

Out-of-Distribution Detection for Reinforcement Learning Agents with Probabilistic Dynamics Models

Tom Haider
Fraunhofer IKS
Munich, Germany
tom.haider@iks.fraunhofer.de

Felippe Schmoeller da Roza
Fraunhofer IKS
Munich, Germany
felippe.schmoeller.da.roza@iks.fraunhofer.de

Karsten Roscher
Fraunhofer IKS
Munich, Germany
karsten.roscher@iks.fraunhofer.de

Stephan Günnemann
Technical University of Munich
Munich, Germany
guennemann@in.tum.de

ABSTRACT

Reliability of reinforcement learning (RL) agents is a largely unsolved problem. Especially in situations that substantially differ from their training environment, RL agents often exhibit unpredictable behavior, potentially leading to performance loss, safety violations or catastrophic failure. Reliable decision making agents should therefore be able to cast an alert whenever they encounter situations they have never seen before and do not know how to handle. While the problem, also known as out-of-distribution (OOD) detection, has received considerable attention in other domains such as image classification or sensory data analysis, it is less frequently studied in the context of RL. In fact, there is not even a common understanding of what OOD actually means in RL. In this work, we want to bridge this gap and approach the topic of OOD in RL from a general perspective. For this, we formulate OOD in RL as severe perturbations of the Markov decision process (MDP). To detect such perturbations, we introduce a predictive algorithm utilizing probabilistic dynamics models and bootstrapped ensembles. Since existing benchmarks are sparse and limited in their complexity, we also propose a set of evaluation scenarios with OOD occurrences. A detailed analysis of our approach shows superior detection performance compared to existing baselines from related fields.

KEYWORDS

Reinforcement Learning; OOD Detection; Anomaly Detection; AI Safety; Sequential Decision Making

ACM Reference Format:

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 *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023)*, London, United Kingdom, May 29 – June 2, 2023, IFAAMAS, 9 pages.

1 INTRODUCTION

Reinforcement learning has been successfully applied to many domains, achieving beyond human-level performance on a wide range of complicated sequential decision-making problems [24, 26, 34].

While it offers a promising path to solving tasks that currently cannot be solved by any other approach, RL is not yet commonly applied to real-world scenarios. One of the main reasons for this is the lack of reliability RL agents provide. Even though many efforts have been made to ensure or even guarantee the desired behavior of RL methods during runtime [1, 4, 8, 36], these assurances only hold for situations that closely resemble those that were encountered during training. In situations that substantially differ from the learning environment however, learned models and policies are prone to produce unpredictable outputs, without signaling any uncertainty about the current inputs. This can lead to performance loss, safety violations or even catastrophic failure. Detecting unseen situations is, therefore, a necessary measure towards RL systems that can be deployed in real-world scenarios [17].

Naturally, this problem not only applies to RL, but to learning-based systems in general. In fact, *understanding and detecting differences between inputs a system was trained on and inputs a system is deployed on*, is a widely studied problem in the field of Machine Learning (ML). Following [40], the problem can be generally subdivided into two major streams: **1) Covariate Shift Detection**, focusing mostly on sensory anomalies (e.g., sensor noise, sensor drift), and **2) Semantic Shift Detection**, focusing on semantic anomalies (e.g., occurrence of new semantic concepts), often also referred to as Out-of-distribution (OOD) inputs. But how does this translate to the context of RL? Essentially, **1)** is well understood, since methods from classical sensor noise detection also directly apply to most RL setups. The equivalent of **2)** however, i.e., recognizing semantically anomalous situations, is a largely underexplored field. While some work does exist under the name of *OOD Detection for RL*, they mostly cover sensor noise, limited evaluation scenarios or narrow problem definitions.

The goal of our work is to provide a more general scheme of how OOD can be approached for RL problems, in terms of problem formulation, detection algorithms and evaluation scenarios. While previous work often considers sensor noise as OOD, we argue that this definition is not sufficient. Instead, we propose a formalization of OOD as severe perturbations of the MDP, which effectively change the semantics of the system. Although being relatively simplistic, this formalization allows us to build clear expectations towards the RL agent and the detection algorithm, which we can evaluate. Our main contribution is a novel algorithm for detecting semantic perturbations in MDPs, utilizing probabilistic dynamics

models and bootstrapped ensembles. Whilst several previous works have explored uncertainty-aware deep neural networks [23] for learning dynamics models [9, 11, 14], our work is, to the best of our knowledge, the first to link these approaches for OOD detection in the context of RL. We analyze our approach both on existing benchmarks with sensor noise, and novel evaluation environments with semantic shift scenarios. Empirical results indicate superior detection performance of our proposed detection algorithm compared to baselines.

2 RELATED WORK

According to [40], *the goal of Out-of-Distribution (OOD) detection is to classify test samples within the label space and to reject samples with semantics outside the support of the label space*. Similarly, anomaly detection (AD), is the task of *identifying patterns that deviate substantially from the predefined normality*. This can either be caused by covariate or semantic shift. Most previous works are however inconsistent, when referring to different types of anomalies and hence, AD, novelty-detection, outlier-detection or OOD-detection are often used interchangeably. In this work, we try to adhere to the taxonomy from [40] as much as possible.

In a comprehensive survey, [7] provides a detailed analysis of many classical algorithms for AD (both sensory and semantic), including classification-, clustering-, nearest-neighbor-, and statistics-based approaches. [18] introduces a baseline for OOD-detection in classification tasks utilizing maximum SoftMax probabilities. In a more recent work, [6] review deep-learning based techniques for OOD detection. [22] specifically focus on time series and compare both classical and modern techniques on a novel benchmark. Surprisingly they find classical algorithms to be generally superior. However, none of the works from these surveys and benchmarks include RL as an area of application.

Nevertheless, there also exist several efforts more closely related to OOD in RL. [28] approach AD in RL on a conceptual level. [16] study the problem of domain shift in RL in a broader sense and suggest decomposing the disturbances into aspects of the MDP. [35] provide a first approach to detect OOD in RL, using the entropy of predicted actions of an agent as a score to detect novel states. However, they only consider a simple *GridWorld* task or the *LunarLander* environment with only two types of disturbances, (changed grid-layout and changed target-platform position respectively). Furthermore, their approach is limited to Q-value based algorithms only. [27] make a step towards more semantically meaningful disturbances (e.g. change of gravity, external forces) but only consider *Pong* & discrete *Cartpole* as evaluation environments. They do not propose a solution towards the benchmark they introduce but instead evaluate [35] on it. Unfortunately however, this benchmark is still proprietary. [3, 15] also introduce benchmarks for detection in GridWorld environments. [10] coin the term Out-of-Distribution Dynamics Detection (OODD) and propose a sequential model based on agent observations to detect OOD states. However, they construct these *OOD* states solely by adding observational noise.

While in this work we deliberately focus on RL, AD also received considerable attention in the closely related field of robotics. For instance, [19] use an SVM on embeddings from an Encoder-Decoder

network to detect anomalies on synthetic data and real-world robot manipulation tasks. [2] deploy variational autoencoders for anomaly detection in a supervised manner with outlier exposure. [39] deploy a feature extractor and normalizing flow with an input of RGB, depth and surface normals for robot navigation. [21] combine outputs of a path-planner with visual inputs for robot navigation. [32] use the reconstruction probability of an auto-regressive LSTM-VAE as an anomaly score estimator for a robot assisted feeding task. The latter three of these works are however designed for very specific applications and are not generally applicable in other domains. [19] on the other hand offers a more general approach but unfortunately only uses proprietary data and code.

In summary, we make 3 important observations: 1) approaches from the field of robotics are mostly very problem specific and are not universally applicable to other tasks. 2) Existing evaluation scenarios for semantic anomaly detection in RL are limited, either in terms of task complexity or in the types of disturbances they introduce. 3) Methods for OOD detection in RL are very sparse and related approaches from other domains such as classification tasks can not directly be translated to RL. What is more, there appears to be no existing algorithm that aims at *semantic* OOD detection for RL agents. We attempt to fill this gap by introducing a model-based OOD detection algorithm for learning-based agents. To evaluate this algorithm, we introduce a suite of evaluation scenarios that are both more complex than existing works and encompass OOD behavior deeply rooted in the semantics of RL tasks.

3 PRELIMINARIES

3.1 Problem Formulation

We consider RL agents as decision-making systems that sequentially interact with their environment by taking actions. The standard formalization of such problems is a discrete-time Markov decision process (MDP) [33]. An MDP is a tuple $\mathcal{M} := (\mathcal{S}, \mathcal{A}, r, f, \mu_0)$. \mathcal{S} denotes the state space, \mathcal{A} the action space, and $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ the reward function. $f : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ is the transition function which describes the system dynamics, also called dynamics function. $\mu_0 : \mathcal{S} \mapsto [0, 1]$ is the starting state distribution.

The agent’s goal in each time step is to take the action that maximizes the sum of future rewards $\sum_t^{\infty} \gamma^t r(s_t, a_t)$, where $\gamma \in [0, 1]$ is a discount factor that prioritizes near-term rewards.

3.2 Forward Dynamics Model Learning

Learning a forward dynamics model is the task of approximating a system’s true (but unknown) dynamics function $f(s, a)$. This can be achieved by generating experience from the system via interaction and fitting to the collected transitions [29]. More formally, let f_θ denote a learnable discrete dynamics function, parameterized by θ , that maps from the current state and action to the next state at time $t + \Delta t$: $f_\theta(s_t, a_t) = s_{t+\Delta t}$. Given an arbitrary control policy $\pi(a_t | s_t)$, we can collect data from the environment, by rolling out action sequences: $\mathcal{D} = \{(s_n, a_n), s_{n+1}\}_{n=1}^N$. To learn the dynamics function, we can simply fit $f_\theta(s, a)$ by minimizing the MSE loss: $\arg \min_{\theta} \sum_{n=1}^N \|f_\theta(s_n, a_n) - s_{n+1}\|^2$.

4 OOD FOR SEQUENTIAL DECISION MAKING TASKS

As introduced in section 2, [40] provide a clear problem definition for OOD in classification tasks. In sequential decision-making problems, however, there is no concept of a well-defined label space. There is not a single correct class label for a given input, but instead many different optimal decision-making policies can exist, representing equally optimal decisions given the same observation of the world. This makes the distinction between In-Distribution (ID) and OOD inputs very subtle since we cannot simply define ID and OOD classes. Instead, we turn to a scenario-based approach and define ID and OOD scenarios following the MDP composition introduced in [16]. That is, we sub-divide any anomalous scenarios into the individual components that build up the anomalous MDP. In other words, two tasks \mathcal{M}_A and \mathcal{M}_B , can be different only in the aspects that construct the individual MDPs.

With this, we formulate a conservative definition of OOD detection in the regime of decision-making problems as to *process test samples coming from the training MDP and reject samples coming from any semantically distinct MDP*.

More formally, let $\Gamma : \mathcal{M} \mapsto \mathcal{M}$ be a semantic perturbation to any aspect of an MDP. Samples drawn from the training MDP $\mathcal{S} \sim \mathcal{M}_A$ build up the ID space. Samples drawn from $\mathcal{S} \sim \Gamma(\mathcal{M}_A)$ build up the OOD space.

But what is a *semantic* perturbation? We argue that a semantic perturbation should effectively shift the transition function of an MDP, i.e. by introducing new semantic concepts or changing the environment dynamics. This in contrast to observational noise (covariate shift), which can be interpreted as a way of introducing partial observability, leaving the underlying dynamics of the MDP unaffected.

This leaves open the question on how strong a perturbation must be, before we expect to detect it. In this work we will mostly focus on the two edge cases:

- a) The perturbation is minor: $\Gamma(\mathcal{M}_A) \approx \mathcal{M}_A$
- b) The perturbation is severe: $\Gamma(\mathcal{M}_A) \neq \mathcal{M}_A$

Although this may oversimplify the problem in some cases, it allows us to formulate a clear hypothesis on what behavior we expect from an agent or OOD detector. That is, for a) we expect any control policy to perform just as if there was no perturbation present. Coherently, we cannot expect an OOD detector to cast an alarm. Vice versa, for b) we expect a control policy to suffer from the disturbance, leading to performance degradation or unexpected behavior. The OOD detector should however cast an alarm in this case.

5 MODEL BASED OOD DETECTION

Recent advances in model-based RL (MBRL) [9, 29] have shown that learned dynamics models can achieve high predictive performance for short-term prediction horizons, but suffer severely for longer prediction horizons [20]. We make the observation, however, that due to the interactive nature of RL problems, we can frame OOD detection as a 1-step prediction problem, classifying one transition at a time. In the following, we outline the intuition behind our approach.

Algorithm 1 Model-Based Out-of-Distribution detection

–Training–

Require: policy $\pi_A(a|s)$

- 1: generate dataset \mathcal{D} by following $\pi_A(a|s)$
- 2: Split \mathcal{D} into \mathcal{D}_{train} and \mathcal{D}_{val}
- 3: fit f_θ on \mathcal{D}_{train} using eq. (2)
- 4: calculate threshold τ for similarity measure $h(\cdot)$ on validation set \mathcal{D}_{val}

–Testing–

Require: policy $\pi_A(a|s)$, dyn. model $f_\theta(s, a)$, threshold τ

- 5: **for** Time $t = 0$ to TaskHorizon **do**
 - 6: predict $s'_{t+1} = f_\theta(s_t, a_t)$
 - 7: apply action $a_t \sim \pi_A(s_t)$ to obtain s_{t+1}
 - 8: **if** $h(\cdot) < \tau$ **then**
 - 9: $\{(s_t, a_t), s_{t+1}\} \leftarrow$ ID
 - 10: **else**
 - 11: $\{(s_t, a_t), s_{t+1}\} \leftarrow$ OOD
 - 12: **end if**
 - 13: **end for**
-

Given an arbitrary decision-making policy π_A , which is ideally already optimized to solve some task \mathcal{M}_A , we generate experience from task \mathcal{M}_A by following the policy for N timesteps. The resulting dataset $\mathcal{D} = \{s_n, a_n, s_{n+1}\}_{n=1}^N$ is then split into a training set \mathcal{D}_{train} and validation set \mathcal{D}_{val} . The training set is utilized to fit a dynamics model f_θ , whereas the validation set is used to establish a threshold τ on the model’s prediction error.

During deployment of the policy π_A in its target environment, the procedure is as follows. At any given state s_t , we query an action a_t from the policy π_A and use the forward dynamics model to predict the outcome of the action: $f_\theta(s_t, a_t)$. This prediction can be compared against the actual outcome s_{t+1} , observed after applying a_t in the environment. Transitions are thus labeled:

$$\hat{y}(s_t, a_t, s_{t+1}) = \begin{cases} ID, & \text{if } h(f_\theta(s_t, a_t), s_{t+1}) < \tau \\ OOD, & \text{otherwise} \end{cases} \quad (1)$$

where $h(\cdot)$ is a similarity function that can be interpreted as an anomaly score. Algorithm 1 summarizes the entire procedure. Evidently, our approach requires two essential components: **A)** An expressive function approximator to accurately model the system dynamics and **B)** a discriminative similarity measure to compare model predictions with actual outcomes in an environment. In the following, we describe both in more detail.

5.1 Function Approximator

Prior work has shown that high-capacity parametric models such as Deep Neural Networks (DNNs) can accurately learn complex dynamics functions [30]. Especially in combination with uncertainty estimation techniques [23], DNNs have been shown to be highly precise and relatively sample efficient. Most importantly, and in contrast to simpler approaches such as time-varying linear models or Gaussian processes, they also scale well to high-dimensional problems [9]. We, therefore, adopt the Probabilistic DNN Ensemble introduced in [23] as our forward dynamics model.

A probabilistic DNN parameterizes a probability distribution. The type of distribution can be arbitrarily complex though in our case we simply induce a Gaussian distribution, predicting the mean

and the variance of each output feature. For training we use the negative log prediction probability as our loss function:

$$-\log p_{\theta}(s_{t+1} | s_t, a_t) = \frac{\log \sigma_{\theta}^2(s_t, a_t)}{2} + \frac{(s_{t+1} - \mu_{\theta}(s_t, a_t))^2}{2\sigma_{\theta}^2(s_t, a_t)} + \text{const.} \quad (2)$$

Probabilistic outputs can model aleatoric uncertainty, i.e., the inherent stochasticity in the system. However, they can not model epistemic uncertainty, i.e., uncertainty about the model parameters. A simple but highly effective technique to model epistemic uncertainty is via bootstrapped ensembles [12, 31]. We consider an ensemble of B models, and define the ensemble output as the set of predictive probability distributions $z_{\theta} = \{f_{\theta,b}(s, a)\}_{b=1}^B$, where the subscript θ, b to refers the parameters of the “ b^{th} ” model in the ensemble. Each ensemble member starts from different initialization parameters and is trained on a different bootstrap of the dataset $\mathcal{D}_{\text{train}}$. The individual datasets $\mathcal{D}_{\text{train},b}$ are generated by drawing from $\mathcal{D}_{\text{train}}$ with replacement. We found $B = 5$ sufficient for all our experiments. By using an ensemble of probabilistic models we can appropriately model both epistemic and aleatoric uncertainty. We coin this model Probabilistic Ensemble Dynamics Model (PEDM). We found $B = 5$ sufficient for all our experiments. By using an ensemble of probabilistic models we can appropriately model both epistemic and aleatoric uncertainty. We coin this model Probabilistic Ensemble Dynamics Model (PEDM).

5.2 Anomaly Score (Similarity Measure)

Each ensemble member maps from the current state and action to the feature-wise mean and variance of a Gaussian distribution over the next state:

$$f_{\theta}(s_t, a_t) = \mathcal{N}(\mu_{\theta}(s_t, a_t), \sigma_{\theta}(s_t, a_t)), \quad (3)$$

where σ are the entries of a diagonal covariance matrix. The resulting set of probability distributions can be used to calculate an anomaly score as follows: Given (s_t, a_t) , we draw from each ensemble member’s output distribution K times, resulting in a set of $B * K$ representative particles:

$$\{s'_{t+1}\}^{K,B} = \left\{ \left\{ f_{\theta,b}(s_t, a_t) \right\}_{k=1}^K \right\}_{b=1}^B. \quad (4)$$

Each particle represents a possible outcome of the action a_t applied in the current state s_t and all particles together represent a non-continuous distribution of possible outcomes. We can then calculate the prediction error for each particle:

$$d(s'_{t+1,i}, s_{t+1,i}) = \frac{1}{|s|} \sum_{i=1}^{|s|} \left(s'_{t+1,i} - s_{t+1,i} \right)^2. \quad (5)$$

In all our experiments, we found $K = 200$ particles per ensemble member to be sufficient. The final anomaly score is obtained by aggregating over all ensemble member scores:

$$h(z_{\theta}(s_t, a_t), s_{t+1}) = \text{aggr.} \left(\left\{ d(s'_{t+1}, s_{t+1}) \right\}^{K,B} \right). \quad (6)$$

The choice of aggregation function effectively influences the sensitivity of the resulting detector and is nontrivial. We provide a more detailed discussion on this in section 5. Furthermore, we would like to mention that there are certainly other methods to calculate

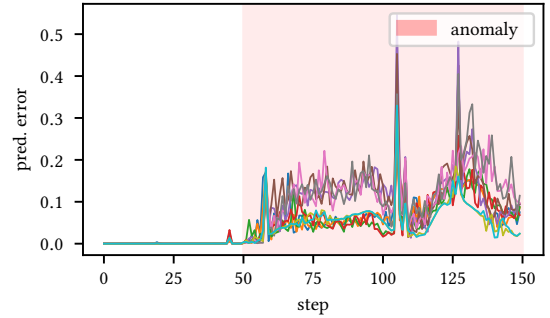


Figure 1: Individual particle’s prediction error of the PEDM on *Pusher* with anomalous force applied to the robot arm after 50 timesteps. Independent of the environment, an optimal anomaly score should be zero when there is no anomaly and high otherwise.

a meaningful anomaly score from the predicted distributions. In section 5 we also compare our approach against three alternative methods.

Figure 1 visualizes our approach on an exemplary anomalous scenario. Independent of the environment, an optimal anomaly score should be zero when there is no anomaly and high otherwise. Visibly, the prediction error of most particles shows this behavior and amplifies as soon as the anomaly is introduced. This indicates a sharp boundary between ID and OOD transitions as well as high confidence of the predictor, since all particles show similar behavior.

6 EXPERIMENTAL RESULTS

This section provides a detailed evaluation of our approach on relevant testing scenarios, for which we consider the following procedure and metrics:

Evaluation Procedure: We consider a given policy $\pi_A(a|s)$ that is optimized for some episodic task \mathcal{M}_A with episode length $H \in \mathbb{N}$. In each episode, we introduce an anomaly $\Gamma(\mathcal{M}_A)$ at a random time point $t_a \in (t_0, t_H)$ and apply this anomaly until the end of the episode. Transitions $[(s_{t_0}, s_{t_1}), \dots, (s_{t_{a-1}}, s_{t_a})]$ are labeled as ID, transitions $[(s_{t_a}, s_{t_{a+1}}), \dots, (s_{t_{H-1}}, s_{t_H})]$ are treated as OOD. This procedure is repeated for multiple random time points and different initial environment states to account for the effects injection time and initial values can have on the dynamics. Thus, the resulting dataset is balanced (in expectation).

Evaluation Metrics: Analogous to literature on binary classification, we consider the Area under the Receiver Operator Characteristic (AUC) and the F1-score as our primary evaluation metrics. All metrics are calculated for transitions and labels from 100 episodes with random injection times and random environment seeds.

6.1 Environments with Semantic Perturbations

Given the limitations of existing benchmarks depicted in section 2, we introduce a set of OOD scenarios based on established, high-dimensional continuous robotic control tasks in the mujoco physics engine [38]. Specifically, we consider *CartPole*, *HalfCheetah*, *Reacher* and *Pusher* from [9].

Table 1: *Minor Disturbances* - AUC of the detection algorithm and reward difference of the control policy averaged over scenarios with *minor* disturbances (100 episodes with random injection times for each disturbance type).

AUC	Cartpole	HalfCheetah	Pusher	Reacher
<i>reward</i>	0%	0%	-1%	-1%
GAUSSIAN	0.27	0.48	0.23	0.43
GMM	0.25	0.47	0.32	0.42
IFOREST	0.23	0.47	0.25	0.39
KNN	0.27	0.50	0.44	0.60
KNN+	0.29	0.50	0.44	0.55
LSTM	0.35	0.49	0.29	0.31
LSTM+	0.30	0.50	0.27	0.31
RIQN	0.46	0.47	0.25	0.40
PEDM	0.55	0.53	0.58	0.47

In each environment we apply the same types of semantic disturbances (only one at a time):

- **Body Mass factor:** The mass of all body parts is multiplied by some constant factor.
- **Action factor:** The action vector a produced by the policy is multiplied by some constant factor.
- **Action Offset:** Some constant is added to the action vector a produced by the policy.
- **Action noise:** Gaussian noise is applied to the action centering around the original a with constant variance.
- **Force Vector:** External constant force vector applied to a single body part of the robot. In *Cartpole* the force is applied to the pole, in *HalfCheetah* to the front foot and in *Reacher/Pusher* to the forearm of the robot.

It is worth noting that actions also actively influence the dynamic behavior of an MDP and therefore we argue, that modifying the action before it is applied can be regarded as a semantic disturbance. This is in contrast to observational noise, which can be interpreted as a way of introducing partial observability. The dynamic behavior of the MDP however is unaffected by the observations of the agent.

As stated in section 3, we focus on two edge cases of anomalous behavior for this work, namely minor and severe perturbations. For minor instances of the disturbances described above, we change the original environment parameters or action values by 1% in magnitude. For severe disturbances, the original values are perturbed up to 100% in magnitude. Coherently, for the *force applied* disturbance, relatively weak forces are applied as minor disturbances while strong force vectors are applied for severe cases. These parameters are carefully selected for every environment such that they visibly influence the control policy, while not degrading it entirely, making anomalies non-trivial to detect. A detailed list of these parameters is provided in the appendix.¹

6.2 Evaluation on Environments with OOD

We compare our approach against several baselines from the AD literature. As described in section 2 and to the best of our knowledge, only two approaches specifically targeted at OOD in RL exist: RIQN

¹Code and supplementary material are publicly available at: <https://github.com/FraunhoferIKS/pedm-ood>

Table 2: *Severe Disturbances* - AUC of the detection algorithm and reward difference of the control policy averaged over scenarios with *severe* disturbances (100 episodes with random injection times for each disturbance type).

AUC	Cartpole	HalfCheetah	Pusher	Reacher
<i>reward</i>	-2%	-15%	-9%	12%
GAUSSIAN	0.39	0.69	0.58	0.5
GMM	0.48	0.75	0.71	0.66
IFOREST	0.42	0.76	0.57	0.49
KNN	0.49	0.92	0.80	0.78
KNN+	0.65	0.93	0.81	0.76
LSTM	0.55	0.84	0.59	0.55
LSTM+	0.64	0.84	0.68	0.57
RIQN	0.55	0.67	0.42	0.46
PEDM	0.96	0.93	0.85	0.90

[10] and UBOOD [35]. Since the latter can only apply to discrete action problems, only RIQN is included in our benchmarks. This results in the following baselines:

- **GAUSSIAN:** Likelihood estimation on a (multivariate) Gaussian distribution;
- **GMM:** Likelihood estimation on a mixture of (multivariate) Gaussian distributions;
- **IFOREST:** Anomaly score estimation via Isolation Forest algorithm;
- **KNN:** Distance calculation to the k-Nearest Neighbors in the nominal training data;
- **LSTM:** Distance calculation based on the predictions of a regressive LSTM-RNN;
- **RIQN:** from [10] with mean L1-prediction error as anomaly score.

To account for the fact that our proposed approach considers states and actions as inputs, we also add the same information to two of the baseline models. KNN+ refers to a KNN on (s_t, a_t, s_{t+1}) tuples, LSTM+ to an LSTM with actions as additional model inputs. More details on each baseline are provided in the appendix.

For testing, we train an RL agent until convergence in the nominal task. This serves as a control-policy the OOD detector aims to monitor. In all our experiments use Twin Delayed DDPG (TD3) [13], a direct successor of DDPG [25] and a state-of-the-art policy gradient algorithm for continuous action spaces. Since our approach is agnostic to the control policy, it can be replaced by any other controller in practice. To generate data, we collect experience with the TD3 agent in each nominal environment for 50 episodes. 90 percent of the data is used for training the prediction models and the rest is only used for validation/thresholding. The results for minor and severe disturbances are shown in table 1 and table 2 respectively and are discussed in the following.

6.2.1 *Minor Disturbances.* The AUC scores in table 1 indicate that all approaches struggle to differentiate normal from perturbed transitions on scenarios with minor disturbances.

Analyzing the environment reward, however, it appears that the performance of the policy is unaffected by all disturbances, yielding nearly the same average accumulated reward as in the respective

Table 3: Existing Benchmark AUC of the detection algorithm and reward of the control policy on existing benchmark for scenarios from [10]. Results for RIQN are taken directly from the original publication. All values are averaged over 100 episodes with random anomaly injection timepoints.

Anomaly/metric		Acrobot		CartPole		LunarLander		Ant		HalfCheetah		Hopper		Walker	
		AUC	rew.	AUC	rew.	AUC	rew.	AUC	rew.	AUC	rew.	AUC	rew.	AUC	rew.
RIQN	Nominal	-	-79	-	500	-	213.55	-	3297	-	2821	-	2678	-	2241
	Anom Avg.	0.95	-96	0.91	431	0.91	139.20	0.94	2164	0.92	2062	0.92	1877	0.92	1634
ours	Nominal	-	-81	-	500	-	222.33	-	3293	-	2824	-	2658	-	2238
	Gaussian Noise	1.00	-191	1.00	311	1.00	-129.41	1.00	1882	0.99	2011	1.00	1320	1.00	1511
	Sensor Shutdown	1.00	-96	0.89	296	0.87	29.71	0.99	1863	0.99	1003	0.99	1244	0.99	1095
	Calibr. Failure	1.00	-101	0.96	381	0.89	136.42	1.00	2221	0.96	2164	0.95	1558	0.93	1413
	Sensor Drift	1.00	-80	0.57	468	0.74	192.81	0.87	3173	0.65	2807	0.77	2599	0.63	2217
	Anom Avg.	1.00	-117	0.85	364	0.87	57.38	0.97	2285	0.90	1996	0.93	1680	0.89	1559

nominal scenario. This is a strong indication, that minor perturbations do not substantially affect the behavior of the deployed control policy. Thus it can be acceptable, under most circumstances, that these instances are not detected.

What is also peculiar is that most baselines perform even worse than average, especially in *Cartpole* and *Pusher*. While this might be counter intuitive at first, we have a clear explanation for this: These environments include a short but prominent transient phase, where the robot moves from its initial state to a stable position, where it stays for the rest of the episode. This initial phase is therefore a region of relatively low density, compared to the stable phase. Approaches following the density of the data therefore yield a high anomaly score for the initial phase, which is even higher than for the actual OOD phase. Since the transient phase is only a small part of the episode, however, this leads to observed low AUC scores. A more detailed investigation of this phenomenon can be found in the appendix. For PEDM this is not the case since it effectively learns the dynamics of the environment, not only looking at the data density. While it also has a slightly higher prediction error initially, the difference is not as prominent as in other approaches. Since the PEDM prediction error is however very stable over the course of an episode and not notably different when a minor disturbance is present, this leads to an AUC of ~ 0.5 .

6.2.2 Severe disturbances. For severe disturbances, the pattern is considerably different. As apparent from table 2, most approaches achieve beyond average performance. Most notably, the reward of the control policy visibly suffers in the majority of scenarios, the exception being *Cartpole*. Apparently, the TD3 policy is relatively robust on *Cartpole*, countering even strong disturbances. Thus, baselines struggle for the same reason as for minor disturbances. PEDM on the other hand, has a strong AUC score on all environments. It is conditioned on the policy’s actions, and thus, highly sensitive to the direct outcomes of actions. This is the case even if the policy is robust enough to correct smaller control errors and never leaves the high density region it already encountered during training. This is also the reason why KNN+ and LSTM+ perform slightly better than the default versions. Conditioning these models on the action pushes them closer towards an actual dynamics model. Overall LSTM+ and especially KNN+ are very strong baselines already. KNN+ is even on par with our approach on *HalfCheetah*.

Table 4: Comparison of different techniques for computing an anomaly score from model predictions. \mathcal{A} -dim, \mathcal{S} -dim refer to the dimensions of the Action- and State-space.

AUC	Cartpole	HalfCheetah	Pusher	Reacher
\mathcal{A} -dim, \mathcal{S} -dim	1, 5	5, 23	7, 20	7, 17
PEDM-var	0.49	0.7	0.74	0.48
PEDM-pdf	0.95	0.84	0.81	0.87
PEDM-cdf	0.95	0.85	0.8	0.87
PEDM-samp.	0.96	0.93	0.85	0.89

At this point we want to mention that while the lower reward is a direct indicator for some changes to the MDP, the opposite is not the case. The MDP can in theory be altered drastically without altering the policy reward at all. However, we do not consider such hidden cases in this work, and only focus on scenarios where the reward is directly correlated with the magnitude of a disturbance to the MDP.

6.3 Evaluation on Benchmark with Sensor Noise

We also evaluate our approach on the scenarios from [10] where different types of sensor noise are applied. Results are presented in table 3. Looking only at the aggregate AUC over all disturbances our approach appears comparable with the baseline - surpassing it in 3, and getting outperformed in 4 of the 7 environments. Interestingly, our approach achieves high AUC values for all disturbances but *Sensor Drift*. This is however the disturbance that appears to affect the reward of the control policy the least, indicating only a minor perturbation of the nominal MDP. Unfortunately, we were unable to reproduce the results from the original publication and therefore cannot make any more accurate comparisons at this point. In summary, our approach also works well for severe cases of covariate shift and is at least comparable to the existing baseline in this setting.

6.4 Ablation Studies

6.4.1 Analysis of Anomaly Score techniques. As mentioned above, there are also other techniques to map the model predictions z_θ to an anomaly score. In the following, we describe three alternatives to our sampling-based approach:

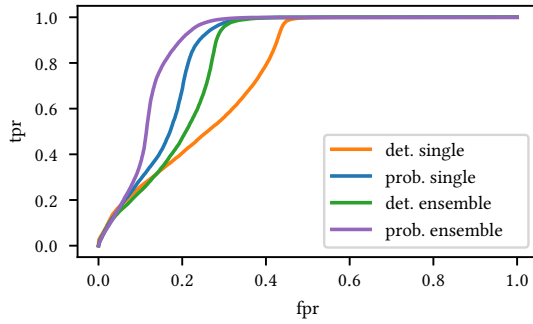


Figure 2: ROC - Influence of dynamics model on *HalfCheetah* with anomalous force applied

- (1) Prediction-error-based, analytical pdf:
Directly estimating the likelihood of the observed state s_{t+1} under the predicted distribution.
- (2) Prediction-error-based, analytical cdf (hypothesis test):
Computing the probability of observing a sample at least as extreme as s_{t+1} w.r.t. the predicted distribution.
- (3) Prediction-variance-based:
Computing the variance over all predicted particles and use this as the final anomaly score.

In table 4 we compare these different techniques. The variance-based technique consistently performs worse than techniques using the prediction error. We conjecture that ensemble members often make similar but erroneous and over-confident (low-variance) predictions on anomalous states. The aggregated variance is therefore also very low for OOD samples and only serves as a weak anomaly score.

The sampling-based technique also consistently outperforms the two analytical approaches. This can be explained by the mismatch between likelihood and probability mass in high dimensional multivariate distributions [5, 37], which results in very low likelihood estimates even for ID samples. Tiny prediction errors can therefore completely deter these estimates. The sampling based approach on the other hand is simply an empirical solution of calculating the distance to the typical set. This also scales well to higher dimensions, resulting in a much more robust anomaly score. The higher the dimension of the problem, the more pronounced is this effect. The results from table 4 also directly reflect this. We provide a more detailed explanation of this phenomenon in the appendix.

6.4.2 Dynamics Model. The type of dynamics model is central to our method. In figure 2 we visualize this on an exemplary evaluation scenario for different dynamics model choices: a single deterministic DNN, a single probabilistic DNN, an ensemble of deterministic DNNs, and an ensemble of probabilistic DNNs. The results indicate the importance of accounting for uncertainty in the model predictions, especially the combination of both aleatoric and epistemic uncertainty.

6.4.3 Aggregation Function. In our approach, individual prediction particles are aggregated for the final anomaly score. In this work, we investigated several statistics for this and visualize the impact

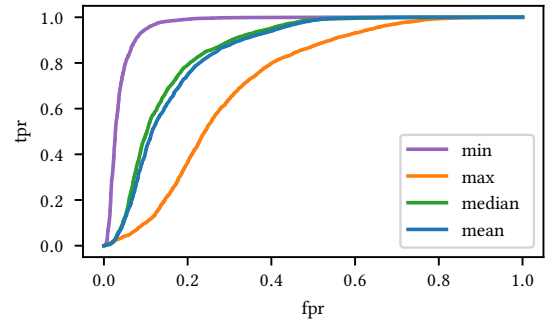


Figure 3: ROC- Different aggregation methods for individual model predictions on *Reacher* with disturbed action magnitude

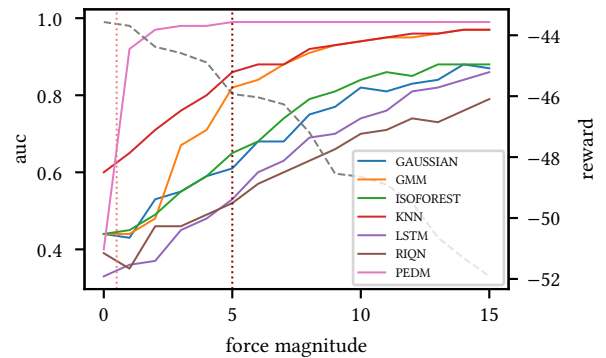


Figure 4: Detector performance (auc) and policy reward over force magnitude for *Reacher*. Grey dashed line: policy reward; pink dotted line: *minor disturbance* setting; brown dotted line: *severe disturbance* setting. All datapoints represent an average over 100 episodes with random injection times.

of a few of these measures in figure 3. The results indicate that the minimum-mse performs best, since one correct prediction suffices for an ID label and not all/the majority of predictions have to be correct. We found this to be consistent over all evaluation scenarios. In theory, if the sample size is large enough, the minimum distance between samples and the true observation approximates 0, even for OOD samples. However, we did not experience this behavior with the used sampling sizes of ~ 200 particles per ensemble member - which is a relatively low value w.r.t. to the dimensionality of the problems.

6.4.4 Disturbance Magnitude. The OOD detector should cast an alert if a normal behavior of the policy can not be expected. In the environments we consider, the magnitude of a disturbance roughly controls how much the behavior of the policy is influenced, which directly manifests in the accumulated reward. The OOD detector should therefore be able to recognize a disturbance in scenarios where the reward drops significantly. Figure 4 shows that the performance of all detectors indeed increases as the disturbance

Table 5: F1 score resulting from different classification thresholds in Algorithm 1 averaged all severe instances of our evaluation environments.

F1	Cartpole	HalfCheetah	Pusher	Reacher
agg-max	0.92	0.60	0.75	0.82
agg-max·1.5	0.91	0.53	0.74	0.80
agg-mean	0.62	0.83	0.71	0.78

gets bigger and the reward diminishes. In this example, we can also observe that PEDM already shows very good detection performance for small reductions of the reward. This means that it can detect disturbances before they affect the performance of the nominal policy beyond an acceptable level.

Figure 4 also visualizes the values for *minor* and *severe* perturbations. Although they are somewhat arbitrary single point descriptions of a complex relationship, they are still good representations of the overall trends and at the same time allow for better aggregate comparisons.

6.4.5 Threshold. So far we investigated only AUC, which is a theoretical measure considering all possible thresholds for a detector. However, for a real world deployment, a specific threshold on $h(\cdot)$ needs to be established during validation. Assuming that we do not have access to any OOD data, we compare the following approaches.

- mean validation score: $\tau = \frac{1}{N} \sum_{N_{val}} h(\cdot)$
- max. validation score: $\tau = \max \{h(\cdot)\}_{n=1}^{N_{val}}$
- 1.5 max. validation score: $\tau = 1.5 \cdot \max \{h(\cdot)\}_{n=1}^{N_{val}}$.

Intuitively the mean anomaly score on the validation set (which does not contain OOD instances) is a very conservative metric, prioritizing recall over specificity. Therefore, to increase specificity, we also consider the max validation score and the 1.5 fold of the max validation score as an alternative.

The results are presented in table 5. For *Cartpole*, *Pusher* and *Reacher*, the max validation error appears to be a reasonable, though not perfect threshold. For *HalfCheetah* on the other hand, the mean validation error seems to be the better choice. We explain this by the amount of validation data that is available. For all experiments, we use 5 of the 50 available episodes for validation. While *Cartpole*, *Pusher* and *Reacher* have episode lengths of 200, 150 and 150 respectively, *HalfCheetah* episodes are 1000 steps long. The increased amount of resulting validation data causes more peaks in the validation error, making the max. error much too conservative of a measure. We, therefore, see our thresholding only as a first step and acknowledge its considerable shortcomings, especially in low data regimes, where calibrating the classification threshold in ID data alone can be difficult.

7 CONCLUSION

In this work, we studied the problem of OOD detection in the context of RL and provided a general approach towards problem formulation, detection algorithms and evaluation scenarios. We started with a simple problem formulation in terms of severe perturbations of the MDP. Building on this understanding of the problem, we introduced a model-based OOD detection algorithm that allows us to assess how much an observed transition differs from situations the

agents was trained for. To evaluate this method, we introduced a set of OOD scenarios based on well-established robotic control tasks. In contrast to previous work studying anomaly detection in RL, the disturbances we consider in this work are deeply rooted in the semantics of each task, effectively changing its underlying transition function. Results show that our approach works well in cases of severe disturbances but struggles in cases, that do not visibly influence the behavior of the control policy it monitors. We accept this as a tolerable shortcoming since such perturbations have weaker implications on the reliability of an agent. In contrast, we see the detection of severe disturbances (which do influence the behavior of a control policy) as a necessary measure towards confirming an agent’s functioning during deployment. We acknowledge, that a clear distinction between minor and severe disturbances can not always be made. This classification is only a starting point and further efforts are required to study borderline cases. Nonetheless, we believe that this approach is an important step towards real world applications of RL, as it can mitigate failures caused by disturbances that are severe but non-trivial to detect.

It is important to note that OOD detection alone does not give any guarantees about the agent performance on ID data. Also, it does not provide any answers on how to react to OOD samples once detected and therefore still requires suitable fallback policy (e.g. handing over to a human operator). It is therefore only a first step towards more reliable RL methods.

Finally, our work only considers fully observable tasks without any long-term dependencies or observational abstraction. An exciting avenue for future research is therefore to investigate how our approach can be adapted to partially observable scenarios or scenarios with OOD behavior that can only be observed over longer time horizons.

ACKNOWLEDGMENTS

This work was funded by the Bavarian Ministry for Economic Affairs, Regional Development and Energy as part of a project to support the thematic development of the Institute for Cognitive Systems.

REFERENCES

- [1] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. 2018. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.
- [2] Davide Azzalini, Luca Bonali, and Francesco Amigoni. 2021. A minimally supervised approach based on variational autoencoders for anomaly detection in autonomous robots. *IEEE Robotics and Automation Letters* 6, 2 (2021), 2985–2992.
- [3] Jonathan Balloch, Zhiyu Lin, Mustafa Hussain, Aarun Srinivas, Robert Wright, Xiangyu Peng, Julia Kim, and Mark Riedl. 2022. Novgrid: A flexible grid world for evaluating agent response to novelty. *arXiv preprint arXiv:2203.12117* (2022).
- [4] Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. 2017. Safe model-based reinforcement learning with stability guarantees. *Advances in neural information processing systems* 30 (2017).
- [5] Bob Carpenter. 2017. Typical sets and the curse of dimensionality. *Stan Software* (2017).
- [6] Raghavendra Chalapathy and Sanjay Chawla. 2019. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407* (2019).
- [7] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)* 41, 3 (2009), 1–58.
- [8] Yinlam Chow, Ofir Nachum, Aleksandra Faust, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. 2019. Lyapunov-based safe policy optimization for continuous control. *arXiv preprint arXiv:1901.10031* (2019).
- [9] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. 2018. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems* 31 (2018).

- [10] Mohamad H Danesh and Alan Fern. 2021. Out-of-Distribution Dynamics Detection: RL-Relevant Benchmarks and Results. *arXiv preprint arXiv:2107.04982* (2021).
- [11] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. 2016. Learning and policy search in stochastic dynamical systems with bayesian neural networks. *arXiv preprint arXiv:1605.07127* (2016).
- [12] Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.
- [13] 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.
- [14] Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. 2016. Improving PILCO with Bayesian neural network dynamics models. In *Data-efficient machine learning workshop, ICML*, Vol. 4. 25.
- [15] Shivam Goel, Gyan Tatiya, Matthias Scheutz, and Jivko Sinapov. 2021. Novel-gridworlds: A benchmark environment for detecting and adapting to novelties in open worlds. In *AAMAS Adaptive Learning Agents (ALA) Workshop*.
- [16] Tom Haider, Felipe Schmoeller Roza, Dirk Eilers, Karsten Roscher, and Stephan Günnemann. 2021. Domain Shifts in Reinforcement Learning: Identifying Disturbances in Environments. In *AI Safety@IJCAI*.
- [17] Dan Hendrycks, Nicholas Carlini, John Schulman, and Jacob Steinhardt. 2021. Unsolved problems in ml safety. *arXiv preprint arXiv:2109.13916* (2021).
- [18] Dan Hendrycks and Kevin Gimpel. 2016. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136* (2016).
- [19] Rachel Hornung, Holger Urbanek, Julian Klodmann, Christian Osendorfer, and Patrick Van Der Smagt. 2014. Model-free robot anomaly detection. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 3676–3683.
- [20] Michael Janner, Qiyang Li, and Sergey Levine. 2021. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems* 34 (2021), 1273–1286.
- [21] Tianchen Ji, Arun Narenthiran Sivakumar, Girish Chowdhary, and Katherine Driggs-Campbell. 2022. Proactive anomaly detection for robot navigation with multi-sensor fusion. *IEEE Robotics and Automation Letters* 7, 2 (2022), 4975–4982.
- [22] Kwei-Herng Lai, Daochen Zha, Junjie Xu, Yue Zhao, Guanclu Wang, and Xia Hu. 2021. Revisiting time series outlier detection: Definitions and benchmarks. In *Thirty-fifth conference on neural information processing systems datasets and benchmarks track (round 1)*.
- [23] 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).
- [24] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research* 17, 1 (2016), 1334–1373.
- [25] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).
- [26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).
- [27] Aaqib Parvez Mohammed and Matias Valdenegro-Toro. 2021. Benchmark for out-of-distribution detection in deep reinforcement learning. *arXiv preprint arXiv:2112.02694* (2021).
- [28] 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.
- [29] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. 2018. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 7559–7566.
- [30] Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. 2020. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*. PMLR, 1101–1112.
- [31] Ian Osband. 2016. Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout. In *NIPS workshop on bayesian deep learning*, Vol. 192.
- [32] Daehyung Park, Yuuna Hoshi, and Charles C Kemp. 2018. A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. *IEEE Robotics and Automation Letters* 3, 3 (2018), 1544–1551.
- [33] Martin L Puterman. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.
- [34] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. 2020. Mastering atari, go, chess and shogi by planning with a learned model. *Nature* 588, 7839 (2020), 604–609.
- [35] 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).
- [36] Krishnan Srinivasan, Benjamin Eysenbach, Sehoon Ha, Jie Tan, and Chelsea Finn. 2020. Learning to be safe: Deep rl with a safety critic. *arXiv preprint arXiv:2010.14603* (2020).
- [37] MTCAJ Thomas and A Thomas Joy. 2006. *Elements of information theory*. Wiley-Interscience.
- [38] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 5026–5033.
- [39] Lorenz Wellhausen, René Ranftl, and Marco Hutter. 2020. Safe robot navigation via multi-modal anomaly detection. *IEEE Robotics and Automation Letters* 5, 2 (2020), 1326–1333.
- [40] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. 2021. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334* (2021).