Task-Driven Out-of-Distribution Detection with Statistical Guarantees for Robot Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract: Our goal is to perform *out-of-distribution (OOD) detection*, i.e., to detect when a robot is operating in environments that are drawn from a different distribution than the environments used to train the robot. We leverage Probably Approximately Correct (PAC)-Bayes theory in order to train a policy with a *guaranteed bound* on performance on the training distribution. Our key idea for OOD detection then relies on the following intuition: violation of the performance bound on test environments provides evidence that the robot is operating OOD. We formalize this via statistical techniques based on p-values and concentration inequalities. The resulting approach (i) provides guaranteed confidence bounds on OOD detection, and (ii) is *task-driven* and sensitive only to changes that impact the robot's performance. We demonstrate our approach on a simulated example of grasping objects with unfamiliar poses or shapes. We also present both simulation and hardware experiments for a drone performing vision-based obstacle avoidance in unfamiliar environments (including wind disturbances and different obstacle densities). Our examples demonstrate that we can perform task-driven OOD detection within just a handful of trials. Comparisons with baselines also demonstrate the advantages of our approach in terms of providing statistical guarantees and being insensitive to task-irrelevant distribution shifts.

Keywords: Out-of-distribution detection, generalization, PAC-Bayes

1 Introduction

Imagine a drone trained to perform vision-based navigation using a dataset of indoor environments and deployed in environments with varying wind conditions, obstacle densities, or lighting (Fig. 1). Similarly, consider a robot arm manipulating a new set of objects or an autonomous vehicle deployed in a new city. State-of-the-art techniques for learning-based control of robots typically struggle to generalize to such *out-of-distribution* (OOD) environments. This lack of OOD generalization is particularly pressing in safety-critical settings, where the price of failure is high. In this work, we focus on the problem of autonomously *detecting* when a robot is operating in environments drawn from a different distribution than the one used to train the robot. This ability to perform *OOD detection* has the potential to improve the safety of robotic systems operating in OOD environments. For example, a drone operating in a new set of environments could either deploy a highly conservative policy or cease its operations altogether. In addition, OOD detection can also allow the robot to improve its policy by re-training using additional data collected from the new environments.

There are two important desiderata that OOD detection approaches for safety-critical robotic systems should ideally satisfy. First, we would like to develop OOD detection techniques with *guaranteed confidence bounds*. Second, we would like our OOD detectors to be *task-driven* and only sensitive to *task-relevant* changes in the robot's environment. As an example, consider again the drone navigation setting in Fig. 1 and suppose that the robot's policy is insensitive to changes in color and lighting. Here, the robot's OOD detector should *not* trigger even if the robot is operating in environments with different color/lighting and should only trigger if there are task-relevant variations (e.g., variations in the obstacle density). Unfortunately, current approaches (Sec. 2) do not typically satisfy both desiderata; they are often based on heuristics and not task-driven in general.

Statement of Contributions. We develop task-driven OOD detection techniques with statistical guarantees on correctness. To this end, we make three specific contributions (see Fig. 1 for an overview).

- Given a dataset of environments drawn from an (unknown) training distribution, we develop a pipeline based on *generalization theory* for training control policies with a *guaranteed bound*

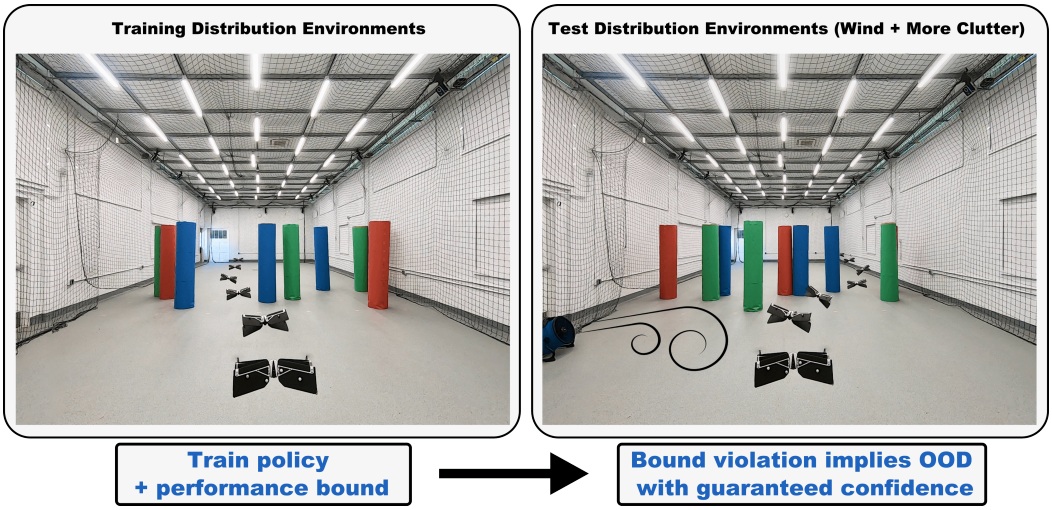


Figure 1: A schematic of our overall approach. We learn policies with guaranteed bounds on expected performance on the training distribution. Violation of this bound during deployment implies that the robot is operating OOD (with high confidence). We present simulation experiments in manipulation and navigation settings, and hardware experiments for a drone navigating in new environments with varying wind conditions and clutter.

on performance (a bound on the expected cost of the policy on the unknown training distribution). Specifically, we leverage recently-developed *derandomized Probably Approximately Correct (PAC)-Bayes* bounds that are well-suited to enable OOD detection (Sec. 4.1).

- We develop two OOD detection techniques based on the following intuition (Sec. 4.2): if the costs incurred when the robot is deployed in new environments violate the bound on the policy's performance, then this indicates that the robot is operating OOD. We formalize this via p-values and concentration inequalities, and derive statistical guarantees on the false positive rate of OOD detection. Since our OOD detection scheme operates by observing the costs incurred on new environments, it is only triggered by task-relevant changes that impact the robot's performance.
- We demonstrate our approach on two simulated examples (Sec. 5): (i) a robotic manipulator grasping a new set of objects, and (ii) a drone navigating a new set of environments. Comparisons with baselines demonstrate the advantages of our approach in terms of providing statistical guarantees and being insensitive to task-irrelevant shifts. We also present a thorough set of hardware experiments for vision-based drone navigation with varying wind conditions and clutter (Fig. 1). Taken together, our simulation and hardware experiments demonstrate the ability of our approach to perform task-driven OOD detection within just a handful of trials for systems with nonlinear/hybrid dynamics, rich sensing modalities (e.g., depth sensing), and neural network-based policies.

2 Related work

Anomaly/OOD detection in supervised learning. Anomaly detection in low-dimensional signals has been well-studied in the signal processing literature (see [1] for a review). Recent work in machine learning has focused on OOD detection for high-dimensional inputs (e.g., images) in supervised learning settings (see [2] for a review). Popular approaches use threshold-based detectors for the output distribution of a given pre-trained neural classifier [3, 4, 5]. Other methods use a specific pipeline for training a neural network in order to improve OOD detection on test samples [6, 7, 8]. However, these methods are often susceptible to adversarial attacks [9]. Thus, approaches for addressing adversarial data have been developed [10, 9, 11]. Some of these approaches are also able to provide theoretical guarantees of performance on adversarial data [9, 12, 13]. Other methods provide PAC-style statistical guarantees [14, 15] or p-values [16]. However, these methods typically focus on supervised learning settings and often require specific network outputs (e.g., softmax) that are incompatible with non-classification tasks. In contrast, we focus on OOD detection for policy learning settings in robotics and do not make assumptions about the specific structure of the policy.

Task-driven OOD detection. The methods above are aimed at detecting *any* distributional shift in the data and can be sensitive even to task-*irrelevant* shifts (i.e., ones that do not impact performance) as we demonstrate in our experiments (Sec. 5). A recent approach determines an estimate of input atypicality for pre-trained networks and uses it as an OOD detector in supervised learning settings [17]. Recent methods have also been developed specifically for reinforcement learning (RL) [18,

19, 20, 21, 22]. In particular, [22] presents a general task-driven approach for OOD detection on sequential rewards, which is optimal in certain settings. However, neither this method nor others in the RL context provide statistical guarantees on detection. We propose an OOD detection framework which is both task-driven and provides statistical guarantees by leveraging generalization theory.

Generalization theory. Generalization theory provides a way to learn hypotheses (in supervised learning) with a bound on the true expected loss on the underlying data-generating distribution given only a finite number of training examples. Original frameworks include Vapnik-Chervonenkis (VC) theory [23] and Rademacher complexity [24]. However, these methods often provide vacuous generalization bounds for high-dimensional hypothesis spaces (e.g., neural networks). Bounds based on PAC-Bayes generalization theory [25, 26, 27] have recently been shown to provide strong guarantees in a variety of settings [28, 29, 30, 31, 32, 33], and have been significantly extended and improved [34, 35, 36, 37, 38, 39]. PAC-Bayes has also recently been extended to learn policies for robots with guarantees on generalization to novel environments [40, 41, 42, 43]. In the present work, we leverage recently-proposed *derandomized* PAC-Bayes bounds [44]; this framework allows us to train a single deterministic policy with a guaranteed bound on expected performance on the training distribution (in contrast to [40, 41, 42, 43], which train stochastic neural network policies). This forms the basis for our OOD detection framework: by observing violations of the PAC-Bayes bound on test environments, we are able to perform task-driven OOD detection with statistical guarantees.

3 Problem formulation

Dynamics and environments. Let $s_{t+1} = f_E(s_t, a_t)$ describe the robot's dynamics, where $s_t \in \mathcal{S} \subseteq \mathbb{R}^{n_s}$ is the state of the robot at time-step t, $a_t \in \mathcal{A} \subseteq \mathbb{R}^{n_a}$ is the action, and $E \in \mathcal{E}$ is the environment that the robot is operating in. "Environment" here broadly refers to factors that are external to the robot, e.g., a cluttered room that a drone is navigating, disturbances such as wind gusts, or an object that a manipulator is grasping. The dynamics of the robot may be nonlinear/hybrid. We denote the robot's sensor observations (e.g., RGB-D images) by $o_t \in \mathcal{O} \subseteq \mathbb{R}^{n_o}$.

Cost functions. The robot's task is encoded via a cost function and we let $C_E(\pi)$ denote the cost incurred by a (deterministic) policy π when deployed in environment E over a finite time horizon T. The policy $\pi \in \Pi$ is a mapping from (histories of) sensor observations to actions (e.g., parameterized using a neural network). In the context of obstacle avoidance, the cost could capture how close the drone gets to an obstacle; in the context of grasping, the cost could be 0 if the robot successfully lifts the object or 1 otherwise. We assume that the cost is bounded; without further loss of generality, we assume $C_E(\pi) \in [0, 1]$. We also assume that the robot has access to the cost $C_E(\pi)$ after performing a rollout on E (i.e., at the end of an episode of length T). This is a relatively benign assumption in robotics contexts since the cost function often has physical meaning and can be measured by the robot's sensors. For example, a drone equipped with a depth sensor can measure the smallest reported depth value during its operation in an environment, and a manipulator equipped with a camera and/or force-torque sensor can measure if it successfully grasped an object. We make no further assumptions on the cost function (e.g., we *do not* assume continuity, Lipschitzness, etc.).

Training distribution. We assume that the robot has access to a training dataset $S = \{E_1, \ldots, E_m\}$ of m environments drawn i.i.d. from a training distribution $\mathcal{D}$, i.e. $S \sim \mathcal{D}^m$. Importantly, we *do not* assume any explicit knowledge of $\mathcal{D}$ or the space $\mathcal{E}$ of environments. We only assume indirect access to $\mathcal{D}$ in the form of the finite training dataset S.

Test distribution. After training, the robot is deployed on environments in $S' = \{E'_1, \ldots, E'_n\}$ drawn from a test distribution $\mathcal{D}'$: $S' \sim \mathcal{D}'^n$. Again, we do not assume explicit knowledge of $\mathcal{D}'$.

Goal: task-driven OOD detection. After being deployed in (a typically small number of) environments in S', the robot's goal is to detect if these environments were drawn from a different distribution than the training distribution (i.e., if $\mathcal{D}'$ is different from $\mathcal{D}$). Moreover, our goal is to perform *task-driven* OOD detection. In particular, our OOD detector should only trigger if:

$$C_{\mathcal{D}'}(\pi) \coloneqq \mathop{\mathbb{E}}_{E' \sim \mathcal{D}'} C_{E'}(\pi) > C_{\mathcal{D}}(\pi) \coloneqq \mathop{\mathbb{E}}_{E \sim \mathcal{D}} C_E(\pi) \ . \tag{1}$$

Thus, our OOD detector should be *insensitive* to changes in the environment distribution that do not adversely[1] impact the robot's performance. This is a challenging task since we only assume access to a finite number of environments from $\mathcal{D}$ and $\mathcal{D}'$. Moreover, our goal is to develop an OOD detection framework that is broadly applicable in challenging settings involving nonlinear/hybrid dynamics, rich sensing modalities (e.g., RGB-D), and neural network-based policies.

[1] The OOD detection scheme we present can also be modified to detect if the expected cost on $\mathcal{D}'$ is *smaller* than the expected cost on $\mathcal{D}$; however, we focus on the other case since that is of greater practical interest.

4 Approach

Our overall approach is illustrated in Fig. 1. First, we train a policy with an associated *guarantee* on the expected cost on the training distribution $\mathcal{D}$ (Sec. 4.1). We then apply our OOD detection scheme which formalizes the following intuition: violation of the bound during deployment implies (with high confidence) that the test distribution $\mathcal{D}'$ is OOD in a task-relevant manner (Sec. 4.2).

4.1 Policy training via derandomized PAC-Bayes bounds

Given a training dataset $S = \{E_1, \ldots, E_m\}$ of m environments drawn i.i.d. from the training distribution $\mathcal{D}$, our goal is to learn a policy π with a guaranteed bound on the expected cost $C_{\mathcal{D}}(\pi) := \mathbb{E}_{E\sim\mathcal{D}} C_E(\pi)$. Since our OOD detection scheme will rely on violations of the bound, it is important to obtain bounds that are as tight as possible. In this work, we utilize the *Probably Approximately Correct (PAC)-Bayes* framework [25, 26, 27] to train policies with strong guarantees. More specifically, we leverage recently-developed *derandomized* PAC-Bayes bounds [44], which are well-suited to the OOD detection setting (as we explain further below).

PAC-Bayes applies to settings where one chooses a *distribution* over policies (e.g., a distribution over weights of a neural network), and learning algorithms that have the following structure: (1) choose a *"prior"* distribution P_0 over policies *before* observing any data (this can be used to encode domain/expert knowledge); (2) obtain a training dataset S and choose a *posterior* distribution P over policies. Let P be the output of an algorithm A which takes P_0 and S as input. Denote the cost incurred by a policy π on the training environments in S as $C_S(\pi) := \frac{1}{m}\sum_{E\in S} C_E(\pi)$. The following result is our primary theoretical tool for training policies with bounds on performance.

Theorem 1 *For any distribution $\mathcal{D}$, prior distribution P_0, $\delta \in (0,1)$, cost bounded in $[0,1]$, and deterministic algorithm A which outputs the posterior distribution P, we have the following:*

$$\mathop{\mathbb{P}}_{(S,\pi)\sim(\mathcal{D}^m\times P)}\left[C_{\mathcal{D}}(\pi) \le \overline{C}_\delta(\pi, S)\right] \ge 1-\delta \ , \tag{2}$$

where $\overline{C}_\delta(\pi,S) := C_S(\pi) + \sqrt{R}$, $R := \big(D_2(P\|P_0) + \ln\frac{2\sqrt{m}}{(\delta/2)^3}\big)/(2m)$, and D_2 is the Rényi Divergence for $\alpha = 2$ defined as: $D_2(P||P_0) = \ln\big(\mathbb{E}_{\pi\sim P_0}\big[\big(\frac{P(\pi)}{P_0(\pi)}\big)^2\big]\big)$.

Proof. The proof is in App. A.1. We use [44, Theorem 2], a general pointwise PAC-Bayes bound. We perform the reduction from supervised learning to policy learning presented in [40]. □

This result allows us to obtain policies with guaranteed bounds on the expected cost. In particular, we can search for a posterior P in order to minimize the sum of the training cost and the "regularizer" $\sqrt{R}$. We describe such training methods via backpropagation and blackbox optimization below. Sampling from the resulting posterior P provides a policy with a bound on $C_{\mathcal{D}}(\pi)$ that holds with high probability (over the sampling of the training dataset S and the policy π).

Recent work has demonstrated the effectiveness of PAC-Bayes to provide strong bounds for deep neural networks [28, 31, 33] and specifically for policy learning [40, 41, 42, 43]. However, the bounds used by these approaches do not provide a viable approach for performing OOD detection. The approaches are based on traditional PAC-Bayes bounds, where a *distribution* P over policies (e.g., a distribution over neural network weights) is chosen; the resulting bound is on $\mathbb{E}_{\pi\sim P} C_{\mathcal{D}}(\pi)$ instead of $C_{\mathcal{D}}(\pi)$. Thus, given a test dataset S' of environments, many policies from the distribution P must be sampled in order to evaluate/bound the expected cost on S'. This is not feasible in an OOD detection setting, where there is single execution on the test environments. Our use of the derandomized PAC-Bayes bound in Theorem 1 avoids this issue since we can bound $C_{\mathcal{D}}(\pi)$ for a *particular* policy sampled from P.

Training with backpropagation. We provide an approach for optimizing the bound provided in Theorem 1. We work with the family of posterior distributions P defined by multivariate Gaussians with diagonal covariance $\mathrm{diag}(s)$, i.e., $P = \mathcal{N}(\mu, \mathrm{diag}(s))$. Further, let $\psi := (\mu, \log s)$; we use the shorthand $\mathcal{N}_\psi$ for $\mathcal{N}(\mu, \mathrm{diag}(s))$. We denote π_w with weights $w \sim \mathcal{N}_\psi$ as a parameterization of the robot's policy (e.g., neural networks with weights w). We also choose a Gaussian as our prior P_0. We then optimize ψ to minimize the PAC-Bayes bound in Theorem 1 via backpropagation. This is possible because the Renyi divergence can be computed in closed-form for multivariate Gaussians; see App. A.4 for details. After training, we sample a w from the trained posterior and fix this for deployment on test environments. Then, we compute the PAC-Bayes upper bound $\overline{C}_\delta(\pi, S)$.

Training with evolutionary strategies. To train robot control policies in settings where backpropagation is not feasible (e.g., presence of a "blackbox" in the form of a simulator or robot hardware in

the forward pass), we use Evolutionary Strategies (ES) [45] which is a class of blackbox optimizers. ES addresses this challenge by estimating the gradient via a Monte-Carlo estimator:

$$\nabla_\psi C_S(\mathcal{N}_\psi) := \frac{1}{m}\sum_{E\in S}\nabla_\psi \mathop{\mathbb{E}}_{w\sim\mathcal{N}_\psi}[C_E(\pi_w)] = \frac{1}{m}\sum_{E\in S}\mathop{\mathbb{E}}_{w\sim\mathcal{N}_\psi}[C_E(\pi_w)\nabla_\psi \ln \mathcal{N}_\psi(w)] \ . \tag{3}$$

Although we can compute the gradient of the regularizer analytically (as mentioned above), using different methods to estimate the gradient of the empirical cost (ES) and the gradient of the regularizer (analytically) results in poor convergence. To alleviate this, we estimate the regularizer's gradient using ES as well by leveraging the expectation form of Rényi divergence in Theorem 1. Details for estimating the gradient of the PAC-Bayes upper bound with ES are provided in App. A.5.

4.2 Task-driven OOD detection with statistical guarantees

We now tackle the problem of OOD detection as defined in Sec. 3. The PAC-Bayes training pipeline from Sec. 4.1 produces a policy π and an associated bound $\overline{C}_\delta(\pi, S)$ on the expected cost $C_\mathcal{D}(\pi)$ that holds with probability $1-\delta$ over the sampling of the training dataset $S \sim \mathcal{D}^m$ and the policy $\pi \sim P$. Our key idea for OOD detection is that if our PAC bound $\overline{C}_\delta(\pi, S)$ is violated by π in the test environments S' (drawn from the test distribution $\mathcal{D}'$), then this indicates that the test environments are OOD. We formalize this intuition below using two popular frequentist statistical inference tools — hypothesis testing via p-value and confidence interval overlap.

Method 1: Hypothesis testing

In accordance with our notion of task-driven OOD detection (1) described in Sec. 3, let H_0 be the null-hypothesis which claims that $C_{\mathcal{D}'}(\pi) \leq C_\mathcal{D}(\pi)$ and let H_1 be the alternate hypothesis which claims that $C_{\mathcal{D}'}(\pi) > C_\mathcal{D}(\pi)$ for a given policy π. Then, to perform a hypothesis test, we compute the p-value and check if it drops below a significance level $\alpha \in (0,1)$, which is chosen before looking at the data. The p-value bounds the probability of drawing test environments with average cost larger than the observed test cost $C_{S'}(\pi)$, assuming that the null-hypothesis holds. In the event that the p-value is smaller than α, we can conclude that under the null-hypothesis the observed test dataset S' had a very small probability of being drawn; therefore, the null-hypothesis H_0 can be rejected. A mathematically precise definition of the p-value is given as follows.

Definition 1 (adapted from [46]) *Let $\mathcal{D}'$ be the test distribution and $S' \sim \mathcal{D}'^n$ be an observed dataset. Let π be the robot's control policy. Then, the p-value is defined as:*

$$p(S') := \mathop{\mathbb{P}}_{\hat{S}\sim\mathcal{D}'^n}[C_{\hat{S}}(\pi) \geq C_{S'}(\pi) \mid H_0] \ . \tag{4}$$

We present an upper bound on the p-value by leveraging the PAC-Bayes generalization bound (Theorem 1). This upper bound holds with probability $1-\delta$ (over the sampling of S and π).

Theorem 2 *Let $\mathcal{D}$ be the training distribution and P be the posterior distribution on the space of policies obtained through the training procedure described in Sec. 4.1. Let $S' \sim \mathcal{D}'^n$ be a test dataset, $p(S')$ be the p-value for S' defined in Definition 1, and $\delta \in (0,1)$. Then,*

$$\mathop{\mathbb{P}}_{(S,\pi)\sim(\mathcal{D}^m\times P)}[p(S') \leq \exp(-2n\overline{\tau}(S)^2)] \geq 1-\delta \ , \tag{5}$$

where $\overline{\tau}(S) := \max\{C_{S'}(\pi) - \overline{C}_\delta(\pi, S), 0\}$.

Proof. The proof relies on Hoeffding's inequality and is provided in App. A.2. □

Method 2: Confidence interval on the difference in expected train and test costs

We now present another method for performing task-driven OOD detection (Sec. 3 and Eq. (1)) by lower bounding $C_{\mathcal{D}'}(\pi) - C_\mathcal{D}(\pi)$ with high probability. If the lower bound is positive, then with high confidence we can conclude that the robot is operating in (task-relevant) OOD environments as the test cost is larger than the train cost. Furthermore, this lower bound serves as a measure of how "far" the two distributions are from the perspective of the task.

Theorem 3 *Let $\mathcal{D}$ be the training distribution, $\mathcal{D}'$ be the test distribution, and P be the posterior distribution on the space of policies obtained through the training procedure described in Sec. 4.1. Let $\delta, \delta' \in (0,1)$ such that $\delta + \delta' < 1$, $\gamma := \sqrt{\frac{\ln(1/\delta')}{2n}}$, and $\Delta C := C_{S'}(\pi) - \gamma - \overline{C}_\delta(\pi, S)$. Then,*

$$\mathop{\mathbb{P}}_{(S,\pi,S')\sim(\mathcal{D}^m\times P\times\mathcal{D}'^n)}[C_{\mathcal{D}'}(\pi) - C_\mathcal{D}(\pi) \geq \Delta C] \geq 1-\delta-\delta' \ . \tag{6}$$

Proof. A detailed proof of this theorem is provided in App. A.3. □

To use Theorem 3 for detecting if a test dataset S' is OOD, we first choose δ and δ' to obtain a desired confidence level $1 - \delta - \delta'$ for our OOD detector. Using these constants and the policy π drawn from P trained using the PAC-Bayes upper bound, we then compute ΔC. If $\Delta C > 0$, then with confidence at least $1 - \delta - \delta'$ we can claim that $C_{\mathcal{D}'} > C_D$, and therefore, with the same level of confidence the test dataset S' must be OOD according to our task-relevant notion in (1). Furthermore, we obtain a guaranteed upper bound of $\delta + \delta'$ on the false positive rate of our OOD detector, i.e., the rate at which our detector will misclassify training distribution data as OOD.

5 Examples

We demonstrate the ability of our approach to perform task-driven OOD detection with guaranteed confidence bounds on two simulated examples: a manipulator grasping a new set of objects and a drone navigating a new set of environments. For the navigation task, we compare our methods with popular OOD detection baselines. Our code is provided in the supplementary material.

5.1 Robotic grasping

Overview. We use the Franka Panda arm (Fig. 2(a)) for grasping objects in the PyBullet simulator [47] and build upon the open-source code provided in [42]. The robot employs a vision-based control policy that uses a depth map of the object obtained from an overhead camera and returns an open-loop action $a := (x, y, z, \theta)$ which corresponds to the desired grasp position and yaw orientation of the gripper. We train the manipulator to grasp mugs placed in $SE(2)$ poses drawn from a specific distribution. Then, we demonstrate the efficacy of our OOD detection framework by (i) gradually modifying the distribution on the mug poses and (ii) changing the objects from mugs to bowls.

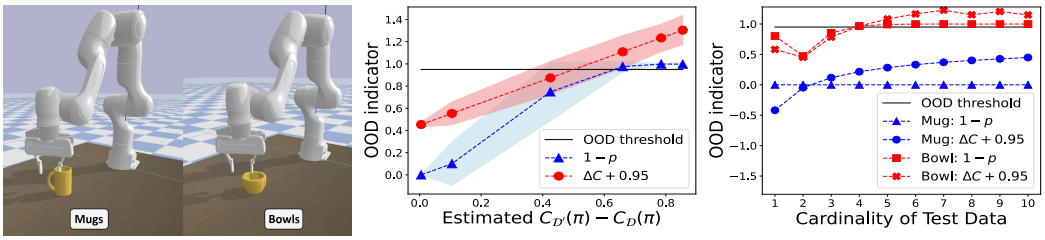
Control policy. The control policy is a deep neural network (DNN) which inputs a 128×128 depth map of the object and a latent state $z \in \mathbb{R}^{10}$ sampled from a multivariate Gaussian distribution $\mathcal{N}_\psi$ with a diagonal covariance, and outputs an open-loop grasp action a; see Fig. A.1 in App. A.6.1 for the policy. In [42], the distribution $\mathcal{N}_\psi$ on the latent space encodes prior domain/expert knowledge.

Training. Mugs from the ShapeNet dataset [48] are randomly scaled in all dimensions to generate a training dataset S of 500 mugs. These mugs are then placed upright on the table in random positions sampled from the uniform distribution $\mathcal{U}([0.45 \text{ cm}, 0.55 \text{ cm}] \times [-0.05 \text{ cm}, 0.05 \text{ cm}])$ and random yaw orientations sampled from the uniform distribution $\mathcal{U}([-\pi \text{ rad}, \pi \text{ rad}])$. If the robot is able to lift the mug by 10 cm and the gripper palm does not contact it, then we consider the rollout successful and assign a cost of 0; otherwise the cost is set to 1. In training, we optimize the distribution $\mathcal{N}_\psi$ on the latent space to minimize the PAC-Bayes upper bound provided in Theorem 1 using ES, as described in Sec. 4.1, while the weights of the CNN and MLP networks in Fig. A.1 in App. A.6.1 remain fixed. The prior $\mathcal{N}_{\psi_0}$ is chosen as the normal distribution with zero mean and identity covariance. The weights of the DNN are borrowed from [42] and the training is warm-started with the posterior provided in [42, App. A5.1]. A policy π is sampled from the trained posterior $\mathcal{N}_\psi$ and the PAC-Bayes bound for this policy is computed as $\overline{C}_\delta(\pi, S) = 0.1$ with $\delta = 0.01$.

OOD detection results. We perform OOD detection using the two methods presented in Theorem 2 and Theorem 3. For detection with p-value, we choose a significance level $\alpha = 95\%$, while, for detection using ΔC we choose a confidence level of 95%, i.e., $\delta + \delta' = 0.05$, which ensures that the false-positive rate of our detector is no greater than 5%. We perform two experiments to demonstrate the efficacy of our approach for OOD detection. First, we make the distribution on the mug's initial placement progressively more challenging; see App. A.6.1 for the exact distributions. For each distribution, we sample 20 test datasets of cardinality 10 and compute our OOD indicators: (i) the lower bound on $1 - p$ (where p is the p-value) and (ii) ΔC using Theorem 3. Fig. 2(b) plots the mean (dashed line) and a one standard deviation spread (shaded region) for the OOD indicators as a function of $C_{\mathcal{D}'} - C_{\mathcal{D}}$ (estimated via exhaustive sampling). Note that we plot $\Delta C + 0.95$ so that the OOD threshold is the same (0.95) for both methods. As the cost of the policy deteriorates on test distributions our OOD indicators reliably increase, capturing the shift of the test distributions away from the training distribution. In the second experiment, we change the objects that the manipulator must grasp from mugs to bowls. Fig. 2(c) shows that with a small test dataset S' of cardinality 5, both our approaches detect OOD when bowls are used (red curves). As expected, our OOD detectors are not triggered for mugs (blue curves), which are drawn from the training distribution.

5.2 Vision-based obstacle avoidance with a drone

Overview. In both the simulation and hardware portions of this example, we aim to avoid an obstacle field with the Parrot Swing drone; this is an agile quadrotor/fixed-wing hybrid drone shown in Fig. 1. We train a DNN control policy in a simulation setup based on the hardware system shown in Fig. 1.



(a) (b) (c)

Figure 2: OOD detection for grasping. **(a)** Franka Panda arm in PyBullet grasping a mug (left) and a bowl (right). **(b)** Performance of our OOD detectors for different distributions on mug placement. **(c)** Comparison of our OOD detectors for grasping mugs and grasping bowls.

The policy takes in a 50×50 depth image and outputs a softmax corresponding to a set of pre-computed motion primitives with the goal of avoiding obstacles by the largest distance. Since we designed the simulation portion of this example with application to hardware in mind, we have created motion primitives by capturing (with a Vicon motion tracking system) the trajectories of open-loop control inputs, which result in different maneuvers; the two images in Fig. 1 represent two of these trajectories. The use of motion primitives results in straightforward training (described below), and also allows us to perform accurate sim-to-real transfer (since the motion primitives are recorded directly from the hardware system).

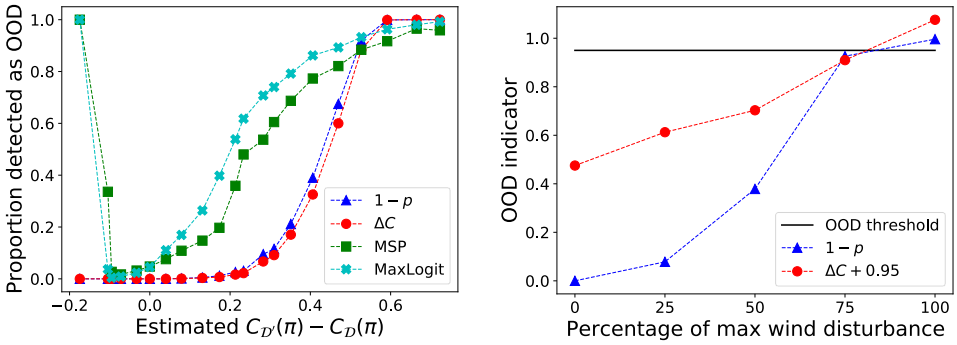
Training. We generate environments by randomly placing a set of cylindrical obstacles whose locations are sampled from the uniform distribution $\mathcal{U}([4.5\text{ m}, 7\text{ m}] \times [-3.5\text{ m}, 3.5\text{ m}])$ relative to the drone's starting point. In order to determine a cost $C_E(\pi)$ associated with a rollout on an environment, we record the minimum distance d_{min} from the obstacles (as recorded by the robot's 120° field of view depth sensor) and assign a cost of $\max(0, 1 - \frac{d_{\text{min}}}{30\text{cm}})$. Given the size of the Parrot Swing (≈ 40 cm), a cost of 0 can be achieved by obstacle gaps of 1 m or larger. Training environments have 9 obstacles and have at least one gap which is wide enough to navigate through. We train the prior $\mathcal{N}_{\psi_0}$ over policies by transforming the problem into a supervised learning setting. For each of $10,000$ training environment in S, we compute the minimum distance achieved by each primitive in simulation. The softmax of the resulting distances is used to create a label for each input depth map. We then use the cross-entropy loss for training. The result is a policy trained to assign larger values to motion primitives which achieve a larger distance from obstacles. We use another $10,000$ environments to train the posterior distribution using Algorithm A.1. We sample a policy π from the trained posterior and compute the PAC-Bayes bound $\overline{C}_\delta(\pi, S) = 0.222$ for $\delta = 0.01$.

Simulation Results

We compare our task-driven OOD detection approach with two baselines: (i) maximum softmax probability (MSP) [3] (an effective and popular baseline for OOD detection), and (ii) MaxLogit [5] (a recent state-of-the-art OOD detection baseline). We note that these baselines are specifically designed for networks which output categorical distributions, and thus provide strong benchmarks. We determine the threshold for detection for the baselines by choosing a false positive rate of 5% for the training environments. However, note that the baselines do not provide any guarantees; they may violate the false positive rate even on new environments drawn from the training distribution (as we observe in our experiments). We select a p-value of 0.05 as the indication that an environment is OOD, and a guaranteed false-positive rate of $\delta + \delta' = 5\%$ for the confidence interval method. Each method receives a dataset S' of 10 environments. We use the average outputs of the baselines on the 10 environments for OOD detection. We generate OOD environments of varying difficulty by changing the number of obstacles and the maximum or minimum gap-size between obstacles.

Varied environment difficulty. We sample 2000 datasets S' (of size 10) for each setting and plot the proportion of these detected as OOD for all methods in Fig. 3(a). At 0 on the x-axis of this plot, we draw datasets from the training distribution. All other environments are OOD. We note that the p-value and confidence interval-based methods perform similarly. The plot demonstrates our ability to perform task-driven OOD detection. In particular, we only detect the "harder" distributions (i.e., environments with higher increased cost) as OOD, while the baselines are triggered even for the easier distributions. In addition, neither baseline provides any guarantees and both violate the 5% false-positive rate on the training distribution. Our methods have a low false positive rate, albeit at the cost of conservatism for moderate distribution shifts.

Task-irrelevant shift. We also compare the baselines with our methods on a distribution where environments consist of 4 (uniformly) randomly located obstacles. In this setting, the control policy



(a) (b)

Figure 3: OOD Detection on settings of variable difficulty. **(a)** Comparison of the performance of our OOD detectors with baselines MSP [3] and MaxLogit [5]. **(b)** Comparison of our OOD detectors on the Parrot Swing hardware for increasing wind disturbance.

achieves a near-identical expected cost $C_{\mathcal{D}'}(\pi)$ (as estimated by exhaustive sampling of environments) to the expected training cost $C_{\mathcal{D}}(\pi)$ (in particular, $C_{\mathcal{D}'}(\pi)$ - $C_{\mathcal{D}}(\pi) = -0.02$). For this setting, MSP [3] classified 95.7% of test datasets as OOD and MaxLogit [5] classified 50.9% as OOD. Thus, the baselines are triggered by a task-irrelevant shift in the distribution. In contrast, our methods had an OOD detection rate of 0.1% in this setting.

Hardware Results

We use a Parrot Swing drone for the hardware experiments (Fig. 1). We simulate a depth sensor for the drone (as if the sensor was mounted on the drone) by generating a synthetic depth image using the positions of objects from the Vicon motion capture system. We do not provide any other information to the policy (e.g., obstacle positions). We generate each environment the same way as in simulation, and then place the real-world obstacles in the generated locations.

Varied environment difficulty and wind disturbances. We deploy the policy trained in simulation on three kinds of OOD environments in hardware: (i) environments with a smaller number of obstacles ("easier" environments), (ii) environments with smaller gaps between obstacles (i.e., "harder" environments), and (iii) environments with wind generated using a fan (Fig. 1 right) with the same obstacle distribution as training. Videos from each setting are provided in the supplementary materials. For each setting, we run 10 trials on the hardware and use this for OOD detection. As expected, our OOD detectors are not triggered by the easier environments. For the harder environments, we compute $1 - p \geq 0.81$ and $\Delta C = -0.11$. Results from the windy environments are shown in Fig. 3(b) for increasing values of wind (up to about $5\ \mathrm{m/s}$). We note that the sim-to-real distribution shift (corresponding to the zero wind case) is not viewed as being OOD in a task-relevant manner by our approaches. Both our approaches assign increasingly high confidence as the wind speed is increased. We note that disturbances such as wind cannot be detected via the depth image given to the robot's policy. Thus any OOD detection technique which relies solely on the output of the policy, such as MSP [3] and MaxLogit [5], would be unable to detect these environments as OOD.

6 Conclusion

We have presented a framework for performing task-driven OOD detection with statistical guarantees. Our approach uses PAC-Bayes theory to train a policy with a bound on the expected cost on the training distribution. We then perform OOD detection on test environments by checking for violations of the bound (using p-values and concentration inequalities). Our simulated and hardware experiments demonstrate the ability of our approach to perform OOD detection within a handful of trials. Comparisons with baselines also demonstrate two advantages: our OOD detectors (i) are only sensitive to task-relevant distribution shifts, and (ii) provide statistical guarantees on detection.

Challenges and future work. The approach we present here allows us to bound the false positive rate. An interesting theoretical question is whether we can also bound the false negative rate. It would also be of practical interest to extend our approach to settings where the robot encounters environments in an *online* manner (instead of the batch setting we consider here). Another particularly exciting direction is to develop versions of our approach that are more *proactive*; instead of having to incur costs on the test environments, one could potentially perform OOD detection based on *predicted* costs (thus avoiding the need to potentially fail on the test environments).

References

[1] M. Basseville. Detecting Changes in Signals and Systems – A Survey. *Automatica*, 24:309–326, 1988.

[2] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller. A Unifying Review of Deep and Shallow Anomaly Detection. *Proceedings of the IEEE*, 109(5):756–795, 2021.

[3] D. Hendrycks and K. Gimpel. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. *Proceedings of the International Conference on Learning Representations*, 2017.

[4] S. Liang, Y. Li, and R. Srikant. Enhancing the Reliability of Out-of-Distribution Image Detection in Neural Networks. *Proceedings of the International Conference on Learning Representations*, 2018.

[5] D. Hendrycks, S. Basart, M. Mazeika, M. Mostajabi, J. Steinhardt, and D. Song. Scaling Out-of-Distribution Detection for Real-World Settings. *arXiv preprint arXiv:1911.11132*, 2020.

[6] T. DeVries and G. W. Taylor. Learning Confidence for Out-of-Distribution Detection in Neural Networks. *arXiv preprint arXiv:1802.04865*, 2018.

[7] K. Lee, H. Lee, K. Lee, and J. Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *Proceedings of the International Conference on Learning Representations*, 2018.

[8] D. Hendrycks, M. Mazeika, and T. G. Dietterich. Deep Anomaly Detection with Outlier Exposure. *Proceedings of the International Conference on Learning Representations*, 2019.

[9] J. Chen, Y. Li, X. Wu, Y. Liang, and S. Jha. Robust Out-of-distribution Detection for Neural Networks. *arXiv preprint arXiv:2003.09711*, 2020.

[10] K. Lee, K. Lee, H. Lee, and J. Shin. A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks. *Proceedings of the Conference on Advances in Neural Information Processing Systems*, 2018.

[11] J. Bitterwolf, A. Meinke, and M. Hein. Certifiably Adversarially Robust Detection of Out-of-Distribution Data. *Proceedings of the Conference on Advances in Neural Information Processing Systems*, 2020.

[12] A. Meinke and M. Hein. Towards Neural Networks That Provably Know When They Don't Know. *Proceedings of the International Conference on Learning Representations*, 2020.

[13] A. Meinke, J. Bitterwolf, and M. Hein. Provably Robust Detection of Out-of-distribution Data (almost) for free. *arXiv preprint arXiv:2106.04260*, 2021.

[14] M. A. Siddiqui, A. Fern, T. G. Dietterich, and S. Das. Finite Sample Complexity of Rare Pattern Anomaly Detection. *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2016.

[15] S. Liu, R. Garrepalli, T. G. Dietterich, A. Fern, and D. Hendrycks. Open Category Detection with PAC Guarantees. *Proceedings of the International Conference on Machine Learning*, 2018.

[16] S. Bates, E. Candès, L. Lei, Y. Romano, and M. Sesia. Testing for Outliers with Conformal p-values. *arXiv preprint arXiv:2104.08279*, 2021.

[17] A. Sharma, N. Azizan, and M. Pavone. Sketching Curvature for Efficient Out-of-Distribution Detection for Deep Neural Networks. *arXiv preprint arXiv:2102.12567*, 2021.

[18] A. Sedlmeier, T. Gabor, T. Phan, L. Belzner, and C. Linnhoff-Popien. Uncertainty-Based Out-of-Distribution Classification in Deep Reinforcement Learning. *arXiv preprint arXiv:2001.00496*, 2019.

[19] A. Sedlmeier, R. Müller, S. Illium, and C. Linnhoff-Popien. Policy Entropy for Out-of-Distribution Classification. *Artificial Neural Networks and Machine Learning – ICANN*, 2020.

[20] F. Cai and X. Koutsoukos. Real-time Out-of-distribution Detection in Learning-Enabled Cyber-Physical Systems. *Proceedings of the International Conference on Cyber-Physical Systems*, 2020.

[21] T. Wu and J. Ortiz. RLAD: Time Series Anomaly Detection through Reinforcement Learning and Active Learning. *arXiv preprint arXiv:2104.00543*, 2021.

[22] I. Greenberg and S. Mannor. Detecting Rewards Deterioration in Episodic Reinforcement Learning. *Proceedings of the International Conference on Machine Learning*, 2021.

[23] V. N. Vapnik and A. Y. Chervonenkis. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Dokl. Akad. Nauk*, 181(4), 1968.

[24] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[25] J. Shawe-Taylor and R. C. Williamson. A PAC Analysis of a Bayesian Estimator. *Proceedings of the Conference on Computational Learning Theory*, 1997.

[26] D. A. McAllester. Some PAC-Bayesian theorems. *Machine Learning*, 37(3):355–363, 1999.

[27] M. Seeger. PAC-Bayesian Generalisation Error Bounds for Gaussian Process Classification. *Journal of Machine Learning Research*, 3(Oct):233–269, 2002.

[28] G. K. Dziugaite and D. M. Roy. Computing Nonvacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data. *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2017.

[29] J. Langford and J. Shawe-Taylor. PAC-Bayes & margins. *Proceedings of the Conference on Advances in Neural Information Processing Systems*, 2003.

[30] P. Germain, A. Lacasse, F. Laviolette, and M. Marchand. PAC-Bayesian Learning of Linear Classifiers. In *Proceedings of the International Conference on Machine Learning*, pages 353–360. ACM, 2009.

[31] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky. Spectrally-Normalized Margin Bounds for Neural Networks. In *Advances in Neural Information Processing Systems*, pages 6240–6249, 2017.

[32] Y. Jiang, B. Neyshabur, H. Mobahi, D. Krishnan, and S. Bengio. Fantastic Generalization Measures and Where to Find Them. *Proceedings of the International Conference on Learning Representations*, 2020.

[33] M. Pérez-Ortiz, O. Rivasplata, J. Shawe-Taylor, and C. Szepesvári. Tighter Risk Certificates for Neural Networks. *arXiv preprint arXiv:2007.12911*, 2020.

[34] O. Catoni. *Statistical Learning Theory and Stochastic Optimization*. École d'Été de Probabilités de Saint-Flour 2001. Springer, 2004.

[35] O. Catoni. *PAC-Bayesian Supervised Classification: The Thermodynamics of Statistical Learning*, volume 56 of *Lecture notes - Monograph Series*. Institute of Mathematical Statistics, 2007.

[36] D. McAllester. A PAC-Bayesian Tutorial with A Dropout Bound. *arXiv preprint arXiv:1307.2118*, 2013.

[37] O. Rivasplata, V. M. Tankasali, and C. Szepesvari. PAC-Bayes with Backprop. *arXiv preprint arXiv:1908.07380*, 2019.

[38] N. Thiemann, C. Igel, O. Wintenberger, and Y. Seldin. A Strongly Quasiconvex PAC-Bayesian Bound. *Machine Learning Research*, 76:1–26, 2017.

[39] G. K. Dziugaite and D. M. Roy. Data-dependent PAC-Bayes priors via differential privacy. *Proceedings of the Conference on Advances in Neural Information Processing Systems*, 2018.

[40] A. Majumdar and M. Goldstein. PAC-Bayes Control: Synthesizing Controllers that Provably Generalize to Novel Environments. *Proceedings of the Conference on Robot Learning*, 87: 293–305, 2018.

[41] S. Veer and A. Majumdar. Probably Approximately Correct Vision-Based Planning using Motion Primitives. *Proceedings of the Conference on Robot Learning*, 2020.

[42] A. Z. Ren, S. Veer, and A. Majumdar. Generalization Guarantees for Imitation Learning. *Proceedings of the Conference on Robot Learning*, 2020.

[43] A. Majumdar, A. Farid, and A. Sonor. PAC-Bayes control: learning policies that provably generalize to novel environments. *The International Journal of Robotics Research*, 40:574–593, 2021.

[44] P. Viallard, P. Germain, A. Habrard, and E. Morvant. A General Framework for the Derandomization of PAC-Bayesian Bounds. *arXiv preprint arXiv:2102.08649*, 2021.

[45] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(27):949–980, 2014.

[46] R. V. Hogg, J. McKean, and A. T. Craig. *Introduction to Mathematical Statistics*. Pearson Education, 2005.

[47] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2018.

[48] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[49] A. Maurer. A Note on the PAC Bayesian Theorem. *arXiv preprint cs.LG/0411099*, 2004.

[50] M. Gil, F. Alajaji, and T. Linder. Rényi divergence measures for commonly used univariate continuous distributions. *Information Sciences*, 249:124–131, 2013.