
Failure Prediction at Runtime for Generative Robot Policies

Ralf Römer^{1,*} Adrian Kobras^{1,*} Luca Worbis¹ Angela P. Schoellig^{1,2,3}

¹ Technical University of Munich, Germany; Learning Systems and Robotics Lab;
Munich Institute of Robotics and Machine Intelligence (MIRMI)

² Robotics Institute Germany ³ Munich Center for Machine Learning

ralf.roemer@tum.de

Abstract

Imitation learning (IL) with generative models, such as diffusion and flow matching, has enabled robots to perform complex, long-horizon tasks. However, distribution shifts from unseen environments or compounding action errors can still cause unpredictable and unsafe behavior, leading to task failure. Therefore, early failure prediction during runtime is essential for deploying robots in human-centered and safety-critical environments. We propose FIPER, a general framework for Failure Prediction at Runtime for generative IL policies that does not require failure data. FIPER identifies two key indicators of impending failure: (i) out-of-distribution (OOD) observations detected via random network distillation in the policy’s embedding space, and (ii) high uncertainty in generated actions measured by a novel action-chunk entropy score. Both failure prediction scores are calibrated using a small set of successful rollouts via conformal prediction. A failure alarm is triggered when both indicators, aggregated over short time windows, exceed their thresholds. We evaluate FIPER across five simulation and real-world environments involving diverse failure modes. Our results demonstrate that FIPER better distinguishes actual failures from benign OOD situations and predicts failures more accurately and earlier than existing methods. We thus consider this work an important step towards more interpretable and safer generative robot policies. Code, data, and videos are available at tum-lsy.github.io/fiper_website.

1 Introduction

Imitation learning (IL) for robotics usually involves high-dimensional, diverse, and multimodal data [69, 39]. Recent advances in generative modeling, such as diffusion models [64, 25] and flow matching [40], have led to significant progress in IL algorithms, substantially expanding the range of complex, long-horizon tasks that robots can perform [11, 78, 56, 6, 5]. Despite these improvements, generative IL policies remain imperfect even when trained on large-scale robot data [39, 50]: Unexpected visual or state shifts and compounding errors in action predictions can cause erratic or unsafe behavior, ultimately leading to task failure [37, 1, 42, 75]. In human-centered and safety-critical settings, it is therefore crucial to predict such failures as early as possible during runtime to enable timely intervention [44] or safe fallbacks [8] or to ask human experts to demonstrate the task [74]. However, failure prediction is difficult because it cannot be treated as a typical classification problem for two main reasons: First, it is often not possible to generate examples of failures [18, 42], as this would endanger the robot and its environment. Second, the

*Equal contribution.

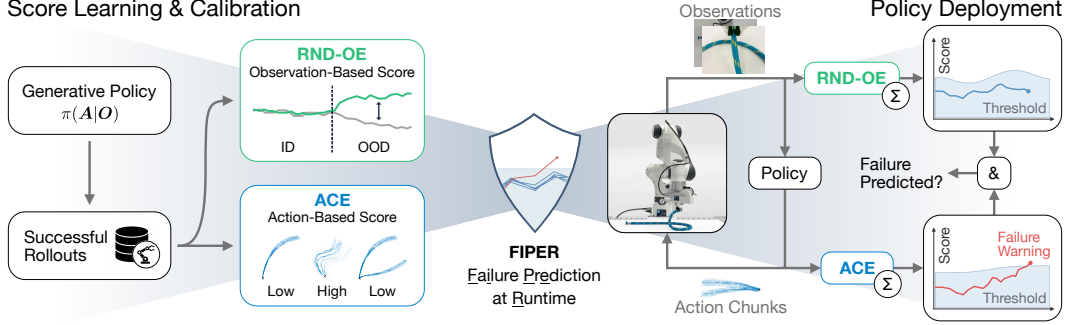


Figure 1: FIPER can predict task failures of generative robot policies during runtime without using any failure data. FIPER detects two key signals indicative of impending failure: (i) consecutive out-of-distribution (OOD) observations via random network distillation in the policy’s observation embedding space (RND-OE) and (ii) persistently high uncertainty in generated actions via action-chunk entropy (ACE). Both scores are calibrated based on a few successful rollouts and aggregated over a sliding window. If both submodules issue a warning, then FIPER predicts a failure. Its task-agnostic design enables FIPER to issue accurate and early failure warnings in diverse environments.

range of potential failure modes during closed-loop operation in complex, real-world environments is vast [1, 75], meaning that creating labeled data is not a viable option.

Existing approaches for anticipating and detecting policy failures fall into two broad categories: Those that look for out-of-distribution (OOD) observations or actions [37, 75, 22] and those that externally monitor the robot’s behavior or progress using vision-language-models (VLMs) [1, 17]. Both approaches have drawbacks: Pure OOD detectors trigger on any novel situation, even if the policy can generalize to it. At the same time, VLM-based methods only raise alarms after errors manifest, providing no foresight about impending failure. Most closely related to our work, Xu et al. [75] propose using a scalar uncertainty score derived from the policy inputs. However, as future behavior is determined by the *actions* of the policy, observation-only methods can miss early warning signs present in the action distribution. In summary, current approaches either isolate observations and actions, depend on failure-mode labels, or issue alerts too late for intervention.

To address these limitations, we propose Failure Prediction at Runtime (FIPER), a lightweight framework for predicting failures of generative robot policies that requires no examples of failures. FIPER is built on our insight that actual failures coincide with two indicators: (i) successive deviations of the observations from the patterns expected in successful rollouts and (ii) prolonged high uncertainty in the stochastic actions generated by the policy. We propose measuring (i) by leveraging random network distillation (RND) [9] in the policy’s observation embedding space and (ii) based on the entropy in the observation-conditioned action chunk distributions. We statistically calibrate our observation- and action-based uncertainty scores on a small set of successful rollouts using conformal prediction [3]. At runtime, FIPER raises an alarm when *both* scores, aggregated over short time windows, exceed their respective thresholds. This design, illustrated in Figure 1, enables FIPER to predict failures accurately *and* at an early stage with a low rate of false alarms. We evaluate FIPER for widespread diffusion- and flow-based IL policies across diverse simulation and real-world environments that exhibit various types of failures. In comparison to state-of-the-art baselines, our proposed scores can better distinguish actual failures from OOD situations, and FIPER achieves the highest accuracy and lowest detection time across all environments. Our findings suggest that FIPER is a promising step towards more interpretable and safer deployment of generative robot policies.

2 Related Work

Generative Policies for IL: IL is an effective method to teach a robot new tasks by training a policy directly on expert demonstrations [55, 77, 45], but has historically suffered from compounding errors due to (covariate) distribution shifts [58, 59] as well as limited ability to capture multimodal expert behavior [38, 31]. Recent progress in IL [78, 79, 28, 2, 52, 11] can be mainly attributed to the development of powerful generative modeling techniques [34, 64, 65, 40] and the availability of high-quality demonstration data [32, 50, 10]. Generative diffusion models [64, 25, 65] in particular

excel at capturing the high-dimensional, multimodal distributions commonly encountered in human demonstration data [69]. Diffusion Policy [11] first demonstrated the potential of using a diffusion model conditioned on visual observations for receding horizon control in complex manipulation tasks. Further studies have extended this approach to additional input modalities like language or touch [56, 24], improved robustness [71], or incorporated multi-task capabilities [72]. Recently, generative models using flow matching [40] instead of diffusion have demonstrated faster inference and improved trajectory smoothness for IL policies [6, 7, 26, 5]. To improve generalization, recent work has also trained IL policies on large-scale, cross-embodiment data [50], with the option to fine-tune them for a particular robot or task [33, 68, 6, 43]. Despite the impressive capabilities of generative IL policies, they inherit the fundamental properties of IL, often behaving unpredictably and failing when deployed beyond their training distribution [35].

Uncertainty Quantification and OOD Detection: Detecting OOD samples is a key challenge across various subfields of machine learning [76, 60, 63], with numerous methods proposed to address it. Some approaches rely on a priori access to OOD data [23, 16, 42, 70], which is, however, often unavailable or not comprehensive enough [53]. Another line of work directly measures epistemic uncertainty [27], using model calibration [20], ensembling methods [36, 49, 46, 61] or Monte-Carlo dropout (MC) [19, 49, 13], which can then be used to detect OOD cases. Random network distillation (RND) [9], originally proposed to incentivize exploration in reinforcement learning (RL), has been shown to outperform both deep ensembles and MC dropout [12]. Prior work has adopted RND to uncertainty quantification [12] and confidence estimation in offline RL [22]. Other approaches use the reconstruction error or latent embeddings of autoencoders [57, 73, 30, 47, 51] or directly measure deviations in the observation or embedding space [67, 62, 51] to detect dissimilarity from the training distribution. Our method combines RND in the observation embedding space with a novel entropy-based score, which is conceptually related to Bayesian methods [19] and to the idea that sampling a batch instead of a single action can be used to detect failures [1].

Failure Detection for IL Policies: Reliable handling and detection of policy failures is crucial to ensure safe deployment of robots, especially in human-centered environments [54, 42]. OOD detection alone is insufficient for recognizing failures in IL, as the policy might be able to generalize in some novel situations. The failure prediction problem is also complicated by the closed-loop operation and multimodal behavior of generative policies, which can generate very different actions for the same observation [1]. Agia et al. [1] propose addressing this challenge by combining a cumulative temporal consistency score that quantifies the divergence between the overlapping components of two consecutive action distributions, with a VLM to detect task progression failures. Using VLMs for monitoring policies and reasoning about failures has also been proposed in other works [17], but these approaches are inherently slow and unable to predict failures early. Liu et al. [42] train a failure detector on image embeddings of a visual world model and identify OOD states using clustering. Still, their method relies on failure examples similar to other works [18]. ReDiffuser [22] uses an RND model trained on trajectories to estimate the reliability of sampled decisions and select the most reliable one. However, ReDiffuser is tailored to state-based policies and cannot account for high-dimensional visual inputs. Lee et al. [37] use the loss of a diffusion policy to predict failures, but their approach is not directly applicable to other generative models, such as flow matching. FAIL-Detect [75] fits a flow matching model to the distribution of observation embeddings and detects policy failures by quantifying the likelihood of observations under the learned distribution. Existing methods [1, 75, 42, 22] have in common that they aim to detect failures either only from the policy inputs *or* outputs. In contrast, FIPER takes both into account, allowing us to better distinguish actual failures from benign OOD situations that the policy can handle.

3 Problem Setup

We consider a robotic system performing a given task, which we model as a Markov decision process (MDP) with observation $\mathbf{o}_k \in \mathcal{O}$ and action $\mathbf{a}_k \in \mathcal{A}$ at timestep $k = 0, 1, \dots$. We assume a finite horizon T for the MDP, but the task can also be completed in fewer than T timesteps. A generative IL policy $\pi(\mathbf{A}|\mathbf{O})$ is trained to match the observation-conditioned action distribution in a demonstration dataset \mathcal{D} . At each policy timestep $t = 0, h, 2h, \dots$, the learned policy generates a chunk (sequence) of H future actions $\mathbf{A}_t = (\mathbf{a}_{t|t}, \dots, \mathbf{a}_{t+H-1|t}) \sim \pi(\cdot|\mathbf{O}_t)$ conditioned on an observation history $\mathbf{O}_t = (\mathbf{o}_{t-T_h+1}, \dots, \mathbf{o}_t)$ of length $T_h \geq 1$. The first $h \leq H$ actions $\mathbf{a}_{t:t+h-1|t}$ are applied

to the system before re-planning at timestep $t + h$. This general policy formulation encompasses common, state-of-the-art IL methods [11, 78] and vision-language-action models (VLAs) [6, 33].

Depending on the initial condition \mathbf{o}_0 , the stochastic transitions, and the stochastic actions generated by the policy, the system may succeed or fail to complete the task within T timesteps. Our goal is to detect as *accurately* and *early* as possible situations for which the policy, when executed for the remaining timesteps up to T , would not succeed in completing the task. For this, we aim to design a failure predictor $F(\cdot)$ that takes the trajectory $\tau_{:t} = (\mathbf{O}_0, \mathbf{A}_0, \mathbf{O}_h, \mathbf{A}_h, \dots, \mathbf{O}_t, \mathbf{A}_t)$ up to the current timestep t as input and predicts the final rollout outcome $F(\tau_{:t}) \in \{0, 1\}$. If at any time t , $F(\tau_{:t}) = 1$, the rollout is flagged as `Fail` with detection time t .

4 Methodology

We design FIPER to detect two characteristics of policy failures: Consecutive OOD observations and high uncertainty in the conditional action distributions. These failure indications are handled by two submodules, whose outputs are combined in the overall failure predictor $F(\cdot)$. We do not rely on failure data and use only a few successful ID policy rollouts for training and calibration.

4.1 Detecting OOD Observations via Random Network Distillation (RND-OE)

Generative IL policies have demonstrated some generalization capabilities [11, 78], but they still often struggle in situations that deviate substantially from their training distribution [39]. To predict policy failures based on observations, we therefore aim to detect if \mathbf{O}_t differs from ID situations in a way that is likely to negatively affect the policy’s performance. For this purpose,

we adopt RND [9], a novelty detector first proposed for exploration in reinforcement learning. RND consists of two neural networks with m -dimensional outputs, a randomly initialized target network $g(\cdot)$ and a predictor network $f_\theta(\cdot)$ parameterized by θ . The target network is frozen, and the predictor network is trained to match the outputs of the target network on ID data. We apply RND to the policy inputs and train the predictor on ID success data $\mathcal{D}_{\text{ID}} \sim q_\pi$ to minimize the loss

$$\mathcal{L}(\theta) = \mathbb{E}_{(\mathbf{O}_t, \mathbf{A}_t) \sim \mathcal{D}_{\text{ID}}} [s_{\text{RND}}(\mathbf{O}_t)], \quad \text{where} \quad s_{\text{RND}}(\mathbf{O}_t) = \|\mathbf{f}_\theta(\mathbf{O}_t) - \mathbf{g}(\mathbf{O}_t)\|_2. \quad (1)$$

Intuitively, the predictor and target networks produce similar outputs for observations they have seen before, but their outputs diverge on novel, unseen data. We reuse and freeze the observation encoder $h(\cdot)$ of the policy in both networks, which has two benefits: First, we can detect anomalies directly in the policy’s embedding space, which are more indicative of failures than OOD raw observations. Second, using the pre-trained feature extractor $h(\cdot)$ allows us to train the RND model even from a small dataset \mathcal{D}_{ID} . Due to this design, we refer to $s_{\text{RND}}(\mathbf{O}_t)$ as RND observation embedding (RND-OE) score. To ensure the predictor \mathbf{f}_θ is expressive enough to approximate the target network \mathbf{g} well on the ID dataset, we design \mathbf{f}_θ to be slightly larger than \mathbf{g} (see Appendix A.2). After training, we can use $s_{\text{RND}}(\mathbf{O}_t)$ to measure how much an observation embedding $\mathbf{O}_t^e = \mathbf{h}(\mathbf{O}_t)$, which action generation is effectively conditioned on, deviates from the patterns in successful ID rollouts. This idea is visualized in Figure 2.

Recent IL policies [11, 78] can often handle brief and less severe OOD situations, especially when trained on massive data [39]. However, multiple consecutive OOD observations are likely to cause compounding errors in action predictions [4] from which the policy cannot recover. For this reason, we construct our observation-based failure prediction score $\eta_{\text{O}}(\tau_{:t})$ by aggregating the RND-OE score $s_{\text{RND}}(\mathbf{O}_t)$ over a sliding time window of size w_{O} as

$$\eta_{\text{O}}(\tau_{:t}) = \sum_{k=0}^{\min(w_{\text{O}}-1, t/h)} s_{\text{RND}}(\mathbf{O}_{t-kh}) = \underbrace{s_{\text{RND}}(\mathbf{O}_t) + s_{\text{RND}}(\mathbf{O}_{t-h}) + \dots}_{\leq w_{\text{O}} \text{ times}}. \quad (2)$$

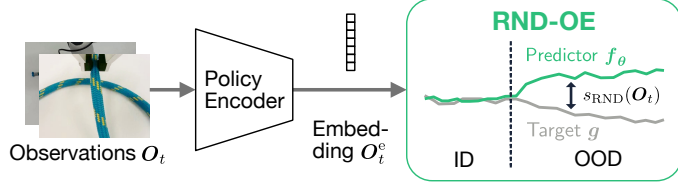


Figure 2: RND-OE recognizes failure-prone out-of-distribution (OOD) situations using random network distillation (RND) in the policy’s observation embedding space.

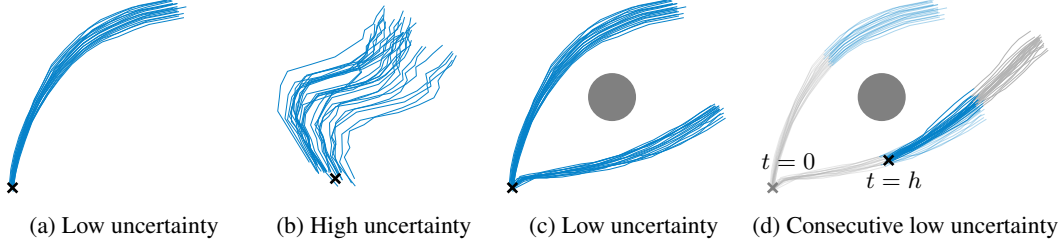


Figure 3: (a) to (c) In imitation learning from multimodal demonstrations, uncertainty in generated actions is reflected in entropy rather than variance. (d) Our action-chunk entropy (ACE) score is designed to handle observation-dependent action multimodality, whereas STAC [1] typically associates timesteps at which the policy decides on a behavior mode with high uncertainty.

If $\eta_O(\tau_{:t})$ exceeds a threshold $\gamma_{O,t}$ at any policy timestep t , the recent observations indicate imminent task failure, i.e., our observation-based failure predictor is given by

$$F_O(\tau_{:t}) = \mathbb{1}(\eta_O(\tau_{:t}) > \gamma_{O,t}). \quad (3)$$

We will discuss the calculation and calibration of the threshold $\gamma_{O,t}$ in Section 4.3.

4.2 Detecting High Action Uncertainty via Action-Chunk Entropy (ACE)

While we can infer *current* OOD situations from the observations, the *future* system evolution is ultimately determined by the actions that are generated by the policy and applied to the system. To improve model interpretability, which is especially important in decision-making systems [20], we aim to capture uncertainty at an intent level. To this end, we leverage the conditional action distribution $\pi(\mathbf{A}|\mathbf{O}_t) = \pi(\mathbf{A}|\mathbf{O} = \mathbf{O}_t)$ at the current observation \mathbf{O}_t to predict failures. Data for IL often contains action multimodality, with the number of modes generally being observation-dependent and unknown [29, 69]. Thus, a generative IL policy can generate very different (e.g., in an L_2 sense) actions for the same observation in a successful rollout. For this reason, merely measuring the variance in generated actions tells us little about uncertainty, as illustrated in Figure 3 (a) to (c). However, we observe that action multimodality in IL is usually of a discrete nature. For example, a robot can first pick up object A, B or C, grasp objects from the side or above, and avoid obstacles left or right. Therefore, each generated action should clearly correspond to one of the observation-dependent modes for successful task completion.

In principle, sharpness of the modes of $\pi(\mathbf{A}|\mathbf{O}_t)$ can be quantified by the entropy $E_{O_t}(\mathbf{A}) = -\int_{\mathcal{A}^H} p_\pi(\mathbf{A}_t|\mathbf{O}_t) \log p_\pi(\mathbf{A}_t|\mathbf{O}_t) d\mathbf{A}_t$. However, the likelihood $p_\pi(\mathbf{A}_t|\mathbf{O}_t)$ is unknown for diffusion- or flow-based policies π , so we need to approximate $E_{O_t}(\mathbf{A})$. For this, we sample a batch $\mathbf{A}_t = (\mathbf{A}_t^{(1)}, \dots, \mathbf{A}_t^{(B)})$ of B action chunks $\mathbf{A}_t^{(j)} = (\mathbf{a}_{t|t}^{(j)}, \dots, \mathbf{a}_{t+H-1|t}^{(j)}) \sim \pi(\cdot|\mathbf{O}_t)$, $j = 1, \dots, B$, at each policy timestep t . Since the dimensionality of $\mathbf{A}_t \in \mathcal{A}^H$ grows exponentially with the action chunk length H , directly approximating $E_{O_t}(\mathbf{A})$ would require an excessively large batch size B , exceeding the parallelization capabilities of common GPUs. As we aim for failure prediction *during runtime*, we reduce computational complexity by treating the prediction timesteps t through $t + H - 1$ separately and define the action-chunk entropy (ACE) score as

$$s_{\text{ACE}}(\mathbf{A}_t) = \sum_{i=0}^{H-1} \hat{E}(\mathbf{a}_{t+i|t}^{(1)}, \dots, \mathbf{a}_{t+i|t}^{(B)}). \quad (4)$$

Here, $\hat{E}(\cdot)$ measures the entropy of an action prediction step $t+i$ based on the actions sampled for that timestep. We adopt a dimension-wise binning approach to calculate $\hat{E}(\cdot)$, which we found to be more computationally efficient, robust, and easier to tune than other methods [14, 1] (see Appendix A.1 for details). Similar to the observation-based score (2), we aggregate (4) over a sliding window of w_A policy timesteps and define our action-based failure predictor $F_A(\cdot)$ with a threshold $\gamma_{A,t}$, i.e.,

$$\eta_A(\tau_{:t}) = \sum_{k=0}^{\min\{w_A-1, t/h\}} s_{\text{ACE}}(\mathbf{A}_{t-kh}) = \underbrace{s_{\text{ACE}}(\mathbf{A}_t) + s_{\text{ACE}}(\mathbf{A}_{t-h}) + \dots}_{\leq w_A \text{ times}}, \quad (5)$$

$$F_A(\tau_{:t}) = \mathbb{1}(\eta_A(\tau_{:t}) > \gamma_{A,t}). \quad (6)$$

4.3 Observation- AND Action-Based Failure Prediction with FIPER

Not all OOD observations lead to failure, and there may be temporary high (aleatoric) uncertainty in the generated actions even in successful rollouts, for example, due to the suboptimality and diversity of demonstrations. To obtain robust predictions specifically about *task failure*, we flag a rollout as **Fail** if and only if both failure predictors (3) and (6) raise a warning, i.e., FIPER combines their outputs with a logical conjunction as

$$F(\tau_{:t}) = F_O(\tau_{:t}) \wedge F_A(\tau_{:t}) = \mathbb{1}(\eta_O(\tau_{:t}) > \gamma_{O,t} \wedge \eta_A(\tau_{:t}) > \gamma_{A,t}). \quad (7)$$

To calibrate the two thresholds $\gamma_{O,t}$ and $\gamma_{A,t}$ in (7), we can leverage conformal prediction (CP) [3] for functional data [15]. For this, we use a calibration dataset $\mathcal{D}_c = \{\tau^i\}_{i=1}^M \sim q_\pi$ of M i.i.d. successful policy rollouts $\tau^i = (O_0^i, A_0^i, O_h^i, A_h^i, \dots, O_{T_i}^i, A_{T_i}^i)$. Since the uncertainty scores (2) and (5) vary considerably during each rollout, often being smallest at $t = 0$, we design their respective thresholds to be time-varying. We compute $\gamma_{O,t}$ and $\gamma_{A,t}$ in a similar way and, therefore, focus on the former here for brevity. As proposed by Diquigiovanni et al. [15], we split \mathcal{D}_c into two disjoint parts and use them to separately calculate the time-varying mean $\mu_{O,t}$ and band-width $b_{O,t}(\delta)$ of the score signals $\eta_O(\tau_{:t}^i)$, $t = 0, h, \dots, i = 1, \dots, M$. Thereby, $1 - \delta$ controls the proportion of signals staying in $[\mu_{O,t} - b_{O,t}(\delta), \mu_{O,t} + b_{O,t}(\delta)]$ for the entire time (see Appendix A.3 for more details). Since $\eta_O \geq 0$ by design, we only require an upper time-varying threshold $\gamma_{O,t} = \mu_{O,t} + b_{O,t}$ to enforce a desired upper bound on the probability of raising false alarms with (3) (similarly for (6)). As $\eta_O(\tau_{:t})$ and $\eta_A(\tau_{:t})$ are not independent, our combined predictor (7) satisfies the same bound.

Proposition 1. *Set $\delta \in (0, 1)$, and define the thresholds $\gamma_{O,t}$ and $\gamma_{A,t}$ as described above. Then, the probability that the failure predictor (7) of FIPER flags a new successful ID rollout $\tau \sim q_\pi$ of length $T' \leq T$ as **Fail** at any policy timestep $t \leq T'$ satisfies the upper bound*

$$\mathbb{P}(\exists t \in \{0, h, \dots, T'\} \text{ s.t. } F(\tau_{:t}) = 1) \leq \delta. \quad (8)$$

A more rigorous formulation of Proposition 1 and a proof are given in Appendix A.4. We can view δ as a design parameter, with a larger value increasing sensitivity to failures at the expense of more false alarms. Proposition 1 quantifies the ability of FIPER to recognize successes as such, not failures. To derive such a result, we would have to assume the availability of failure data, which we explicitly do not do for the reasons above. In addition to the threshold definition above, which we refer to as one-sided *CP band*, we also investigate two alternatives: A *CP constant* threshold $\gamma_{O,t} = \gamma_O$ defined as the $1 - \delta$ quantile of the set of conformity scores $\{\max_t \eta_O(\tau_{:t}^i)\}_{i=1}^M$ and a simpler *time-varying* threshold $\gamma_{O,t}$ defined as the $1 - \delta$ quantile of $\{\eta_O(\tau_{:t}^i)\}_{i=1}^M$. Proposition 1 also applies to the CP constant but not to the time-varying threshold.

Remark 1 (Rollout data). The training data for RND-OE, \mathcal{D}_{ID} , and the calibration data \mathcal{D}_c are sampled from the same distribution q_π but would need to be disjoint for Proposition 1 to hold rigorously. However, setting $\mathcal{D}_{\text{ID}} = \mathcal{D}_c$ worked well in our experiments, so we collect only one rollout dataset.

5 Experiments

We conduct extensive experiments across a diverse set of environments, mainly aimed at answering the following research questions: **Q1**) How well can our proposed scores distinguish failures from mere OOD situations? **Q2**) Does FIPER benefit from combining an observation- and an action-based failure predictor? **Q3**) Can FIPER predict failures more accurately and earlier than existing methods?

Environments. We consider three popular benchmark environments, SORTING [29], STACKING [29] and PUSH-T [11], and two real-world tasks, PRETZEL and PUSHCHAIR [1]. These environments differ in terms of robot embodiment, observation and action space, degree of action multimodality, and task duration, thus representing a diverse set of failure modes. We briefly explain the environments and how we create OOD scenarios to induce policy failures in the following:

- **SORTING:** A Franka robot needs to push two blocks on a table into their color-matching boxes. **OOD:** We change the dimensions of the blocks and the positions of the target boxes.
- **STACKING:** A Franka robot needs to stack three blocks on a target area. The task is multimodal, as there are six possible block arrangements. **OOD:** We vary the block sizes and target location.
- **PUSH-T:** The agent must push a planar T-shaped object into a target configuration. **OOD:** We use the data from Sentinel [1], which includes variations in the shape and dimensions of the T-object.

- **PRETZEL:** A Franka robot needs to fold a rope into a pretzel-like shape. OOD: We vary the rope’s initial configuration and rotate it around its own axis, which changes the bending behavior.
- **PUSHCHAIR:** A mobile manipulator needs to push a chair to a target position. OOD: We use the data from Sentinel [1], which includes variations of the initial chair pose.

Implementation. Our IL policies take one or two RGB images and the robot’s proprioceptive information as inputs. We consider different generative modeling techniques and policy backbones: For PUSH, PRETZEL, and PUSHCHAIR, we use denoising diffusion [25] with a temporal U-Net [11] backbone, and for SORTING and STACKING, we use flow matching [40] with a transformer backbone from ACT [78]. As image encoder, we use ResNet-18 [21] due to its demonstrated effectiveness for feature extraction in robotics [11, 80, 29]. We compute the ACE score in the Cartesian space of predicted end-effector positions to obtain task-relevant and interpretable uncertainty information in a computationally efficient way. For RND-OE, we use an MLP with 6 (4) layers for the predictor (target) network. Further details on model architectures, hyperparameters, and training are provided in Appendix B.

Baselines. We consider four state-of-the-art baselines for recognizing failures of IL policies. For all methods that involve offline training or clustering, we use the same ID success data \mathcal{D}_{ID} .

- PCA-kmeans [42] computes and clusters the principal components of the observation embeddings in \mathcal{D}_{ID} . During runtime, the distance of the same principal components of \mathcal{O}_t^e to the nearest cluster center is calculated to detect OOD observations.
- logpZO [75] learns the distribution of observation embeddings via flow matching. For a new observation, starting from \mathcal{O}_t^e , the ODE is solved backwards via the learned vector field to obtain a latent noise sample $\mathbf{Z}_{\mathcal{O}_t}$, and $\|\mathbf{Z}_{\mathcal{O}_t}\|_2^2$ is used as uncertainty score.
- STAC [1] calculates the divergence between the temporally overlapping components of the action chunk distributions at two consecutive policy timesteps to detect temporally inconsistent behavior.
- RND-A is closely adapted from He et al. [22] and learns a confidence function using RND that measures the reliability of generated actions for successful task completion.

Metrics. We label successful rollouts as negative and failed ones as positive and define the true-positive-rate (TPR) and true-negative-rate (TNR) in the standard way. To account for imbalances in the numbers of successes and failures, we report the balanced accuracy $\text{Acc} = \frac{1}{2}(\text{TPR} + \text{TNR})$. The normalized detection time $\text{DT} = \min_t \{t | F(\tau_{:t}) = 1\} / T \in [0, 1]$ is calculated only for failed rollouts correctly flagged as Fail. We mark the DT if the TPR or TNR is below 0.4. In these cases, failures are either rarely detected or almost all rollouts are immediately flagged as Fail. Accuracy and DT are standard metrics in prior work [1, 75], but they are not ideal for evaluating failure prediction, which should be accurate *and* fast. Always waiting until the last rollout timestep before making a “prediction” on the outcome likely results in high accuracy, but is not suitable for predicting failures before they occur. Conversely, simply flagging each rollout as Fail in the first timestep would yield a perfect DT. Therefore, we propose timestep-wise accuracy (TWA) as a novel evaluation metric for failure prediction methods. TWA differs from accuracy in that a true positive is assigned a value $1 - \text{DT}$ instead of 1, rewarding *earlier* correct failure predictions more strongly.

Evaluation Protocol. We use $M = 50$ successful rollouts for the three simulation environments and $M = 10$ for the two real-world tasks to train the learning-based failure predictors and calibrate the thresholds. Since there is generally no “best” quantile value for calibration (see Appendix C.4), we average all results over $1 - \delta \in \{0.9, 0.91, \dots, 0.99\}$. We treat the window size and threshold type (CP band, CP constant or time-varying) as method-specific hyperparameters. Thus, we use the value $w_{A,O} \in \{1, \dots, 50\}$ and threshold type that achieve the highest TWA across all environments, which we argue represents the best tradeoff between prediction accuracy, low DT, and robustness.

6 Results and Discussion

Our proposed scores can distinguish failures from OOD. Although they are often correlated, accurate failure prediction requires differentiating between OOD situations to which the policy can generalize and actual failures. To evaluate the effectiveness of our proposed observation- and action-based uncertainty scores in making this distinction, we divide rollouts into four categories, depending on their outcome and whether we have introduced OOD scenarios (see Section 5). We

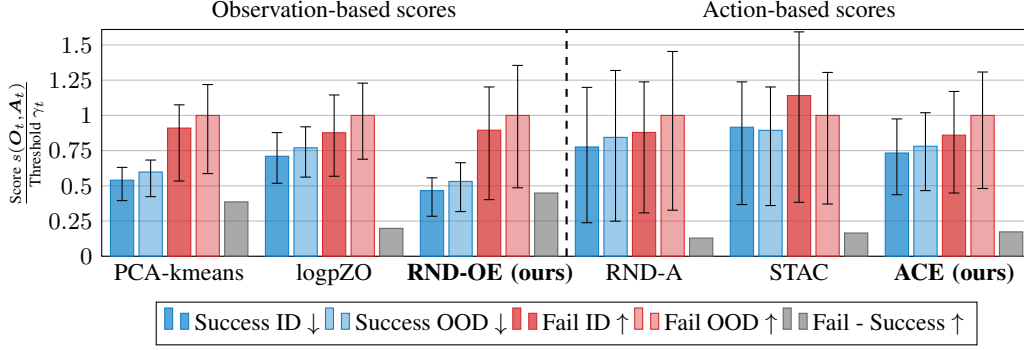


Figure 4: Our proposed scores (1) and (4) can distinguish failures from benign out-of-distribution (OOD) situations that the policy can generalize to. We group the rollouts into four categories along two axes: Success vs. Fail, and in-distribution (ID) vs. OOD. Robust failure prediction requires distinguishing Success OOD from Fail ID. We report the mean values across all tasks and five seeds, with the bars indicating the 25/75% quantiles.

Table 1: FIPER achieves superior failure prediction performance than the baselines. We highlight the best and underline the second-best value in each row. Detection times (DT) in brackets have corresponding TPR or TNR below 0.4, both of which indicate poor discrimination between successes and failures. We report the mean and standard deviation across five seeds.

Task	Metric	PCA-kmeans [42]	logpZO [75]	RND-A [22]	STAC [1]	RND-OE (ours)	ACE (ours)	FIPER (ours)
SORTING	TWA ↑	0.49±0.00	<u>0.54</u> ±0.00	0.55 ±0.01	0.46±0.00	0.52±0.01	<u>0.54</u> ±0.00	<u>0.54</u> ±0.00
	Acc. ↑	0.56±0.00	0.67 ±0.00	0.62±0.01	0.48±0.00	0.59±0.02	0.65±0.00	<u>0.66</u> ±0.00
	DT ↓	(0.15±0.00)	0.48±0.01	0.34±0.03	(0.46±0.00)	(0.17±0.03)	0.30 ±0.00	<u>0.32</u> ±0.00
STACKING	TWA ↑	0.66 ±0.00	0.58±0.00	0.50±0.00	0.59±0.00	<u>0.62</u> ±0.04	<u>0.62</u> ±0.00	<u>0.62</u> ±0.00
	Acc. ↑	0.75 ±0.00	0.69±0.01	0.56±0.01	0.66±0.00	0.70±0.06	<u>0.73</u> ±0.00	<u>0.73</u> ±0.00
	DT ↓	0.19±0.00	0.49±0.00	(0.44±0.01)	(0.38±0.00)	0.16 ±0.05	0.27±0.00	0.28±0.00
PUSHT	TWA ↑	0.53±0.00	0.52±0.00	0.52±0.01	0.58 ±0.00	0.54±0.01	<u>0.56</u> ±0.00	0.55±0.00
	Acc. ↑	0.58±0.00	0.55±0.00	0.55±0.01	0.71±0.00	0.55±0.01	<u>0.71</u> ±0.00	0.71 ±0.00
	DT ↓	(0.11±0.00)	(0.26±0.00)	(0.20±0.02)	0.52±0.00	(0.02±0.00)	0.31 ±0.00	<u>0.32</u> ±0.00
PRETZEL	TWA ↑	0.64±0.00	0.58±0.03	0.51±0.05	0.51±0.00	0.55±0.02	0.75 ±0.00	<u>0.68</u> ±0.03
	Acc. ↑	0.65±0.00	0.65±0.04	0.53±0.05	0.67±0.00	0.72±0.02	<u>0.82</u> ±0.00	0.85 ±0.00
	DT ↓	(0.01±0.00)	<u>0.24</u> ±0.04	(0.52±0.36)	0.44±0.00	0.44±0.02	0.13 ±0.00	0.33±0.07
PUSHCHAIR	TWA ↑	0.50±0.00	<u>0.78</u> ±0.02	0.71±0.05	0.73±0.00	0.74±0.11	0.69±0.00	0.83 ±0.02
	Acc. ↑	0.50±0.00	<u>0.92</u> ±0.02	0.82±0.05	0.88±0.00	0.80±0.14	0.80±0.00	0.96 ±0.02
	DT ↓	(0.00±0.00)	0.26±0.01	<u>0.21</u> ±0.02	0.30±0.00	0.11 ±0.07	0.23±0.00	0.27±0.00
Average	TWA ↑	0.57±0.00	0.60±0.01	0.56±0.02	0.57±0.00	0.59±0.04	<u>0.63</u> ±0.00	0.65 ±0.01
	Acc. ↑	0.61±0.00	0.69±0.01	0.62±0.03	0.68±0.00	0.67±0.05	<u>0.74</u> ±0.00	0.78 ±0.00
	DT ↓	(0.09±0.00)	0.35±0.01	0.34±0.09	0.42±0.00	0.18 ±0.03	<u>0.25</u> ±0.00	0.30±0.02

expect the uncertainty scores to increase in the following order: Success ID \leq Success OOD $<$ Fail ID \leq Fail OOD. In particular, the gap between Success OOD and Fail ID provides information about a score’s ability to distinguish between OOD and failure, which affects the robustness of any failure predictor method using the score. We provide the average scores, divided by the respective CP band threshold, across all tasks in Figure 4. RND-OE and ACE show a clear separation between Success OOD and Fail ID. Compared to the other observation-based methods, PCA-kmeans and logpZO, our RND-OE score yields a larger difference between the mean values for Success OOD and Fail ID. The same holds for our ACE score compared to RND-A and STAC. However, the gap between Fail and Success is generally smaller for the action-based scores, indicating that failures are harder to detect from the policy outputs than from the inputs. In the following, we investigate whether the results regarding single-timestep uncertainty scores transfer to failure prediction performance.

FIPER predicts failures more accurately and earlier. We benchmark FIPER against four baselines and also include our individual observation- and action-based failure predictors (3) and (6). Table 1 summarizes the results of our evaluation. A more detailed version, including TPR and TNR, is

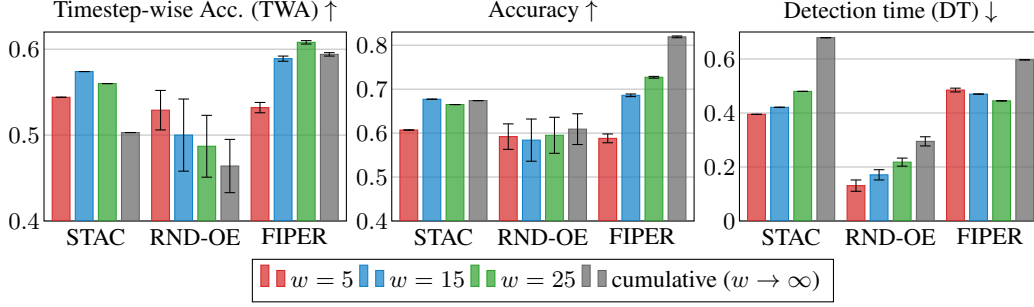


Figure 5: We compare our approach of aggregating uncertainty over a sliding window to accumulating scores over all past timesteps [1]. The latter method detects failure rollouts very late, mostly due to their greater length, whereas our approach can *predict* failures earlier. We use the same *CP constant* threshold definition for all methods, following STAC [1]. We normalize DT by the maximum episode length and average the results across five seeds, with the bars indicating the standard deviation.

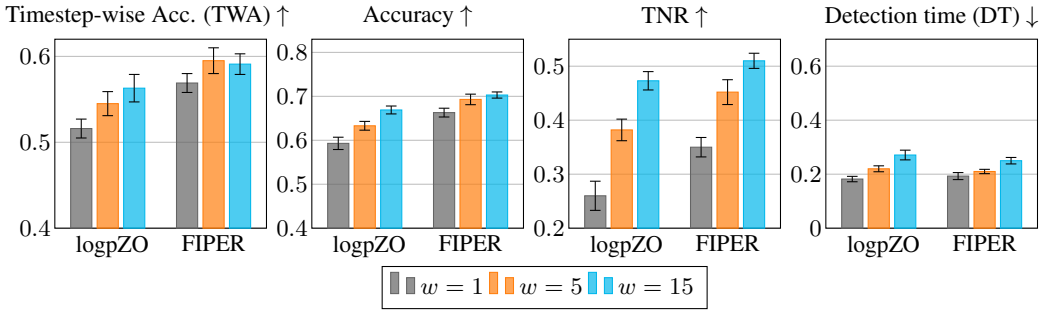


Figure 6: We compare our approach of aggregating uncertainty over a sliding window to making a decision based only on the current timestep, as proposed for logpZO [75]. Our method is much more robust to isolated OOD timesteps and greatly reduces false alarms (higher TNR) while only slightly increasing DT and improving overall accuracy. This comparison uses a *time-varying* threshold for all methods. We normalize DT by the maximum episode length and average the results across five seeds, with the bars indicating the standard deviation.

provided in Appendix C.2. FIPER obtains the highest TWA of 0.65 and accuracy of 0.78, and a lower DT of 0.30 than the baselines. Our two individual failure-predictors, RND-OE and ACE, can achieve even faster failure prediction when used alone, albeit with a lower TWA and accuracy. An overall TPR of 0.92 indicates that FIPER can predict different types of failures across diverse environments with high reliability. Compared to STAC, which also considers the distribution of generated actions, ACE predicts failures much more accurately, especially in the SORTING, STACKING, and PRETZEL environments that involve a high degree of action multimodality (see Figure 3 (d)). Among the observation-based methods, RND-OE achieves by far the lowest DT while performing better than PCA-kmeans and comparable to logpZO in terms of TWA and accuracy. PCA-kmeans is largely unable to distinguish OOD from failure, achieving a TNR of only 0.24, which shows that a larger difference between the average scores in Success and Fail rollouts (see Figure. 4) does not necessarily correlate with superior failure prediction performance. The baseline comparison demonstrates that FIPER represents the best combination of the desired properties of a failure predictor, i.e., high accuracy and early warnings.

Aggregating uncertainty scores over multiple timesteps robustly predicts failures. The two prior works most similar to ours either accumulate scores over all previous timesteps [1] or only use data from the current timestep [75]. We compare both approaches to our proposed method of aggregating scores over a sliding window. As the use of cumulative scores was proposed along with STAC [1], we use this baseline in the first comparison. The results shown in Figure 5 demonstrate that score accumulation can increase accuracy, but at the cost of much slower detection. In fact, we observe that cumulative uncertainty scores perform much better on tasks with a large difference in duration between successful and failed rollouts. Thus, we conclude that this approach recognizes

Table 2: Impact of the logical combination of the individual observation- and action-based failure predictors (3) and (6) of FIPER and the threshold calculation. The logical conjunction yields higher TWA and accuracy, showing that most failures are characterized by OOD observations *and* high entropy in generated actions. We report the mean and standard deviation across five seeds.

Operator	Threshold	TWA \uparrow	Acc. \uparrow	DT \downarrow	TPR \uparrow	TNR \uparrow
AND	CP band	0.62 ± 0.00	0.78 ± 0.01	0.47 ± 0.00	0.72 ± 0.01	0.84 ± 0.00
	CP constant	0.61 ± 0.00	0.73 ± 0.03	0.45 ± 0.00	0.59 ± 0.00	0.87 ± 0.00
	time-varying	0.65 ± 0.01	0.78 ± 0.00	0.30 ± 0.02	0.92 ± 0.00	0.65 ± 0.01
OR	CP band	0.54 ± 0.03	0.58 ± 0.03	(0.08 ± 0.02)	0.98 ± 0.02	0.18 ± 0.08
	CP constant	0.59 ± 0.04	0.68 ± 0.05	0.20 ± 0.03	0.93 ± 0.03	0.43 ± 0.12
	time-varying	0.51 ± 0.01	0.53 ± 0.01	(0.02 ± 0.01)	1.00 ± 0.00	0.05 ± 0.02

failure rollouts primarily due to their greater length, which naturally leads to a higher cumulative score. In comparison, considering only the most recent timesteps enables us to actually *predict* failures and raise a warning early. This capability is especially important in safety-critical scenarios such as surgical assistance or collaborative assembly, even if it results in more false alarms. We also investigate using a single-timestep score (i.e., $w = 1$), as proposed for logpZO by FAIL-Detect [75]. As shown in Figure 6, this leads to a very low TNR, i.e., flagging most successful rollouts as Fail, while only slightly reducing DT. The overall failure prediction performance, as measured by TWA, improves when using a sliding window. Note that Figures 5 and 6 are not intended as a comparison of different failure prediction scores (e.g., STAC vs. RND-OE), as we fix the window size and threshold type. In summary, the results underline that aggregating uncertainty scores over the most recent timesteps is crucial for predicting failures at an early stage and distinguishing them from successes.

Failures manifest in observations *and* actions. FIPER predicts a failure if and only if both the observation- *and* action-based scores exceed their thresholds. We ablate the two design decisions of logical combination and threshold type, and report the results in Table 2. The logical conjunction (AND) yields higher TWA and accuracy, primarily due to its much higher TNR compared to OR. Intuitively, waiting for both scores to exceed their threshold increases the robustness to OOD success cases. Yet, we find that this more conservative approach can nonetheless predict 92% of all failures. This supports our claim that policy failure is associated with simultaneous OOD observations and high entropy in generated actions. As a variation of FIPER, the use of a logical disjunction of our two failure predictors provides a powerful alternative when the primary goal is to achieve very low DT and high TPR. This may be particularly desirable if failures could directly endanger people or cause damage to expensive items. Moreover, we observe that the threshold type has a significant impact on performance. The CP band and CP constant thresholds achieve a very high TNR in accordance with Proposition 1. The time-varying threshold yields the highest TWA for FIPER, while the inherently more sensitive OR ablation should be used with the more conservative CP constant threshold.

7 Conclusions and Limitations

The design of FIPER is motivated by the insight that actual task failures of generative diffusion- and flow-based policies are associated with successive OOD observations and high-uncertainty conditional action distributions. Our experiments across diverse environments and tasks demonstrate that our proposed observation- and action-based uncertainty scores can distinguish OOD from failure situations, and that FIPER can predict failures more accurately and earlier than existing methods. FIPER makes no assumptions about specific failure modes and requires no failure data, enabling its use in diverse human-centered environments where interpretable and safe robot behavior is critical.

While we use only a few successful rollouts for calibration, the need to collect them and train an RND-OE model that is separate from the policy is still a limitation of our approach. FIPER requires little runtime computation in our considered environments, but this could change with a very high-dimensional action space (e.g., for humanoid robots). We consider only single-task vision-based IL policies in this work. Adapting FIPER to recent large-scale VLAs [33, 6, 5], additional input modalities or reinforcement learning with generative policies represent interesting avenues for future work. Since our approach to failure prediction does not rely on access to the training data of the policy and operates directly in the observation embedding space, we expect it to work well for such potential extensions as well.

Acknowledgements

Ralf Römer gratefully acknowledges the support of the research group ConVeY funded by the German Research Foundation under grant GRK 2428. This work has been supported by the Robotics Institute Germany, funded by BMFT grant 16ME0997K.

References

- [1] Christopher Agia, Rohan Sinha, Jingyun Yang, Zi-ang 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] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? In *International Conference on Learning Representations (ICLR)*, 2023.
- [3] Anastasios N Angelopoulos, Stephen Bates, et al. Conformal prediction: A gentle introduction. *Foundations and Trends® in Machine Learning*, 16(4):494–591, 2023.
- [4] Suneel Belkhale, Yuchen Cui, and Dorsa Sadigh. Data quality in imitation learning. *Conference on Neural Information Processing Systems (NeurIPS)*, 36:80375–80395, 2023.
- [5] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.
- [6] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. $\pi 0$: A vision-language-action flow model for general robot control. In *Robotis: Science and Systems (RSS)*, 2025.
- [7] Max Braun, Noémie Jaquier, Leonel Rozo, and Tamim Asfour. Riemannian flow matching policy for robot motion learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5144–5151. IEEE, 2024.
- [8] Lukas Brunke, Melissa Greeff, Adam W Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5(1):411–444, 2022.
- [9] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations (ICLR)*, pages 1–17, 2019.
- [10] Federico Ceola, Lorenzo Natale, Niko Sünderhauf, and Krishan Rana. Lhmanip: A dataset for long-horizon language-grounded manipulation tasks in cluttered tabletop environments. *arXiv preprint arXiv:2312.12036*, 2023.
- [11] 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.
- [12] 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)*, 2019.
- [13] Yuchen Cui, David Isele, Scott Niekum, and Kikuo Fujimura. Uncertainty-aware data aggregation for deep imitation learning. In *International Conference on Robotics and Automation (ICRA)*, pages 761–767. IEEE, 2019.
- [14] Sylvain Delattre and Nicolas Fournier. On the kozachenko–leonenko entropy estimator. *Journal of Statistical Planning and Inference*, 185:69–93, 2017.
- [15] 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.

- [16] Andrija Djurisić, Nebojsa Božanić, Arjun Ashok, and Rosanne Liu. Extremely simple activation shaping for out-of-distribution detection. In *International Conference on Learning Representations (ICLR)*, 2023.
- [17] Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation. In *International Conference on Learning Representations (ICLR)*, 2025.
- [18] Alec Farid, David Snyder, Allen Z Ren, and Anirudha Majumdar. Failure prediction with statistical guarantees for vision-based robot control. In *Robotics: Science and Systems (RSS)*. IEEE, 2022.
- [19] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning (ICML)*, pages 1050–1059. PMLR, 2016.
- [20] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning (ICML)*, pages 1321–1330. PMLR, 2017.
- [21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [22] 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)*, 2024.
- [23] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, 2018.
- [24] Carolina Higuera, Akash Sharma, Chaithanya Krishna Bodduluri, Taosha Fan, Patrick Lancaster, Mrinal Kalakrishnan, Michael Kaess, Byron Boots, Mike Lambeta, Tingfan Wu, et al. Sparsh: Self-supervised touch representations for vision-based tactile sensing. In *Conference on Robot Learning (CoRL)*, 2024.
- [25] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Conference on Neural Information Processing Systems (NeurIPS)*, 33:6840–6851, 2020.
- [26] Xixi Hu, Qiang Liu, Xingchao Liu, and Bo Liu. Adaflow: Imitation learning with variance-adaptive flow-based policies. *Conference on Neural Information Processing Systems (NeurIPS)*, 37:138836–138858, 2024.
- [27] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021.
- [28] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning (ICML)*, 2022.
- [29] Xiaogang Jia, Denis Blessing, Xinkai Jiang, Moritz Reuss, Atalay Donat, Rudolf Lioutikov, and Gerhard Neumann. Towards diverse behaviors: A benchmark for imitation learning with human demonstrations. In *International Conference on Learning Representations (ICLR)*, 2024.
- [30] Philipp Joppich, Sebastian Dorn, Oliver De Candido, Jakob Knollmüller, and Wolfgang Utschick. Classification and uncertainty quantification of corrupted data using supervised autoencoders. In *Physical Sciences Forum*, volume 5, page 12. MDPI, 2022.
- [31] Liyiming Ke, Sanjiban Choudhury, Matt Barnes, Wen Sun, Gilwoo Lee, and Siddhartha Srinivasa. Imitation learning as f-divergence minimization. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pages 313–329, 2021.

- [32] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems*, 2024.
- [33] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning (CoRL)*, 2024.
- [34] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.
- [35] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanus Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning (ICML)*, pages 5637–5664. PMLR, 2021.
- [36] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Conference on Neural Information Processing Systems (NeurIPS)*, 30, 2017.
- [37] Sung-Wook Lee, Xuhui Kang, and Yen-Ling Kuo. Diff-Dagger: Uncertainty estimation with diffusion policy for robotic manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4845–4852, 2025.
- [38] Yunzhu Li, Jiaming Song, and Stefano Ermon. Infogail: Interpretable imitation learning from visual demonstrations. *Conference on Neural Information Processing Systems (NeurIPS)*, 30, 2017.
- [39] Fanqi Lin, Yingdong Hu, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. In *International Conference on Learning Representations (ICLR)*, 2025.
- [40] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.
- [41] Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky TQ Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.
- [42] 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.
- [43] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. RDT-1b: a diffusion foundation model for bimanual manipulation. In *International Conference on Learning Representations (ICLR)*, 2025.
- [44] Zeyi Liu, Arpit Bahety, and Shuran Song. Reflect: Summarizing robot experiences for failure explanation and correction. In *Conference on Robot Learning (CoRL)*, pages 3468–3484, 2023.
- [45] 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. *arXiv preprint arXiv:2108.03298*, 2021.
- [46] Kunal Menda, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Ensembledagger: A bayesian approach to safe imitation learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5041–5048. IEEE, 2019.
- [47] Marion Neumeier, Sebastian Dorn, Michael Botsch, and Wolfgang Utschick. Reliable trajectory prediction and uncertainty quantification with conditioned diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3461–3470, 2024.

- [48] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning (ICML)*, pages 8162–8171, 2021.
- [49] Farzad Nozarian, Christian Müller, and Philipp Slusallek. Uncertainty quantification and calibration of imitation learning policy in autonomous driving. In *International Workshop on the Foundations of Trustworthy AI Integrating Learning, Optimization and Reasoning (TAILOR)*, pages 146–162. Springer, 2020.
- [50] Abby O’Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.
- [51] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. Deep learning for anomaly detection: A review. *ACM Computing Surveys (CSUR)*, 54(2):1–38, 2021.
- [52] Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, et al. Imitating human behaviour with diffusion models. In *International Conference on Learning Representations (ICLR)*, 2023.
- [53] Wilbert Pumacay, Ishika Singh, Jiafei Duan, Ranjay Krishna, Jesse Thomason, and Dieter Fox. The COLOSSEUM: A benchmark for evaluating generalization for robotic manipulation. In *Robotics: Science and Systems (RSS)*, 2024.
- [54] 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.
- [55] Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislau Bölöni, and Sergey Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *International Conference on Robotics and Automation (ICRA)*, pages 3758–3765. IEEE, 2018.
- [56] Moritz Reuss and Rudolf Lioutikov. Multimodal diffusion transformer for learning from play. In *2nd Workshop on Language and Robot Learning: Language as Grounding*, 2023.
- [57] Charles Richter and Nicholas Roy. Safe visual navigation via deep learning and novelty detection. In *Proceedings of Robotics: Science and Systems (RSS)*, 2017.
- [58] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 661–668, 2010.
- [59] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 627–635, 2011.
- [60] Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795, 2021.
- [61] Ralf Römer, Armin Lederer, Samuel Tesfazgi, and Sandra Hirche. Vision-based uncertainty-aware motion planning based on probabilistic semantic segmentation. *IEEE Robotics and Automation Letters*, 8(11):7825–7832, 2023.
- [62] Rohan Sinha, Amine Elhafsi, Christopher Agia, Matthew Foutter, Edward Schmerling, and Marco Pavone. Real-time anomaly detection and reactive planning with large language models. In *Robotics: Science and Systems (RSS)*, 2024.
- [63] Rohan Sinha, Apoorva Sharma, Somrita Banerjee, Thomas Lew, Rachel Luo, Spencer M Richards, Yixiao Sun, Edward Schmerling, and Marco Pavone. A system-level view on out-of-distribution data in robotics. *arXiv preprint arXiv:2212.14020*, 2022.

- [64] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning (ICML)*, pages 2256–2265. pmlr, 2015.
- [65] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021.
- [66] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.
- [67] Yiyu Sun, Yifei Ming, Xiaojin Zhu, and Yixuan Li. Out-of-distribution detection with deep nearest neighbors. In *International Conference on Machine Learning (ICML)*, pages 20827–20840. PMLR, 2022.
- [68] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems (RSS)*, 2024.
- [69] Julen Urain, Ajay Mandlekar, Yilun Du, Mahi Shafiuallah, Danfei Xu, Katerina Fragkiadaki, Georgia Chalvatzaki, and Jan Peters. Deep generative models in robotics: A survey on learning from multimodal demonstrations. *arXiv preprint arXiv:2408.04380*, 2024.
- [70] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 550–564, 2018.
- [71] 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.
- [72] Yixiao Wang, Yifei Zhang, Mingxiao Huo, Ran Tian, Xiang Zhang, Yichen Xie, Chenfeng Xu, Pengliang Ji, Wei Zhan, Mingyu Ding, et al. Sparse diffusion policy: A sparse, reusable, and flexible policy for robot learning. In *Conference on Robot Learning (CoRL)*, 2024.
- [73] 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.
- [74] Philipp Wu, Yide Shentu, Qiayuan Liao, Ding Jin, Menglong Guo, Koushil Sreenath, Xingyu Lin, and Pieter Abbeel. Robocopilot: Human-in-the-loop interactive imitation learning for robot manipulation. *arXiv preprint arXiv:2503.07771*, 2025.
- [75] Chen Xu, Tony Khuong Nguyen, Emma Dixon, Christopher Rodriguez, Patrick Miller, Robert Lee, Paarth Shah, Rares Ambrus, Haruki Nishimura, and Masha Itkina. Can we detect failures without failure data? Uncertainty-aware runtime failure detection for imitation learning policies. In *Robotics: Science and Systems (RSS)*, 2025.
- [76] Jingkan Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *International Journal of Computer Vision (IJCV)*, 132(12):5635–5662, 2024.
- [77] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *International Conference on Robotics and Automation (ICRA)*, pages 5628–5635. IEEE, 2018.
- [78] Tony Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems (RSS)*, 2023.
- [79] Tony Z Zhao, Jonathan Tompson, Danny Driess, Pete Florence, Kamyar Ghasemipour, Chelsea Finn, and Ayaan Wahid. Aloha unleashed: A simple recipe for robot dexterity. In *Robotics: Science and Systems (RSS)*, 2024.

- [80] Yifeng Zhu, Abhishek Joshi, Peter Stone, and Yuke Zhu. Viola: Imitation learning for vision-based manipulation with object proposal priors. In *Conference on Robot Learning (CoRL)*, pages 1199–1210. PMLR, 2023.

Appendix

Table of Contents

A	Method Details	18
A.1	Action-Chunk Entropy (ACE)	18
A.2	Random Network Distillation with Observation Embeddings (RND-OE)	19
A.3	Conformal Prediction	19
A.4	Logical Combination	21
B	Additional Experimental Details	22
B.1	Environment Details	22
B.2	Discussion on other Tasks	24
B.3	Policy Details	24
B.4	Metrics	25
B.5	Baselines	26
B.6	Compute Resources	26
C	Extended Results	26
C.1	Uncertainty Scores	26
C.2	Extended Baseline Comparison	27
C.3	Impact of the Window Size for Score Aggregation	28
C.4	Impact of the Threshold Computation	29
C.5	Discussion of Action-Based Scores	31
C.6	Discussion of Observation-Based Scores	32
D	Limitations	32

A Method Details

Below, we have summarized the most important mathematical symbols used in this work.

- T : Maximum episode length
- $k = 0, 1, \dots, T$: Timestep
- \mathbf{o}_k : Robot observation
- \mathbf{a}_k : Robot action
- T_h : Observation history length
- H : Action chunk length
- h : Action execution steps
- $t = 0, h, \dots, T$: Policy timestep
- \mathbf{O}_t : Observation history
- \mathbf{A}_t : Action chunk
- \mathbf{A}_t : Batch of action chunks
- B : Action-chunk batch size
- $\pi(\mathbf{A}|\mathbf{O})$: Generative IL policy
- $\tau_{:t}$: Trajectory of observation-action pairs up to policy timestep t
- $\mathbf{f}_\theta(\mathbf{O}_t)$: Trainable RND predictor network
- $\mathbf{g}(\mathbf{O}_t)$: Frozen RND target network
- $s_{\text{RND}}(\mathbf{O}_t)$: RND observation embedding (RND-OE) score
- $\hat{E}(\cdot)$: Entropy of an action prediction step
- $s_{\text{ACE}}(\mathbf{A}_t)$: Action-chunk entropy (ACE) score
- w_O : Window size for aggregating $s_{\text{RND}}(\mathbf{O}_t)$
- w_A : Window size for aggregating $s_{\text{ACE}}(\mathbf{A}_t)$
- $\eta_O(\tau_{:t})$: Observation-based failure prediction score at policy timestep t
- $\eta_A(\tau_{:t})$: Action-based failure prediction score at policy timestep t
- $\gamma_{O,t}$: Threshold for $\eta_O(\tau_{:t})$
- $\gamma_{A,t}$: Threshold for $\eta_A(\tau_{:t})$
- \mathcal{D}_c : Calibration dataset
- M : Number of calibration rollouts
- q_π : Distribution of ID successful policy rollouts
- δ : Desired FPR

A.1 Action-Chunk Entropy (ACE)

To estimate uncertainty in generated actions, we sample a batch of B action chunks

$$\mathbf{A}_t = (\mathbf{A}_t^{(1)}, \dots, \mathbf{A}_t^{(B)}), \quad \mathbf{A}_t^{(j)} = (\mathbf{a}_{t|t}^{(j)}, \dots, \mathbf{a}_{t+H-1|t}^{(j)}) \sim \pi(\cdot | \mathbf{O}_t), \quad (9)$$

at policy timestep t and compute the action-chunk entropy (ACE) score

$$s_{\text{ACE}}(\mathbf{A}_t) = \sum_{i=0}^{H-1} \hat{E}(\mathbf{a}_{t+i|t}^{(1)}, \dots, \mathbf{a}_{t+i|t}^{(B)}), \quad (10)$$

where $\hat{E}(\cdot)$ measures the entropy of an action prediction step $t + i$ based on the actions $\mathbf{a}_{t+i|t}^{(j)} \in \mathcal{A} \subseteq \mathbb{R}^D$, $j = 1, \dots, B$, sampled for that timestep. For the following explanation of the calculation of $\hat{E}(\cdot)$, we use the notation $\mathbf{a}_{t+i|t}^{(j)} = \mathbf{a}_{t+i}^{(j)} = [a_{t+i,1}^{(j)}, \dots, a_{t+i,D}^{(j)}]^\top$

for brevity. Let $\{\mathbf{A}_t, \dots, \mathbf{A}_{N_c t}\}$ be the set of action chunks in the calibration dataset \mathcal{D}_c , where $\mathbf{A}_{nt} = (\mathbf{a}_{nt}, \dots, \mathbf{a}_{nt+H-1})$, $n = 1, \dots, N_c$. Offline, we calculate the maximum range of actions as

$$R_d = \max_{\substack{n \in \{1, \dots, N_c\} \\ i \in \{0, \dots, H-1\}}} a_{nt+i,d} - \min_{\substack{n \in \{1, \dots, N_c\} \\ i \in \{0, \dots, H-1\}}} a_{nt+i,d} \quad (11)$$

for each dimension $d = 1, \dots, D$ and define the fixed cell size as αR_d , where $\alpha \in (0, 1)$ is called cell size factor. For a new action batch (9), each action dimension d is partitioned into

$$N_d^i = \left\lceil \frac{\max_{j \in \{1, \dots, B\}} a_{t+i,d}^{(j)} - \min_{j \in \{1, \dots, B\}} a_{t+i,d}^{(j)}}{\alpha R_d} \right\rceil \quad (12)$$

bins for each step $i = 0, \dots, H-1$ in the prediction horizon. This induces $N^i = \prod_{d=1}^D N_d^i$ bin cells, which we denote by $\mathcal{C}_c^i \subset \mathbb{R}^D$, $c = 1, \dots, N^i$. We then build timestep-wise histograms for the B actions sampled for each timestep $t+i$ over these cells, yielding the probabilities $p_c^i = \frac{n_c^i}{B}$, where $n_c^i = |j : \mathbf{a}_{t+i}^{(j)} \in \mathcal{C}_c^i|$ is the number of actions for timestep $t+i$ that are contained in cell \mathcal{C}_c^i . Finally, the action entropy for each timestep in (10) is given by

$$\hat{E}(\mathbf{a}_{t+i|t}^{(1)}, \dots, \mathbf{a}_{t+i|t}^{(B)}) = - \sum_{c=1}^{N^i} p_c^i \log_2 p_c^i. \quad (13)$$

Our approach of treating timesteps separately leverages the fact that actions sampled for the same timestep tend to lie closer together, significantly reducing the total number of bins. Thus, compared to computing a single histogram over all timesteps, we require a much smaller action batch size to obtain a sufficiently accurate estimate of action entropy.

We calculate the ACE score in the Cartesian space using predicted end-effector positions. This is possible for any manipulation task (and even for locomotion or navigation) by using forward kinematics, even if the policy outputs joint angles or velocities. We adopt this approach not only for computational reasons but also to obtain interpretable uncertainty scores in the same 3D space where the task is performed. High uncertainty in Cartesian space means that the policy is unsure about the correct path to take (e.g., where to grasp an object). In contrast, entropy in, for example, joint velocities, does not necessarily correspond to task-relevant uncertainty. We also experimented with computing the ACE score using other action representations, such as full 6D poses, but did not observe improvements in performance.

A.2 Random Network Distillation with Observation Embeddings (RND-OE)

An RND model [9] detects OOD situations by training a predictor network $\mathbf{f}_\theta(\cdot)$ to match the outputs of a frozen target network $\mathbf{g}(\cdot)$ on ID data. Both networks receive the same inputs, and after training, the outputs of the predictor and target networks typically differ more significantly for inputs that deviate from the training distribution than for those that align with it. Our RND-OE model utilizes the observation embeddings from the encoder module of a generative policy as input and is trained on a calibration dataset that exclusively consists of a few ID successful policy rollouts. Thereby, we employ a multi-layer fully connected neural network architecture for both the target and predictor networks of the RND-OE model. Both networks take observation embeddings as input and output an m -dimensional feature vector. The target network \mathbf{g} comprises four fully connected layers with dimensions $\dim(\mathbf{O}^e) \rightarrow 1024 \rightarrow 2048 \rightarrow 4096 \rightarrow m$, each followed by a LeakyReLU activation function except for the last one. Its weights are randomly initialized and frozen before training. The predictor network \mathbf{f}_θ mirrors the target network's first three layers but includes two additional fully connected layers with ReLU activation and decreasing dimensions, i.e., $\dim(\mathbf{O}^e) \rightarrow 1024 \rightarrow 2048 \rightarrow 4096 \rightarrow 2048 \rightarrow 1024 \rightarrow m$. The RND score is calculated as the pairwise distance between the outputs of the predictor and the target network. We use the same network architecture across all environments, and only the input dimension $\dim(\mathbf{O}^e)$ varies between the environments. The training parameters for the RND-OE models are given in Table 3.

A.3 Conformal Prediction

For clarity, we restate the conformal prediction (CP) problem we consider for failure prediction in the following. We assume the availability of a calibration dataset $\mathcal{D}_c = \{\boldsymbol{\tau}^i\}_{i=1}^M \sim q_\pi$ of M independent

Table 3: Training parameters of the RND-OE model.

Parameter	Value
Batch size	256
Epochs	250
Learning rate	1×10^{-4}
Learning rate scheduler	cosine
Optimizer	AdamW
Optimizer weight decay	1×10^{-5}
Optimizer epsilon	1×10^{-8}
Train/validation split	0.9/0.1

and identically distributed (i.i.d) successful policy rollouts and define $\mathcal{I} = \{1, \dots, M\}$. For a new successful rollout $\tau = (\mathbf{O}_0, \mathbf{A}_0, \mathbf{O}_h, \mathbf{A}_h, \dots, \mathbf{O}_{T'}, \mathbf{A}_{T'}) \sim q_\pi$ from the same distribution, we aim for an upper bound δ on the probability that this rollout is incorrectly flagged as Fail, i.e.,

$$\mathbb{P}(\exists t \in \{0, h, \dots, T'\} \text{ s.t. } \eta(\tau_{:t}) > \gamma_t) \leq \delta, \quad (14)$$

where $\eta(\tau_{:t})$ and γ_t are the score function and corresponding threshold of the failure predictor $F(\tau_{:t}) = \mathbb{1}(\eta(\tau_{:t}) > \gamma_t)$ (see Sections 4.1 and 4.2). To achieve this, the threshold γ_t needs to be defined using a suitable nonconformity measure. A simple (but conservative) approach is to consider the maximum failure prediction score over an entire calibration rollout

$$\bar{\eta}(\tau^i) = \max_t \eta(\tau_{:t}^i), \quad i \in \mathcal{I}. \quad (15)$$

Theorem 1 (Adapted from [3], Theorem D.1). *Let τ^1, \dots, τ^M and τ be i.i.d.. Then, defining*

$$\gamma = \inf \left\{ \beta : \frac{|\{i \in \mathcal{I} : \bar{\eta}(\tau^i) \leq \beta\}|}{M} \geq \frac{\lceil (M+1)(1-\delta) \rceil}{M} \right\} \quad (16)$$

as the empirical $\frac{\lceil (M+1)(1-\delta) \rceil}{M}$ quantile of $\{\bar{\eta}(\tau^i) : i \in \mathcal{I}\}$ ensures that

$$\mathbb{P}(\bar{\eta}(\tau) \leq \gamma) \geq 1 - \delta. \quad (17)$$

Proposition 2 (Bounded FPR with a CP constant threshold). *Suppose the calibration rollouts τ^1, \dots, τ^M and the test rollout τ are i.i.d., and let γ be defined according to (16). Then, setting $\gamma_t = \gamma$ ensures that (14) holds.*

Proof. The definition of the nonconformity score (15) implies that

$$\bar{\eta}(\tau) = \max_t \eta(\mathbf{O}_0, \mathbf{A}_0, \dots, \mathbf{O}_t, \mathbf{A}_t) \geq \eta(\mathbf{O}_0, \mathbf{A}_0, \dots, \mathbf{O}_t, \mathbf{A}_t) = \eta(\tau_{:t}), \quad \forall t \in \{0, h, \dots, T'\}. \quad (18)$$

Therefore,

$$\mathbb{P}(\exists t \in \{0, h, \dots, T'\} \text{ s.t. } \eta(\tau_{:t}) > \gamma_t) \leq \mathbb{P}(\exists t \in \{0, h, \dots, T'\} \text{ s.t. } \bar{\eta}(\tau) > \gamma_t) \quad (19)$$

$$= \mathbb{P}(\bar{\eta}(\tau) > \gamma) \quad (20)$$

$$= 1 - \mathbb{P}(\bar{\eta}(\tau) \leq \gamma) \quad (21)$$

$$\leq \delta, \quad (22)$$

where we have used $\gamma_t = \gamma$ in (20) and Theorem 1 in (22). \square

We can obtain a tighter set of thresholds by defining a CP band, as described by Diquigiovanni et al. [15]. For clarity, we define $\eta_i(t) = \eta(\tau_{:t}^i)$ and $\eta(t) = \eta(\tau_{:t})$. We split the set \mathcal{I} into two disjoint parts, \mathcal{I}_1 and \mathcal{I}_2 , of sizes M_1 and $M_2 = M - M_1$, respectively. The first set is used to calculate the sample functional mean as

$$\mu(t) = \frac{1}{M_1} \sum_{i=1}^M \mathbb{1}(i \in \mathcal{I}_1) \eta_i(t), \quad t = 0, 1, \dots, T. \quad (23)$$

The maximum deviation from the mean for a calibration rollout $i \in \mathcal{I}_2$, scaled by a modulation function $s(t)$ such as, for example, $s(t) = 1/T$, is given by

$$R_i^s = \max_t \left| \frac{\eta_i(t) - \mu(t)}{s(t)} \right|. \quad (24)$$

Theorem 2 (Adapted from [15], Appendix A.3). *Let τ^1, \dots, τ^M and τ be i.i.d., and let $\mu(t)$ be the sample functional mean defined according to (23). Then, defining k^s as the empirical $1 - \delta$ quantile of $\{R_i^s : i \in \mathcal{I}_2\}$ guarantees that*

$$\mathbb{P}(\eta(t) \in [\mu(t) - k^s s(t), \mu(t) + k^s s(t)], \forall t) \geq 1 - \delta. \quad (25)$$

For more information on the construction of alternative modulation functions $s(t)$, refer to Diquigiovanni et al. [15].

Proposition 3 (Bounded FPR with a one-sided CP band). *Suppose the calibration rollouts τ^1, \dots, τ^M and the test rollout τ are i.i.d., and let $\mu(t)$ and k^s be defined according to (23) and Theorem 2, respectively. Then, setting $\gamma_t = \mu(t) + k^s s(t)$ ensures that (14) holds.*

Proof. Theorem 2 implies that

$$1 - \delta \leq \mathbb{P}(\eta(t) \in [\mu(t) - k^s s(t), \mu(t) + k^s s(t)], \forall t) \quad (26)$$

$$\leq \mathbb{P}(\eta(t) \in (-\infty, \mu(t) + k^s s(t)], \forall t) \quad (27)$$

$$= \mathbb{P}(\eta(t) \leq \mu(t) + k^s s(t), \forall t) \quad (28)$$

Moreover, we can rewrite the probability term in (14) as

$$\mathbb{P}(\exists t \in \{0, h, \dots, T'\} \text{ s.t. } \eta(\tau_{:t}) > \gamma_t) = 1 - \mathbb{P}(\eta(\tau_{:t}) \leq \gamma_t, \forall t) \quad (29)$$

Substituting $\eta(\tau_{:t}) = \eta(t)$ and $\gamma_t = \mu(t) + k^s s(t)$ and plugging (28) into (29) directly yields (14), which concludes the proof. \square

Remark 2. In addition to the one-sided CP band described above, we have also tested a simpler time-varying threshold $\gamma_t = \mu(t) + \chi(1 - \delta)\sigma(t)$, where $\chi(\cdot)$ is the inverse cumulative distribution function of the univariate Gaussian, and $\sigma(t)$ is the standard deviation of $\{\eta_i(t) : i \in \mathcal{I}_2\}$ with respect to $\mu(t)$. This threshold definition treats the rollout timesteps independently and avoids taking the maximum over t , as in (15) and (24), resulting in a more sensitive failure predictor that is more likely to raise warnings, albeit at the cost of an increase in false alarms. A performance comparison of all three threshold definitions is provided in Figure 12.

A.4 Logical Combination

FIPER combines the outputs of the observation-based failure predictor (3) and the action-based failure predictor (6) with a logical conjunction (AND). Consequently, a rollout is only flagged as Fail if both predictors exceed their respective thresholds, ensuring high robustness against benign OOD situations that the policy can handle. If both thresholds $\gamma_{O,t}$ and $\gamma_{A,t}$ are separately calibrated according to Proposition 2 or Proposition 3, FIPER satisfies the same upper bound on the FPR as the individual failure predictors.

Proposition 4 (Proposition 1 extended). *Set $\delta \in (0, 1)$, and define the thresholds $\gamma_{O,t}$ and $\gamma_{A,t}$ for the failure prediction scores $\eta_O(\tau_{:t})$ and $\eta_A(\tau_{:t})$ using conformal prediction according to Propositions 2 or 3. Then, the probability that the failure predictor (7) of FIPER incorrectly flags a new successful ID rollout $\tau \sim q_\pi$ of length $T' \leq T$ as Fail at any policy timestep $t \leq T'$ satisfies the upper bound*

$$\mathbb{P}(\exists t \in \{0, h, \dots, T'\} \text{ s.t. } F(\tau_{:t}) = 1) \leq \delta. \quad (30)$$

Proof. For each timestep $t \in \{0, h, \dots, T'\}$, we have

$$\mathbb{P}(F(\tau_{:t}) = 1) = \mathbb{P}(F_O(\tau_{:t}) = 1 \wedge F_A(\tau_{:t}) = 1) \quad (31)$$

$$= \mathbb{P}(F_A(\tau_{:t}) = 1) \mathbb{P}(F_O(\tau_{:t}) = 1 \mid F_A(\tau_{:t}) = 1) \quad (32)$$

$$\leq \mathbb{P}(F_A(\tau_{:t}) = 1). \quad (33)$$

The result directly follows from the fact that $\mathbb{P}(\exists t \in \{0, h, \dots, T'\} \text{ s.t. } F_A(\tau_{:t}) = 1) \leq \delta$ by definition of $\gamma_{A,t}$. \square

As a natural alternative to the logical conjunction, we have also investigated a logical disjunction (OR) of the two outputs. This variant can predict *all* failures, but it often triggers false alarms, as shown in Table 2. Since FIPER also achieves a high TPR of 0.92 with the logical AND, combined with a much lower rate of false alarms and a higher TWA, we argue that the logical conjunction of the observation- and action-based failure predictor is the better choice for accurate and robust failure prediction. As an alternative to combining the detector outputs, we have also investigated a weighted combination of the normalized scores (1) and (4). However, this approach would add an additional (task-dependent) hyperparameter and did not show superior performance in our experiments.

B Additional Experimental Details

B.1 Environment Details

Table 4: Details about our environments and datasets.

Environment	SORTING	STACKING	PUSHT	PRETZEL	PUSHCHAIR
Type	sim	sim	sim	real-world	real-world
$\dim(\mathbf{o})$	[2, 3, 96, 96]	[2, 3, 96, 96]	[3, 512, 512]	[3, 240, 320]	[3, 270, 480]
$\dim(\mathbf{O}^e)$	128	128	64	512	96
$\dim(\mathbf{a})$	2	8	2	5	3
$\dim(\mathbf{s})$	3	8	2	5	13
# Calibration rollouts	50	50	50	10	10
# Test rollouts	400	800	300	20	20
# Test rollouts (ID)	100	200	150	0	0
# Test rollouts (OOD)	300	600	150	20	20
Success rate (ID)	0.89	0.60	0.40	-	-
Success rate (OOD)	0.57	0.36	0.29	0.5	0.67
Avg. episode length	47	79	31	55	8
Max. episode length	75	120	38	70	22

This section provides an overview of the environments we use to evaluate our failure prediction framework. We summarize the details about the environments and datasets in Table 4. For PUSHT and PUSHCHAIR, we use the publicly available rollout datasets from Agia et al. [1], which are released under an MIT License.

B.1.1 Simulation Environments

The three simulation tasks, SORTING, STACKING, and PUSHT, are visualized in Figure 7.

SORTING: In this task, a Franka robot must push two blocks into their corresponding color-matching target boxes on a tabletop. The visual observations are obtained from a wrist camera and a third-person camera. To induce more policy failures in our test data, we vary the dimensions of the blocks and the positions of the target boxes. A rollout is considered successful if the policy manages to place both blocks in their respective boxes within the maximum episode length. If one of the blocks falls from the table but does not land inside its corresponding box, it is impossible for the policy to correct this mistake; therefore, we end the rollout prematurely.

STACKING: A Franka robot is trained to stack three colored blocks in a target area, using joint velocities and gripper commands as actions. Similar to SORTING, two RGB images are used as visual inputs. The overall complexity of this task leads to a very low success rate of the complete task (referred to as STACKING-3 in D3IL [29]), especially when introducing OOD scenarios. Therefore, we only consider the simpler STACKING-1 task for testing, which is successfully completed if one of the three blocks has been placed into the target area (see Figure 7). This allows us to include OOD situations by varying the dimensions of the blocks and the location of the target area, while still achieving a sufficiently high success rate to evaluate failure prediction. This task exhibits strong action multimodality, and failures often occur when the robot is unable to grasp a block.

PUSHT: A planar T-shaped object needs to be pushed into a target configuration. We use the rollout data from Agia et al. [1], which includes variations of the scale and dimensions of the T-object. A rollout is considered successful if the policy manages to push the “T” into the goal area with at least

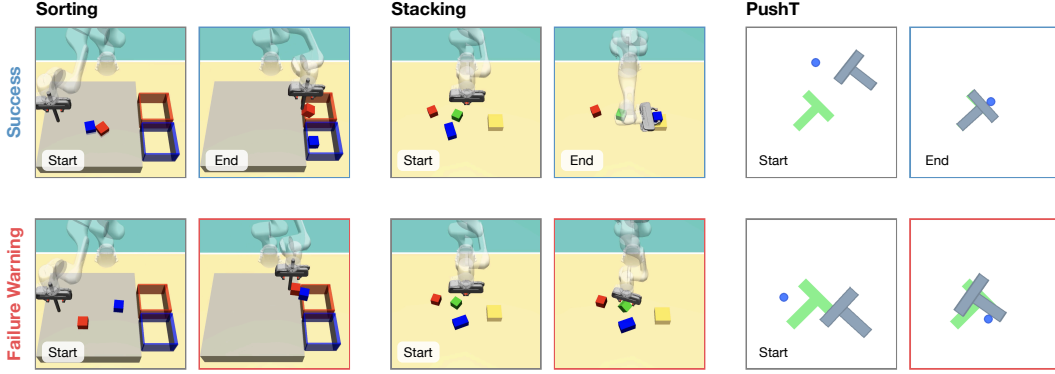


Figure 7: Overview of the simulation tasks with exemplary successful and failed rollouts.

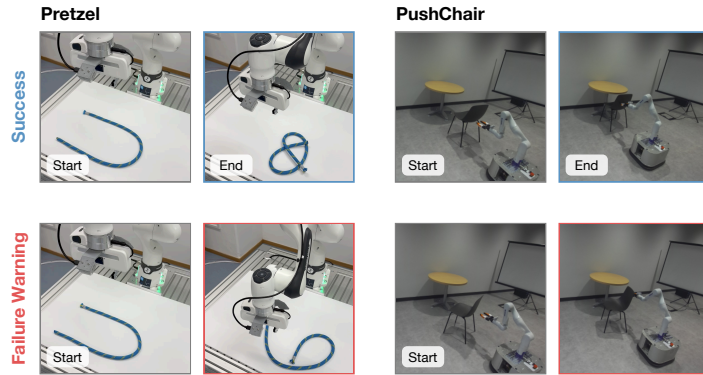


Figure 8: Overview of the real-world tasks with exemplary successful and failed rollouts.

90% overlap. Since the policy has learned to approach the “T” from different directions, this task exhibits strong action multimodality. Failures for this task can occur when the policy is unable to properly move the “T” due to its changed size, or if it gets stuck in a suboptimal pose, where the “T” somewhat overlaps with the target area but in an incorrect orientation (see failure case in Figure 7).

B.1.2 Real-World Tasks

Figure 8 depicts the two real-world tasks considered in this work, namely Pretzel and PUSHCHAIR.

PRETZEL: In this task, a Franka robot needs to form a rope into the shape of a pretzel. A successfully formed pretzel comprises three intersections of the rope that divide the area into three distinct parts when looking from above, as shown in the successful rollout in Figure 8. We count a rollout as successful if the robot manages to form a correct pretzel within 70 policy inference timesteps, otherwise, it is considered a failure. The maximum episode length is thereby determined by the longest successful rollout in our test set. To invoke more policy failures in our test data, we vary the rope’s initial position and apply rotations about its axis. Because the rope’s material is anisotropic, axial rotations can change its bending behavior. In that case, once the robot places the rope onto the table, the rope does not remain in this configuration but moves back toward its original pose (see failure warning in Figure 8). Unlike variations of the initial conditions, this OOD behavior can not be detected from the initial observation images, but only becomes apparent after multiple timesteps when one end of the rope has been placed. In our experiments, vertical and horizontal position perturbations lead to 6 and 2 failures, respectively, and axial rotations induce an additional 2 failures.

PUSHCHAIR: A single-arm mobile manipulator needs to push a chair to a target position underneath a table. We use the data from Agia et al. [1], which includes variations of the initial chair pose. Failures occur when the chair rotates substantially about the vertical axis, which can happen, for example, if the robot does not push against the center of the backrest (see Figure 8).

Table 5: Implementation details for our generative IL policies based on flow matching (FM) [40] and denoising diffusion probabilistic models (DDPM) [25]. The policy hyperparameters for PushT and PushChair are taken from Agia et al. [1].

Task	Sorting	Stacking	PushT	Pretzel	PushChair
Generative modeling technique	FM	FM	DDPM	DDPM	DDPM
Backbone	ACT [78]	ACT [78]	U-Net [11]	U-Net [11]	U-Net [11]
Observation history length T_h	1	5	1	1	1
Action chunk length H	8	8	16	16	16
Action execution steps h	4	4	8	8	4
Action batch size B	32	32	256	30	256
Integration steps	16	16	100	100	100
Training epochs	20	20	2000	100	2000
Training batch size	256	256	256	64	256
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
Learning rate	5×10^{-4}	5×10^{-4}	3×10^{-5}	1×10^{-4}	3×10^{-5}

B.2 Discussion on other Tasks

While we focus on manipulation in our experiments, we believe that FIPER is also directly applicable to other tasks, such as navigation and locomotion, provided that two conditions are met:

- There is a clear definition of "task failure". This might not be as obvious for some locomotion tasks, such as tracking a desired reference base velocity, which do not have the same episodic character as typical manipulation and navigation tasks. Intuitively, task failure is well-defined if there is a maximum episode length and a clear objective that should be achieved within that time frame. For locomotion, this objective could be defined, for example, as staying within a certain tolerance around the reference speed.
- The actions can be represented in or transformed to Cartesian space. This is not a strict requirement, but we think it is highly beneficial for the reasons mentioned in Appendix A.1. It applies to navigation, where the entropy can be computed for the base positions predicted from the action chunks. For locomotion, one possible approach would be to compute the action entropy for the positions of each foot individually and then combine these scores by averaging or taking the maximum.

B.3 Policy Details

Given a demonstration dataset $\mathcal{D} = \{(\mathbf{A}_0, \mathbf{O}_0), (\mathbf{A}_1, \mathbf{O}_1), \dots\}$ of observation-action chunk pairs sampled from an unknown expert policy $q(\mathbf{A}|\mathbf{O})$, we aim to learn an IL policy $\pi(\mathbf{A}|\mathbf{O})$ that approximates the conditional action expert distribution as much as possible, i.e., $\pi(\mathbf{A}|\mathbf{O}) \approx q(\mathbf{A}|\mathbf{O})$. We consider IL policies based on the two most popular generative modeling techniques, diffusion and flow matching, and denote their learnable parameters by ψ . Details on the implementation of our policies are provided in Table 5. For SORTING and STACKING, we combine flow matching with a transformer backbone. For PUSHT, PRETZEL, and PUSHCHAIR, we combine diffusion with a CNN backbone. We have made this choice independently of the tasks' characteristics to evaluate our method's performance for different policy formulations and architectures.

B.3.1 Diffusion Policy

Denoising diffusion probabilistic models (DDPM) [25] introduce latent variables $\mathbf{A}^0, \dots, \mathbf{A}^R$, where $\mathbf{A}^0 = \mathbf{A}$, and construct a forward diffusion Markov process

$$q(\mathbf{A}^r | \mathbf{A}^{r-1}, \mathbf{O}, r) = \mathcal{N}(\sqrt{1 - \beta_r} \mathbf{A}^{r-1}, \beta_r \mathbf{I}), \quad (34)$$

where $r = 1, \dots, R$ is the diffusion time step, and $\beta_{1:R}$ is a pre-defined noise schedule, such as the cosine noise schedule [48]. The noise schedule is chosen such that $q(\mathbf{A}^R) \approx \mathcal{N}(\mathbf{O}, \mathbf{I})$. The objective is to reverse the forward process by a learnable denoising (backward) process

$$\pi(\mathbf{A}^{r-1} | \mathbf{A}^r, \mathbf{O}, r) = \mathcal{N}(\boldsymbol{\mu}_\psi(\mathbf{A}^r, \mathbf{O}, r), \sigma_r^2 \mathbf{I}), \quad (35)$$

where $\sigma_r^2 = \beta_r \frac{1-\alpha_r-1}{1-\alpha_r}$ with $\alpha_r = \prod_{i=1}^r (1 - \beta_i)$. This can be achieved by learning to predict the Gaussian noise added to \mathbf{A}^0 via the surrogate training loss

$$\mathcal{L}_{\text{DP}}(\psi) = \mathbb{E}_{r \sim \mathcal{U}(\{1, \dots, R\}), (\mathbf{A}^0, \mathbf{O}) \sim \mathcal{D}, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} [\|\epsilon - \epsilon_\psi(\sqrt{\alpha_r} \mathbf{A}^0 + \sqrt{1 - \alpha_r} \epsilon, \mathbf{O}, r)\|_2]. \quad (36)$$

After training, a new action sequence conditioned on an observation \mathbf{O}_t can be generated by first sampling $\mathbf{A}^R \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and then iterating over the learned denoising process (35) with $\mu_\psi(\mathbf{A}^r, \mathbf{O}_t, r) = \frac{1}{\sqrt{1-\beta_r}} \left(\mathbf{A}^r - \frac{\beta_r}{\sqrt{1-\alpha_r}} \epsilon_\psi(\mathbf{A}^r, \mathbf{O}, r) \right)$ for $r = R, \dots, 1$, resulting in $\mathbf{A}^0 \sim \pi(\cdot | \mathbf{O}_t)$.

B.3.2 Flow Matching Policy

Denoising diffusion can be viewed as solving a stochastic differential equation (SDE) [66]. In contrast, flow matching [40] aims to learn the straight velocity field of an ordinary differential equation (ODE) that transports samples from a simple base distribution—typically a standard Gaussian—to the unknown target distribution. To be more consistent with Appendix B.3.1, we slightly deviate from the notation commonly used in the flow matching literature and denote samples from the source and target distribution by $\mathbf{A}^1 \sim q_1 = \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\mathbf{A}^0 \sim q_0 = q(\mathbf{A} | \mathbf{O})$, respectively, instead of the other way around. A flow $\frac{d}{ds} \phi_s(\mathbf{A}) = \mathbf{u}_s(\phi_s(\mathbf{A}))$ determined by a velocity field \mathbf{u}_s induces a probability path q_s from q_1 to q_0 if $\mathbf{A}^s = \phi_s(\mathbf{A}^1) \sim q_s$. Instead of directly approximating the unknown velocity field \mathbf{u}_s , flow matching is trained by conditioning on a target sample $\mathbf{A}^0 \sim q_0$ via the conditional flow matching loss

$$\mathcal{L}_{\text{FM}}(\psi) = \mathbb{E}_{s \sim \mathcal{U}([0,1]), (\mathbf{A}^0, \mathbf{O}) \sim \mathcal{D}, \mathbf{A}^s \sim q_s(\cdot | \mathbf{A}^0)} [\|\mathbf{v}_\psi(\mathbf{A}^s, \mathbf{O}, s) - \mathbf{u}_s(\mathbf{A}^s | \mathbf{A}^0)\|_2]. \quad (37)$$

There are different ways to define the conditional velocity field $\mathbf{u}_s(\mathbf{A}^s | \mathbf{A}^0)$ [41]. We use a simple linear interpolation $\mathbf{A}^s = (1-s)\mathbf{A}^0 + s\mathbf{A}^1$ with $\mathbf{u}_s(\mathbf{A}^s | \mathbf{A}^0) = \frac{\mathbf{A}^0 - \mathbf{A}^s}{s} = \mathbf{A}^0 - \mathbf{A}^1$, which allows simplifying the loss to

$$\mathcal{L}_{\text{FM}}(\psi) = \mathbb{E}_{s \sim \mathcal{U}([0,1]), (\mathbf{A}^0, \mathbf{O}) \sim \mathcal{D}, \mathbf{A}^1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} [\|\mathbf{v}_\psi(\mathbf{A}^s, \mathbf{O}, s) - (\mathbf{A}^0 - \mathbf{A}^1)\|_2]. \quad (38)$$

After training, we can generate a new action chunk $\mathbf{A}^0 \sim q(\cdot | \mathbf{O})$ by sampling $\mathbf{A}^1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and numerically integrating the ODE with the learned velocity field, for example, through a simple Euler integration

$$\mathbf{A}^{s-\delta s} = \mathbf{A}^s + \delta s \mathbf{v}_\psi(\mathbf{A}^s, \mathbf{O}, s) \quad (39)$$

with $\lceil 1/\delta s \rceil$ steps.

B.4 Metrics

We denote the number of failed rollouts (positives) by #P and the number of successful rollouts (negatives) by #N. The number of true positives, i.e., failed rollouts correctly flagged as Fail by the failure predictor, is denoted by #TP. Similarly, #FP, #TN, and #FN are the numbers of false positives, true negatives, and false negatives, respectively. It holds that #TP + #FN = #P and #FP + #TN = #N. The accuracy and balanced accuracy are given by

$$\begin{aligned} \text{ACCURACY} &= \frac{\#TP + \#TN}{\#P + \#N}, \\ \text{BALANCED ACCURACY} &= \frac{1}{2} \left(\frac{\#TP}{\#P} + \frac{\#TN}{\#N} \right). \end{aligned}$$

We introduce a novel metric to measure the two desired capabilities of a failure predictor, raising a failure warning correctly *and* early, with a single scalar. Denoting the detection times of true positives by $t_i, i = 1, \dots, \#TP$, we define the (balanced) timestep-wise-accuracy (TWA) as

$$\text{TWA} = \frac{\sum_i (1 - t_i/T) + \#TN}{\#P + \#N}, \quad (40)$$

$$\text{BALANCED TWA} = \frac{1}{2} \left(\frac{\sum_i (1 - t_i/T)}{\#P} + \frac{\#TN}{\#N} \right). \quad (41)$$

Our idea behind this metric (and the reason for its name) is as follows: When considering a failed episode of length T which is correctly flagged as Fail at time t , the failure predictor is correct about the final rollout outcome only at timesteps t, \dots, T . Thus, the proportion of timesteps for which the prediction is correct is given by $\frac{T-t}{T} = 1 - \frac{t}{T}$.

To avoid bias from imbalanced datasets with an unequal number of failed and successful rollouts, we report the balanced accuracy and balanced TWA throughout this work, omitting "balanced" for brevity.

B.5 Baselines

PCA-kmeans: Liu et al. [42] train a failure detector on image embeddings of a visual world model, assuming the availability of failure trajectories. Using Principal Component Analysis (PCA) to reduce the embedding dimension and k-means clustering to calculate 64 centroids, failure situations are detected based on the distance to the closest centroid. For our experiments, we reimplement the method for the policy embedding space and compute the clusters from successful ID rollouts.

logpZO: Xu et al. [75] propose a failure detection approach based on learning the distribution of observation embeddings via flow matching. For a new observation O_t , the ODE is solved backwards starting from O_t^e via the learned velocity field, yielding a latent noise sample Z_{O_t} . The dissimilarity score is determined from the likelihood of Z_{O_t} under the Gaussian source distribution and given by $\|Z_{O_t}\|_2^2$. We reimplement this approach as a baseline, adopting a U-Net architecture as proposed by the authors.

STAC: Agia et al. [1] propose measuring the cumulative statistical temporal action consistency (STAC) to detect policy failures. STAC calculates the divergence between the temporally overlapping components of the action distributions at two consecutive policy timesteps using the maximum mean discrepancy (MMD) with a radial basis function (RBF) kernel. We directly adopt STAC as a baseline, using some of the code released by the authors under an MIT License. Through our experiments, we observe that STAC struggles with observation-dependent action multimodality, a characteristic that most of our tasks exhibit. The reason for this is that the temporally overlapping parts of the action sequence batches directly before and after committing to a behavior mode are often very different because their generation is conditioned on *different* observations, as visualized in Figure 3 d).

RND-A: He et al. [22] use RND trained on trajectories to learn a confidence score that measures the reliability of generated actions for successful task completion. We adapt this approach to action chunks, adopting some of the code released by the authors under an MIT License, and calibrate the resulting uncertainty scores to predict policy failures.

Several recent works have proposed using VLMs to externally monitor a robot and reason about erroneous behavior [1, 17]. However, because these methods lack information about the policy’s input and output distribution, they are inherently unsuitable for predicting failures early before they occur, which is one of the main objectives of this work. For this reason, we do not use VLM-based approaches as baselines.

B.6 Compute Resources

We conduct all experiments on a workstation with 64 GB of RAM, an NVIDIA GeForce RTX 4090 GPU, and an Intel Core i9-285 K CPU. Evaluating FIPER and the baselines for all tasks, window sizes, and quantile values takes about one hour per seed. We note that computing the ACE score and running generative (vision-based) policies could be challenging on some mobile platforms with limited computational resources.

C Extended Results

C.1 Uncertainty Scores

As a supplement to Figure 4, we show the distribution of uncertainty scores for the four rollout categories Success ID, Success OOD, Fail ID, and Fail OOD, in Figure 9. Here, we consider only

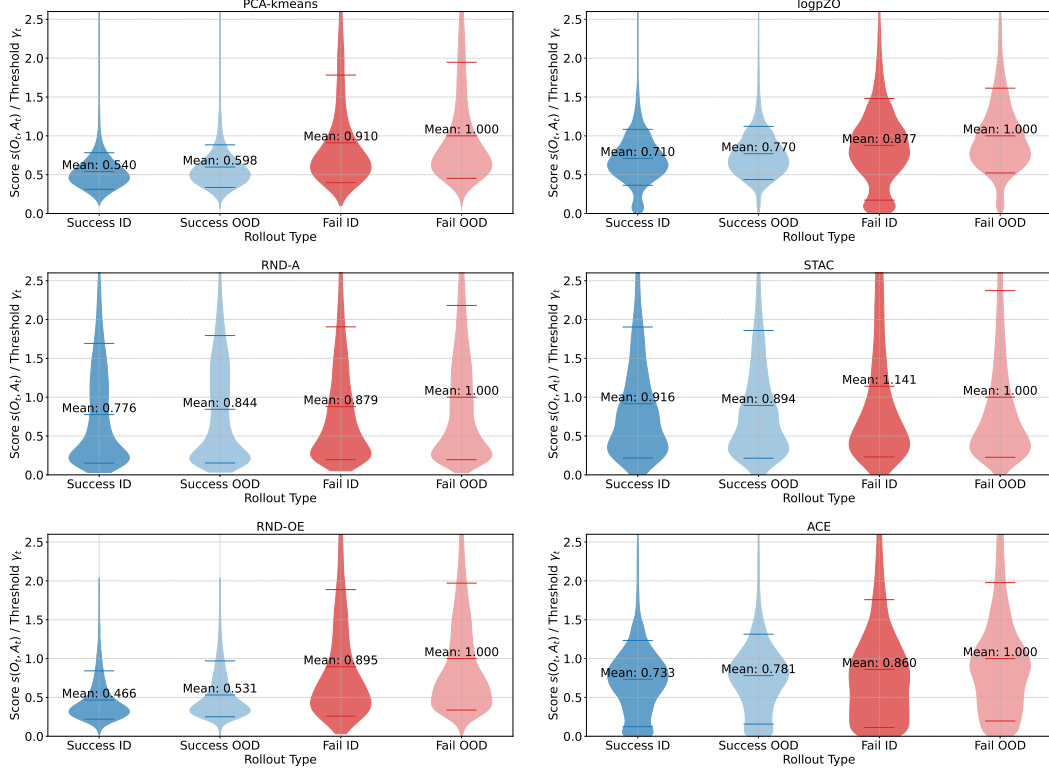


Figure 9: Distribution of the uncertainty scores $s(\mathbf{O}_t, \mathbf{A}_t)$, normalized by the corresponding CP band threshold γ_t , for different types of rollouts. We divide all values by the mean for Fail OOD for better comparability. The top and bottom horizontal lines correspond to the 90% and 10% quantiles, respectively.

the three simulation tasks, as there is no distinction between ID and OOD rollouts for the real-world tasks (see Table 4). RND-OE shows a clearer distinction between successful OOD rollouts and failed ID rollouts than the other observation-based scores, PCA-kmeans and logpZO, mainly because high RND-OE scores occur much more frequently in failure rollouts than in successful rollouts, regardless of ID or OOD. The data for the ACE score has outliers because the time-varying threshold sometimes takes very small values, which makes it difficult to visually interpret the distribution. Nonetheless, high values occur much more frequently for Fail ID than Success OOD rollouts, and our action-based failure predictor alone achieves the second-highest accuracy, which underlines the strengths of ACE to distinguish failed from successful rollouts. Moreover, our proposed approach of aggregating uncertainty scores over a sliding window increases the robustness against such outliers in the calibration rollouts.

C.2 Extended Baseline Comparison

Table 6 is an extended version of Table 1, additionally including the TPR and TNR. To avoid selecting a window size w that is more beneficial for some methods but less so for others, we use the value of w that yields the highest TWA across all tasks. This “best” window size is also reported in Table 6. The results show that no baseline achieves a TPR and TNR greater than 0.4 for all environments, while the lowest value with FIPER is a TNR of 0.44 for the PUSH T environment. This highlights that FIPER can predict failures and recognize OOD situations that lead to success more robustly than existing approaches. Compared to our two primary baselines, STAC and logpZO, which are both designed specifically for failure detection, FIPER achieves a notably higher overall TPR, while at the same time not suffering from a low TNR like PCA-kmeans. Importantly, FIPER’s strong performance is evident across a diverse set of environments, rather than being limited to a particular type of robot embodiment or task.

Table 6: We compare FIPER and its individual failure predictors (3) and (6) to four state-of-the-art baselines for OOD and failure detection. We highlight the best and underline the second-best value in each row. Detection times (DT) in brackets have corresponding TPR or TNR below 0.4, both of which indicate poor discrimination between successes and failures. We report the mean and standard deviation across five seeds. This table is an extension of Table 1.

Task	Metric	PCA-kmeans [42]	logpZO [75]	RND-A [22]	STAC [1]	RND-OE	ACE	FIPER
Best w		40	15	2	15	1	45	25/50
Best threshold		time-var.	CP constant	CP constant	CP constant	CP constant	time-var.	time-var.
SORTING	TWA \uparrow	0.49 ± 0.00	<u>0.54 ± 0.00</u>	0.55 ± 0.01	0.46 ± 0.00	0.52 ± 0.01	<u>0.54 ± 0.00</u>	<u>0.54 ± 0.00</u>
	Acc. \uparrow	0.56 ± 0.00	0.67 ± 0.00	0.62 ± 0.01	0.48 ± 0.00	0.59 ± 0.02	<u>0.65 ± 0.00</u>	<u>0.66 ± 0.00</u>
	DT \downarrow	(0.15 ± 0.00)	0.48 ± 0.01	0.34 ± 0.03	(0.46 ± 0.00)	(0.17 ± 0.03)	0.30 ± 0.00	<u>0.32 ± 0.00</u>
	TPR \uparrow	0.93 ± 0.00	0.57 ± 0.01	0.43 ± 0.09	0.09 ± 0.00	0.92 ± 0.03	0.76 ± 0.00	0.75 ± 0.00
	TNR \uparrow	0.20 ± 0.00	0.77 ± 0.01	0.81 ± 0.08	0.86 ± 0.00	0.26 ± 0.06	0.54 ± 0.00	0.57 ± 0.00
STACKING	TWA \uparrow	0.66 ± 0.00	0.58 ± 0.00	0.50 ± 0.00	0.59 ± 0.00	<u>0.62 ± 0.04</u>	<u>0.62 ± 0.00</u>	<u>0.62 ± 0.00</u>
	Acc. \uparrow	0.75 ± 0.00	0.69 ± 0.01	0.56 ± 0.01	0.66 ± 0.00	0.70 ± 0.06	<u>0.73 ± 0.00</u>	<u>0.73 ± 0.00</u>
	DT \downarrow	<u>0.19 ± 0.00</u>	0.49 ± 0.00	(0.44 ± 0.01)	(0.38 ± 0.00)	0.16 ± 0.05	<u>0.27 ± 0.00</u>	0.28 ± 0.00
	TPR \uparrow	1.00 ± 0.00	0.48 ± 0.02	0.27 ± 0.04	0.35 ± 0.00	0.98 ± 0.02	0.84 ± 0.00	0.84 ± 0.00
	TNR \uparrow	0.51 ± 0.00	0.90 ± 0.00	0.85 ± 0.02	0.96 ± 0.00	0.42 ± 0.14	0.61 ± 0.00	0.62 ± 0.00
PUSHT	TWA \uparrow	0.53 ± 0.00	0.52 ± 0.00	0.52 ± 0.01	0.58 ± 0.00	0.54 ± 0.01	<u>0.56 ± 0.00</u>	0.55 ± 0.00
	Acc. \uparrow	0.58 ± 0.00	0.55 ± 0.00	0.55 ± 0.01	0.71 ± 0.00	0.55 ± 0.01	<u>0.71 ± 0.00</u>	0.71 ± 0.00
	DT \downarrow	(0.11 ± 0.00)	(0.26 ± 0.00)	(0.20 ± 0.02)	0.52 ± 0.00	(0.02 ± 0.00)	0.31 ± 0.00	<u>0.32 ± 0.00</u>
	TPR \uparrow	1.00 ± 0.00	0.17 ± 0.00	0.31 ± 0.03	0.49 ± 0.00	0.80 ± 0.04	0.99 ± 0.00	0.98 ± 0.00
	TNR \uparrow	0.17 ± 0.00	0.93 ± 0.00	0.80 ± 0.02	0.93 ± 0.00	0.31 ± 0.06	0.43 ± 0.00	0.44 ± 0.00
PRETZEL	TWA \uparrow	0.64 ± 0.00	0.58 ± 0.03	0.51 ± 0.05	0.51 ± 0.00	0.55 ± 0.02	0.75 ± 0.00	<u>0.68 ± 0.03</u>
	Acc. \uparrow	0.65 ± 0.00	0.65 ± 0.04	0.53 ± 0.05	0.67 ± 0.00	0.72 ± 0.02	<u>0.82 ± 0.00</u>	0.85 ± 0.00
	DT \downarrow	(0.01 ± 0.00)	<u>0.24 ± 0.04</u>	(0.52 ± 0.36)	0.44 ± 0.00	0.44 ± 0.02	0.13 ± 0.00	0.33 ± 0.07
	TPR \uparrow	1.00 ± 0.00	0.54 ± 0.05	0.13 ± 0.12	0.69 ± 0.00	0.77 ± 0.12	1.00 ± 0.00	1.00 ± 0.00
	TNR \uparrow	0.30 ± 0.00	0.76 ± 0.05	0.92 ± 0.06	0.64 ± 0.00	0.67 ± 0.09	0.64 ± 0.00	0.70 ± 0.00
PUSHCHAIR	TWA \uparrow	0.50 ± 0.00	<u>0.78 ± 0.02</u>	0.71 ± 0.05	0.73 ± 0.00	0.74 ± 0.11	0.69 ± 0.00	0.83 ± 0.02
	Acc. \uparrow	0.50 ± 0.00	<u>0.92 ± 0.02</u>	0.82 ± 0.05	0.88 ± 0.00	0.80 ± 0.14	0.80 ± 0.00	0.96 ± 0.02
	DT \downarrow	(0.00 ± 0.00)	0.26 ± 0.01	<u>0.21 ± 0.02</u>	0.30 ± 0.00	0.11 ± 0.07	0.23 ± 0.00	0.27 ± 0.00
	TPR \uparrow	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	0.98 ± 0.03	1.00 ± 0.00	1.00 ± 0.00
	TNR \uparrow	0.00 ± 0.00	0.83 ± 0.03	0.64 ± 0.10	0.75 ± 0.00	0.61 ± 0.30	0.60 ± 0.00	0.93 ± 0.04
Average	TWA \uparrow	0.57 ± 0.00	0.60 ± 0.01	0.56 ± 0.02	0.57 ± 0.00	0.59 ± 0.04	<u>0.63 ± 0.00</u>	0.65 ± 0.01
	Acc. \uparrow	0.61 ± 0.00	0.69 ± 0.01	0.62 ± 0.03	0.68 ± 0.00	0.67 ± 0.05	<u>0.74 ± 0.00</u>	0.78 ± 0.00
	DT \downarrow	(0.09 ± 0.00)	0.35 ± 0.01	0.34 ± 0.09	0.42 ± 0.00	0.18 ± 0.03	<u>0.25 ± 0.00</u>	0.30 ± 0.02
	TPR \uparrow	0.99 ± 0.00	0.55 ± 0.02	0.43 ± 0.06	0.52 ± 0.00	0.89 ± 0.05	<u>0.92 ± 0.00</u>	0.92 ± 0.00
	TNR \uparrow	0.24 ± 0.00	0.84 ± 0.02	0.80 ± 0.06	0.83 ± 0.00	0.45 ± 0.13	0.56 ± 0.00	0.65 ± 0.01

C.3 Impact of the Window Size for Score Aggregation

Complementary to our discussion in Section 6, we provide more detailed results about our proposed approach of aggregating uncertainty scores over a sliding window of length w . Figure 10 shows the impact of w for a different threshold definitions, averaged across quantiles $1 - \delta \in \{0.9, 0.91, \dots, 0.99\}$. For FIPER, we set $w_O = w_A = w$ in this analysis for simplicity. Using $w = 1$, i.e., considering only the current observation and/or action [75], leads to low failure prediction accuracy. Increasing the window size improves accuracy, but mostly at the expense of slower detection. In general, we observe that the optimal window size with respect to TWA varies significantly across different failure prediction methods and threshold types. This supports our approach of reporting the main results in Tables 1 and 6 for the window size that achieves the highest TWA across all tasks for the respective method.

It mainly depends on the task whether a decrease in DT justifies a drop in accuracy. In domains where rapid response to imminent failures is crucial, such as autonomous surgical assistance or collaborative assembly lines, earlier failure warnings may be highly desirable, even if they mean more false alarms. Conversely, in a logistics scenario, misplacement of objects is often not dangerous, whereas every incorrect failure warning would have to be checked manually by a human and would thus be costly. In this case, a more conservative aggregation over a longer window may better balance operational efficiency and failure prevention.

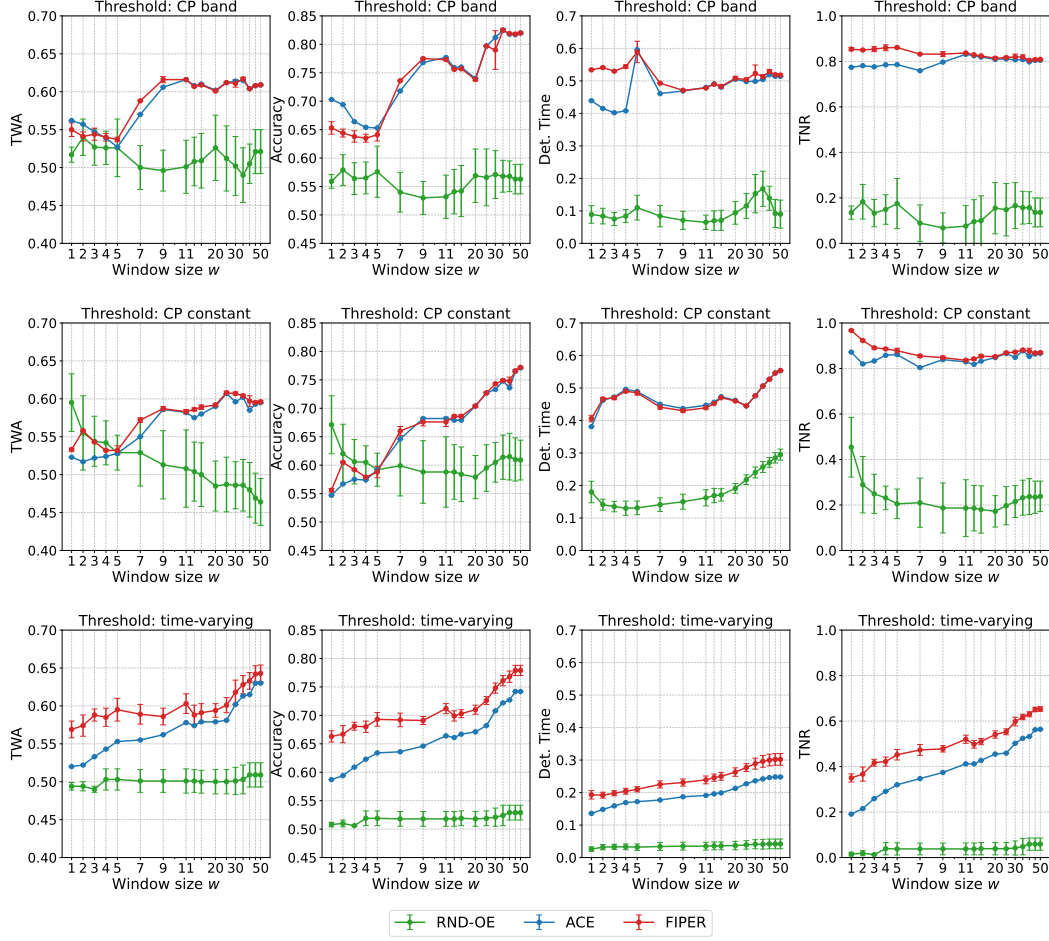


Figure 10: Impact of the sliding window size w for three threshold types. The results are averaged across five seeds, with the bars indicating the standard deviation.

C.4 Impact of the Threshold Computation

The performance of a failure predictor is affected by the definition of its threshold, in particular, by (i) the type of threshold and (ii) the design parameter δ for calibration, which controls the desired FPR. We have conducted multiple ablation studies to analyze the impact of these factors.

C.4.1 Quantiles

We average our results over a set of quantiles $1 - \delta \in \{0.9, 0.91, \dots, 0.99\}$, arguing that there is no “best” quantile value for calibration, and because we want to avoid cherry-picking a specific quantile that is more favorable to the performance of one method than others. In Figure 11, we show the impact of different quantile values $1 - \delta$ on TWA, DT, TPR, and TNR, using an exemplary window size of $w = 15$ for all methods. We observe that $1 - \delta$ directly affects both TPR and TNR. This is expected, as the quantile value effectively controls the threshold at which a rollout is flagged as Fail. Consequently, increasing $1 - \delta$ leads to fewer predicted failures, which decreases TPR and increases TNR. These two effects largely cancel out in terms of overall accuracy and TWA, leading us to the above claim that there is no “best” quantile.

False Positive Rate (FPR) As discussed in Appendix A.3, the hyperparameter $\delta \in (0, 1)$ effectively controls the FPR on the calibration rollouts. As shown in Figure 11, the CP constant threshold and one-sided CP band defined in Propositions 2 and 3 lead to very high TNR but fairly low TPR. In contrast, our simpler time-varying threshold yields a much higher TPR at the cost of raising false

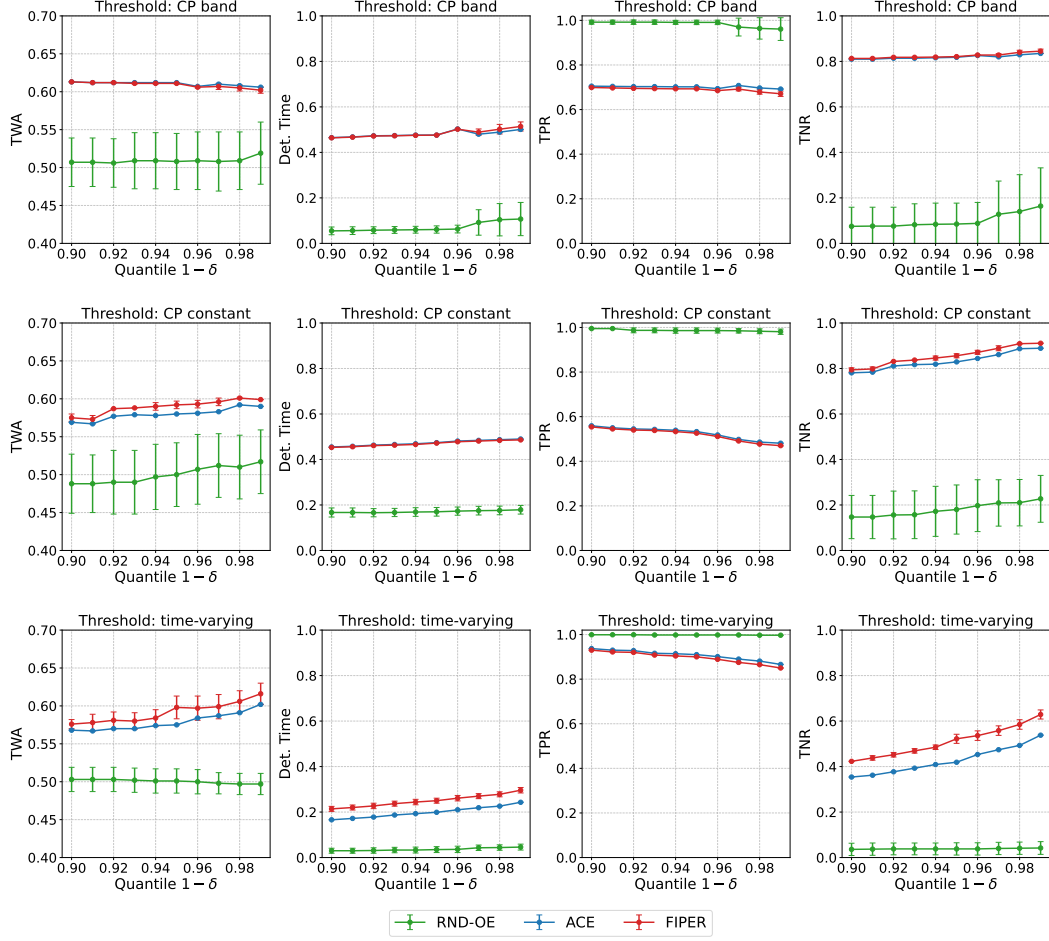


Figure 11: Impact of the quantile value $1 - \delta$ used for calibration for three threshold types. The results are reported for a sliding window size of $w = 15$ and averaged across five seeds, with the bars indicating the standard deviation.

alarms more frequently. Which threshold type is most suitable, therefore, depends on whether recognizing all failures or avoiding false alarms has higher priority. The fact that neither a constant threshold nor a one-sided CP band achieves the desired FPR exactly matches the results reported in prior works [1, 75] and stems from the fact that Propositions 2 and 3 only hold for ID rollouts, whereas we also evaluate on OOD scenarios. We hypothesize that strictly separating the rollouts used for training the failure predictors from those used for calibration, as well as using a larger calibration dataset, could mitigate this issue.

C.4.2 Threshold Type

For our main results in Tables 1 and 6, we report the results with the best-performing threshold definition (CP band, CP constant, or time-varying) for each method. While no single threshold type consistently outperforms the others across all methods, we perform a direct comparison in Figure 12 and analyze the results below.

CP-based thresholds detect rather than predict failures. Since the CP band and CP constant thresholds are defined such that the probability of raising a false alarm is upper-bounded, their values γ_t are higher than those of the time-varying threshold that is calculated timestep-wise. Consequently, the CP-based thresholds yield a much higher TNR but also predict failures much later. FIPER achieves the highest TWA values overall with our time-varying threshold and a larger window size. While accuracy may be more important than early detection in some cases, we argue that

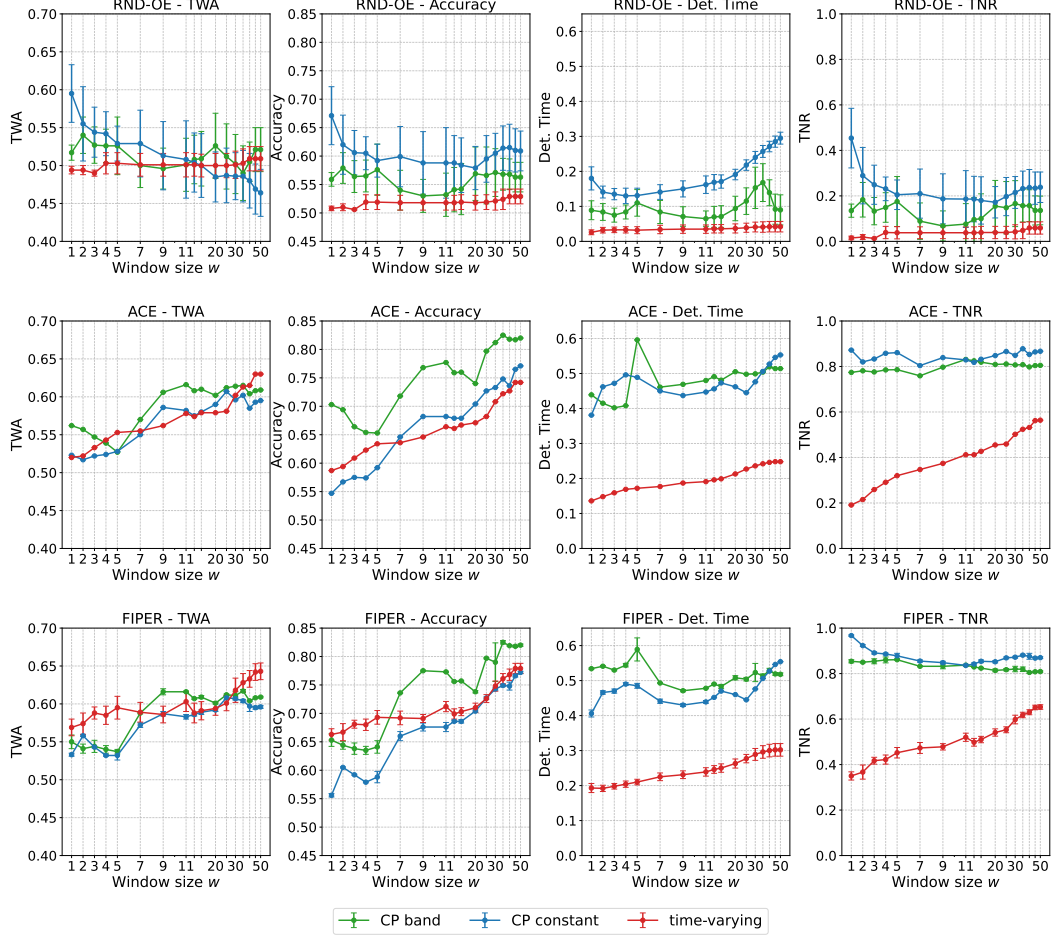


Figure 12: Impact of the threshold type used by the failure predictors. The results are averaged across five seeds, with the bars indicating the standard deviation.

the conservative nature of CP-based thresholds is more suitable for failure detection rather than prediction.

Constant thresholds are less accurate for FIPER The CP band and time-varying thresholds yield higher TWA and accuracy for FIPER. We attribute this to the fact that these threshold types effectively compare the failure prediction score at timestep t against the scores in the calibration dataset for the *same* timestep. Consequently, they are more sensitive to temporal anomalies in the score values, which can stem, for example, from the robot failing to grasp an object. However, the higher TNR of the constant threshold shows that the CP band and time-varying threshold raise false alarms more frequently in generalization scenarios, where the policy may solve subtasks in varying order or with inconsistent timing.

C.5 Discussion of Action-Based Scores

We briefly compare our action-based failure predictor, ACE, specifically against the two baselines that also operate on the policy output, RND-A and STAC. The results in Table 6 demonstrate that ACE achieves higher TWA and accuracy and lower DT than RND-A and STAC. This demonstrates the superiority of our proposed ACE score for quantifying task-relevant uncertainty from the generated actions. STAC [1], a recent baseline, notably achieves a TPR below 0.5 in the SORTING, STACKING, and PUSHT environments that are characterized by strong action multimodality. In comparison, our entropy-based score can predict between 76% and 99% of all failures for these tasks. ACE is specifically designed to capture the uncertainty inherent in multimodal action distributions, and

it outperforms STAC’s temporal consistency score in our experiments. Figure 3 illustrates the conceptual advantage of measuring entropy instead of temporal consistency in the action distributions in tasks that involve action multimodality.

C.6 Discussion of Observation-Based Scores

RND-OE yields a clearer separation between successful and failed rollouts than the observation-based baselines, PCA-kmeans and logpZO, as shown in Figures 4 and 9. Moreover, RND-OE can predict failures more rapidly and achieves the lowest DT overall. This can be especially beneficial in scenarios involving humans, where high sensitivity to different kinds of failures is crucial. In comparison, PCA-kmeans suffers from a low overall TNR, effectively labeling most rollouts as failures (see Table 6). Although logpZO can match the accuracy and TWA of RND-OE (Table 6), our RND-OE score achieves much lower DT, highlighting its advantages for predicting failures before they occur. We find that the design of FIPER benefits from the high sensitivity of RND-OE because a rollout is only flagged as `Fail` if there is also high action uncertainty.

We also considered RND models that take the raw observation images rather than embeddings as input, which is closer to the original version of RND proposed by Burda et al. [9]. However, operating directly in the embedding space of the policy performed significantly better, and we argue that this approach offers two key advantages: First, because the observation encoder is trained end-to-end with the policy, the embeddings naturally filter out details in the observations that are irrelevant to the policy and, thus, for reasoning about task failure. Conversely, performing OOD detection directly on raw images can yield a high FPR in real-world tasks, as the RND model cannot separate visual noise from policy-relevant features that are indicative of failure. Second, since all common IL policies have an observation encoding module, directly using this pre-trained feature extractor generally allows us to RND-OE from a smaller rollout dataset, reducing data collection effort.

D Limitations

This section summarizes what we consider to be the main limitations of this work.

The failure prediction accuracy might still be insufficient for certain applications. Although FIPER outperforms state-of-the-art baselines across various tasks, its overall failure prediction accuracy for the best TWA is only 78%, highlighting the difficulty of early failure prediction in the absence of failure data. This accuracy may still be too low for certain applications, such as a robot operating in an assembly line, where true policy failures need to be recognized early and reliably, but false alarms are costly. Possible ways to improve the failure prediction accuracy could be increasing the number of calibration rollouts or using the training data of the policy.

The inclusion of historical data could be further improved. FIPER considers historical data solely through the aggregation of past uncertainty scores using a sliding window, without incorporating this historical context into the actual calculation of uncertainty scores. We hypothesize that the history of previous observations and action predictions could contain early warning signs indicative of policy failures, potentially enhancing the ability of FIPER to anticipate them.

RND-OE needs to be trained. We outline the benefits of using RND-OE compared to other observation-embedding-based methods in Appendix C.6. However, having to train an RND-OE model that is separate from the policy is still a limitation of our approach.

We only test RND-OE with image-based observation embeddings. In our experiments, we only use vision-based IL policies as they do not require specialized sensors. Therefore, although we expect FIPER to work well in principle for additional input modalities, such as language, touch or audio, this remains an untested hypothesis for the time being.

Time-varying thresholds can be restrictive. Time-varying thresholds can be disadvantageous for two reasons. First, they implicitly assume that the temporal sequence of events in a successful rollout is always similar. However, this may not be the case, for example, if the policy manages to grasp an object only in the second attempt, temporally shifting the trajectory from its expected pattern.

Therefore, if the training data contains multiple temporally distinct ways of completing the task, a constant threshold may be more suitable. Second, a time-varying threshold can only be calculated for timesteps that are represented sufficiently often in the calibration dataset. This can be a problem if rollouts have different lengths and may require padding.

Aleatoric uncertainty could impact ACE. If there is high variability (aleatoric uncertainty) in the demonstration data, this is generally reflected by the learned action distribution of the policy. In this case, the variability within generated action chunk batches can be high even for successful calibration rollouts, potentially leading to a larger threshold, which may result in a lower TPR. We think that disentangling aleatoric from epistemic uncertainty for generative policies remains an interesting avenue for future research.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: We benchmark our proposed method against four state-of-the-art baselines and evaluate performance across a diverse set of five environments, both in simulation and the real world, as summarized in Section 5. Moreover, we provide a detailed discussion of our claims based on the experimental results in Section 6, representing a clear and accurate reflection on the paper's contributions and scope as stated in the abstract and introduction.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: Section 7 discusses several limitations of our work, such as the need to collect successful rollouts, and we provide a more detailed discussion in Appendix D.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We provide assumptions and proofs for Proposition 1 in Appendix A.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We clearly and fully describe the details of our proposed method in Sections 4 and 5 and Appendix A. We also release the code and data to facilitate replication of our results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide links to the code, data, and instructions to reproduce our experimental results on the project website: tum-lsy.github.io/fiper_website.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We specify all relevant training and test details in Appendix B.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: In line with similar works, we report accuracy and average detection time as key metrics for evaluating the quality of failure prediction. We report the mean and standard deviation of the results across five random seeds. Moreover, we average our results across a set of quantile values to further strengthen the statistical significance.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We provide the relevant information on compute resources and execution times in Appendix B.6.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: The research conducted in this paper conforms with the NeurIPS Code of Ethics in every respect.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We discuss the potential of our method to improve safety and interpretability of robots operating alongside humans in the introduction and conclusion. We think that our work, which is aimed at predicting failures so that we can then avoid them, has no negative impacts on society.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our released algorithms and data do not have a high risk for misuse.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We cite all original papers for the models, methods, and data we use in Section 5. We respect all licenses and terms of use, which are mentioned in Appendix B.5.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Our provided code and datasets are accompanied by documentation.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Our paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: Our paper does not involve research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core methods development in this work do not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.