Uncertainty-Aware Failure Detection for Imitation Learning Robot Policies

Chen Xu[1]*, **Tony Nguyen**[1], **Patrick Miller**[1], **Robert Lee**[2], **Paarth Shah**[1], **Rares Ambrus**[1], **Haruki Nishimura**[1], **Masha Itkina**[1]

[1]**Toyota Research Institute (TRI),** [2]**Woven by Toyota**

Abstract: Recent years have witnessed impressive robotic manipulation systems driven by advances in imitation learning and generative modeling, such as diffusion- and flow-based approaches. However, these systems can still fail due to suboptimality, inconsistency of stochastic actions, or unfavorable out-of-distribution operating conditions. To ensure dependable operation of such systems before deployment in safety-critical situations, such as in human environments, we need to reliably detect their failures in real-time during inference. In this paper, we propose a modular two-stage approach for failure detection in imitation learning-based robotic manipulation tasks. Our method combines extracting scalar signals that correlate with policy failures and conformal prediction to accurately identify failures while providing statistical guarantees. We investigate both learned and posthoc scalar signal candidates, finding learned signals to be most performant for failure detection. We show the effectiveness of our approach through extensive experiments on diverse robotic manipulation tasks, showcasing its ability to detect failures accurately and quickly. Our results highlight the potential of our method to enhance the safety and reliability of imitation learning-based robotic systems as they continue to improve and become ready for real-world deployment.

Keywords: Failure/OOD detection, runtime monitoring, imitation learning

1 Introduction

Robotic manipulation plays a crucial role in many applications, such as manufacturing, logistics, and healthcare [1]. Recently, imitation learning algorithms have shown tremendous success in learning complex manipulation skills from demonstrations using stochastic generative modeling, such as diffusion- [2, 3] and flow-based methods [4]. However, despite their outstanding results, policy networks can fail due to poor stochastic sampling from the action distribution. Additionally, the models may encounter out-of-distribution (OOD) conditions where the input observations deviate significantly from the training data distribution. In such cases, the generated actions may be unreliable or even dangerous. Therefore, it is imperative to detect these failures as quickly as possible once they occur to ensure the safety and reliability of the robotic system.

Detecting failures in robotic manipulation tasks poses several challenges. First, the input data for failure detection, such as environment observations, are often high-dimensional and have complicated distributions. This makes it difficult to identify discriminative features that distinguish between successful and failed executions, especially in the imitation learning setting where a reward function is not defined. Second, there are countless opportunities for failure due to the complex nature of manipulation tasks and the wide range of possible environmental conditions (see Fig. 2). Consequently, failure detectors need to be general and robust to handle diverse failure scenarios.

In prior work, failure detection is often achieved through binary classification of in-distribution (ID) and OOD conditions [5, 6]. Training data for imitation learning naturally consists of successful

*This work was done during an internship at the Toyota Research Institute.

Workshop on Safe and Robust Robot Learning for Operation in the Real World (SAFE-ROL) at CoRL 2024.

trajectories only, making any failure OOD. We follow this general strategy in this work. However, prior methods often require OOD data for training, which poses a significant challenge as collecting and annotating a diverse set of failure examples can be time-consuming, expensive, and even infeasible in many real-world scenarios. Moreover, the classifiers trained on a specific set of OOD data may not generalize well to unseen failure modes. Hence, there is a need for failure detection approaches that operate without relying on explicit OOD data during training.

To address these challenges, we propose a two-stage approach (Fig. 1) for failure detection in generative imitation-learning policies for manipulation. **In the first stage**, we extract scalar signals from input data during inference that are discriminative between successes and failures. These signals compress high-dimensional, complex input data into a scalar form that captures the essential characteristics of successful rollouts. We investigate both learned and posthoc signal candidates, finding learned signals to be most accurate for failure detection. A key novelty of our method is the ability to learn failure detection signals without access to failure data. Our learned detection score candidates are output by models that are trained offline, resulting in faster inference than batch sampling of robot action predictions as in prior work [7]. **In the second stage**, we use conformal prediction (CP) to build time-varying thresholds to sequentially determine whether a signal indicates a failure. CP provides statistical guarantees on model predictions [8]. By incorporating CP into our pipeline, we obtain reliable and adaptive thresholds that adjust to the changing dynamics of the manipulation task. Unlike prior work [7], we adopt functional CP [9] that yields a time-varying prediction band that is more suitable for dynamic time-series data.

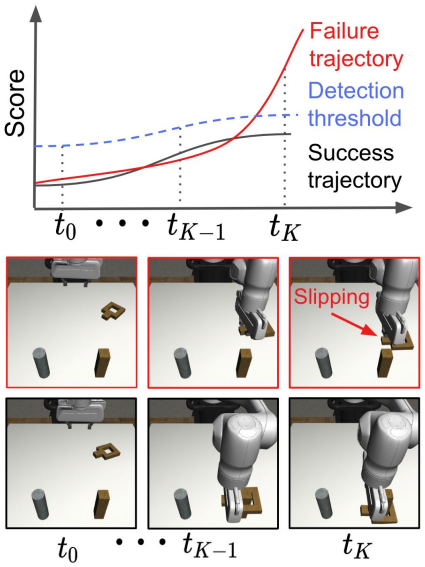


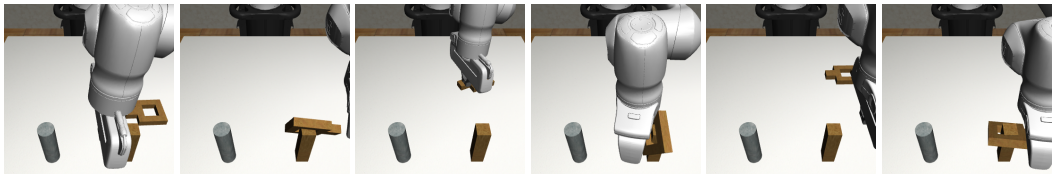
Figure 1: Proposed failure detection framework. **(Top - Illustration)** Failure detection scores are computed for each rolled out trajectory, and a detection threshold is determined using a conformal prediction band. Failure at time t_K causes the score to spike above the threshold, triggering failure detection. **(Bottom)** The robot behaves normally while attempting to place square on the peg until a failure occurs at t_K, as shown in the top right image, where the square slips out of the gripper. This failure is captured by the spike in the failure score in the plot.

Our contributions are as follows. We propose a modular two-stage framework for failure detection in generative imitation learning-based robotic manipulation. We propose a means to learn scalar signals from successful ID policy rollouts that during inference distinguish between successes and failures. We combine this scalar score with temporally-adaptive thresholds from functional CP that trigger failure detection with statistical guarantees. Our framework is flexible to incorporate new signal and threshold designs. We show that our approach detects failures accurately and quickly on diverse robotic manipulation tasks, both in simulation and in real-world settings.

2 Related Work

Imitation Learning for Robotic Manipulation. Imitation learning has emerged as a powerful paradigm for teaching robots complex skills by learning from expert demonstrations. Diffusion models [10] have shown promise for imitation learning in robotics. These models learn to denoise trajectories sampled from a Gaussian distribution, effectively capturing the multi-modal action distributions often present in demonstrations [11, 2]. Diffusion models have been used to learn observation-conditioned policies [2, 12], integrate semantic information via language conditioning [11, 13, 14], and improve robustness and generalization [15, 16, 17]. More recently, flow-based generative models have been proposed as an alternative to diffusion in imitation learning, demonstrating faster inference [18] and flexibility beyond Gaussian priors [4].

Out-of-Distribution (OOD) Detection. The task of detecting robotic failures can be generally viewed as anomaly detection, which falls under the broader framework of OOD detection [19]. One



(a) Square misplaced on peg. (b) Tilted upward on peg. (c) Slipped out of gripper. (d) Tilted downward on peg. (e) Square not picked up. (f) Tilted slightly (almost success).

Figure 2: Diverse failure types observed for a single trained policy g on a simple pick-and-place task (put square on peg). These failures occurred at different times across multiple rollouts and include the square slipping out of the gripper or being misplaced (e.g., with tilted position) on the peg.

approach is to frame OOD detection as a binary classification problem, where the OOD class is assigned a label of one. This formulation learns the decision boundary between ID and OOD data by training a binary classifier [5, 6], but requires OOD data during training. Meanwhile, density-based approaches attempt to model the distribution of ID data [20, 21], alleviating the need for OOD data during training. However, these methods can be more computationally expensive and harder to optimize compared to classification-based approaches. Similarly, control-theoretic approaches like [22] learn contrastive energy-based models to detect OOD states from ID data only, but they often require a representation of the dynamics model. Evidential deep learning methods [23, 24, 25] learn parameters for second-order distributions (e.g., Dirichlet) to approximate and decouple epistemic uncertainty from aleatoric uncertainty. Lastly, distance-based approaches [26, 27, 28] directly identify OOD samples by computing their distance to ID ones, where they avoid the need for training but tend to have limited performance compared to other approaches.

Failure Detection in Robotics. Detecting failures in robotic systems is important for ensuring safety and reliability, as failures can lead to undesired behaviors in human environments [29, 30]. Various approaches have been proposed, such as building fast anomaly classifiers based on LLM embeddings [31], constructing failure classifiers using BC-RNN policy backbones [32], and using the reconstruction error from variational autoencoders to detect anomalies in mobile manipulation [33]. In addition, Ren et al. [34] provide uncertainty sets for actions generated by LLM-based planners, prompting human intervention when the set is ambiguous. However, these works do not focus on generative imitation learning policies, which are our primary interest. For diffusion-based policies, [35] uses CP to provide a set in the trajectory space that is guaranteed to contain diffusion model outputs with a user-defined probability. [36] proposes using random network distillation (RND) to detect OOD trajectories and select reliable ones. Neither work directly considers within-rollout failure detection, and our two-stage framework combines the advantages of both approaches. Recently, Agia et al. [7] use a statistical temporal action consistency (STAC) measure in conjunction with vision-language models (VLMs) to detect failures within rollouts. STAC is the closest SOTA method for our setting, and we demonstrate improved empirical performance against it as we combine learned scores with a time-varying CP band.

3 Problem setup

Our focus in this work is to detect when a generative imitation learning policy fails to complete its task during execution. We define the following notation. Let $g(A_t \mid O_t)$ denote the generator, where O_t represents the observation of the environment (e.g., image features and robot states) at time t, and g is a stochastic predictor of a sequence of actions $A_t = (A_{t|t}, A_{t+1|t}, \ldots, A_{t+H-1|t})$ for the next H time steps. The first $H' < H$ actions $A_{t:t+H'|t}$ are extracted and executed, after which the robot replans by generating a new sequence of H actions at time $t+H'$. Recent works have trained effective generators g via diffusion [2] and flow matching (FM) [4]. Given a new initial condition O_0, the generator g outputs a trajectory $\tau_t = (O_0, A_0, O_{H'}, A_{H'}, \ldots, O_t, A_t)$ after $t = kH'$ time steps of execution for $k \geq 1$. Failure detection can thus be framed as designing a score $D(\tau_t; \theta) \to \{0, 1\}$ with parameters θ, which takes in the current trajectory and makes a decision. If $D(\tau_t; \theta) = 1$, the rollout is flagged as a failure at time step t. For instance, in a pick-and-place task, a failure may be detected after the robot fails to pick up the object or misses the target position.

4 Failure Detection Framework

Given action-observation data (A_t, O_t), we propose a two-stage framework:

1. Train a scalar score $D_M(A_t, O_t; \theta) \to \mathbb{R}$ ($_M$ for "method" for the score) on action and/or observation pairs from successful trajectories only.
2. Calibrate time-varying thresholds η_t based on a CP band.

The final score $D(\tau_t; \theta) = \mathbb{1}(D_M(A_t, O_t; \theta) > \eta_t)$ raises a failure flag if the scalar score D_M exceeds the threshold η_t at time step t. This two-stage framework allows for flexibility to incorporate new scores in stage 1 or new thresholds in stage 2. See Fig. 1 for an overview of the framework.

4.1 Design of Scalar Scores

When designing a scalar score that is indicative of policy failure, we consider the following desiderata: **(1) One-class**: The method should not require failure data during training as it may be too diverse to enumerate (see Fig. 2). **(2) Light-weight**: The method should allow for fast inference to enable real-time robot control. **(3) Discriminative**: The method should yield gaps in scores for successful and failed rollouts. To avoid overfitting on historical data and enable efficient real-time detection [7], the score D_M also only takes the current pair $(A_t, O_t) \in \tau_t$ as inputs rather than the growing trajectory history. We leave exploration of historical time series data for failure detection to future work. To meet our desiderata, we select and build on the following approach categories.

(a) Learned data density: we fit a density estimator to the observations. The intuition is that observations far away from the support of the success distribution may be indicative of failure. The approach we term `logpO` [37] fits a continuous normalizing flow (CNF) to the set of observations $\{O_t\}_{t \geq 0}$. A low $\log p(O_{t'})$ indicates an unlikely observation, indicating possible failure.

(b) Second-order: these methods learn parameters for second-order distributions that can separate aleatoric and epistemic uncertainty [38]. `NatPN` [23] imposes a Dirichlet prior on class probabilities and optimizes model parameters by minimizing a Bayesian loss. We apply `NatPN` to the observations O_t. We also consider multivariate deep evidential regression `DER` [24] applied to the predicted actions. DER assumes $A_t \mid O_t$ follows a multivariate Gaussian distribution with Wishart prior.

(c) One-class discriminator: we consider methods that provide a learned continuous metric from successful observations. The one-class discriminator `RND` [36] initializes a random target network $f_T(\cdot)$ and a predictor network $f(\cdot; \theta)$. The predictor is trained to minimize $\mathbb{E}_{(A_t, O_t) \sim \text{ID trajectory}}[D_M(A_t, O_t; \theta)]$ for $D_M(A_t, O_t; \theta) = ||f_T(A_t, O_t) - f(A_t, O_t; \theta)||_2^2$ on successful demonstration data. Intuitively, RND learns a mapping from the available data (A_t, O_t) to a preset random function. If the learned mapping starts to deviate from the expected random output, the input data is likely OOD. In this category, we also consider consistency flow matching (`CFM`) [39], which measures trajectory curvature with empirical variance of the observation-to-noise forward flow. The intuition is that on ID data, the forward flow is trained to be straight and consistent. Thus, high trajectory curvature indicates the input observations are OOD.

(d) Posthoc metrics: we investigate methods that compute a scalar score analytically without learning. We use `SPARC` [40] to measure the smoothness of predicted actions. We expect `SPARC` to be useful for robot jitter failures, which are empirically frequent in OOD scenarios. The recent SOTA in success-based failure detection, STAC [7], falls in the posthoc method category. However, since it comes with its own statistical evaluation procedure, we describe it as our main baseline in Section 5.

4.2 Sequential Threshold Design with Conformal Prediction (CP)

We design time-varying thresholds η_t such that a failure is flagged when $D_M(A_t, O_t; \theta)$ exceeds η_t. Functional CP [9] is a framework that wraps around a time-series of any scalar score D_M (lower indicates success) and yields a distribution-free prediction band C_α with user-specified significance level $\alpha \in (0, 1)$. Under mild conditions [41, 42, 43], C_α contains any ID score $D_M(A, O; \theta)$ with probability of at least $1 - \alpha$. If $D_M(A, O; \theta) \notin C_\alpha$, we can confidently reject that (A, O) is ID.

For sequential failure detection, we build C_α as a one-sided time-varying CP band. The band is one-sided as we are only concerned with high values of the scalar score D_M, which indicate the

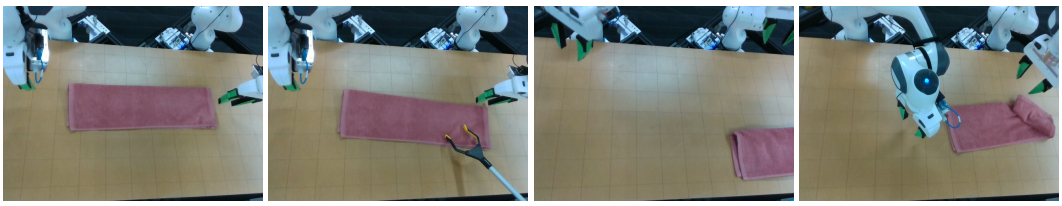
trajectory is OOD (i.e., a failure). Given N success rollouts as the validation data, we obtain scalar scores $\mathcal{D}_{cal} = \{D_M(A_t^i, O_t^i; \theta) : i = 1, \ldots, N \text{ and } t = 1, H', \ldots, T\}$. The CP band is a set of intervals $C_\alpha = \{[\text{lower}_t, \text{upper}_t] : t = 1, H', \ldots, T\}$, where $\text{lower}_t \equiv \min(\mathcal{D}_{cal})$ since the band is one-sided. To obtain the upper bound, we follow [9], computing the time-varying mean μ_t and band width h, so that $\text{upper}_t = \mu_t + h$. Further details of upper bound construction are in Appendix A. Theoretically, for a new success rollout $\tau_T = (O_0, A_0, \ldots, O_T, A_T)$, with probability at least $1 - \alpha$, the score $D_M(A_t, O_t; \theta) \in [\text{lower}_t, \text{upper}_t]$ for all $t = 1, H', \ldots, T$. By defining the threshold $\eta_t = \text{upper}_t$ and setting failures as the positive class, the decision rule $\mathbb{1}(D_M(A_t, O_t; \theta) > \eta_t)$ controls the false positive rate (success is marked as failure) at level α.

5 Experiments

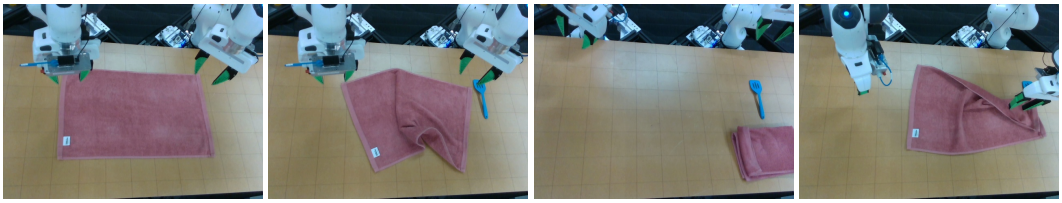
We test our two-stage failure detection framework in both simulation and on a real-world robot platform. Our experiments span multiple environments, each presenting unique challenges in terms of types of tasks and distribution shifts. We empirically investigate an extensive set of both learned and posthoc scalar scores within our failure detection framework. The NatPN [23], DER [24], and RND [44] methods we consider for the scalar score are SOTA OOD detection techniques that do not require OOD data for training. We additionally baseline against STAC [7] as the SOTA approach in success-based failure detection for generative imitation learning policies. We refer to Appendix B for more details of the policy training and learned scalar scores.

Tasks. In simulation, we consider the **Square** and **Transport** tasks from the open-source Robomimic benchmark [45]. The **Square** task asks the robot to pick up a square nut and place it on a rod, which requires precision. The **Transport** task asks two robot arms to transfer a hammer from a closed container on a shelf to a target bin on another shelf, involving coordination between the robots. In the real-world experiments, we consider a **RedTowelFolding** task on a bimanual Franka Emika Panda robot station (see Fig. 8). The **RedTowelFolding** task is the most challenging setting. The two robot arms must fold a deformable red towel twice and push it to the table corner. This task is long-horizon and requires both precision and coordination. We construct OOD settings for each task. In simulation, we adjust the third-person camera 10 degrees upwards at the first time step after $t = 50$ to simulate a camera bump mid-rollout. For the real-world experiment (see Fig. 3), we either interrupt after the first fold (challenging ID scenario) or create an OOD initial condition.

Baselines. We compare learned and posthoc scalar scores within our framework, as well as against the SOTA approach STAC [7]. Specifically, STAC operates by generating batches (e.g., 256) of



(a) Before interruption (b) After interruption (c) Success case (d) Failure case



(e) ID initial condition (f) OOD initial condition (g) Success case (h) Failure case

Figure 3: Experimental settings for the **RedTowelFolding** task. **Interruption after 1st fold (top row):** In (b), the human pulls the towel from (a) towards the bottom during a policy rollout. We note that such recovery behavior is sometimes present in the training data, so the task may succeed as in (c). A failure case is shown in (d). **OOD initial condition (bottom row):** Compared to ID (e), we start with a crumpled towel with a blue spatula distractor to the right of the towel as in (f). Neither condition is present in the training data, thus although the task could succeed as in (g), the success rate is low and the robot typically fails like in (h).

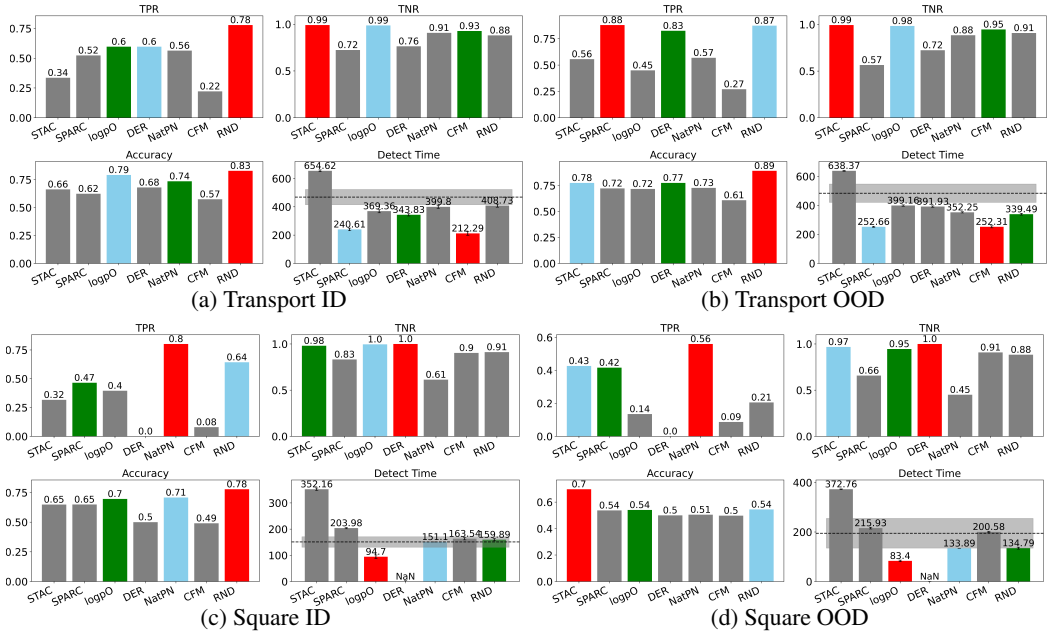
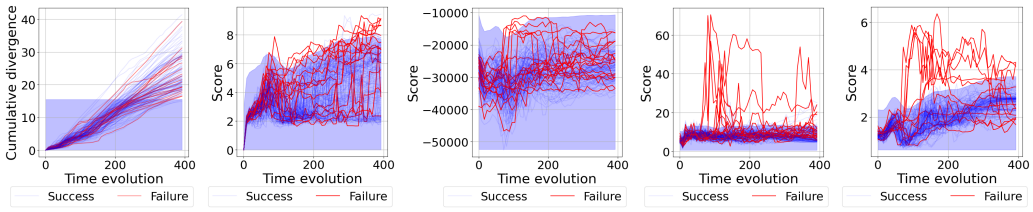


Figure 4: Quantitative results in simulation (best, second, third). We group together posthoc (STAC, SPARC), density-based (logpO), second-order (DER, NatPN), and one-class (CFM, RND) methods. The dashed lines in the Detect Time plots represent average successful trajectory time in that setting with standard error. RND achieves the best Accuracy on all except the Square OOD task. RND gets top-3 performance in 10/16 metric and task combinations, while STAC achieves that in 6/16 cases, showing that RND is more robust across settings. Overall, learned methods seem to have more capacity to detect failures than posthoc ones.

predicted actions at each time step. It then computes the statistical distance (e.g., maximum mean distance (MMD)) between temporally overlapping regions of two consecutive predictions, where the MMD is approximated by batch elements. Intuitively, this MMD measures the "surprise" in the predictions over the rollout and subsequently, STAC makes a detection using CP. Note that instead of computing a CP band for a temporal sequence, STAC computes a single threshold based on empirical quantiles of the cumulative divergence in a validation set. We reproduce the method and adopt hyperparameters used in their push-T example, where we generate a batch of 256 action predictions per time step. We did not employ the VLM component of the STAC failure detector to remain real-time feasible. We also did not filter the predicted actions [7] as we are including pick-and-place tasks. Due to the long STAC inference time (even after parallelization) and resulting high system latency, we omit its comparison on the real-world **RedTowelFolding** task. Lastly, we omit comparisons against ensembles [46], which were outperformed by RND for OOD detection [44].

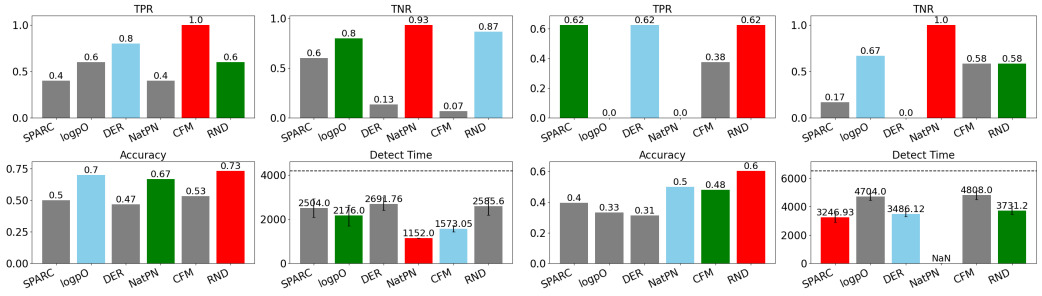
Evaluation protocol: The underlying policy network g is trained with flow matching [47, 48], which has demonstrated SOTA performance on complex imitation learning tasks [49, 4]. In particular, image features are extracted using a ResNet [50] backbone trained jointly with g. The ResNet features of camera images concatenated with robot state constitute observations O_t. Note that our two-stage approach is applicable to other policy networks (e.g., diffusion policy [2]), but we leave this exploration to future work. Table 1 shows success rate across the tasks. To compute the time-varying thresholds η_t, we roll out the FM policy 1000 (resp. 50) times in ID environments for simulation (resp. real-world), collecting only successful trajectories. 30% of successful rollouts are used for computing μ_t and the remaining 70% for the CP band width. Since the **RedTowelFolding** task is particularly challenging, we additionally relax the problem to use setting-dependent thresholds. We take 30 of 50 rollouts for each setting (interruption and OOD) to compute the CP threshold. This setup is less practical as it assumes access to rollouts in the new OOD environment.

We then test the model performance on 2000 (resp. 50) test rollouts for simulation (resp. real-world); for **RedTowelFolding** with setting-dependent CP thresholds, we test on 20 test rollouts. For ID simulation tasks, we test on 1000 test rollouts as the first 1000 are used to calibrate the CP band. To quantify failure detection performance, we adopt the following metrics similar to [7]: **(1)** true

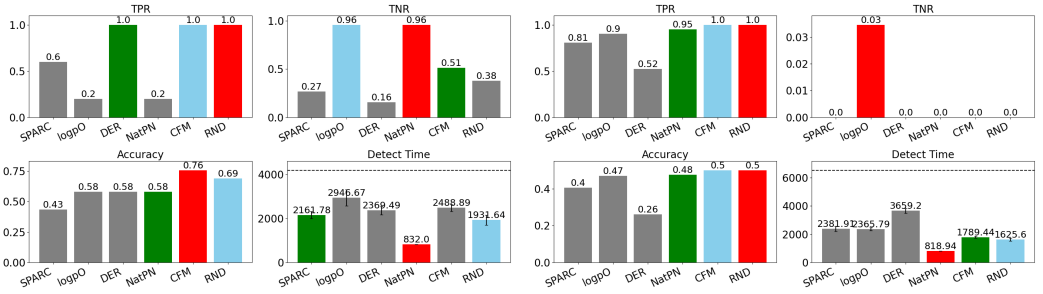


(a) `STAC` (b) `SPARC` (c) `logpO` (d) `NatPN` (e) `RND`

Figure 5: Qualitative results for 150 test set samples on **Square ID** across posthoc (STAC, SPARC) and learned (logpO, NatPN, RND) methods overlaid with the CP bands. We use the non-temporal CP score for STAC as per [7]. The curves are colored by the ground truth success/failure status of the rollout. STAC marks most trajectories as failures due to the low CP threshold and class separation. SPARC catches some failures, but does not detect failures that exhibit smooth trajectories. logpO has high diversity of ID scores, causing wide CP bands and low TPR. NatPN and RND both have tight CP bands and high failure/success separation.



(a) (Setting-dependent band) Interruption after 1st fold (b) (Setting-dependent band) OOD initial condition



(c) (ID-success band) Interruption after 1st fold (d) (ID-success band) OOD initial condition

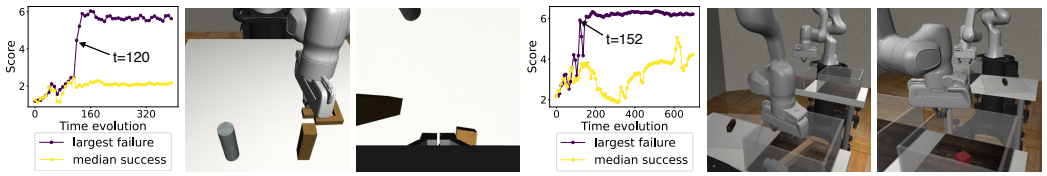
Figure 6: Quantitative results in the real-world using two ways to compute the CP band. The figure layout is the same as Fig. 4 (best, second, third). RND achieves the best Accuracy when using the setting-dependent band, but mostly predicts failures (high TPR but low TNR) under the ID-success band. All methods perform worse with the ID-success band as the successes in these difficult settings vastly differ from the ID successes.

positive rate (TPR), **(2)** true negative rate (TNR), **(3)** accuracy = (TPR + TNR) / 2, and **(4)** detection time = $\mathbb{E}_{(A_t,O_t)\sim\text{test rollouts}}[\arg\min_{t=1,H',\ldots,T} \mathbb{1}(D_M(A_t,O_t;\theta) > \eta_t)]$, which computes the average failure detection time. Here, failure rollouts are denoted with one and success rollouts with zero.

6 Results

We present our key experimental findings, with particular focus on the differences between learned and posthoc scalar scores, the computational efficiency of our approach as compared to the SOTA approach, and the effect of the decision thresholds on the performance metrics.

Learned scalar scores often outperform posthoc scores. The quantitative results on simulation tasks are in Fig. 4 and in Fig. 6 for the real-world task. In simulation, we notice that `RND` reaches the highest Accuracy on all except the Square OOD task, hence achieving the best balance between TPR and TNR. Overall, learned scalar scores (which include `RND`) tend to perform better than posthoc approaches in terms of higher Accuracy, as two among the best three approaches are often learned scores. One exception is the Square OOD task, where all methods perform poorly; STAC reaches high Accuracy yet its Detection Time significantly exceeds the task completion time, making it impractical. Qualitatively, Fig. 5 visualizes the detection scores. Unlike learned methods, posthoc ones often fail to show significant score separation between successful and failed trajectories.



(a) Square (b) Agent view (c) Robot view (d) Transport (e) Agent 1 view (f) Agent 2 view

Figure 7: Physical meaning of RND scores. Subfigures (a)-(c) are for the Square ID task and (d)-(f) are for the Transport ID task. 'Largest failure' is the failure trajectory with the largest cumulative scalar score. At time indices with sudden increases in the score, the cameras also capture physical changes in the environment: **(left)** square slipped from the robot gripper and **(right)** both robots failed to pick up the hammer and the red object.

Computational advantage. Compared to STAC, a posthoc approach which requires generating 256 action predictions per time step, the inference of our learned scores is significantly faster. Specifically, in simulation for a batch of 50 rollouts on one A6000 GPU, computing RND scores takes only 0.04 s (on **Square**) and 0.033 s (on **Transport**) per time step whereas STAC requires 1.45 s on both tasks after parallelization. This amounts to $36\times$ to $44\times$ slower inference. Reducing the action batch size from 256 could speed up STAC, yet this would reduce the statistical power of the approach.

Environment-dependent thresholds aid performance on the real-world task. When the CP band is computed on successful rollouts from an environment (or setting), we call it a 'setting-dependent band'. Meanwhile, if we only compute the CP band once on successful rollouts from the ID environment, we call it 'ID-success band'. In simulation (see Fig. 4), we only considered the ID-success band as this is more practical (i.e., no collection of a hold-out set of rollouts per new environment) and methods could reach good performance. On the real task (see Fig. 6), however, most methods tend to drop in both Accuracy and Detection Time, resulting in as low as zero TNR in the OOD case. This means the methods become over-conservative and predict most (if not all) rollouts as failures. We suspect this likely happens because scores computed on successful trajectories in OOD environments are inherently higher than those in the ID environment, making the ID-success band too narrow. We find that real-world policies are more affected by changes to the environment. In fact, in many instances, although the trajectory ends up completing the task, it does so poorly (slow, jittery, etc.) and perhaps should be considered as a failure in comparison to ID successful rollouts.

RND as a learned score is physically meaningful. As seen in Fig. 7, we notice strong correlation between the rise in scalar scores and what happens in the environment. Specifically, as captured by the cameras, sudden jumps in the scalar score are caused by (1) **Square**: the square dropping as indicated by the robot gripper being closed (2) **Transport**: the front robot arm (Agent 1 view) failing to fetch the hammer and the back robot arm (Agent 2 view) failing to pick up the trash. Thus, spikes in the failure detection scores likely indicate undesirable changes in the underlying environment. We can retrospectively analyze the root cause of such changes.

7 Conclusion

We presented a modular two-stage failure detection approach for generative imitation learning in manipulation tasks. Our method combines a learned scalar signal and functional conformal prediction to accurately and efficiently identify failures while providing statistical guarantees. Through extensive experiments on diverse tasks, we show that the RND score achieves the overall best performance among the considered candidates. Our results highlight the potential of our method to enhance the safety and reliability of robotic systems in real-world applications.

Our work provides the foundation for several avenues for future work. First, while our experiments focused on flow-based policies, extending the evaluation to diffusion policies would provide a more comprehensive understanding of the method's robustness to different policy architectures. Second, we find timely detection of failures based on learned scores in real-world data remains challenging. Some promising directions to ease the problem include: considering multimodal sensory information and expanding the scalar score input to contain temporal data. Lastly, the detected failure data collected could be iteratively used by the policy to further improve performance and robustness.

Acknowledgments

We thank the anonymous reviewers who gave useful comments and helped us improve the quality of the paper. We also thank our colleagues at TRI, Vitor Guizilini and Ben Burchfiel, for thought-provoking discussions that contributed as inspiration for the ideas in this paper.

References

[1] M. Hägele, K. Nilsson, J. N. Pires, and R. Bischoff. Industrial robotics. *Springer handbook of robotics*, pages 1385–1422, 2016.

[2] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[3] T. Z. Zhao, J. Tompson, D. Driess, P. Florence, S. K. S. Ghasemipour, C. Finn, and A. Wahid. ALOHA unleashed: A simple recipe for robot dexterity. In *8th Annual Conference on Robot Learning*, 2024.

[4] K. Chen, E. Lim, K. Lin, Y. Chen, and H. Soh. Don't start from scratch: Behavioral refinement via interpolant-based policy diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[5] A. Vyas, N. Jammalamadaka, X. Zhu, D. Das, B. Kaul, and T. L. Willke. Out-of-distribution detection using an ensemble of self supervised leave-out classifiers. In *Proceedings of the European conference on computer vision (ECCV)*, pages 550–564, 2018.

[6] A. Djurisic, N. Bozanic, A. Ashok, and R. Liu. Extremely simple activation shaping for out-of-distribution detection. In *The Eleventh International Conference on Learning Representations*, 2023.

[7] C. Agia, R. Sinha, J. Yang, Z. Cao, R. Antonova, M. Pavone, and J. Bohg. Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=yqLFb0RnDW.

[8] G. Shafer and V. Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(3), 2008.

[9] J. Diquigiovanni, M. Fontana, S. Vantini, et al. The importance of being a band: Finite-sample exact distribution-free prediction sets for functional data. *STATISTICA SINICA*, 1:1–41, 2024.

[10] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[11] T. Yu, T. Xiao, J. Tompson, A. Stone, S. Wang, A. Brohan, J. Singh, C. Tan, D. M, J. Peralta, K. Hausman, B. Ichter, and F. Xia. Scaling Robot Learning with Semantically Imagined Experience. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi:10.15607/RSS.2023.XIX.027.

[12] A. Sridhar, D. Shah, C. Glossop, and S. Levine. Nomad: Goal masked diffusion policies for navigation and exploration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 63–70. IEEE, 2024.

[13] L. Chen, S. Bahl, and D. Pathak. Playfusion: Skill acquisition via diffusion from language-annotated play. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=afF8RGcBBP.

[14] M. Reuss and R. Lioutikov. Multimodal diffusion transformer for learning from play. In *2nd Workshop on Language and Robot Learning: Language as Grounding*, 2023.

[15] W. K. Kim, M. Yoo, and H. Woo. Robust policy learning via offline skill diffusion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 13177–13184, 2024.

[16] D. Wang, S. Hart, D. Surovik, T. Kelestemur, H. Huang, H. Zhao, M. Yeatman, J. Wang, R. Walters, and R. Platt. Equivariant diffusion policy. In *8th Annual Conference on Robot Learning*, 2024.

[17] Y. Wang, G. Yin, B. Huang, T. Kelestemur, J. Wang, and Y. Li. GenDP: 3d semantic fields for category-level generalizable diffusion policy. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=7wMlwhCvjS.

[18] X. Hu, B. Liu, X. Liu, and Q. Liu. Adaflow: Imitation learning with variance-adaptive flow-based policies. *arXiv preprint arXiv:2402.04292*, 2024.

[19] J. Yang, K. Zhou, Y. Li, and Z. Liu. Generalized out-of-distribution detection: A survey. *International Journal of Computer Vision*, pages 1–28, 2024.

[20] W. Liu, X. Wang, J. Owens, and Y. Li. Energy-based out-of-distribution detection. *Advances in neural information processing systems*, 33:21464–21475, 2020.

[21] X. Du, Z. Wang, M. Cai, and S. Li. Towards unknown-aware learning with virtual outlier synthesis. In *International Conference on Learning Representations*, 2022.

[22] F. Castañeda, H. Nishimura, R. McAllister, K. Sreenath, and A. Gaidon. In-distribution barrier functions: Self-supervised policy filters that avoid out-of-distribution states. In *5th Annual Learning for Dynamics & Control Conference (L4DC)*, volume 211, pages 286–299. PMLR, 2023.

[23] B. Charpentier, O. Borchert, D. Zügner, S. Geisler, and S. Günnemann. Natural posterior network: Deep bayesian predictive uncertainty for exponential family distributions. In *International Conference on Learning Representations*, 2022.

[24] L. Bramlage, M. Karg, and C. Curio. Plausible uncertainties for human pose regression. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15087–15096, 2023. doi:10.1109/ICCV51070.2023.01389.

[25] M. Itkina and M. Kochenderfer. Interpretable self-aware neural networks for robust trajectory prediction. In *Conference on Robot Learning*, pages 606–617. PMLR, 2023.

[26] C. Xu and Y. Xie. Conformal anomaly detection on spatio-temporal observations with missing data. *arXiv preprint arXiv:2105.11886*, 2021.

[27] S. Basart, M. Mantas, M. Mohammadreza, S. Jacob, and S. Dawn. Scaling out-of-distribution detection for real-world settings. In *International Conference on Machine Learning*, 2022.

[28] L. Tao, X. Du, J. Zhu, and Y. Li. Non-parametric outlier synthesis. In *The Eleventh International Conference on Learning Representations*, 2023.

[29] R. Natarajan, S. R. P, S. C. Bose, H. Gururaj, F. Flammini, and S. Velmurugan. Fault detection and state estimation in robotic automatic control using machine learning. *Array*, 19:100298, 2023. ISSN 2590-0056. doi:https://doi.org/10.1016/j.array.2023.100298. URL https://www.sciencedirect.com/science/article/pii/S2590005623000231.

[30] Q. M. Rahman, P. Corke, and F. Dayoub. Run-time monitoring of machine learning for robotic perception: A survey of emerging trends. *IEEE Access*, 9:20067–20075, 2021. doi:10.1109/ACCESS.2021.3055015.

[31] R. Sinha, A. Elhafsi, C. Agia, M. Foutter, E. Schmerling, and M. Pavone. Real-Time Anomaly Detection and Reactive Planning with Large Language Models. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi:10.15607/RSS.2024.XX.114.

[32] H. Liu, S. Dass, R. Martín-Martín, and Y. Zhu. Model-based runtime monitoring with interactive imitation learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[33] J. Wong, A. Tung, A. Kurenkov, A. Mandlekar, L. Fei-Fei, S. Savarese, and R. Martín-Martín. Error-aware imitation learning from teleoperation data for mobile manipulation. In *Conference on Robot Learning*, pages 1367–1378. PMLR, 2022.

[34] A. Z. Ren, A. Dixit, A. Bodrova, S. Singh, S. Tu, N. Brown, P. Xu, L. Takayama, F. Xia, J. Varley, Z. Xu, D. Sadigh, A. Zeng, and A. Majumdar. Robots that ask for help: Uncertainty alignment for large language model planners. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=4ZK8ODNyFXx.

[35] J. Sun, Y. Jiang, J. Qiu, P. T. Nobel, M. Kochenderfer, and M. Schwager. Conformal prediction for uncertainty-aware planning with diffusion dynamics model. In *Neural Information Processing Systems*, 2023.

[36] N. He, S. Li, Z. Li, Y. Liu, and Y. He. ReDiffuser: Reliable decision-making using a diffuser with confidence estimation. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 17921–17933. PMLR, 21–27 Jul 2024.

[37] C. Xu, X. Cheng, and Y. Xie. Normalizing flow neural networks by JKO scheme. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[38] M. Sensoy, L. Kaplan, and M. Kandemir. Evidential deep learning to quantify classification uncertainty. 31, 2018.

[39] L. Yang, Z. Zhang, Z. Zhang, X. Liu, M. Xu, W. Zhang, C. Meng, S. Ermon, and B. Cui. Consistency flow matching: Defining straight flows with velocity consistency. *arXiv preprint arXiv:2407.02398*, 2024.

[40] S. Balasubramanian, A. Melendez-Calderon, A. Roby-Brami, and E. Burdet. On the analysis of movement smoothness. *Journal of neuroengineering and rehabilitation*, 12:1–11, 2015.

[41] V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic learning in a random world*, volume 29. Springer, 2005.

[42] C. Xu and Y. Xie. Conformal prediction for time series. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10):11575–11587, 2023.

[43] C. Xu and Y. Xie. Sequential predictive conformal inference for time series. In *International Conference on Machine Learning*, pages 38707–38727. PMLR, 2023.

[44] K. Ciosek, V. Fortuin, R. Tomioka, K. Hofmann, and R. Turner. Conservative uncertainty estimation by fitting prior networks. In *International Conference on Learning Representations*, 2020.

[45] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *5th Annual Conference on Robot Learning*.

[46] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[47] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

[48] C. Xu, X. Cheng, and Y. Xie. Local flow matching generative models. *arXiv preprint arXiv:2410.02548*, 2024.

[49] Q. Rouxel, A. Ferrari, S. Ivaldi, and J.-B. Mouret. Flow matching imitation learning for multi-support manipulation. *arXiv preprint arXiv:2407.12381*, 2024.

[50] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[51] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

[52] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

A CP band construction

Following [9] we split the set of calibration scores $\mathcal{D}_{cal}$ into two disjoint parts $\mathcal{D}_{cal_A}$ and $\mathcal{D}_{cal_B}$ with sizes N_1 and N_2. We first compute the mean successful trajectory $\mu_t = N_1^{-1} \sum_{i=1}^{N_1} D_M(A_t^i, O_t^i; \theta)$ for $t = 1, \ldots, T$ on $\mathcal{D}_{cal_A}$. Then, for $j = 1, \ldots, N_2$, we compute $D_j = \max(\{\mu_t - D_M(A_t^j, O_t^j; \theta)\}_{t=1}^T)$, which is the max deviation over rollout length from the mean prediction to the scalar score. Note that the $\max$ is taken because the CP band is intended to reflect the entire trajectory. We define $S = \{D_j, j = 1, \ldots, N_2\}$ as the collection of such max deviations. The band width h is finally computed as the $(1-\alpha)$-quantile of S and the upper bound is $\text{upper}_t = \mu_t + h$.

B Experimental Details

Policy training with flow matching: We follow the setup in [2] and use the same hyperparameters to train the policies. The only difference is that instead of optimizing with the diffusion loss, we change the objective to be a flow-matching loss between $A_t|O_t$ and Z, the standard Gaussian. We adopt the CNN UNet architecture for the policy and jointly train the ResNet encoder for image observations and the policy network. The observations O_t thus include both the ResNet embedding

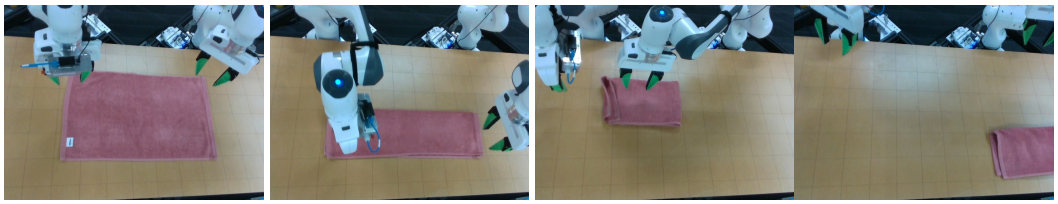
Table 1: Success rate of the flow policy on test data in each task-environment combination. These test data are used to test failure detection methods as well. On the real task, we mark some cells with * when the number of failures out of 50 rollouts is no greater than 5. In such cases, we shuffle the rollout indices and include all the failure ones in the test set, so that the failure detection metrics have higher statistical significance. The true success rate of flow policy on RedTowelFolding ID across the entire 50 rollouts is 0.96.

(a) Simulation tasks

Square ID	Square OOD	Transport ID	Transport OOD
0.90 (1000 rollouts)	0.63 (2000 rollouts)	0.85 (1000 rollouts)	0.63 (2000 rollouts)

(b) Real-world task

	RedTowelFolding ID	RedTowelFolding ID+Interruption	RedTowelFolding OOD
Setting-dependent band	0.9* (20 rollouts)	0.75* (20 rollouts)	0.60 (20 rollouts)
ID-success band	0.9* (20 rollouts)	0.90 (50 rollouts)	0.58 (50 rollouts)



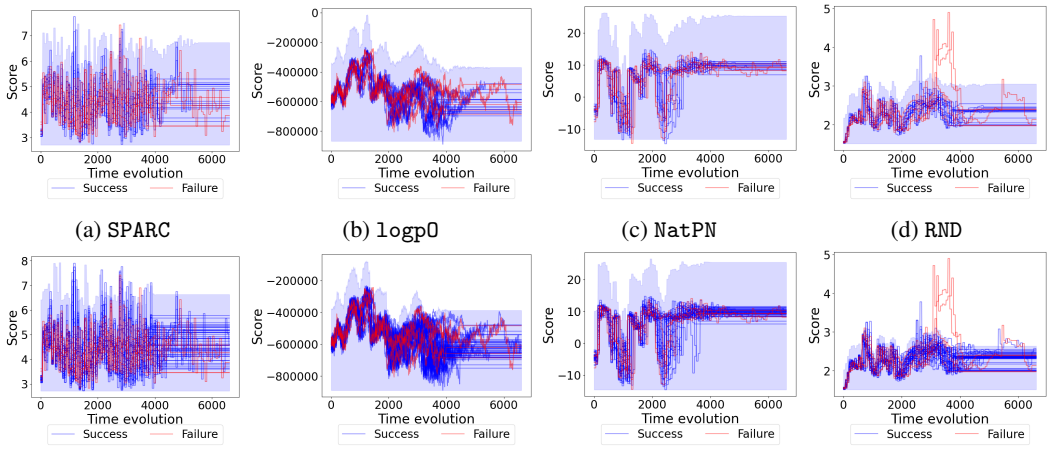
(a) Initial condition (b) After 1st fold (c) After 2nd fold (d) Final success

Figure 8: The real-world on-robot **RedTowelFolding** experimental setting. Starting with a flat towel, the two arms need to first fold the towel along the short side, and then the right arm needs to perform the second fold along the long side. Finally, the towel needs to be pushed to the bottom right corner to be considered a success.

of images from the camera and the non-visual information (e.g., gripper position and robot position). On the simulation tasks, we train for a different number of epochs (800 epochs on **Square** and 300 epochs on **Transport**). On the real **RedTowelFolding**, we train the networks for 1000 epochs.

Scalar failure detection scores: After learning the policy network g with the ResNet encoder for camera images, we first obtain $\{(A_t, O_t)\}$ for each task using the same training demonstration data for policy network. For the post-hoc approach `SPARC`, it utilizes the arc length of the Fourier magnitude spectrum obtained from the trajectory. To learn and test the scalar scores, we adopt the following setup:

1. `CFM`: We use a 4x smaller network with identical architecture as the policy network. It is unconditional and takes in observations O_t as inputs. We train for 200 epochs with a batch size of 128, using the Adam optimizer [51] with a constant 1e-4 learning rate.
2. `logpO`: The network (taking O_t as inputs) has the same architecture as the policy network but is 4x smaller. We traiin for 500 epochs with a batch size of 128, using the Adam optimizer with a constant 1e-4 learning rate. For a new observation $O_{t'}$, its density $\log p(O_{t'})$ is obtained via the instantaneous change-of-variable formula [52].
3. `DER`: The network to parametrize the Normal-Inverse Wishart parameters has the same architecture as the policy network but is 4x smaller. It takes in O_t as inputs. We train for 200 epochs with a batch size of 128, using the Adam optimizer with a constant 1e-4 learning rate.



(a) `SPARC` (b) `logpO` (c) `NatPN` (d) `RND`

(e) `SPARC` (f) `logpO` (g) `NatPN` (h) `RND`

Figure 9: Failure detection scores for 20 (top row with setting-dependent band) or 50 (bottom row with ID-success band) samples from the test set on **RedTowelFolding** across posthoc (SPARC) and learned (logpO, NatPN, RND) methods overlaid with the CP bands. Among all methods, only RND has tight CP bands and high qualitative separation between failures and successes.

4. `NatPN`: We first use K-means clustering to obtain class labels Y for the observations $X = O_t$. We then consider the case where Y follows a categorical distribution with a Dirichlet prior on the distribution parameters. To lean the parameters, we then follow [23] to use the tabular encoder with 16 flow layers. We set the learning rate to be 1e-3 and train for a maximum of 1000 epochs.
5. `RND`: On simulation, we use a 4x smaller network as the policy network, which takes in both A_t and O_t as inputs (O_t as the conditioning variable). We train for 200 epochs with a batch size of 128, using the Adam optimizer with a constant 1e-4 learning rate. On real data, we use network with the same size as the policy network to improve performance. We train for 2000 epochs with a batch size of 512, using the Adam optimizer with a constant 1e-4 learning rate. During inference, a high $D_M(A_t, O_t; \hat{\theta})$ indicates a large mismatch between the predictor and target outputs, which we hypothesize results from the pair (A_t, O_t) not being from a successful trajectory.

C Additional Results

We include the following results:

- Fig. 8 shows a successful demonstration of the robot folding the red towel, starting with a flat red towel in (a).
- Fig. 9 shows the failure detection scores of posthoc and learned methods, which are overlaid with CP band.