axis displays the average reward, while the x-axis represents the coefficient values ranging from 0.1 to 0.9. Our method, FDA, is highlighted with red bars and consistently scores the highest average rewards for most coefficient settings. This performance underscores the FDA's ability to effectively select representative trajectories from environments that have been thoroughly learned. It utilizes this information to transfer knowledge from past environments, optimizing current policies. Selecting trajectories based on reward, boundaries, global distribution and td error performs relatively well for certain coefficient values but generally yields lower average rewards than FDA. The error bars represent the standard deviation across multiple experimental runs, reflecting the variation in results. This variation indicates

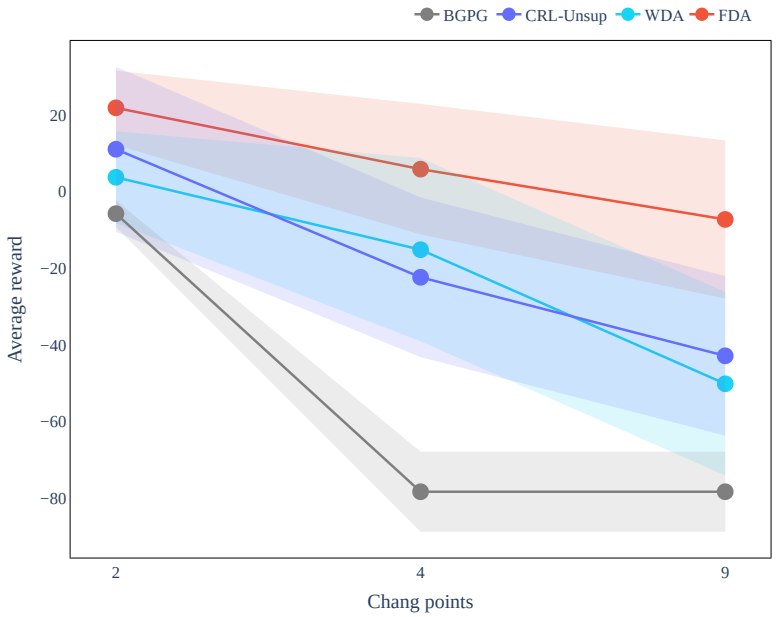


Figure 5.7: The average rewards of different methods in increasingly non-stationary environments. The results are based on 5 runs with different seeds.

Table 5.3: F1 scores for change detection methods across environments with increasing change points.

	2 changes	4 changes	9 changes
CRL-Unsup	0.71±0.13	0.60±0.17	0.37±0.08
WMMD	0.45±0.17	0.40±0.11	0.28±0.19
ODCP	0.25±0.15	0.21±0.11	0.20±0.13
FRCL	0.44± 0.29	0.32 ± 0.07	0.31 ± 0.10
FROMP	0.60 ± 0.09	0.43 ± 0.26	0.27 ± 0.08
FDA	**0.75±0.30**	**0.63±0.13**	**0.53±0.29**

the reliability and robustness of each method under different conditions. In summary, the figures demonstrate that the FDA surpasses other trajectory selection strategies across a broad range of regularization coefficients. This highlights its capability as a superior method for leveraging past learning to enhance decision-making in new environments.

Change Frequency When the environment changes *very gradually*, there will be no significant changes, and then our FDA will behave the same as the standard RL

(e.g., BGPG used in the paper). Notably, these environments could be adapted well by the standard RL. In contrast, when the environment is dramatically non-stationary, i.e., abrupt changes happen very often, can FDA maintain performance? Therefore, we evaluate the performance of methods in extremely non-stationary environments. Fig. 5.7 provides a comparative overview of the different algorithms' performances across the *basic* environment with varying numbers of change points. As indicated in red, the FDA consistently achieved higher average rewards than the other methods when the environment changed in more frequently changing environments, demonstrating its robustness in dealing with multiple change points. Our observations indicate that increasing the number of change points from 2 to 4 will lead to a noticeable decline in the performance of all methods. Among them, FDA trajectories have the smallest decrease, maintaining a noticeable advantage. This suggests that FDA is particularly effective in adapting to complex environments with frequent changes. Additionally, Tab. 5.3 displays the detection accuracy in environments that change often. As the number of change points increases, all methods are affected. In highly non-stationary environments, the policy might not fully adapt in each setting, resulting in trajectories displaying random patterns and failing to represent the underlying knowledge accurately. Furthermore, the detection outcomes are also compromised by the model's inability to converge. This is a limitation of our method; if the policy is not fully converged, the detection and adaptation performance will be affected. However, even under these circumstances, our approach still exhibits better performance compared to other approaches, particularly traditional DRL methods.

5.5 Summary

This paper introduces a novel functional deep reinforcement learning approach for detecting changes and adapting in non-stationary environments. We have proposed a Bayesian

surprise-based method for detecting environmental changes. This approach accurately identifies change points, enabling the agent to respond to changes in the environment's dynamics quickly. Then, we developed a functional regularization technique based on the representative trajectories and change information. This method helps the policy to adapt rapidly in non-stationary environments, ensuring robust and efficient learning. The new strategy for effectively selecting trajectories is to prioritize trajectories close to the decision boundaries and have high cumulative rewards, ensuring that the most informative and valuable trajectories are used for adaptation. The proposed framework, Functional Detection Adaptation (FDA), can provide significant advancements, particularly in dynamic and unpredictable environments. The FDA enhances the agent's ability to learn and adapt, ultimately improving performance and efficiency in various applications.

Expanding our on-policy framework to accommodate off-policy algorithms is a promising direction for future work. Further, in a multi-agent context, our detection-adaptation framework could be crucial for understanding and responding to the complex dynamics that emerge from agent interactions.

CHAPTER 6

AN APPROACH FOR LATENT DYNAMICS

This chapter addresses research objectives 1, 4, and 5 mentioned in the Chapter. 1. The primary contribution of this chapter is a model-based reinforcement learning algorithm that features a change-sensitive and adaptive latent space representation. A key novelty is that our method identifies change points in the non-stationary environment directly in the latent space, enabling online detection and adaptation. We demonstrate through extensive experiments on challenging real non-stationary environments with high-dimensional inputs, where reward distributions and state transition dynamics vary in unknown ways over time, that our algorithm achieves robust performance by quickly recognizing and adjusting to the environmental changes in the latent space.

This chapter is based on Z. Liu, J. Lu, J. Xuan and G. Zhang, "Learning Latent and Changing Dynamics in Real Non-stationary Environments," in IEEE Transactions on Knowledge and Data Engineering [under review].

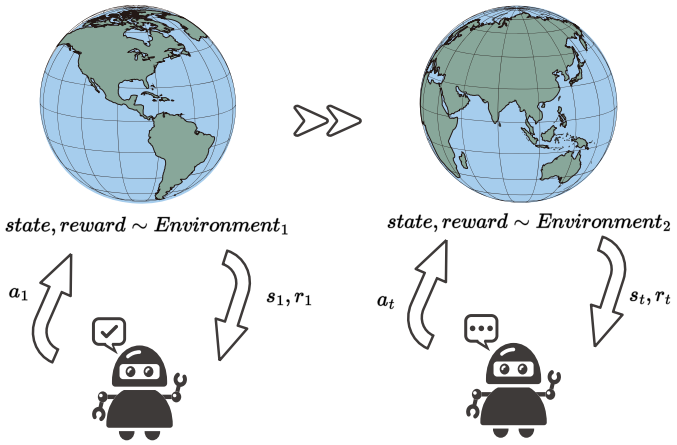


Figure 6.1: In some non-stationary environments where the transition probability and reward function change independently, a well-trained agent may find that its existing knowledge is no longer sufficient in the new environment.

6.1 Background

Although many model-free RL algorithms perform excellently in various applications, they offer high sample complexity and low sample efficiency in some environments like robotics or video games [31]. That is to say, model-free RL algorithms often need to take tens of millions of steps[21] to train a policy that is good enough, which is impractical in many open-world applications, especially health and safety-related scenarios. Unlike direct policy optimization in model-free RL, model-based RL uses an indirect learning method that learns and maintains a high-quality environmental planning and prediction model known for higher sample efficiency [233], e.g., video game[234], biological sequence design[235], and robotics [236].

Model-based RL performance is highly dependent on the quality of the learned environment model throughout interactions. A common assumption is that the environment remains fixed during the entire course of interaction, i.e., the reward distributions and state transitions do not vary in time. This assumption does not always hold in practice, as Figure 6.1 indicates. On the contrary, many environments are non-stationary and may change with time. For example, robots performing outdoor tasks may encounter new

and unfamiliar terrains; they may need to maintain stability in sudden gusts of wind or adapt to changes in lighting when moving from sunlight into a dark area. Additionally, various breaking news may influence the trading market, making it crucial to identify the sudden situation and adapt the current investment models and strategies to new conditions. More examples can be found in online advertisement auctions [237], dynamic pricing [238] and traffic management [239], demonstrating the importance of developing methods addressing model-based RL in non-stationary environments. The changes in such an environment are usually unpredictable and may arrive in rapid succession. Even well-trained agents will fail in non-stationary environments because their focus is learning through interaction without considering environmental changes and unexpected perturbations. Once the environment with which an RL agent interacts changes, its original strategies or models no longer work. In other words, traditional model-based RL agents cannot adapt to new variations rapidly, even for simple problems [240]. Hence, quickly adapting to possible environmental changes is vital for a robust model-based RL.

Recently, some research [11, 62, 241] efforts generalized model-based RL to non-stationary environments by considering it as a serial transition in various Markov Decision Processes (MDPs). This was a bold initiative but still not good enough because transition timing between MDPs needs to be given in advance, which is also impractical sometimes. For example, a delivery drone does not know wind changes in advance, and a chatbot does not know when a user starts a new topic. These context changes, which we identify as *real non-stationary environments* (a formal definition is given in the next section), are unpredictable and unexpected. Although RL agents can gradually adapt to slightly changing environments during continuous interactions, they experience significant reward drops when sudden changes occur. Post-change learning phases are particularly challenging due to the lack of new data, making it difficult to escape local minima. Therefore, detecting change points is crucial for fast performance recovery.

Actively identifying changes allows us to use change information, providing valuable knowledge for adaptation. Additionally, detecting change points enables the model to respond promptly to the new environment. Following the original RL update means the system will take a long time to achieve good performance, highlighting the necessity of an adaptation scheme for this problem. With the detection and adaptation ability, we aspire for agents to recognize changes and promptly adapt by re-purposing knowledge and information as humans do.

To solve this challenging problem, this chapter proposes a new solution named Learn Latent and Changing Dynamics (LLCD) in real non-stationary environments. More specifically, since some environments with image or sensor input typically do not constitute a Markovian and compact space [22] and are often characterized by their complexity and high dimensionality, identifying change within such a raw observation space is challenging. As such, detecting and adapting to the changes becomes more challenging with few trials and samples. Therefore, our idea is to move model learning into a latent space, which is expected to demonstrate Markov transition properties more clearly and, at the same time, significantly reduce dimensionality compared to the raw observation space. Our method actively detects possible changes through a Bayesian online change point detection method [71] and learns environment models in the latent space, where the learned models are independent of specific situations and transfer well to other contexts. The main contributions of this work are:

- A model-based reinforcement learning algorithm, LLCD, features a change-sensitive and adaptive latent space.
- An environment change detection scheme that uses Bayesian theory in the latent space and provides predictive distributions to the new environment learning.
- A model adaptation method based on the predictions, speeding RL performance recovery when facing changed environments.

We show experimentally that our approach achieves robust performance in challenging real non-stationary environments where reward distribution and state transition vary with time in an unknown manner.

6.2 Problem Formulation

We formalize three key definitions and establish our core problem setting to clarify the objectives of this chapter and delineate the scope of the problem.

Definition 6.1 (Markov Decision Process). A Markov decision process (MDP) is a discrete-time stochastic control process with a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function and $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the state transition probability.

Given the current state s_t and the action a_t, the environment transits to the next state according to the probability distribution $p(s_{t+1} \mid s_t, a_t)$, while $r(r_t|s_t, a_t)$ returns a reward according to the state s_t and the chosen action a_t. Model-based RL aims to enable an agent to construct functional representations of $p(s_{t+1} \mid s_t, a_t)$ and $r(r_t|s_t, a_t)$ to obtain optimal trajectories that maximize cumulative reward.

Definition 6.2 (Stationary and non-stationary environment). If an environment can be completely characterized by one and only one MDP, it is called a stationary environment; if a sequence of MDPs is required, it is called a non-stationary environment.

To draw a clear line between traditional non-stationary environments, we define our target as follows,

Definition 6.3 (Real Non-stationary Environments). A real non-stationary environment is defined as a chronological sequence of MDPs $\{\mathcal{M}_k = \langle \mathcal{S}, \mathcal{A}, P_k, R_k \rangle\}$ with unknown change points $\mathcal{T}$, where $\mathcal{S}$ and $\mathcal{A}$ are the shared state and action spaces; R_k

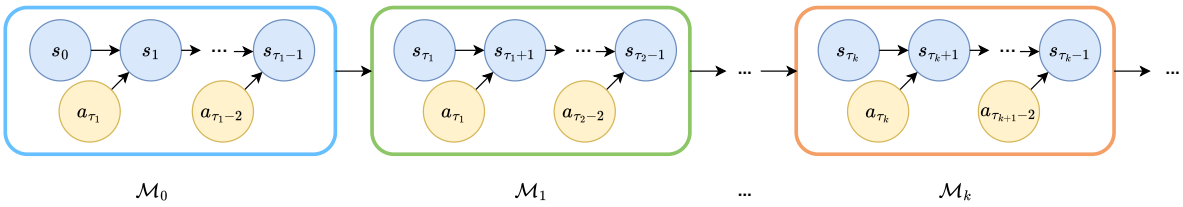


Figure 6.2: The formal MDP setting of the non-stationary environment. The switch points of MDPs are unknown.

and P_k are the reward function and the dynamics of the k-th MDP; and $\mathcal{T} = \{\tau_1, \tau_2, ...,\}$ is a non-decreasing integer series that means the environment changes from the previous MDP $\mathcal{M}_k$ to a new one $\mathcal{M}_{k+1}$ at change point τ_k.

As Figure 6.2 indicates, the changing dynamics of a real non-stationary environment can be expressed as

$$p(s_{t+1} \mid s_t, a_t) = \begin{cases} p_0(s' \mid s, a), & t < \tau_1 \\ p_1(s' \mid s, a), & \tau_1 \le t < \tau_2 \\ \vdots & \\ p_k(s' \mid s, a), & \tau_k \le t < \tau_{k+1} \\ \vdots & \end{cases} \tag{6.1}$$

and the reward function will be

$$r(s) = \begin{cases} r_0(s), & t < \tau_1 \\ r_1(s), & \tau_1 \le t < \tau_2 \\ \vdots & \\ r_k(s), & \tau_k \le t < \tau_{k+1} \\ \vdots & \end{cases}. \tag{6.2}$$

If the change points $\mathcal{T} \equiv \{\tau_1, \tau_2, ...,\}$ are not given beforehand, it degenerates into traditional non-stationary environments. It should be noted that the action space and state space of the environment are assumed to be fixed because it is straightforward to identify the environment changes when they occur.

Theorem 6.4 (RL in real Non-stationary Environments). *Assume that there is a chronological sequence of different MDPs* $\{\mathcal{M}_{k=1:K}\}$ *with change points* $\{\tau_1, \tau_2, ..., \tau_K\}$. *An agent will sequentially interact with these MDPs, where the change points are not given in advance. Through ongoing interactions, the agent needs to acquire the highest reward.*

This chapter aims to learn and maintain an MDP $\hat{\mathcal{M}}_k$ model that best suits the current environment. Then the model is used to maximize the expected sum of rewards. Hence, the objective is

$$\max_{a_{1:T-1}} \sum_k \mathbb{E}_{p_k(s_{t+1}|s_t,a_t)} \left[\sum_{\tau_k}^{\tau_{k+1}-1} r(s_t) \right] \tag{6.3}$$

where $p_k(s_{t+1} \mid s_t, a_t)$ is the transition of environment, $r(s_t)$ is the reward function, and $\{\tau_1, \tau_2, ..., \tau_{k-1}\}$ are the change points detected by our method. The core problem is accurately representing the environment through the transition and reward models.

6.3 Methodology

We aim to design a model-based RL algorithm to manage complex and high-dimensional environments with non-stationary dynamics, utilizing change point monitoring and rapid adaptation. The natural idea is to learn a model from the image observation and detect the distribution drift of the observed data. However, it has some practical problems: 1) modeling high-dimensional RL environments is challenging because some data, like images, do not always constitute a Markovian space in practice [22, 23]; 2) directly monitoring a change in the complex distribution of raw observations is difficult due to the considerable noise and sparsity of some high-dimensional data. We address these issues by modeling the raw observation using Reccurent State-space Model(RSSM) that can learn a pure latent space transition, similar to recently proposed methods [23, 242, 243, 244], where the image of each time step is considered observation and the latent states represent the underlying process that generates the observations.

6.3.1 Learning Latent Dynamics

At a discrete time step t of reinforcement learning, the tuple $\{o_t, a_t, r_t\}_{t=1}^{T}$ will be introduced, where o_t is the image observation, a_t is the action vector, and r_t is the reward that the environment feeds back. In the context of RL, at a discrete time step t, the tuple $\{o_t, a_t, r_t\}_{t=1}^{T}$ represents a sequence of observations o_t, actions a_t and the reward r_t. A model-based RL should learn the following two things:

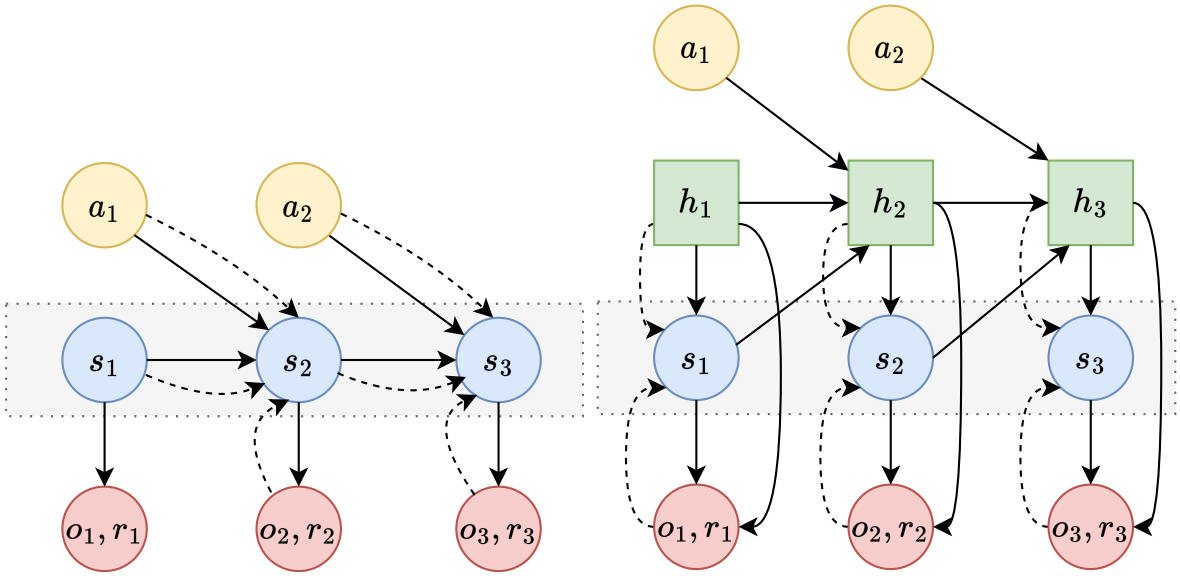
$$\begin{aligned} \text{Reward model} \quad & r_t \sim p(r_t \mid o_t, a_t) \\ \text{Transition model} \quad & o_t \sim p(o_t \mid o_{t-1}, a_{t-1}). \end{aligned} \tag{6.4}$$

Directly estimating the above two models can be challenging due to several factors. In addition to the aforementioned concerns of computational cost and the absence of the Markov property, there is also another issue, namely, image inputs may lack well-defined transition or reward function mappings, as assumed. The reason is that images typically contain a vast amount of information, making extracting relevant features or interpreting pixel-level changes as meaningful transitions difficult [245]. Therefore, we adopted a method of learning transition and reward functions in a compact Markov latent space to ensure the accuracy of the model and enable efficient planning.

A typical latent state-space model is composed by

$$\begin{aligned} \text{Reward model} \quad & r_t \sim p(r_t \mid s_t) \\ \text{Transition model} \quad & s_t \sim p(s_t \mid s_{t-1}, a_{t-1}) \\ \text{State decoder} \quad & o_t \sim p(o_t \mid s_t) \end{aligned} \tag{6.5}$$

where s_t is a latent state at time t with significantly lower dimensionality compared with o_t, and the relationship with o_t is expressed by a state decoder. As shown in Figure 6.3 (a), a hidden state sequence $\{s_t\}_{t=1}^{T}$ is used to define the generative process of the images and rewards. Given these components, a model-predictive control (MPC [246]) is used to search for the best sequence of actions based on the observations on each time step.



(a) State-space Model(SSM) (b) Reccurent State-space Model(RSSM)

Figure 6.3: The figure shows a latent dynamic model with recurrent states. In the figure, stochastic variables are depicted as circles, whereas deterministic variables are shown as squares. Solid lines are used to represent the generative process, while dashed lines signify the inference model.

We do not have any extra policy or value neural network like model-free RL. This model is purely stochastic, which helps prevent overfitting and allows for future possibilities. However, this uncertainty can make it difficult to retain information over several time steps. In most environments, rewards are often influenced by more than just short-term observations, and the effects of actions or observations may not become apparent for an extended period.

Following [23], we added a recurrent deterministic state to our model based on typical latent dynamics to address this problem. This allows the model to remember information in the time series and make accurate predictions while maintaining uncertainty. The

final model is illustrated in Figure 6.3 (b), and its components are as follows:

$$\begin{aligned} \text{State decoder} \quad & o_t \sim p(o_t \mid h_t, s_t) \\ \text{Reward model} \quad & r_t \sim p(r_t \mid h_t, s_t) \\ \text{Stochastic transition model} \quad & s_t \sim p(s_t \mid h_t) \\ \text{Recurrent state model} \quad & h_t = f(h_{t-1}, s_{t-1}, a_{t-1}), \end{aligned} \tag{6.6}$$

where the recurrent state model is normally implemented using a Recurrent Neural Network (RNN) to handle sequential data, accepting the current input data and previously received inputs, it is noted that the model uses pure deterministic variable convergences based on a single value to arrive at local minima, preventing the model from capturing multiple features. This is detrimental to exploration-dependent RL training and may disable the agent from exploring enough paths. Therefore, our model combines stochastic and deterministic variables to retain part of the uncertainty while remembering the previous input.

To estimate these distributions, we use the variational inference. The posterior state distribution is approximated by the encoder $q(s_t \mid h_t, o_t)$, and the state posterior for a whole time series is simply factorized as $q(s_{1:T} \mid o_{1:T}, a_{1:T}) = \prod_{t=1}^{T} q(s_t \mid h_t, o_t)$. It is noted that the model uses pure deterministic variable convergences based on a single value to arrive at local minima, preventing the model from capturing multiple features. This is detrimental to exploration-dependent RL training and may disable the agent from exploring enough paths. Therefore, our model combines stochastic and deterministic variables to retain part of the uncertainty while remembering the previous input.

6.3.2 Detecting Environment Changes

As stated above, the dynamics of a real non-stationary environment vary unpredictably with time. In such environments, detecting change points becomes essential for several reasons. First, sudden environmental changes can cause significant performance drops if

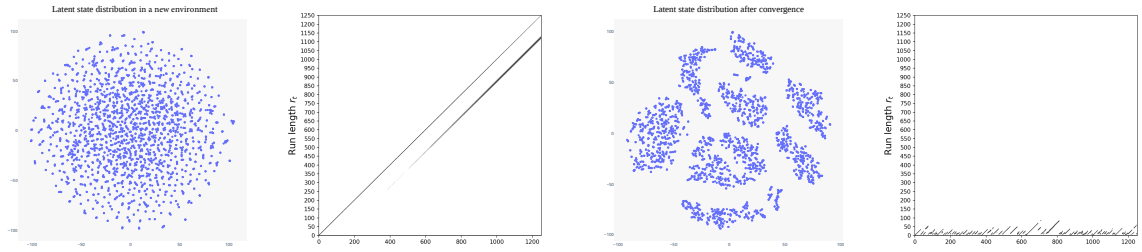
not promptly addressed. Change detection enables the model to adapt promptly when a change is detected. Additionally, change points provide valuable information about when and how the environment has changed. By identifying these points, the algorithm can adjust its policy to suit the new conditions better, leading to more effective adaptation. Therefore, the requirements of this detection method are: 1) it should be fast because we are working online, and a judgment should be made at every time step; 2) it should be able to output useful knowledge to guide the following adaption beyond the change points. In previous work [13], monitoring rewards is useful for detecting environment changes but is not applicable in some scenarios with sparse or delayed rewards. Also, the standalone reward drop cannot provide enough information to guide adaptation.

In this paper, inspired by Bayesian online change point detection [71, 247], we firstly introduce a new latent variable *run length* $l_t \in \mathbb{Z}^+$ for each time step t, which represents the time step since the last change point given the data so far observed. If a change happens at time t, we have $l_t = 0$; and the observations between two change points are assumed to follow the same distribution. To complete the Bayesian model, l_t is given a Bernoulli process prior as

$$P(l_t \mid l_{t-1}) = \begin{cases} 1/\beta & \text{if } l_t = 0, \\ 1 - 1/\beta & \text{if } l_t = l_{t-1} + 1, \\ 0 & \text{otherwise.} \end{cases} \tag{6.7}$$

which results in a geometric distribution with a mean $\beta \in \mathbb{R}_{>0}$ over partition lengths and $\beta \in \mathbb{R}^+$ is a hyperparameter. Assume that we have a sequence of latent states $\{s_1, s_2, ..., s_t\}$ inferred by the transition model. The posterior distribution of l can be derived by

$$P(l_t \mid \boldsymbol{s}_{1:t}) = \frac{P(l_t, \boldsymbol{s}_{1:t})}{P(\boldsymbol{s}_{1:t})} = \frac{P(l_t, \boldsymbol{s}_{1:t})}{\sum_{l_t} P(l_t, \boldsymbol{s}_{1:t})} \tag{6.8}$$



(a) State in a new environment (b) Detection results of data in (a). (c) State in a learned environment (d) Detection results of data in (c).

Figure 6.4: The figure shows the distribution of the latent state in 5 episodes after convergence (left) and after change points (right) in the Reacher-easy environment and the corresponding detected run length posteriors. The data is dimensionality reduced using t-distributed Stochastic Neighbor Embedding (t-SNE,[1]). When the model converges, the state shows a certain structure, while after the change point, the state distribution tends to be random.

where the joint distribution $P(l_t, \boldsymbol{s}_{1:t})$ can be written recursively as

$$\begin{aligned} P(l_t, s_{1:t}) &= \sum_{l_{t-1}} P(l_t, l_{t-1}, \boldsymbol{s}_{1:t}) \\ &= \sum_{l_{t-1}} P(l_t \mid l_{t-1}) P\left(s_t \mid l_{t-1}, \boldsymbol{s}_{t-l_t:t-1}\right) P(l_{t-1}, \boldsymbol{s}_{1:t-1}) \end{aligned} \tag{6.9}$$

then, the predictive distribution is calculated by

$$P(s_{t+1} \mid \boldsymbol{s}_{1:t}) = \sum_{l_t} P\left(s_{t+1} \mid l_t, \boldsymbol{s}_{t-l_t:t-1}\right) P(l_t \mid \boldsymbol{s}_{1:t}). \tag{6.10}$$

Following the above update procedure, we can obtain the run length posterior $P(l_t \mid \boldsymbol{s}_{1:t})$ for each time point but cannot directly use it to make the decision on environmental change or not like the classical Bayesian online change point detection [71, 247] did. The reason for this is that different from the classical Bayesian online change point detection, where the data points are assumed to be independent and identically distributed, the latent states here are sequentially dependent, and their distributions are constantly changing due to the update of the transition model in (6.6). When the model in (6.6) is well-trained in an environment, a meaningful structure will be formed in latent state space, and state transitions within this space will account

Algorithm 6 Environment Change Detection

Require: Latent states $\boldsymbol{s}_{1:T} = \{s_1, s_2, \ldots, s_T\}$, current run length sequence $\boldsymbol{L}_{\text{curr}} = \{l_0\}$, and previous run lengths $\mathscr{L}_{\text{prev}}$ which contain the maximum run length in terms of each detection window.

Initialize $isChange = False$, $l_0 = 0$, $P(l_0 = 0) = 1$, $P(s_1) \sim N(\nu_{\text{prior}}, \chi_{\text{prior}})$.
for s_t in $\boldsymbol{s}_{1:T}$ **do**
    Evaluate the predictive probability $P(s_t \mid \boldsymbol{s}_{1:t-1})$.
    Calculate the joint distribution according to Equation (6.11).
    Calculate the evidence $P(\boldsymbol{s}_{1:t})$.
    Determine run length distribution according to Equation (6.8) ,select l_t with a maximum probability and $\boldsymbol{L}_{\text{curr}} \leftarrow \boldsymbol{L}_{\text{curr}} \cup \{l_t\}$.
    Update predictive distribution statistics and perform prediction $P(s_{t+1})$ according to Equation (6.10).
end for
if $\max(\boldsymbol{L}_{\text{curr}}) > 3 * \text{Var}(\mathscr{L}_{\text{prev}})$ **then**
    $isChange = \text{True}$.
end if
$\mathscr{L}_{\text{prev}} \leftarrow \mathscr{L}_{\text{prev}} \cup \max(\boldsymbol{L}_{\text{curr}})$.
return Predictive distribution $P(\boldsymbol{s}_{1:T})$, $isChange$, $\mathscr{L}_{\text{prev}}$.

for the behavior of the RL agent. Hence, the state distributions at different time steps will be different in a large probability; that is, l_t will be near 0 in a large probability, the same as our observations (see Figure 6.4 for example). When an RL agent enters a new environment, the model trained in the previous environment will no longer behave well. The model structure will have large variations, such as transition and reward models, to adapt to this new environment. Hence, the states in this stage (from entering a new environment to being well-trained) exhibit a near-random pattern and follow the same Gaussian distribution. In other words, l_t in this stage will keep growing. According to these observations, we can detect environmental change by monitoring the run length each time. If it experiences significant growth at some time step, there will be a change; otherwise, there is no change. In the implementation, our determination method employed the 3-sigma rule [248]. The complete detection procedure is summarised in Algorithm 6, and we make the following remarks about this detection method:

Remark 6.5. The distributions in Equations (6.8) and (6.11) are all exponential family

distributions, so the sequential update is in closed form. Furthermore, the dimensionality of latent state space is significantly lower than that of raw observation space. Hence, this detection method is lightweight.

Remark 6.6. There is a byproduct from the above detection method: $P\left(s_t \mid \boldsymbol{s}_{t-l_t:t-1}\right)$ in Equation (6.11) that is the predictive distribution over the latent state. Such a distribution could assist in learning the transition model, as detailed in the following section.

At each time step, it is necessary to compute the distribution. Thus, the cost brought by using raw observation data for computation is extremely high. However, conducting detection on the latent state not only effectively improves computational efficiency but also leverages the predictive distribution to benefit model training. Figure 6.5 illustrates the structure of latent dynamics, change detection and adaptation components. It is evident that both the prediction distribution s_{t+1}, furnished by the change detection mechanism, and the inferred state s_{t+1} from the subsequent temporal dynamics encapsulate pertinent data from the prior environmental context. When environmental changes occur, leading to the lack of sufficient data utilization, the information provided by the change detection mechanism substantively facilitates the training of the model.

6.3.3 Learning latent and changing dynamics

Following [23, 22], we use the variational autoencoder to learn the complex posterior distributions in (6.6) with state decoder $p(o_t|s_t)$, encoder $q(s_t|o_t, s_{t-1})$, and reward function $r(s_t)$. The variational distributions are defined as

$$\begin{aligned} q(s_{1:t} \mid o_{1:t}, a_{1:t}) &\propto \prod_t p(o_t|s_t) q(s_t|s_{t-1}, a_{t-1}), \\ q(r_{1:t} \mid o_{1:t}) &\propto \prod_t p(r_t|s_t) q(s_t|s_{t-1}, a_{t-1}, o_t), \end{aligned} \tag{6.11}$$

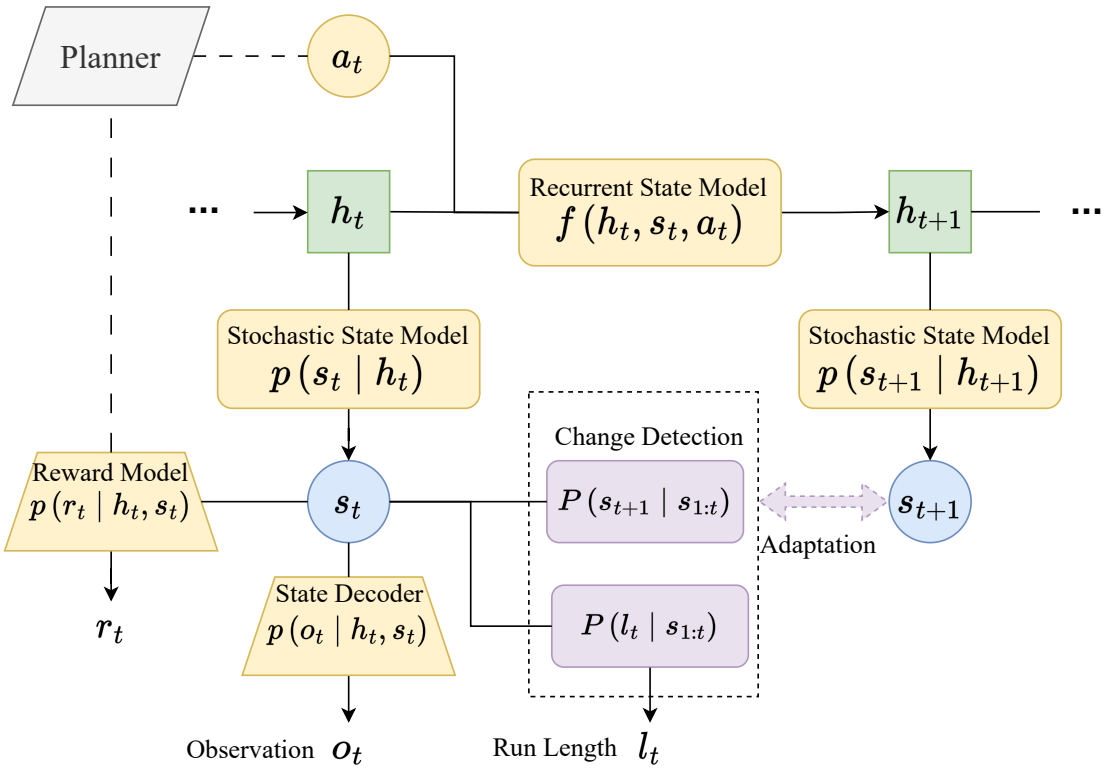


Figure 6.5: Illustration of latent and changing dynamics with change detection and adaptation. The change detection is performed on latent states $\{s_1, s_2, \ldots, s_t\}$, and if a change point is detected, an adaptation term will be added to the training.

where s_0 could be a fixed value or a sample from a non-informative prior, like a Gaussian distribution with zero mean and identity covariance matrix, and a always maximizes the reward. This probabilistic representation framework can reflect the uncertainty of environments.

Once we detect changes in the environment, we can obtain a prediction distribution $p(s_{t+1} \mid \boldsymbol{s}_{1:t})$ which can provide useful information when there is not enough data for new environments. The reason is that when the environment switches, the distribution fitted by change detection contains this information over previous information. This will help the model adapt to the current environment. Consequently, when changes occur in the environment, the information on state transitions can help the agent swiftly identify the optimal trajectory. Therefore, we incorporate it as a regularization term into the loss

function. The final objective is to maximize the following evidence lower bound (ELBO),

$$\sum_{t=1}^{T}\Bigg(\mathrm{E}_{q(s_t|o_{\leq t},a_{<t})}[\ln p(o_t \mid s_t)] - \mathrm{E}_{q(s_{l:t-1})}\mathrm{KL}[q(s_t \mid h_t, o_t) \,||\, p(s_t \mid h_t)] \\ \underbrace{-\beta \mathrm{E}_{q(s_{l:t-1})}\mathrm{KL}[q(s_t \mid h_t, o_t) \,||\, (s_{t+1} \mid \boldsymbol{s}_{1:t})]}_{\text{adaptation term}}\Bigg) + \sum_{t=1}^{T}\mathrm{E}_{q(r_t|o_{\leq t},a_{<t})}[\ln p(r_t \mid s_t)], \tag{6.12}$$

where β controls how much the detection information regularizes the model. At the beginning of the new environment, we use the adaptation term to help model adaptation rapidly. After several update iterations, the regularization will be removed because the model has collected enough data about the new environment. It is noted that if there is no change point detected, the learning objective will follow the original RL objective:

$$\sum_{t=1}^{T}\Bigg(\mathop{\mathrm{E}}_{q(s_t|o_{\leq t},a_{<t})}[\ln p(o_t \mid s_t)] - \mathop{\mathrm{E}}_{q(s_{l:t-1})}[\mathrm{KL}q(s_t \mid h_t, o_t) \,||\, p(s_t \mid h_t)]\Bigg) \\ + \sum_{t=1}^{T}\mathrm{E}_{q(r_t|o_{\leq t},a_{<t})}[\ln p(r_t \mid s_t)]. \tag{6.13}$$

Adaptation Using Detected Information Once we detect changes in the environment, we can obtain a prediction distribution $p(s_{t+1} \mid \boldsymbol{s}_{1:t})$ which can provide useful information when there is not enough data for new environments. The reason is that when the environment switches, the distribution fitted by change detection contains this information over previous information. This will help the model adapt to the current environment. Consequently, when changes occur in the environment, the information on state transitions can help the agent swiftly identify the optimal trajectory. Therefore, we incorporate it as a regularization term into the loss function. Based on Equation (6.13), we have

$$\sum_{t=1}^{T}\Bigg(\mathrm{E}_{q(s_t|o_{\leq t},a_{<t})}[\ln p(o_t \mid s_t)] + \sum_{t=1}^{T}\mathrm{E}_{q(r_t|o_{\leq t},a_{<t})}[\ln p(r_t \mid s_t)] \\ - \mathrm{E}_{q(s_{l:t-1})}[\mathrm{KL}[q(s_t \mid h_t, o_t) \,||\, p(s_t \mid h_t)]] - \beta \mathrm{E}_{q(s_{l:t-1})}[\mathrm{KL}[q(s_t \mid h_t, o_t) \,||\, (s_{t+1} \mid \boldsymbol{s}_{1:t})]]\Bigg) \tag{6.14}$$

where β controls how much the detection information regularizes the model. At the beginning of the new environment, we use the adaptation term to rapidly help model

adaptation. After several update iterations, the regularization will be removed because the model has collected enough data about the new environment.

6.3.4 Planning in Real Non-stationary Environments

Building upon the learned latent dynamics $s_t \sim p(s_t \mid h_t)$ and reward distribution $r_t \sim p(r_t \mid h_t, s_t)$ as described in Equation (6.14), we adopt a planning approach that involves maximizing the expected cumulative reward over a temporal horizon. The integral in the planning equation efficiently marginalizes the latent states s_t, taking into account the history of states and actions to yield the best immediate action a_t^*:

$$a^* t = \arg\max a_t \int_{s_t} p(r_t \mid s_t) p(s_t \mid s_{1:t-1}, a_{1:t-1}). \tag{6.15}$$

Upon computing a_t^*, the prediction of the subsequent latent state is informed by both the previous latent state and the chosen action, as denoted by$h_t = f(h_{t-1}, s_{t-1}, a_{t-1})$ and $s_t \sim p(s_t \mid h_t)$.

To ascertain the most advantageous action, we apply the Cross-Entropy Method (CEM [249, 250, 251]), leveraging its population-based optimization characteristics. CEM distinguishes itself through its iterative sample selection process, focusing on a subset of top-performing candidates to fine-tune the action space towards the highest yields. This method facilitates comprehensive planning that extends beyond immediate considerations, incorporating the impact of potential actions on the agent's future trajectory. Such foresight allows the agent to deliberate on the balance between immediate rewards and future payoffs, ensuring that the actions selected contribute maximally to the cumulative reward over the planning horizon H. In implementing CEM, we provide a stochastic yet systematically structured exploration of potential actions, guiding the agent toward the sequence of actions that promises the greatest reward. This approach harnesses the

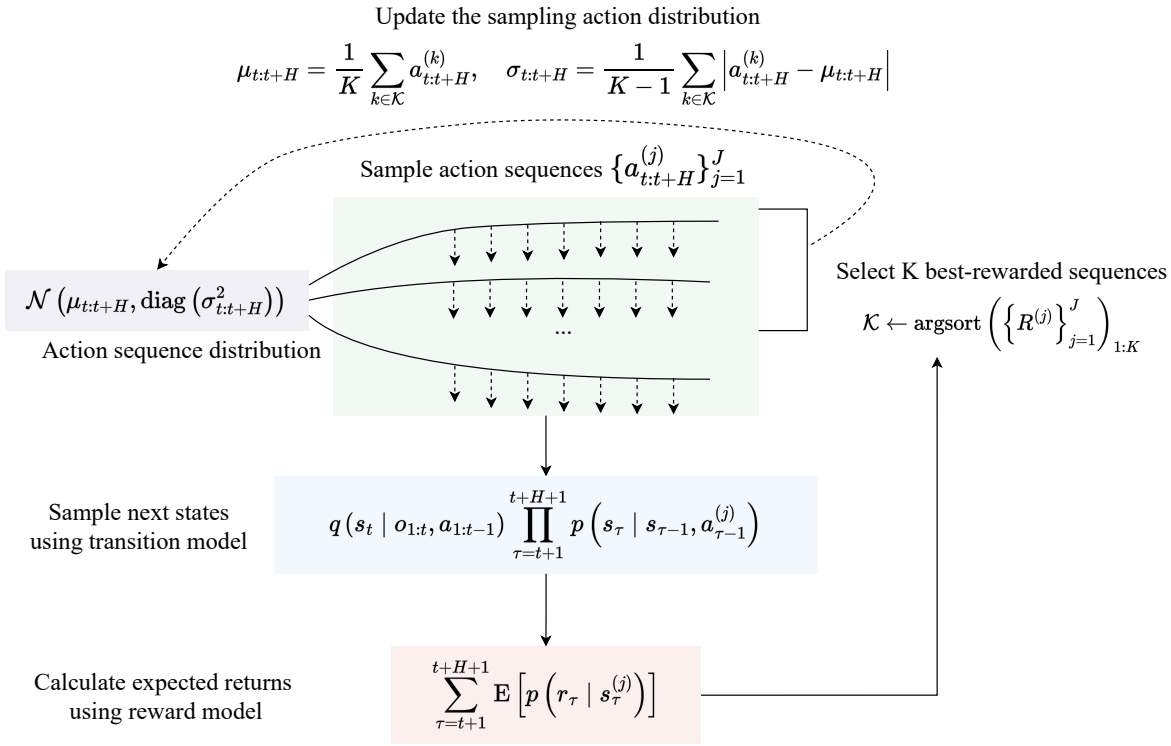


Figure 6.6: The illustration of an iteration of planning at time t. The sampling distribution is initialized as $\mathcal{N}(0, \mathbb{I})$. To evaluate a sampled action sequence using the transition and reward model, we sample a state trajectory with the beginning state s_t and sum over the mean rewards expected along the series. Then, the elite action sequences with the highest reward are picked to refine the sampling distribution parameters.

strengths of both stochastic sampling and deterministic optimization, thereby enhancing the agent's capability to make informed decisions in complex environments.

As illustrated in Figure 6.6, at each iteration, we repeatedly sample J action sequences from a time-evolving diagonal Gaussian distribution $a_{t:t+H} \sim \mathcal{N}\left(\mu_{t:t+H}, \mathrm{diag}\left(\sigma^2_{t:t+H}\right)\right)$. These action sequences are simulated using the learned models to obtain approximate resulting state sequences and a reward sum along the sequence. Then the sample distribution is re-fitted according to the best K sequences. After I iterations, the planner returns the mean μ_t of the sample distribution of the current time step. The action sampling distribution parameters are set initially as $\left(\mu_{t:t+H} = 0, \sigma^2_{t:t+H} = \mathbb{I}\right)$.

The effectiveness of our planning hinges critically on the accuracy of the underlying

Algorithm 7 LLCD (Learn Latent and Changing Dynamics)

Require: State decoder $p(o_t|s_t)$, encoder $p(s_t|o_{\leq t}, a_{\leq t})$, transition model $p(s_t \mid s_{t-1}, a_{t-1})$, reward model $p(r_t|s_t)$, and memory $\mathscr{D}$.

Initialize model parameters θ randomly, and initialize memory $\mathscr{D}$ with random seed episodes.
for episode $i \leq E$ **do**
  for update step $s = 1...S$ **do**
    Sample $\{(o_t, a_t, r_t)_{t=k}^{L+k}\}_{i=1}^{B} \sim \mathscr{D}$ randomly from $\mathscr{D}$.
    if Change detected **and** i within adapt window **then**
      Compute the loss from Equation (6.14) and update parameters $\theta \leftarrow \theta - \alpha \nabla_\theta \mathscr{L}(\theta)$.
    else
      Compute the loss from Equation (6.13) and update parameters $\theta \leftarrow \theta - \alpha \nabla_\theta \mathscr{L}(\theta)$.
    end if
  end for
  $o_1 \leftarrow$ env.reset ()
  for time step $t = 1...T$ **do**
    Infer latent state by the encoder $q(s_t \mid O_{\leq t}, a_{<t})$.
    Select $a_t \leftarrow \text{planner}(q(s_t \mid o_{\leq t}, a_{<t}))$ and take action to get $r_t, o_{t+1} \leftarrow \text{env.step}(a_t)$.
  end for
  $\mathscr{D} \leftarrow \mathscr{D} \cup \{(o_t, s_t, a_t, r_t)_{t=1}^{T}\}$
  Perform change detection according to Algorithm 6 on $\{s_t\}_{detect\_window}$. // See Table 6.1 for detecting window details.
end for

models employed. In dynamic settings, the agility with which the model adapts to new environmental conditions directly influences the precision of subsequent planning. An adept model swiftly aligns itself with the altered environment, enhancing the accuracy of plans and thereby expediting the model's convergence. This interdependence illustrates the significance of a model's adaptability to non-stationary environments, manifested through the precise estimation of state transitions and reward functions. Accurate transition and reward models in non-stationary environments are fundamental for reliable state approximation and reward computation, serving as the cornerstones of our planning algorithm. These models inform the planning process, enabling the agent to make foresighted decisions that align with the shifting dynamics of the environment.

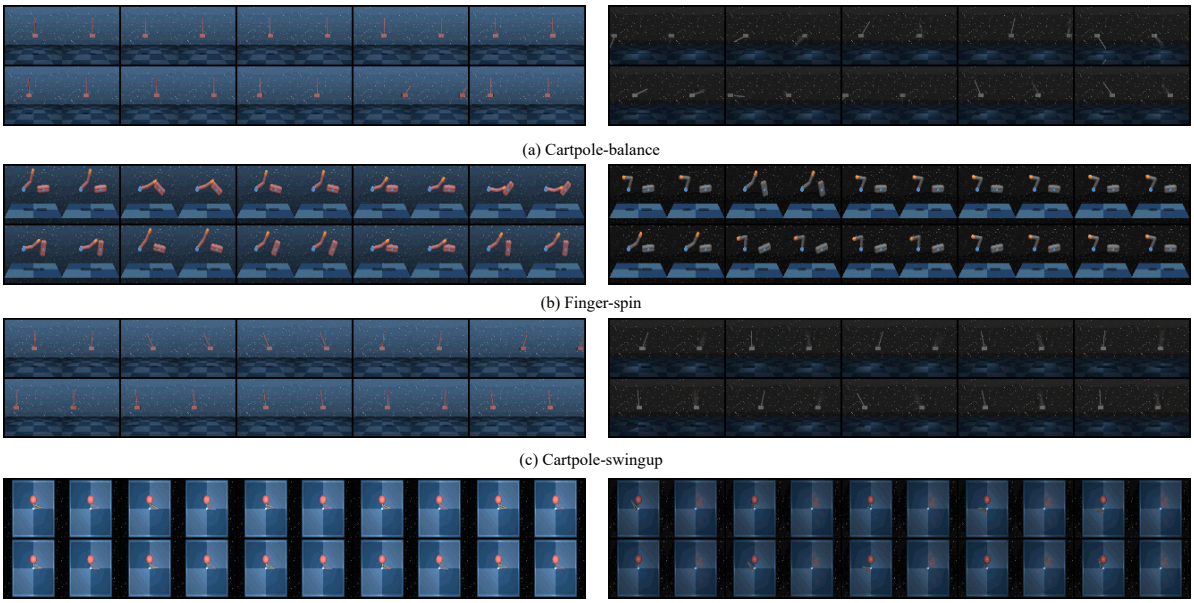
The more promptly and accurately these models can adjust to new conditions, the more effectively they can guide the agent's decision-making process, ensuring that actions are relevant and optimally tuned to the present context.

Enhancing this adaptability involves continually refining these models, incorporating new data, and updating estimates to reflect the current environment. By doing so, we enable the planning algorithm to maintain a trajectory that is not only anticipatory but also responsive to environmental changes. Algorithm 7 delineates the step-by-step workflow of our LLCD approach, encapsulating the entire adaptive cycle, detecting changes and updating models to plan and execute actions. This comprehensive workflow is designed to be robust and flexible, accommodating the intricacies of various non-stationary environments and paving the way for more intelligent and versatile reinforcement learning agents.

6.4 Experiments and Analysis

Our experiments aim to address the core problem: the existing model-based RL methods will be significantly affected by the real non-stationary environment with high-dimensional observations, especially in the event of sudden changes. In contrast, our method can rapidly adapt to sudden non-stationarity and obtain higher rewards.

We explore the problem by answering four key questions: 1) Does LLCD perform better than model-based RL without any detection and adaptation methods? 2) Can the latent space help identify environment change points rather than using raw observation data? 3) Does the training phase in which the change point is located affect performance? That is, if the change occurs before the model converges, what will happen? 4) Is LLCD sensitive to environmental change degrees, and to what extent?



(a) Cartpole-balance

(b) Finger-spin

(c) Cartpole-swingup

(d) Reacher-easy

Figure 6.7: Image-based control environments used in our experiments, where all the basic environments are from dm_control. We change the skyboxes and component materials at the change point to simulate different lighting conditions.

6.4.1 Environments and setup

The environments are designed to exhibit sudden changes at unknown time steps. The changes should be apparent from the pixel observation. Thus, as shown in Figure 6.7, we change the robotics materials and the sky-box of whole environments to simulate the lighting change for *dm_control*.

We evaluate our method LLCD on *dm_control* [21], which offers several continuous control tasks with image observations. In all environments, the only observations are third-person camera images of size $64 \times 64 \times 3$ pixels. The environments are as follows:

- **Cartpole-balance:** The object of this task is to balance an un-actuated pole by applying forces to a cart at its base. The objective of the task is to balance a pole on top of a cart for a given number of timesteps.

- **Cartpole-swingup:** The system consists of a pole, which acts as an inverted pendulum attached to a cart. The force applied to the cart can be controlled, and

the goal is to swing the pole up and balance it around the upward position.

- **Reacher-easy:** The two-link planar reacher with a randomized target location is a typical application of RL algorithms with robotic manipulators. The agent will get rewards when the end effector penetrates the target sphere.

- **Finger-spin:** This task requires predicting two separate objects and the interactions between them. It includes contact dynamics between the finger and the object. In the spin task, the body must be continually rotated.

6.4.2 Comparisons

Our standard baseline follows the Deep Planning Network (PlaNet, [23]), corresponding to our method without any detection or adaptation process. This allows us to evaluate the necessity of researching an approach that aims to adapt to non-stationary environments. Also, we compare our method with CRL-Unsup[13] and Policy Consolidation (PC, [81]). CRL-Unsup detects environment change points using reward and adapts Elastic Weight Consolidation (EWC, [8]) to learn from previous knowledge. PC remembers the agent,Äôs policy at various timescales and regularizes the current policy based on its history to improve learning within different tasks. The baseline we used in these two methods is Proximal Policy Optimization (PPO, [32]). We also choose two model-free methods, the on-policy method PPO and the off-policy method Soft Actor-Critic (SAC, [35]).

For LLCD and PlaNet, as shown in Eq. (6.6) the input observation o_t size is $3 \times 64 \times 64$; the latent state s_t size is 1×30, and the recurrent hidden state h_t size is 1×200. The sample batch size for each update is 64, with a learning rate of 1×10^{-3}. The state encoder consists of four convolutional layers and a fully connected layer; the reward model has three fully connected layers; the transition model has three fully connected layers; the recurrent state model is implemented by GRU [252]. The neural network structure and training parameters are listed in Table 6.1 in the implementation details.

Table 6.1: The model structures and training parameters of our method.

Network Structure	
State decoder	4* Convolutional +1*Linear Networks
Reward model	3*Linear Networks
Transition model	1*Linear Network
Recurrent state model	1*Linear Network+1*GRU
Training details	
Learning rate 1×10^{-3}	Update step 100
Recurrent state size 200	Latent state size 30
Chunk size 60	Batch size 50
Input size $3 \times 64 \times 64$	Planning horizon 12
Detection window 1000	Adaptation ratio 0.1

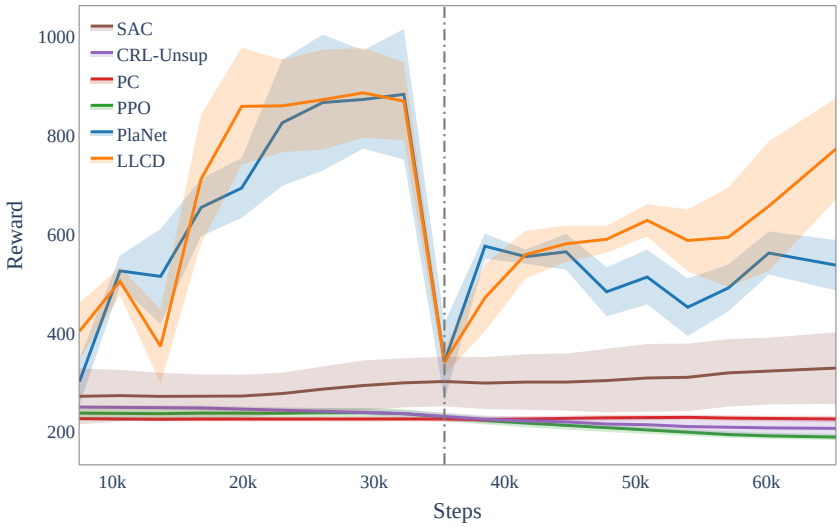
Table 6.2: The performance (average return) of trained agents on non-stationary environments with change point at the middle stage. The blank value represents that the method fails in the corresponding task after the same training step as LLCD and PlaNet.

	Cartpole-swingup	Cartpole-balance	Reacher-easy	Finger-spin
LLCD(Ours)	**462.0± 8.2**	**646.4±13.1**	**700.1 ± 23.2**	**410.5±9.7**
PlaNet	449.8±6.2	541.8±9.2	514.7±32.9	407.4±7.8
CRL-Unsup	97.8±13.7	268.6±27.1	80.4±8.2	/
PC	81.5±8.3	271.4±9.5	77.0±7.4	/
PPO	73.1±9.6	204.9±11.2	82.6±11.2	/
SAC	156.3±15.6	286.3±14.9	123.2±13.8	179.8±20.2

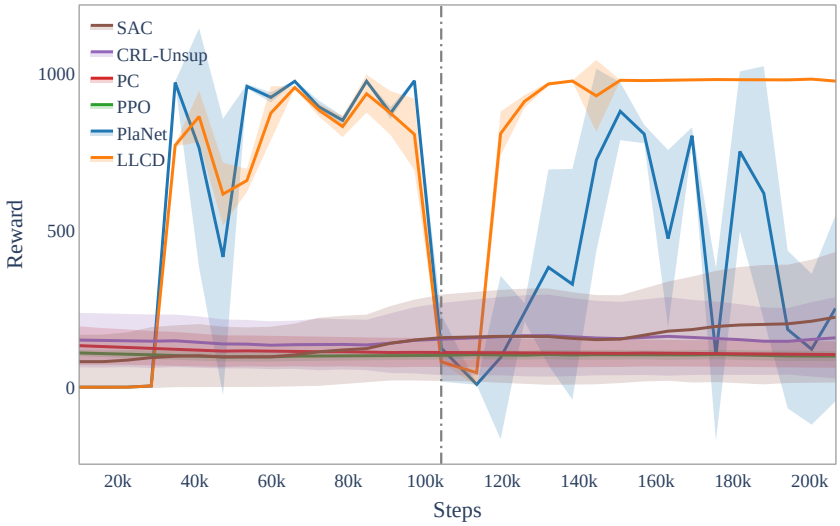
6.4.3 Overall performance

Our experiments simulate the non-stationary environments. Over the training time steps, a change point is set at the middle stage of training to simulate a sudden change in lighting level, which is not shown in the model or algorithm.

The reward curve in Figure 6.8 proves that our method LLCD can converge more rapidly than the baseline when change occurs as the response speed to sudden change is faster than the baseline. After the change point, our method achieves higher rewards



(a) cartpole-balance



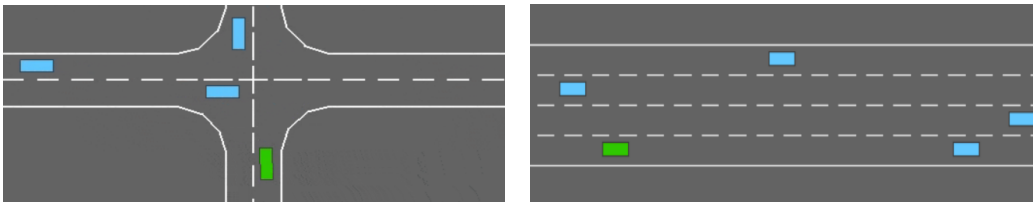
(b) reacher-easy

Figure 6.8: The test reward of our problem setting. The solid line depicts the average reward, while the dashed lines indicate the minimum and maximum reward values. The shaded area represents the reward's standard deviation over multiple runs.

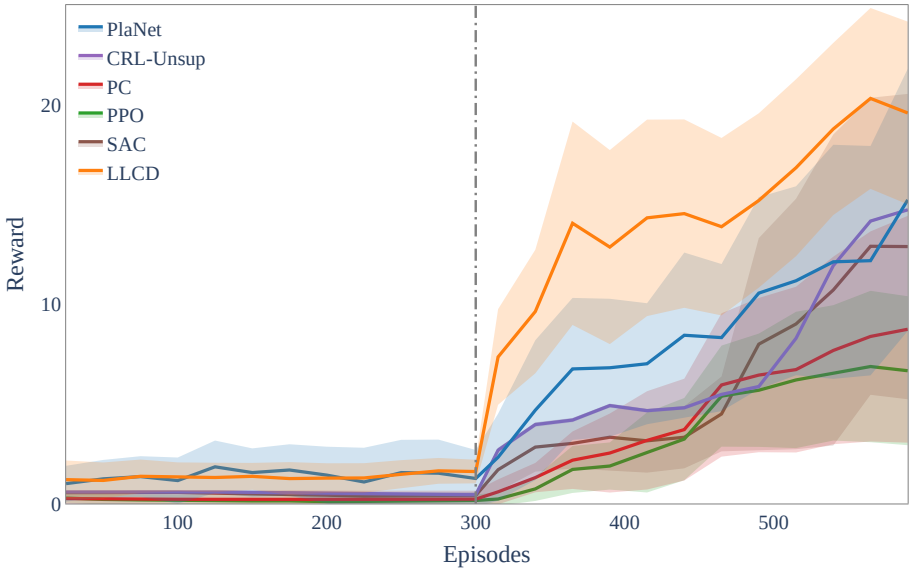
sooner, while the baseline still struggles with new environments. This phenomenon demonstrates that our approach can accurately identify the change points, and our regularization term can also contribute to adapting to the new environment when a change point is detected. For the cartpole-balance environment, LLCD (denoted by orange) and

PlaNet demonstrate superior performance compared to other methods. Before the change point, LLCD and PlaNet rapidly achieved high rewards, indicating efficient learning and adaptation. Post-change, LLCD quickly recovers and surpasses PlaNet, achieving higher rewards with lower variance. This showcases LLCD's ability to detect changes and adapt promptly, maintaining robust performance. In the reacher-easy environment, LLCD and PlaNet also outperform other methods. LLCD's rapid convergence before the change point and swift recovery afterward highlight its effectiveness in handling non-stationary environments. The larger fluctuations in PlaNet's performance post-change suggest it is less stable than LLCD. Other methods, such as CRL-Unsup, PC, PPO, and SAC, struggle to achieve significant rewards, reflecting their lower adaptability and sample efficiency. The results support our view that when an environment changes, the model can only acquire a small amount of data from the new environment after the change point, and training with adaptation is crucial at this time. Adaptation helps to obtain more information in changed environments.

The complete results are shown in Table 6.2, the average return of all comparison methods over four different environments. It is noted that both CRL-Unsup and PC perform below the model-based benchmark. The probable reason for this is PPO, as an on-policy method, inherently requires collecting a large amount of data for training and has relatively lower sample utilization efficiency. Within the same agent steps, LLCD and PlaNet have higher data utilization because both of them sample sequences $\{(o_t, a_t, r_t)_{t=k}^{L+k}\}_{i=1}^{B} \sim \mathscr{D}$ from the data memory repeatedly at each update step. For this reason, model-based RL has more significant application potential for many scenarios with challenging sampling requirements, such as safety-critical settings and autonomous driving scenarios. Overall, our method achieved the highest average return in all domains.



(a) The simulated non-stationary environment: a car drives through an intersection and then enters a highway.



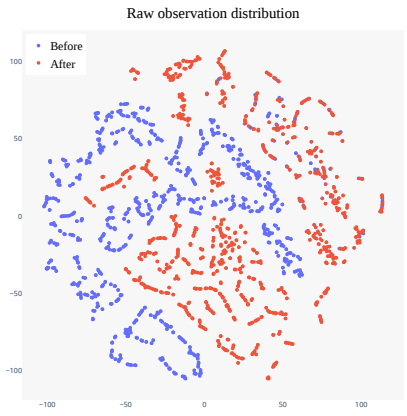
(b) The reward curve of different methods. The solid lines indicate the mean over multiple runs with 10 seeds. The shaded area is standard deviation.

Figure 6.9: The simulated driving scenario and experiment results.

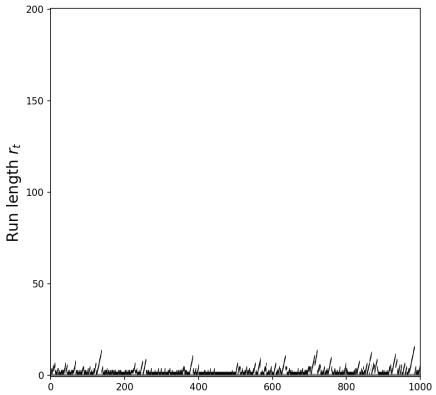
6.4.4 Ablation Studies

Autonomous driving scenarios To evaluate the proposed method in a more realistic scenario, we choose *highway-env* [253] to simulate driving scenes in daily life, which is driving through an intersection and then entering a highway:

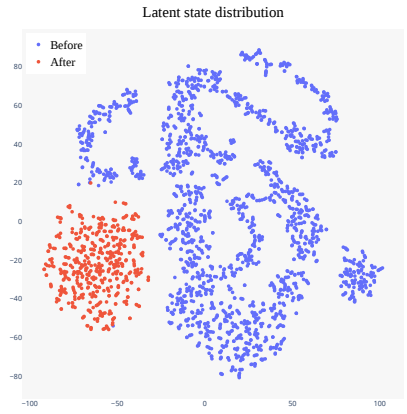
- **intersection-v1**: An intersection negotiation task with dense traffic. The task requires the agent to make quick decisions to avoid collisions and ensure safe passage.



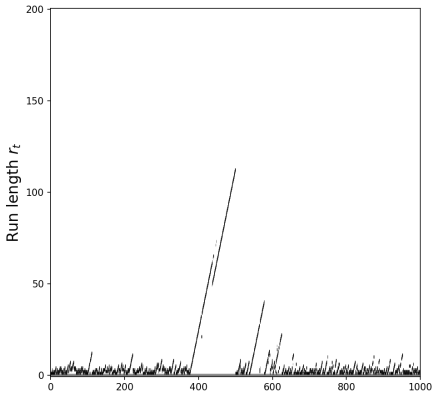
(a) T-sne plot of raw observation



(b) Detection result of observation



(c) T-sne plot of latent state



(d) Detection results of latent state

Figure 6.10: Visualization of the observation and latent state distribution using t-SNE. The distribution shift is clearly seen in the latent state rather than raw observation data. Also, the approximated run length of Bayesian online change detection for the corresponding data is presented. The raw data does not have an obvious change of run length distribution $P(l_t \mid s_{1:t})$, while the latent state has a significant variation.

- **highway-v0**: An ego-vehicle is driving on a multilane highway populated with other vehicles. The agent's objective is to reach a high speed while avoiding collisions with neighboring vehicles. Driving on the right side of the road is also rewarded.

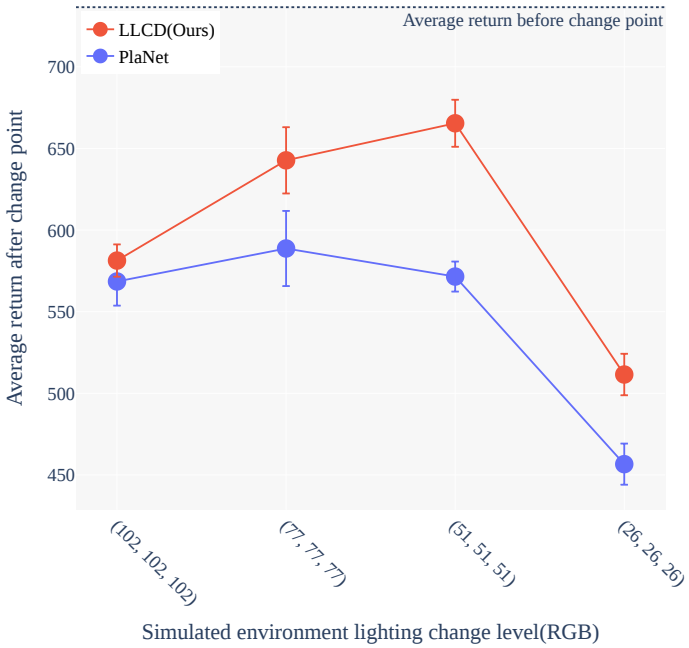


Figure 6.11: The average return after the change point varies under different degrees of environmental changes of Cartpole-balance. The aboriginal RGB value of the environment skybox is (102, 153, 204), and the change degree increases from left to right.

The settings and results in the autonomous driving environments are shown in Figure 6.9. The autonomous driving environment's shift from intersection to highway introduces significant changes in the state dynamics and required models. The ability of LLCD to quickly adjust to this shift is evident in the sharp increase in rewards after the change point, highlighting its robustness in non-stationary settings. The superior performance of LLCD compared to both PlaNet and the model-free methods demonstrates its sample efficiency in rapidly learning and adapting to new environmental conditions. Among the model-free methods, there is less pronounced improvement post the change point, indicating a slower learning speed to the new driving conditions introduced in the simulation. This is expected, as model-free methods typically lack the sample efficiency that model-based methods possess. The results underscore LLCD's performance in environments requiring rapid adaptation to non-stationary conditions. LLCD efficiently leverages the distribution predictions produced by the detection scheme, enabling it to

Table 6.3: The steps needed to converge for different methods.

Method	Steps needed for convergence			
	cartpole-balance	cartpole-swingup	reacher-easy	finger-spin
PPO/CRL-Unsup	3.7×10^5	4.1×10^5	1.16×10^5	8.1×10^5
SAC	6.0×10^4	2.4×10^5	1.6×10^5	1.2×10^5
PC	3.7×10^5	4.7×10^5	1.03×10^5	7.7×10^5
PlaNet/LLCD	4.8×10^4	1.2×10^5	6.0×10^4	5.8×10^4

outperform both PlaNet and model-free methods, making it a promising approach for various non-stationary environment settings.

Sample Efficiency Table 6.3 illustrates the number of steps needed for different methods to converge across various tasks: cartpole-balance, cartpole-swingup, reacher-easy, and finger-spin. Our method, LLCD, alongside PlaNet, exhibits superior sample efficiency compared to model-free methods like PPO/CRL-Unsup, SAC, and PC. Specifically, LLCD and PlaNet require significantly fewer steps to converge across all tasks. This superior performance of LLCD and PlaNet can be attributed to their model-based nature, which allows them to utilize samples more effectively by learning a model of the environment and leveraging it for planning and decision-making. In contrast, model-free methods rely solely on direct interactions with the environment, leading to less efficient use of samples and thus requiring more steps to achieve convergence.

Latent state Learning from raw data is straightforward. Still, there are several difficulties: 1) The transition model using observation data is $p(o_t|o_{t-1},a_{t-1})$, which needs to have a Markov property and only depends on the observation at time $t-1$. This assumption is obviously unreasonable in a continuous control task with image observations. Image data have various features, such as shapes, colors, textures, and patterns. It is more challenging for environment models to extract specific transitions from this information. 2) Using high-dimensional image data will lead to significantly increased computational costs. Raw image data contains a large number of pixels, and

most may not contribute to the RL model or the specific task at hand.

To solve these issues, we learn the transition and reward models and then detect environment change points in the latent space. As Figure 6.10 shows, the raw data does not reflect a noticeable distribution shift when change occurs. The raw data before and after the change do not overlap exactly, but it is difficult to distinguish. The high-dimensional nature of raw data also poses some difficulties if it is used for change detection. In contrast, the distribution of the latent state before and after the change is clearly distinguished, which indicates that we can better identify the change in the environment in the latent space. The latent state eliminates the redundant information of the high-dimensional image input and thus transparently reflects the change of the environment transition.

Furthermore, the detected run length distribution according to Equation. (6.8) is also shown in Figure 6.10. After the environmental change points, the run length distribution $P(l_t \mid s_{1:t}) = P(l_t, s_{1:t}) / \sum_{l_t} P(l_t, s_{1:t})$ still follows a similar pattern at each time step, although the distribution of raw data has switched to another extent. However, the run length of the latent state distribution has significantly changed around the change point. Hence, using latent space will help us effectively identify the change point when the environment changes. This finding further reinforces the motivation of the learning transition and reward models in latent space.

Sensitivity to change levels As shown in Figure 6.11, the average reward after the change point varies under different degrees of environmental changes. The initial RGB value of the environment skybox is RGB (102, 153, 204), and we simulated four change degrees by modifying the RGB level. From the figure, we can observe several interesting facts. First, we may assume that the smaller the degree of change in the observations, the less intuitively the impact on the model will be. Yet we found that within a specific range of environmental change, the performance fluctuates within a

specific range. Figure 6.10 illustrates that the change in observation distribution is not entirely equivalent to the change in the latent state. However, significant observation changes can lead to noticeable performance drops in latent states. Second, our method LLCD performs better in each change level than the baseline. When the environment changes slightly, the performance of the model may not drop noticeably, so the average return is closed with mild changes. Finally, LLCD behaves more differently from the baseline when the environmental changes become more drastic. The reason for this is that when the changes are relatively minor, the information provided by our fitted distribution makes a more limited contribution than the information the model obtains through learning. When the environment approaches complete darkness (RGB (26, 26, 26)), both performances experience a sharp decline; however, LLCD still performs better than the benchmark, indicating that our adaptation regularization is effective in assisting training during drastic changes.

The results show that our approach works well for environmental changes that cause the model to exhibit significant performance degradation. Both our method and the baseline show significant performance loss if there are very drastic changes in the environment. Nevertheless, LLCD still shows a solid ability to adapt to dramatic changes.

Change point position sensitivity The previous experiment simulated a change point at the middle stage of training, after which we will test the performance with random change to explain the effect of change points. If the change occurs before the model fully converges, as shown in Table 6.4, LLCD can achieve a higher average return in all domains than the baseline. This indicates that our detection and adaptation methods can quickly make it fit new environments, even when the model does not fully fit the current environment. In the case of insufficient environmental data, it is also valid to use the detection distribution to regularize the latent state's posterior distribution. Similarly, LLCD outperforms the baseline when the model has fully converged. The

Table 6.4: Performance (average return) of trained agents on non-stationary environments with a change point in the early and late stages of training. For the early change, a random change point was set within the first 20% – 30% of the total training steps, and for the late change, it was set within 70% – 80% of the total training steps.

Change in the early stage		
	Cartpole-swingup	Cartpole-balance
LLCD (Ours)	**156.7±16.8**	**741.9±12.9**
PlaNet	135.9±19.6	627.0±8.9
	Reacher-easy	Finger-spin
LLCD (Ours)	**694.4±20.6**	**248.0±15.4**
PlaNet	693.9±28.4	196.0±13.7
Change in the late stage		
	Cartpole-swingup	Cartpole-balance
LLCD (Ours)	**129.7±28.4**	**751.5±12.3**
PlaNet	123.5±17.7	742.1±11.5
	Reacher-easy	Finger-spin
LLCD (Ours)	**725.7±26.6**	**369.7±20.3**
PlaNet	690.1±16.6	367.1±21.9

results indicate that our method is not very sensitive to the location of change points and is robust in different types and environments with varying points of change.

6.5 Summary

In this chapter, we formalized the problem of model-based reinforcement learning in non-stationary environments with unknown change points. We introduce LLCD, a framework that effectively identifies changes in the non-stationary environment and adjusts the learned model to better fit the new one. Our method learns the transition model and reward model in a latent space and plans the action using the represented environment. At the same time, our approach detects the change points of non-stationary environments online by monitoring the latent state and is able to adapt to new environments rapidly

using a regularization term related to the detection results. The experiments show that LLCD achieves robust performance including higher response speed and average return in various non-stationary image-based continuous control environments where state distributions vary with time in an unknown manner and with unknown change points.

In future work, we plan to investigate the theoretical guarantee of the proposed ideas related to policy improvement. Another exciting work will be to investigate the mutual boosting of detection and adaptation. Finally, applying this work to real-world applications is also highly attractive.

CHAPTER 7

CONCLUSION AND FUTURE RESEARCH

Deep Reinforcement Learning in non-stationary environments has been challenging and enlightening. Examples can be found in various domains, such as robotics, finance and healthcare. This thesis addresses the critical challenge of deep reinforcement learning in non-stationary environments with unknown change points. To maximize the long-term rewards, it is essential to identify the environmental changes and respond to them promptly and appropriately. Aiming at the goals, we have proposed comprehensive frameworks and innovative methods for environmental change detection and rapid adaptation. Our approach encompasses model-free and model-based RL, leveraging techniques such as analysis of joint distribution variations, policy behavior changes, Bayesian surprise, and latent space dynamics learning to detect environment change points accurately. By preserving valuable information and adjusting policy adaptation based on detected changes, we have enhanced the robustness and effectiveness of RL agents in dynamic and unpredictable settings.

In Chapters. 3, 4 and 5, we formalized the problem of model-free reinforcement learning in non-stationary environments with unknown change points. In Chapter 3,

we developed an end-to-end algorithm involving two steps in the change detection and adaptation process. First, points of change are detected from the joint distribution of state and policy. Second, a distance-relaxed gradient-constrained adaptor quickly trains a new policy with the help of former well-trained policies. The proposed framework, Detection-Adaptation RL (DARL), is empirically proven to be effective in both high-dimensional and simple environments. Chapter 4 proposes a behavior-based change detection method representing a novel approach to monitoring and responding to environmental shifts by closely analyzing policy behavior. Furthermore, the online adaptation mechanism integrates this behavioral information, providing a self-adjusted regularization term. Behavior-Aware Detection Adaptation (BADA) outperforms commonly used baselines in various aspects. Further, Chapter 5 enhances the adaptation mechanism from function space by approximating DNN to the Gaussian Process. With the approximated Gaussian Process, the environmental change points are identified by Bayesian surprise. To reduce complexity and storage costs, we proposed a novel trajectory selection method to pick the most representative ones for an environment, which can provide valuable knowledge for functional regularization. The experiment shows the proposed Functional Detection Adaptation (FDA) has good performance in non-stationary environments. In Chapter 6, we formalized the problem of model-based reinforcement learning in non-stationary environments with unknown change points. We introduce Learn Latent and Changing Dynamics (LLCD), which detects the change points of non-stationary environments online by monitoring the latent state. It can adapt to new environments rapidly using a regularization term related to the detection results. The experiments show that LLCD achieves robust performance, including higher response speed and average return in various non-stationary image-based continuous control environments.

The methodologies presented in this thesis contribute to advancing the field of reinforcement learning, offering practical solutions for real-world applications where the

assumption of stationarity does not hold and ensuring continued optimal performance in the face of non-stationary environments.

Future Research

In future work, we aim to delve deeper into the theoretical aspects of our proposed ideas, particularly concerning policy improvement guarantees. We plan to develop further strategies that maximize the utility of collected samples, enhancing the efficiency and effectiveness of our methods. An avenue is to refine exploration strategies to ensure that the collected samples cover a diverse set of states and actions. Techniques such as intrinsic motivation, where agents are rewarded for exploring new and informative states, could be utilized. This not only enhances sample diversity but also prevents the agent from getting stuck in local optima. Furthermore, meta-learning approaches can be incorporated to enable our agents to adapt to new tasks with minimal data quickly. By training a meta-learner on a range of tasks, the agent can apply its learning algorithm to new tasks, significantly improving its adaptation capabilities. Exploring the potential of combining our approaches with state-of-the-art deep learning architectures will further advance the capabilities of our models.

Multi-agent reinforcement learning can be categorized into competitive and cooperative objectives. For each agent, the state it perceives is closely related to the actions of other agents, inherently creating a non-stationary factor. Extending our detection-adaptation framework to a multi-agent scenario is quite compelling. In a multi-agent context, our detection-adaptation framework could be crucial for understanding and responding to the complex dynamics that emerge from agent interactions. In competitive settings, for instance, each agent must continuously adapt its strategy to outmaneuver the others, and detecting shifts in opponents' strategies becomes as essential as recognizing environmental changes. Similarly, in cooperative settings, agents need to

detect and adapt to the evolving strategies of their collaborators to achieve shared goals more effectively. Ultimately, such an extension would enable agents to maintain robust performance despite the non-stationarity that defines multi-agent environments, fostering a deeper level of strategic interaction and collaboration. This could pave the way for more advanced applications of multi-agent systems in areas such as autonomous driving, where agents must navigate complex and dynamic scenarios with cooperative and competitive elements.

Finally, the real-world applicability of our reinforcement learning methodologies is not just an aspirational goal; it is a critical pathway to proving their value. By transitioning from simulated environments to real-world scenarios, we can tackle the inherent complexities and unpredictable nature of real-life applications. In robotics, our methods can be applied to enhance autonomous manipulation and navigation, allowing robots to learn from their interactions with the physical world. By doing so, robots can become more adept at tasks such as assembly, logistics, and human assistance in domestic or industrial settings. The field of autonomous vehicles stands to benefit significantly from our work. Our methodologies can enhance the safety and reliability of these systems by enabling vehicles to learn from a vast array of driving conditions and scenarios. In healthcare, the potential impact of our reinforcement learning systems is profound. By learning from patient data, medical robots can support surgeries and care. At the same time, intelligent systems can assist in diagnosing and developing tailored treatment plans, leading to better patient outcomes and more efficient healthcare services. Our ambition is to bridge the gap between theoretical models and practical implementations, ensuring that our reinforcement learning systems are not only theoretically sound but also practically viable.

BIBLIOGRAPHY

[1] G. E. Hinton and S. Roweis, "Stochastic neighbor embedding," *Advances in neural information processing systems*, vol. 15, 2002.

[2] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–18, 2021.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[4] D. Abel, A. Barreto, B. Van Roy, D. Precup, H. P. van Hasselt, and S. Singh, "A definition of continual reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[5] R. S. Sutton, A. Koop, and D. Silver, "On the role of tracking in stationary environments," in *International conference on Machine learning (ICML)*, pp. 871–878, 2007.

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

[7] Y. Chandak, G. Theocharous, S. Shankar, M. White, S. Mahadevan, and P. Thomas, "Optimizing for the future in non-stationary mdps," in *International Conference on Machine Learning (ICML)*, pp. 1414–1425, 2020.

[8] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

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

[10] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell, "Progress & compress: A scalable framework for continual learning," in *International Conference on Machine Learning(ICML)*, pp. 4528–4537, 2018.

[11] B. C. Da Silva, E. W. Basso, A. L. Bazzan, and P. M. Engel, "Dealing with non-stationary environments using context detection," in *International conference on Machine learning (ICML)*, pp. 217–224, 2006.

[12] X. Chen, X. Zhu, Y. Zheng, P. Zhang, L. Zhao, W. Cheng, P. Cheng, Y. Xiong, T. Qin, J. Chen, *et al.*, "An adaptive deep rl method for non-stationary environments with piecewise stable context," *Advances in Neural Information Processing Systems*, vol. 35, pp. 35449–35461, 2022.

[13] V. Lomonaco, K. Desai, E. Culurciello, and D. Maltoni, "Continual reinforcement learning in 3d non-stationary environments," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 248–249, 2020.

[14] S. Padakandla, K. Prabuchandran, and S. Bhatnagar, "Reinforcement learning algorithm for non-stationary environments," *Applied Intelligence*, vol. 50, no. 11, pp. 3590–3606, 2020.

[15] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 6467–6476, 2017.

[16] P. Pan, S. Swaroop, A. Immer, R. Eschenhagen, R. Turner, and M. E. E. Khan, "Continual deep learning by functional regularisation of memorable past," *Advances in neural information processing systems*, vol. 33, pp. 4453–4464, 2020.

[17] M. K. Titsias, J. Schwarz, A. G. d. G. Matthews, R. Pascanu, and Y. W. Teh, "Functional regularisation for continual learning with gaussian processes," *arXiv preprint arXiv:1901.11356*, 2019.

[18] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," *arXiv preprint arXiv:1611.05397*, 2016.

[19] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *4th International Conference on Learning Representations*, 2016.

[20] D. Isele and A. Cosgun, "Selective experience replay for lifelong learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

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

[22] O. Rybkin, C. Zhu, A. Nagabandi, K. Daniilidis, I. Mordatch, and S. Levine, "Model-based reinforcement learning via latent-space collocation," in *International*

Conference on Machine Learning, pp. 9190–9201, PMLR, 2021.

[23] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in *International conference on machine learning*, pp. 2555–2565, PMLR, 2019.

[24] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[25] X. Wang, L. Ke, Z. Qiao, and X. Chai, "Large-scale traffic signal control using a novel multiagent reinforcement learning," *IEEE Transactions on Cybernetics*, vol. 51, no. 1, pp. 174–187, 2021.

[26] Z. Wan, C. Jiang, M. Fahad, Z. Ni, Y. Guo, and H. He, "Robot-assisted pedestrian regulation based on deep reinforcement learning," *IEEE Transactions on Cybernetics*, vol. 50, no. 4, pp. 1669–1682, 2020.

[27] W. Bai, Q. Zhou, T. Li, and H. Li, "Adaptive reinforcement learning neural network control for uncertain nonlinear system with input saturation," *IEEE Transactions on Cybernetics*, vol. 50, no. 8, pp. 3433–3443, 2020.

[28] J. Li, Y. Ma, R. Gao, Z. Cao, A. Lim, W. Song, and J. Zhang, "Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem," *IEEE Transactions on Cybernetics*, vol. 52, no. 12, pp. 13572–13585, 2022.

[29] Z. Wang, H.-X. Li, and C. Chen, "Reinforcement learning-based optimal sensor placement for spatiotemporal modeling," *IEEE Transactions on Cybernetics*, vol. 50, no. 6, pp. 2861–2871, 2020.

[30] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural*

information processing systems, vol. 12, 1999.

[31] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, pp. 1889–1897, PMLR, 2015.

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

[33] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning (ICML)*, pp. 1928–1937, 2016.

[34] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*, pp. 387–395, Pmlr, 2014.

[35] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, pp. 1861–1870, PMLR, 2018.

[36] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*, pp. 1587–1596, PMLR, 2018.

[37] S. D. Holcomb, W. K. Porter, S. V. Ault, G. Mao, and J. Wang, "Overview on deepmind and its alphago zero ai," in *Proceedings of the 2018 international conference on big data and education*, pp. 67–71, 2018.

[38] M. Schneckenreither and S. Haeussler, "Reinforcement learning methods for operations research applications: The order release problem," in *International Conference on Machine Learning, Optimization, and Data Science*, pp. 545–559, Springer, 2018.

[39] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[40] K.-L. A. Yau, J. Qadir, H. L. Khoo, M. H. Ling, and P. Komisarczuk, "A survey on reinforcement learning models and algorithms for traffic signal control," *ACM Computing Surveys (CSUR)*, vol. 50, no. 3, pp. 1–38, 2017.

[41] M. M. Afsar, T. Crump, and B. Far, "Reinforcement learning based recommender systems: A survey," *ACM Computing Surveys*, vol. 55, no. 7, pp. 1–38, 2022.

[42] K. Shao, Z. Tang, Y. Zhu, N. Li, and D. Zhao, "A survey of deep reinforcement learning in video games," *arXiv preprint arXiv:1912.10944*, 2019.

[43] M. A. Okyere, R. Forson, and F. Essel-Gaisey, "Positive externalities of an epidemic: The case of the coronavirus (covid-19) in china," *Journal of medical virology*, vol. 92, no. 9, pp. 1376–1379, 2020.

[44] Y. Cheng and W. Zhang, "Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels," *Neurocomputing*, vol. 272, pp. 63–73, 2018.

[45] G. Frost, F. Maurelli, and D. M. Lane, "Reinforcement learning in a behaviour-based control architecture for marine archaeology," in *OCEANS 2015-Genova*, pp. 1–5, IEEE, 2015.

[46] I. Carlucho, M. De Paula, S. Wang, Y. Petillot, and G. G. Acosta, "Adaptive low-level control of autonomous underwater vehicles using deep reinforcement learning," *Robotics and Autonomous Systems*, vol. 107, pp. 71–86, 2018.

[47] X. Cao, C. Sun, and M. Yan, "Target search control of auv in underwater environment with deep reinforcement learning," *IEEE Access*, vol. 7, pp. 96549–96559, 2019.

[48] H. Hu, S. Song, and C. P. Chen, "Plume tracing via model-free reinforcement learning method," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 8, pp. 2515–2527, 2019.

[49] J. Zhu, J. Zhu, Z. Wang, S. Guo, and C. Xu, "Hierarchical decision and control for continuous multitarget problem: Policy evaluation with action delay," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 2, pp. 464–473, 2018.

[50] J. Hu, H. Zhang, and L. Song, "Reinforcement learning for decentralized trajectory design in cellular uav networks with sense-and-send protocol," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6177–6189, 2018.

[51] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Conference on Robot Learning*, pp. 785–799, PMLR, 2023.

[52] N. M. Shafiullah, Z. Cui, A. A. Altanzaya, and L. Pinto, "Behavior transformers: Cloning k modes with one stone," *Advances in neural information processing systems*, vol. 35, pp. 22955–22968, 2022.

[53] J.-B. Yi, J. Kim, T. Kang, D. Song, J. Park, and S.-J. Yi, "Anthropomorphic grasping of complex-shaped objects using imitation learning," *Applied Sciences*, vol. 12, no. 24, p. 12861, 2022.

[54] H. Shi, X. Li, K.-S. Hwang, W. Pan, and G. Xu, "Decoupled visual servoing with fuzzy q-learning," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 1, pp. 241–252, 2016.

[55] G. Shani, D. Heckerman, R. I. Brafman, and C. Boutilier, "An mdp-based recommender system.," *Journal of Machine Learning Research*, vol. 6, no. 9, 2005.

[56] X. Zhao, L. Zhang, Z. Ding, L. Xia, J. Tang, and D. Yin, "Recommendations with negative feedback via pairwise deep reinforcement learning," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1040–1048, 2018.

[57] S. Liu, Y. Chen, H. Huang, L. Xiao, and X. Hei, "Towards smart educational recommendations with reinforcement learning in classroom," in *2018 IEEE international conference on teaching, assessment, and learning for engineering (TALE)*, pp. 1079–1084, IEEE, 2018.

[58] P. Wei, S. Xia, R. Chen, J. Qian, C. Li, and X. Jiang, "A deep-reinforcement-learning-based recommender system for occupant-driven energy optimization in commercial buildings," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6402–6413, 2020.

[59] L. Wang, W. Zhang, X. He, and H. Zha, "Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2447–2456, 2018.

[60] X. Zhao, L. Xia, L. Zhang, Z. Ding, D. Yin, and J. Tang, "Deep reinforcement learning for page-wise recommendations," in *Proceedings of the 12th ACM conference on recommender systems*, pp. 95–103, 2018.

[61] E. Hadoux, A. Beynier, and P. Weng, "Sequential decision-making under non-stationary environments via sequential change-point detection," in *Learning over Multiple Contexts (LMCE)*, 2014.

[62] T. Banerjee, M. Liu, and J. P. How, "Quickest change detection approach to optimal control in markov decision processes with model changes," in *American Control Conference (ACC)*, pp. 399–405, 2017.

[63] A. G. Tartakovsky and V. V. Veeravalli, "General asymptotic bayesian theory of quickest change detection," *Theory of Probability & Its Applications*, vol. 49, no. 3, pp. 458–497, 2005.

[64] T. L. Lai, "Information bounds and quick detection of parameter changes in stochastic systems," *IEEE Transactions on Information theory*, vol. 44, no. 7, pp. 2917–2929, 1998.

[65] G. V. Moustakides, A. S. Polunchenko, and A. G. Tartakovsky, "Numerical comparison of cusum and shiryaev–roberts procedures for detecting changes in distributions," *Communications in Statistics-Theory and Methods*, vol. 38, no. 16-17, pp. 3225–3239, 2009.

[66] L. N. Alegre, A. L. Bazzan, and B. C. da Silva, "Minimum-delay adaptation in non-stationary reinforcement learning via online high-confidence change-point detection," in *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 97–105, 2021.

[67] J. D. Healy, "A note on multivariate cusum procedures," *Technometrics*, vol. 29, no. 4, pp. 409–412, 1987.

[68] T. Sibanda and N. Sibanda, "The cusum chart method as a tool for continuous monitoring of clinical outcomes using routinely collected data," *BMC medical research methodology*, vol. 7, pp. 1–7, 2007.

[69] T. Azayev and K. Zimmerman, "Blind hexapod locomotion in complex terrain with gait adaptation using deep reinforcement learning and classification," *Journal of Intelligent & Robotic Systems*, pp. 1–13, 2020.

[70] K. Prabuchandran, N. Singh, P. Dayama, A. Agarwal, and V. Pandit, "Change point detection for compositional multivariate data," *Applied Intelligence*, pp. 1–26, 2021.

[71] R. P. Adams and D. J. MacKay, "Bayesian online changepoint detection," *stat*, vol. 1050, p. 19, 2007.

[72] L. Itti and P. Baldi, "Bayesian surprise attracts human attention," *Vision research*, vol. 49, no. 10, pp. 1295–1306, 2009.

[73] M. Nagayoshi, H. Murao, and H. Tamaki, "Reinforcement learning for dynamic environment: a classification of dynamic environments and a detection method of environmental changes," *Artificial Life and Robotics*, vol. 18, no. 1-2, pp. 104–108, 2013.

[74] B. L. Welch, "The generalization of 'students' problem when several different population varlances are involved," *Biometrika*, vol. 34, no. 1-2, pp. 28–35, 1947.

[75] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, "Vizdoom: A doom-based ai research platform for visual reinforcement learning," in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 1–8, IEEE, 2016.

[76] X. Lin, P. Guo, C. Florensa, and D. Held, "Adaptive variance for changing sparse-reward environments," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 3210–3216, IEEE, 2019.

[77] A. Xie, J. Harrison, and C. Finn, "Deep reinforcement learning amidst continual structured non-stationarity," in *International Conference on Machine Learning(ICML)*, pp. 11393–11403, 2021.

[78] A. Achille, M. Lam, R. Tewari, A. Ravichandran, S. Maji, C. C. Fowlkes, S. Soatto, and P. Perona, "Task2vec: Task embedding for meta-learning," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6430–6439, 2019.

[79] L. Lan, Z. Li, X. Guan, and P. Wang, "Meta reinforcement learning with task embedding and shared policy," *arXiv preprint arXiv:1905.06527*, 2019.

[80] G. Berseth, D. Geng, C. Devin, N. Rhinehart, C. Finn, D. Jayaraman, and S. Levine, "Smirl: Surprise minimizing rl in dynamic environments," *arXiv preprint arXiv:1912.05510*, 2020.

[81] C. Kaplanis, M. Shanahan, and C. Clopath, "Policy consolidation for continual reinforcement learning," *arXiv preprint arXiv:1902.00255*, 2019.

[82] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," in *Advances in Neural Information Processing Systems(NIPS)*, pp. 5824–5836, 2020.

[83] B. Liu, X. Liu, X. Jin, P. Stone, and Q. Liu, "Conflict-averse gradient descent for multi-task learning," in *Advances in Neural Information Processing Systems(NIPS)*, pp. 18878–18890, 2021.

[84] R. Yang, H. Xu, Y. Wu, and X. Wang, "Multi-task reinforcement learning with soft modularization," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4767–4777, 2020.

[85] L. Sun, H. Zhang, W. Xu, and M. Tomizuka, "Paco: Parameter-compositional multi-task reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 21495–21507, 2022.

[86] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, *et al.*, "A generalist agent," *arXiv preprint arXiv:2205.06175*, 2022.

[87] K.-H. Lee, O. Nachum, M. S. Yang, L. Lee, D. Freeman, S. Guadarrama, I. Fischer, W. Xu, E. Jang, H. Michalewski, *et al.*, "Multi-game decision transformers,"

Advances in Neural Information Processing Systems, vol. 35, pp. 27921–27936, 2022.

[88] A. Yu and R. Mooney, "Using both demonstrations and language instructions to efficiently learn robotic tasks," in *International Conference on Learning Representations (ICLR)*, 2023.

[89] A. Kumar, R. Agarwal, X. Geng, G. Tucker, and S. Levine, "Offline q-learning on diverse multi-task data both scales and generalizes," *arXiv preprint arXiv:2211.15144*, 2022.

[90] H. Yuan and Z. Lu, "Robust task representations for offline meta-reinforcement learning via contrastive learning," in *International Conference on Machine Learning*, pp. 25747–25759, PMLR, 2022.

[91] Y. Sun, S. Ma, R. Madaan, R. Bonatti, F. Huang, and A. Kapoor, "Smart: Self-supervised multi-task pretraining with control transformers," *arXiv preprint arXiv:2301.09816*, 2023.

[92] A. A. Taiga, R. Agarwal, J. Farebrother, A. Courville, and M. G. Bellemare, "Investigating multi-task pretraining and generalization in reinforcement learning," in *The Eleventh International Conference on Learning Representations*, 2022.

[93] A. Maurer, M. Pontil, and B. Romera-Paredes, "The benefit of multitask representation learning," *Journal of Machine Learning Research*, vol. 17, no. 81, pp. 1–32, 2016.

[94] C. DEramo, D. Tateo, A. Bonarini, M. Restelli, and J. Peters, "Sharing knowledge in multi-task deep reinforcement learning," in *Eighth International Conference on Learning Representations (ICLR 2020)*, OpenReview. net, 2020.

[95] K. Khetarpal, M. Riemer, I. Rish, and D. Precup, "Towards continual reinforcement learning: A review and perspectives," *Journal of Artificial Intelligence Research*,

vol. 75, pp. 1401–1476, 2022.

[96] B. Cheung, A. Terekhov, Y. Chen, P. Agrawal, and B. Olshausen, "Superposition of many models into one," *Advances in neural information processing systems*, vol. 32, 2019.

[97] M. Wortsman, V. Ramanujan, R. Liu, A. Kembhavi, M. Rastegari, J. Yosinski, and A. Farhadi, "Supermasks in superposition," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15173–15184, 2020.

[98] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell, "Policy distillation," *arXiv preprint arXiv:1511.06295*, 2015.

[99] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[100] T. Zhang, X. Wang, B. Liang, and B. Yuan, "Catastrophic interference in reinforcement learning: A solution based on context division and knowledge distillation," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[101] M. Igl, G. Farquhar, J. Luketina, J. Böhmer, and S. Whiteson, "Transient non-stationarity and generalisation in deep reinforcement learning," in *9th International Conference on Learning Representations*, 2021.

[102] Y. Oh, J. Shin, E. Yang, and S. J. Hwang, "Model-augmented prioritized experience replay," in *International Conference on Learning Representations*, 2021.

[103] C. Henning, M. Cervera, F. D'Angelo, J. Von Oswald, R. Traber, B. Ehret, S. Kobayashi, B. F. Grewe, and J. Sacramento, "Posterior meta-replay for continual learning," *Advances in neural information processing systems*, vol. 34, pp. 14135–14149, 2021.

[104] X.-H. Liu, Z. Xue, J. Pang, S. Jiang, F. Xu, and Y. Yu, "Regret minimization experience replay in off-policy reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 17604–17615, 2021.

[105] C. Atkinson, B. McCane, L. Szymanski, and A. Robins, "Pseudo-rehearsal: Achieving deep reinforcement learning without catastrophic forgetting," *Neurocomputing*, vol. 428, pp. 291–307, 2021.

[106] Z. Daniels, A. Raghavan, J. Hostetler, A. Rahman, I. Sur, M. Piacentino, and A. Divakaran, "Model-free generative replay for lifelong reinforcement learning: Application to starcraft-2," *arXiv preprint arXiv:2208.05056*, 2022.

[107] S. Kessler, J. Parker-Holder, P. J. Ball, S. Zohren, and S. J. Roberts, "Same state, different task: Continual reinforcement learning without interference," *ArXiv*, vol. abs/2106.02940, 2021.

[108] J.-B. Gaya, T. Doan, L. Caccia, L. Soulier, L. Denoyer, and R. Raileanu, "Building a subspace of policies for scalable continual learning," in *International Conference of Learning Representations*, 2023.

[109] W.-C. Tseng, J.-S. Lin, Y.-M. Feng, and M. Sun, "Toward robust long range policy transfer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 9958–9966, 2021.

[110] S. Lee, S. Behpour, and E. Eaton, "Sharing less is more: Lifelong learning in deep networks with selective layer transfer," in *International Conference on Machine Learning*, pp. 6065–6075, PMLR, 2021.

[111] M. B. Chang, S. Levine, A. Gupta, and T. L. Griffiths, "Automatically composing representation transformations as a means for generalization," in *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[112] C. Tessler, S. Givony, T. Zahavy, D. Mankowitz, and S. Mannor, "A deep hierarchical approach to lifelong learning in minecraft," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.

[113] V. Campos, A. Trott, C. Xiong, R. Socher, X. Giró-i Nieto, and J. Torres, "Explore, discover and learn: Unsupervised discovery of state-covering skills," in *International Conference on Machine Learning*, pp. 1317–1327, PMLR, 2020.

[114] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, "Diversity is all you need: Learning skills without a reward function," in *International Conference on Learning Representations*, 2018.

[115] A. Barreto, D. Borsa, S. Hou, G. Comanici, E. Aygün, P. Hamel, D. Toyama, S. Mourad, D. Silver, D. Precup, *et al.*, "The option keyboard: Combining skills in reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[116] K. Lu, A. Grover, P. Abbeel, and I. Mordatch, "Reset-free lifelong learning with skill-space planning," in *International Conference on Learning Representations*, 2020.

[117] C. Jin, A. Krishnamurthy, M. Simchowitz, and T. Yu, "Reward-free exploration for reinforcement learning," in *International Conference on Machine Learning*, pp. 4870–4879, PMLR, 2020.

[118] A. Touati and Y. Ollivier, "Learning one representation to optimize all rewards," *Advances in Neural Information Processing Systems*, vol. 34, pp. 13–23, 2021.

[119] W. Zhang, D. Zhou, and Q. Gu, "Reward-free model-based reinforcement learning with linear function approximation," *Advances in Neural Information Processing Systems*, vol. 34, pp. 1582–1593, 2021.

[120] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019.

[121] H. Fu, S. Yu, M. Littman, and G. Konidaris, "Model-based lifelong reinforcement learning with bayesian exploration," *Advances in Neural Information Processing Systems*, vol. 35, pp. 32369–32382, 2022.

[122] R. Aljundi, P. Chakravarty, and T. Tuytelaars, "Expert gate: Lifelong learning with a network of experts," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3366–3375, 2017.

[123] A. Xie and C. Finn, "Lifelong robotic reinforcement learning by retaining experiences," in *Conference on Lifelong Learning Agents*, pp. 838–855, 2022.

[124] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[125] Z. Zhu, K. Lin, A. K. Jain, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 11, pp. 13344–13362, 2023.

[126] A. Y. Ng, D. Harada, and S. J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, (San Francisco, CA, USA), p. 278,Äì287, Morgan Kaufmann Publishers Inc., 1999.

[127] E. Wiewiora, G. Cottrell, and C. Elkan, "Principled methods for advising reinforcement learning agents," in *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, p. 792,Äì799, AAAI Press, 2003.

[128] A. Harutyunyan, S. Devlin, P. Vrancx, and A. Nowe, "Expressing arbitrary reward functions as potential-based advice," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, Feb. 2015.

[129] D. P. Bertsekas, "Approximate policy iteration: A survey and some new methods," *Journal of Control Theory and Applications*, vol. 9, no. 3, pp. 310–335, 2011.

[130] J. Chemali and A. Lazaric, "Direct policy iteration with demonstrations," in *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, p. 3380,Äì3386, AAAI Press, 2015.

[131] B. Kim, A.-m. Farahmand, J. Pineau, and D. Precup, "Learning from limited demonstrations," in *Advances in Neural Information Processing Systems* (C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, eds.), vol. 26, Curran Associates, Inc., 2013.

[132] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, G. Dulac-Arnold, J. Agapiou, J. Z. Leibo, and A. Gruslys, "Deep q-learning from demonstrations," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18, AAAI Press, 2018.

[133] T. Brys, A. Harutyunyan, H. B. Suay, S. Chernova, M. E. Taylor, and A. Nowé, "Reinforcement learning from demonstration through shaping," in *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, p. 3352,Äì3358, AAAI Press, 2015.

[134] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*

(D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, eds.), pp. 4565–4573, 2016.

[135] B. Kang, Z. Jie, and J. Feng, "Policy optimization with demonstrations," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018* (J. G. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 2474–2483, PMLR, 2018.

[136] A. A. Rusu, S. G. Colmenarejo, Ç. Gülçehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell, "Policy distillation," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2016.

[137] E. Parisotto and R. Salakhutdinov, "Neural map: Structured memory for deep reinforcement learning," *arXiv preprint arXiv:1702.08360*, 2017.

[138] W. M. Czarnecki, R. Pascanu, S. Osindero, S. M. Jayakumar, G. Swirszcz, and M. Jaderberg, "Distilling policy distillation," in *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan* (K. Chaudhuri and M. Sugiyama, eds.), vol. 89 of *Proceedings of Machine Learning Research*, pp. 1331–1340, PMLR, 2019.

[139] F. Fernández and M. M. Veloso, "Probabilistic policy reuse in a reinforcement learning agent," in *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006* (H. Nakashima, M. P. Wellman, G. Weiss, and P. Stone, eds.), pp. 720–727, ACM, 2006.

[140] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[141] A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, D. Silver, and H. van Hasselt, "Successor features for transfer in reinforcement learning," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA* (I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, eds.), pp. 4055–4065, 2017.

[142] G. D. Konidaris and A. G. Barto, "Autonomous shaping: knowledge transfer in reinforcement learning," in *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006* (W. W. Cohen and A. W. Moore, eds.), vol. 148 of *ACM International Conference Proceeding Series*, pp. 489–496, ACM, 2006.

[143] H. Bou-Ammar and M. E. Taylor, "Reinforcement learning transfer via common subspaces," in *Adaptive and Learning Agents - International Workshop, ALA 2011, Held at AAMAS 2011, Taipei, Taiwan, May 2, 2011, Revised Selected Papers* (P. Vrancx, M. Knudson, and M. Grzes, eds.), vol. 7113 of *Lecture Notes in Computer Science*, pp. 21–36, Springer, 2011.

[144] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "Learning invariant feature spaces to transfer skills with reinforcement learning," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[145] A. Lazaric, M. Restelli, and A. Bonarini, "Transfer of samples in batch reinforcement learning," in *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008* (W. W. Cohen, A. McCallum, and S. T. Roweis, eds.), vol. 307 of *ACM International Conference Proceeding Series*, pp. 544–551, ACM, 2008.

[146] H. Bou-Ammar, K. Tuyls, M. E. Taylor, K. Driessens, and G. Weiss, "Reinforcement learning transfer via sparse coding," in *International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012, Valencia, Spain, June 4-8, 2012 (3 Volumes)* (W. van der Hoek, L. Padgham, V. Conitzer, and M. Winikoff, eds.), pp. 383–390, IFAAMAS, 2012.

[147] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *CoRR*, vol. abs/1606.04671, 2016.

[148] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra, "Pathnet: Evolution channels gradient descent in super neural networks," *CoRR*, vol. abs/1701.08734, 2017.

[149] P. Dayan, "Improving generalization for temporal difference learning: The successor representation," *Neural Comput.*, vol. 5, no. 4, pp. 613–624, 1993.

[150] T. D. Kulkarni, A. Saeedi, S. Gautam, and S. J. Gershman, "Deep successor reinforcement learning," *CoRR*, vol. abs/1606.02396, 2016.

[151] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep reinforcement learning with successor features for navigation across similar environments," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*, pp. 2371–2378, IEEE, 2017.

[152] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015* (F. R. Bach and D. M. Blei, eds.), vol. 37 of *JMLR Workshop and Conference Proceedings*, pp. 1312–1320, JMLR.org, 2015.

[153] A. Barreto, D. Borsa, J. Quan, T. Schaul, D. Silver, M. Hessel, D. J. Mankowitz, A. Zídek, and R. Munos, "Transfer in deep reinforcement learning using successor features and generalised policy improvement," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018* (J. G. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 510–519, PMLR, 2018.

[154] B. Eysenbach, S. Asawa, S. Chaudhari, S. Levine, and R. Salakhutdinov, "Off-dynamics reinforcement learning: Training for transfer with domain classifiers," in *International Conference on Learning Representations(ICLR)*, 2021.

[155] A. Chen, A. Sharma, S. Levine, and C. Finn, "You only live once: Single-life reinforcement learning," in *Advances in Neural Information Processing Systems(NIPS)*, vol. 35, pp. 14784–14797, 2022.

[156] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "Rl2: Fast reinforcement learning via slow reinforcement learning," *arXiv preprint arXiv:1611.02779*, 2016.

[157] E. Aghapour and N. Ayanian, "Double meta-learning for data efficient policy optimization in non-stationary environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9935–9942, 2021.

[158] Z. Xu, X. Chen, and L. Cao, "Fast task adaptation based on the combination of model-based and gradient-based meta learning," *IEEE Transactions on Cybernetics*, vol. 52, no. 6, pp. 5209–5218, 2020.

[159] A. Hallak, D. Di Castro, and S. Mannor, "Contextual markov decision processes," *arXiv preprint arXiv:1502.02259*, 2015.

[160] A. Modi and A. Tewari, "Contextual markov decision processes using generalized linear models," *CoRR*, vol. abs/1903.06187, 2019.

[161] A. Modi, N. Jiang, S. Singh, and A. Tewari, "Markov decision processes with continuous side information," in *Algorithmic Learning Theory*, pp. 597–618, PMLR, 2018.

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

[163] N. Jiang, A. Krishnamurthy, A. Agarwal, J. Langford, and R. E. Schapire, "Contextual decision processes with low bellman rank are pac-learnable," in *International Conference on Machine Learning*, pp. 1704–1713, PMLR, 2017.

[164] J. Kwon, Y. Efroni, C. Caramanis, and S. Mannor, "Rl for latent mdps: Regret guarantees and a lower bound," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24523–24534, 2021.

[165] G. Tennenholtz, N. Merlis, L. Shani, M. Mladenov, and C. Boutilier, "Reinforcement learning with history dependent dynamic contexts," in *International Conference on Machine Learning*, pp. 34011–34053, PMLR, 2023.

[166] H. Mao, S. B. Venkatakrishnan, M. Schwarzkopf, and M. Alizadeh, "Variance reduction for reinforcement learning in input-driven environments," *arXiv preprint arXiv:1807.02264*, 2018.

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

[168] K. J. Åström, "Optimal control of markov processes with incomplete state information i," *Journal of mathematical analysis and applications*, vol. 10, pp. 174–205, 1965.

[169] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[170] N. Vlassis, M. L. Littman, and D. Barber, "On the computational complexity of stochastic controller optimization in pomdps," *ACM Transactions on Computation Theory (TOCT)*, vol. 4, no. 4, pp. 1–8, 2012.

[171] Y. Xiong, N. Chen, X. Gao, and X. Zhou, "Sublinear regret for learning pomdps," *Production and Operations Management*, vol. 31, no. 9, pp. 3491–3504, 2022.

[172] Z. D. Guo, S. Doroudi, and E. Brunskill, "A pac rl algorithm for episodic pomdps," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics* (A. Gretton and C. C. Robert, eds.), vol. 51 of *Proceedings of Machine Learning Research*, (Cadiz, Spain), pp. 510–518, PMLR, 09–11 May 2016.

[173] A. Anandkumar, D. Hsu, and S. M. Kakade, "A method of moments for mixture models and hidden markov models," in *Proceedings of the 25th Annual Conference on Learning Theory* (S. Mannor, N. Srebro, and R. C. Williamson, eds.), vol. 23 of *Proceedings of Machine Learning Research*, (Edinburgh, Scotland), pp. 33.1–33.34, PMLR, 25–27 Jun 2012.

[174] C. Jin, S. Kakade, A. Krishnamurthy, and Q. Liu, "Sample-efficient reinforcement learning of undercomplete pomdps," *Advances in Neural Information Processing Systems*, vol. 33, pp. 18530–18539, 2020.

[175] H. Guo, Q. Cai, Y. Zhang, Z. Yang, and Z. Wang, "Provably efficient offline reinforcement learning for partially observable markov decision processes," in *International Conference on Machine Learning*, pp. 8016–8038, PMLR, 2022.

[176] Y. Jin, Z. Yang, and Z. Wang, "Is pessimism provably efficient for offline rl?," in *International Conference on Machine Learning*, pp. 5084–5096, PMLR, 2021.

[177] A. Zanette, M. J. Wainwright, and E. Brunskill, "Provable benefits of actor-critic methods for offline reinforcement learning," *Advances in neural information processing systems*, vol. 34, pp. 13626–13640, 2021.

[178] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE transactions on knowledge and data engineering*, vol. 31, no. 12, pp. 2346–2363, 2018.

[179] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

[180] P. Domingos and G. Hulten, "Mining High-Speed Data Streams," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, (New York, NY, USA), pp. 71–80, Association for Computing Machinery, 2000.
event-place: Boston, Massachusetts, USA.

[181] C. Manapragada, G. I. Webb, and M. Salehi, "Extremely Fast Decision Tree," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, (New York, NY, USA), pp. 1953–1962, Association for Computing Machinery, 2018.
event-place: London, United Kingdom.

[182] B. Krawczyk, "Tensor decision trees for continual learning from drifting data streams," *Machine Learning*, vol. 110, pp. 3015–3035, Dec. 2021.

[183] V. Losing, B. Hammer, and H. Wersing, "KNN Classifier with Self Adjusting Memory for Heterogeneous Concept Drift," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pp. 291–300, 2016.

[184] F. Dong, J. Lu, Y. Song, F. Liu, and G. Zhang, "A Drift Region-Based Data Sample Filtering Method," *IEEE Transactions on Cybernetics*, vol. 52, no. 9, pp. 9377–9390, 2022.

[185] H. Yu, J. Lu, and G. Zhang, "Continuous Support Vector Regression for Nonstationary Streaming Data," *IEEE Transactions on Cybernetics*, vol. 52, no. 5, pp. 3592–3605, 2022.

[186] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet, "A Survey on Ensemble Learning for Data Stream Classification," *ACM Comput. Surv.*, vol. 50, Mar. 2017.

Place: New York, NY, USA Publisher: Association for Computing Machinery.

[187] J. Z. Kolter and M. A. Maloof, "Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts," *Journal of Machine Learning Research*, vol. 8, no. 91, pp. 2755–2790, 2007.

[188] Y. Sun, K. Tang, Z. Zhu, and X. Yao, "Concept Drift Adaptation by Exploiting Historical Knowledge," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4822–4832, 2018.

[189] P. Zhao, L.-W. Cai, and Z.-H. Zhou, "Handling concept drift via model reuse," *Machine Learning*, vol. 109, pp. 533–568, Mar. 2020.

[190] F. Soleymani and E. Paquet, "Financial portfolio optimization with online deep reinforcement learning and restricted stacked autoencoder‚ÄîDeepBreath," *Expert Systems with Applications*, vol. 156, p. 113456, 2020.

[191] M. N. Fekri, H. Patel, K. Grolinger, and V. Sharma, "Deep learning for load forecasting with smart meter data: Online Adaptive Recurrent Neural Network," *Applied Energy*, vol. 282, p. 116177, 2021.

[192] S. Yen, M. Moh, and T.-S. Moh, "CausalConvLSTM: Semi-Supervised Log Anomaly Detection Through Sequence Modeling," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 1334–1341, 2019.

[193] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 723–773, 2012.

[194] J. Bernstein, A. Vahdat, Y. Yue, and M.-Y. Liu, "On the distance between two neural networks and the stability of learning," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 21370–21381, 2020.

[195] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 1057–1063, 2000.

[196] B. C. Csáji and L. Monostori, "Value function based reinforcement learning in changing Markovian environments," *Journal of Machine Learning Research*, vol. 9, no. 54, pp. 1679–1709, 2008.

[197] Z. Huang, W. Shao, X. Wang, L. Lin, and P. Luo, "Rethinking the pruning criteria for convolutional neural network," in *Advances in Neural Information Processing Systems(NIPS)*, pp. 16305–16318, 2021.

[198] F. Pukelsheim, "The three sigma rule," *The American Statistician*, vol. 48, no. 2, pp. 88–91, 1994.

[199] E. R. Ziegel, "Statistical case studies for industrial process improvement," *Technometrics*, vol. 40, no. 2, pp. 163–163, 1998.

[200] A. Jacot, C. Hongler, and F. Gabriel, "Neural tangent kernel: Convergence and generalization in neural networks," in *Advances in Neural Information Processing*

Systems (NIPS), pp. 8580–8589, 2018.

[201] J. Lu, J. Xuan, G. Zhang, and X. Luo, "Structural property-aware multilayer network embedding for latent factor analysis," *Pattern Recognition*, vol. 76, pp. 228–241, 2018.

[202] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[203] M. Chevalier-Boisvert, L. Willems, and S. Pal, "Minimalistic gridworld environment for openai gym," 2018.

[204] M. Wydmuch, M. Kempka, and W. Jaśkowski, "Vizdoom competitions: Playing doom from pixels," *IEEE Transactions on Games*, vol. 11, no. 3, pp. 248–259, 2019.

[205] A. Mosavi, Y. Faghan, P. Ghamisi, P. Duan, S. F. Ardabili, E. Salwana, and S. S. Band, "Comprehensive review of deep reinforcement learning methods and applications in economics," *Mathematics*, vol. 8, no. 10, p. 1640, 2020.

[206] A. Delarue, R. Anderson, and C. Tjandraatmadja, "Reinforcement learning with combinatorial actions: An application to vehicle routing," *Advances in Neural Information Processing Systems*, vol. 33, pp. 609–620, 2020.

[207] K. M. Oikonomou, I. Kansizoglou, and A. Gasteratos, "A hybrid spiking neural network reinforcement learning agent for energy-efficient object manipulation," *Machines*, vol. 11, no. 2, p. 162, 2023.

[208] M. Hu, J. Zhang, L. Matkovic, T. Liu, and X. Yang, "Reinforcement learning in medical image analysis: Concepts, applications, challenges, and future directions," *Journal of Applied Clinical Medical Physics*, vol. 24, no. 2, p. e13898, 2023.

[209] P. Tiwari, A. Lakhan, R. H. Jhaveri, and T.-M. Gronli, "Consumer-centric internet of medical things for cyborg applications based on federated reinforcement learning," *IEEE Transactions on Consumer Electronics*, 2023.

[210] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *The Conference on Robot Learning (CoRL)*, pp. 1094–1100, 2020.

[211] A. Pacchiano, J. Parker-Holder, Y. Tang, K. Choromanski, A. Choromanska, and M. Jordan, "Learning to score behaviors for guided policy optimization," in *International Conference on Machine Learning*, pp. 7445–7454, 2020.

[212] C. Villani *et al.*, *Optimal transport: old and new*, vol. 338.
Springer, 2009.

[213] V. M. Panaretos and Y. Zemel, "Statistical aspects of wasserstein distances," *Annual review of statistics and its application*, vol. 6, pp. 405–431, 2019.

[214] I. Olkin and F. Pukelsheim, "The distance between two random vectors with given dispersion matrices," *Linear Algebra and its Applications*, vol. 48, pp. 257–263, 1982.

[215] L. Xu, "Approximation of stable law in wasserstein-1 distance by stein's method," *The Annals of Applied Probability*, vol. 29, no. 1, pp. 458–504, 2019.

[216] W. J. Welch, "Construction of permutation tests," *Journal of the American Statistical Association*, vol. 85, no. 411, pp. 693–698, 1990.

[217] C. D. Van Borkulo, R. van Bork, L. Boschloo, J. J. Kossakowski, P. Tio, R. A. Schoevers, D. Borsboom, and L. J. Waldorp, "Comparing network structures on three aspects: A permutation test.," *Psychological methods*, 2022.

[218] P. Good, *Permutation tests: a practical guide to resampling methods for testing hypotheses*.

Springer Science & Business Media, 2013.

[219] M. Wydmuch, M. Kempka, and W. Jaśkowski, "ViZDoom Competitions: Playing Doom from Pixels," *IEEE Transactions on Games*, vol. 11, no. 3, pp. 248–259, 2019.

The 2022 IEEE Transactions on Games Outstanding Paper Award.

[220] A. Bellot and M. van der Schaar, "A kernel two-sample test with selection bias," in *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence* (C. de Campos and M. H. Maathuis, eds.), vol. 161 of *Proceedings of Machine Learning Research*, pp. 205–214, 2021.

[221] Y. Sasaki *et al.*, "The truth of the f-measure," *Teach tutor mater*, vol. 1, no. 5, pp. 1–5, 2007.

[222] A. Benjamin, D. Rolnick, and K. Kording, "Measuring and regularizing networks in function space," in *International Conference on Learning Representations*, 2018.

[223] L. V. Kantorovich, "Mathematical methods of organizing and planning production," *Management science*, vol. 6, no. 4, pp. 366–422, 1960.

[224] T. Furmston and D. Barber, "Variational methods for reinforcement learning," in *International Conference on Artificial Intelligence and Statistics*, pp. 241–248, 2010.

[225] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, vol. 2.

MIT press Cambridge, MA, 2006.

[226] M. E. E. Khan, A. Immer, E. Abedi, and M. Korzepa, "Approximate inference turns deep networks into gaussian processes," *Advances in neural information processing systems*, vol. 32, 2019.

[227] T. Galy-Fajou, F. Wenzel, C. Donner, and M. Opper, "Multi-class gaussian process classification made conjugate: Efficient inference via data augmentation," in *Uncertainty in artificial intelligence*, pp. 755–765, PMLR, 2020.

[228] D. R. Burt, S. W. Ober, A. Garriga-Alonso, and M. van der Wilk, "Understanding variational inference in function-space," in *Third Symposium on Advances in Approximate Bayesian Inference*, 2020.

[229] S. Sujit, S. Nath, P. Braga, and S. Ebrahimi Kahou, "Prioritizing samples in reinforcement learning with reducible loss," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[230] L. A. Atherton, D. Dupret, and J. R. Mellor, "Memory trace replay: the shaping of memory consolidation by neuromodulation," *Trends in neurosciences*, vol. 38, no. 9, pp. 560–570, 2015.

[231] G. Z. Grudic and L. H. Ungar, "Localizing search in reinforcement learning," in *AAAI/IAAI*, pp. 590–595, 2000.

[232] P. Ma, M. Mahoney, and B. Yu, "A statistical perspective on algorithmic leveraging," in *International conference on machine learning*, pp. 91–99, PMLR, 2014.

[233] M. P. Deisenroth, G. Neumann, J. Peters, *et al.*, "A survey on policy search for robotics," *Foundations and trends in Robotics*, vol. 2, no. 1-2, pp. 388–403, 2013.

[234] J. Ke, F. Xiao, H. Yang, and J. Ye, "Learning to delay in ride-sourcing systems: A multi-agent deep reinforcement learning framework," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 5, pp. 2280–2292, 2022.

[235] C. Angermueller, D. Dohan, D. Belanger, R. Deshpande, K. Murphy, and L. Colwell, "Model-based reinforcement learning for biological sequence design," in *International conference on learning representations*, 2019.

[236] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 153–173, 2017.

[237] J. Lu, C. Yang, X. Gao, L. Wang, C. Li, and G. Chen, "Reinforcement learning with sequential information clustering in real-time bidding," in *ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 1633–1641, 2019.

[238] S. Chawla, N. R. Devanur, A. R. Karlin, and B. Sivan, "Simple pricing schemes for consumers with evolving values," in *Annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pp. 1476–1490, 2016.

[239] C. Chen, H. Wei, N. Xu, G. Zheng, M. Yang, Y. Xiong, K. Xu, and Z. Li, "Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control," in *AAAI Conference on Artificial Intelligence (AAAI)*, vol. 34, pp. 3414–3421, 2020.

[240] E. Bengio, J. Pineau, and D. Precup, "Interference and generalization in temporal difference learning," in *International Conference on Machine Learning*, pp. 767–777, PMLR, 2020.

[241] E. Lecarpentier and E. Rachelson, "Non-stationary markov decision processes, a worst-case approach using model-based reinforcement learning," *Advances in neural information processing systems*, vol. 32, 2019.

[242] M. Karl, M. Soelch, J. Bayer, and P. van der Smagt, "Deep variational bayes filters: Unsupervised learning of state space models from raw data," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[243] A. Doerr, C. Daniel, M. Schiegg, N.-T. Duy, S. Schaal, M. Toussaint, and T. Sebastian, "Probabilistic recurrent state-space models," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 1280–1289, PMLR, 10–15 Jul 2018.

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

[245] R. Rafailov, T. Yu, A. Rajeswaran, and C. Finn, "Offline reinforcement learning from images with latent space models," in *Learning for Dynamics and Control*, pp. 1154–1168, PMLR, 2021.

[246] A. G. Richards, *Robust constrained model predictive control*. PhD thesis, Massachusetts Institute of Technology, 2005.

[247] R. Alami, O. Maillard, and R. Féraud, "Restarted bayesian online change-point detector achieves optimal detection delay," in *International conference on machine learning*, pp. 211–221, PMLR, 2020.

[248] F. Pukelsheim, "The three sigma rule," *The American Statistician*, vol. 48, pp. 88–91, 1994.

[249] R. Y. Rubinstein, "Optimization of computer simulation models with rare events," *European Journal of Operational Research*, vol. 99, no. 1, pp. 89–112, 1997.

[250] H. Bharadhwaj, K. Xie, and F. Shkurti, "Model-predictive control via cross-entropy and gradient-based optimization," in *Learning for Dynamics and Control*, pp. 277–286, PMLR, 2020.

[251] C. Pinneri, S. Sawant, S. Blaes, J. Achterhold, J. Stueckler, M. Rolinek, and G. Martius, "Sample-efficient cross-entropy method for real-time planning," in *Confer-*

ence on Robot Learning, pp. 1049–1065, PMLR, 2021.

[252] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL* (A. Moschitti, B. Pang, and W. Daelemans, eds.), pp. 1724–1734, ACL, 2014.

[253] E. Leurent, "An environment for autonomous driving decision-making." `https://github.com/eleurent/highway-env`, 2018.