Can We Detect Failures Without Failure Data? Uncertainty-Aware Runtime Failure Detection for Imitation Learning Policies

Chen Xu[1], Tony Khuong Nguyen[1], Emma Dixon[1], Christopher Rodriguez[1], Patrick Miller[1], Robert Lee[2], Paarth Shah[1], Rares Ambrus[1], Haruki Nishimura[1], and Masha Itkina[1]
[1]Toyota Research Institute (TRI), [2]Woven by Toyota (WbyT)
chen.xu@tri.global

***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. As robot policy performance increases, so does the complexity and time horizon of achievable tasks, inducing unexpected and diverse failure modes that are difficult to predict a priori. To enable trustworthy policy deployment in safety-critical human environments, reliable runtime failure detection becomes important during policy inference. However, most existing failure detection approaches rely on prior knowledge of failure modes and require failure data during training, which imposes a significant challenge in practicality and scalability. In response to these limitations, we present FAIL-Detect, a modular two-stage approach for failure detection in imitation learning-based robotic manipulation. To accurately identify failures from successful training data alone, we frame the problem as sequential out-of-distribution (OOD) detection. We first distill policy inputs and outputs into scalar signals that correlate with policy failures and capture epistemic uncertainty. FAIL-Detect then employs conformal prediction (CP) as a versatile framework for uncertainty quantification with statistical guarantees. Empirically, we thoroughly investigate both learned and post-hoc scalar signal candidates on diverse robotic manipulation tasks. Our experiments show learned signals to be mostly consistently effective, particularly when using our novel flow-based density estimator. Furthermore, our method detects failures more accurately and faster than state-of-the-art (SOTA) failure detection baselines. These results highlight the potential of FAIL-Detect to enhance the safety and reliability of imitation learning-based robotic systems as they progress toward real-world deployment.**

I. Introduction

Robotic manipulation has applications in many important fields, such as manufacturing, logistics, and healthcare [19]. Recently, imitation learning algorithms have shown tremendous success in learning complex manipulation skills from human demonstrations using stochastic generative modeling, such as diffusion- [12, 65] and flow-based methods [9, 45]. However, despite their outstanding results, policy networks can fail due to poor stochastic sampling from the action distribution. The models may also encounter out-of-distribution (OOD) conditions where the input observations deviate 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 quickly 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, is often high-dimensional with complicated distributions. This makes it difficult to identify discriminative features that distinguish between successful and failed executions, particularly 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 must be general and robust to diverse failure scenarios.

In imitation learning, training data naturally consists of successful trajectories, making failed trajectories OOD. Prior work often tackles failure detection through binary classification of ID and OOD conditions [33]. Thus, many of these methods [24, 17, 18, 33] require OOD data for training the failure classifier. This poses significant challenges since collecting and annotating a comprehensive set of failure examples is often time-consuming, expensive, and even infeasible in many real-world scenarios. Moreover, classifiers trained on specific sets of OOD data may not generalize well to unseen failure modes. To address these limitations, we develop a failure detection approach that operates without OOD data, overcoming the need of failure examples while maintaining robust performance. See Appendix A related works.

Our contributions are as follows. We propose **FAIL-Detect**: **F**ailure **A**nalysis in **I**mitation **L**earning – **Detect**ing failures without failure data (see Fig. 1). **In the first stage**, we extract scalar signals from policy inputs and/or outputs (e.g., robot states, visual features, generated future actions) that are discriminative between successes and failures during policy inference. We investigate both learned and post-hoc signal candidates, finding learned signals to be the most accurate for failure detection. A key novelty of our method is the ability to learn failure detection signals without access to failure data. Aside from being performant, our method enables faster inference than prior work [1], which requires sampling multiple robot actions during inference. **In the second stage**, we use conformal prediction (CP) [52, 47] to construct a time-varying threshold to sequentially determine when a score indicates failure with statistical guarantees on false positive

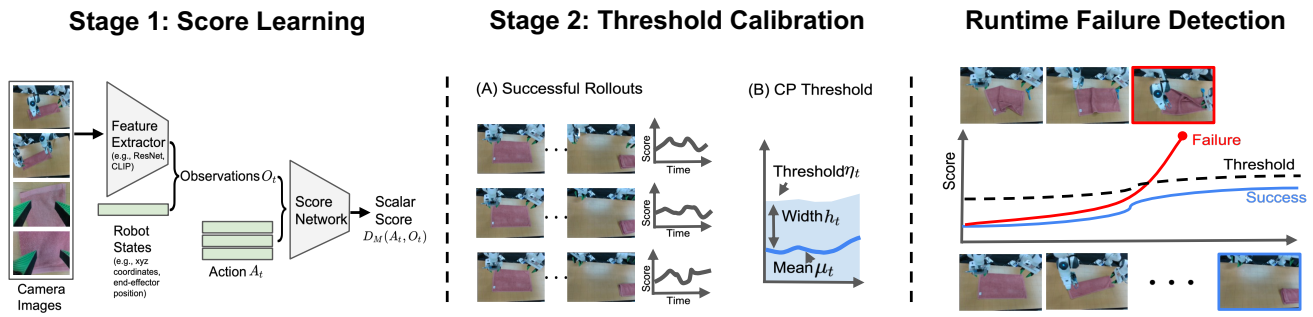
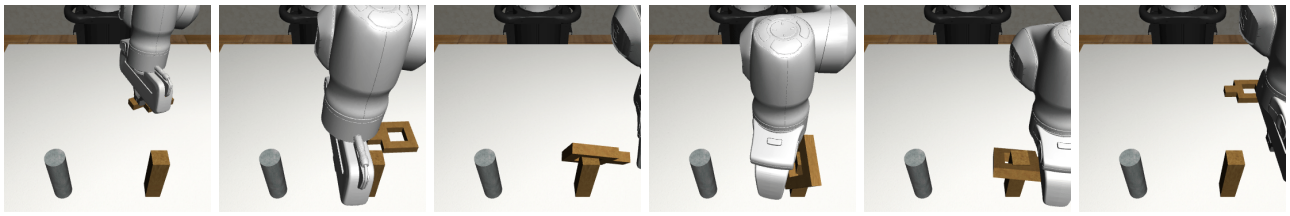


Fig. 1: **FAIL-Detect**: **F**ailure **A**nalysis in **I**mitation **L**earning – **Detect**ing failures without failure data. We propose a two-stage approach to failure detection. **(Left - Stage I)** Multi-view camera images and robot states are distilled into failure detection scalar scores. Images are first passed through a feature extractor and then, along with robot states, constitute observations O_t. Both O_t and generated future robot actions A_t can serve as inputs to a score network D_M. This network outputs scalar scores $D_M(A_t, O_t)$ that capture characteristics of successful demonstration data. **(Middle - Stage II)** Scores from a calibration set of successful rollouts are then used to compute a mean μ_t and band width h_t to build the time-varying conformal prediction threshold. **(Right - Runtime Failure Detection)** A successful trajectory (bottom) has scores that consistently remain below the threshold. When a failure occurs (top), such as failure to fold the towel, the score spikes above the threshold, triggering failure detection (red box).



(a) Slipped out early. (b) Slipped out late. (c) Tilted upward. (d) Tilted downward. (e) Tilted slightly. (f) Not picked up.

Fig. 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 time steps across multiple rollouts and include the square slipping out of the gripper or being misplaced (e.g., with tilted position) on the peg. FAIL-Detect is able to handle the wide range of failures observed at test time.

rates. By integrating adaptive functional CP [14] into our pipeline, we obtain thresholds that adjust to the changing dynamics of manipulation tasks unlike static thresholds used in prior work [1]. We show that FAIL-Detect identifies failures accurately and quickly on diverse robotic manipulation tasks, both in simulation and on robot hardware, outperforming SOTA failure detection baselines.

II. Problem Setup and FAIL-Detect Framework

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 environment observation (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 executed, after which the robot re-plans by generating a new sequence of H actions at time $t+H'$. Recent works have trained effective generators g via DP [12] and FM [9]. Given an initial condition O_0 and the generator g to output the next actions, we obtain a trajectory $\tau_t = (O_0, A_0, O_{H'}, A_{H'}, \ldots, O_t, A_t)$ up to $t = kH'$ $(k \geq 1)$ execution time steps. Failure detection can thus be framed as designing a decision function $D(\cdot;\theta) : \tau_t \to \{0, 1\}$ with parameters θ, which takes in the current trajectory and makes a decision. If the decision $D(\tau_t;\theta) = 1$, 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.

We now introduce the failure detection framework; see Fig. 1 for an overview of the framework. Given action-observation data (A_t, O_t), we propose a two-stage framework to design the decision function $D(\cdot;\theta)$:

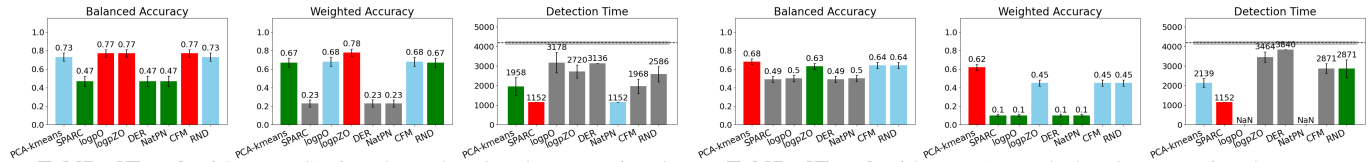
1) Train a scalar score model $D_M(\cdot;\theta) : (A_t, O_t) \to \mathbb{R}$ (for score "method" M) on action and/or observation pairs from successful trajectories only. See Appendix B for details.
2) Calibrate time-varying thresholds η_t based on a CP band. See Appendix C for details.

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

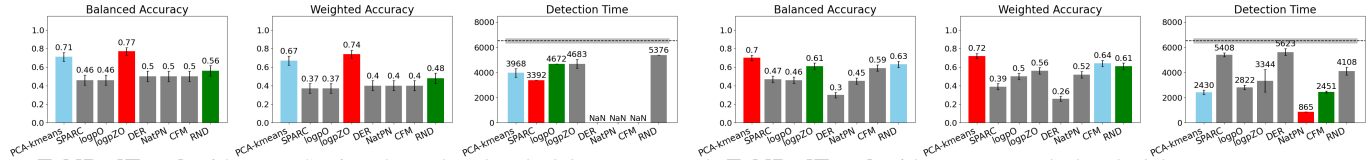
III. Results

We describe the experiment setup in Appendix D, which includes tasks description, baseline choices, and evaluation protocol. We present our experimental findings addressing the following research questions:

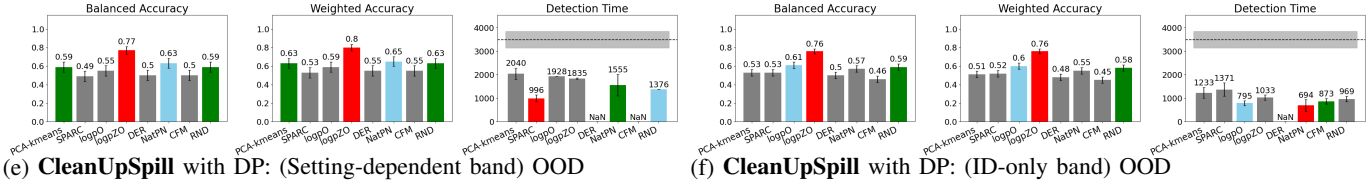
A. How performant is failure detection without failure data?



(a) **FoldRedTowel** with FM: (Setting-dependent band) ID + Disturb (b) **FoldRedTowel** with FM: (ID-only band) ID + Disturb

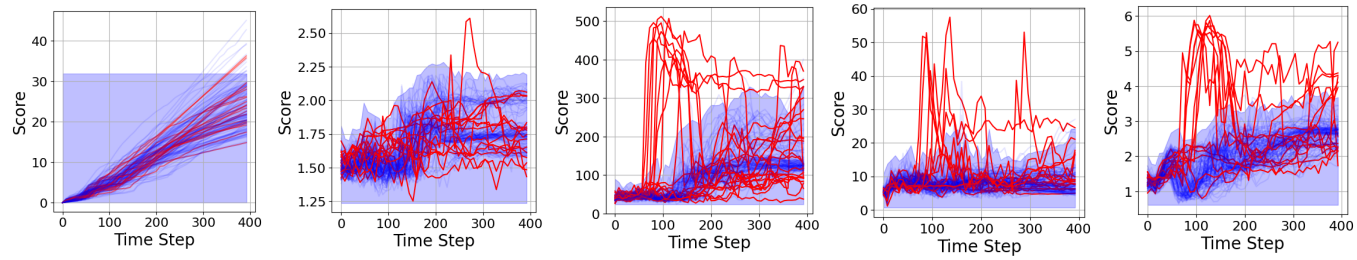


(c) **FoldRedTowel** with FM: (Setting-dependent band) OOD (d) **FoldRedTowel** with FM: (ID-only band) OOD



(e) **CleanUpSpill** with DP: (Setting-dependent band) OOD (f) **CleanUpSpill** with DP: (ID-only band) OOD

Fig. 3: Quantitative results for the robot hardware experiments across two tasks with policies trained using FM and DP. We consider two different ways to compute the CP band: "setting-dependent" using successful trajectories from each OOD/ID environment and "ID-only" using only the trajectories from the ID environment. For balanced accuracy and weighted accuracy, higher is better and for detection time, lower is better. Additional metrics are reported in Fig. 13 and Fig. 14. The figure layout is the same as Fig. 8 (best, second, third), and 'NaN' detection time indicates that no test rollout was detected as failed. Once again the learned approaches outperform the post-hoc methods. Note we do not present STAC here as it was slow to run on hardware in real-time. In the small sample size regime, logpZO remains robust in **combined accuracy**, achieving top-1 performance in the highest number of cases (8/12) and top-3 performance in 11/12 cases. RND underperforms by never reaching top-1 performance, yet it always achieves top-3 performance. In contrast, the PCA-kmeans baseline reaches top-1 performance in 4/12 cases and top-3 performance in 10/12 cases. In **detection time**, the post-hoc SPARC method is the fastest in 4/6 cases, yet it never achieves top-1 performance. PCA-kmeans is robust in speed as it attains top-3 performance in 4/6 cases. logpZO still remains practical with detection times well below the average success trajectory completion time.



(a) STAC (b) PCA-kmeans (c) logpZO (d) NatPN (e) RND

Fig. 4: Qualitative results of failure detection scores overlaid with CP bands. The curves are colored by the ground truth success/failure status of the rollout (failure = red and success = blue). We show 150 test rollouts on **Square ID** across post-hoc baselines (STAC, PCA-kmeans) and learned FAIL-Detect methods (logpZO, NatPN, RND). We use the constant CP threshold for STAC as per [1]. Note that post-hoc baseline methods mark most trajectories as successes due to the poor failure/success separation. In comparison, learned metrics have tight CP bands and higher failure/success separation.

B. What is the impact of learned vs. post-hoc scores on failure detection?

C. Do failure detections align with human intuition?

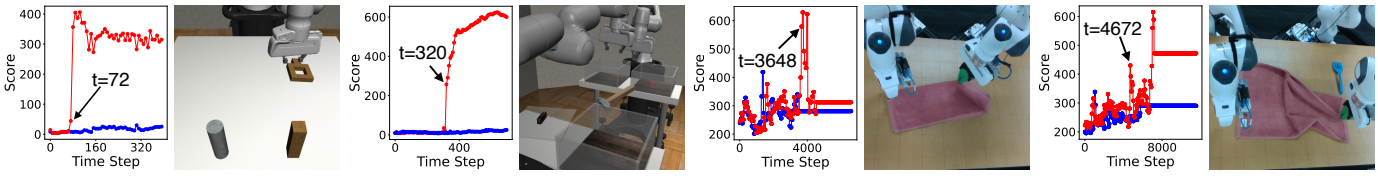
A. How performant is failure detection without failure data?

A key question we consider is whether failure detection is possible and performant without enumerating all possible failure scenarios, which is practically infeasible. We conduct extensive experiments across simulation and robot hardware tasks to answer this question. We evaluate balanced accuracy, weighted accuracy, and detection time to assess whether failures can be identified reliably and quickly.

FAIL-Detect achieves high accuracy with fast detection. Our two-stage framework demonstrates strong performance across both accuracy metrics and detection speed. For example, the average *best* balanced accuracy across FAIL-Detect's score candidates is $\sim 78\%$ in simulation (Fig. 8) and $\sim 72\%$ on the robot hardware tasks (Fig. 3). This performance shows the capacity of failure-free failure detection methods to robustly identify failures across many scenarios. Notably, FAIL-Detect maintains viable detection time across various score designs, with average *best* detection time faster than successful trajectory completion.

B. What is the impact of learned vs. post-hoc scores on failure detection?

Learned scores outperform post-hoc scores. Looking at performance across simulation and robot hardware tasks, we



(a) Square (b) Transport (c) FoldRedTowel ID + Disturb (d) FoldRedTowel OOD

Fig. 5: Physical interpretation of logpZO, the most successful and robust learned score method. Failed trajectory scores are in red and successful ones are in blue. Each figure shows the failure detection time and the corresponding camera view. **(Simulation)** In Fig. 5a, failure is flagged when the square slips from the gripper. In Fig. 5b, failure is detected when both arms drop the hammer. **(On-robot)** In Fig. 5c, failure is alerted as the second fold attempt fails. In Fig. 5d, failure is detected as the left robot arm fails to complete the first fold.

find that learned scalar scores hold an advantage over post-hoc scores in failure detection. In simulation (Fig. 8), logpZO and RND are the best two methods, achieving top-1 performance in 10/16 and 5/16 cases, respectively. STAC is the best in the post-hoc category for top-1 accuracy in 3/16 cases, yet PCA-kmeans is never the best. Overall, there is a large performance gap between the learned and post-hoc methods, especially in terms of the best overall accuracy. We did notice that post-hoc methods perform better in the OOD cases than in ID scenarios. We hypothesize this may be due to a clearer distinction between successful ID trajectories and failed OOD trajectories, for example, for SPARC if the OOD trajectory exhibits significant jitter.

In terms of detection time, logpZO is the most efficient, achieving the fastest time in 3/8 cases, while PCA-kmeans does so in only 1/8 cases. Notably, STAC's detection time consistently exceeds practical limits, surpassing the average success trajectory time.

For the robot hardware experiments (Fig. 3), with a much lower rollout data regime for calibration than in simulation (see Table III) and a wider diversity of behaviors and observations, logpZO remains robust as the best method, reaching top-1 highest balanced accuracy and weighted accuracy in 8/12 scenarios. The PCA-kmeans baseline is second best with 4/12 top-1 ranking. RND underperforms by never achieving top-1 performance, yet it always ranks among the top-3 best methods. Additionally, most methods, including logpZO and RND, maintain practical detection times well below the average successful trajectory time. SPARC is on average the fastest as it attains top-1 performance in 4/6 cases, but it exhibits poor accuracy.

Overall, across all experiments, we find our novel learned logpZO score within the FAIL-Detect framework to be most consistent in performance. The post-hoc methods are often at the extremes of performance (either doing well or poorly) depending on the particular setting.

Qualitative score trends. Visualization of the detection scores (Figs. 4 and 9) confirms that learned methods are more discriminative with better score separation between successful and failed trajectories compared to post-hoc approaches. STAC also suffers from a single calibration threshold that is time invariant. In Appendix E, we present comprehensive ablation studies examining performance sensitivity to CP significance level α (Fig. 10).

Computational advantage. Some post-hoc methods require sampling from the stochastic policy repeatedly to achieve a performant failure score. For example, STAC requires generating 256 action predictions per time step. Although the computational efficiency could be improved by generating fewer predictions, this compromises its statistical reliability. On the other hand, our learned scores offer significantly faster inference speeds compared to STAC. For instance, testing on an A6000 GPU with 50 rollouts, logpZO score computation takes $0.04\,\mathrm{s}$ (**Square**) and $0.033\,\mathrm{s}$ (**Transport**) per time step, while STAC requires $1.45\,\mathrm{s}$ for both tasks, amounting to a 36-44 times slowdown.

C. Do failure detections align with human intuition?

FAIL-Detect's alerts demonstrate strong correlation with observable failure indications in the environment (Fig. 5). When scores exceed the decision threshold, these moments often align with meaningful changes in the physical state of the task. In simulation environments, the detection scores capture distinct failure patterns with high precision. For instance, for the **Square** task, abrupt increases in scores coincide with the moment the gripper loses its hold on the square. Similarly, in the **Transport** task, score spikes identify the instant when the hammer slips during inter-arm transfer. Real-world applications demonstrate similarly compelling results: the system effectively detects both human-induced disruptions leading to an incomplete second towel fold (**FoldRedTowel ID + Disturb**) and OOD initial conditions resulting in an improper first towel fold (**FoldRedTowel OOD**).

This correspondence between score spikes and physical events is encouraging for FAIL-Detect's capacity to capture task-relevant subtlety. The framework successfully translates complex environmental changes into quantifiable metrics, with score variations serving as reasonable indicators of failure events. Moreover, this correlation offers valuable diagnostic capabilities for potential policy improvement. Instead of requiring an exhaustive a priori enumeration of potential failure modes, which is an inherently challenging endeavor, our approach enables potential targeted analysis of observed failures. By examining executions within temporal windows surrounding a failure detection, one can efficiently identify failure types for subsequent analysis.

Real-World Deployability and Generalization

Our proposed two-stage FAIL-Detect, powered by the novel logpZO score, consistently outperforms baselines across a wide range of tasks. Importantly, FAIL-Detect can identify previously unseen failures without failure training data. This is crucial for real-world deployment, where collecting exhaustive failure scenarios is undesirable and costly. By enabling robust failure detection under different conditions, our FAIL-Detect also shows good generalization across tasks and environments. These qualities make FAIL-Detect particularly suited for real-world robotic applications, where safety, reliability, and adaptability are desired.

References

[1] Christopher Agia, Rohan Sinha, Jingyun Yang, Ziang Cao, Rika Antonova, Marco Pavone, and Jeannette Bohg. Unpacking Failure Modes of Generative Policies: Runtime Monitoring of Consistency and Progress. In *Conference on Robot Learning (CoRL)*, 2024.

[2] Sivakumar Balasubramanian, Alejandro Melendez-Calderon, Agnes Roby-Brami, and Etienne Burdet. On the Analysis of Movement Smoothness. *Journal of Neuroengineering and Rehabilitation*, 12:1–11, 2015.

[3] Steven Basart, Mazeika Mantas, Mostajabi Mohammadreza, Steinhardt Jacob, and Song Dawn. Scaling Out-of-Distribution Detection for Real-world Settings. In *International Conference on Machine Learning (ICML)*, 2022.

[4] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164*, 2024.

[5] Lennart Bramlage, Michelle Karg, and Cristóbal Curio. Plausible Uncertainties for Human Pose Regression. In *International Conference on Computer Vision (ICCV)*, pages 15087–15096. IEEE, 2023.

[6] Max Braun, Noémie Jaquier, Leonel Rozo, and Tamim Asfour. Riemannian Flow Matching Policy for Robot Motion Learning. *arXiv preprint arXiv:2403.10672*, 2024.

[7] Fernando Castañeda, Haruki Nishimura, Rowan McAllister, Koushil Sreenath, and Adrien Gaidon. In-Distribution Barrier Functions: Self-Supervised Policy Filters that Avoid Out-of-Distribution States. In *Learning for Dynamics & Control Conference (L4DC)*, volume 211, pages 286–299. PMLR, 2023.

[8] Bertrand Charpentier, Oliver Borchert, Daniel Zügner, Simon Geisler, and Stephan Günnemann. Natural Posterior Network: Deep Bayesian Predictive Uncertainty for Exponential Family Distributions. In *International Conference on Learning Representations (ICLR)*, 2022.

[9] Kaiqi Chen, Eugene Lim, Kelvin Lin, Yiyang Chen, and Harold Soh. Don't Start from Scratch: Behavioral Refinement via Interpolant-based Policy Diffusion. In *Robotics: Science and Systems (RSS)*, 2024.

[10] Lili Chen, Shikhar Bahl, and Deepak Pathak. PlayFusion: Skill Acquisition via Diffusion from Language-Annotated Play. In *Conference on Robot Learning (CoRL)*, 2023.

[11] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural Ordinary Differential Equations. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

[12] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Robotics: Science and Systems (RSS)*, 2023.

[13] Kamil Ciosek, Vincent Fortuin, Ryota Tomioka, Katja Hofmann, and Richard Turner. Conservative Uncertainty Estimation By Fitting Prior Networks. In *International Conference on Learning Representations (ICLR)*, 2020.

[14] Jacopo Diquigiovanni, Matteo Fontana, Simone 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.

[15] Andrija Djurisic, Nebojsa Bozanic, Arjun Ashok, and Rosanne Liu. Extremely Simple Activation Shaping for Out-of-Distribution Detection. In *International Conference on Learning Representations (ICLR)*, 2023.

[16] Xuefeng Du, Zhaoning Wang, Mu Cai, and Sharon Li. Towards Unknown-aware Learning with Virtual Outlier Synthesis. In *International Conference on Learning Representations (ICLR)*, 2022.

[17] Matt Foutter, Rohan Sinha, Somrita Banerjee, and Marco Pavone. Self-Supervised Model Generalization using Out-of-Distribution Detection. In *First Workshop on Out-of-Distribution Generalization in Robotics at CoRL 2023*, 2023.

[18] Cem Gokmen, Daniel Ho, and Mohi Khansari. Asking for Help: Failure Prediction in Behavioral Cloning Through Value Approximation. In *International Conference on Robotics and Automation (ICRA)*, pages 5821–5828. IEEE, 2023.

[19] Martin Hägele, Klas Nilsson, J Norberto Pires, and Rainer Bischoff. Industrial Robotics. *Springer Handbook of Robotics*, pages 1385–1422, 2016.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, 2016.

[21] Nantian He, Shaohui Li, Zhi Li, Yu Liu, and You He. ReDiffuser: Reliable Decision-Making Using a Diffuser with Confidence Estimation. In *International Conference on Machine Learning (ICML)*, volume 235, pages 17921–17933. PMLR, 21–27 Jul 2024.

[22] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:6840–6851, 2020.

[23] Xixi Hu, qiang liu, Xingchao Liu, and Bo Liu. AdaFlow:

Imitation Learning with Variance-Adaptive Flow-Based Policies. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[24] Arda Inceoglu, Eren Erdal Aksoy, and Sanem Sariel. Multimodal Detection and Classification of Robot Manipulation Failures. *Robotics and Automation Letters*, 2023.

[25] Masha Itkina and Mykel Kochenderfer. Interpretable Self-aware Neural Networks for Robust Trajectory Prediction. In *Conference on Robot Learning (CoRL)*, pages 606–617. PMLR, 2023.

[26] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with Diffusion for Flexible Behavior Synthesis. In *International Conference on Machine Learning (ICML)*, pages 9902–9915. PMLR, 2022.

[27] Ramneet Kaur, Kaustubh Sridhar, Sangdon Park, Yahan Yang, Susmit Jha, Anirban Roy, Oleg Sokolsky, and Insup Lee. CODiT: Conformal Out-of-Distribution Detection in Time-Series Data for Cyber-Physical Systems. In *Proceedings of the ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023)*, ICCPS '23, page 120–131, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400700361. doi: 10.1145/3576841.3585931.

[28] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An Open-Source Vision-Language-Action Model. *arXiv preprint arXiv:2406.09246*, 2024.

[29] Woo Kyung Kim, Minjong Yoo, and Honguk Woo. Robust Policy Learning via Offline Skill Diffusion. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 38, pages 13177–13184, 2024.

[30] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In Yoshua Bengio and Yann LeCun, editors, *International Conference on Learning Representations (ICLR)*, 2015.

[31] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[32] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow Matching for Generative Modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

[33] Huihan Liu, Shivin Dass, Roberto Martín-Martín, and Yuke Zhu. Model-based Runtime Monitoring with Interactive Imitation Learning. In *International Conference on Robotics and Automation (ICRA)*, pages 4154–4161. IEEE, 2024.

[34] Huihan Liu, Yu Zhang, Vaarij Betala, Evan Zhang, James Liu, Crystal Ding, and Yuke Zhu. Multi-Task Interactive Robot Fleet Learning with Visual World Models. In *Conference on Robot Learning (CoRL)*, 2024.

[35] Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based Out-of-Distribution Detection. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:21464–21475, 2020.

[36] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. In *International Conference on Learning Representations (ICLR)*, 2017.

[37] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[38] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What Matters in Learning from Offline Human Demonstrations for Robot Manipulation. In *Conference on Robot Learning (CoRL)*, 2021.

[39] Rajesh Natarajan, Santosh Reddy P, Subash Chandra Bose, H.L. Gururaj, Francesco Flammini, and Shanmugapriya 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.

[40] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An Open-Source Generalist Robot Policy. In *Robotics: Science and Systems (RSS)*, Delft, Netherlands, 2024.

[41] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning Transferable Visual Models from Natural Language Supervision. In *International Conference on Machine Learning (ICML)*, pages 8748–8763. PMLR, 2021.

[42] Quazi Marufur Rahman, Peter Corke, and Feras Dayoub. Run-Time Monitoring of Machine Learning for Robotic Perception: A Survey of Emerging Trends. *IEEE Access*, 9:20067–20075, 2021.

[43] Allen Z. Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, Zhenjia Xu, Dorsa Sadigh, Andy Zeng, and Anirudha Majumdar. Robots That Ask For Help: Uncertainty Alignment for Large Language Model Planners. In *Conference on Robot Learning (CoRL)*, 2023.

[44] Moritz Reuss and Rudolf Lioutikov. Multimodal Diffusion Transformer for Learning from Play. In *2nd Workshop on Language and Robot Learning: Language as Grounding*, 2023.

[45] Quentin Rouxel, Andrea Ferrari, Serena Ivaldi, and Jean-Baptiste Mouret. Flow matching Imitation Learning for Multi-Support Manipulation. In *International Conference on Humanoid Robots (Humanoids)*, pages 528–535.

IEEE, 2024.

[46] Murat Sensoy, Lance Kaplan, and Melih Kandemir. Evidential Deep Learning to Quantify Classification Uncertainty. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[47] Glenn Shafer and Vladimir Vovk. A Tutorial on Conformal Prediction. *Journal of Machine Learning Research*, 9(3), 2008.

[48] Rohan Sinha, Amine Elhafsi, Christopher Agia, Matt Foutter, Edward Schmerling, and Marco Pavone. Real-Time Anomaly Detection and Reactive Planning with Large Language Models. In *Robotics: Science and Systems (RSS)*, 2024.

[49] Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. Nomad: Goal Masked Diffusion Policies for Navigation and Exploration. In *International Conference on Robotics and Automation (ICRA)*, pages 63–70. IEEE, 2024.

[50] Jiankai Sun, Yiqi Jiang, Jianing Qiu, Parth Talpur Nobel, Mykel Kochenderfer, and Mac Schwager. Conformal Prediction for Uncertainty-Aware Planning with Diffusion Dynamics Model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[51] Leitian Tao, Xuefeng Du, Jerry Zhu, and Yixuan Li. Non-parametric Outlier Synthesis. In *International Conference on Learning Representations (ICLR)*, 2023.

[52] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*, volume 29. Springer, 2005.

[53] Apoorv Vyas, Nataraj Jammalamadaka, Xia Zhu, Dipankar Das, Bharat Kaul, and Theodore L Willke. Out-of-Distribution Detection Using an Ensemble of Self Supervised Leave-out Classifiers. In *European Conference on Computer Vision (ECCV)*, pages 550–564, 2018.

[54] Dian Wang, Stephen Hart, David Surovik, Tarik Kelestemur, Haojie Huang, Haibo Zhao, Mark Yeatman, Jiuguang Wang, Robin Walters, and Robert Platt. Equivariant Diffusion Policy. In *Conference on Robot Learning (CoRL)*, 2024.

[55] Yanwei Wang, Tsun-Hsuan Wang, Jiayuan Mao, Michael Hagenow, and Julie Shah. Grounding Language Plans in Demonstrations Through Counterfactual Perturbations. In *International Conference on Learning Representations (ICLR)*, 2024.

[56] Yixuan Wang, Guang Yin, Binghao Huang, Tarik Kelestemur, Jiuguang Wang, and Yunzhu Li. GenDP: 3D Semantic Fields for Category-Level Generalizable Diffusion Policy. In *Conference on Robot Learning (CoRL)*, 2024.

[57] Josiah Wong, Albert Tung, Andrey Kurenkov, Ajay Mandlekar, Li Fei-Fei, Silvio Savarese, and Roberto Martín-Martín. Error-Aware Imitation Learning from Teleoperation Data for Mobile Manipulation. In *Conference on Robot Learning (CoRL)*, pages 1367–1378. PMLR, 2022.

[58] Chen Xu and Yao Xie. Conformal Prediction for Time Series. *Transactions on Pattern Analysis and Machine Intelligence*, 45(10):11575–11587, 2023.

[59] Chen Xu and Yao Xie. Sequential Predictive Conformal Inference for Time Series. In *International Conference on Machine Learning (ICML)*, pages 38707–38727. PMLR, 2023.

[60] Chen Xu, Xiuyuan Cheng, and Yao Xie. Normalizing Flow Neural Networks by JKO Scheme. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[61] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized Out-of-Distribution Detection: A Survey. *International Journal of Computer Vision*, pages 1–28, 2024.

[62] Ling Yang, Zixiang Zhang, Zhilong Zhang, Xingchao Liu, Minkai Xu, Wentao Zhang, Chenlin Meng, Stefano Ermon, and Bin Cui. Consistency Flow Matching: Defining Straight Flows with Velocity Consistency. *arXiv preprint arXiv:2407.02398*, 2024.

[63] Tianhe Yu, Ted Xiao, Jonathan Tompson, Austin Stone, Su Wang, Anthony Brohan, Jaspiar Singh, Clayton Tan, Dee M, Jodilyn Peralta, Karol Hausman, Brian Ichter, and Fei Xia. Scaling Robot Learning with Semantically Imagined Experience. In *Robotics: Science and Systems (RSS)*, 2023.

[64] Qinglun Zhang, Zhen Liu, Haoqiang Fan, Guanghui Liu, Bing Zeng, and Shuaicheng Liu. FlowPolicy: Enabling Fast and Robust 3D Flow-based Policy via Consistency Flow Matching for Robot Manipulation. *arXiv preprint arXiv:2412.04987*, 2024.

[65] Tony Z. Zhao, Jonathan Tompson, Danny Driess, Pete Florence, Seyed Kamyar Seyed Ghasemipour, Chelsea Finn, and Ayzaan Wahid. ALOHA Unleashed: A Simple Recipe for Robot Dexterity. In *Conference on Robot Learning (CoRL)*, 2024.

APPENDIX

A. 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 policy (DP) [12] using diffusion models [22] has emerged as highly performant in this space. DP learns to denoise trajectories sampled from a Gaussian distribution, effectively capturing the multi-modal action distributions often present in human demonstrations [63, 12]. Diffusion models have been used to learn observation-conditioned policies [12, 49], integrate semantic information via language conditioning [63, 10, 44], and improve robustness and generalization [29, 54, 56]. Concurrently, vision-language-action models like Octo [40] and OpenVLA [28] have shown promise in generalist robot manipulation by leveraging large-scale pretraining on diverse datasets. More recently, flow matching (FM) generative models have been proposed as an alternative to diffusion in imitation learning [4, 23, 6, 64], offering faster inference and greater flexibility (i.e., extending beyond Gaussian priors [9]), while achieving competitive or superior success rates. We test FAIL-Detect on DP and FM imitation learning architectures.

OOD Detection. The task of detecting robot failures can be viewed as anomaly detection, which falls under the broader framework of OOD detection [61]. Ensemble methods [31], which combine predictions from multiple models to improve robustness and estimate uncertainty, have long been regarded as the de facto approach for addressing this problem. However, they are computationally expensive as they require training and running inference on multiple models. Another popular approach frames OOD detection as a classification problem [34]. This formulation learns the decision boundary between ID and OOD data by training a binary classifier [53, 15], but requires OOD data during training. In contrast, density-based approaches [35, 16, 60], one-class discriminators based on random networks [13, 21], and control-theoretic methods [7] aim to model information from ID data without relying on OOD data during training. Density-based methods attempt to capture the distribution of ID data, yet they can be challenging to optimize. One-class discriminators have shown superior performance over deep ensembles in practice but can be sensitive to the design of the discriminator model. Control-theoretic approaches use contrastive energy-based models; however, they often require a representation of the system's dynamics. Furthermore, evidential deep learning methods [8, 5, 25] learn parameters for second-order distributions (e.g., Dirichlet) to approximate epistemic uncertainty (due to limited model knowledge or OOD inputs) from aleatoric uncertainty (due to inherent randomness in the data). Lastly, distance-based approaches [3, 51, 27] identify OOD samples by computing their distance to ID samples in the input or latent space, avoiding the need for training but exhibiting limited performance compared to other approaches. We consider many of the listed model variants as score candidates in FAIL-Detect.

Failure Detection in Robotics. Detecting failures in robotic systems is important for ensuring safety and reliability, as failures can lead to undesirable behaviors in human environments [39, 42, 34]. Various approaches have been proposed, such as building fast anomaly classifiers based on LLM embeddings [48] and using the reconstruction error from variational autoencoders (VAE) to detect anomalies in behavior cloning (BC) policies for mobile manipulation [57]. Separately, Ren et al. [43] construct uncertainty sets from conformal prediction for actions generated by an LLM-based planner, prompting human intervention when the set is ambiguous. These works do not consider failure detection in the setting of generative imitation learning policies. On the other hand, Gokmen et al. [18] learn a state value function that is trained jointly with a BC policy and can be used to predict failures. Liu et al. [33] propose an LSTM-based failure classifier for a BC-RNN policy using latent embeddings from a conditional VAE. Given a Transformer-based policy and a world model to predict future latent embeddings, Liu et al. [34] train a failure detection classifier on the embeddings. To handle previously unseen states, they also propose a SOTA OOD detection method, which we adapt as a baseline to our approach (see PCA-kmeans in Appendix D). However,

TABLE I: Overview of score methods evaluated in this work. The input was selected either based on the structure and requirements of each method or, when multiple input combinations were possible, based on empirical performance. All methods except STAC (which proposes a different calibration method; see Appendix D) use time-varying CP bands described in Appendix C.

Method	Type	Input	Category	Novelty	Original application
logpZO	Learned	O_t	Density estimation	Novel	N/A
logpO	Learned	O_t	Density estimation	Adapted [60]	Likelihood estimation on tabular data
NatPN	Learned	O_t	Second-order	Adapted [8]	OOD detection for classification and regression
DER	Learned	(O_t, A_t)	Second-order	Adapted [5]	OOD detection for human pose estimation
RND	Learned	(O_t, A_t)	One-class discriminator	Adapted [21]	Reinforcement learning [21]; OOD detetcion [13]
CFM	Learned	O_t	One-class discriminator	Adapted [62]	Efficient sampling of flow models
SPARC	Post-hoc	A_t	Smoothness measure	Adapted [2]	Smoothness analysis for time series data
STAC	Post-hoc	A_t	Statistical divergence	Baseline [1]	Failure detection for generative imitation learning policies
PCA-kmeans	Post-hoc	O_t	Clustering	Baseline [34]	OOD detection during robot execution

unlike FAIL-Detect, these methods require collecting failed trajectories a priori to detect failures. Meanwhile, Wang et al. [55] uses self-reset to collect additional failure data and train a classifier to identify failure modes. In their on-robot experiments, approximately 2000 trajectories (roughly 2 hours) had to be collected using self-reset, making scalability challenging. For diffusion-based policies, Sun et al. [50] reduce model uncertainty by producing prediction intervals for rewards of predicted trajectories. He et al. [21] propose using random network distillation (RND) to detect OOD trajectories and select reliable ones. These works do not directly consider runtime failure detection. Our two-stage solution for this problem combines the advantages of both approaches. The closest SOTA method to FAIL-Detect by Agia et al. [1] introduces a statistical temporal action consistency (STAC) measure in conjunction with vision-language models (VLMs) to detect failures within rollouts at runtime. STAC does not require failure data, consists of a score computed post-hoc from a batch of predicted actions and a constant-time CP threshold to flag failures, and is evaluated in the context of DP. We demonstrate improved empirical performance over STAC by integrating *learned* failure detection scores with a *time-varying* CP band.

B. Design of Scalar Scores

To construct scores indicative of failures, we propose a novel score candidate and several adaptations of existing approaches originally developed for other applications. See Table I for an overview of the scoring methods we consider.

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 manipulation. **(3) Discriminative**: The method should yield gaps in scores for successful and failed rollouts. To avoid overfitting on historical data, the score network D_M only takes the latest T_O steps ($T_O = 2$ following [12]) of past observations O_t alongside future action A_t as inputs, rather than the growing trajectory history. To meet our desiderata, we select and build on the following approach categories.

(a) Learned data density: we fit a normalizing flow-based density estimator to the observations, where data far from the distribution of successful trajectory observations may indicate failure. The approach we term lopO [60] fits a continuous normalizing flow (CNF) f_θ to the set of observations $\{O_t\}_{t\geq 0}$. A low $\log p(O_{t'})$ for a new observation $O_{t'}$ implies it is unlikely, indicating possible failure. Note the computation of $\log p(O_{t'})$ requires integration of the divergence of f_θ over the ODE trajectory, which is difficult to estimate in high dimensions. Additionally, we introduce our novel logpZO approach, which leverages the same CNF f_θ to evaluate the likelihood of a noise estimate Z_{O_t} (conditioned on an observation O_t). Using the forward ODE process, we compute Z_{O_t} by integrating f_θ over the unit interval [0,1], starting from O_t as the ODE initial condition. When O_t is ID, Z_{O_t} is approximately Gaussian, leading to $p(Z_{O_t}) = C\exp(-0.5|Z_{O_t}|^2)$. Thus, a high value of $|Z_{O_t}|_2^2$ corresponds to a low likelihood $p(Z_{O_t})$ in the noise space. More precisely, we explain how the proposed novel logpZO works step by step:

- **Step 1:** Fit a flow matching model f_θ between observations $\{O_t\}$ (i.e., image embeddings and proprioception) and latent noise $\{Z\} \sim \mathcal{N}(0, I)$, so that for $s \in [0, 1]$:

$$f_\theta(O_t[s], s) \approx Z - O_t,$$
$$O_t[s] = O_t + s(Z - O_t).$$

- **Step 2:** Given a new observation $O_{t'}$ at time step t', perform one-step prediction to obtain latent noise estimate:

$$Z_{O_{t'}} = O_{t'} + f_\theta(O_{t'}, 0).$$

  When $O_{t'}$ is in-distribution, by the flow matching formulation, $Z_{O_{t'}}$ is close to samples drawn from $\mathcal{N}(0, I)$.
- **Step 3:** Compute density of latent noise (up to a constant) using squared norm:

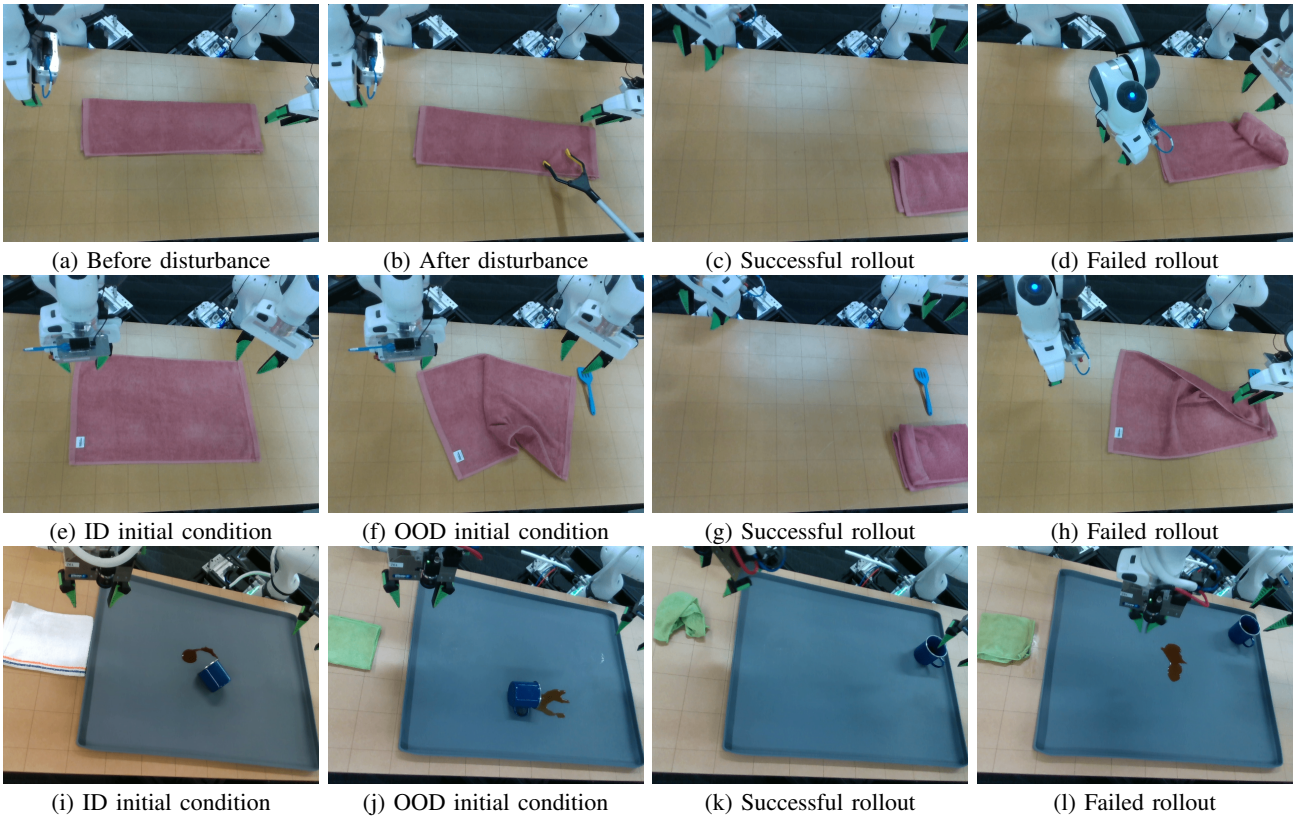
$$\log p(Z_{O_{t'}}) \propto -\|Z_{O_{t'}}\|_2^2$$

  High norm values of $\|Z_{O_{t'}}\|_2^2$ indicate lower likelihood, which is caused by anomalous observations $O_{t'}$. We thus use $\|Z_{O_{t'}}\|_2^2$ as the logpZO score.

The key distinction between lopO and logpZO lies in their domains: the former assesses likelihood in the original observation space, while the latter does so in the latent noise space. We expect the latter to be better because its computation does not require the divergence of f_θ integrated over $[0, 1]$, a hard-to-estimate quantity in high dimensions.

(b) Second-order: these methods learn parameters for second-order distributions that can separate aleatoric and epistemic uncertainty [46]. NatPN [8] imposes a Dirichlet prior on class probabilities and optimizes model parameters by minimizing a Bayesian loss. To use NatPN, we discretize the observations O_t using K-means and apply NatPN to the discretized version. We also consider multivariate deep evidential regression DER [5], which assumes $A_t|O_t$ follows a multivariate Gaussian distribution with a Wishart prior and learns its parameters.

(c) One-class discriminator: we consider methods that learn a continuous metric, but do not directly model the distribution of input data. The one-class discriminator RND [21] initializes random target $f_T(\cdot)$ and random predictor $f(\cdot;\theta)$ networks.



(a) Before disturbance (b) After disturbance (c) Successful rollout (d) Failed rollout

(e) ID initial condition (f) OOD initial condition (g) Successful rollout (h) Failed rollout

(i) ID initial condition (j) OOD initial condition (k) Successful rollout (l) Failed rollout

Fig. 6: Robot hardware experiment scenarios. **(Top row) FoldRedTowel** with Disturbance: In (b), the human pulls the towel from the position in (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). **(Middle row) FoldRedTowel** OOD: 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). **(Bottom row) CleanUpSpill** OOD: Compared to ID (i), we start with a green towel as in (j). The training data only contains white and gray towels and, therefore, although the task could succeed as in (k), the robot typically fails like in (l) with a low success rate.

The target is frozen, while 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 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) [62], 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 data is OOD.

(d) Post-hoc metrics: we investigate methods that compute a scalar score analytically without learning. We use SPARC [2] 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 [1], falls in the post-hoc method category. However, since it comes with its own statistical evaluation procedure, we describe it as one of our main baselines in Appendix D. Additionally, we term the OOD detection method by Liu et al. [34] as PCA-kmeans, which also falls in this category. We retrofit it within our two-stage framework as another baseline in Appendix D.

C. Sequential Threshold Design with Conformal Prediction

We design a time-varying threshold η_t such that a failure is flagged when $D_M(A_t,O_t;\theta)$ exceeds η_t. To do so, we leverage functional CP [14], a framework that wraps around a time series of any scalar score $D_M(A_t,O_t;\theta)$ (higher indicates failure) and yields a distribution-free prediction band C_α with user-specified significance level $\alpha \in (0,1)$. Under mild conditions [52, 58, 59], C_α contains any ID score $D_M(A_t,O_t;\theta)$ with probability of at least $1-\alpha$ for the entire duration of the rollout. If $D_M(A_t,O_t;\theta) \notin C_\alpha$, we can confidently reject that (A_t,O_t) 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 $D_M(A_t,O_t;\theta)$, which indicate the trajectory is OOD (i.e., a failure). Given N successful rollouts as the calibration data, we obtain scalar scores $\mathcal{D}_{cal} = \{D_M(A_t^i,O_t^i;\theta) : i = 1,\dots,N \text{ and } t = 1,H',\dots,T\}$. The CP band is a set of intervals $C_\alpha = \{[\text{lower}_t, \text{upper}_t] : t = 1,H',\dots,T\}$, where $\text{lower}_t \equiv \min(\mathcal{D}_{cal})$

since the band is one-sided. To obtain the upper bound, we follow [14], computing the time-varying mean μ_t and band width h_t, so that $\text{upper}_t = \mu_t + h_t$.

More precisely, following [14], 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 $D_j = \max(\{(\mu_t - D_M(A_t^j, O_t^j; \theta))/s_{cal_A}(t)\}_{t=1}^{T})$, which is the max deviation over rollout length from the mean prediction to the scalar score. The function $s_{cal_A}(t)$ is called a "modulation" function that depends on the dataset $\mathcal{D}_{cal_A}$. In our experiment, we consider either

$$s_{cal_A}(t) = 1/T \tag{1}$$

$$s_{cal_A}(t) = \max_{k \in \mathcal{H}} |D_M(A_t^k, O_t^k, \theta) - \mu_t|, \tag{2}$$

where $\mathcal{H} = [N_1]$ if $(N_1 + 1)(1 - \alpha) > N_1$, otherwise $\mathcal{H} = \{k \in [N_1] : \max_{t \in [T]} |D_M(A_t^k, O_t^k, \theta) - \mu_t| \leq \gamma\}$ for $\gamma = (1 - \alpha)$-quantile of $\{\max_{t \in [T]} |D_M(A_t^m, O_t^m, \theta) - \mu_t|\}_{m=1}^{N_1}$. Intuitively, Eq. (2) adapts the width of prediction bands based on the non-extreme behaviors of the functional data. It does so by minimizing the influence of outliers whose maximum absolute residuals lie within the upper α quantile of all maximum values. Additionally, 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 $\text{upper}_t = \mu_t + h s_{cal_A}(t)$. We pick $\alpha = 0.05$ (or 95% confidence interval) throughout experiments (see Table Table III on hyperparameter choices).

Theoretically, for a new successful 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 to one, the decision rule $\mathbb{1}(D_M(A_t, O_t; \theta) > \eta_t)$ controls the false positive rate (successes marked as failures) at level α.

D. Experiments

We test our two-stage failure detection framework in both simulation and on robot hardware. 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 post-hoc scalar scores (see Table I) within our FAIL-Detect framework (see results in Section III).

a) Task descriptions: In simulation, we consider the **Square**, **Transport**, **Can**, and **Toolhang** tasks from the open-source Robomimic benchmark[1] [38]. In the robot hardware experiments, we consider two tasks on a bimanual Franka Emika Panda robot station that are significantly more challenging: **FoldRedTowel** and **CleanUpSpill** (see Fig. 7).

Specifically, we have

- **(Simulation)** The tasks from the Robomimic benchmark [38] are as follows. 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. The **Can** task asks the robot to place a coke can from a large bin into a smaller target bin, requiring greater precision than the **Square** task. The **Toolhang** task asks the robot to assemble a frame with several components, requiring the most dexterity and precision among the four tasks.
- **(Real tasks)** In the **FoldRedTowel** task, the two robot arms must fold a red towel twice and push it to the table corner. In the **CleanUpSpill** task, one robot arm must lift a cup upright that has fallen and caused a spill, while the other robot

[1]We omit the **Lift** task as both FM and DP policies achieve 100% success.

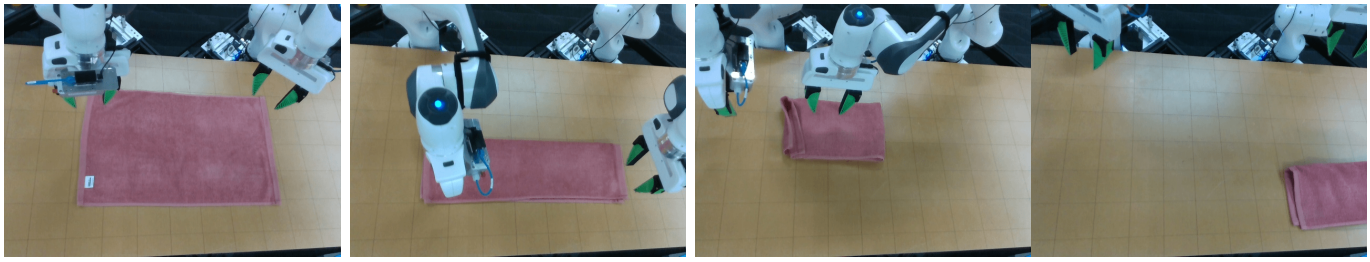
TABLE II: Success rate of the flow policy on test data in each task-environment combination. These test data is used to test failure detection methods as well. On the real task, we mark some cells with * when the number of failures out of test rollouts is no greater than 5. In such cases, we shuffle the rollout indices and include all the failure ones in the test set, so that the failure detection metrics have higher statistical significance. Across the entire 50 rollouts, the true success rate of FM policy on FoldRedTowel ID is 0.96, and that of DP on CleanUpSpill ID is 0.82.

(a) Simulation tasks

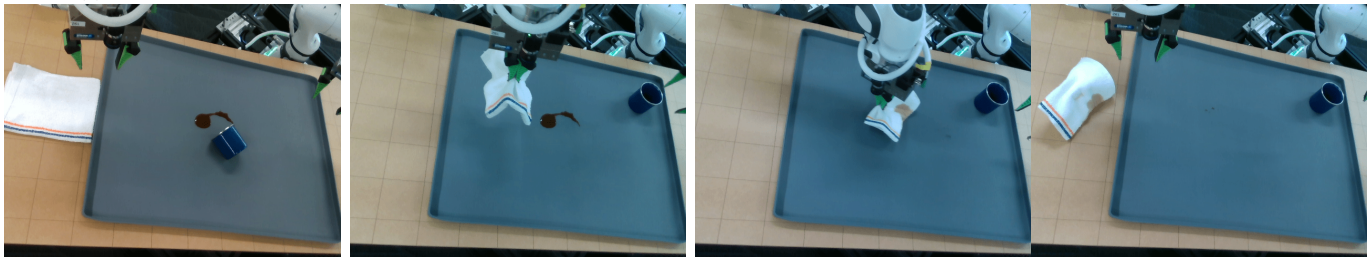
	Square ID	Square OOD	Transport ID	Transport OOD	Can ID	Can OOD	Toolhang ID	Toolhang OOD
FM Policy	0.90 (1000 rollouts)	0.63 (2000 rollouts)	0.85 (1000 rollouts)	0.63 (2000 rollouts)	0.98 (1000 rollouts)	0.84 (2000 rollouts)	0.77 (1000 rollouts)	0.53 (2000 rollouts)
DP	0.93 (125 rollouts)	0.63 (250 rollouts)	0.84 (125 rollouts)	0.76 (250 rollouts)	0.98 (125 rollouts)	0.95 (250 rollouts)	0.82 (125 rollouts)	0.54 (250 rollouts)

(b) Robot hardware tasks

	FoldRedTowel ID via FM policy	FoldRedTowel ID + Disturb via FM policy	FoldRedTowel OOD via FM policy	CleanUpSpill ID via DP	CleanUpSpill OOD via DP	CleanUpSpill ID via FM policy	CleanUpSpill OOD via FM policy
Setting-dependent band	0.9* (20 rollouts)	0.75* (20 rollouts)	0.60 (20 rollouts)	0.70 (20 rollouts)	0.45 (20 rollouts)	0.90* (20 rollouts)	0.90* (20 rollouts)
ID-only band	0.9* (20 rollouts)	0.90 (50 rollouts)	0.58 (50 rollouts)	0.70 (20 rollouts)	0.52 (50 rollouts)	0.90* (20 rollouts)	0.76 (50 rollouts)



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



(e) Initial condition (f) About to wipe (g) Wiping (h) Final success

Fig. 7: The on-robot experimental settings. **(Top row) FoldRedTowel**: 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. **(Bottom row) CleanUpSpill**: starting with spills caused by a fallen cup on the platform, the right arm must first lift the cup to an upright position, while the left arm must pick up a towel and wipe the spills. To achieve success, the spills must be completely cleaned, and the towel must be returned to its original position.

arm must pick up a white towel and wipe the spills on the platform. Both tasks are long-horizon and require precision and coordination to manipulate deformable objects.

We construct OOD settings for each task. In simulation, we adjust the third-person camera $10\,\mathrm{cm}$ upwards at the first time step after $t = 50$ to simulate a camera bump mid-rollout[2]. For the on-robot **FoldRedTowel** task, we disturb the task after the first fold (challenging ID scenario) and create an OOD initial condition by crumpling the towel (seen in less than $\sim$15% of the data) and adding a never before seen distractor (blue spatula). For the **CleanUpSpill** task, we create an OOD initial condition by changing the towel to a novel green towel (see Fig. 6).

b) Baselines: We baseline FAIL-Detect against STAC [1] and PCA-kmeans [34] as SOTA approaches in success-based failure detection for generative imitation learning policies. STAC operates by generating batches (e.g., 256) of 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, the 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 calibration 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 as real-time feasible as possible. Due to the long STAC inference time (even after parallelization) and resulting high system latency, we omit its comparison on the two robot hardware tasks. In our second baseline, Liu et al. [34] tackle failure detection by training a failure classifier, which requires the collection of failure training data. However, this approach is not applicable to our setup as we assume access to only successful human demonstrations for training and successful rollouts for calibration. Instead, we incorporate their proposed OOD detection method as a post-hoc scalar score in the first stage of FAIL-Detect to construct a fair baseline. We use the performant time-varying CP band to obtain thresholds in the second stage. The method measures the distance of a new observation $O_{t'}$ at test time index t' from the set of training data $\{O_t\}_{t\geq 0}$, which consist of visual encoded features jointly trained with the policy on the demonstration data. PCA-kmeans first uses PCA to embed the training features and then applies K-means clustering to the embedded data to obtain $K = 64$ centroids. After embedding $O_{t'}$ using the same principal components, the method computes the smallest Euclidean distance between the embedding and the K centroids. This distance serves as the OOD metric (higher values indicate greater OOD). We omit comparison against ensembles [31], a popular OOD detection technique, due to RND having shown improved performance over ensembles in prior work [13] and their prohibitively high computational cost.

[2]We use $t = 15$ for **Can**, which has the shortest task completion time.

TABLE III: Hyperparams in evaluation protocol. We include the details for the policy networks in Table IIIa and the hyperparameters for CP band calibration in simulation in Table IIIb and in robot hardware experiments in Tables IIIc and IIId. For simulation tasks, training is done on one NVIDIA RTX A6000 GPU with 48GB memory. For the experiments on hardware, training is done on eight NVIDIA A100-SXM4-80GB GPUs with 80GB memory.

(a) Policy network

	Dimension of (A_t, O_t)	Architecture of (policy g, visual encoder for O_t)	Policy training specification (optimizer, lr, lr scheduler, batch size, number of epochs)
(Simulation) Square	(160, 274)	(UNet [26], ResNet [20])	(AdamW [37], 1e-4, cosine [36], 64, 800)
(Simulation) Can	(160, 274)	(UNet [26], ResNet [20])	(AdamW [37], 1e-4, cosine [36], 64, 800)
(Simulation) Toolhang	(160, 274)	(UNet [26], ResNet [20])	(AdamW [37], 1e-4, cosine [36], 64, 300)
(Simulation) Transport	(320, 548)	(UNet [26], ResNet [20])	(AdamW [37], 1e-4, cosine [36], 64, 300)
(Real) FoldRedTowel	(320, 4176)	(UNet [26], ResNet [20])	(AdamW [37], 1e-4, cosine [36], 96, 1000)
(Real) CleanUpSpill	(320, 6732)	(UNet [26], CLIP [41])	(AdamW [37], 1e-4, cosine [36], 36, 500)

(b) (Simulation) CP band calibration and testing

	Square ID	Square OOD	Can ID	Can OOD	Toolhang ID	Toolhang OOD	Transport ID	Transport OOD
CP band modulation	Eq. (2)	Eq. (2)	Eq. (2)	Eq. (2)	Eq. (2)	Eq. (2)	Eq. (2)	Eq. (2)
CP significance level	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
(FM policy) Num successes for CP band mean	269	269	296	296	224	224	253	253
(FM policy) Num successes for CP band width	629	629	693	693	523	523	591	591
(FM policy) Num test rollouts for evaluation	1000	2000	1000	2000	1000	2000	1000	2000
(DP) Num successes for CP band mean	31	31	34	34	27	27	27	27
(DP) Num successes for CP band width	82	82	90	90	70	70	71	71
(DP) Num test rollouts for evaluation	125	250	125	250	125	250	125	250

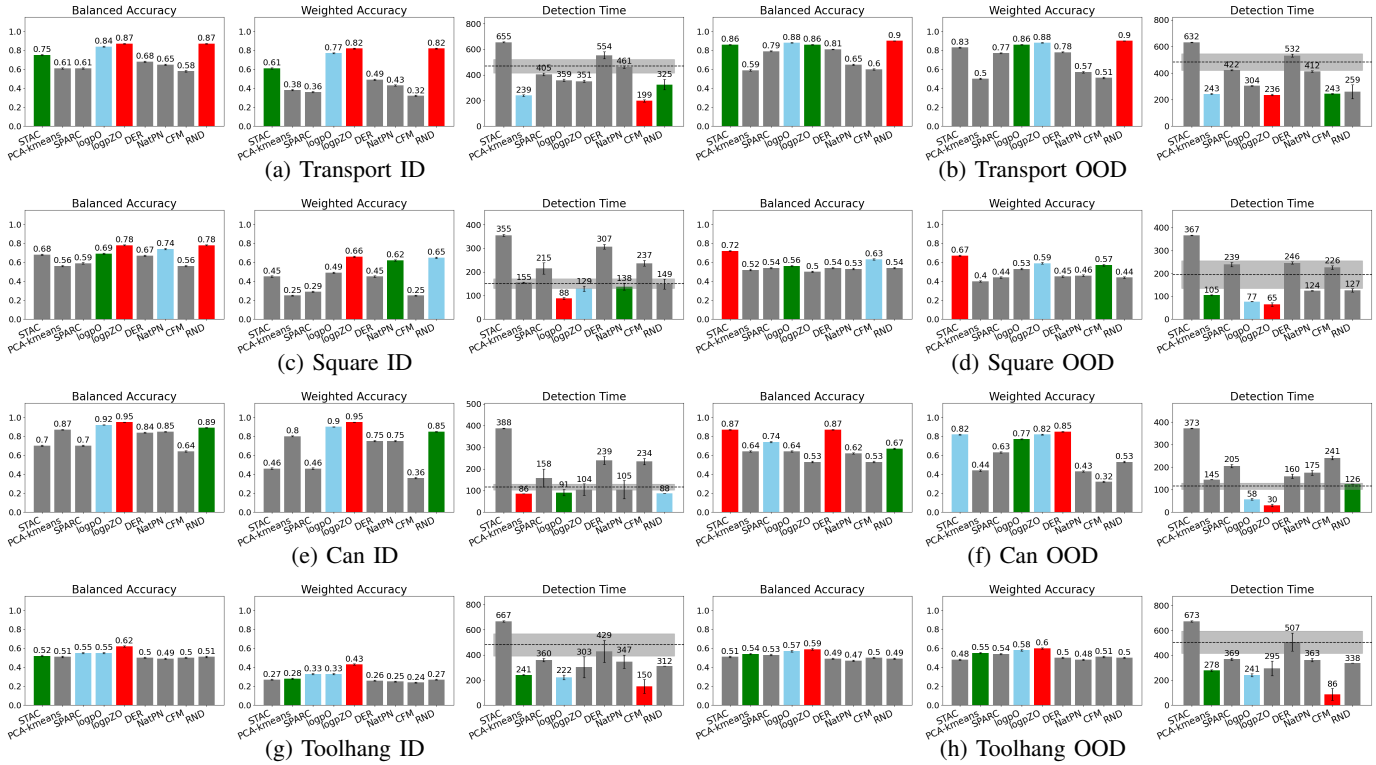
(c) (Hardware: FoldRedTowel) CP band calibration and testing

	ID Disturb (Setting-dependent)	ID Disturb (ID-only)	OOD (Setting-dependent)	OOD (ID-only)
CP band modulation	Eq. (1)	Eq. (1)	Eq. (1)	Eq. (1)
CP significance level	0.05	0.05	0.05	0.05
Num successes for CP band mean	7	7	4	7
Num successes for CP band width	23	23	13	23
Num test rollouts for evaluation	20	50	20	50

(d) (Hardware: CleanUpSpill) CP band calibration and testing

	OOD (DP) (Setting-dependent)	OOD (DP) (ID-only)	OOD (FM policy) (Setting-dependent)	OOD (FM policy) (ID-only)
CP band modulation	Eq. (2)	Eq. (2)	Eq. (1)	Eq. (1)
CP significance level	0.05	0.05	0.05	0.05
Num successes for CP band mean	6	5	6	9
Num successes for CP band width	14	12	14	18
Num test rollouts for evaluation	20	50	20	50

c) Evaluation Protocol: To quantify failure detection performance, we denote failed rollouts as one and successful rollouts as zero. We then adopt the following standard metrics: **(1)** true positive rate (TPR), **(2)** true negative rate (TNR), **(3)** balanced



(a) Transport ID (b) Transport OOD (c) Square ID (d) Square OOD (e) Can ID (f) Can OOD (g) Toolhang ID (h) Toolhang OOD

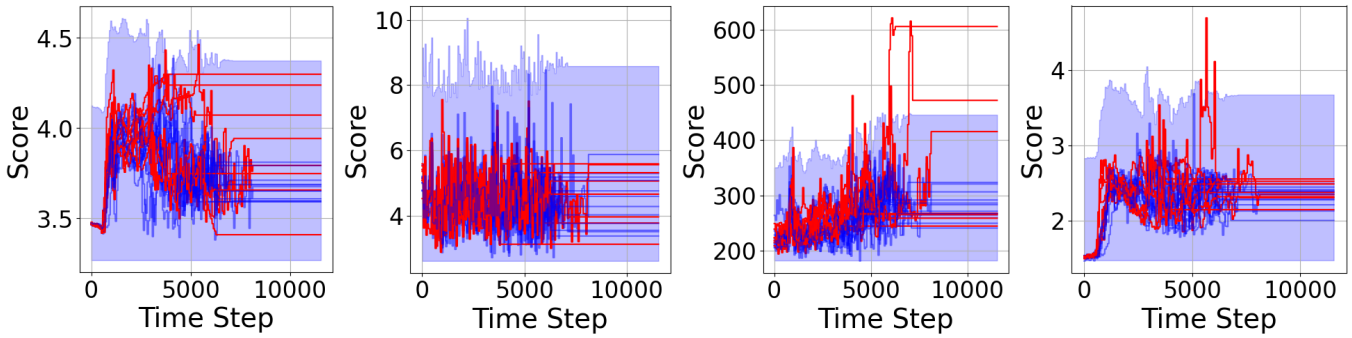
Fig. 8: Quantitative failure detection results for simulation tasks on FM policy (best, second, third); results with TPR and TNR are in Fig. 11 and results on DP are in Fig. 12. For balanced accuracy and weighted accuracy, higher is better and for detection time, lower is better. The CP band for each task is calibrated with successful rollouts under ID initial conditions only (i.e., the same band is used for ID and OOD test cases). We group together post-hoc (STAC, PCA-kmeans, SPARC), density-based (logpO, logpZO), second-order (DER, NatPN), and one-class (CFM, RND) methods and show barplots with standard errors. The dashed line in the Detection Time plots represents the average successful trajectory time in that setting with standard error. Overall, learned methods outperform post-hoc ones in failure detection. In terms of **combined accuracy** (balanced accuracy and weighted accuracy), logpZO and RND are the best two methods, reaching top-1 performance in 10/16 and 5/16 cases, respectively. Moreover, logpZO reaches top-3 performance in 14/16 cases, while RND does so in 9/16 cases. In comparison, the baselines STAC and PCA-kmeans reach top-1 performance in 3/16 and 0/16 cases, respectively. Note that STAC reaches top-3 performance in 8/16 cases, while PCA-kmeans does so in 3/16 cases. The learned methods also achieve the fastest **detection time**, with one of the learned methods always getting the best overall detection time in all but one case. In terms of best top-1 performance, logpZO is the fastest method in 3/8 cases, RND in 0/8 cases, and the PCA-kmeans baseline does so in 1/8 cases. In contrast, STAC is the slowest in nearly all cases, detecting failures only after the average success trajectory time, rendering the detection not practical.

accuracy = (TPR + TNR) / 2, **(4)** weighted accuracy = β·TPR + $(1-\beta)\cdot$ TNR for $\beta = \frac{\#\text{Successful rollouts}}{\#\text{Rollouts}}$, and **(5)** 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 from the start of the rollout. The balanced accuracy metric equally represents classes in an imbalanced dataset (e.g., few successful rollouts in an OOD setting). Weighted accuracy represents how well a method matches the true success / failure distribution. Due to the high human time cost of performing real-robot rollouts, we evaluate FAIL-Detect and the baselines on significantly fewer rollouts in the robot hardware tasks (i.e., 50 rollouts) compared to the simulation tasks (i.e., 2000 rollouts).

d) Policy backbone and the calibration of CP bands: Table II shows success rate across the tasks. Meanwhile, See Table III for hyperparameters regarding

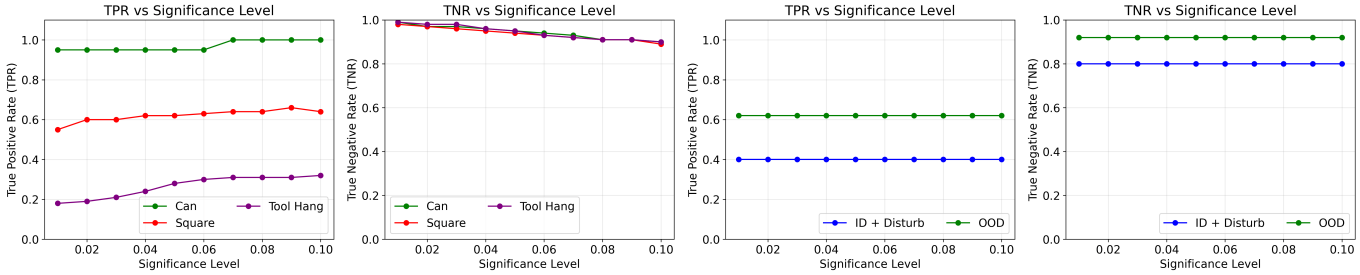
- Dimension of actions A_t and observations O_t per task.
- Architecture details of the policy backbone g and the choice of image encoder.
- Training specifics of g (i.e., optimizer, learning rate and scheduler, and number of epochs).
- Number of successful rollouts used to calibrate the CP bands and the number of test rollouts. Note that on simulation tasks, we roll out DP fewer times because it requires significantly longer time (higher number of denoising steps) than FM policies to generate actions.

We further explain the design and training of the policy network g. The underlying policy network g is trained with flow matching [32] and/or diffusion models [22]. We follow the setup in [12] and use the same hyperparameters to train the policies. When using flow matching [32] 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. Image features are extracted using either a ResNet or a CLIP backbone trained jointly with g. These image features concatenated with robot state constitute



(a) PCA-kmeans (b) SPARC (c) logpZO (d) RND

Fig. 9: Qualitative results of detection scores overlaid with CP bands on the real **FoldRedTowel OOD** task. The layout is the same as Fig. 4. We notice that spikes of scores computed on failed trajectories are more evident for the learnt logpZO and RND than for the post-hoc PCA-kmeans and SPARC.



(a) Simulation (b) Robot hardware (**FoldRedTowel**)

Fig. 10: TPR and TNR vs. CP significance level in simulation and real tasks.

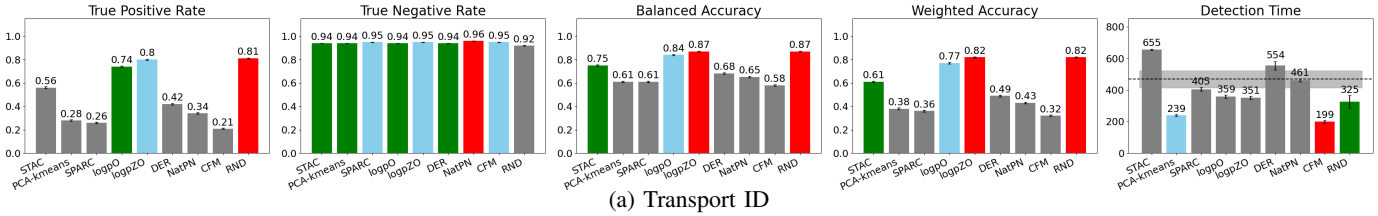
observations O_t.

e) Training 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:

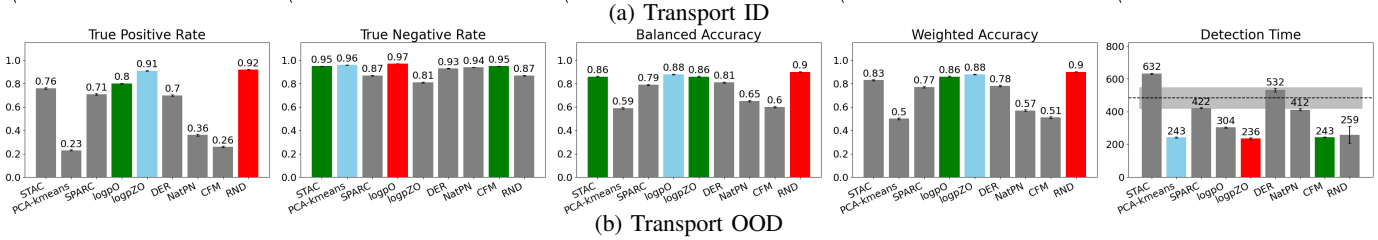
1) CFM: We use a 4x smaller network with identical architecture as the policy network. It is unconditional and takes in observations O_t as inputs. We train for 200 epochs with a batch size of 128, using the Adam optimizer [30] with a constant 1e-4 learning rate.
2) lopO and logpZO: We let the flow network (taking O_t as inputs) has the same architecture as the policy network. On simulation, we let the flow network to be 4x smaller than the policy network with identical architecture and on real data, we keep identical model sizes between the two. On simulation (resp. real data), we train for 500 (resp. 2000) epochs with a batch size of 128 (resp. 512), 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 [11].
3) DER: The network to parametrize the Normal-Inverse Wishart parameters has the same architecture as the policy network but is 4x smaller with identical architecture. 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.
4) NatPN: We first use K-means clustering with 64 clusters 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 [8] 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 with identical architecture 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.

E. Ablation

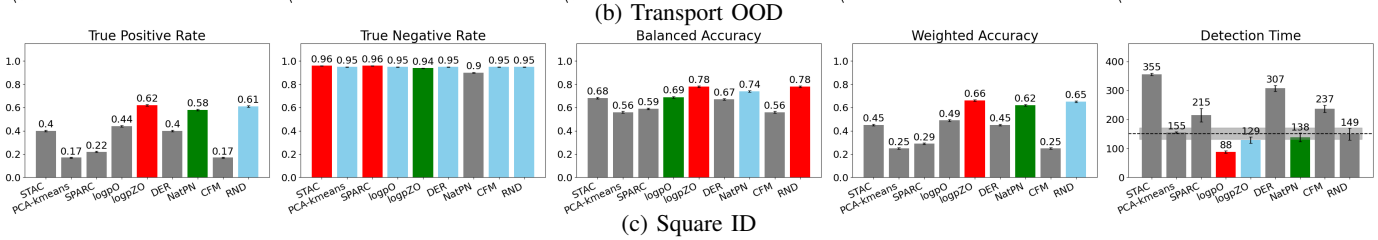
We conduct ablation studies on the behavior of our method under varying CP significance levels α. In Fig. 10, we show TPR and TNR for 10 equally spaced values of $\alpha \in [0.01, 0.1]$ using logpZO. As expected, higher α increases TPR and decreases TNR, since more rollouts are flagged as failures. This trend is clearer in simulation; in real tasks, the effect is muted due to the limited number of rollouts and therefore constant calibration quantiles for small α. Overall, $\alpha = 0.05$ offers a robust trade-off, which is what we used in all experiments.



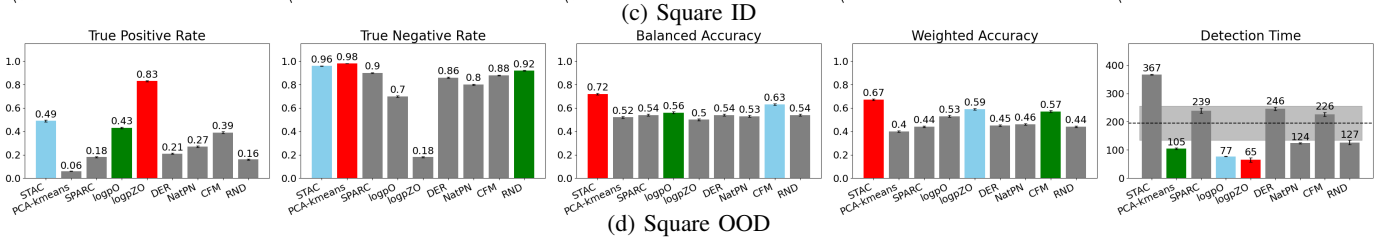
(a) Transport ID



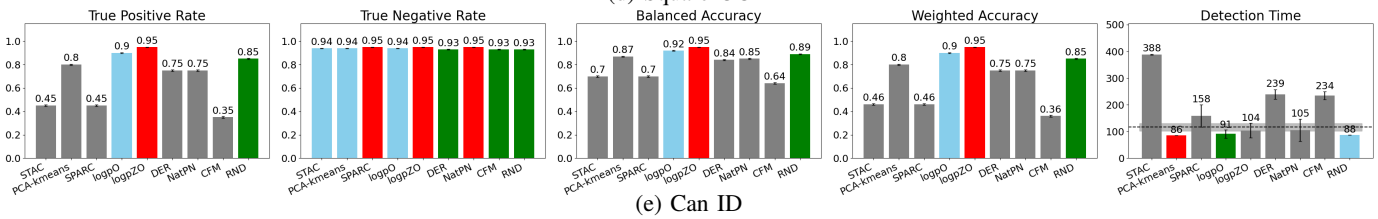
(b) Transport OOD



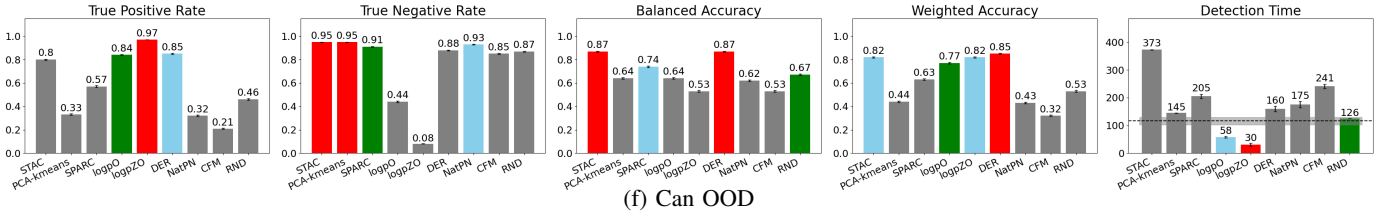
(c) Square ID



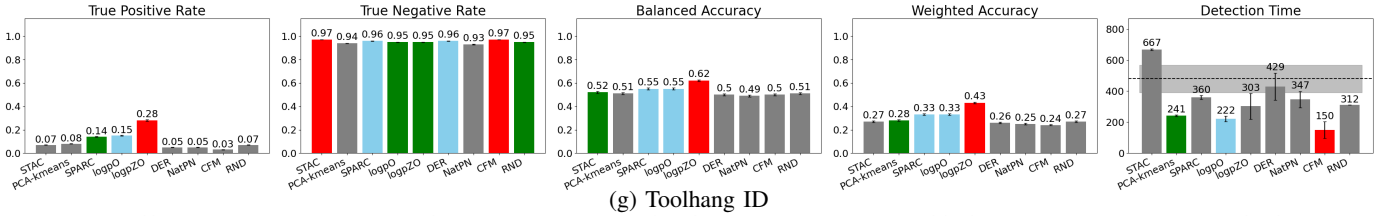
(d) Square OOD



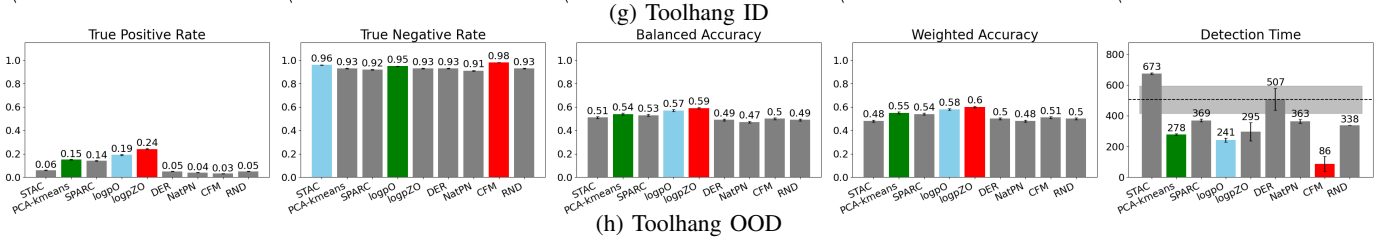
(e) Can ID



(f) Can OOD

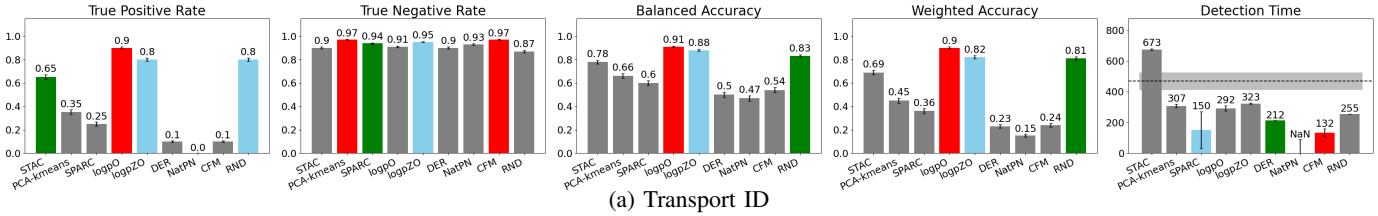


(g) Toolhang ID

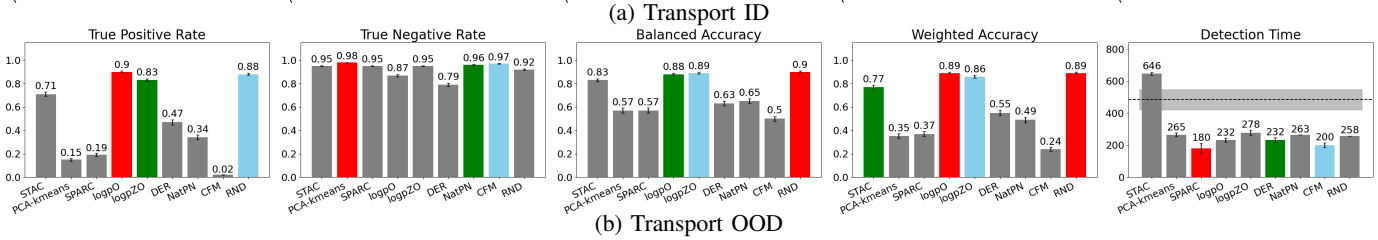


(h) Toolhang OOD

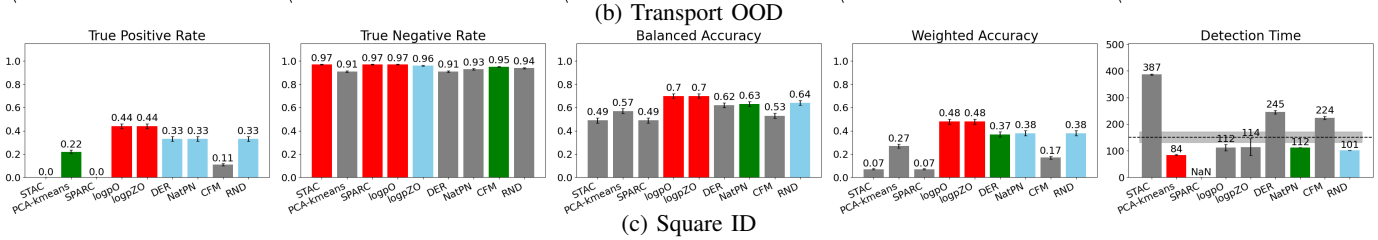
Fig. 11: Quantitative results in simulation tasks by FM policy (best, second, third), which augments Fig. 8 by including all quantitative metrics. The takeaways are similar as before, where logpZO and RND are the top-2 best-performing method overall.



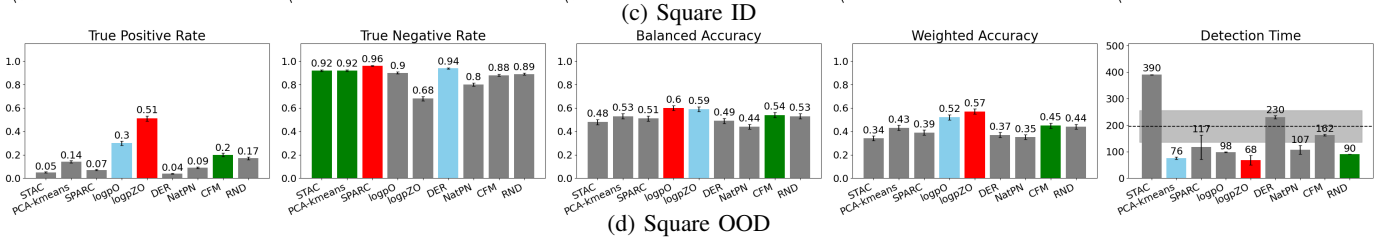
(a) Transport ID



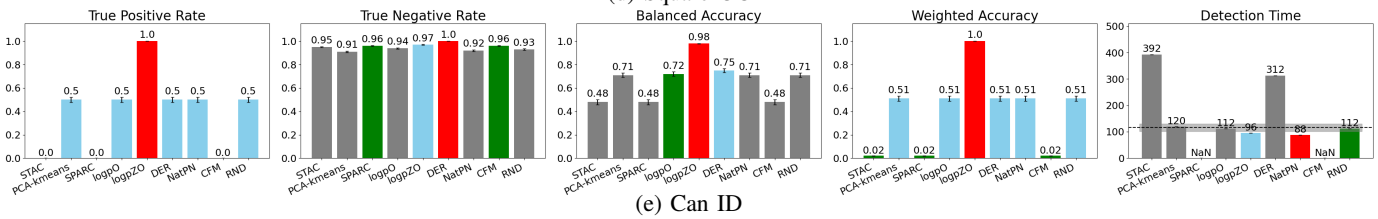
(b) Transport OOD



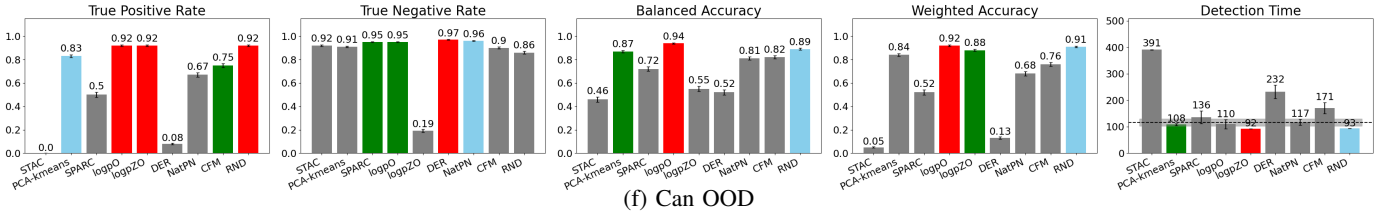
(c) Square ID



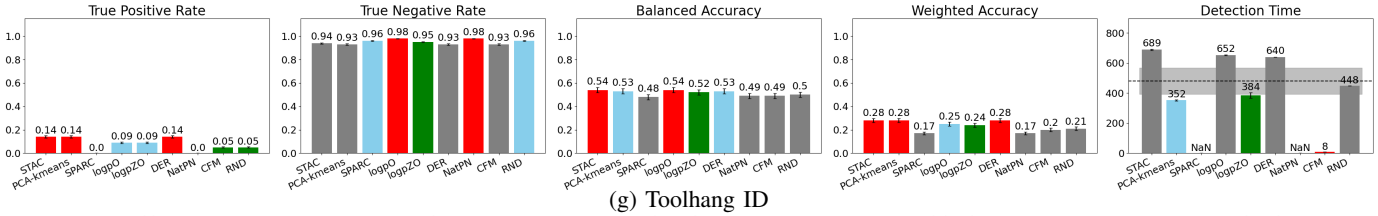
(d) Square OOD



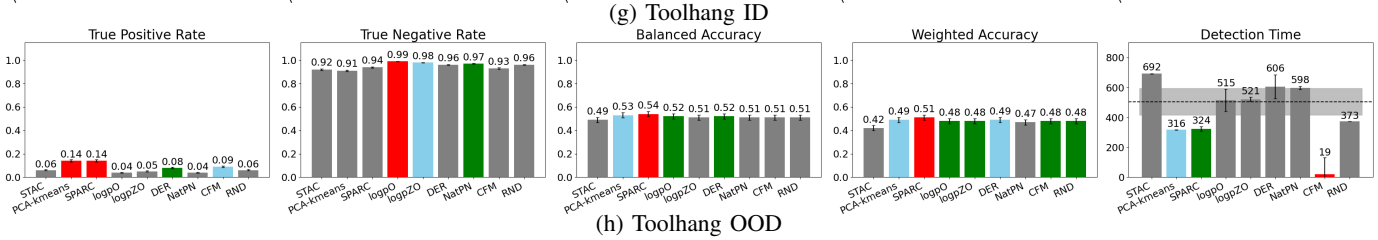
(e) Can ID



(f) Can OOD

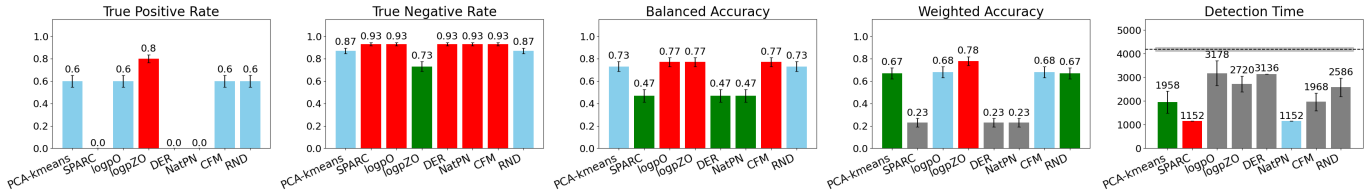


(g) Toolhang ID

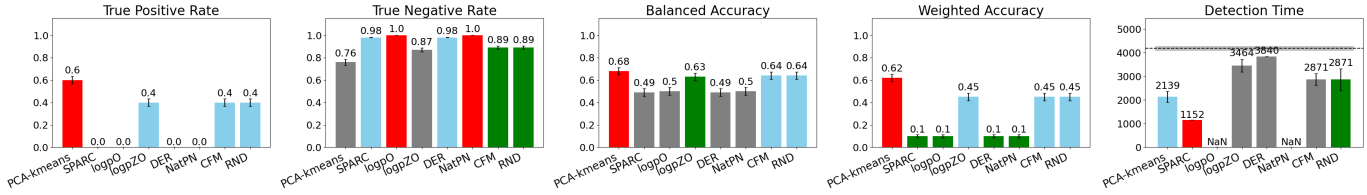


(h) Toolhang OOD

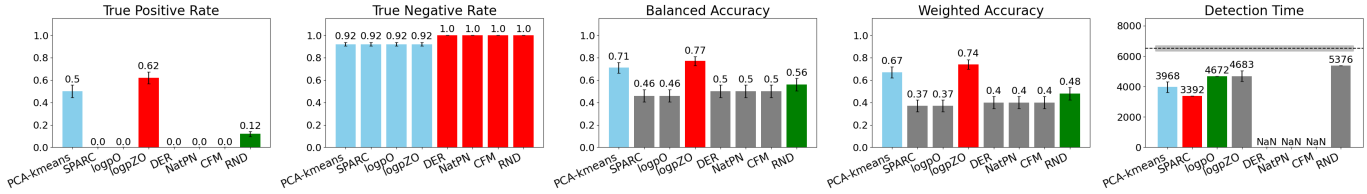
Fig. 12: Quantitative results in simulation tasks by DP (best, second, third). The layout is identical to Fig. 8. We similarly observe that learned methods seem to have more capacity to detect failures than post-hoc ones, with RND and logpZO being the best-performing methods.



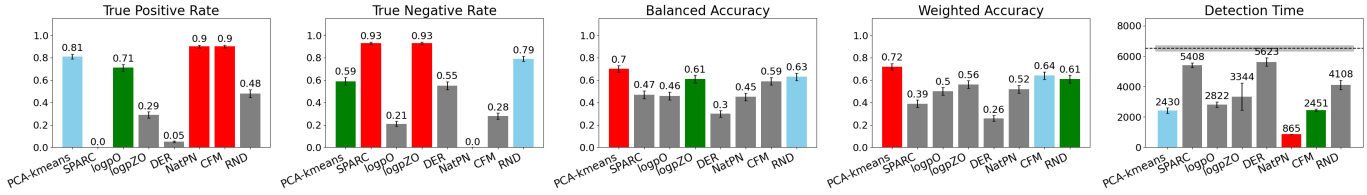
(a) FM policy: (Setting-dependent band) ID + Disturb



(b) FM policy: (ID-only band) ID + Disturb

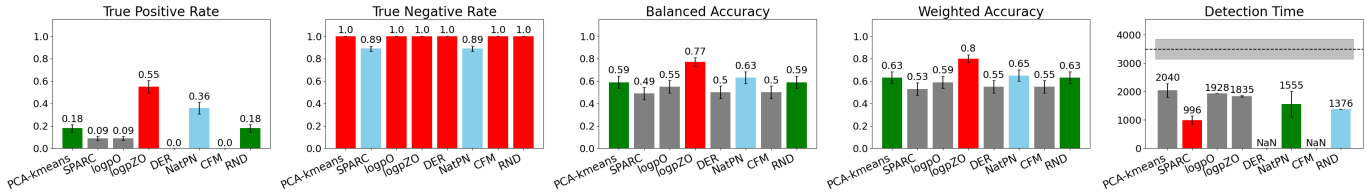


(c) FM policy: (Setting-dependent band) OOD initial condition

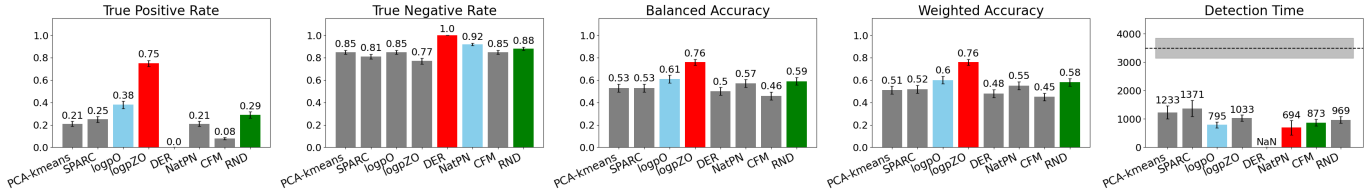


(d) FM policy: (ID-only band) OOD initial condition

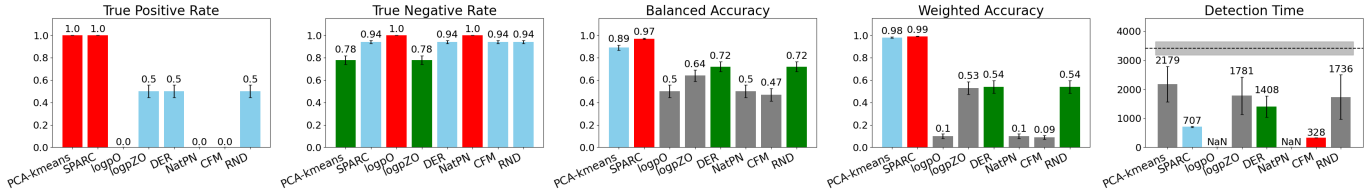
Fig. 13: Quantitative results on the **FoldRedTowel** robot hardware task using two ways to compute the CP band (best, second, third). logpZO remains to be the most robost method overall.



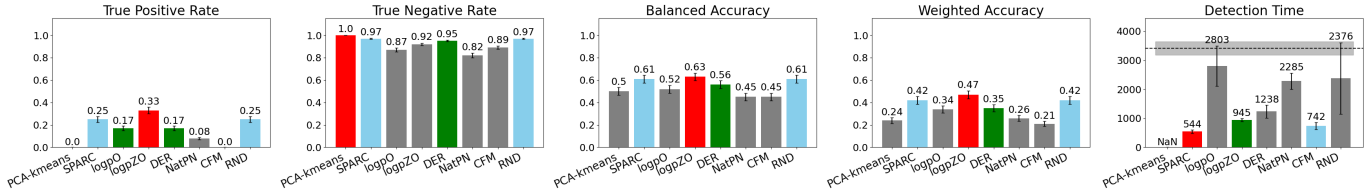
(a) DP: (Setting-dependent band) OOD initial condition



(b) DP: (ID-only band) OOD initial condition



(c) FM policy: (Setting-dependent band) OOD initial condition



(d) FM policy: (ID-only band) OOD initial condition

Fig. 14: Quantitative results on the **CleanUpSpill** robot hardware task using two ways to compute the CP band (best, second, third). logpZO remains to be the most robost method overall.