EDSVM: A Novel Approach for Environment Drift Detection in Reinforcement Learning Using Synthetic Drifted Examples

Zhizhou Fang[a,b,*,1] **and Uwe Zdun**[b,**,1]

[a]Doctoral School Computer Science (DoCS), University of Vienna, Austria
[b]Research Group Software Architecture, Faculty of Computer Science, University of Vienna, Austria
ORCID (Zhizhou Fang): https://orcid.org/0000-0002-7013-8182, ORCID (Uwe Zdun): https://orcid.org/0000-0002-6233-2591

Abstract. In Reinforcement Learning (RL) environments, detecting environment drift is essential for maintaining robust policy performance in production systems, particularly within the context of MLOps. This paper proposes EDSVM, a novel environment drift detection method, which trains Support Vector Machines on undrifted and synthetic drifted examples generated by altering transition dynamics. By using decision function values as drift indicators, our method achieves competitive results compared to state-of-the-art baselines for the area-under-the-curve (AUC) metric. Additionally, we evaluate the performance of EDSVM when integrated with various Change Point Detection algorithms in terms of delay and false alarms, highlighting its potential for automating the monitoring of RL policies and supporting adaptive updates to production pipelines in MLOps workflows.

1 Introduction

Reinforcement Learning (RL) is a branch of Machine Learning (ML) in which agents learn to make decisions by interacting with an environment. Rather than being explicitly instructed on which actions to take, RL agents discover optimal policies through trial and error, guided by the rewards they receive from the environment [26]. In production settings, maintaining the performance of RL agents over time is critical, particularly in dynamic environments where subtle changes can significantly impact decision-making effectiveness. Machine Learning Operations (MLOps) provides a framework for efficiently deploying, monitoring, and maintaining ML models, including RL policies, in production environments [23]. Li et al. [18] proposed the concept of Reinforcement Learning Operations (RLOps), applying MLOps to RL. RLOps focuses on the practices that deliver RL policies to production environments. Within RL, MLOps is particularly crucial for monitoring agent performance and detecting potential degradation caused by environment drift—changes in the underlying transition dynamics [28]. Environment drift poses a significant challenge in RL systems, as even minor shifts in the environment can lead to reduced policy effectiveness and necessitate timely detection and intervention.

* Corresponding Author. Email: zhizhou.fang@univie.ac.at
** Corresponding Author. Email: uwe.zdun@univie.ac.at

[1] Equal contribution.

This study proposes a novel approach for detecting environment drift in RL systems by leveraging Support Vector Machines (SVMs) trained on both undrifted examples from training environments and synthetic drifted examples. The synthetic drifted examples are generated by modifying the transition dynamics, where synthetic observations are created through interpolation between current, past, and future states, with drift rates sampled from predefined intervals. By incorporating these synthetic examples into the training process, our method effectively enables SVMs to distinguish between drifted and undrifted data. The decision function values (distances to the decision boundary) are used as anomaly scores, i.e., indicators of drift, monitored by change point detection algorithms such as Page-Hinkley [22], ADWIN [5], and KSWIN [24], allowing us to detect environment drift accurately.

We address the following research questions:

- **RQ1** Can synthetic drifted examples effectively enhance environment drift detection of SVM in reinforcement learning systems?
- **RQ2** How do different change point detection algorithms impact the effectiveness of our environment drift detection methods?

Through these research questions, our goal is to evaluate the effectiveness of SVM-based monitoring metrics and their ability to improve the detection of environment drift in RL systems. The main contributions of this paper are as follows:

1. We propose a novel approach to environment drift detection in RL systems by leveraging SVMs trained on synthetic drifted examples. This approach introduces a technique for generating synthetic examples that simulate various drift scenarios by modifying transition dynamics.
2. We perform a comprehensive experimental evaluation across multiple RL environments, benchmarking our method against state-of-the-art baselines. The results demonstrate our approach's effectiveness, robustness, and competitive performance, as well as its ability to distinguish undrifted and drifted environments under diverse scenarios.
3. We explore the effects of integrating different change point detection algorithms, including Page-Hinkley, ADWIN, and KSWIN, into our method. These algorithms are designed to detect shifts in data distributions over time, and their integration enables us to en-

hance the precision and reliability of our environment drift detection framework. Specifically, we evaluate their impact on two critical metrics: detection delay and false alarms. By analyzing these metrics in combination with the decision function values produced by our SVM-based method, we aim to identify the most effective algorithm for balancing timely detection while minimizing unnecessary interventions. Our results offer insights into the trade-offs among these algorithms and their suitability for dynamic RL environments.

The remainder of this paper is structured as follows: Section 2 outlines the problem statement and introduces the research questions that guide our study. Section 3 provides an overview of related work on environment drift detection in RL systems, situating our approach within the existing literature. Section 4 details our proposed methodology, including the generation of synthetic drifted examples, the training of SVMs for drift detection, and the integration of change point detection algorithms to enhance performance in production settings. Section 5 describes the experimental setup, presents the results, and compares the effectiveness of our method against state-of-the-art baselines. Section 6 discusses our findings, highlighting the strengths and limitations of the proposed approach, and outlines potential directions for future research. Finally, Section 7 discusses the threats to the validity of our work, and Section 8 concludes.

2 Problem Statement

Environment drift in RL refers to changes in the underlying Markov decision process of the environment, characterized by shifts in the conditional distribution $P(o_{t+1}|o_t, a_t)$, where o_t and o_{t+1} represent the observations at time steps t and $t + 1$, respectively, and a_t is the action taken at time t. Detecting environment drift intends to identify the specific time step when this shift occurs, allowing timely adjustments to the RL policy.

3 Related Works

This section will discuss related works on coping with environmental drift detection and compare these works with our method.

3.1 Concept Drift Detection in Supervised Learning

In supervised learning, concept drift refers to shifts in the relationship between input features and the target variable over time [14]. This phenomenon can be represented as changes in the conditional distribution $P(y|x)$, where x denotes the input features and y denotes the target variable. Detecting concept drift is crucial for maintaining the performance of ML models in dynamic environments, and numerous studies have been conducted to address this challenge. Many approaches to concept drift detection rely on monitoring the prediction error of ML models through statistical tests. For instance, the Drift Detection Method (DDM), proposed by Gama et al. [13], tracks the online error rate of a model and raises the alarm when statistically significant changes are observed. Similarly, the Early Drift Detection Method (EDDM), introduced by Baena-Garcia et al. [2], focuses on detecting gradual drifts by analyzing the intervals between classification errors. Building on these methods, Frias-Blanco et al. [12] introduced the Hoeffding Drift Detection Method (HDDM), which enhances DDM by incorporating Hoeffding's inequality to improve its sensitivity to drift. Additionally, Yan [30] proposed the Accurate Concept Drift Detection Method (ACDDM), which also leverages Hoeffding's inequality to detect inconsistencies in error rates for drift detection. A key limitation of these methods is their reliance on labeled data, which can be difficult to obtain in production environments. To address this, Baier et al. [3] developed a concept drift detection method for neural networks that exploits model uncertainty, reducing the dependence on labeled data. For a broader perspective, Bayram et al. [4] provides a comprehensive review of existing concept drift detection methods, offering valuable insights into the field's current state.

3.2 Environment Drift Detection in Reinforcement Learning

Compared with the large number of drift detection works for supervised learning models, environment drift detection for reinforcement agents has not been sufficiently studied.

Wang et al. [28] proposed a method where a detector was employed to detect environment drift by exploring the environment with equal probability. Incremental learning was then used to update the Q-function, allowing it to adapt to the drifted environment. However, this approach has several limitations: (1) Deploying a detector for exploration in the production environments is impractical. (2) The method defines environment drift in the context of a deterministic environment, making it less applicable to stochastic environments. (3) The incremental learning strategy is tailored for tabular Q-learning, limiting its applicability to other reinforcement learning algorithms.

Greenberg and Mannor [15] introduced a method for detecting environment drift by monitoring the deterioration of episodic performance. This approach tracks the episodic returns of an RL agent and sends alarms when significant declines occur, indicating potential changes in the environment. While effective in identifying drift, this method has limitations in realistic production settings where episodes are not well-defined. As a result, more general approaches that do not rely on episodic structures are necessary for effectively detecting and managing environment drift in continuous production environments.

Fang and Zdun [11] introduced GPAction, a method that utilizes a Gaussian Process to predict actions based on state transitions. The prediction error of these actions is used as an indicator to detect drifts. However, this approach is specifically designed for environments with continuous action spaces, limiting its applicability to discrete action space scenarios.

In contrast to these previous works, our approach can detect drifts in RL environments with infinite state spaces, continuous and discrete action spaces, and stochastic dynamics, all at the granularity of individual time steps. Additionally, our method operates independently of the underlying RL policies, offering a more flexible and general solution for environment drift detection.

Several studies frame environment drift detection as either an Anomaly Detection (AD) or an Out-of-Distribution (OOD) detection problem. Müller et al [21] developed the concept of anomaly detection in reinforcement learning. Other works [9, 17] employed neural networks to predict future states, using the prediction error as an indicator of out-of-distribution events in the context of the respective environments. Wang et al. [27] proposed an anomaly detection framework, OIL-AD, for sequential decision processes. Their approach utilizes a transformer to learn both the Q-function and state-value function, using the resulting function values as features for a detection model to predict the anomaly level.

Compared to these methods, which rely on complex neural networks and transformers, our approach leverages the simplicity and efficiency of Support Vector Machines (SVMs). By using SVMs trained on synthetic drifted data, our method offers a more computationally efficient solution without the need for intricate network architectures or extensive training. This simplicity makes our approach more practical for real-time environment drift detection, especially in scenarios where computational resources are limited.

4 Methodology

This section outlines our methodology for detecting environment drift in Reinforcement Learning environments, organized into three key components. First, we detail the process of generating synthetic drifted examples during training, a critical step in enabling the drift detector to distinguish between drifted and undrifted scenarios. These synthetic examples replicate subtle environment changes, allowing the model to learn patterns indicative of drift. Second, we describe the application of Support Vector Machines for drift detection, along with optimization strategies designed to enhance the model's performance and sensitivity. Finally, we explain the integration of change point detection algorithms, such as Page-Hinkley, ADWIN, and KSWIN, into our framework. A diagram illustrating the full EDSVM workflow is provided in Figure 1.

4.1 Generating Synthetic Drifted Examples for Training

To effectively train a model for environment drift detection, we generate synthetic drifted examples that simulate the behavior of the environment under drifted conditions. These synthetic examples are the foundation for distinguishing between undrifted and drifted states during the model's training phase.

We begin by assuming an RL environment where the dynamic is defined by the conditional distribution $P(o_{t+1}|o_t, a_t)$, with o_t representing the observation at time t, and a_t the action that is taken. To introduce synthetic drift, we modify this transition in a controlled manner. For each undrifted example (o_t, a_t, o_{t+1}), we synthesize a corresponding drifted example (o_t, a_t, o'_{t+1}), where o'_{t+1} denotes the fabricated observation of the synthetic drifted example.

The fabricated observation o'_{t+1} is generated by applying a weighted combination of the corresponding original observation o_{t+1} and the adjacency observations o_t or o_{t+2}, as shown by Equation 1 and 2, respectively.

$$o'_{t+1} = (1-\lambda)o_{t+1} + \lambda o_t + \epsilon, \epsilon \sim \mathcal{N}(0, 0.005) \quad (1)$$

$$o'_{t+1} = (1-\lambda)o_{t+1} + \lambda o_{t+2} + \epsilon, \epsilon \sim \mathcal{N}(0, 0.005) \quad (2)$$

By incorporating o_t in Equation 1, we introduce a drift that reflects a situation where the drifted observation o'_{t+1} is closer to the previous observation o_t than the undrifted observation o_{t+1}. This mimics cases where the environment struggles to progress normally. On the other hand, using o_{t+2} in Equation 2 introduces a forward-looking aspect, creating a drifted state that suggests an acceleration in the environment dynamics. λ is a small positive number denoting the drift rate sampled from a uniform distribution $\mathcal{U}(0.02, 0.05)$. Keeping λ small ensures that the fabricated observation of the drifted example remains close to the undrifted one, simulating subtle shifts in the environment dynamics. This helps train the model to detect more realistic, gradual and unobvious drifts often encountered in real-world scenarios [7]. ϵ represents a slight noise sampled from a normal distribution with a mean of 0 and a standard deviation of 0.005. Adding this noise combined with the varying drift rate λ introduces extra randomness to the fabricated observation, mimicking the inherent variability and uncertainty that can occur in real-world environments.

4.2 Training and Optimization of EDSVM

The motivation for using SVMs [8] in our approach is based on several key advantages that make SVMs particularly well-suited for detecting environment drift. Unlike other classifiers, such as Random Forests or Neural Networks, SVMs are designed to maximize the decision margin between classes, leading to a well-defined hyperplane that optimally separates positive and negative classes. This property is particularly valuable for detecting subtle shifts, as it enables SVMs to identify the most effective separation, even in high-dimensional feature spaces. Furthermore, SVMs tend to be less prone to overfitting, especially when working with limited training data, and their outputs, specifically the distances from the decision boundary, provide a clear and interpretable metric.

During training, the SVM is trained on a combination of undrifted and synthetic drifted examples, where each example is represented as a tuple $(o_t, a_t, \Delta o_t)$, with $\Delta o_t = o_{t+1} - o_t$ capturing the state change. The training process optimizes the SVM to maximize the margin between drifted (labeled as 1) and synthetic undrifted examples (labeled as -1), ensuring the model learns a clear boundary separating these classes. In this context, we use Optuna [1], a hyperparameter optimization tool based on Bayesian optimization using a tree-structured parzen estimator [29], to fine-tune the SVM hyperparameters, specifically aiming to maximize the difference in decision function values (distances from the hyperplane) between undrifted and synthetic drifted examples. This optimization ensures that the SVM can better distinguish subtle shifts in the environment dynamics.

4.3 Integration of Change Point Detection Algorithms

In production settings, instead of directly classifying each instance as drifted or undrifted, we continuously monitor the sequence of drift scores, defined as the values of negative decision functions. These decision function values correspond to the distances from the decision boundary, with significant and persistent increases from the baseline of undrifted

values observed during the training stage, serving as indicators of environment drift. To further enhance this approach, we incorporate change point detection algorithms designed for monitoring real-valued sequences, such as Page-Hinkley, ADWIN, and KSWIN, to analyze these drift score sequences and identify points where significant shifts occur. This integration enables nuanced and dynamic detection of drift by capturing trends and patterns over time while leveraging the strengths of statistical change point detection methods.

5 Experiments

This section evaluates the effectiveness of our proposed environment drift detection approach. First, the experimental setup, including the RL environments and policies used, is described in detail. Next, we describe the methods used to simulate environment drifts. We then discuss the baseline methods selected for comparison and the performance metrics used for evaluation. Following this, we present the

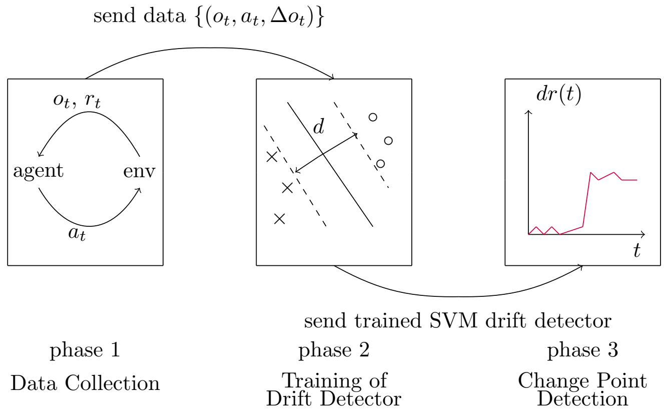

Figure 1. Overview of the three-phase framework for detecting environment drift in reinforcement learning. **Phase 1** collects undrifted transitions and generates synthetic drifted transitions. **Phase 2** trains and optimizes the SVM drift detector by maximizing the margin d between undrifted examples ($\circ$) and synthetic drifted examples ($\times$). **Phase 3** applies change point detection to the sequence of drift scores output by the SVM drift detector. Here, o_t denotes the observation at time step t, a_t the action at time step t, $\Delta o_t = o_{t+1} - o_t$ the change in observation, and $dr(t)$ the drift score at time t.

results of the experiments and conclude with an ablation study to analyze (1) the effect of synthetic drifted examples and (2) the effect of different change point detection algorithms, namely Page-Hinkley, ADWIN, and KSWIN, on false alarms and delay of environment drift detection. To enable the applicability of our study, the complete code and implementation details for reproducing the experiments are available in a Zenodo repository [10].

5.1 Experimental Setup

We conducted our experiments in five different RL environments, each with specific RL policies, to evaluate the performance of our method in detecting environment drift.

5.1.1 Environments & Policies

The following RL environments and policies are used in our experiments. Each environment presents unique dynamics and challenges, requiring tailored policies to achieve optimal control.

CartPole with DQN The Cart Pole environment features a pole mounted on a cart, with the pole attached by an unactuated joint. The cart moves along a frictionless track, and the goal is to keep the pole balanced in an upright position by applying forces that move the cart left or right. In this environment, we utilized a Deep Q-learning (DQN) [20] policy to control the cart.

LunarLander with PPO The LunarLander environment involves guiding a spacecraft to safely land on a designated landing pad. The lander is equipped with main and side engines, which must be controlled to achieve a soft landing while avoiding obstacles and minimizing fuel consumption. A Proximal Policy Optimization (PPO) [25] policy is applied to navigate the lander.

Mujoco Hopper with PPO The Hopper environment, simulated in MuJoCo, features a two-dimensional one-legged robot that must learn to hop forward as quickly and stably as possible. The robot's balance and propulsion are controlled using a PPO policy, which is trained to maximize forward velocity while minimizing instability.

Mujoco Halfcheetah with SAC The HalfCheetah environment simulates a two-dimensional robot with a cheetah-like body, where the goal is to achieve fast and stable forward motion. A Soft Actor-Critic (SAC) [16] policy is employed to train the agent, leveraging both exploration and exploitation to optimize performance in this challenging continuous control task.

Mujoco Humanoid with SAC The Humanoid environment involves a high-dimensional humanoid robot tasked with maintaining balance and walking forward efficiently. The SAC policy is trained to manage complex dynamics and degrees of freedom, enabling the robot to navigate the environment while minimizing energy usage and maintaining stability.

5.1.2 Simulations of Environment Drift

According to Yang et al. [31], there are two types of environment drifts: semantic drifts and noisy observations. Semantic drifts refer to changes in the expected outcome of the environment given the current observation and action, denoted as $\mu_{o_{t+1}|o_t,a_t}$. These can be simulated by modifying the environment parameters or applying uniform noise to actions. Noisy observations relate to changes in the uncertainty of the observations, represented as $\sigma^2_{o_{t+1}|o_t,a_t}$. This type of drift corresponds to sensor noise and can be simulated by adding small Gaussian noise to the observations o_{t+1}. Our study focuses on minor drifts, where the changes in episodic rewards are limited to less than 5%. It makes more sense to detect minor drifts than severe drifts, as in the case of severe drifts, rewards can often be directly used as a signal for environment drifts.

To simulate semantic drift, we introduced targeted modifications to each environment:

1. **CartPole**: We increased the force magnitude from 10.0 to 11.5.
2. **LunarLander**: A wind effect with a power of 0.7 was enabled.
3. **Hopper**: The action vector was multiplied by a uniform noise vector sampled from $U(0.8, 1)$.
4. **HalfCheetah**: The action vector was multiplied by a uniform noise vector sampled from $U(0.95, 1)$.
5. **Humanoid**: The action vector was multiplied by a uniform noise vector sampled from $U(1, 1.2)$.

For noisy observation simulations, Gaussian noise with $\sigma = 0.005$ was added to the observations in each environment to emulate sensory anomalies. The effects of the drift simulations on episodic rewards are presented in Table 1.

Env	Semantic Drift	Noisy Observation
CartPole	0.03%	-0.01%
LunarLander	-3.72%	-0.52%
Hopper	-1.10%	-0.01%
HalfCheetah	-3.10%	-1.50%
Humanoid	-4.40%	-1.20%

Table 1. Impact of environment drifts on the percentage change in average episodic rewards across different environments. For the CartPole environment, the step rewards were modified to be the cosine of the pole angle, as the original step reward (returning 1 for each step) lacked sufficient information on pole balancing performance.

To train our method with synthetic examples, each training environment is run for 20000 steps to collect both undrifted examples (labeled as 1) and synthetic drifted examples (labeled as -1). This data collection process is consistent across all environments and is used not only to train our method but also to prepare datasets for the baseline methods. Of the collected data, 70% is allocated for training, while the remaining 30% is reserved for validation. The validation set is used for hyperparameter tuning and to select the best-performing model during the training process, ensuring robust performance across different scenarios. An SVM with a radial basis function (RBF) kernel is used, and Optuna [1] is utilized to tune the hyperparameters C and γ. The optimization process maximizes the difference between the decision function values for undrifted and synthetic drifted examples, ensuring that the SVM effectively distinguishes between the two classes.

5.2 Baselines

To evaluate the performance of our environment drift detection method (**EDSVM**), we compare it against the following state-of-the-art baselines. These include two prediction error-based methods: **PEDM** [17] and **RIQN** [9]. Additionally, we include two commonly used unsupervised anomaly detection techniques: Local Outlier Factor (**LOF**) [6] and Isolated Forest (**IForest**) [19]. This comprehensive evaluation highlights the effectiveness of our approach in detecting a wide range of environment drifts.

- **PEDM** [17]: This baseline consists of a probabilistic ensemble of neural networks that predict o_{t+1} based on o_t and a_t. The mean squared error of this model serves as an indicator of environment drift.
- **RIQN** [9]: This baseline applies recurrent implicit quantile network (RIQN) to predict o_{t+1} given o_t. The absolute value of the prediction error is used as an indicator of environment drift.
- **LOF** [6]: This baseline leverages the Local Outlier Factor algorithm, trained on undrifted examples represented as $(o_t, a_t, \Delta o_t)$. During training, Optuna is employed to optimize the decision function values on undrifted validation examples. The negative decision function value serves as the drift indicator, signaling potential environment changes.
- **IFO** [19]: This baseline trains the Isolated Forest algorithm with undrifted examples represented as $(o_t, a_t, \Delta o_t)$. During training, Optuna is applied to optimize the decision function values on undrifted validation examples. The negative decision function value is used as an environment drift indicator.

5.3 Evaluation Metrics

Each experiment is run for 6,000 steps to simulate the production environment, with environment drift injected at the 3,000th step. To assess the effectiveness of our method and baselines, we evaluate the Area Under the Curve (AUC) over the entire 6,000 steps of the drift indicator. AUC is particularly suited for this evaluation because it is independent of thresholding, allowing for a consistent and fair comparison across different methods, often using varying thresholding techniques to process the drift indicator.

To ensure robust evaluation, we train three models for each method and run the experiment on each environment ten times per model, resulting in a total of 30 records for each method. This setup allows us to capture variability and assess the consistency of each method's performance across multiple runs.

For our method, we further assess its performance using additional metrics: delay and false alarms. These metrics are evaluated under different change point detection algorithms, including Page-Hinkley, ADWIN, and KSWIN, which will be discussed in Section 5.5.

5.4 Results

In this section, we present the results of our experiments across the five RL environments, evaluating the effectiveness of our proposed method, EDSVM, and comparing it with the baseline approaches. We analyze the performance of each method based on the AUC metric, which indicates the model's ability to distinguish between undrifted and drifted states. The results of semantic drifts and noisy observations are summarized in Table 2 and Table 3, respectively.

Env	EDSVM	PEDM	RIQN	LOF	IFO
CartPole	0.83	**1.00**	0.59	0.51	0.51
LunarLander	**0.61**	0.55	0.60	0.53	0.52
Hopper	**0.57**	0.54	0.49	0.52	0.49
HalfCheetah	**0.74**	0.56	0.50	0.49	0.50
Humanoid	**0.69**	0.62	0.53	0.50	0.50

Table 2. Mean AUC values for detecting semantic drifts in different environments using various methods.

Env	EDSVM	PEDM	RIQN	LOF	IFO
CartPole	0.81	**0.85**	0.59	0.51	0.51
LunarLander	**0.99**	0.67	0.55	0.51	0.50
Hopper	**0.99**	0.84	0.52	0.50	0.50
HalfCheetah	**0.59**	0.55	0.54	0.50	0.50
Humanoid	**0.99**	0.50	0.50	0.51	0.50

Table 3. Mean AUC values for detecting noisy observation drifts in different environments using various methods.

The results in Tables 2 and 3 highlight the consistent and robust performance of EDSVM in detecting both semantic drifts and noisy observation drifts across various environments. EDSVM achieves the highest AUC values in four out of five environments for each type of drift, including complex domains such as HalfCheetah, Hopper, and Humanoid. Even in the few cases where another method (e.g., PEDM) slightly outperforms it, EDSVM remains highly competitive.

5.5 Ablation Study

In this section, we present an ablation study to investigate the impact of synthetic drift examples and evaluate the performance of the three change point detection algorithms, Page-Hinkley, ADWIN, and KSWIN, in terms of detection delay and false alarm rates. This analysis provides valuable insights into the role of synthetic examples in improving drift detection and evaluates the practical effectiveness of these algorithms in conjunction with our proposed method.

5.5.1 *Effects of Synthetic Drifted Data*

To analyze the effects of synthetic drifted examples, we compare the AUC of EDSVM with a One-Class SVM (OSVM) trained exclusively on undrifted examples. Optuna is employed to optimize the hyperparameters ν and γ to maximize the decision function values on undrifted validation examples. This approach highlights the contribution of synthetic drifted examples in enhancing the drift detection performance of EDSVM. The results of semantic drifts and noisy observation drifts are listed in Table 4 and Table 5, respectively.

	CP	LL	HO	HC	HN
OSVM	0.53	0.57	0.45	0.42	0.60
EDSVM	0.83	0.61	0.57	0.74	0.69

Table 4. Mean AUC values of One-Class SVM (OSVM) and EDSVM for detecting semantic drifts in different environments. Abbreviations: CP = CartPole, LL = LunarLander, HO = Hopper, HC = HalfCheetah, HN = Humanoid.

	CP	LL	HO	HC	HN
OSVM	0.53	0.82	0.82	0.49	0.85
EDSVM	0.81	0.99	0.99	0.59	0.99

Table 5. Mean AUC values of One-Class SVM (OSVM) and EDSVM for detecting noisy observation drifts in different environments. Abbreviations: CP = CartPole, LL = LunarLander, HO = Hopper, HC = HalfCheetah, HN = Humanoid.

The results in Tables 4 and 5 highlight the superior performance of EDSVM compared to One-Class SVM (OSVM) in detecting both semantic and noisy observation drifts across all environments. These findings demonstrate the effectiveness of the generated synthetic drifted examples based on our method.

5.5.2 *Evaluation of Delay and False Alarms across Change Point Detection Algorithms*

In this section, we evaluate the performance of the different change point detection algorithms, Page-Hinkley, ADWIN, and KSWIN, regarding delay and false alarms when integrated with our method. Delay measures the time it takes for the algorithm to detect a drift after it occurs, while false alarms quantify the number of incorrect drift detections during undrifted periods. These metrics provide a comprehensive assessment of the practical applicability of each algorithm in real-world scenarios, where timely and accurate drift detection is crucial. The results of semantic drifts and noisy observation drifts are listed in Table 6 and Table 7, respectively. To ensure consistency, the anomaly scores from the production environment (i.e., negative decision function values) are normalized using the mean and standard deviation of the anomaly scores from the training environment. We use the default settings of each change point detection algorithm. Specifically, for Page-Hinkley, we set $\delta = 0.005$, $\alpha = 0.9999$, and threshold to 50.0. For ADWIN, we use $\delta = 0.002$. For KSWIN, α is set to 0.005, and window size is set to 100.

		CP	LL	HO	HC	HN
PH	D	2.43	105.57	2.33	53.9	606.13
	F	2.8	4.5	1.73	1.97	3.43
AD	D	567.13	867.57	151	832.5	1510.67
	F	1.13	1.3	0.3	0.3	0.83
KS	D	2633.4	2857.5	2935.5	2363.03	2746.83
	F	9.93	11.63	25.5	8.67	8.53

Table 6. Mean delay (D) and false alarms (F) for detecting semantic drifts of change point detection algorithms in different environments. The units of delay (D) and false alarms (F) are the number of steps, where smaller values indicate better performance. Abbreviations: PH = Page-Hinkley, AD = ADWIN, KS = KSWIN, CP = CartPole, LL = LunarLander, HO = Hopper, HC = HalfCheetah, HN = Humanoid.

		CP	LL	HO	HC	HN
PH	D	9.7	1.17	1.17	99.67	8
	F	2.63	3.9	2.23	1.73	4.53
AD	D	610.86	39	35.8	1693.2	45.4
	F	1.3	1.23	0.33	0.5	1.3
KS	D	2466.2	2697.3	2880.6	2449.7	2761.27
	F	10.63	11.3	24.6	9.03	9.33

Table 7. Mean delay (D) and false alarms (F) for detecting noisy observation drifts of change point detection algorithms in different environments. The units of delay (D) and false alarms (F) are the number of steps, where smaller values indicate better performance. Abbreviations: PH = Page-Hinkley, AD = ADWIN, KS = KSWIN, CP = CartPole, LL = LunarLander, HO = Hopper, HC = HalfCheetah, HN = Humanoid.

According to Table 6 and Table 7, Page-Hinkley has the lowest average delay in detecting both semantic and noisy observation drifts and also demonstrates a good balance between delay and false alarms. ADWIN achieves the lowest false alarms in detecting both semantic and noisy observation drifts, making it ideal for scenarios where accuracy is more critical than speed. However, it suffers from higher delays. Compared with Page-Hinkley and ADWIN, KSWIN consistently performs the worst, with the highest delays and false alarms across all environments and drift types. Therefore, it is not recommended for either semantic or noisy observation drift detection.

6 Discussion

The results presented in Section 5.4 demonstrate the effectiveness of our proposed method in detecting environment drifts across various RL environments.

Comparison of Performance EDSVM demonstrated consistently strong performance in terms of AUC, ranking among the top methods across all environments for both semantic and noisy observation drifts. It achieved the highest AUC values in most settings, especially in complex environments such as HalfCheetah, Hopper, and Humanoid, underscoring its robustness as a general-purpose drift detection approach. In contrast, unsupervised baselines such as LOF and IFO performed noticeably worse, suggesting that generic anomaly detection methods may not be well-suited for capturing subtle environment changes in RL.

Effectiveness of Synthetic Drifted Examples The results in Tables 4 and 5 highlight the effectiveness of synthetic drifted examples in improving drift detection. EDSVM consistently outperformed OSVM in both semantic and noisy observation drifts across all environments, with notable AUC gains in complex scenarios such as

HalfCheetah and Humanoid. These findings demonstrate that synthetic drifted examples enhance the model's ability to generalize and detect subtle environment changes, making EDSVM a robust choice for diverse drift types.

Delay & False Alarms under Different Change Point Detection Algorithms The results (Tables 6 and 7) show that Page-Hinkley achieved the lowest delays, making it ideal for real-time drift detection, though with slightly higher false alarms than ADWIN. ADWIN minimized false alarms, offering greater precision at the cost of higher delays. KSWIN performed the worst, with the highest delays and false alarm rates, making it unsuitable for effective drift detection.

Future Directions A key future direction is integrating EDSVM into CI/CD pipelines (commonly employed in MLOps) to enable automated monitoring, drift detection, and adaptive policy updates in production environments. This would streamline the process of maintaining reinforcement learning systems by ensuring the timely detection of environment drifts and triggering appropriate responses.

Another promising avenue is to investigate the effectiveness of synthetic drifted examples for training other supervised machine learning models beyond SVMs. Exploring models such as neural networks, decision forests, or ensemble methods could reveal alternative approaches that either complement or outperform EDSVM under specific conditions, potentially improving the flexibility and generalizability of environment drift detection frameworks.

Additionally, we aim to extend this framework for multi-agent RL systems and evaluate its performance under more complex and realistic drift scenarios.

7 Threats to Validity

This section discusses the threats to validity of our study.

Internal Validity The generation of synthetic drift may not fully capture the complexity of real-world drifts. Furthermore, the performance of EDSVM depends on careful hyperparameter optimization, and suboptimal configurations can affect the results. The evaluation is limited to the evaluated change point detection, which may overlook more powerful alternatives. To mitigate these risks, we have introduced different types of drifts, compared our method to a broad range of baselines, and used three state-of-the-art change point detection algorithms (Page-Hinkley, ADWIN, and KSWIN).

External Validity The experiments were conducted in a limited number of RL environments that may not fully represent the diversity of real-world applications. Furthermore, the method assumes that environment drifts are primarily caused by changes in transition probability distributions, which may not fully cover all aspects of drifts. Although we evaluated our work in five environments with different policies, two types of drifts, and different complexity, an evaluation in more real-world cases would further improve the external validity.

Construct Validity The use of synthetic drift examples for training and evaluation may not fully reflect the behavior of the method when detecting real drifts, which may exhibit more complex patterns. Furthermore, while AUC, false alarms, and detection delays are standard metrics, they may not capture all practical performance aspects, such as the cost of false alarms or delayed detection. Future work could validate the method using real-world drift data and explore alternative evaluation metrics that are better aligned with practical requirements.

8 Conclusion

This paper presented a novel approach for detecting environment drift in reinforcement learning systems using Support Vector Machines trained on synthetic drifted examples. Our methodology effectively addresses the challenges of detecting both semantic and noisy observation drifts without relying solely on real drifted data, which can be costly or difficult to obtain. By incorporating synthetic drifted examples, our method, EDSVM, achieves robust drift detection across diverse RL environments.

Through extensive experiments, we demonstrated that EDSVM performs competitively with state-of-the-art baselines, including PEDM and RIQN, in terms of AUC across various drift scenarios. The ablation study further explored the impact of synthetic drifted examples, revealing their critical role in enhancing detection performance. We also evaluated the change point detection algorithms Page-Hinkley, ADWIN, and KSWIN in the context of analyzing delay and false alarms. These insights underscore the practicality and versatility of our approach in real-world RL applications.

Looking forward, integrating EDSVM into CI/CD/CT pipelines offers an exciting opportunity to enable automated monitoring, drift detection, and policy adaptation in production environments. Additionally, future works could explore the effectiveness of synthetic drifted examples for other supervised learning models for environment drift detection, and extend this framework to multi-agent RL systems.

Acknowledgements

This work was supported by the Austrian Research Promotion Agency (FFG) through the project MODIS (No. FO999895431).

References

[1] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[2] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno. Early drift detection method. In *Fourth international workshop on knowledge discovery from data streams*, volume 6, pages 77–86. Citeseer, 2006.

[3] L. Baier, T. Schlör, J. Schöffer, and N. Kühl. Detecting concept drift with neural network model uncertainty. *arXiv preprint arXiv:2107.01873*, 2021.

[4] F. Bayram, B. S. Ahmed, and A. Kassler. From concept drift to model degradation: An overview on performance-aware drift detectors. *Knowledge-Based Systems*, 245:108632, 2022.

[5] A. Bifet and R. Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 443–448. SIAM, 2007.

[6] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[7] Y. Chen, Z. Chai, Y. Cheng, and H. Rangwala. Asynchronous federated learning for sensor data with concept drift. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 4822–4831. IEEE, 2021.

[8] C. Cortes. Support-vector networks. *Machine Learning*, 1995.

[9] M. H. Danesh and A. Fern. Out-of-distribution dynamics detection: Rl-relevant benchmarks and results. *arXiv preprint arXiv:2107.04982*, 2021.

[10] Z. Fang. Code and data for "edsvm: A novel approach for environment drift detection in reinforcement learning using synthetic drifted examples". Zenodo, 2025. Available at https://doi.org/10.5281/zenodo.16947486.

[11] Z. Fang and U. Zdun. Detecting environment drift in reinforcement learning using a gaussian process. In *Proceedings of the IEEE 36th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2024.

[12] I. Frias-Blanco, J. del Campo-Ávila, G. Ramos-Jimenez, R. Morales-Bueno, A. Ortiz-Diaz, and Y. Caballero-Mota. Online and non-parametric drift detection methods based on hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):810–823, 2014.

[13] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. In *Advances in Artificial Intelligence–SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29-Octotber 1, 2004. Proceedings 17*, pages 286–295. Springer, 2004.

[14] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.

[15] I. Greenberg and S. Mannor. Detecting rewards deterioration in episodic reinforcement learning. In *International Conference on Machine Learning*, pages 3842–3853. PMLR, 2021.

[16] 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*, pages 1861–1870. PMLR, 2018.

[17] T. Haider, K. Roscher, F. Schmoeller da Roza, and S. Günnemann. Out-of-distribution detection for reinforcement learning agents with probabilistic dynamics models. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pages 851–859, 2023.

[18] P. Li, J. Thomas, X. Wang, A. Khalil, A. Ahmad, R. Inacio, S. Kapoor, A. Parekh, A. Doufexi, A. Shojaeifard, et al. Rlops: Development life-cycle of reinforcement learning aided open ran. *IEEE Access*, 10: 113808–113826, 2022.

[19] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.

[20] V. Mnih. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[21] R. Müller, S. Illium, T. Phan, T. Haider, and C. Linnhoff-Popien. Towards anomaly detection in reinforcement learning. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 1799–1803, 2022.

[22] E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

[23] A. Paleyes, R.-G. Urma, and N. D. Lawrence. Challenges in deploying machine learning: a survey of case studies. *ACM computing surveys*, 55 (6):1–29, 2022.

[24] C. Raab, M. Heusinger, and F.-M. Schleif. Reactive soft prototype computing for concept drift streams. *Neurocomputing*, 416:340–351, 2020.

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

[26] R. Sutton and A. Barto. Reinforcement learning: An introduction. mit press. *Google Scholar*, pages 329–331, 2018.

[27] C. Wang, S. Erfani, T. Alpcan, and C. Leckie. Oil-ad: An anomaly detection framework for sequential decision sequences. *arXiv preprint arXiv:2402.04567*, 2024.

[28] Z. Wang, C. Chen, H.-X. Li, D. Dong, and T.-J. Tarn. Incremental reinforcement learning with prioritized sweeping for dynamic environments. *IEEE/ASME Transactions on Mechatronics*, 24(2):621–632, 2019.

[29] S. Watanabe. Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance. *arXiv preprint arXiv:2304.11127*, 2023.

[30] M. Yan. Accurate detecting concept drift in evolving data streams. ict express 6 (4): 332–338, 2020.

[31] J. Yang, K. Zhou, Y. Li, and Z. Liu. Generalized out-of-distribution detection: A survey. *International Journal of Computer Vision*, 132 (12):5635–5662, 2024.