Rethinking Out-of-Distribution Detection for Reinforcement Learning: Advancing Methods for Evaluation and Detection

Linas Nasvytis
Harvard University
Cambridge, MA, United States
linasnasvytis@fas.harvard.edu

Kai Sandbrink
University of Oxford
Oxford, United Kingdom
kai.sandbrink@lmh.ox.ac.uk

Jakob Foerster
University of Oxford
Oxford, United Kingdom
jakob.foerster@engs.ox.ac.uk

Tim Franzmeyer†
University of Oxford
Oxford, United Kingdom
frtim@robots.ox.ac.uk

Christian Schroeder de Witt†
University of Oxford
Oxford, United Kingdom
cs@robots.ox.ac.uk

ABSTRACT

While reinforcement learning (RL) algorithms have been successfully applied across numerous sequential decision-making problems, their generalization to unforeseen testing environments remains a significant concern. In this paper, we study the problem of out-of-distribution (OOD) detection in RL, which focuses on identifying situations at test time that RL agents have not encountered in their training environments. We first propose a clarification of terminology for OOD detection in RL, which aligns it with the literature from other machine learning domains. We then present new benchmark scenarios for OOD detection, which introduce anomalies with temporal autocorrelation into different components of the agent-environment loop. We argue that such scenarios have been understudied in the current literature, despite their relevance to real-world situations. Confirming our theoretical predictions, our experimental results suggest that state-of-the-art OOD detectors are not able to identify such anomalies. To address this problem, we propose a novel method for OOD detection, which we call DEXTER (Detection via Extraction of Time Series Representations). By treating environment observations as time series data, DEXTER extracts salient time series features, and then leverages an ensemble of isolation forest algorithms to detect anomalies. We find that DEXTER can reliably identify anomalies across benchmark scenarios, exhibiting superior performance compared to both state-of-the-art OOD detectors and high-dimensional changepoint detectors adopted from statistics.

KEYWORDS

Reinforcement Learning; Out-of-Distribution Detection; Anomaly Detection; Robust RL; AI Safety

ACM Reference Format:
Linas Nasvytis, Kai Sandbrink, Jakob Foerster, Tim Franzmeyer†, and Christian Schroeder de Witt†. 2024. Rethinking Out-of-Distribution Detection for Reinforcement Learning: Advancing Methods for Evaluation and Detection. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

1 INTRODUCTION

Deep reinforcement learning (RL) algorithms [15, 22, 27] have been applied to numerous sequential decision-making problems, ranging from robotics [1, 31] and nuclear fusion [11] to solar geoengineering [10]. However, their low reliability in situations that are not well represented in the training environment hinders the deployment of RL agents in safety-critical scenarios [6, 25]. Such discrepant deployment situations are often referred to as out-of-distribution scenarios. The task of identifying whether a given environment differs from the train-time environment is hence referred to as out-of-distribution (OOD) detection [16]. OOD detection constitutes an important desiderata for the deployment of RL agents in the real world, as reliable OOD detection would increase the safety of deployed AI agents by allowing contingency actions to be taken in unfamiliar or dangerous situations. Examples of such contingency options include the automatic parking of an autonomous car on the side of the road, or raising security escalations in cyber systems.

The OOD detection framework assumes that the agent has access to data from the training process, which must be used to develop a mechanism for detecting OOD scenarios in the unknown deployment environment within a minimal amount of interactions. Such OOD deployment environments are often simulated by adding anomalies (e.g. sensor or process noise) to the train-time environments [9, 16, 23]. In prior work, benchmark environments consider either injecting independent and identically distributed (i.i.d.) anomalies, such as Gaussian noise, or time-independent anomalies, such as changing gravity [9, 16, 23]. We argue that these scenarios fail to capture the *temporally dependent* nature of many real-world anomalies. For example, in the case of an autonomous robot, any dust or smudge appearing on its camera lens would lead to a series of readings that contain systematic errors, instead of causing random misreadings of environmental data. Further, previous works [9, 16] employ decision rules based on per-step anomaly scores. While such methods may prove empirically effective in some cases, we propose an information-theoretically grounded approach based on sequential hypothesis testing [32], which results in a decision rule based on accumulated anomaly scores over multiple timesteps.

† Equal supervision

We start by introducing novel benchmark environments, which include temporally dependent anomalies, and find that state-of-the-art OOD detection methods fail in such scenarios. To detect such anomalies, we introduce a novel detection mechanism called **D**etection via **Ex**traction of **T**ime S**e**ries **R**epresentations (DEXTER). This mechanism first extracts time-series features, which are then used with a random-forest algorithm to compute anomaly scores. We leverage information-theoretically optimal sequential hypothesis testing techniques to derive a cumulative sum (CUSUM) detector using the full history of DEXTER's anomaly scores, which we refer to as DEXTER+C. Lastly, we evaluate DEXTER and DEXTER+C on a range of novel and standard benchmark environments and compare their performance to relevant baselines. We find that DEXTER significantly outperforms state-of-the-art detectors across various metrics, including Area under the Receiver Operator Characteristic (AUROC) scores. Importantly, DEXTER+C significantly decreases the number of timesteps needed to detect OOD scenarios.

Our work makes the following contributions:

- We propose a clarification of the terminology of OOD detection in reinforcement learning.
- We introduce new testing scenarios for OOD detection in reinforcement learning, which consider a broad class of disturbances focused on temporally-correlated noise.
- We propose a new detector, which we refer to as DEXTER (**D**etection via **Ex**traction of **T**ime S**e**ries **R**epresentations), as well as a new decision rule, DEXTER+C, and show that these outperform state-of-the-art methods across relevant scenarios.

2 RELATED WORK

2.0.1 Algorithms for OOD Detection. The research on out-of-distribution (OOD) detection in reinforcement learning [16, 24] is more limited compared to supervised and unsupervised learning. However, interest in the field has grown more recently. To our knowledge, Sedlmeier et al. [29] outline the first practical method for OOD detection in reinforcement learning. The authors use epistemic uncertainty of the agent's actions to quantify the anomaly scores of different states, reasoning that epistemic uncertainty tends to be higher in areas of low data density. Mohammed and Valdenegro-Toro [23] introduce a benchmark to evaluate generalized OOD detection methods in reinforcement learning on three environments. In Cartpole and Pendulum, gravity is varied, while in Pong, noise is added to state observations. Danesh and Fern [9] propose a more extensive benchmark to test OOD detection, implementing different types of observational noise across seven classic RL environments. Additionally, they introduce a new OOD detector called the Recurrent Implicit Quantile Network (RIQN). At each time step, RIQN uses the current and prior states in the environment, $s_{1:t}$, to generate auto-regressive predictions for the next δ states, $s_{t+1:t+\delta}$, then computes the difference between its predictions and the realized environment states, and uses this difference as the anomaly score for a given transition (s_t, s_{t+1}). The authors demonstrate that RIQN outperforms several baseline detectors across several of the proposed anomalous environments.

The state-of-the-art OOD detection method, Probabilistic Ensemble Dynamics Model (PEDM), was proposed by Haider et al. [16], and consists of two components. First, a 1-step *forward dynamics model* f_θ is learned, modeling the transition dynamics of the training environment, implemented as a Probabilistic Deep Neural Network Ensemble model [18]. For a given state and action pair (s_t, a_t), this model predicts the next state in the environment as $s'_{t+1} = f_\theta(s_t, a_t)$. Second, an *anomaly score generator* is applied, which compares the predictions of the world dynamics model to outcomes in the test-time deployment environment, and generates an anomaly score for each transition.

To measure performance in detecting anomalies applied to the observation space, Haider et al. [16] use the environments from Danesh and Fern [9]. For the detection of anomalies applied to the transition dynamics of the RL environment, the authors propose four additional benchmark environments. They modify the classical Cartpole, HalfCheetah, Pusher, and Reacher environments by adding semantic anomalies, such as changing the gravity or multiplying the velocity applied to all body parts of the agent by a constant factor. The performance of PEDM is measured against the RIQN algorithm [9], as well as several other benchmark models. For each algorithm, the performance is measured according to AUROC. Based on the results, PEDM outperforms the other detection algorithms across almost all newly-proposed environments with anomalous transition dynamics. Moreover, within the environments from Danesh and Fern [9], PEDM is tied with the RIQN model for the best performance (with RIQN exhibiting significantly worse performance on the new benchmarks).

2.0.2 Limitations of current OOD detection approaches. We argue that current approaches to OOD detection suffer from three weaknesses. First, most of the literature of OOD detection in RL makes the simplifying assumption of injecting independent and identically distributed (i.i.d.) noise across different timesteps in the environment [9, 16, 23], which can hence be reliably detected with one-step detection approaches. However, relevant real-world scenarios will likely feature more complex disturbances, which have temporally correlated anomalies. Second, these approaches rely on anomaly detection via prediction error, computing anomaly scores based on the prediction error from a forward-dynamics model trained on training samples. Such prediction models are unlikely to detect temporally-correlated anomalies which may only have a minor effect on the transition dynamics. Third, the decision rules for online out-of-distribution detection rely only on the individual anomaly scores observed at each time step, instead of taking into account the full history of anomaly scores. While such approaches may prove empirically effective, they neglect information that would be used in information-theoretically grounded sequential hypothesis testing [32].

2.0.3 Sequential Hypothesis Testing. Beyond machine learning, the task of online detection of whether a sample of time series data differs from a predefined distribution has also been analyzed in statistics, primarily in the field of changepoint detection (CPD). Sequential hypothesis testing, as pioneered by Wald in his sequential probability ratio test (SPRT) [32], aims to discern between two hypotheses in an online manner, using as few samples as possible. CUSUM methods, a popular technique in CPD, are designed to detect shifts in the mean or variance of a process [26]. CUSUM

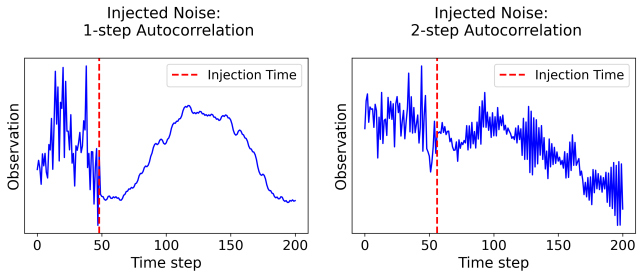

Figure 1: Illustration of temporally autocorrelated anomalies. Left: at injection time (t = 48), noise applied to the observation changes from no correlation to 1-step autocorrelation. Right: at injection time (t = 56), noise changes from no correlation to 2-step autocorrelation.

methods focus on capturing cumulative information over time, providing a mechanism to identify time-dependent anomalies.

3 BACKGROUND AND NOTATION

A Markov Decision Process (MDP) is defined by a tuple (S, A, P, R). Here, S is a finite set of states, A is a finite set of actions, $P : S \times A \times S \to [0, 1]$ defines the state transition probabilities, and $R : S \times A \to \mathbb{R}$ is the reward function. The goal of the agent at each time step is to maximize the cumulative sum of discounted future rewards $\sum_t^{t\to\infty} \gamma^t R(s_t, a_t)$, where $\gamma \in [0, 1]$ is the discount rate.

In the general machine learning literature – which includes supervised learning, unsupervised learning, and reinforcement learning – there are several terms used to describe the task of detecting if a new sample of data differs from a specified training set. The terms can include out-of-distribution (OOD) detection, anomaly detection, novelty detection, and outlier detection. To clarify the terminology, Yang et al. [33] provide an in-depth literature review of this topic, focusing primarily on supervised learning, where each data point contains an input-label pair $(x,y) \in X \times Y$, where X is the input (sensory) space, and Y the label (semantic) space. A data distribution is defined as a joint distribution $P(X, Y)$ over $X \times Y$. A distribution shift can occur in either the marginal distribution $P(X)$, or both $P(Y)$ and $P(X)$. As noted by Yang et al. [33], a shift in $P(Y)$ naturally triggers shift in $P(X)$. Denote the distributions of normal and anomalous data points by P and P', respectively. **Covariate shift (sensory) anomalies** occur when the inputs of normal data points are drawn from an in-distribution $P(X)$, whereas inputs of anomalies are drawn from out-of-distribution $P'(X)$, such that $P(X) \neq P'(X)$. However, no distribution shift occurs in the label space: $P(Y) = P'(Y)$. **Semantic shift anomalies** occur when a distributional shift occurs in the label space, such that $P(Y) \neq P'(Y)$. Following this distinction, the authors define:

(1) **Sensory anomaly detection** as the task of detecting covariate shift anomalies, i.e. samples from $P'(X)$.
(2) **Semantic anomaly detection** as the task of detecting semantic shift anomalies, i.e. samples from $P'(Y)$.
(3) **Out-of-distribution (OOD) detection** as the sub-domain within semantic anomaly detection, where in-distribution samples are drawn from multiple classes (i.e. label space Y is not binary).
(4) **Generalized out-of-distribution detection** as the general task of detecting all anomalies, i.e. both sensory and semantic anomalies.

The main issue is that the training and testing data in reinforcement learning does not contain a label space Y. As a result, the terms *out-of-distribution detection* and *anomaly detection* are often used interchangeably [16], and some of their definitions seem to conflict with each other. A detailed overview of such terminological discussions on OOD detection in reinforcement learning can be found in the Appendix.

4 TERMINOLOGY OF OOD DETECTION IN REINFORCEMENT LEARNING

Given terminological discussions on how to label different types of anomalies for OOD detection in reinforcement learning, we propose terminology adapted from the framework introduced by Yang et al. [33], while incorporating insights from existing literature from Danesh and Fern [9] and Haider et al. [16]. We first differentiate between two kinds of anomalies based on their effects on the MDP: First, **sensory anomalies** change the observation that the agent receives (e.g. adding observational noise), while leaving the underlying environment dynamics unchanged. Hence, these are closely related to covariate shift anomalies defined in Yang et al. [33]. In contrast, **semantic anomalies** change the underlying environment dynamics (e.g. changing the gravity in the environment). Hence, these are very similar to semantic shift anomalies defined in Yang et al. [33]. We further define **Generalized Out-of-Distribution Detection** as the task of detecting either of the two.

Definition 1. ***Sensory anomaly detection*** *in reinforcement learning refers to the task of identifying sensory anomalies. A sensory anomaly is a perturbation to the reinforcement learning environment, which changes the observations O that the agent receives, but leaves the underlying environment dynamics unchanged. If the change in observation leads the environment to become partially observable, such an anomaly changes the underlying Markov Decision Process (MDP) to a Partially Observable Markov Decision Process (POMDP).*

Definition 2. ***Semantic anomaly detection*** *in reinforcement learning refers to the task of identifying semantic anomalies. Semantic anomalies are perturbations to the reinforcement learning environment, which change the transition function of a Markov Decision Process $P(S'|S, A)$ by changing the environment dynamics.*

Definition 3. ***Generalized out-of-distribution detection*** *in reinforcement learning refers to the task of identifying any type of anomaly in the environment, hence including sensory anomaly detection and semantic anomaly detection.*

The proposed terminology unites the task of detecting all types of anomalies in reinforcement learning environments under the single term of generalized out-of-distribution detection, while drawing a meaningful distinction between the two major types of anomalies that could exist in an environment. This is especially relevant as some recent works have been inaccurate about the changes to the MDP that the anomalies introduce (see Appendix for a discussion).

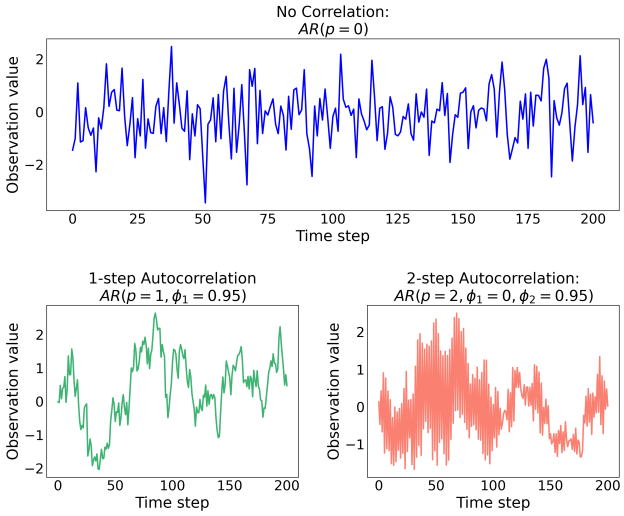

Figure 2: Illustrations of Autoregressive (AR) model with parameters for no correlation (top), 1-step autocorrelation (bottom-left), and 2-step autocorrelation (bottom-right), which is used to create three different types of noise in new testing scenarios.

5 NOVEL TESTING SCENARIOS FOR OOD DETECTION

We now propose new benchmark environments for generalized OOD detection that contain both sensory and semantic anomalies. In contrast to the benchmark environments used in previous works, these environments contain temporally correlated anomalies, which is achieved by generating and adding noise using the autoregressive (AR) model [3].

5.0.1 Autocorrelated noise patterns. To create custom benchmark environments with different types of noise correlations, we use an AR model of order p, denoted by AR(p), where the noise value Y_t of a series at any point in time is linearly dependent on its own p past values:

$$Y_t = \mu + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \varepsilon_t$$

Using this AR model, we implement noise with three different types of autocorrelations:

(1) **No correlation**, in which case the noise is not correlated across time, i.e., containing white noise, given as $Y_t = \epsilon_t$.
(2) **1-step correlation**, in which case the noise is autocorrelated across each step, hence given as $Y_t = \phi_1 Y_{t-1} + \epsilon_t$.
(3) **2-step correlation**, where the noise is autocorrelated only at every second step (i.e., step 2, step 4), in which case, $Y_t = \phi_2 Y_{t-2} + \epsilon_t$.

Examples of the three types of noise for correlation coefficients are displayed for $\phi = 0.95$ in Figure 2.

Based on the autocorrelated noise patterns introduced above, we introduce three new benchmark environments.

5.0.2 ARTS: Autoregressive Time Series environments. Autoregressive Time Series (ARTS) serves as our baseline environment. At the beginning of each episode, we generate a 1-dimensional vector using the AR process, with the number of elements equal to the maximum number of episode steps. At each time step t, the agent receives a *1-dimensional observation* o_t that contains the noise generated by the AR process, while the environment state is treated as constant.

5.0.3 ARNO: Autoregressive Noised Observation environments. In the Autoregressive Noised Observation (ARNO) setting, we introduce **sensory anomalies**. At the beginning of the episode, we generate a noise matrix, where each row of the matrix is a time series, independently drawn from the AR Process. The number of rows in the matrix corresponds to the dimension of a single observation in the environment.

For each step t, the agent takes an action a_t, which is passed to the environment's state transition function to generate the next state: $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$. *After* each state transition, the noise vector sliced from the matrix is added to the state s_{t+1}, transforming it into an observation o_{t+1}. This simulates a scenario where the environment's underlying state undergoes a distortion before being presented to the agent, similar to the effects of a camera glitch or a sensory failure.

5.0.4 ARNS: Autoregressive Noised State environments. In Autoregressive Noised State (ARNS) environments, we introduce **semantic anomalies**. Similarly to ARNO, we generate the AR noise matrix at the beginning of each episode, with the same shape. At each step t, the agent takes an action a_t, which is passed to the environment's state transition function. However, *before* the transition, a noise vector sliced from the matrix is applied to each dimension of the state vector, effectively changing the transition function. This simulates a scenario where, for example, the underlying physics or rules of the environment are unpredictably changed due to systematic issues.

6 DEXTER: DETECTION VIA EXTRACTION OF TIME SERIES REPRESENTATIONS

We now move on to discuss our proposed algorithm for out-of-distribution detection in reinforcement learning, which we refer to as **DEXTER** (Detection via Extraction of Time-series Representations). The model is composed of two components: first, a **feature extractor** $f(s_1, ...s_n)$ that extracts relevant time series features from a given time series data; and second, an **anomaly detector** $h(f(s_1, ...s_n)$, which we implement as an ensemble of isolation forest models that takes as an input the extracted features, and outputs an anomaly score for the time series.

6.0.1 Feature extractor f. Given a sample of states $s_0, s_1, \ldots s_t$, DEXTER first extracts the relevant time series features from each state dimension. The goal behind the proposed detector is to be anomaly agnostic: Instead of choosing features that could detect a specific anomaly (e.g. autocorrelations), we aim to extract a diversity of features to maximize the number of anomalies DEXTER could detect. For this purpose, we use the `tsfresh` feature extractor [8], which captures features of a diverse set of statistics of the time series, including fundamental descriptive statistics (e.g.

number of peaks, minimum, maximum, and median values), autocorrelation statistics (e.g. autocorrelation coefficients for k lags; partial autocorrelation function [2] at the given lag k), advanced features (e.g. descriptive statistics of the absolute Fourier transform spectrum [21] and approximate entropy [12]).

6.0.2 Isolation Forest Algorithm h. Next, we fit an ensemble of isolation forest algorithms, which predicts the probability of an anomaly based on the extracted features. The isolation forest algorithm is an ensemble-based unsupervised machine learning method designed for anomaly detection, introduced by Liu et al. [19]. Given a training set, the algorithm first constructs multiple random trees to partition data points. For each tree, the algorithm selects a sample of training data, and recursively partitions the sample of data points by randomly selecting a split value that falls between the minimum and maximum values of a given attribute (i.e. dimension). Once an ensemble of such isolation trees is constructed, a test data point is passed down each of these trees. The path length it takes to isolate the data point is averaged over the trees to produce an anomaly score for the data point. The algorithm relies on the assumption that anomalous points will be more easily separable from the rest of the sample [20].

Isolation forest has the following beneficial properties: First, it is an **unsupervised** method, which is essential as we are not provided anomalous data and hence must train an anomaly-agnostic detector. Second, it has **linear time complexity** [19], allowing to scale to high-dimensional data. Third, it allows for **anomaly scoring**, enabling a nuanced interpretation of how "anomalous" a data point is, instead of providing a mere binary classification.

6.0.3 Training and testing. An overview of DEXTER Anomaly Score computation is displayed in Algorithm 1. We will denote the training data as a sample $D = \{s_n, a_n, s_{n+1}\}_{n=1}^{N}$.

Training. First, DEXTER partitions D into windows of size W states. For each window w_i, the model extracts time-series features along each dimension d, $1 \leq d \leq m$, to obtain $f_{i,d}$. Once feature extraction for all windows is complete: for each dimension d, all extracted features are concatenated to make f_d. This aggregate feature set is used to fit the ensemble of isolation forest models, where each model is assigned to a different state dimension, resulting in m total models. For each model in the ensemble IF_d, the aim is to instill an understanding of the predominant patterns present in the given dimension of in-distribution observations.

Dexter Anomaly Score Computation. For a given timestep T, DEXTER collates the last W states to constitute a window, and extracts features f'_d along each dimension in m. The ensemble of trained isolation forest models is then utilized to compute anomaly scores for these features. The concluding anomaly score for a timestep is calculated as the arithmetic mean of the scores spanning all the dimensions in m.

Algorithm 1 DEXTER

Require: State dimensions m, window size W, policy π, dataset D of training transitions

Training
Initialize ensemble of Isolation Forests IF = $\{\text{IF}_1, ..., \text{IF}_m\}$
Partition D into windows $w_1, ..., w_N$ of size W
for each w_i in windows **do**
  for dimension d from 1 to m **do**
    Extract time series features $f_{i,d}$
  end for
end for
for dimension d from 1 to m **do**
  Form f_d by concatenating $f_{i,d}$
  Train IF_d using f_d
end for

DEXTER Anomaly Score Computation
for Time t from 0 to T **do**
  Action $a_t \leftarrow \pi(s_t)$, observe s_{t+1}
  Update window with s_{t+1}
  for dimension d from 1 to m **do**
    Extract features f_d
    Compute score a_d with IF_d
  end for
  Set A_T as average over all a_d
end for

6.1 Sequential Hypothesis Testing with DEXTER+C

The previously described DEXTER algorithm outputs an anomaly score after each transition in the test-time deployment environment. We now introduce DEXTER+C (short for DEXTER+CUSUM), which uses the information-theoretic hypothesis testing CUSUM algorithm [26] to derive a decision rule for when to classify a test-time deployment as OOD. DEXTER+C is detailed in Algorithm 2. First, the average DEXTER anomaly score $\bar{A}$ for a held-out set of training transitions is computed. Then, the CUSUM detection threshold τ is computed by evaluating CUSUM on held-out training transitions and choosing τ such that a targeted False Positive Rate (*FPR*) is achieved on the set of held-out transitions. This tunable targeted *FPR* indicates the ratio of anomaly-free episodes that are falsely classified as OOD. Note that both steps only require access to training transitions and that the only hyperparameter is the targeted *FPR*. At test time, an online CUSUM detector with detection threshold τ is employed. This detector updates the CUSUM score S_t based on the previous timestep's score and the current DEXTER anomaly score A_t. If the CUSUM score S_t exceeds τ, an OOD scenario is detected and the execution is halted.

7 EMPIRICAL EVALUATION

We first focus on the newly introduced benchmark scenarios ARNO, ARNS, and ARTS, which contain temporally correlated anomalies. We implement ARNO scenarios with three levels of noise (described in Section 7.1) in three different environments – Cartpole, Reacher, and Acrobot [4]. We implement ARNS scenarios with analogous levels of noise on Cartpole and Reacher, as the implementation on Acrobot leads to inconsistent agent policies, which are described in more detail in the Appendix. Lastly, as the ARTS environment does not contain a reward signal, we only implement a single noise level.

Algorithm 2 DEXTER+C

Require: targeted *FPR*, state dimensions m, window size W, policy π, validation dataset D of training transitions

Compute Mean Anomaly Score $\bar{A}$ using Validation Set
Compute DEXTER anomaly scores for $\frac{1}{2}D$, storing in $\{ A_t \}$
$\bar{A} \leftarrow$ mean of $\{ A_t \}$

Compute CUSUM Threshold τ using Validation Set
$S_{max\_list} \leftarrow []$
for each episode ep in $\frac{1}{2}D$ **do**
  $S_0 \leftarrow 0, S_{max} \leftarrow 0$
  for each score A_t in ep **do**
    $S \leftarrow S + A_t - \bar{A}$
    $S_{max} \leftarrow \max(S, S_{max})$
  end for
  Append S_{max} to S_{max_list}
end for
$\tau \leftarrow$ 1-*FPR* percentile of S_{max_list}

DEXTER+C Out-of-Distribution Detection
$S_0 \leftarrow 0$
for Time t from 0 to T **do**
  Compute A_t using DEXTER Anomaly Score Computation
  $S_t \leftarrow \max(0, S_{t-1} + A_t - \bar{A})$
  if $S_t > \tau$ **then**
    Raise out-of-distribution alert
    Break
  end if
end for

The codebase and supplementary materials are publicly available at: https://github.com/LinasNas/DEXTER.

We afterward consider the common benchmark scenarios introduced by Haider et al. [16], which contain either i.i.d anomalies or time-independent anomalies (five scenarios in total), two noise levels for each anomaly, and two different environments (Cartpole and Reacher), resulting in 20 different evaluation frameworks.

Previous works [9, 16] focus on AUROC computed for per-transition anomaly scores as the main metric for detector performance. However, AUROC scores do not yield a decision rule for classifying observation as OOD. We hence also consider the required timesteps to classify the anomaly-containing test-time environment as OOD as a metric for detector performance, we refer to this metric as *Detection Time*.

7.1 Evaluations for ARTS, ARNS and ARNO Scenarios

Noise levels in each environment. In ARNO and ARNS environments, we apply three different levels of noise to generate the anomalies. The levels are classified as Light, Medium, and Strong, based on their effect on the agent's reward in the environment. The procedure to choose these levels is as follows. First, we normalize the noise magnitude by the standard deviation of that dimension's observation. Then, we apply *uncorrelated noise* of different magnitudes to the environment and train a reinforcement learning agent using Proximal Policy Optimization (PPO) [28] in discrete action-space environments, and Twin Delayed Deep Deterministic (TD3)[14] algorithm for continuous action-space environments, until it converges to a stable cumulative episodic reward. We then measure how much, averaged over 50 episodes, the reward differs from the reward achieved when an agent's policy is optimized in an undisturbed environment. Lastly, in each environment, we identify three magnitudes of noise, which we classify as:

- Light noise: reduces $\sum_{t=0}^{T} r_t$ by $\sim 1\%$
- Medium noise: reduces $\sum_{t=0}^{T} r_t$ by $\sim 25\%$
- Strong noise: reduces $\sum_{t=0}^{T} r_t$ by $\sim 50\%$

Table 1: ARTS scenarios: Detector performance (AUC above, Detection time below).

		1-step	2-step
AUROC ↑	CPD: OCD	0.79	0.77
	CPD: Chan	0.64	0.64
	PEDM	0.51	0.5
	DEXTER	**0.89**	**0.83**
Det. Time ↓	PEDM+C	200.0	200.0
	DEXTER+C	**19.8**	**28.0**

7.1.1 Evaluation Procedure. Our experimental evaluation follows recent frameworks [9, 16]. We first apply uncorrelated noise of each of the three levels. Then for each noise level, we train a policy $\pi_A(a|s)$ until it is optimized for the episodic task with episode length $H \in \mathbb{N}$.

In each episode, we introduce an anomaly at a random time $t_a \in (t_0 + 5, t_H - 5)$, and apply this anomaly until the end of the episode. That anomaly changes the correlation structure of noise from *no correlation* to either *1-step* or *2-step* correlation, as it can be observed in Figure 1. We test each of the two cases separately. All transitions before the anomaly is applied, $[(s_{t_0}, s_{t_1}), \ldots, (s_{t_a-1}, s_{t_a})]$ are labeled as in-distribution, and all transitions after, $[(s_{t_a}, s_{t_{a+1}})$ $\ldots, (s_{t_{H-1}}, s_{t_H})]$ are labelled as anomalous.

To account for the effects of initialization and varying injection times, we repeat this procedure for multiple random point time points and different initial environment states. Therefore, in expectation, we obtain a balanced dataset.

Importantly, the noise level throughout the episode is constant, since after the injection time, we simply change the correlation structure of the noise, but not its magnitude. The experiments are implemented in such a way since we are specifically interested in whether the detectors can identify structural changes to the noise, rather than detect the sudden emergence of noise itself, unlike most prior benchmark environments [9, 16].

7.1.2 DEXTER and DEXTER+C. We implement DEXTER (described in Algorithm 1) with a window size of $W = 10$ timesteps, since it strikes a balance between a sample that is long enough to allow for meaningful time series feature extraction, while being short enough to allow for quick anomaly detection. We implement DEXTER+C (described in Algorithm 2) with a target *FPR* of 1%.

Table 2: ARNO scenarios: Detector performance (AUC above, Detection time below).

		Cartpole						Acrobot						Reacher					
		Light Noise		Medium Noise		Strong Noise		Light Noise		Medium Noise		Strong Noise		Light Noise		Medium Noise		Strong Noise	
	Detector	1-step	2-step	1-step	2-step	1-step	2-step	1-step	2-step	1-step	2-step	1-step	2-step	1-step	2-step	1-step	2-step	1-step	2-step
AUROC ↑	CPD: OCD	0.67	0.69	0.76	0.72	0.78	0.73	0.64	0.65	0.75	0.76	0.8	0.79	0.51	0.51	0.51	0.51	0.52	0.52
	CPD: Chan	0.69	0.68	0.72	0.75	0.75	0.73	0.62	0.59	0.76	0.71	0.86	0.77	0.51	0.51	0.52	0.52	0.53	0.53
	PEDM	0.55	0.62	0.6	0.51	0.6	0.55	0.57	0.54	0.52	0.54	0.5	0.53	**0.81**	0.5	0.84	0.51	0.87	0.5
	DEXTER	**0.81**	**0.85**	**0.89**	**0.9**	**0.93**	**0.9**	**0.74**	**0.71**	**0.96**	**0.91**	**0.99**	**0.95**	0.67	**0.6**	**0.91**	**0.63**	**0.97**	**0.61**
Det. Time ↓	PEDM+C	138.8	143.2	133.0	79.3	199.1	183.1	163.9	199.2	79.7	197.2	177.2	169.9	**21.3**	200.7	**19.3**	**193.6**	18.1	200.8
	DEXTER+C	**23.2**	**15.2**	**13.1**	**19.9**	**14.85**	**14.7**	**32.1**	**42.5**	**12.7**	**13.9**	**8.0**	**11.5**	60.9	**72.1**	20.0	197.1	**12.4**	**110.6**

Table 3: ARNS scenarios: Detector performance (AUC above, Detection time below).

		Cartpole						Reacher					
		Light Noise		Medium Noise		Strong Noise		Light Noise		Medium Noise		Strong Noise	
	Detector	1-step	2-step	1-step	2-step	1-step	2-step	1-step	2-step	1-step	2-step	1-step	2-step
AUROC ↑	CPD: OCD	0.66	0.66	0.68	0.68	0.67	0.68	0.51	0.51	0.51	0.51	0.51	0.51
	CPD: Chan	0.67	0.68	0.68	0.69	0.68	0.7	0.51	0.51	0.51	0.51	0.51	0.51
	PEDM	0.66	0.64	0.63	0.61	0.59	0.56	0.52	0.51	**0.55**	0.55	0.51	0.5
	DEXTER	**0.73**	**0.73**	**0.88**	**0.8**	**0.84**	**0.77**	**0.56**	**0.62**	0.51	**0.7**	**0.55**	**0.67**
Det. Time ↓	PEDM+C	33.0	44.5	53.1	50	44.5	125.3	**189.4**	197.7	201.0	201.0	**195.8**	197.2
	DEXTER+C	**32.2**	**16.9**	**17.8**	**37.8**	**24.0**	**27.4**	199.0	**95.1**	**193.6**	**61.1**	198.7	**66.7**

Table 4: Benchmark scenarios: Detector performance (AUC above, Detection time below).

		Cartpole										Reacher									
		Action Fact.		Action Noise		Action Offset		Body M. Fact.		Force Vector		Action Fact.		Action Noise		Action Offset		Body M. Fact.		Force Vector	
	Detector	Minor	Severe	Minor	Severe	Minor	Severe	Minor	Severe	Minor	Severe	Minor	Severe	Minor	Severe	Minor	Severe	Minor	Severe	Minor	Severe
AUROC ↑	PEDM	0.59	**0.94**	0.66	**0.98**	**0.82**	**1.0**	0.6	**0.9**	0.59	**1.0**	0.62	**0.95**	0.59	**0.99**	0.61	**0.98**	0.64	0.56	0.74	**0.98**
	DEXTER	**0.76**	0.64	**0.75**	0.55	0.76	0.71	**0.75**	0.66	**0.72**	0.56	**0.73**	0.69	**0.72**	0.62	**0.74**	0.66	**0.71**	**0.73**	**0.76**	0.69
Det. Time ↓	PEDM+C	153.1	**1.7**	76.3	**1.7**	200.0	**1.1**	151.3	**2.9**	127.1	**1.1**	199.6	196.6	200.8	**52.7**	200.5	192.3	199.3	201.0	200.1	200.8
	DEXTER+C	**44.8**	50.4	**42.1**	69.4	**44.0**	52.8	**46.2**	50.9	**48.2**	200.0	**53.6**	**53.3**	**51.9**	129.9	**46.8**	**65.6**	**53.7**	**52.1**	**44.0**	**55.2**

7.1.3 Probabilistic Ensemble Dynamics Model (PEDM). We implement the PEDM from Haider et al. [16] as the state-of-the-art benchmark for generalized out-of-distribution detection. For a fair comparison, we further also implement a CUSUM-based PEDM detector analogously to DEXTER+C, which we refer to as PEDM-C.

7.1.4 Changepoint detectors. In the newly introduced scenarios, we also implement change point detectors from Chen et al. [7], which the authors introduce under the name OCD, as well as another changepoint detector proposed by Chan [5]. OCD conducts likelihood ratio tests against simple alternatives of different scales in each coordinate, and then aggregates these test statistics across scales and coordinates. Unlike OCD, which is tailored for detecting changes in the mean of a multi-dimensional Gaussian data stream, a detector from Chan [5] tests the null hypothesis of data following a multivariate normal distribution against an alternative where one of the coordinates has a mixture distribution. Such methods constitute state-of-the-art models used to identify changes in high-dimensional data streams, with a focus on changes to the mean, from online changepoint detection literature in statistics. More information on their implementation is provided in the Appendix.

7.2 Detection Performance Metrics

To evaluate the performance of the different detectors, we consider both AUROC as well as average detection time, i.e. the average number of steps before an anomalous episode is classified as out-of-distribution. If the anomaly is not classified as OOD before the end of the episode, we set the detection time to the maximum episode length.

7.3 Detection Performance Results in novel scenarios.

The results for both performance metrics and for all detectors across all scenarios and environments are given in Tables 1, 2, 3.

AUROC results. We find that DEXTER outperforms all other detectors in the vast majority of scenarios and noise levels, achieving the best performance in 2 out of 2 ARTS settings, 17 out of 18 ARNO settings, and 11 out of 12 ARNS settings. We observe that PEDM detector performs no better than random guessing in ARTS setting (average AUROC of 0.51), as well as marginally better than random guessing in ARNO scenarios (average AUROC of 0.59). The main exception is the 1-step correlated noise setting in ARNO Reacher environment, where PEDM performs with an average of 0.84 AUROC across the three noise levels, and outperforms DEXTER on the Light Noise level. The performance of PEDM on ARNS is similar, with an average AUROC score of 0.57 across all noise levels and environments. Such results confirm our hypothesis about the model's limitations outlined in Section 2.0.2. CPD detectors perform only marginally better than PEDM across the three types of noise in Cartpole and Acrobot, though their AUROC scores across Reacher

environments in ARNO and ARNS settings are below PEDM, with an average of around 0.51. The scores of changepoint detectors in these settings are also quite repetitive, since in the vast majority of cases, they consistently detect an anomaly at one of the earliest steps in the episode. Since both algorithms are designed to identify changes to the *mean* of time series data, this may be especially difficult in high-dimension noisy environments, such as Reacher.

Detection time results. We find that DEXTER+C outperforms PEDM+C by a large margin in the vast majority of settings, in accordance with the higher AUROC scores.

7.4 Detection Performance Results in Benchmark Scenarios.

We observe in Table 2 that DEXTER (DEXTER+C) generally outperforms PEDM (PEDM+C) for minor anomaly scenarios, while the opposite holds true for severe anomalies. This finding is as expected, as DEXTER requires a larger time-series window to extract features. These results suggest that a combination of both DEXTER (DEXTER+C) and PEDM (PEDM+C) approaches might result in optimal outcomes.

8 CONCLUSION AND FUTURE WORK

This paper focused on the problem of generalized OOD detection in reinforcement learning. We started by outlining the terminological discussions that are currently present in the literature, and offered a framework that formalizes the problem and aligns its terminology with the broader field of generalized OOD detection in machine learning. We then outlined potential shortcomings in the architectures of current state-of-the-art detectors, and introduced a new set of testing scenarios for generalized OOD detection in RL.

By introducing noise with different types of autocorrelations, these scenarios focus on anomalies that, to the best of our knowledge, have not been explicitly studied in the literature, but are highly relevant to many real-world scenarios. We also proposed a new generalized OOD detection model called DEXTER, which first extracts relevant time series features from observations, and then applies an ensemble of isolation forest algorithms to detect potential anomalies. Its extension DEXTER+C further establishes a decision rule by use of CUSUM.

Lastly, we also adopted several existing changepoint detection methods from statistics for generalized OOD detection. After a series of experiments, we found that DEXTER outperforms the existing state-of-the-art models for OOD detection on the novel test scenarios. However, in some benchmark scenarios, a combination of previously used approaches together with DEXTER appears to yield improved results. In future work, we plan to explore such combinations of different detection mechanisms.

Despite the successes of our method, it still faces several important limitations, which point in the directions to take in future work. First, our evaluations are limited to simulated environments. Future work should test on sim-to-real settings. Second, we do not test on scenarios with noise that is correlated across dimensions. Future work should address cross-dimensional feature extraction. Third, DEXTER uses a fixed window length to detect anomalies, which is fixed before interacting with the test-time environment. Future work should study how this can be replaced by a sliding scale and/or hierarchical pyramid of window sizes. Lastly, drawing on existing work [30], we foresee the process of effectively selecting relevant features at test-time as a natural step to improve DEXTER's efficiency in high-dimensional environments.

Generalized OOD detection is critical to ensure the safety of RL algorithms when they are deployed in the real world. Unlike the kinds of out-of-distribution perturbations that have been considered in previous work, the robotic systems such as drones that have already been used in the real world [17] risk facing noise that is correlated across timepoints. This noise could come from physical disturbances such as a malfunctioning robot joint or a partially broken camera lens. If the disturbance at each individual time point is slight, it risks not to be caught by systems relying on one-step transitions like PEDM.

However, over a long time period, these disturbances can cause significant damage to the system, decreasing performance and putting potentially dangerous strain on the rest of the robot. Further, not addressing temporally correlated noise risks leaving the control system open to adversarial attacks, which induce perturbations that are correlated across time, such as illusory attacks [13]. This suggests that inclusion of a DEXTER-like system for detection of temporally-correlated noise – possibly in conjunction with a PEDM-like system optimized at detecting one-step perturbations – is necessary to ensure safe deployment of RL systems.

More broadly, we hope that our work helps shift future OOD dynamics detection towards more general approaches, improving both AI safety and security. Importantly, this work helps facilitate the move from individual time-step decision rules to information-theoretic optimal CUSUM detection methods.

ACKNOWLEDGMENTS

We'd like to thank James Duffy for an in-depth overview of out-of-distribution detection methods in econometrics and statistics. We further thank Philip Torr for helpful comments. LN was supported by the Center for AI Safety (CAIS) and Future of Life Institute for this project. KS was supported by a Cusanuswerk Doctoral Fellowship. CSDW received generous support from the Cooperative AI Foundation. This project received funding by Armasuisse Science+Technology.

REFERENCES

[1] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. 2020. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research* (2020).
[2] GE Box, Gwilym M Jenkins, and Gregory C Reinsel. 2015. i GM Ljung, Time series analysis: forecasting and control.
[3] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time series analysis: forecasting and control.* John Wiley & Sons.
[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. arXiv:arXiv:1606.01540
[5] Hock Peng Chan. 2017. Optimal sequential detection in multi-stream data. (2017).
[6] Stephanie CY Chan, Samuel Fishman, John Canny, Anoop Korattikara, and Sergio Guadarrama. 2019. Measuring the reliability of reinforcement learning algorithms. *arXiv preprint arXiv:1912.05663* (2019).
[7] Yudong Chen, Tengyao Wang, and Richard J Samworth. 2022. High-dimensional, multiscale online changepoint detection. *Journal of the Royal Statistical Society Series B: Statistical Methodology* 84, 1 (2022), 234–266.
[8] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W Kempa-Liehr. 2018. Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing* 307 (2018), 72–77.

[9] Mohamad H Danesh and Alan Fern. 2021. Out-of-Distribution Dynamics Detection: RL-Relevant Benchmarks and Results. *arXiv preprint arXiv:2107.04982* (2021).

[10] Christian Schroeder de Witt and Thomas Hornigold. 2019. Stratospheric Aerosol Injection as a Deep Reinforcement Learning Problem. https://doi.org/10.48550/arXiv.1905.07366 arXiv:1905.07366 [physics, stat].

[11] Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. 2022. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature* 602, 7897 (2022), 414–419.

[12] Alfonso Delgado-Bonal and Alexander Marshak. 2019. Approximate entropy and sample entropy: A comprehensive tutorial. *Entropy* 21, 6 (2019), 541.

[13] Tim Franzmeyer, João F Henriques, Jakob N Foerster, Philip HS Torr, Adel Bibi, and Christian Schroeder de Witt. 2022. Illusionary Attacks on Sequential Decision Makers and Countermeasures. *arXiv preprint arXiv:2207.10170* (2022).

[14] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*. PMLR, 1587–1596.

[15] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR.

[16] Tom Haider, Karsten Roscher, Felippe Schmoeller da Roza, and Stephan Günnemann. 2023. 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*. 851–859.

[17] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. 2023. Champion-Level Drone Racing Using Deep Reinforcement Learning. *Nature* 620, 7976 (Aug. 2023), 982–987. https://doi.org/10.1038/s41586-023-06419-4

[18] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems* 30 (2017).

[19] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 eighth ieee international conference on data mining*. IEEE, 413–422.

[20] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2012. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6, 1 (2012), 1–39.

[21] S Lawrence Marple Jr and William M Carey. 1989. Digital spectral analysis with applications.

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* (2015).

[23] Aaqib Parvez Mohammed and Matias Valdenegro-Toro. 2021. Benchmark for out-of-distribution detection in deep reinforcement learning. *arXiv preprint arXiv:2112.02694* (2021).

[24] Robert Müller, Steffen Illium, Thomy Phan, Tom Haider, and Claudia Linnhoff-Popien. 2022. Towards Anomaly Detection in Reinforcement Learning. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. 1799–1803.

[25] Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. 2018. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282* (2018).

[26] E. S. Page. 1954. Continuous Inspection Schemes. *Biometrika* 41, 1/2 (1954), 100–115. https://doi.org/10.2307/2333009 Publisher: [Oxford University Press, Biometrika Trust].

[27] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. *Proximal Policy Optimization Algorithms*. Technical Report. arXiv.

[28] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[29] Andreas Sedlmeier, Thomas Gabor, Thomy Phan, Lenz Belzner, and Claudia Linnhoff-Popien. 2019. Uncertainty-based out-of-distribution classification in deep reinforcement learning. *arXiv preprint arXiv:2001.00496* (2019).

[30] Hui Yie Teh, I Kevin, Kai Wang, and Andreas W Kempa-Liehr. 2021. Expect the unexpected: unsupervised feature selection for automated sensor anomaly detection. *IEEE Sensors Journal* 21, 16 (2021), 18033–18046.

[31] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. MuJoCo: A physics engine for model-based control. In *2012 International Conference on Intelligent Robots and Systems*.

[32] A. Wald. 1945. Sequential Tests of Statistical Hypotheses. *The Annals of Mathematical Statistics* 16, 2 (1945), 117–186. https://www.jstor.org/stable/2235829 Publisher: Institute of Mathematical Statistics.

[33] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. 2021. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334* (2021).