UNCERTAINTY QUANTIFICATION FOR FLOW-BASED GENERALIST ROBOT POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Generalist robot policies, such as vision-language-action models (VLAs) and world-action models (WAMs), combine powerful pretrained backbones with expressive generative action heads trained via flow matching on large-scale robotic datasets. Despite their strong empirical performance in robotic manipulation, these policies lack mechanisms to quantify confidence in their predictions and to detect when their actions may be unreliable. This presents a critical limitation for real-world deployment in non-stationary environments, where models inevitably encounter scenarios outside their pretraining distribution and may fail without warning. To address this, we derive an efficient method to quantify epistemic uncertainty in flow-matching models by leveraging velocity-field disagreement (VFD) across a small ensemble. We successfully use this uncertainty estimate for detecting failures during deployment and active fine-tuning of flow-based generalist policies. For the latter, we propose SAVE, a simple yet effective method for uncertainty-guided active multitask fine-tuning that reduces the number of costly expert demonstrations required to adapt generalist policies to new tasks. We conduct experiments in simulation and the real world, across VLAs and a WAM. VFD yields better-calibrated uncertainty estimates predictive of downstream performance and detects failures with 8 pp higher overall accuracy than existing methods. Across three real-world tasks, SAVE improves final average success from $39\,\%$ to $47\,\%$ with a fixed demonstration budget. Our results show that measuring epistemic uncertainty with VFD enhances both failure awareness and adaptation of generalist robot policies.

1 INTRODUCTION

The premise of generalist robot policies (Physical Intelligence et al., 2025; Kim et al., 2026) is to bring the large-scale pre-training paradigm that has been successful in deep learning in recent years (Dosovitskiy et al., 2021; Awais et al., 2025) into the physical world. Driven by the increasing availability of large-scale robotics datasets (O'Neill et al., 2024), generalist policies have demonstrated impressive performance in long-horizon tasks, exhibiting remarkable robustness and emergent zero-shot capabilities (Black et al., 2025). Architecturally, the field has largely converged on combining a pre-trained vision-language or video model backbone with an action-generation diffusion transformer trained via flow matching (FM) (Lipman et al., 2023) to generate continuous robot actions. Flow-based generalist policies provide stable training, relatively fast inference, and the expressivity needed to model the complex, highly multimodal distributions of real-world physical demonstrations.

While generalist policies are pre-trained on large-scale static datasets spanning a broad distribution of objects and tasks (O'Neill et al., 2024; AgiBot-World et al., 2025), deploying these policies in the real world inevitably exposes them to non-stationarity. Users repurpose robots for novel tasks, object appearances may change, and environments evolve. In such scenarios, pre-trained policies may perform well on some tasks but fail completely on others, without explicit knowledge of which tasks are fully understood from pre-training and which require additional data. The inability of these models to communicate *what they do not know* raises two key problems. First, generalist policies tend to confidently execute erratic actions in out-of-distribution scenarios (Gu et al., 2025) rather than abstaining or asking for help when uncertain (Ren et al., 2023), raising significant safety concerns (Li et al., 2026; Gu et al., 2025). Second, because these policies cannot inherently identify their knowledge gaps, adapting them to new domains currently requires collecting large numbers of human expert demonstrations (NVIDIA et al., 2025), which is prohibitively expensive.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

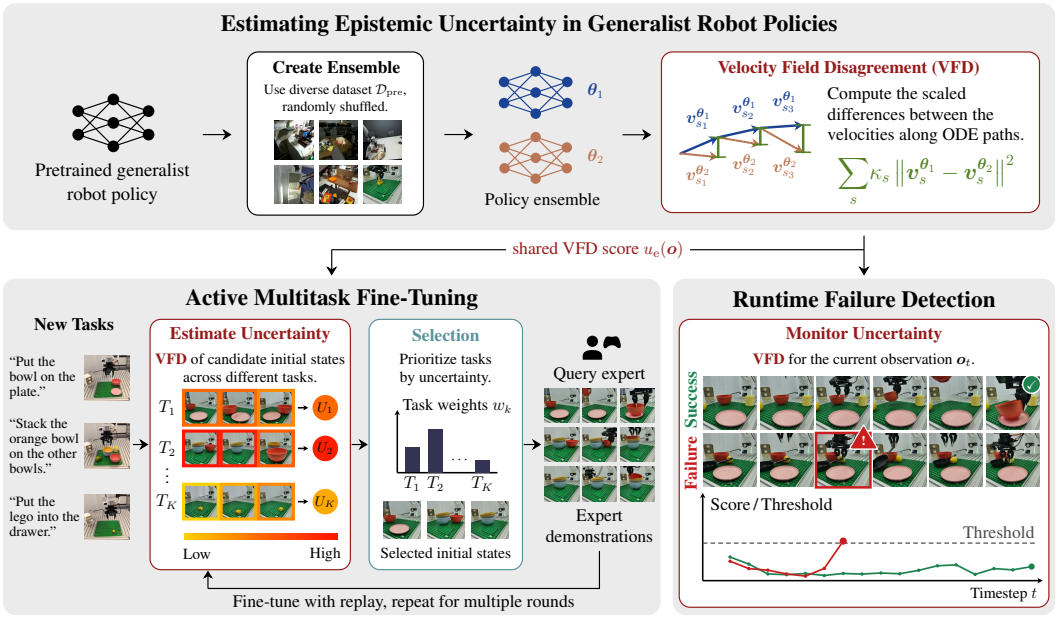

Figure 1: **Top:** We measure epistemic uncertainty in flow-based generalist policies by measuring scaled differences between ensembled velocity fields. **Bottom:** VFD enhances the adaptability and reliability of generalist policies in multiple ways. Left: We propose SAVE, an active fine-tuning method that allocates expert demonstration budget to tasks and initial states with the highest uncertainty. Right: VFD is an effective runtime monitor for detecting failures during deployment.

To address these problems, our work focuses on uncertainty quantification (UQ) for flow-based generalist policies. Leveraging the generative formulation of these models, we propose an effective method for estimating epistemic uncertainty by measuring divergence between ensembled velocity fields. Building on this estimator, we propose sample-efficient active fine-tuning via velocity-field epistemic uncertainty (SAVE) for flow-based generalist policies. Our main contributions are threefold:

- We propose velocity-field disagreement (VFD), a tractable epistemic uncertainty score for FM models, based on a velocity-field expression for the KL divergence between two FM models.
- We propose SAVE, a framework for uncertainty-guided active fine-tuning of flow-based generalist policies that uses VFD to prioritize tasks and initial states for collecting expert demonstrations.
- We empirically evaluate our methods in simulation and the real world, demonstrating that VFD predicts task performance more reliably and detects deployment failures more accurately than baselines, and that SAVE reduces the amount of expert data required for multitask adaptation.

2 Related Work

Generalist Robot Policies. Generalist policies have emerged as a dominant approach to learning complex manipulation tasks by fine-tuning pre-trained vision-language or video backbones on robotic data (Kim et al., 2024; Physical Intelligence et al., 2025; Kim et al., 2026; Yuan et al., 2026). Earlier vision-language-action models (VLAs) map visual and language observations to discrete or continuous action tokens via regression or classification heads (Kim et al., 2024; Zitkovich et al., 2023), but the state of the art has adopted FM as an expressive framework for modeling the highly multimodal action distributions in large-scale datasets (Physical Intelligence et al., 2025; Fang et al., 2026). A related concept, world-action models (WAMs) (Kim et al., 2026; Yuan et al., 2026), leverages the spatiotemporal priors of video models to enhance action generation, typically using latent flow matching. Despite their expressiveness, generalist policies do not natively quantify uncertainty and thus cannot identify situations in which they lack knowledge of which action to perform (Agia et al., 2024; Lee et al., 2025), preventing safe, autonomous real-world deployment. This motivates estimating *where* the model is most uncertain: before deployment, to prioritize situations for querying an expert, and during deployment, to detect failures.

Uncertainty Quantification for Generative Models. Uncertainty in deep learning is typically decomposed into epistemic (model) and aleatoric (data) uncertainty (Hüllermeier & Waegeman, 2021). While the latter is an irreducible property of the data generation process, the epistemic component can be reduced by collecting new and, importantly, the *right data*. Depending on the model class, UQ has been instantiated in many different ways. Test-time dropout (Loquercio et al., 2020) and inference on augmented inputs (Ayhan & Berens, 2018) may, for instance, be applied directly at test time and are mostly architecture-agnostic. More powerful approaches, such as deep ensembles (Lakshminarayanan et al., 2017) and Laplace approximations (Daxberger et al., 2021), require training multiple models or modifying the training procedure. With the rise of modern generative models (Ho et al., 2020; Lipman et al., 2023), which, despite their expressivity, often remain overconfident (Karczewski et al., 2025), several recent works have investigated UQ for this model class. Jazbec et al. (2025) apply a last-layer Laplace approximation to diffusion models and measure variability in a semantic representation space using a Gaussian approximation, which may, however, be unsuitable for multi-modal distributions. DECU (Berry et al., 2024) trains an ensemble of latent diffusion models and performs pairwise distance estimation using the denoising means. Other recent works rely on similarity metrics in language space (Franchi et al., 2025), leverage hypernetworks (Chan et al., 2024), or train explicit confidence predictors (Mei et al., 2025). UQ is also an active area of research in autoregressive models (Shorinwa et al., 2025), with domain-specific techniques spanning from differential entropy over vocabulary (Ling et al., 2024), to perplexity (Fadeeva et al., 2024), and Dirichlet evidence (Ma et al., 2025), but these methods primarily measure aleatoric uncertainty, i.e., data ambiguity.

Active Learning. A direct application of reliable uncertainty estimates is in active data selection, a well-established research field with deep roots in experimental design (Chaloner & Verdinelli, 1995) and active learning (Settles, 2009), including Bayesian disagreement-based acquisition for deep models (Gal et al., 2017; Kirsch et al., 2019). While a large body of literature has proposed domain-agnostic approaches (Ash et al., 2020; Holzmüller et al., 2023), several recent works have focused on active learning in MDPs, particularly for learning policies (Judah et al., 2012; He et al., 2025). Several existing approaches operate offline (Hejna et al., 2025; Agia et al., 2025) and filter a fixed dataset based on mutual information between states and actions or influence detection in a static setting. A long-running line of research (Cui et al., 2019; Karli et al., 2025) operates instead in an online setting, and tries to infer *when* a human labeling effort is required. Closer to our approach, Bagatella et al. (2025) also focus on an iterative setting but propose an algorithm that queries *entire* expert demonstrations to maximize information gain over multi-task expert trajectories. While this approach has strong regret guarantees, it does not scale well to generalist policies, unlike our proposed method.

3 Preliminaries

Bayesian Uncertainty Quantification. Given a dataset $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)})\}_{n=1}^{N}$ and a neural network parameterized by $\boldsymbol{\theta} \in \Theta$, trained to learn the relationship between $\boldsymbol{y} \in \mathcal{Y}$ and $\boldsymbol{x} \in \mathcal{X}$, Bayesian modeling considers the posterior predictive distribution

$$p(\boldsymbol{x} \mid \boldsymbol{y}, \mathcal{D}) = \int p(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathcal{D}) \mathrm{d}\boldsymbol{\theta}, \tag{1}$$

where $p(\boldsymbol{\theta} \mid \mathcal{D})$ is the posterior and generally intractable for deep neural networks (Hüllermeier & Waegeman, 2021). Therefore, (1) has to be approximated by a Monte-Carlo (MC) estimate $p(\boldsymbol{x} \mid \boldsymbol{y}, \mathcal{D}) \approx \frac{1}{M} \sum_{i=1}^{M} p(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{\theta}_i)$, where weight samples from the posterior $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_M \sim p(\boldsymbol{\theta} \mid \mathcal{D})$ can be obtained using approaches such as ensembling (Lakshminarayanan et al., 2017) or dropout (Gal & Ghahramani, 2016). The total uncertainty in the generation $\boldsymbol{x}$ for an input $\boldsymbol{y}$ is given by the entropy of the predictive distribution (1), $H\big[p(\boldsymbol{x} \mid \boldsymbol{y}, \mathcal{D})\big] = -\int p(\boldsymbol{x} \mid \boldsymbol{y}, \mathcal{D}) \log p(\boldsymbol{x} \mid \boldsymbol{y}, \mathcal{D}) \mathrm{d}\boldsymbol{x}$, which includes both irreducible aleatoric uncertainty and epistemic uncertainty, which can be reduced by collecting additional data. Subtracting the aleatoric component, given by the expected entropy, yields the epistemic uncertainty

$$I(\boldsymbol{x}, \boldsymbol{\theta} \mid \boldsymbol{y}, \mathcal{D}) = H\big[p(\boldsymbol{x} \mid \boldsymbol{y}, \mathcal{D})\big] - \mathbb{E}_{\boldsymbol{\theta}}\big[H\big[p(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{\theta})\big]\big] = \mathbb{E}_{\boldsymbol{\theta}}[D_{\mathrm{KL}}(p(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{\theta}) \,||\, p(\boldsymbol{x} \mid \boldsymbol{y}, \mathcal{D}))], \tag{2}$$

which equals the mutual information between $\boldsymbol{x}$ and $\boldsymbol{\theta}$ and can be viewed as the expected disagreement between a model learned from the data and the true posterior predictive (1) at a test input $\boldsymbol{y}$.

Flow Matching. Generative modeling aims to learn an unknown distribution q from a set of samples $\mathcal{D} \sim q$. We first consider unconditional generation, where $\mathcal{D} = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N}, \boldsymbol{x}^{(n)} \in \mathbb{R}^d$. FM models a time-dependent velocity field $\boldsymbol{u}_s : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ generating a probability path $(p_s)_{0 \leq s \leq 1}$

that transforms a source distribution p_0 (e.g., a Gaussian) to the data distribution $p_1 = q$. This means the flow $\boldsymbol{\phi}_s : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ solving the ODE $\frac{\mathrm{d}}{\mathrm{d}s}\boldsymbol{\phi}_s(\boldsymbol{x}) = \boldsymbol{u}_s(\boldsymbol{\phi}_s(\boldsymbol{x}))$, $\boldsymbol{\phi}_0(\boldsymbol{x}) = \boldsymbol{x}$ satisfies $\boldsymbol{x}_s = \boldsymbol{\phi}_s(\boldsymbol{x}_0) \sim p_s$ for $\boldsymbol{x}_0 \sim p_0$. As $\boldsymbol{u}_s(\boldsymbol{x})$ is unknown, conditional FM (Lipman et al., 2023) with $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)})\}_{n=1}^N$ constructs p_s as the marginal of simpler conditional paths $p_s(\boldsymbol{x} \mid \boldsymbol{x}_1)$, where $\boldsymbol{x}_1 \sim q$. A common choice is optimal transport (OT) Gaussian conditional paths $p_s(\boldsymbol{x} \mid \boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x} \mid s\boldsymbol{x}_1, (1-s)^2\boldsymbol{I})$, for which the conditional velocity field $\boldsymbol{u}_s(\boldsymbol{x} \mid \boldsymbol{x}_1)$ has an analytical expression. A neural network $\boldsymbol{v}_s^{\boldsymbol{\theta}}$ is then trained to regress onto this target vector using the loss

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{s\sim\mathrm{Unif}([0,1]),(\boldsymbol{x}_1,\boldsymbol{y})\sim\mathrm{Unif}(\mathcal{D}),\boldsymbol{x}_s\sim p_s(\cdot\mid\boldsymbol{x}_1)}\left[\left\|\boldsymbol{v}_s^{\boldsymbol{\theta}}(\boldsymbol{x}_s,\boldsymbol{y}) - \boldsymbol{u}_s(\boldsymbol{x}_s \mid \boldsymbol{x}_1)\right\|_2^2\right]. \tag{3}$$

New samples $\boldsymbol{x} \sim p_1(\cdot \mid \boldsymbol{y})$ from the learned distribution can be generated, for example, through Euler integration starting from $\boldsymbol{x}_0 \sim p_0$ with step size $\delta s = 1/N_\mathrm{s}$ and $\boldsymbol{x}_{s+\delta s} = \boldsymbol{x}_s + \boldsymbol{v}_s^{\boldsymbol{\theta}}(\boldsymbol{x}_s, \boldsymbol{y})\delta s$.

Flow-Based Generalist Policies. We consider robot policies $\pi_{\boldsymbol{\theta}} : \mathcal{O} \to \Delta(\mathcal{A})$ that map multimodal observations $\boldsymbol{o}_t \in \mathcal{O}$ to distributions over actions $\boldsymbol{a}_t \in \mathcal{A}$, where an observation $\boldsymbol{o}_t = (\boldsymbol{s}_t, \boldsymbol{I}_t, \boldsymbol{l})$ typically contains the proprioceptive state $\boldsymbol{s}_t$, camera images $\boldsymbol{I}_t$ and language instruction $\boldsymbol{l}$. Flow-based policies are trained through behavior cloning to match the distribution underlying large-scale datasets $\mathcal{D} = \{(\boldsymbol{o}^{(n)}, \boldsymbol{a}^{(n)})\}_{n=1}^N \sim q$, using the loss (3) with $\boldsymbol{y} = \boldsymbol{o}_t$. The choice of output differs between model classes: VLAs usually generate action chunks $\boldsymbol{x} = \boldsymbol{A}_t = (\boldsymbol{a}_t, \boldsymbol{a}_{t+1}, \ldots, \boldsymbol{a}_{t+H-1})$, whereas WAMs are trained to also predict future observations, i.e., $\boldsymbol{x} = (\boldsymbol{A}_t, \boldsymbol{O}_t)$.

4 Estimating Epistemic Uncertainty of Flow-Matching Models

Training an FM model yields an estimate $\boldsymbol{\theta}$ of the model parameters given the data $\mathcal{D}$. For a new conditioning input $\boldsymbol{y}$, how can we quantify whether the model exhibits high epistemic uncertainty that could be reduced by collecting more data? A simple estimate of total predictive uncertainty is the conditional entropy $H[p(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{\theta})]$. However, as this quantity conflates epistemic and aleatoric uncertainty, we need to isolate the epistemic component (2). To this end, we denote $p^{\boldsymbol{\theta}_i}(\boldsymbol{x} \mid \boldsymbol{y}) = p(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{\theta}_i)$ and consider the MC approximation of (2), that is,

$$I(\boldsymbol{x}, \boldsymbol{\theta} \mid \boldsymbol{y}, \mathcal{D}) \approx \frac{1}{M}\sum_{i=1}^M D_\mathrm{KL}\left(p^{\boldsymbol{\theta}_i}(\boldsymbol{x} \mid \boldsymbol{y}) \,||\, p(\boldsymbol{x} \mid \boldsymbol{y}, \mathcal{D})\right) \tag{4a}$$

$$\approx \frac{1}{M}\sum_{i=1}^M D_\mathrm{KL}\left(p^{\boldsymbol{\theta}_i}(\boldsymbol{x} \mid \boldsymbol{y}) \,||\, \frac{1}{M}\sum_{j=1}^M p^{\boldsymbol{\theta}_j}(\boldsymbol{x} \mid \boldsymbol{y})\right) \tag{4b}$$

$$\leq \frac{1}{M^2}\sum_{i,j=1}^M D_\mathrm{KL}\left(p^{\boldsymbol{\theta}_i}(\boldsymbol{x} \mid \boldsymbol{y}) \,||\, p^{\boldsymbol{\theta}_j}(\boldsymbol{x} \mid \boldsymbol{y})\right), \tag{4c}$$

where $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M \sim p(\boldsymbol{\theta} \mid \mathcal{D})$, and (4c) follows by applying Jensen's inequality to each term in the outer sum, using the convexity of the KL divergence in its second argument. Hence, the average pairwise KL divergence between FM models trained on the same data $\mathcal{D}$, evaluated for a conditioning input $\boldsymbol{y}$, represents an approximate upper bound of the epistemic uncertainty.

In principle, the KL divergence between two FM models can be estimated by computing the likelihoods of individual samples, which entails solving an augmented ODE that involves the divergence of the learned velocity field. However, computing this divergence is prohibitively expensive in high dimensions. To make computation tractable, we first derive a relationship between pairwise KL divergence and the marginal velocity fields. This identity motivates a practical disagreement estimator based on learned velocity fields. For the proof, we refer to Appendix A.2.

Theorem 4.1. *Consider two distributions $p^1(\boldsymbol{x} \mid \boldsymbol{y})$, $p^2(\boldsymbol{x} \mid \boldsymbol{y})$ induced by velocity fields $\boldsymbol{u}_s^1(\boldsymbol{x}, \boldsymbol{y})$, $\boldsymbol{u}_s^2(\boldsymbol{x}, \boldsymbol{y})$ with OT Gaussian conditional probability paths satisfying regularity assumptions (cf. Appendix A.2). Then, the KL divergence between the two distributions is*

$$D_\mathrm{KL}\big(p^1(\boldsymbol{x} \mid \boldsymbol{y}) \,||\, p^2(\boldsymbol{x} \mid \boldsymbol{y})\big) = \int_0^1 \frac{s}{1-s}\mathbb{E}_{\boldsymbol{x}_s\sim p_s^1(\boldsymbol{x}\mid\boldsymbol{y})}\left[\left\|\boldsymbol{u}_s^1(\boldsymbol{x}_s,\boldsymbol{y}) - \boldsymbol{u}_s^2(\boldsymbol{x}_s,\boldsymbol{y})\right\|_2^2\right]\mathrm{d}s. \tag{5}$$

For trained FM models, we can apply this disagreement functional to the learned velocity fields. In this way, we can approximate the KL divergence terms in (4c) by sampling $\boldsymbol{x}_0 \sim p_0$, performing Euler integration using the learned velocity field $\boldsymbol{v}_s^{\boldsymbol{\theta}_1}$ and calculating the *velocity-field disagreement (VFD)*

at intermediate ODE states $\boldsymbol{x}_0, \boldsymbol{x}_{\delta s}, \dots$, weighted by $\kappa_s = \frac{s}{1-s}$, as

$$D_{\mathrm{KL}}\big(p^{\boldsymbol{\theta}_1}(\boldsymbol{x} \,|\, \boldsymbol{y}) \,||\, p^{\boldsymbol{\theta}_2}(\boldsymbol{x} \,|\, \boldsymbol{y})\big) \approx \frac{1}{N_s} \mathbb{E}_{\boldsymbol{x}_0 \sim p_0}\left[\sum_{\ell=0}^{N_s-1} \kappa_{s_\ell} \left\| \boldsymbol{v}_{s_\ell}^{\boldsymbol{\theta}_1}(\boldsymbol{x}_{s_\ell}, \boldsymbol{y}) - \boldsymbol{v}_{s_\ell}^{\boldsymbol{\theta}_2}(\boldsymbol{x}_{s_\ell}, \boldsymbol{y}) \right\|_2^2\right], \quad (6)$$

where $s_\ell = \ell \delta s$. Finally, by denoting $\mathcal{V} = (\boldsymbol{v}_s^{\boldsymbol{\theta}_1}, \dots, \boldsymbol{v}_s^{\boldsymbol{\theta}_M})$, plugging (6) into (4c) and averaging over non-identical pairs, we can define the tractable **VFD score for epistemic uncertainty estimation**

$$u_{\mathrm{e}}(\boldsymbol{y}; \mathcal{V}) = \frac{1}{M(M-1)N_s} \mathbb{E}_{\boldsymbol{x}_0 \sim p_0}\left[\sum_{i,j=1; i \neq j}^{M} \sum_{\ell=0}^{N_s-1} \kappa_{s_\ell} \left\| \boldsymbol{v}_{s_\ell}^{\boldsymbol{\theta}_i}(\boldsymbol{x}_{s_\ell}^{(i)}, \boldsymbol{y}) - \boldsymbol{v}_{s_\ell}^{\boldsymbol{\theta}_j}(\boldsymbol{x}_{s_\ell}^{(i)}, \boldsymbol{y}) \right\|_2^2\right], \quad (7)$$

where $\boldsymbol{x}_{s_\ell + \delta s}^{(i)} = \boldsymbol{x}_{s_\ell}^{(i)} + \boldsymbol{v}_{s_\ell}^{\boldsymbol{\theta}_i}\big(\boldsymbol{x}_{s_\ell}^{(i)}, \boldsymbol{y}\big)\delta s$, $\boldsymbol{x}_0^{(i)} = \boldsymbol{x}_0$. In practice, we approximate the expectation in (7) by forward-integrating a small batch of actions, leveraging GPU parallelization.

This uncertainty estimate can be used in several ways to make generalist robot policies more reliable and adaptable, with *failure detection* and *active learning* being of particular interest. The former deploys uncertainty in a relatively straightforward way, flagging a failure when uncertainty is high (Xu et al., 2025; Römer et al., 2025). Active fine-tuning involves more complex design choices, and we present an effective algorithm that utilizes VFD in the following section.

5 Uncertainty-Guided Active Multitask Fine-Tuning

We consider the following problem: Given a flow-based generalist policy and a set of tasks, how can we improve overall performance across tasks with minimum data collection effort? More formally, we consider a set of K tasks $\mathcal{T} = \{T_k\}_{k=1}^K$, each of which can be described by an MDP $\mathcal{M}_k = (\mathcal{O}, \mathcal{A}, P_k, R_k, \rho_k, \gamma)$, where $P_k : \mathcal{O} \times \mathcal{A} \to \Delta(\mathcal{O})$, $R_k : \mathcal{O} \to \mathbb{R}$ and $\rho_k \in \Delta(\mathcal{O})$ are task-specific dynamics, rewards and initial observation distributions, and $\gamma \in (0, 1)$ is a discount factor. Given a policy $\pi : \mathcal{O} \to \Delta(\mathcal{A})$, its single-task performance is simply $J_k(\pi) = \mathbb{E}_{\pi, P_k, \rho_k} \sum_{t=0}^{\infty} \gamma^t R_k(\boldsymbol{o}_t)$, and its multitask performance is $J(\pi) = \frac{1}{K}\sum_{k=1}^K J_k(\pi)$. Our goal is to fine-tune a generalist policy $\pi_{\boldsymbol{\theta}}$ (pre-trained on large-scale, diverse data) to maximize its average multitask performance. For each task $T_k \in \mathcal{T}$, we assume the availability of a pool of L candidate initial observations $\mathcal{O}_k = \{\boldsymbol{o}_{kl}\}_{l=1}^L \sim \rho_k$ which, in practice, could include scenarios the policy might encounter at the start of a rollout. Without any environment interaction, a subset of observations $\hat{\mathcal{O}} \subset \cup_{k=1}^K \mathcal{O}_k$ needs to be selected, and an expert demonstrator $\pi^* \approx \mathrm{argmax}_\pi J(\pi)$ will return a demonstration starting from each of the selected observations: $\tau^* = (\boldsymbol{o}_0, \boldsymbol{a}_0, \dots)$ with $\boldsymbol{a}_t \sim \pi^*(\boldsymbol{o}_t)$ and $\boldsymbol{o}_{t+1} \sim P_k(\boldsymbol{o}_t, \boldsymbol{a}_t)$ for each $\boldsymbol{o}_0 \in \hat{\mathcal{O}}$.

Active data acquisition requires deciding which demonstrations are *most valuable to collect*. We address this problem by using VFD to measure epistemic uncertainty for candidate initial configurations. Higher values indicate the policy is more uncertain about how to act in a given scene, suggesting limited exposure during pre-training and a potential benefit from additional expert demonstrations.

Generalist Policy Ensemble. The high cost of pre-training foundation models makes training multiple generalist policies from scratch prohibitively expensive. Instead, we take a single pre-trained base policy π_{b} and fine-tune it M times on a diverse dataset $\mathcal{D}_{\mathrm{pre}}$, randomly shuffled each time. In this way, we obtain a policy ensemble $\Pi_0 = \big(\pi_0^{(1)}, \dots, \pi_0^{(M)}\big)$ for uncertainty estimation.

Uncertainty-Guided Data Acquisition. We train the ensemble over R rounds of active fine-tuning. At each round r, we evaluate the current ensemble Π_r on all candidate initial observations and compute, for all tasks $k \in \{1, \dots, K\}$ and candidate initial observations $\boldsymbol{o}_{kl} \in \mathcal{O}_k$, the VFD epistemic uncertainty $u_{kl}^{(r)} = u_{\mathrm{e}}(\boldsymbol{o}_{kl}; \Pi_r)$, as defined in (7). A simple strategy would then be to always select the most uncertain candidate observation. However, this approach can overemphasize a small set of outlier scenarios, leading to poor task coverage and small multitask improvement. Thus, we first aggregate uncertainty at the task level using the mean score

$$U_k^{(r)} = \frac{1}{L}\big(u_{k1}^{(r)} + \ldots + u_{kL}^{(r)}\big), \qquad k = 1, \dots, K. \quad (8)$$

It may then be tempting to directly request a batch of demonstrations from the most uncertain task. However, this is both theoretically flawed, as mutual information would need to be updated

as soon as a single new demonstration is collected, and practically suboptimal, as an excessive number of demonstrations could be collected for a simple task. While posterior uncertainty can be computed under strong assumptions about the model class (e.g., analytically, without even observing $\boldsymbol{x}$ in the case of Gaussian processes (Bagatella et al., 2025)), its estimation with neural networks remains, in general, intractable. As collecting demonstrations in batches remains practically necessary for computational reasons, we instead introduce diversity by defining a categorical sampling distribution $\mathrm{Cat}(W^{(r)})$ with a temperature parameter $\tau \geq 0$ and weights $W^{(r)} = (w_1^{(r)}, \dots, w_K^{(r)})$ over tasks, where $w_k^{(r)} = \left(U_k^{(r)}\right)^\tau / \sum_{k'=1}^{K} \left(U_{k'}^{(r)}\right)^\tau$. Larger values of τ concentrate probability mass on the most uncertain tasks, and $\tau = 0$ corresponds to uniform sampling. We perform n_e expert queries per round and, for each, first sample a task index $k \sim \mathrm{Cat}(W^{(r)})$, effectively prioritizing tasks with high estimated uncertainty while retaining exploration over all tasks. We then select the most uncertain initial observation within the sampled task,

$$\boldsymbol{o}_k^* = \arg\max_{\boldsymbol{o}_{kl} \in \mathcal{O}_k} u_{kl}^{(r)}, \tag{9}$$

and request an expert demonstration starting from $\boldsymbol{o}_0 = \boldsymbol{o}_k^*$; $\tau^* = (\boldsymbol{o}_0, \boldsymbol{a}_0, \dots)$. Repeating this procedure n_e times yields a batch of newly collected demonstrations $\mathcal{D}_\mathrm{new}^{(r)}$.

Iterative Fine-Tuning. Fine-tuning a generalist policy solely on new demonstrations can lead to forgetting of previously acquired capabilities (Du et al., 2026; Römer et al., 2026). To avoid this issue, we fine-tune the policy ensemble on a mixture of pre-training and queried data, $\mathcal{D}_\mathrm{train}^{(r)} \sim \lambda\ \mathrm{Unif}(\mathcal{D}_\mathrm{pre}) + (1-\lambda)\ \mathrm{Unif}(\mathcal{D}_\mathrm{new}^{(\leq r)})$, where $\mathcal{D}_\mathrm{new}^{(\leq r)}$ contains all demonstrations collected up to round r, and $\lambda \in [0, 1]$ is the replay ratio. By determining which tasks currently need data most and then selecting the highest-uncertainty initial observations *within* those tasks, SAVE balances exploitation of uncertain scenes with exploration across tasks for sample-efficient multitask adaptation.

6 Experiments

Our experiments in simulation and the real world primarily aim at answering three research questions:

- **Q1:** How well-calibrated is VFD, i.e., to what extent do its values reflect the epistemic uncertainty and task-completion capabilities of flow-based generalist policies?
- **Q2:** Can VFD identify where additional data should be collected for active fine-tuning?
- **Q3:** How informative are VFD uncertainty estimates for detecting failures during deployment?

6.1 Experimental Setup

Simulation Environments. We conduct our experiments on three benchmarks. `Push-T` (Chi et al., 2023) simulates a 2D agent pushing a T-shaped object into a target area. `LIBERO` (Liu et al., 2023) simulates a Franka manipulator across various household environments, and we focus on its most challenging suite, `LIBERO-Long`, which includes 10 long-horizon tasks. To evaluate failure detection capabilities, we use `LIBERO-Plus` (Fei et al., 2026), which introduces seven perturbation dimensions, including object layouts and camera viewpoints, yielding a wide range of failure modes.

Real-World Tasks. We design three tasks for an FR3 robot: 1) Placing a bowl on a plate (`Place`), 2) stacking a bowl on top of three other bowls (`Stack`), and 3) placing a Lego brick into a drawer and closing it (`Drawer`). The tasks consist of two, two, and three phases, respectively (e.g., lifting the bowl and placing it down for `Place`), and we assign partial credit for each successfully completed subtask.

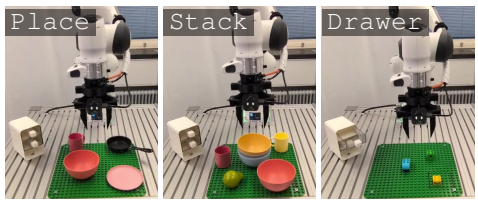

Figure 2: **Real-world tasks.**

Policies and Implementation Details. We use three generalist policies: SmolVLA (550M, Shukor et al., 2025) attaches an action expert to a SmolVLM-2 backbone; X-VLA (900M, Zheng et al., 2026) uses soft-prompting for cross-embodiment learning; and FastWAM (6B, Yuan et al., 2026), built on top of a Wan2.2-5B video DiT backbone, is trained with a joint video modeling and action generation objective. The three policies are conditioned on a language command, two camera images, and the proprioceptive state. In `Push-T`, we use a simple FM-Policy trained from scratch. All policies have

Table 1: **Calibration analysis.** Negative Spearman's rank correlation (↑) between uncertainty estimates and per-task success rates in `LIBERO-Long`, averaged across iterative fine-tuning rounds.

Policy	Action-L2	ACE	DECU	GU	Entropy	Perplexity	**VFD (ours)**
SmolVLA	$0.50^{\pm 0.13}$	$0.31^{\pm 0.12}$	$0.31^{\pm 0.13}$	$0.62^{\pm 0.00}$	$0.10^{\pm 0.12}$	$-0.04^{\pm 0.09}$	$\mathbf{0.71}^{\pm 0.03}$
X-VLA	$0.33^{\pm 0.24}$	$0.14^{\pm 0.05}$	$0.06^{\pm 0.04}$	$0.48^{\pm 0.06}$	N/A	N/A	$\mathbf{0.53}^{\pm 0.02}$
FastWAM	$-0.12^{\pm 0.06}$	$-0.29^{\pm 0.07}$	$0.03^{\pm 0.07}$	$0.27^{\pm 0.04}$	N/A	N/A	$\mathbf{0.44}^{\pm 0.17}$

an ODE step size of $\delta s = 0.1$, and we use an ensemble of size $M = 2$. We repeat all experiments with three seeds (ten for `Push-T`) and report the mean and standard deviation across seeds.

Iterative Fine-Tuning. We investigate uncertainty calibration and active learning using the iterative fine-tuning described in Section 5. For `Push-T`, we train an ensemble on 50 out of the 206 available demonstrations. For `LIBERO`, $\mathcal{D}_{\text{pre}}$ contains all demonstrations from the 30 tasks in the `LIBERO-Goal`, `-Spatial`, and `-Object` suites, and we also include three tasks from `LIBERO-Long` to instill different levels of initial task competence. The base ensemble used in the real-world active fine-tuning experiments is trained on ≈2,500 DROID episodes (Khazatsky et al., 2024). We perform $R = 15$ (`Push-T`: 30) rounds of iterative active fine-tuning and query $n_e = 5$ (real world: 10) expert demonstrations per round, using a replay ratio of $\lambda = 0.5$ (`Push-T`: 0.3).

6.2 Calibration (Q1): How Well Does VFD Measure Epistemic Uncertainty?

Setup. We measure calibration using Spearman's rank correlation coefficient between the per-task uncertainties (8) and success rates, with a value of -1 indicating that the success rate decreases monotonically with increasing uncertainty. To avoid bias from data acquisition, we select new demonstrations uniformly at random each round. We consider six UQ baselines. **Action-L2** computes the pairwise L_2 distance between actions generated independently from the ensemble members. **ACE** (Römer et al., 2025) computes the conditional entropy in the action chunk distribution. **DECU** (Berry et al., 2024) performs pairwise distance estimation at a "branching" ODE timestep for denoising diffusion models, and we adapt it for FM. **GU** (Jazbec et al., 2025) generates samples from all ensemble members starting from the same initial noise to approximate the posterior predictive as a Gaussian, and calculates its entropy. **Entropy** (Malinin & Gales, 2021) and **Perplexity** (Fadeeva et al., 2024) compute average per-token entropy and the exponentiated negative mean of the language tokens' log-probabilities, respectively, and they are only directly applicable to SmolVLA.

VFD excels at estimating epistemic uncertainty. The calibration results, averaged over all iterative fine-tuning rounds, are summarized in Table 1. VFD achieves negative Spearman correlations between 0.05 and 0.17 higher than the best-performing baseline, GU. The results indicate that VFD uncertainty of a generalist policy on the initial frames is a strong indicator of how difficult the respective task is for that policy, and we examine the implications for active fine-tuning in Section 6.3.

VFD is sensitive to different input modalities. As generalist policies often struggle with language conditioning, we investigate whether VFD can capture epistemic uncertainty induced solely by prompt variations. We generate five increasingly semantically perturbed versions of each language prompt in `LIBERO-Long` (P1 to P5), detailed in Table 5, and evaluate the success rate and uncertainty for each, using the final SmolVLA policy π_{15} after iterative random data acquisition. The calibration curves are shown in Figure 3, normalized for comparability. VFD yields the strongest monotonic trend, indicating that it can measure epistemic uncertainty induced through different input modalities.

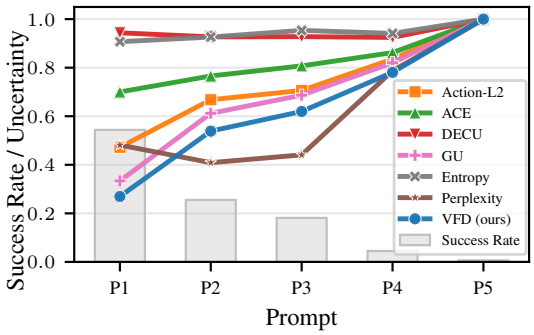

Figure 3: **Calibration w.r.t. the language prompt.** VFD is well-calibrated when injecting uncertainty only through this modality.

6.3 Active Fine-Tuning (Q2): How Well Can VFD Select Which Data to Collect?

Setup. We now investigate whether the superior calibration of VFD transfers to improved data efficiency in active multitask fine-tuning with SAVE. As metrics, we measure the area under the

Table 2: **Active fine-tuning in simulation.** Area under the learning curve (AULC, ↑) and average success rate across the last three rounds (Last-3 SR, ↑) for different data acquisition strategies.

Benchmark	Push-T		LIBERO-Long					
Policy	FM-Policy		SmolVLA		X-VLA		FastWAM	
Metric	AULC	Last-3 SR	AULC	Last-3 SR	AULC	Last-3 SR	AULC	Last-3 SR
Random	$38.0^{\pm 3.2}$	$46.3^{\pm 3.1}$	$44.6^{\pm 2.2}$	$52.6^{\pm 3.3}$	$37.7^{\pm 1.2}$	$45.1^{\pm 0.3}$	$54.2^{\pm 2.5}$	$64.8^{\pm 6.6}$
Diversity	$40.5^{\pm 1.9}$	$51.2^{\pm 3.2}$	$45.5^{\pm 1.8}$	$53.1^{\pm 2.0}$	$40.2^{\pm 1.9}$	$49.0^{\pm 3.3}$	$53.9^{\pm 0.8}$	$63.9^{\pm 0.3}$
AMF	N/A	N/A	$45.6^{\pm 3.3}$	$59.4^{\pm 3.3}$	$39.5^{\pm 1.9}$	$51.0^{\pm 1.3}$	$55.6^{\pm 1.4}$	$69.4^{\pm 1.3}$
SAVE w/ GU	$44.7^{\pm 2.9}$	$54.3^{\pm 3.3}$	$48.6^{\pm 5.0}$	$61.0^{\pm 2.9}$	$40.7^{\pm 3.5}$	$49.2^{\pm 1.8}$	$59.0^{\pm 4.4}$	$70.6^{\pm 6.1}$
SAVE w/ Action-L2	$39.8^{\pm 3.4}$	$45.0^{\pm 2.6}$	$47.2^{\pm 5.1}$	$55.5^{\pm 6.8}$	$39.2^{\pm 0.9}$	$50.9^{\pm 2.2}$	$57.0^{\pm 3.3}$	$71.6^{\pm 5.2}$
SAVE w/ VFD	$\mathbf{46.6}^{\pm 2.5}$	$\mathbf{59.3}^{\pm 3.1}$	$\mathbf{52.3}^{\pm 2.2}$	$\mathbf{66.1}^{\pm 2.8}$	$\mathbf{42.5}^{\pm 1.0}$	$\mathbf{52.9}^{\pm 2.9}$	$\mathbf{61.1}^{\pm 3.5}$	$\mathbf{74.8}^{\pm 3.3}$

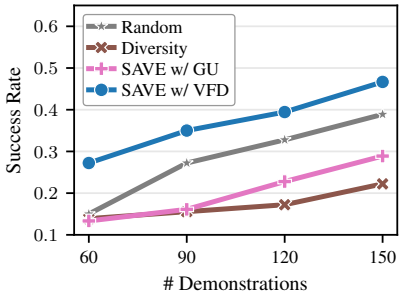

Figure 4: **Active fine-tuning in the real world.** For fixed demonstration budgets, SAVE w/ VFD improves multitask performance the most.

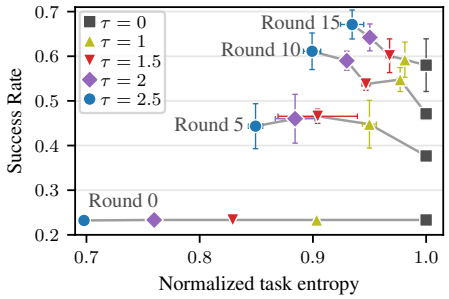

Figure 5: **Exploration vs. exploitation.** SAVE w/ VFD prioritizes tasks with higher uncertainty, trading off diversity for multitask improvement.

learning curve (AULC) and the average success rate over the last three rounds (Last-3 SR). We consider five baselines. **Random** selects demonstrations uniformly at random, **Diversity** (Sener & Savarese, 2018) selects the most visually diverse initial observations using a k-center greedy algorithm, and **AMF** (Bagatella et al., 2025) performs active multi-task fine-tuning by utilizing posterior uncertainty estimates based on last-layer gradient embeddings. In addition, we design two baselines based on SAVE by replacing VFD with GU and Action-L2, which perform second and third in Table 1. We optimize hyperparameters by sweeping over temperature values $\tau \in [0, 2.5]$ for SAVE and over different noise levels for AMF. Further implementation details are provided in Appendix B.

SAVE with VFD uncertainty guidance learns new tasks faster. Table 2 summarizes the simulation results. SAVE w/ VFD consistently yields the highest AULC and Last-3 SR across all policies and environments. It achieves a $+1.9$ to $+5.1$ pp gain in Last-3 SR over the strongest baseline in each setting and a relative increase of 3.6 to 7.6 % in AULC. Among the uncertainty scores evaluated for SAVE, the overall relative ordering is VFD>GU>Action-L2. This mirrors the calibration results in Table 1, highlighting the importance of well-calibrated uncertainty estimates for guiding active fine-tuning. In the real world, we consider the strongest uncertainty-based baseline from simulation, SAVE w/ GU, as well as Random and Diversity. As shown in Figure 4, SAVE w/ VFD consistently achieves success rates that are only (if at all) reached by the other methods after at least 30 additional demonstrations. At 150 demonstrations, SAVE w/ VFD yields an average task success of 47 %, between $+8$ and $+25$ pp higher than the baselines. The higher multitask improvements show that VFD can allocate fine-tuning budget for generalist policies more effectively by quantifying epistemic uncertainty.

Uncertainty guidance benefits task and initial state selection. We isolate the effects of the two uncertainty-guidance mechanisms of SAVE on LIBERO-Long with SmolVLA. Compared to selecting tasks uniformly at random ($\tau = 0$), SAVE w/ VFD increases Last-3 SR by 11.1 pp, and by 3.4 pp compared to selecting random starting configurations for the sampled tasks. Figure 5 visualizes one reason for these performance gains. While pre-trained generalist policies naturally exhibit varying levels of uncertainty across tasks, SAVE w/ VFD allocates demonstration budget to the tasks that require it most, trading off diversity in selected tasks for faster multitask improvement.

Table 3: **Failure detection**. VFD achieves the highest balanced accuracy (Acc.) and the best overall trade-off between accuracy and early detection, as measured by timestep-wise accuracy (TWA).

	Push-T		LIBERO-Plus		Real World		Average	
Method	Acc. ↑	TWA ↑	Acc. ↑	TWA ↑	Acc. ↑	TWA ↑	Acc. ↑	TWA ↑
STAC	$0.55^{\pm 0.00}$	$0.51^{\pm 0.00}$	$0.70^{\pm 0.03}$	$0.59^{\pm 0.02}$	$0.43^{\pm 0.02}$	$0.42^{\pm 0.02}$	$0.56^{\pm 0.02}$	$0.51^{\pm 0.01}$
logpZO	$0.53^{\pm 0.01}$	$0.51^{\pm 0.02}$	$0.68^{\pm 0.02}$	$0.62^{\pm 0.01}$	$0.70^{\pm 0.03}$	$\mathbf{0.64}^{\pm 0.03}$	$0.64^{\pm 0.02}$	$0.59^{\pm 0.02}$
RND-OE	$0.50^{\pm 0.02}$	$0.50^{\pm 0.02}$	$0.67^{\pm 0.01}$	$0.62^{\pm 0.01}$	$0.69^{\pm 0.04}$	$0.61^{\pm 0.02}$	$0.62^{\pm 0.00}$	$0.58^{\pm 0.00}$
ACE	$0.55^{\pm 0.06}$	$0.51^{\pm 0.03}$	$0.75^{\pm 0.02}$	$0.59^{\pm 0.02}$	$0.54^{\pm 0.01}$	$0.50^{\pm 0.00}$	$0.61^{\pm 0.02}$	$0.54^{\pm 0.02}$
VFD	$\mathbf{0.64}^{\pm 0.05}$	$\mathbf{0.56}^{\pm 0.05}$	$\mathbf{0.79}^{\pm 0.00}$	$\mathbf{0.65}^{\pm 0.00}$	$\mathbf{0.73}^{\pm 0.01}$	$0.58^{\pm 0.00}$	$\mathbf{0.72}^{\pm 0.02}$	$\mathbf{0.60}^{\pm 0.02}$

6.4 Failure Detection (Q3): Can VFD be Used for Runtime Monitoring?

Setup. We investigate whether high VFD uncertainty *during deployment* indicates imminent task failure. For the real-world tasks, we create OOD scenarios by adding novel background objects (e.g., pieces of fruit or a pan). We roll out the policy and calculate VFD scores at each action-generation timestep. Following Agia et al. (2024), we calibrate thresholds from 20 (real world: 10) successful rollouts per task, i.e., without failure data, using conformal prediction (Angelopoulos & Bates, 2023). We report balanced accuracy and time-step-wise accuracy (TWA), which rewards correctly detected failures more when an alarm is raised earlier. As baselines, we consider detecting OOD observations by applying random network distillation (**RND-OE**, Römer et al., 2025) or training normalizing flows (**logpZO**, Xu et al., 2025) on the policies' training data, measuring temporal consistency in the action distributions (**STAC**, Agia et al., 2024), and computing the conditional action-chunk entropy (**ACE**, Römer et al., 2025). More details are provided in Appendix B.5.

VFD detects failures more accurately. As reported in Table 3, VFD achieves a balanced accuracy of 72 % across all environments, which is 8 pp higher than the strongest baseline, logpZO. While the observation-based OOD detectors logpZO and RND-OE are trained on the same data as the policy ensemble, VFD is much more robust in distinguishing benign OOD situations where the policy succeeds from actual failures, achieving a TNR of 81 % compared to 58 % and 54 %, respectively. The high accuracy of VFD on LIBERO-Plus and in the real world demonstrates its potential for runtime monitoring of capable generalist policies that can handle some, but not all, novel scenarios.

Reducing computational cost. Table 4 reports the effect of reducing the computational effort of VFD for the real-world rollouts. Perturbing the policy's weights with layer-normalized Gaussian noise instead of training a second model reduces balanced accuracy to 68 %, which is, however, still 14 to 25 pp above the two training-free baselines, STAC and ACE.

Table 4: **Computational cost.** Speeding up the computation of VFD only slightly reduces failure detection accuracy.

Posterior approximation	# Flow times, batch size	Balanced accuracy	Inference time	vs. base policy
Ensemble (4)	4, 8	$0.78^{\pm 0.00}$	+251 ms	2.48×
Ensemble (2)	4, 8	$0.73^{\pm 0.01}$	+89 ms	1.53×
Ensemble (2)	1, 8	$0.72^{\pm 0.01}$	+49 ms	1.29×
Ensemble (2)	1, 1	$0.70^{\pm 0.02}$	+32 ms	1.19×
Weight perturb.	1, 1	$0.68^{\pm 0.01}$	+29 ms	1.17×

7 Conclusion

We study the estimation of epistemic uncertainty in flow-based generalist robot policies and derive VFD as a mathematically grounded and computationally tractable method for this purpose. Building on this, we propose SAVE, a framework for multitask active fine-tuning that allocates the demonstration budget to tasks and initial observations that most require it. VFD is better calibrated than other UQ methods, enabling more data-efficient multitask adaptation with SAVE, and it can detect policy failures more accurately, particularly in OOD settings. In summary, our results establish VFD as an effective method to make generalist policies more reliable and adaptable.

Limitations. We use the uncertainty for the initial observation as a proxy for task difficulty. Our experiments empirically confirm this correlation, but there may be certain scenarios that "look simple" but are challenging for the policy (e.g., due to complex contact dynamics). Moreover, treating uncertainty as the primary signal for determining which data to collect does not account for informational dependencies across tasks. Developing tractable methods to quantify these dependencies and directly maximizing expected information gain for large-scale policies are interesting avenues for future work.

AI USE STATEMENT

Among mandatory disclosure tasks, we used generative AI tools for coding assistance, both for implementation and to support data analysis. We have neither used generative AI tools to formulate mathematical claims, propose or refine hypotheses, assist in writing proofs, nor to help develop theoretical models or conceptual frameworks. We have not used generative AI to generate, clean or reformat datasets, design or provide feedback on research methodology or experiments, or to interpret results.

Additionally, we used generative AI tools for the following tasks with recommended disclosure: creating and modifying scientific figures, identifying relevant literature, and suggesting edits to improve readability. We did not use generative AI for tasks in other categories with recommended disclosure.

We have reviewed all AI-assisted work: code for both implementations and plotting was systematically reviewed, and suggestions regarding related work and readability were incorporated only after human examination. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our research abides by the ICLR Code of Ethics and does not involve human subjects or potentially harmful physical deployment. Our work addresses uncertainty quantification for generalist policies and applies it to failure detection and active fine-tuning. While mostly algorithmic in nature, it shares the ethical and societal considerations of robot learning, particularly involving automation.

This work may have a positive societal impact by improving the reliability of generalist policies in robotic systems. In particular, calibrated epistemic uncertainty can help identify unfamiliar situations, support more data-efficient adaptation, and enable detection of runtime failures before unsafe actions are executed. At the same time, uncertainty estimates are naturally imperfect and may pose risks if deployed as the sole safety mechanism. False confidence could lead to harmful robot behavior, while overly conservative estimates could reduce usability. These risks should be mitigated through conservative threshold design, careful human oversight, safe fallback policies, and rigorous evaluation under realistic deployment conditions before real-world use.

REPRODUCIBILITY STATEMENT

We detail the assumptions made in the derivation of our theoretical results in Appendix A.2. Moreover, we ensure reproducibility of our experimental results by releasing the code at anonymous.4open.science/r/uq_grps and thoroughly reporting training details and hyperparameters in Appendix B.

REFERENCES

Christopher Agia, Rohan Sinha, Jingyun Yang, Ziang Cao, Rika Antonova, Marco Pavone, and Jeannette Bohg. Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress. In *Conference on Robot Learning (CoRL)*, 2024.

Christopher Agia, Rohan Sinha, Jingyun Yang, Rika Antonova, Marco Pavone, Haruki Nishimura, Masha Itkina, and Jeannette Bohg. CUPID: Curating data your robot loves with influence functions. *Conference on Robot Learning (CoRL)*, 2025.

AgiBot-World et al. AgiBot World Colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. In *International Conference on Intelligent Robots and Systems (IROS)*, 2025.

Anastasios N Angelopoulos and Stephen Bates. Conformal prediction: A gentle introduction. *Foundations and Trends in Machine Learning*, 16(4):494–591, 2023.

Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. In *International Conference on Learning Representations (ICLR)*, 2020.

Muhammad Awais, Muzammal Naseer, Salman Khan, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, and Fahad Shahbaz Khan. Foundation models defining a new era in vision: a survey and outlook. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(4):2245–2264, 2025.

Murat Seckin Ayhan and Philipp Berens. Test-time data augmentation for estimation of heteroscedastic aleatoric uncertainty in deep neural networks. In *Medical Imaging with Deep Learning*, 2018.

Marco Bagatella, Jonas Hübotter, Georg Martius, and Andreas Krause. Active fine-tuning of multitask policies. In *International Conference on Machine Learning (ICML)*, 2025.

Lucas Berry, Axel Brando, and David Meger. Shedding light on large generative networks: Estimating epistemic uncertainty in diffusion models. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2024.

Kevin Black et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Conference on Robot Learning (CoRL)*, 2025.

Remi Cadene, Simon Alibert, Francesco Capuano, Michel Aractingi, Adil Zouitine, Pepijn Kooijmans, Jade Choghari, Martino Russi, Caroline Pascal, Steven Palma, et al. LeRobot: An open-source library for end-to-end robot learning. In *International Conference on Learning Representations (ICLR)*, 2026.

Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 10:273–304, 1995.

Matthew Chan, Maria Molina, and Chris Metzler. Estimating epistemic and aleatoric uncertainty with a single model. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *Robotics: Science and Systems (RSS)*, 2023.

Yuchen Cui, David Isele, Scott Niekum, and Kikuo Fujimura. Uncertainty-aware data aggregation for deep imitation learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux - effortless Bayesian deep learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Alexey Dosovitskiy et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

Maximilian Du, Zhanyi Sun, Chen Xu, Paarth Shah, Masha Itkina, and Shuran Song. Memory anchors for continual robot learning. *arXiv preprint arXiv:2608.26545*, 2026.

Bradley Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106:1602–1614, 2011.

Ekaterina Fadeeva et al. Fact-checking the output of large language models via token-level uncertainty quantification. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 9367–9385, 2024.

Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, et al. MolmoAct2: Action reasoning models for real-world deployment. *arXiv preprint arXiv:2605.02881*, 2026.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. LIBERO-Plus: In-depth robustness analysis of vision-language-action models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Gianni Franchi, Nacim Belkhir, Dat Nguyen Trong, Guoxuan Xia, and Andrea Pilzer. Towards understanding and quantifying uncertainty for text-to-image generation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning (ICML)*, 2016.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep Bayesian active learning with image data. In *International Conference on Machine Learning (ICML)*, 2017.

Qiao Gu, Yuanliang Ju, Shengxiang Sun, Igor Gilitschenski, Haruki Nishimura, Masha Itkina, and Florian Shkurti. SAFE: Multitask failure detection for vision-language-action models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Zhanpeng He, Yifeng Cao, and Matei Ciocarlie. Uncertainty comes for free: Human-in-the-loop policies with diffusion models. *arXiv preprint arXiv:2503.01876*, 2025.

Joey Hejna, Suvir Mirchandani, Ashwin Balakrishna, Annie Xie, Ayzaan Wahid, Jonathan Tompson, Pannag Sanketi, Dhruv Shah, Coline Devin, and Dorsa Sadigh. Robot data curation with mutual information estimators. *Robotics: Science and Systems (RSS)*, 2025.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:6840–6851, 2020.

David Holzmüller, Viktor Zaverkin, Johannes Kästner, and Ingo Steinwart. A framework and benchmark for deep batch active learning for regression. *Journal of Machine Learning Research (JMLR)*, 24(164):1–81, 2023.

Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110:457–506, 2021.

Metod Jazbec, Eliot Wong-Toi, Guoxuan Xia, Dan Zhang, Eric Nalisnick, and Stephan Mandt. Generative uncertainty in diffusion models. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2025.

Kshitij Judah, Alan Fern, and Thomas G Dietterich. Active imitation learning via reduction to IID active learning. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2012.

Rafal Karczewski, Markus Heinonen, and Vikas Garg. Diffusion models as cartoonists: The curious case of high density regions. In *International Conference on Learning Representations (ICLR)*, 2025.

Ulas Berk Karli, Tetsu Kurumisawa, and Tesca Fitzgerald. Ask before you act: Token-level uncertainty for intervention in vision-language-action models. In *Second Workshop on Out-of-Distribution Generalization in Robotics at RSS*, 2025.

Alexander Khazatsky et al. DROID: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems*, 2024.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos Policy: Fine-tuning video models for visuomotor control and planning. In *International Conference on Learning Representations (ICLR)*, 2026.

Moo Jin Kim et al. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning (CoRL)*, 2024.

Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. BatchBALD: Efficient and diverse batch acquisition for deep Bayesian active learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

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)*, 2025.

Qi Li, Bo Yin, Weiqi Huang, Ruhao Liu, Bojun Zou, Runpeng Yu, Jingwen Ye, Weihao Yu, and Xinchao Wang. Vision-language-action safety: Threats, challenges, evaluations, and mechanisms. *arXiv preprint arXiv:2604.23775*, 2026.

Chen Ling et al. Uncertainty quantification for in-context learning of large language models. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2024.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Antonio Loquercio, Mattia Segu, and Davide Scaramuzza. A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters (RA-L)*, 2020.

Huan Ma, Jingdong Chen, Joey Tianyi Zhou, Guangyu Wang, and Changqing Zhang. Estimating LLM uncertainty with evidence. *arXiv preprint arXiv:2502.00290*, 2025.

Andrey Malinin and Mark Gales. Uncertainty estimation in autoregressive structured prediction. In *International Conference on Learning Representations (ICLR)*, 2021.

Zhiting Mei, Tenny Yin, Micah Baker, Ola Shorinwa, and Anirudha Majumdar. World models that know when they don't know: Controllable video generation with calibrated uncertainty. *arXiv preprint arXiv:2512.05927*, 2025.

NVIDIA et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Abby O'Neill et al. Open X-embodiment: Robotic learning datasets and RT-X models: Open X-embodiment collaboration. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

Physical Intelligence et al. π_0: A vision-language-action flow model for general robot control. *Robotics: Science and Systems (RSS)*, 2025.

Allen Z Ren et al. Robots that ask for help: Uncertainty alignment for large language model planners. In *Conference on Robot Learning (CoRL)*, 2023.

Ralf Römer, Adrian Kobras, Luca Worbis, and Angela P Schoellig. Failure prediction at runtime for generative robot policies. *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Ralf Römer, Yi Zhang, and Angela P Schoellig. CLARE: Continual learning for vision-language-action models via autonomous adapter routing and expansion. *IEEE Robotics and Automation Letters (RA-L)*, 2026.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations (ICLR)*, 2018.

Burr Settles. Active learning literature survey. Technical report, University of Wisconsin–Madison, 2009.

Ola Shorinwa, Zhiting Mei, Justin Lidard, Allen Z Ren, and Anirudha Majumdar. A survey on uncertainty quantification of large language models: Taxonomy, open research challenges, and future directions. *ACM Computing Surveys*, 58:1–38, 2025.

Mustafa Shukor et al. SmolVLA: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

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.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11975–11986, 2023.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *International Conference on Learning Representations (ICLR)*, 2026.

Brianna Zitkovich et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, 2023.

Appendix

Table of Contents

A Theoretical Results **16**
- A.1 Flow Matching Fundamentals . . . 16
- A.2 Proof of Theorem 4.1 . . . 16

B Experimental Details **18**
- B.1 Computational Resources . . . 18
- B.2 Environments . . . 19
- B.3 Calibration and Active Fine-Tuning . . . 19
- B.4 Active Fine-Tuning Experiments . . . 23
- B.5 Failure Detection . . . 26

C Additional Results **29**
- C.1 Toy Example . . . 29
- C.2 Calibration . . . 29
- C.3 Learning Curves of Active Fine-Tuning . . . 30
- C.4 Statistical Significance of Uncertainty-Guided Data Acquisition . . . 31
- C.5 Iterative Fine-Tuning vs. Retraining . . . 31
- C.6 Hyperparameter Selection for Uncertainty-Guided Active Fine-Tuning . . . 31
- C.7 Failure Detection . . . 33

A Theoretical Results

A.1 Flow Matching Fundamentals

Flow matching (FM) (Lipman et al., 2023) formulates generative modeling as learning a time-dependent vector field $\boldsymbol{u}_s$ that transports samples from a simple base distribution p_0 to the data distribution $p_1 = q$. We consider a noise distribution $p_0(\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_0 \mid \mathbf{0}, \boldsymbol{I})$ and a data distribution $\boldsymbol{x}_1 \sim q$. The OT Gaussian conditional probability path is defined as

$$p_s(\boldsymbol{x} \mid \boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x} \mid s\boldsymbol{x}_1, (1-s)^2 \boldsymbol{I}), \tag{10}$$

and the corresponding flow that pushes the noise distribution p_0 to $p_s(\boldsymbol{x} \mid \boldsymbol{x}_1)$ is given by the affine map

$$\boldsymbol{\phi}_s(\boldsymbol{x}_0) = (1-s)\boldsymbol{x}_0 + s\boldsymbol{x}_1. \tag{11}$$

The conditional velocity field $\boldsymbol{u}_s(\boldsymbol{x} \mid \boldsymbol{x}_1)$ defining this flow via the ODE $\frac{\mathrm{d}}{\mathrm{d}s}\boldsymbol{\phi}_s(\boldsymbol{x}) = \boldsymbol{u}_s(\boldsymbol{\phi}_s(\boldsymbol{x}))$, and generating $p_s(\boldsymbol{x} \mid \boldsymbol{x}_1)$, can be derived as:

$$\boldsymbol{u}_s(\boldsymbol{x} \mid \boldsymbol{x}_1) = \frac{\mathrm{d}}{\mathrm{d}s}\boldsymbol{\phi}_s(\boldsymbol{x}_0) = \boldsymbol{x}_1 - \boldsymbol{x}_0. \tag{12}$$

Marginalizing over the data distribution $q(\boldsymbol{x}_1)$ yields the marginal probability path

$$p_s(\boldsymbol{x}) = \int p_s(\boldsymbol{x} \mid \boldsymbol{x}_1) q(\boldsymbol{x}_1) \mathrm{d}\boldsymbol{x}_1, \tag{13}$$

and the marginal velocity field is defined as

$$\boldsymbol{u}_s(\boldsymbol{x}) = \int \boldsymbol{u}_s(\boldsymbol{x} \mid \boldsymbol{x}_1) \frac{p_s(\boldsymbol{x} \mid \boldsymbol{x}_1) q(\boldsymbol{x}_1)}{p_s(\boldsymbol{x})} \mathrm{d}\boldsymbol{x}_1. \tag{14}$$

Finally, these components must satisfy the continuity equation

$$\frac{\partial}{\partial s} p_s(\boldsymbol{x}) + \operatorname{div}(p_s(\boldsymbol{x}) \boldsymbol{u}_s(\boldsymbol{x})) = 0, \tag{15}$$

which ensures probability mass conservation.

A.2 Proof of Theorem 4.1

To simplify notation, we drop the fixed conditioning input $\boldsymbol{y}$ and the parameterization of the velocity fields as neural networks in the following derivations, considering two terminal distributions $p^1(\boldsymbol{x}) = p_1^1(\boldsymbol{x})$, $p^2(\boldsymbol{x}) = p_1^2(\boldsymbol{x})$ induced by velocity fields $\boldsymbol{u}_s^1(\boldsymbol{x})$, $\boldsymbol{u}_s^2(\boldsymbol{x})$ and identical base distributions $p_0^1(\boldsymbol{x}) = p_0^2(\boldsymbol{x}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$.

We impose the following regularity assumptions. For every $s \in [0, 1)$ and $m \in \{1, 2\}$, we assume that $p_s^m : \mathbb{R}^d \to \mathbb{R}$ and $\boldsymbol{u}_s^m : \mathbb{R}^d \to \mathbb{R}^d$ are continuously differentiable with respect to $\boldsymbol{x}$. Furthermore, for every $s \in [0, 1)$, we assume that there exist constants $C_s, a_s, R_s > 0$ such that, for every $m \in \{1, 2\}$,

$$p_s^m(\boldsymbol{x}) \leq C_s \exp\left(-a_s \|\boldsymbol{x}\|_2^2\right) \qquad \text{for all } \|\boldsymbol{x}\|_2 \geq R_s.$$

We further assume that $\boldsymbol{u}_s^1(\boldsymbol{x})$, $\boldsymbol{u}_s^2(\boldsymbol{x})$, and $\log\left(p_s^1(\boldsymbol{x})/p_s^2(\boldsymbol{x})\right)$ have at most polynomial growth as $\|\boldsymbol{x}\|_2 \to \infty$.

In the proof of Theorem 4.1, we will use the following lemma.

Lemma A.1. *Let $\boldsymbol{u}_s(\boldsymbol{x})$ be the marginal velocity field of an FM model trained with Gaussian OT conditional probability paths, and let $p_s(\boldsymbol{x})$ be the marginal probability path induced by $\boldsymbol{u}_s(\boldsymbol{x})$. Then, for all $s \in [0, 1)$,*

$$\nabla \log p_s(\boldsymbol{x}) = \frac{s\boldsymbol{u}_s(\boldsymbol{x}) - \boldsymbol{x}}{1-s}. \tag{16}$$

Proof. Applying Tweedie's formula (Efron, 2011) to (10) and (13) yields

$$\mathbb{E}[s\boldsymbol{x}_1 \,|\, \boldsymbol{x}_s = \boldsymbol{x}] = \boldsymbol{x} + (1-s)^2 \frac{\nabla p_s(\boldsymbol{x})}{p_s(\boldsymbol{x})} \tag{17}$$

$$\Leftrightarrow \quad \nabla \log p_s(\boldsymbol{x}) = \frac{s\mathbb{E}[\boldsymbol{x}_1 \,|\, \boldsymbol{x}_s = \boldsymbol{x}] - \boldsymbol{x}}{(1-s)^2}. \tag{18}$$

Plugging the conditional velocity (12) into the marginal velocity (14) gives

$$\boldsymbol{u}_s(\boldsymbol{x}) = \int (\boldsymbol{x}_1 - \boldsymbol{x}_0) p_s(\boldsymbol{x}_1 \,|\, \boldsymbol{x}_s = \boldsymbol{x}) \mathrm{d}\boldsymbol{x}_1 \tag{19}$$

$$= \mathbb{E}[\boldsymbol{x}_1 - \boldsymbol{x}_0 \,|\, \boldsymbol{x}_s = \boldsymbol{x}] \tag{20}$$

$$= \frac{\mathbb{E}[\boldsymbol{x}_1 \,|\, \boldsymbol{x}_s = \boldsymbol{x}] - \boldsymbol{x}}{1-s} \tag{21}$$

$$\Leftrightarrow \quad \mathbb{E}[\boldsymbol{x}_1 \,|\, \boldsymbol{x}_s = \boldsymbol{x}] = \boldsymbol{x} + (1-s)\boldsymbol{u}_s(\boldsymbol{x}), \tag{22}$$

where (21) follows from $\boldsymbol{x} = (1-s)\boldsymbol{x}_0 + s\boldsymbol{x}_1 \Leftrightarrow \boldsymbol{x}_1 - \boldsymbol{x}_0 = \frac{\boldsymbol{x}_1 - \boldsymbol{x}}{1-s}$ and taking the conditional expectation. Finally, plugging (22) into (18) yields

$$\nabla \log p_s(\boldsymbol{x}) = \frac{s\boldsymbol{x} + s(1-s)\boldsymbol{u}_s(\boldsymbol{x}) - \boldsymbol{x}}{(1-s)^2} \tag{23}$$

$$= \frac{s\boldsymbol{u}_s(\boldsymbol{x}) - \boldsymbol{x}}{1-s}, \tag{24}$$

which concludes the proof. □

Proof of Theorem 4.1. Recall that we aim to express the KL divergence between two distributions $p^1(\boldsymbol{x})$ and $p^2(\boldsymbol{x})$ parameterized as FM models in terms of their time-dependent velocity fields $\boldsymbol{u}_s^1(\boldsymbol{x})$ and $\boldsymbol{u}_s^2(\boldsymbol{x})$. We write

$$D_{\mathrm{KL}}(p^1(\boldsymbol{x}) \,||\, p^2(\boldsymbol{x})) = D_{\mathrm{KL}}(p^1(\boldsymbol{x}) \,||\, p^2(\boldsymbol{x})) - D_{\mathrm{KL}}(p_0^1(\boldsymbol{x}) \,||\, p_0^2(\boldsymbol{x})) + D_{\mathrm{KL}}(p_0^1(\boldsymbol{x}) \,||\, p_0^2(\boldsymbol{x})) \tag{25}$$

$$= \int_0^1 \frac{\partial}{\partial s} D_{\mathrm{KL}}(p_s^1(\boldsymbol{x}) \,||\, p_s^2(\boldsymbol{x})) \mathrm{d}s, \tag{26}$$

where the last step follows from the fact that $p_0^1 = p_0^2$ since both models use the same source distribution. Applying the product rule, we can reformulate the term in the integral as

$$\frac{\partial}{\partial s} D_{\mathrm{KL}}(p_s^1(\boldsymbol{x}) \,||\, p_s^2(\boldsymbol{x})) = \frac{\partial}{\partial s} \int p_s^1(\boldsymbol{x}) \log \frac{p_s^1(\boldsymbol{x})}{p_s^2(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} \tag{27}$$

$$= \int \frac{\partial}{\partial s} p_s^1(\boldsymbol{x}) \log \frac{p_s^1(\boldsymbol{x})}{p_s^2(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} + \int p_s^1(\boldsymbol{x}) \frac{\partial}{\partial s} \log \frac{p_s^1(\boldsymbol{x})}{p_s^2(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} \tag{28}$$

$$= \int \frac{\partial}{\partial s} p_s^1(\boldsymbol{x}) \log \frac{p_s^1(\boldsymbol{x})}{p_s^2(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} + \int p_s^1(\boldsymbol{x}) \left(\frac{\frac{\partial}{\partial s} p_s^1(\boldsymbol{x})}{p_s^1(\boldsymbol{x})} - \frac{\frac{\partial}{\partial s} p_s^2(\boldsymbol{x})}{p_s^2(\boldsymbol{x})} \right) \mathrm{d}\boldsymbol{x} \tag{29}$$

$$= \int \frac{\partial}{\partial s} p_s^1(\boldsymbol{x}) \log \frac{p_s^1(\boldsymbol{x})}{p_s^2(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} + \underbrace{\int \frac{\partial}{\partial s} p_s^1(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}}_{=0} - \int \frac{p_s^1(\boldsymbol{x})}{p_s^2(\boldsymbol{x})} \frac{\partial}{\partial s} p_s^2(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}. \tag{30}$$

where the second term vanishes since $\int \frac{\partial}{\partial s} p_s^1(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \frac{\partial}{\partial s} \int p_s^1(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \frac{\partial}{\partial s} 1 = 0$. To reformulate the other terms in (30), we proceed as

$$\frac{\partial}{\partial s} D_{\mathrm{KL}}(p_s^1(\boldsymbol{x}) \,||\, p_s^2(\boldsymbol{x})) = \int \frac{\partial}{\partial s} p_s^1(\boldsymbol{x}) \log \frac{p_s^1(\boldsymbol{x})}{p_s^2(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} - \int \frac{p_s^1(\boldsymbol{x})}{p_s^2(\boldsymbol{x})} \frac{\partial}{\partial s} p_s^2(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \tag{31}$$

$$= -\int \mathrm{div}\big(p_s^1(\boldsymbol{x}) \boldsymbol{u}_s^1(\boldsymbol{x})\big) \log \frac{p_s^1(\boldsymbol{x})}{p_s^2(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} + \int \frac{p_s^1(\boldsymbol{x})}{p_s^2(\boldsymbol{x})} \mathrm{div}\big(p_s^2(\boldsymbol{x}) \boldsymbol{u}_s^2(\boldsymbol{x})\big) \, \mathrm{d}\boldsymbol{x} \tag{32}$$

$$= \int p_s^1(\boldsymbol{x}) \boldsymbol{u}_s^{1,\top}(\boldsymbol{x}) \nabla \log \frac{p_s^1(\boldsymbol{x})}{p_s^2(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} - \int p_s^2(\boldsymbol{x}) \boldsymbol{u}_s^{2,\top}(\boldsymbol{x}) \nabla \left(\frac{p_s^1(\boldsymbol{x})}{p_s^2(\boldsymbol{x})} \right) \mathrm{d}\boldsymbol{x} \tag{33}$$

$$= \int p_s^1(\boldsymbol{x}) \boldsymbol{u}_s^{1,\top}(\boldsymbol{x}) \nabla \log \frac{p_s^1(\boldsymbol{x})}{p_s^2(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} - \int p_s^1(\boldsymbol{x}) \boldsymbol{u}_s^{2,\top}(\boldsymbol{x}) \nabla \log \frac{p_s^1(\boldsymbol{x})}{p_s^2(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} \tag{34}$$

$$= \int p_s^1(\boldsymbol{x}) \left(\boldsymbol{u}_s^1(\boldsymbol{x}) - \boldsymbol{u}_s^2(\boldsymbol{x})\right)^\top \nabla \log \frac{p_s^1(\boldsymbol{x})}{p_s^2(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} \tag{35}$$

$$= \int p_s^1(\boldsymbol{x}) \left(\boldsymbol{u}_s^1(\boldsymbol{x}) - \boldsymbol{u}_s^2(\boldsymbol{x})\right)^\top \left(\nabla \log p_s^1(\boldsymbol{x}) - \nabla \log p_s^2(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x}. \tag{36}$$

Here, (32) follows from the continuity equation (15). Equation (33) follows from integration by parts, with the boundary terms vanishing under the regularity assumptions stated above.

Applying Lemma A.1 to the last term in the integral (36) yields

$$\nabla \log p_s^1(\boldsymbol{x}) - \nabla \log p_s^2(\boldsymbol{x}) = \frac{s}{1-s} (\boldsymbol{u}_s^1(\boldsymbol{x}) - \boldsymbol{u}_s^2(\boldsymbol{x})). \tag{37}$$

Defining $\kappa_s = \frac{s}{1-s}$ and plugging (37) into (36) results in

$$\frac{\partial}{\partial s} D_{\mathrm{KL}}(p_s^1(\boldsymbol{x}) \,||\, p_s^2(\boldsymbol{x})) = \int p_s^1(\boldsymbol{x}) (\boldsymbol{u}_s^1(\boldsymbol{x}) - \boldsymbol{u}_s^2(\boldsymbol{x}))^\top (\nabla \log p_s^1(\boldsymbol{x}) - \nabla \log p_s^2(\boldsymbol{x})) \mathrm{d}\boldsymbol{x} \tag{38}$$

$$= \int p_s^1(\boldsymbol{x}) \kappa_s (\boldsymbol{u}_s^1(\boldsymbol{x}) - \boldsymbol{u}_s^2(\boldsymbol{x}))^\top (\boldsymbol{u}_s^1(\boldsymbol{x}) - \boldsymbol{u}_s^2(\boldsymbol{x})) \mathrm{d}\boldsymbol{x} \tag{39}$$

$$= \kappa_s \mathbb{E}_{\boldsymbol{x} \sim p_s^1(\cdot)} \left[\|\boldsymbol{u}_s^1(\boldsymbol{x}) - \boldsymbol{u}_s^2(\boldsymbol{x})\|_2^2 \right]. \tag{40}$$

Finally, we can insert (40) into (26) and obtain

$$D_{\mathrm{KL}}(p^1(\boldsymbol{x}) \,||\, p^2(\boldsymbol{x})) = \int_0^1 \kappa_s \mathbb{E}_{\boldsymbol{x} \sim p_s^1(\cdot)} \left[\|\boldsymbol{u}_s^1(\boldsymbol{x}) - \boldsymbol{u}_s^2(\boldsymbol{x})\|_2^2 \right] \mathrm{d}s, \tag{41}$$

concluding the proof. □

Remark A.2. Although the weighting κ_s in (41) diverges as $s \to 1$, the VFD uncertainty estimate remains finite. In practice, our estimator (7) evaluates the velocity difference on a grid $s_\ell = \ell \delta s$ with $\ell \in \{0, \ldots, N_s - 1\}$, so the largest weight is $\kappa_{1-\delta s} = \frac{1-\delta s}{\delta s}$, which is finite for $\delta s > 0$.

B Experimental Details

B.1 Computational Resources

We provide the code to reproduce our results at anonymous.4open.science/r/uq_grps. Our codebase builds on LeRobot (Cadene et al., 2026), released under the Apache License, and FIPER (Römer et al., 2025), released under the MIT License. The simulation experiments involving FM-Policy and SmolVLA were run on a compute node with two NVIDIA RTX 4090 GPUs, 8 CPU cores, and 64 GB of RAM. With this hardware setup, one active learning experiment with 15 rounds of uncertainty quantification, episode selection, 4,000 gradient steps, and 30 evaluation rollouts per task takes approximately 12.5 hours. The simulation experiments involving X-VLA and FastWAM were conducted on NVIDIA H100 and H200 GPUs, respectively. The models used in our hardware experiments were trained on an NVIDIA H100 GPU, and we ran real-world inference on an NVIDIA

RTX 5090 GPU. The total computing budget for all experiments, including developing the codebase and running preliminary tests, was about 10,000 GPU hours.

B.2 Environments

`Push-T`. The `Push-T` benchmark simulates an agent in a 2D environment that must push a T-shaped object into a drawn target configuration. The observation contains the agent's position and an RGB image of the scene, and the actions are position commands. Despite its low-dimensional action space, `Push-T` is a relatively challenging environment due to the multimodality of the demonstrations and the high diversity of initial configurations. The latter makes it a suitable benchmark for evaluating the selection of initial states for querying expert demonstrations.

`LIBERO`. We use the `LIBERO` simulation benchmark (Liu et al., 2023) (MIT License) in our experiments. The environments include a Franka robotic manipulator with a parallel-jaw gripper in a kitchen environment. For each task, up to 50 human expert demonstrations are available. The observations $\boldsymbol{o}_t$ consist of two RGB images with spatial resolution 256×256, one from a fixed workspace view and one from a camera mounted on the robot's end-effector. Additionally, observations include the 8-dimensional proprioceptive state of the arm, comprising the end-effector position (3D), the end-effector orientation parameterized as an axis-angle (3D), and the gripper joint positions (2D). Actions $\boldsymbol{a}_t$ are continuous and 7-dimensional: 3D delta end-effector translation, 3D delta end-effector orientation, and 1D gripper command.

`LIBERO-Plus`. To evaluate failure detection in OOD settings in simulation, we use `LIBERO-Plus` (Fei et al., 2026), which perturbs the `LIBERO` tasks across seven dimensions: camera viewpoint, robot initial state, language instruction, lighting, background, sensor noise, and object layout, each with several levels of severity. The observation and action spaces are identical to those in `LIBERO`.

Real-world tasks. We conduct hardware experiments with a Franka FR3 manipulator. The observation contains the language command, RGB images from a wrist-mounted and a third-person camera, the Cartesian end-effector pose and continuous gripper state. The policies generate desired end-effector poses (position and Euler angles) and gripper state as actions at a control frequency of 15 Hz. The pose targets are tracked by a low-level Cartesian torque controller operating at 500 Hz. We define three manipulation tasks involving different objects and motion patterns, each consisting of two to three phases (see Figure 6):

- `Place`: Grasp and lift a bowl randomly placed on the table (phase 1) and place it on a plate at a fixed location (phase 2). Language command: *put the bowl on the plate*.
- `Stack`: Grasp and lift a bowl randomly placed on the table (phase 1) and stack it onto three other bowls located at the center of the table (phase 2). Language command: *stack the orange bowl on the other bowls*.
- `Drawer`: Grasp and lift a Lego block pressed into a Lego plate (phase 1), place it into an opened drawer compartment (phase 2), and close the drawer (phase 3). Language command: *put the lego into the drawer*.

B.3 Calibration and Active Fine-Tuning

B.3.1 Ensemble Training

In each experiment, we take a single base checkpoint and fine-tune it into an ensemble using a dataset $\mathcal{D}_{\text{pre}}$. For `Push-T`, $\mathcal{D}_{\text{pre}}$ contains 50 episodes drawn uniformly at random from the available 206 demonstrations. In our experiments on `LIBERO-Long`, $\mathcal{D}_{\text{pre}}$ contains all episodes from the `LIBERO-Goal`, `-Spatial`, and `-Object` suites, each covering 10 tasks. Training on these 30 tasks alone yielded zero initial success rate across the `LIBERO-Long` tasks we consider for active fine-tuning. Since we aim to test algorithms' capabilities to allocate demonstration budget to the tasks that need it most, we instill varying levels of initial task competence by also including tasks $(1, 2, 3)$ from `LIBERO-Long` into $\mathcal{D}_{\text{pre}}$.

Additional training details are provided in Table 6.

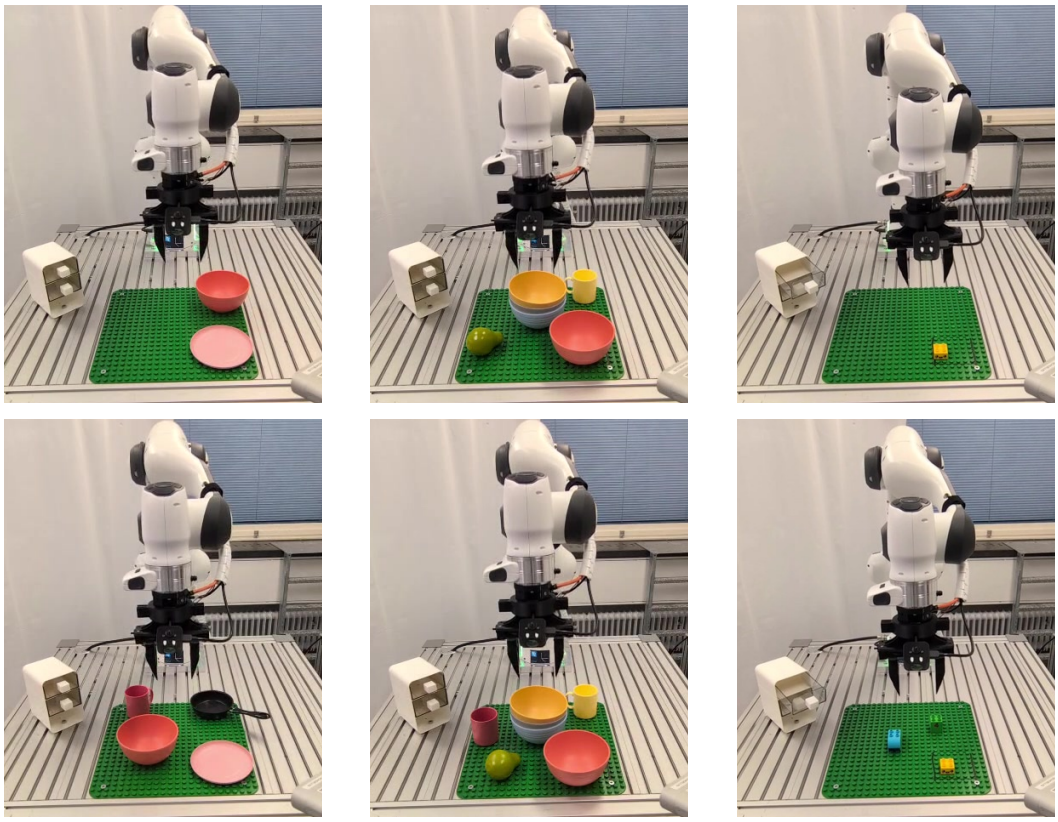

Figure 6: **Real-world tasks.** Top: nominal settings used to collect demonstrations and evaluate active fine-tuning. Bottom: OOD settings with novel distractor objects in our failure detection experiments.

B.3.2 Uncertainty Quantification Methods

The computation of VFD-based epistemic uncertainty estimation is summarized in Algorithm 1 and illustrated in Figure 7. For all policies, including FastWAM, we compute VFD only over the generated action chunks.

In the following, we describe the implementation of the baselines considered in Section 6.2.

Algorithm 1 Efficient Epistemic Uncertainty Estimation for Flow-Matching Models using VFD.

Input: Flow-matching model ensemble $p^{\boldsymbol{\theta}_1}(\boldsymbol{x} \mid \boldsymbol{y})$, $p^{\boldsymbol{\theta}_2}(\boldsymbol{x} \mid \boldsymbol{y})$ with vector fields $\mathcal{V} = (\boldsymbol{v}_s^{\boldsymbol{\theta}_1}, \boldsymbol{v}_s^{\boldsymbol{\theta}_2})$ trained via OT paths, conditioning input $\boldsymbol{y}$, ODE integration step size $\delta s = 1/N_s$.
Output: Estimate of the epistemic uncertainty $u_{\mathrm{e}}(\boldsymbol{y}; \mathcal{V})$ at the conditioning input $\boldsymbol{y}$.
Hyperparameters: Batch size B.
Sample $\boldsymbol{x}_0^{(1,1:B)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, $\boldsymbol{x}_0^{(2,1:B)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$.
Initialize $u_{\mathrm{e}} \leftarrow 0$.
for $\ell = 0$ **to** $N_s - 1$ **do** ▷ *ODE integration.*
    Set $s = \ell \delta s$.
    for each $i \in \{1, 2\}$ **do**
        Compute the expected VFDs $d_{\ell,i}^2 = \frac{1}{B} \sum_{b=1}^{B} \left\| \boldsymbol{v}_s^{\boldsymbol{\theta}_1}\left(\boldsymbol{x}_s^{(i,b)}, \boldsymbol{y}\right) - \boldsymbol{v}_s^{\boldsymbol{\theta}_2}\left(\boldsymbol{x}_s^{(i,b)}, \boldsymbol{y}\right) \right\|_2^2$.
        Update uncertainty estimate $u_{\mathrm{e}} \leftarrow u_{\mathrm{e}} + \frac{1}{2N_s} \frac{s}{1-s} d_{\ell,i}^2$.
        Forward integrate $\boldsymbol{x}_{s+\delta s}^{(i,1:B)} = \boldsymbol{x}_s^{(i,1:B)} + \boldsymbol{v}_s^{\boldsymbol{\theta}_i}\left(\boldsymbol{x}_s^{(i,1:B)}, \boldsymbol{y}\right) \delta s$.
return $u_{\mathrm{e}}(\boldsymbol{y}; \mathcal{V}) = u_{\mathrm{e}}$.

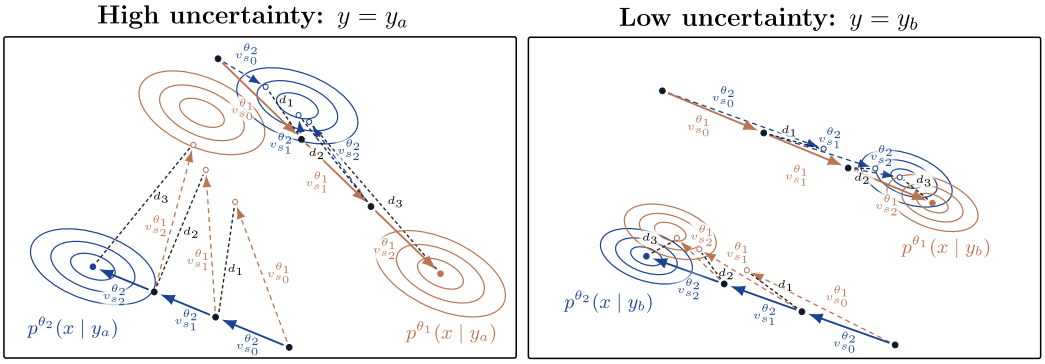

Figure 7: **VFD visualization.** We illustrate the computation of VFD for two conditioning inputs with different levels of epistemic uncertainty: high (left) and low (right).

Action-L2 implementation. Action-L2 estimates epistemic uncertainty from the disagreement in terminal action distributions. Given an observation $\boldsymbol{o}$, we draw independent Gaussian noise samples from p_0 and generate C terminal action chunks from the policy and from each ensemble member. Let

$$\{\boldsymbol{A}_i^{\mathrm{s}}(\boldsymbol{o})\}_{i=1}^{C}$$

denote the action chunks generated by the policy, and let

$$\{\boldsymbol{A}_j^{(m)}(\boldsymbol{o})\}_{j=1}^{C}$$

denote the action chunks generated by ensemble member $m \in \{1, \ldots, M\}$.

The discrepancy between the policy and ensemble member m is computed as the mean pairwise Euclidean distance between their sampled terminal action chunks:

$$d_m(\boldsymbol{o}) = \frac{1}{C^2} \sum_{i=1}^{C} \sum_{j=1}^{C} \left\| \boldsymbol{A}_i^{\mathrm{s}}(\boldsymbol{o}) - \boldsymbol{A}_j^{(m)}(\boldsymbol{o}) \right\|_2 .$$

The Action-L2 epistemic uncertainty estimate is then the average discrepancy across ensemble members:

$$u_{\text{Action-L2}}(\boldsymbol{o}) = \frac{1}{M} \sum_{m=1}^{M} d_m(\boldsymbol{o}).$$

ACE implementation. ACE (Römer et al., 2025) estimates predictive uncertainty from the dispersion of action samples generated by a single policy. Given an observation $\boldsymbol{o}$, we draw C independent Gaussian noise samples and generate C action chunks via FM ODE integration. For each sampled chunk $c \in \{1, \ldots, C\}$, we extract the first three (`Push-T`: two) action dimensions at each timestep t, denoted by $\{\boldsymbol{p}_t^{(c)}\}_{c=1}^{C}$. At each timestep $t \in \{1, \ldots, H\}$ in the prediction horizon H, the sampled positions $\{\boldsymbol{p}_t^{(c)}\}_{c=1}^{C}$ are discretized into a 3D grid. The grid cell size along each coordinate axis is set to 0.03 times the empirical range of the samples along that axis. In our experiments, this performed better than using fixed cell sizes computed offline from calibration data, as proposed by the authors. Let $\mathcal{B}_t$ denote a set of bins over the occupied grid at time t and let $P_t(b)$ be the empirical probability of a sample falling into bin $b \in \mathcal{B}_t$ at time t. The ACE uncertainty estimate is the Shannon entropy of these bin counts, averaged over the prediction horizon:

$$u_{\text{ACE}}(\boldsymbol{o}) = -\frac{1}{H} \sum_{t=1}^{H} \sum_{b \in \mathcal{B}_t} P_t(b) \log_2 P_t(b).$$

DECU implementation. DECU (Berry et al., 2024) estimates epistemic uncertainty in diffusion models from the disagreement among ensemble members' denoising means at an intermediate point of the generative process. Given the close relationship between diffusion and FM, we adapt DECU to FM by replacing the denoising mean with the learned velocity field. Given an observation $\boldsymbol{o}$, we sample C independent Gaussian noise vectors $\boldsymbol{x}_0^{(c)} \sim p_0$ for $c \in \{1, \dots, C\}$. Each ensemble member integrates the FM ODE from $s = 0$ to a branching time $s_b = 0.995$ (as in Berry et al. (2024)) to obtain intermediate states $\boldsymbol{x}_{s_b}^{(c)}$. At s_b, we evaluate the velocity fields of all M ensemble members, $\boldsymbol{v}_i^{(c)} = \boldsymbol{v}_{s_b}^{\boldsymbol{\theta}_i}(\boldsymbol{x}_{s_b}^{(c)}, \boldsymbol{o})$ for $i \in \{1, \dots, M\}$.

Disagreement is measured using the Pairwise-Distance Estimator (PaiDE). For each sampled chunk c, define the squared pairwise velocity-field distances

$$D_{ij}^{(c)} = \left\| \boldsymbol{v}_i^{(c)} - \boldsymbol{v}_j^{(c)} \right\|_2^2, \qquad i, j \in \{1, \dots, M\}.$$

The score for sample c is then

$$d_{\mathrm{DECU}}^{(c)}(\boldsymbol{o}) = -\frac{1}{M} \sum_{i=1}^{M} \log \left(\frac{1}{M} \sum_{j=1}^{M} \exp\left(-D_{ij}^{(c)}\right) \right).$$

The DECU epistemic uncertainty estimate for observation $\boldsymbol{o}$ is the mean score over the C sampled chunks:

$$u_{\mathrm{DECU}}(\boldsymbol{o}) = \frac{1}{C} \sum_{c=1}^{C} d_{\mathrm{DECU}}^{(c)}(\boldsymbol{o}).$$

GU implementation. As described in the main text, GU (Jazbec et al., 2025) approximates the posterior predictive action distribution as a Gaussian. Using our deep ensemble, we compute a raw uncertainty score for each initial observation $\boldsymbol{o}_{kl}$, where k indexes the task and l indexes the observation within that task. For a fixed initial noise sample, let $\boldsymbol{A}^{(i)}(\boldsymbol{o}_{kl}) \in \mathbb{R}^{H \times D_a}$ denote the action trajectory generated by ensemble member $i \in \{1, \dots, M\}$, and let $a_{kl,t,d}^{(i)}$ denote its action value at timestep $t \in \{1, \dots, H\}$ and action dimension $d \in \{1, \dots, D_a\}$.

The raw GU uncertainty score is obtained by summing the log-variance of the ensemble predictions across all timesteps and action dimensions:

$$u_{kl}^{\mathrm{raw}} = \sum_{t=1}^{H} \sum_{d=1}^{D_a} \log \left(\mathrm{Var}_{i=1}^{M} \left[a_{kl,t,d}^{(i)} \right] + \epsilon_{\mathrm{var}} \right),$$

where $\epsilon_{\mathrm{var}} = 10^{-8}$ is added for numerical stability.

Because the differential entropy of a continuous distribution can be negative, these raw scores may also be negative. To obtain non-negative uncertainty weights, we first aggregate the observation-level scores into a raw task-level uncertainty score U_k^{raw} and then shift the task-level scores by

$$U_k = U_k^{\mathrm{raw}} - \min_{k' \in \{1, \dots, K\}} U_{k'}^{\mathrm{raw}} + \epsilon_{\mathrm{off}} \left(\max_{k' \in \{1, \dots, K\}} U_{k'}^{\mathrm{raw}} - \min_{k' \in \{1, \dots, K\}} U_{k'}^{\mathrm{raw}} \right).$$

Here, $\epsilon_{\mathrm{off}} > 0$ is a small offset coefficient. This shift removes negative values, and when the task scores are not all identical, the range-dependent offset moves the minimum-uncertainty task away from zero so that it retains non-zero sampling probability after normalization.

Entropy implementation. Entropy (Malinin & Gales, 2021) utilizes the extent to which the VLM (SmolVLA backbone) successfully predicts the next token at each valid prefix position. Given an observation $\boldsymbol{o}$, the policy constructs a prefix sequence of length S_{prefix} containing images, robot states and language. The VLM predicts the next-token probability distribution ψ_m at each valid prefix

position $m \in \{1, \ldots, S_{\text{prefix}}\}$, with a single forward pass. The uncertainty score is calculated by the Shannon entropy of this distribution, averaged over all valid prefix tokens:

$$u_{\text{Entropy}} = -\frac{1}{S_{\text{prefix}}} \sum_{m=1}^{S_{\text{prefix}}} \sum_{z \in \mathcal{Z}} \psi_m(z) \log \psi_m(z),$$

where $\mathcal{Z}$ is the VLM vocabulary and $\psi_m(z)$ is the predicted probability of token z at position m.

Perplexity implementation. Perplexity (Fadeeva et al., 2024) estimates uncertainty by measuring the likelihood of the language instruction under the VLM backbone. Similar to Entropy, it evaluates the prefix in a single forward pass without action generation. We extract the predicted next-token log-probabilities for the S_{lang} language tokens within their respective prefix. Let z_m be the true language token at sequence position m. The uncertainty score u is the exponentiated negative mean log-probability of these tokens:

$$u_{\text{Perp}} = \exp\left(-\frac{1}{S_{\text{lang}}} \sum_{m=1}^{S_{\text{lang}}} \log \psi_m(z_m)\right).$$

Diversity-Based Greedy. The diversity-based greedy baseline (Sener & Savarese, 2018) selects initial observations to maximize visual diversity. Let $\mathcal{O}_k$ denote the pool of candidate initial observations, and $\hat{\mathcal{O}}$ denote the set of selected observations. We extract visual features for each $\boldsymbol{o}_i \in \mathcal{O}_k$ using a SigLIP (Zhai et al., 2023) vision encoder. We rank all candidates using the k-Center Greedy algorithm. At each step, we select $\boldsymbol{o}^*$ that maximizes the minimum distance Δ to $\hat{\mathcal{O}}$:

$$\boldsymbol{o}^* = \arg \max_{\boldsymbol{o}_i \in \mathcal{O}_k \setminus \hat{\mathcal{O}}} \min_{\boldsymbol{o}_j \in \hat{\mathcal{O}}} \Delta(\boldsymbol{o}_i, \boldsymbol{o}_j),$$

where $\Delta(\boldsymbol{o}_i, \boldsymbol{o}_j)$ is the Euclidean distance between visual features. We update $\hat{\mathcal{O}} \leftarrow \hat{\mathcal{O}} \cup \{\boldsymbol{o}^*\}$ and repeat until all observations are ranked. In each active fine-tuning round, we select the top n_{e} observations from this ranking without replacement.

B.3.3 Calibration Experiments

We primarily measure calibration by comparing the uncertainty estimate for an initial observation with the probability of success in rolling out the policy from that observation. More specifically, we calculate the mean uncertainty (8) per task and evaluate the per-task success rate by rolling out the first ensemble member $\pi_r^{(1)}$ multiple times for different initial states of the robot and the objects in the scene. To assess whether higher uncertainty correlates with a lower success rate, we calculate the Spearman rank correlation coefficient ρ between the two quantities. If and only if the success rate decreases strictly monotonically with higher uncertainty, $\rho = -1$. As a secondary metric, we also report the Pearson correlation coefficient, which measures the strength of the linear relationship between the two quantities.

In Figure 3, we investigate whether the uncertainty scores are well-calibrated when a decrease in success rate stems solely from varying the language command. To this end, we use different language prompts for each task, which are listed in Table 5.

B.4 Active Fine-Tuning Experiments

We provide implementation details about our simulated and real-world active fine-tuning experiments in Table 6 and the full algorithm in Algorithm 2. In our experiments, for a fair comparison, all data-acquisition strategies retrieve demonstrations from the same expert dataset. An initial configuration can only be selected once per fine-tuning round, but in multiple different rounds.

Table 5: **Prompt variations in Figure 3.** We list the prompts used to evaluate calibration of the uncertainty estimators with respect to variations in the language input in `LIBERO-Long`. Changes from the original prompt are marked in bold.

Task	P1 (original)	P2	P3	P4	P5
1	put both the alphabet soup and the tomato sauce in the basket	put the alphabet soup and the tomato sauce **into** the basket	**place** both the alphabet soup and the tomato sauce in the basket	**move** both the alphabet soup and tomato sauce **to** the basket	**transfer the soup cans to** the basket
2	put both the cream cheese box and the butter in the basket	**place** both the cream cheese box and the butter in the basket	put the cream cheese box and the butter **into** the basket	**move** the cream cheese and the butter **into** the basket	**place the dairy products** in the basket
3	turn on the stove and put the moka pot on it	**switch** on the stove and **place** the moka pot on it	**start** the stove and **set** the moka pot on **the burner**	**activate** the stove and put the moka pot on **top of** it	**power** on the **cooktop** and **position the coffee maker** on it
4	put the black bowl in the bottom drawer of the cabinet and close it	put the black bowl **into** the **lower** drawer of the cabinet and **shut** it	**place** the black bowl in the **bottom cabinet drawer** and close it	**store** the black bowl in the **bottom cabinet drawer** and close **the drawer**	**move the dark** bowl to the **lowest cabinet compartment** and **shut** it
5	put the white mug on the left plate and put the yellow and white mug on the right plate	**set** the white mug on the left plate and the yellow and white mug on the right plate	**place** the white mug on the left plate and **place** the yellow and white mug on the right plate	put the white **cup** on the left **dish** and the **yellow-white cup** on the right **dish**	**arrange the mugs: white one** on the left plate, **yellow and white one** on the right plate
6	pick up the book and place it in the back compartment of the caddy	**grab** the book and place it in the back compartment of the caddy	pick up the book and **put** it in the **rear** compartment of the caddy	**take** the book and place it in the back **section** of the caddy	**move** the book to the back compartment of the **organizer**
7	put the white mug on the plate and put the chocolate pudding to the right of the plate	**place** the white mug on the plate and put the chocolate pudding to the right of the plate	put the white **cup** on the **dish** and **move** the chocolate pudding to the right **side of the plate**	**set** the white mug on the plate and **place** the chocolate pudding to the right of the plate	**position the** mug on the plate and **set the** pudding to the right of **it**
8	place both the alphabet soup and the cream cheese box in the basket	**put** both the alphabet soup and the cream cheese box in the basket	**put** the alphabet soup and the cream cheese box **into** the basket	**move** both the alphabet soup and cream cheese box **to** the basket	**transfer the soup and cheese box to** the basket
9	put both moka pots on the stove	**place** both moka pots on the stove	put the **two** moka pots on the stove	**set** both **coffee** pots on the stove	**move the two** moka pots **onto the cooktop**
10	put the yellow and white mug in the microwave and close it	**place** the yellow and white mug in the microwave and close it	put the yellow and white mug **into** the microwave and **shut** it	**move** the yellow and white mug **into** the microwave and close **the door**	**place the yellow-white** mug in the microwave and **shut the door**

AMF implementation. The central idea of AMF (Bagatella et al., 2025) is to use the demonstrations collected so far to estimate which task's next demonstration would most improve performance across all tasks. It does so by selecting the task that maximizes the cumulative mutual information between a new demonstration for that task and the policies for *all* relevant tasks.

Since this method uses previously collected demonstrations for task selection, we start with a prefill phase, collecting one episode per task (for our `LIBERO` experiments, this takes the first two rounds, using the same acquisition budget as the other methods). We encode frames from previous demonstrations with loss-gradient embeddings from the last action-projection layer (Bagatella et al.

Table 6: Implementation details for the active fine-tuning experiments.

Environment	Push-T	LIBERO-Long			Real world
Policy	FM-Policy	SmolVLA	X-VLA	FastWAM	SmolVLA
Ensemble training					
Demonstrations in $\mathcal{D}_{\text{pre}}$	50	1,437	1,437	1,437	2,549
Training steps	30,000	30,000	30,000	30,000	30,000
Batch size	128	32	32	32	24
Learning rate	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}
Learning rate schedule	cosine	cosine	cosine	cosine	cosine
Final learning rate	0	2.5×10^{-6}	2.5×10^{-6}	2.5×10^{-6}	2.5×10^{-6}
Warmup steps	1,000	1,000	1,000	1,000	1,000
Iterative fine-tuning					
Demonstrations per round	5	5	5	5	10
Rounds R	30	15	15	15	15
Training steps per round	2,000	4,000	4,000	4,000	10,000
Batch size	128	32	64	64	32
Learning rate	1×10^{-4}	5×10^{-5}	1×10^{-4}	1×10^{-4}	5×10^{-5}
Learning rate schedule	cosine	cosine	cosine	cosine	cosine
Final learning rate	0	5×10^{-6}	5×10^{-6}	5×10^{-6}	5×10^{-6}
Warmup steps	100	200	200	200	500
Replay ratio λ	0.3	0.5	0.5	0.5	0.5
Evaluation					
Evaluated rounds	all	all	all	all	6, 9, 12, 15
Rollouts per round	200	300	300	300	30

Algorithm 2 Uncertainty-Based Active Fine-Tuning of Flow-Based Generalist Policies with SAVE.

Input: Base policy π_{b}, pool of initial configurations $\{\mathcal{O}_k\}_{k=1}^{K}$, diverse dataset $\mathcal{D}_{\text{pre}}$.
Output: Final policy π_R.
Hyperparameters: Rounds R, expert queries per round n_{e}, temperature τ, replay ratio λ.
From π_{b}, train an ensemble Π_0 on $\mathcal{D}_{\text{pre}}$.
Initialize buffer $\mathcal{D}_{\text{new}}^{(\leq 0)} \leftarrow \emptyset$.
for $r = 0$ **to** $R - 1$ **do** ▷ *Active learning loop.*
  for $k = 1$ **to** K **do**
    for each $\boldsymbol{o}_{kl} \in \mathcal{O}_k$ **do**
      Compute VFD uncertainty $u_{kl}^{(r)} = u_{\text{e}}(\boldsymbol{o}_{kl}; \Pi_r)$ via (7).
    Compute task uncertainty $U_k^{(r)}$ via (8).
  Compute task weights $W^{(r)} = \left(w_1^{(r)}, \ldots, w_K^{(r)}\right)$ via $w_k^{(r)} = \frac{\left(U_k^{(r)}\right)^{\tau}}{\sum_{k'=1}^{K}\left(U_{k'}^{(r)}\right)^{\tau}}$.
  for $\xi = 1$ **to** n_{e} **do** ▷ *Data acquisition.*
    Sample $k \sim \text{Cat}(W^{(r)})$.
    Select the most uncertain initial observation $\boldsymbol{o}^*$ from task T_k via (9).
    Query the expert to collect a demonstration τ^* starting from $\boldsymbol{o}^*$ and add τ^* to $\mathcal{D}_{\text{new}}^{(\leq r)}$.
  Construct training mixture $\mathcal{D}_{\text{train}}^{(r)} \sim \lambda\ \text{Unif}(\mathcal{D}_{\text{pre}}) + (1 - \lambda)\ \text{Unif}(\mathcal{D}_{\text{new}}^{(\leq r)})$.
  $\Pi_{r+1} \leftarrow \text{Fine-tune}(\Pi_r, \mathcal{D}_{\text{train}}^{(r)})$, $D_{\text{new}}^{(\leq r+1)} \leftarrow D_{\text{new}}^{(\leq r)}$.

(2025) ablate this decision and show slight improvement over last-layer features) and construct a normalized kernel using the inner product of these embeddings. Intuitively, frames whose losses suggest similar parameter updates have high kernel-similarity. For a task with more than one previously collected demonstration, we evaluate each demonstration separately. Using the loss-gradient kernel, we compute how the covariance would change if we received information equivalent to that demonstration. This reduces uncertainty at its frames and at other frames whose gradient embeddings are correlated with them. We aggregate the remaining uncertainty by summing frame variances within each demonstration, averaging across demonstrations within each task, and averaging

across tasks equally. We then greedily select the task whose demonstration leaves the lowest remaining total uncertainty until the demonstration budget n_e is reached and perform the same training steps as with the other methods.

Sample Efficiency Computation. The goal of active fine-tuning is to improve performance with as little data as possible. To quantify this capability, we measure *sample efficiency* as follows. Denote the success rate of method A after round r_1 by SR_{A,r_1}. Let $r_2 = \min\{r : \mathrm{SR}_{B,r} \geq \mathrm{SR}_{A,r_1}\}$ be the number of rounds required for method B to reach at least a similar success rate as SR_{A,r_1}. Then, if $r_2 < r_1$, method B requires $\frac{r_1 - r_2}{r_1} \times 100\%$ fewer samples to achieve the same performance, i.e., method B is $(\frac{r_1}{r_2} - 1) \times 100\%$ more sample-efficient than method A. To obtain robust results, we average across $r_1 \in \{R-2, R-1, R\}$. With this evaluation protocol, we find that with SmolVLA, SAVE w/ VFD requires 50 % fewer samples than Random, 45 % fewer samples than Diversity, 40 % fewer samples than SAVE w/ Action-L2, and 22 % fewer samples than SAVE w/ GU, corresponding to 101 %, 83 %, 68 %, and 29 % higher sample efficiency, respectively.

B.5 Failure Detection

For our experiments, we build on the failure prediction framework FIPER (Römer et al., 2025), released under the MIT License.

B.5.1 Environment Details

Table 7 provides details about the rollouts in simulation and the real world. Unless otherwise stated, VFD uses a two-member ensemble consisting of the policy being rolled out and evaluated and a second, independently trained model.

Table 7: Details about the failure detection experiments.

Environment	`Push-T`	`LIBERO-Plus`	Real world
Policy	FM-Policy	SmolVLA	SmolVLA
Action chunk length H	16	50	30
Action execution steps h	8	25	15
Tasks	1	10	3
Calibration rollouts per task	20	20	10
Test rollouts	200	1,800	152
Policy success rate	0.20	0.49	0.56
Timesteps per rollout	50–300	140–520	160–500

`Push-T`. We use the ten FM-Policy ensembles trained on 50 of the 206 demonstrations (Section B.3.1). Since `Push-T` involves no distribution shift, the environment tests the capabilities of failure detectors in in-distribution settings.

`LIBERO-Plus`. The policies are trained on all demonstrations of the `LIBERO-Spatial`, `-Object`, `-Goal`, and `LIBERO-Long` suites. The perturbations reduce the success rate of our SmolVLA policies from 0.79 on the unperturbed `LIBERO-Long` tasks to 0.34 under camera perturbations; 0.37 under sensor noise; 0.39 when perturbing the robot initial state; 0.55 under background perturbations; 0.60 under lighting changes; 0.64 with changed object layout; 0.06 for language perturbations. Due to its strong imbalance of successes and failures, we do not evaluate failure detection on the language perturbation family. For each task and perturbation family, we record 30 test rollouts, one per perturbation variant, which yields 1,800 test rollouts. To calibrate the thresholds, we use 20 successful rollouts per task, which are generated by rolling out the same policy on the unperturbed `LIBERO-Long` tasks.

Real world. We fine-tune SmolVLA together on all three tasks `Place`, `Stack`, and `Drawer`, using 100 demonstrations per task. For each task, we record approximately 50 test rollouts with novel background objects, yielding 152 rollouts, of which 85 succeed. We assess robustness with respect to the calibration data by repeating the evaluation for three independent calibration draws, each using 10 successful in-distribution rollouts per task.

B.5.2 VFD Score for Failure Detection

Since runtime monitoring requires low latency, we use the failure detection experiments to investigate whether the computational cost of VFD can be reduced without substantially decreasing performance. To this end, we modify the VFD score (7) in two ways. First, we only evaluate the velocity fields along the ODE trajectories of the acting policy. These trajectories are integrated anyway to generate actions, and the policy's own velocities at their intermediate states are computed as part of this integration, so VFD only adds evaluations of the other ensemble members' velocity fields at those states. Second, we only evaluate the velocity difference at a subset $\mathcal{S} = \{s_1 < \cdots < s_{N_\mathrm{d}}\} \subseteq \{0, \delta s, \ldots, 1 - \delta s\}$ of N_d flow times, weighted by the spacing between them, instead of at all $N_s = 1/\delta s$ integration steps. Specifically, we compute the VFD score as

$$u_\mathrm{e}(\boldsymbol{y}) = \frac{1}{(M-1)\,B} \sum_{m=2}^{M} \sum_{b=1}^{B} \sum_{k=1}^{N_\mathrm{d}} \kappa_{s_k}\, \Delta s_k \left\| \boldsymbol{v}_{s_k}^{\boldsymbol{\theta}_1}\left(\boldsymbol{x}_{s_k}^{(b)}, \boldsymbol{y}\right) - \boldsymbol{v}_{s_k}^{\boldsymbol{\theta}_m}\left(\boldsymbol{x}_{s_k}^{(b)}, \boldsymbol{y}\right) \right\|_2^2, \tag{42}$$

where $\boldsymbol{x}_s^{(b)}$ is the state of the b-th action chunk at flow time s along the policy's own integration, $\Delta s_k = s_{k+1} - s_k$ for $k < N_\mathrm{d}$, $\Delta s_{N_\mathrm{d}} = 1 - s_{N_\mathrm{d}}$, and $\kappa_s = \frac{s}{1-s}$ as defined in Section 4. We investigate ensemble sizes up to $M = 4$.

B.5.3 Evaluation Protocol

Threshold computation via conformal prediction. For each task, we set a constant threshold γ to the empirical q-quantile of the maximum score within each calibration rollout (Agia et al., 2024), with $q = 0.9$ for all methods and environments. A test rollout is flagged as a failure at the first action-generation timestep at which its score exceeds γ. This corresponds to split conformal prediction (Angelopoulos & Bates, 2023) with the rollout-wise maximum score as the nonconformity score.

Metrics. Let N_fail and N_succ denote the number of failed and successful test rollouts of a task. The true positive rate (TPR) is the fraction of failed rollouts that are flagged, the true negative rate (TNR) the fraction of successful rollouts that are not flagged, and the balanced accuracy is $\mathrm{Acc} = \frac{1}{2}(\mathrm{TPR} + \mathrm{TNR})$. For a correctly flagged failed rollout n, the detection time $\mathrm{DT}_n = t_n^*/(T_\mathrm{max} - 1)$ is the timestep t_n^* of the first flagged step, normalized by the length T_max of the longest rollout of the respective task, and DT denotes its average over all correctly flagged failures. If no failure is flagged, $\mathrm{DT} = 1$. Timestep-wise accuracy (Römer et al., 2025) rewards earlier detections and is calculated as $\mathrm{TWA} = \frac{1}{2}\left(\frac{1}{N_\mathrm{fail}} \sum_{n \in \mathcal{T}} \left(1 - \mathrm{DT}_n\right) + \mathrm{TNR}\right)$, where $\mathcal{T}$ is the set of correctly flagged failed rollouts. Hence, $\mathrm{TWA} \leq \mathrm{Acc}$, with equality if every correctly detected failure is flagged at the first step. In Tables 3, 12, 13 and 14, bold and underlined values denote the best and second-best method per environment and metric. A detector that raises alarms on very few or instantly on all rollouts can attain a low detection time without being useful. Therefore, we rank DT only among methods with $\mathrm{TPR} \geq 0.4$ and $\mathrm{TNR} \geq 0.4$, applied to unrounded values (e.g., the real-world TNR of RND-OE (0.398) is displayed as 0.40 but excluded from the DT ranking). TPR and TNR are not ranked.

B.5.4 Baselines

We evaluate all methods on the same rollouts and using the same procedure to calculate conformal prediction thresholds.

STAC (Agia et al., 2024) measures the temporal consistency of the action chunk distribution. At each policy inference step $0, h, 2h, \ldots$, it compares the action chunks $\boldsymbol{A}_t$ sampled at the current step with those sampled at the previous step $\boldsymbol{A}_{t-h}$ over their temporally overlapping parts of length $H - h$. The score is the maximum mean discrepancy between the two sample sets using an RBF kernel and computed only over the action dimensions corresponding to positions. STAC uses a batch size of 128 for `Push-T` and 32 for `LIBERO-Plus` and the real-world experiments.

logpZO (Xu et al., 2025) fits an FM model to the policy's observation embeddings and scores an embedding $\boldsymbol{e}_t$ by the squared norm $\|\boldsymbol{z}_t\|_2^2$ of the noise $\boldsymbol{z}_t = \boldsymbol{e}_t - \boldsymbol{v}_1(\boldsymbol{e}_t)$ that one backward Euler step of the learned velocity field maps it to. Embeddings far from the training distribution map to

noise of large norm. Following the authors, the velocity field is parameterized as a 1D convolutional U-Net with channel dimensions $(256, 512, 1024)$, kernel size 5, 8 groups, and a 128-dimensional time embedding, trained for 200 epochs with learning rate 10^{-4} and batch size 128.

RND-OE (Römer et al., 2025) applies random network distillation to the observation embeddings. A predictor network is trained to match the output of a fixed, randomly initialized target network, and the failure detection score is defined as the prediction error between the two networks' outputs. Following the authors, we train the predictor with an ℓ_2 loss using AdamW with learning rate 10^{-4}, cosine decay to 10^{-6}, weight decay 10^{-5}, and batch size 256 for up to 250 epochs, with early stopping on a held-out $10\,\%$ validation split.

ACE (Römer et al., 2025) is computed from the same position-only action samples as STAC and described in Section B.3.2.

We do not consider SAFE (Gu et al., 2025) as a baseline because it requires failure data for training. We deliberately calibrate the thresholds only on successful rollouts, which are easier to obtain.

For a fair comparison with VFD, we train logpZO and RND-OE on the same data used to train the policy ensemble. In `Push-T`, we use the 50 demonstrations of the ensemble, and the embedding is the 66-dimensional observation conditioning vector of the FM-Policy. For SmolVLA, the embedding consists of the flattened keys and values from the KV cache of the middle VLM layer over the prefix tokens. In the real world, we fit one detector per task on its 100 demonstrations. In `LIBERO-Plus`, we fit one detector per policy on its training demonstrations, which are subsampled with a stride of 10 for computational reasons. Since the number of instruction tokens varies across tasks, we zero-pad the embeddings after the instruction block to the length of the longest instruction, such that a single detector can cover all tasks.

B.5.5 Computational Cost

Table 4 reports results for the SmolVLA policy used in our real-world experiments. The second row corresponds to the default VFD configuration used in Table 3. We measure the wall-clock inference time per action-generation step on a single NVIDIA RTX 5090 GPU, averaged over 60 repetitions. The base policy, i.e., one VLM prefix pass that computes the KV cache followed by 10 Euler integration steps of the action expert, takes $170\,\mathrm{ms}$ to generate a single action chunk. VFD integrates a batch of B action chunks and, for each of the $M-1$ other ensemble members, requires an additional prefix pass and evaluating that member's velocity field at the N_d ODE states of the base policy for the flow times in $\mathcal{S}$. On this GPU and at $B = 8$, a prefix pass takes approximately $25\,\mathrm{ms}$ and a velocity-field evaluation approximately $14\,\mathrm{ms}$. The inference time column reports the added time per action-generation step, and the relative column reports the total inference time relative to the base policy. We find that using a larger ensemble increases balanced accuracy by $5\,\mathrm{pp}$, albeit at the cost of $162\,\mathrm{ms}$ longer inference.

Weight perturbation as an alternative to ensembling. To assess whether VFD can avoid training an ensemble altogether, we replace the second ensemble member with a perturbed version of the policy itself. To this end, we add independent zero-mean Gaussian noise with standard deviation $\sigma\,\mathrm{std}(\boldsymbol{W})$ to each weight tensor $\boldsymbol{W}$, i.e., scaled to the spread of the tensor's own values. This corresponds to a single sample from a Gaussian posterior centered at the trained weights with a heuristic, tensor-wise diagonal covariance. We draw the perturbation once with a fixed random seed and keep it fixed for all rollouts and steps, including the calibration rollouts, so that the thresholds are calibrated for the same perturbed model that is evaluated on the test rollouts. We perturb the VLM backbone (vision encoder and language model, 460.2M parameters) and the action expert, including its action and time projections (97.4M parameters). The language-modeling head of the VLM is not used for action generation and remains unchanged. We evaluate the perturbed model at the recorded ODE states of the policy, at a single flow time $s = 0.8$ and with $B = 1$. We select σ from combinations of separate noise scales $\sigma_\mathrm{expert}, \sigma_\mathrm{VLM} \in \{0, 0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 0.5\}$ for the two modules. Perturbing only the action expert, which would allow the perturbed model to reuse the policy's KV cache and reduce the relative inference time to $1.08\times$, yields a balanced accuracy of at most 0.62. Balanced accuracy plateaus at 0.66 to 0.68 for $\sigma_\mathrm{VLM} \in [0.1, 0.2]$ and $\sigma_\mathrm{expert} \in [0.05, 0.2]$, and drops below 0.46 for $\sigma_\mathrm{VLM} = 0.5$. Because the VLM is perturbed, the perturbed model requires its own prefix pass, such that the inference time matches that of using an ensemble. In summary, this

experiment indicates that VFD can yield competitive uncertainty estimates even without training a deep ensemble.

C Additional Results

C.1 Toy Example

We illustrate the capabilities of VFD for quantifying epistemic uncertainty in FM models with a simple example.

Example 1. Consider a conditional generative modeling problem with $\boldsymbol{x} = (x_1, x_2)$, $\boldsymbol{y} = (y_1, y_2) \in \mathbb{R}^2$. The ground-truth conditional distribution is a two-component Gaussian mixture with input-dependent means; $q(\boldsymbol{x} \mid \boldsymbol{y}) = 0.5\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_1(\boldsymbol{y}), \boldsymbol{I}) + 0.5\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_2(\boldsymbol{y}), \boldsymbol{I})$, where $\boldsymbol{\mu}_1(\boldsymbol{y}) = (\sin(\pi y_1) + 0.5, \cos(\pi y_2) - 0.5)$ and $\boldsymbol{\mu}_2(\boldsymbol{y}) = (\cos(2\pi y_2) - 0.5, \sin(0.5\pi y_1) + 0.5)$. We draw a set of conditioning inputs $\{\boldsymbol{y}^{(n)}\}_{n=1}^N$ with $N = 4000$ from a uniform distribution over an annulus with inner radius 0.75 and outer radius 1.25 and sample $\boldsymbol{x}^{(n)} \sim q(\boldsymbol{x} \mid \boldsymbol{y}^{(n)})$. We parameterize the velocity field using an MLP and train an ensemble of $M = 5$ FM models with a batch size of 256 for 300 epochs using the standard loss (3). As shown in Figure 8, our VFD score assigns regions far from the training data a high level of epistemic uncertainty, which closely resembles the average KL divergence $\frac{1}{M}\sum_{i=1}^M D_{\mathrm{KL}}(p^{\boldsymbol{\theta}_i} \,||\, q)$ between the learned distributions and the ground truth.

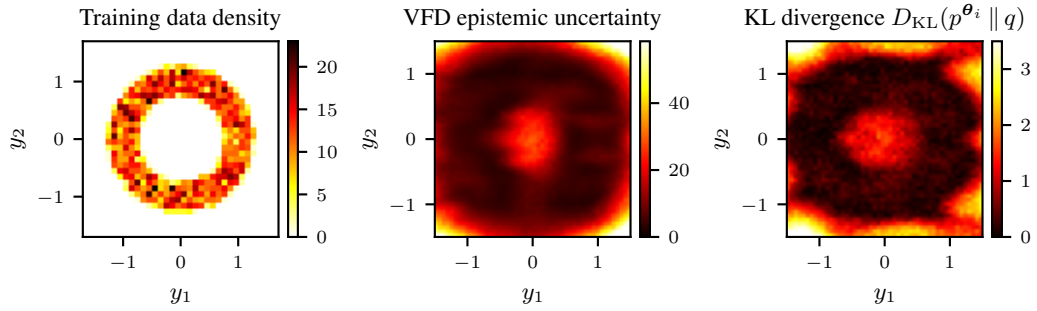

Figure 8: **VFD for a 2D toy example.** We visualize epistemic uncertainty estimation for a 2D generative modeling problem. Our VFD uncertainty score is high for inputs far from the training distribution, similar to the KL divergence between the learned models' conditional distributions and the ground truth.

C.2 Calibration

Table 8 is an extension of Table 1 and also reports the Pearson correlation coefficient, measuring the strength of the linear relationship between per-task uncertainties and success rates.

C.2.1 Ensemble Size Ablation

Since fine-tuning generalist policies is expensive, keeping the ensemble small (or avoiding it entirely, cf. Table 4) is highly desirable. Figure 9 shows the impact of the ensemble size M on calibration for all fine-tuning rounds of SmolVLA on `LIBERO-Long`. The VLA ensembles consistently remain well-calibrated even for $M = 2$, indicating that they do not collapse to a single mode of the posterior. Due to the competitive performance and computational advantage of two-member ensembles, we adopt this ensemble size in the experiments unless stated otherwise.

C.2.2 Ensembling vs. Laplace Approximation

We compare training an ensemble to computing the Laplace approximation (Daxberger et al., 2021) for sampling from the model posterior. Since computing the full Laplace approximation for the

Table 8: **Calibration analysis – extended.** Negative Spearman's rank correlation (↑) and negative Pearson correlation (↑) between uncertainty estimates and per-task success rates in `LIBERO-Long`, averaged across iterative fine-tuning rounds. VFD offers superior calibration compared to other uncertainty estimation approaches.

Policy	Metric	Action-L2	ACE	DECU	GU	Entropy	Perplexity	**VFD (ours)**
SmolVLA	−Spearman	$0.50^{\pm 0.13}$	$0.31^{\pm 0.12}$	$0.31^{\pm 0.13}$	$0.62^{\pm 0.00}$	$0.10^{\pm 0.12}$	$-0.04^{\pm 0.09}$	$\mathbf{0.71}^{\pm 0.03}$
	−Pearson	$0.48^{\pm 0.09}$	$0.36^{\pm 0.08}$	$0.23^{\pm 0.15}$	$0.65^{\pm 0.02}$	$0.23^{\pm 0.21}$	$0.02^{\pm 0.15}$	$\mathbf{0.71}^{\pm 0.02}$
X-VLA	−Spearman	$0.33^{\pm 0.24}$	$0.14^{\pm 0.05}$	$0.06^{\pm 0.04}$	$0.48^{\pm 0.06}$	N/A	N/A	$\mathbf{0.53}^{\pm 0.02}$
	−Pearson	$0.33^{\pm 0.22}$	$0.15^{\pm 0.07}$	$0.06^{\pm 0.05}$	$0.45^{\pm 0.13}$	N/A	N/A	$\mathbf{0.48}^{\pm 0.08}$
FastWAM	−Spearman	$-0.12^{\pm 0.06}$	$-0.29^{\pm 0.07}$	$0.03^{\pm 0.07}$	$0.27^{\pm 0.04}$	N/A	N/A	$\mathbf{0.44}^{\pm 0.17}$
	−Pearson	$-0.03^{\pm 0.14}$	$-0.20^{\pm 0.07}$	$0.06^{\pm 0.05}$	$0.29^{\pm 0.06}$	N/A	N/A	$\mathbf{0.42}^{\pm 0.12}$

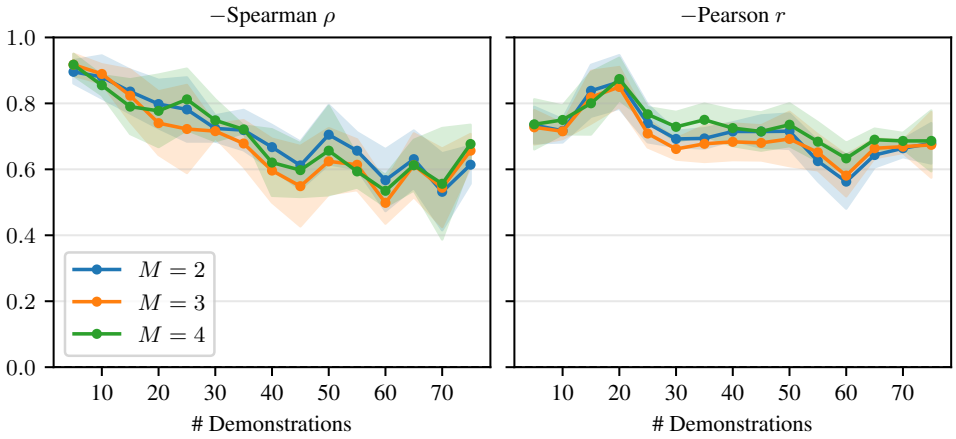

Figure 9: **Ensemble size ablation.** During active fine-tuning, the number of ensemble models has little effect on the calibration of VFD.

entire SmolVLA model (550M) is computationally intractable, we resort to the last-layer Laplace approximation, which is also used by Jazbec et al. (2025) to compute GU in the original work. We fit a diagonal last-layer Laplace approximation with `laplace-torch` (Daxberger et al., 2021) around each trained SmolVLA checkpoint. Concretely, we freeze the full model and place the posterior only on the final action-projection layer, which maps the action expert's hidden states to the predicted FM action velocity. The Laplace posterior is fit as a regression model on the same training episodes used for that active-learning round: for each calibration frame, we sample FM time s, noise $\boldsymbol{x}_0 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, and target action $\boldsymbol{x}_1$, and use the corresponding velocity $\boldsymbol{x}_1 - \boldsymbol{x}_0$ as the regression target. We use all selected training episodes for the round, a calibration fraction of 1.0, and a diagonal Hessian approximation.

Even when limited to the last layer, the computational cost of the last-layer Laplace approximation remains considerable, requiring about 5 hours per checkpoint per round on an NVIDIA RTX 4090 GPU. This is much higher than the 30 minutes it takes to fine-tune a second ensemble member for 4,000 steps on the same GPU. Due to the high cost, we only fit the Laplace approximation for the first 5 rounds. As shown in Figure 10, the two-member ensemble provides much better VFD score calibration than the Laplace approximation despite being faster to train.

C.3 Learning Curves of Active Fine-Tuning

In Figure 11, we provide the full learning curves in our simulation experiments, corresponding to the results reported in Tables 2 and 10. Figure 12 shows the active fine-tuning performance for each of the three real-world tasks.

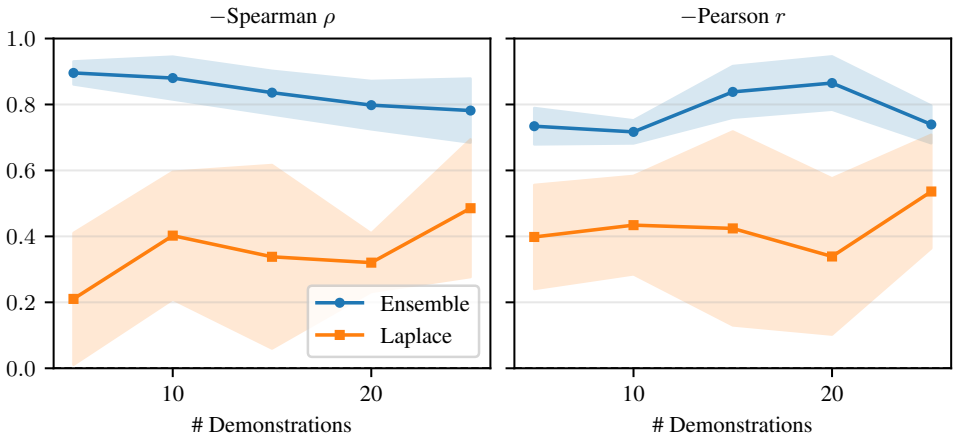

Figure 10: **Ensembling vs. Laplace Approximation.** Calibration is measured by the negative Spearman rank correlation (left) and the negative Pearson correlation (right) between the average task success rate and VFD uncertainty. Two separate ensemble members achieve much better calibration than sampling a second model from the last-layer Laplace approximation.

Table 9: **Statistical significance on `Push-T`.** Mean and sample standard deviation over ten paired training seeds.

Metric	SAVE w/ VFD	SAVE w/ GU	Difference	VFD better	p_{Wilcoxon}	$p_{t\text{-test}}$
AULC	$46.6^{\pm 2.5}$	$44.7^{\pm 2.9}$	$+1.9\ [+0.8, +3.2]$	9/10	0.006	0.010
Last-3 SR	$59.3^{\pm 3.1}$	$54.3^{\pm 3.3}$	$+5.0\ [+2.8, +7.1]$	9/10	0.004	0.001

C.4 Statistical Significance of Uncertainty-Guided Data Acquisition

The active fine-tuning experiments with FM-Policy on `Push-T` require far fewer computational resources than those with the three generalist policies. Therefore, we can repeat experiments across ten seeds in this environment to conduct a statistical test. Specifically, we assess whether the advantage of VFD over the strongest uncertainty-based baseline, GU, observed in all environments, is statistically robust. We compare the two methods in a paired manner: For each seed, both start from the same pre-trained ensemble and use the same training seed. We test the one-sided hypothesis that SAVE w/ VFD achieves higher AULC and Last-3 SR than SAVE w/ GU using the Wilcoxon signed-rank test and the paired t-test. As shown in Table 9, SAVE w/ VFD outperforms SAVE w/ GU in nine out of ten seeds on both metrics, and the improvement is significant at the $1\,\%$ level for both metrics under the Wilcoxon test. These conclusions also hold for two-sided tests at a $2\,\%$ level and after a Holm correction for testing two metrics.

C.5 Iterative Fine-Tuning vs. Retraining

We also test whether our iterative active-learning loop, designed to minimize data-collection effort, requires an additional separate full fine-tuning stage to yield optimal policy performance. To this end, we compare the final-round SmolVLA policy π_{15} to a policy retrained from the base checkpoint using a 50/50 split between the ensemble training data $\mathcal{D}_{\text{pre}}$ and the actively selected data $\mathcal{D}_{\text{new}}^{(\leq 15)}$. When using the demonstrations selected at the best-performing temperature $\tau = 2.5$, the retrained policy achieves an overall success rate of $70.6\%^{\pm 0.9}$, which is only 3.5 pp higher than that of the iteratively fine-tuned policy. This demonstrates that uncertainty-guided acquisition produces a strong policy already during the iterative active learning process.

C.6 Hyperparameter Selection for Uncertainty-Guided Active Fine-Tuning

Figure 13 (left) shows that most of the gain of SAVE w/ VFD over random acquisition stems from the prioritization at the task level, while uncertainty-based initial observation selection yields a minor

Table 10: **Sample efficiency.** Number of demonstrations to reach certain success rates (SR), area under the learning curve (AULC) (↑), and Last-3 SR (↑) for different active fine-tuning strategies. We report the six highest SR thresholds that are multiples of 5 % and reached by at least one method. Thresholds not reached within the maximum demonstration budget are marked as "N/R".

`Push-T`: FM-Policy

SR Threshold	Random	Diversity	SAVE w/ Action-L2	SAVE w/ GU	**SAVE w/ VFD**
$\geq 30\,\%$	$30.0^{\pm 15.3}$	$21.1^{\pm 11.4}$	$17.5^{\pm 13.1}$	$15.6^{\pm 8.8}$	$\mathbf{15.0}^{\pm 8.2}$
$\geq 35\,\%$	$47.5^{\pm 27.7}$	$36.7^{\pm 11.5}$	$29.4^{\pm 17.4}$	$29.4^{\pm 12.6}$	$\mathbf{28.5}^{\pm 13.1}$
$\geq 40\,\%$	$71.5^{\pm 21.9}$	$51.7^{\pm 16.0}$	$48.1^{\pm 18.9}$	$\mathbf{38.9}^{\pm 12.2}$	$43.0^{\pm 8.9}$
$\geq 45\,\%$	$98.5^{\pm 21.0}$	$86.7^{\pm 22.4}$	$80.0^{\pm 38.7}$	$61.7^{\pm 13.9}$	$\mathbf{58.0}^{\pm 14.0}$
$\geq 50\,\%$	N/R	$109.4^{\pm 20.4}$	N/R	$83.3^{\pm 12.2}$	$\mathbf{78.5}^{\pm 16.3}$
$\geq 55\,\%$	N/R	N/R	N/R	$105.0^{\pm 24.2}$	$\mathbf{96.0}^{\pm 16.3}$
AULC	$38.0^{\pm 3.2}$	$40.5^{\pm 1.9}$	$39.8^{\pm 3.4}$	$44.7^{\pm 2.9}$	$\mathbf{46.6}^{\pm 2.5}$
Last-3 SR	$46.3^{\pm 3.1}$	$51.2^{\pm 3.2}$	$45.0^{\pm 2.6}$	$54.3^{\pm 3.3}$	$\mathbf{59.3}^{\pm 3.1}$

`LIBERO-Long`: SmolVLA

SR Threshold	Random	Diversity	AMF	SAVE w/ Action-L2	SAVE w/ GU	**SAVE w/ VFD**
$\geq 40\,\%$	$28.3^{\pm 7.6}$	$\mathbf{23.3}^{\pm 2.9}$	$30.0^{\pm 5.0}$	$25.0^{\pm 5.0}$	$25.0^{\pm 8.7}$	$25.0^{\pm 5.0}$
$\geq 45\,\%$	$33.3^{\pm 7.6}$	$36.7^{\pm 5.8}$	$38.3^{\pm 5.8}$	$30.0^{\pm 8.7}$	$28.3^{\pm 14.4}$	$\mathbf{26.7}^{\pm 5.8}$
$\geq 50\,\%$	$43.3^{\pm 12.6}$	$48.3^{\pm 10.4}$	$46.7^{\pm 7.6}$	$45.0^{\pm 15.0}$	$36.7^{\pm 11.5}$	$\mathbf{30.0}^{\pm 5.0}$
$\geq 55\,\%$	N/R	N/R	$53.3^{\pm 2.9}$	$45.0^{\pm 7.1}$	$55.0^{\pm 10.0}$	$\mathbf{40.0}^{\pm 10.0}$
$\geq 60\,\%$	N/R	N/R	$66.7^{\pm 10.4}$	N/R	$63.3^{\pm 10.4}$	$\mathbf{50.0}^{\pm 5.0}$
$\geq 65\,\%$	N/R	N/R	N/R	N/R	N/R	$\mathbf{62.5}^{\pm 3.5}$
AULC	$44.6^{\pm 2.2}$	$45.5^{\pm 1.8}$	$45.6^{\pm 3.3}$	$47.2^{\pm 5.1}$	$48.6^{\pm 5.0}$	$\mathbf{52.3}^{\pm 2.2}$
Last-3 SR	$52.6^{\pm 3.3}$	$53.1^{\pm 2.0}$	$59.4^{\pm 3.3}$	$55.5^{\pm 6.8}$	$61.0^{\pm 2.9}$	$\mathbf{66.1}^{\pm 2.8}$

`LIBERO-Long`: X-VLA

SR Threshold	Random	Diversity	AMF	SAVE w/ Action-L2	SAVE w/ GU	**SAVE w/ VFD**
$\geq 25\,\%$	$18.3^{\pm 2.9}$	$15.0^{\pm 0.0}$	$18.3^{\pm 2.9}$	$16.7^{\pm 2.9}$	$13.3^{\pm 2.9}$	$\mathbf{11.7}^{\pm 2.9}$
$\geq 30\,\%$	$21.7^{\pm 7.6}$	$20.0^{\pm 0.0}$	$23.3^{\pm 7.6}$	$21.7^{\pm 2.9}$	$16.7^{\pm 7.6}$	$\mathbf{15.0}^{\pm 5.0}$
$\geq 35\,\%$	$26.7^{\pm 7.6}$	$\mathbf{21.7}^{\pm 2.9}$	$30.0^{\pm 5.0}$	$25.0^{\pm 0.0}$	$26.7^{\pm 5.8}$	$23.3^{\pm 2.9}$
$\geq 40\,\%$	$40.0^{\pm 0.0}$	$\mathbf{33.3}^{\pm 7.6}$	$36.7^{\pm 2.9}$	$35.0^{\pm 5.0}$	$35.0^{\pm 10.0}$	$\mathbf{33.3}^{\pm 5.8}$
$\geq 45\,\%$	$50.0^{\pm 13.2}$	$50.0^{\pm 15.0}$	$51.7^{\pm 2.9}$	$56.7^{\pm 7.6}$	$51.7^{\pm 10.4}$	$\mathbf{43.3}^{\pm 2.9}$
$\geq 50\,\%$	N/R	N/R	$65.0^{\pm 10.0}$	$67.5^{\pm 3.5}$	$57.5^{\pm 17.7}$	$\mathbf{55.0}^{\pm 0.0}$
AULC	$37.7^{\pm 1.2}$	$40.2^{\pm 1.9}$	$39.5^{\pm 1.9}$	$39.2^{\pm 0.9}$	$40.7^{\pm 3.5}$	$\mathbf{42.5}^{\pm 1.0}$
Last-3 SR	$45.1^{\pm 0.3}$	$49.0^{\pm 3.3}$	$51.0^{\pm 1.3}$	$50.9^{\pm 2.2}$	$49.2^{\pm 1.8}$	$\mathbf{52.9}^{\pm 2.9}$

`LIBERO-Long`: FastWAM

SR Threshold	Random	Diversity	AMF	SAVE w/ Action-L2	SAVE w/ GU	**SAVE w/ VFD**
$\geq 45\,\%$	$20.0^{\pm 5.0}$	$21.7^{\pm 2.9}$	$25.0^{\pm 0.0}$	$20.0^{\pm 5.0}$	$\mathbf{16.7}^{\pm 2.9}$	$18.3^{\pm 7.6}$
$\geq 50\,\%$	$26.7^{\pm 2.9}$	$30.0^{\pm 0.0}$	$28.3^{\pm 2.9}$	$26.7^{\pm 2.9}$	$26.7^{\pm 5.8}$	$\mathbf{21.7}^{\pm 7.6}$
$\geq 55\,\%$	$38.3^{\pm 2.9}$	$33.3^{\pm 2.9}$	$33.3^{\pm 2.9}$	$31.7^{\pm 2.9}$	$\mathbf{28.3}^{\pm 2.9}$	$\mathbf{28.3}^{\pm 10.4}$
$\geq 60\,\%$	$53.3^{\pm 18.9}$	$56.7^{\pm 10.4}$	$38.3^{\pm 2.9}$	$40.0^{\pm 8.7}$	$35.0^{\pm 5.0}$	$\mathbf{33.3}^{\pm 5.8}$
$\geq 65\,\%$	$62.5^{\pm 17.7}$	$75.0^{\pm 0.0}$	$48.3^{\pm 15.3}$	$58.3^{\pm 2.9}$	$\mathbf{41.7}^{\pm 5.8}$	$\mathbf{41.7}^{\pm 7.6}$
$\geq 70\,\%$	N/R	N/R	N/R	$60.0^{\pm 0.0}$	$57.5^{\pm 24.7}$	$\mathbf{56.7}^{\pm 5.8}$
AULC	$54.2^{\pm 2.5}$	$53.9^{\pm 0.8}$	$55.6^{\pm 1.4}$	$57.0^{\pm 3.3}$	$59.0^{\pm 4.4}$	$\mathbf{61.1}^{\pm 3.5}$
Last-3 SR	$64.8^{\pm 6.6}$	$63.9^{\pm 0.3}$	$69.4^{\pm 1.3}$	$71.6^{\pm 5.2}$	$70.6^{\pm 6.1}$	$\mathbf{74.8}^{\pm 3.3}$

additional improvement. A key reason for this performance gain is shown in the right of Figure 13 and in Figure 15. While pre-trained policies naturally exhibit varying levels of uncertainty across tasks, our uncertainty-guided active fine-tuning strategy can reduce these differences and improve multitask performance more quickly by allocating the demonstration budget to tasks that require it most.

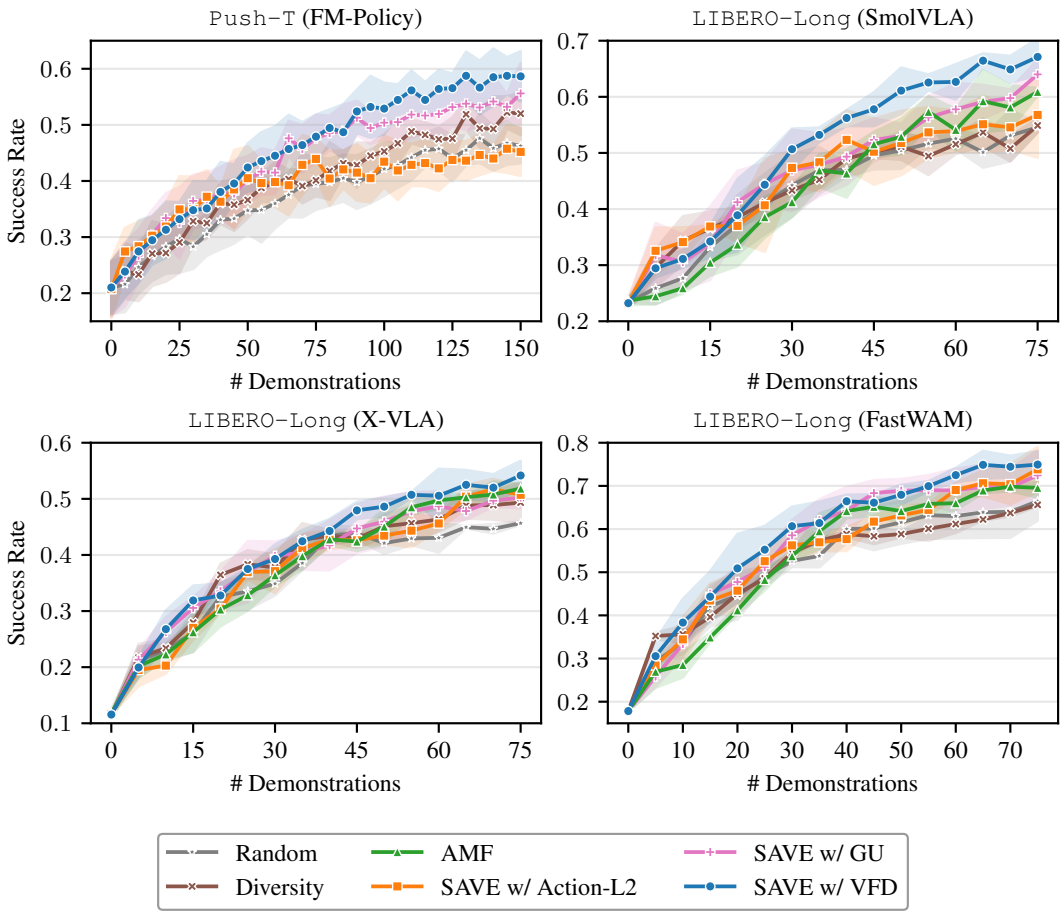

Figure 11: **Active learning curves.** These correspond to the active fine-tuning results reported in Tables 2 and 10. SAVE w/ VFD consistently yields the largest performance gains with the available demonstration budget.

Figure 14 complements this observation, showing a detailed view of the trade-off between exploiting the uncertainty estimates and exploring the task space evenly. The diversity of the categorical task distribution can be measured by its entropy. Over multiple rounds, SAVE tends towards high-entropy categorical distributions over tasks (i.e., more diversity in the selection). We observe a Pareto front: SAVE w/ VFD can trade off this diversity (lower exploration) for faster improvement (higher exploitation) in multitask success by focusing on high-uncertainty tasks (cf. Figure 14).

C.7 Failure Detection

We provide the full results of our failure detection experiments in Tables 12, 13 and 14.

VFD evaluation flow times. We also examine the impact of evaluating the VFD score (42) not at every ODE state, but only at a subset of evaluation flow times $\mathcal{S} \subseteq \{0.2, 0.4, 0.6, 0.8\}$. As shown in Figure 16, later timesteps where the action chunks contain less noise are more informative for estimating epistemic uncertainty. For a large four-member ensemble, restricting the computation of VFD to a single flow timestep $s = 0.8$ does not reduce balanced accuracy. Together with the computational cost analysis in Table 4, this experiment indicates that VFD can be computed efficiently while still being a capable estimator of epistemic uncertainty.

Rollout Visualization. Figures 17 and 18 visualize exemplary real-world rollouts, with the failure detection scores normalized for readability. Generally, we observe that the observation-based



Figure 12: **Real-world active fine-tuning results**. SAVE w/ VFD consistently improves across all tasks, whereas Diversity and SAVE w/ GU stagnate on certain tasks. Notably, random data acquisition is the second-best method on hardware, highlighting the difficulty of effectively allocating demonstration budgets for multitask improvement. These per-task learning curves constitute the average shown in Figure 4.

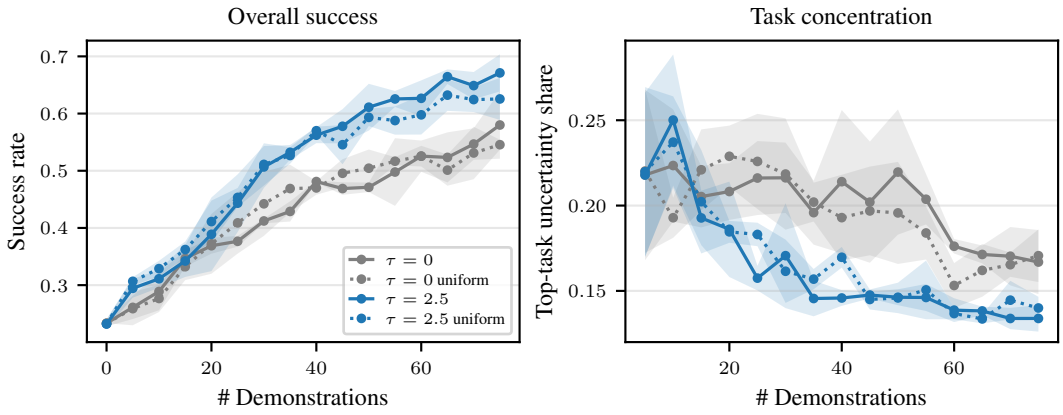

Figure 13: **Effect of the task-sampling temperature τ on SAVE**. Larger τ biases expert queries toward higher-uncertainty tasks, with uniform sampling for $\tau = 0$. **Left:** Uncertainty guidance is beneficial both for sampling tasks and initial observations. Legend entries containing *uniform* correspond to sampling initial observations within a sampled task uniformly instead of uncertainty-guided. **Right:** Uncertainty-based sampling more rapidly reduces the fraction of uncertainty concentrated in the most uncertain task, indicating better allocation of demonstrations to underperforming tasks.

Table 11: **Hyperparameter selection for the uncertainty-guided active fine-tuning methods.** Success rate (SR) in percent (↑) after different active fine-tuning rounds for SmolVLA on `LIBERO-Long`. The settings with the highest final success rate, reported in Tables 2 and 10, are marked with an asterisk (*).

Method	Hyperparameter	Round-5 SR	Round-10 SR	Round-15 SR
AMF	$\sigma = 10^{-4}$	$40.4^{\pm 4.1}$	$52.2^{\pm 4.4}$	$58.1^{\pm 3.1}$
	$\sigma = 10^{-3}$	$40.2^{\pm 4.0}$	$53.3^{\pm 3.7}$	$59.7^{\pm 6.7}$
	$\sigma = 10^{-2}$*	$38.6^{\pm 4.4}$	$52.9^{\pm 3.9}$	$60.9^{\pm 0.8}$
	$\sigma = 10^{-1}$	$39.6^{\pm 5.7}$	$52.3^{\pm 4.8}$	$57.6^{\pm 2.8}$
SAVE w/ Action-L2	$\tau = 0$	$40.6^{\pm 3.0}$	$51.1^{\pm 4.2}$	$54.0^{\pm 4.2}$
	$\tau = 1$	$41.4^{\pm 5.2}$	$46.9^{\pm 6.4}$	$52.2^{\pm 2.2}$
	$\tau = 1.5$*	$40.7^{\pm 10.1}$	$51.8^{\pm 7.0}$	$56.8^{\pm 9.3}$
	$\tau = 2$	$40.7^{\pm 11.0}$	$53.0^{\pm 7.0}$	$55.0^{\pm 3.2}$
	$\tau = 2.5$	$41.8^{\pm 7.0}$	$49.0^{\pm 9.0}$	$53.4^{\pm 6.9}$
SAVE w/ GU	$\tau = 0$	$40.3^{\pm 5.5}$	$49.6^{\pm 7.0}$	$53.6^{\pm 3.0}$
	$\tau = 1$*	$44.4^{\pm 4.6}$	$53.0^{\pm 2.7}$	$64.0^{\pm 3.2}$
	$\tau = 1.5$	$42.0^{\pm 1.5}$	$57.4^{\pm 5.3}$	$59.2^{\pm 6.7}$
	$\tau = 2$	$45.6^{\pm 3.7}$	$57.2^{\pm 4.4}$	$60.6^{\pm 3.5}$
	$\tau = 2.5$	$46.7^{\pm 4.7}$	$59.3^{\pm 4.2}$	$60.1^{\pm 2.7}$
SAVE w/ VFD	$\tau = 0$	$37.7^{\pm 1.2}$	$47.1^{\pm 1.0}$	$58.0^{\pm 7.2}$
	$\tau = 1$	$44.8^{\pm 6.6}$	$54.8^{\pm 3.3}$	$59.2^{\pm 4.8}$
	$\tau = 1.5$	$46.6^{\pm 2.0}$	$53.8^{\pm 1.7}$	$60.1^{\pm 4.6}$
	$\tau = 2$	$46.0^{\pm 6.7}$	$59.0^{\pm 2.6}$	$64.2^{\pm 3.7}$
	$\tau = 2.5$*	$44.3^{\pm 6.2}$	$61.1^{\pm 5.0}$	$67.1^{\pm 4.0}$

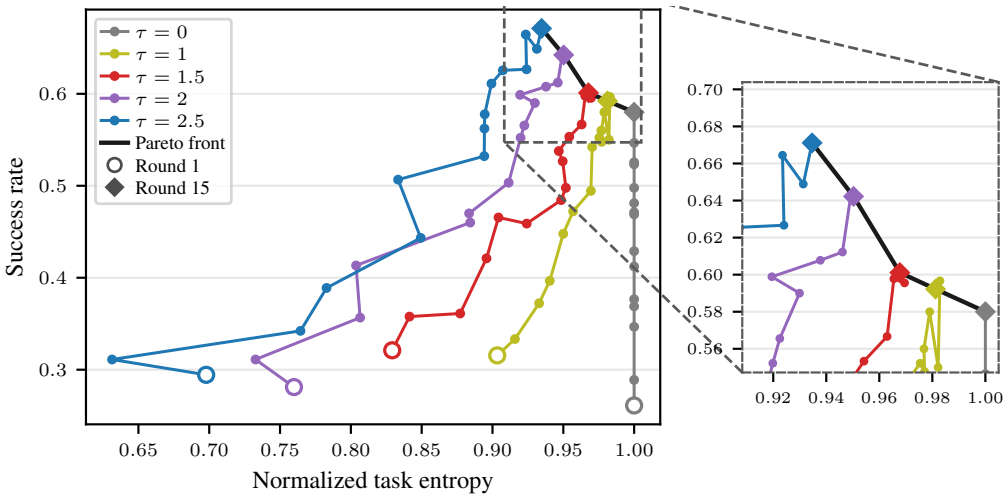

Figure 14: **Pareto front of the exploration-exploitation trade-off.** SAVE reduces the difference in prior knowledge about tasks; for $\tau \leq 2.5$, the temperature parameter controls an exploration–exploitation trade-off between task coverage and final success rate.

OOD detectors, logpZO and RND-OE, are prone to raising false alarms (TNR of 0.49 and 0.40, respectively), which happens in both rollouts shown in Figure 17. In contrast, STAC and ACE are unable to detect most failures (TPR of 0.04 and 0.22, respectively), see Figure 18. Compared to the baselines, VFD offers a middle ground, detecting 60 % of all failures while flagging only 13 % of all successful rollouts as failures and achieving the highest overall balanced accuracy of 73 %.

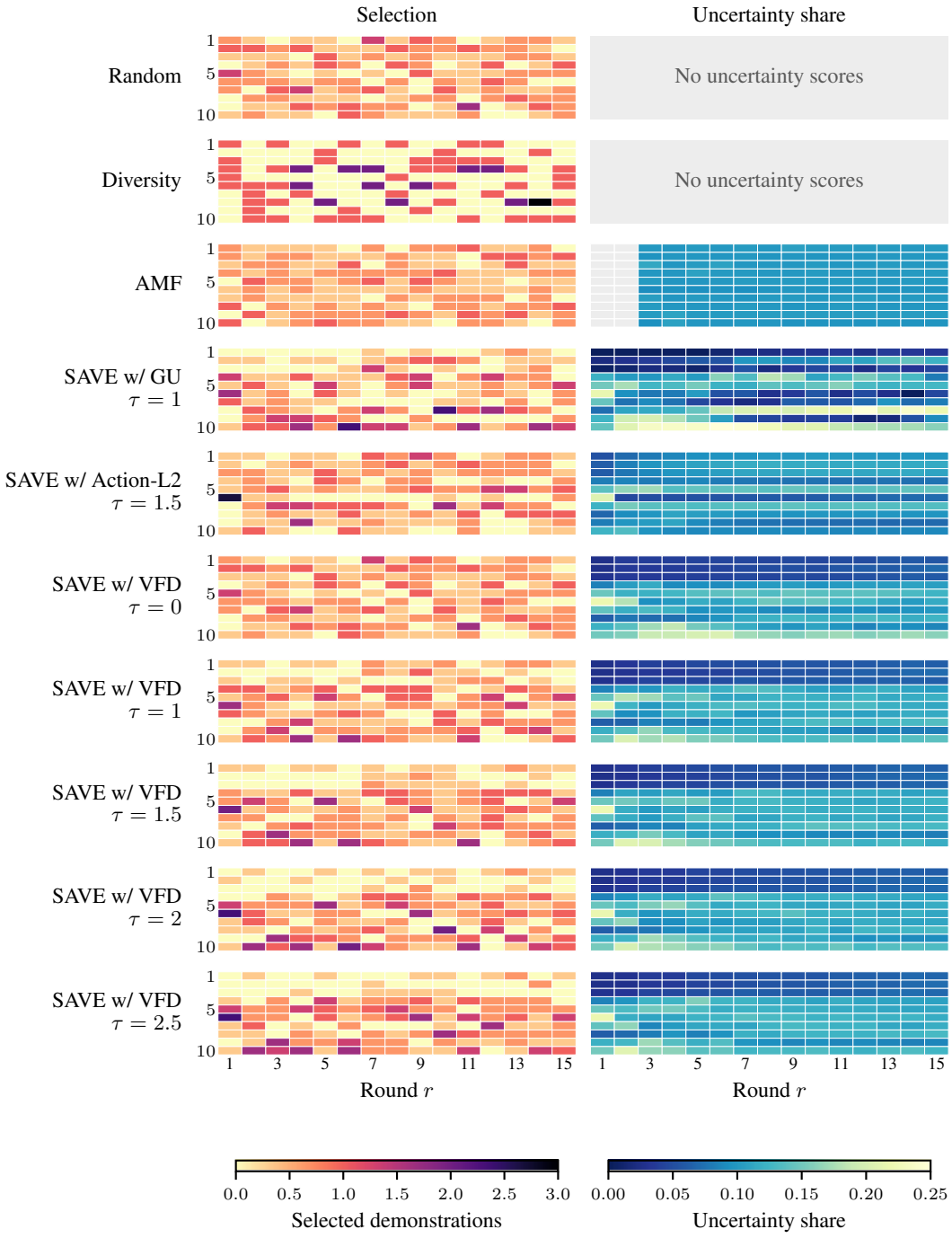

Figure 15: **Task selection**. Behavior of uncertainty-guided data acquisition in active fine-tuning for different temperature values τ over all active fine-tuning rounds with SmolVLA on `LIBERO-Long`. Left: Number of selected episodes per task. Right: Relative share of overall uncertainty per task.

Table 12: **Failure detection results in `Push-T`.** Mean and standard deviation over ten training seeds.

Metric	STAC	logpZO	RND-OE	ACE	**VFD (ours)**
Acc. ↑	$\underline{0.55}^{\pm 0.00}$	$0.53^{\pm 0.01}$	$0.50^{\pm 0.02}$	$\underline{0.55}^{\pm 0.06}$	$\mathbf{0.64}^{\pm 0.05}$
TWA ↑	$\underline{0.51}^{\pm 0.00}$	$\underline{0.51}^{\pm 0.02}$	$0.50^{\pm 0.02}$	$\underline{0.51}^{\pm 0.03}$	$\mathbf{0.56}^{\pm 0.05}$
DT ↓	$0.56^{\pm 0.04}$	$0.23^{\pm 0.09}$	$0.00^{\pm 0.00}$	$0.30^{\pm 0.04}$	$0.41^{\pm 0.05}$
TPR ↑	$0.13^{\pm 0.02}$	$0.18^{\pm 0.09}$	$0.15^{\pm 0.08}$	$0.23^{\pm 0.15}$	$0.38^{\pm 0.08}$
TNR ↑	$0.97^{\pm 0.02}$	$0.87^{\pm 0.07}$	$0.86^{\pm 0.12}$	$0.86^{\pm 0.05}$	$0.89^{\pm 0.07}$

Table 13: **Failure detection on `LIBERO-Plus`.** We group the results by perturbation family and provide the mean and standard deviation over three training seeds.

Perturbation	Metric	STAC	logpZO	RND-OE	ACE	**VFD (ours)**
Camera	Acc. ↑	$0.68^{\pm 0.06}$	$0.65^{\pm 0.02}$	$0.61^{\pm 0.02}$	$\mathbf{0.73}^{\pm 0.00}$	$\underline{0.72}^{\pm 0.05}$
	TWA ↑	$0.56^{\pm 0.04}$	$\mathbf{0.58}^{\pm 0.03}$	$\underline{0.57}^{\pm 0.02}$	$0.56^{\pm 0.01}$	$0.56^{\pm 0.06}$
	DT ↓	$\mathbf{0.32}^{\pm 0.02}$	$0.15^{\pm 0.02}$	$0.08^{\pm 0.01}$	$0.50^{\pm 0.06}$	$\underline{0.37}^{\pm 0.03}$
	TPR ↑	$0.72^{\pm 0.12}$	$0.99^{\pm 0.01}$	$1.00^{\pm 0.00}$	$0.73^{\pm 0.09}$	$0.90^{\pm 0.04}$
	TNR ↑	$0.63^{\pm 0.04}$	$0.32^{\pm 0.04}$	$0.22^{\pm 0.04}$	$0.74^{\pm 0.08}$	$0.54^{\pm 0.12}$
Robot init.	Acc. ↑	$0.70^{\pm 0.02}$	$0.74^{\pm 0.02}$	$0.72^{\pm 0.02}$	$\underline{0.76}^{\pm 0.04}$	$\mathbf{0.78}^{\pm 0.03}$
	TWA ↑	$0.61^{\pm 0.01}$	$\underline{0.68}^{\pm 0.03}$	$\mathbf{0.69}^{\pm 0.03}$	$0.64^{\pm 0.05}$	$\underline{0.68}^{\pm 0.04}$
	DT ↓	$0.30^{\pm 0.02}$	$\underline{0.11}^{\pm 0.03}$	$\mathbf{0.07}^{\pm 0.01}$	$0.39^{\pm 0.05}$	$0.22^{\pm 0.03}$
	TPR ↑	$0.68^{\pm 0.07}$	$0.99^{\pm 0.02}$	$0.99^{\pm 0.01}$	$0.72^{\pm 0.10}$	$0.91^{\pm 0.04}$
	TNR ↑	$0.73^{\pm 0.04}$	$0.49^{\pm 0.02}$	$0.46^{\pm 0.04}$	$0.80^{\pm 0.06}$	$0.65^{\pm 0.02}$
Sensor noise	Acc. ↑	$0.67^{\pm 0.06}$	$0.66^{\pm 0.04}$	$0.66^{\pm 0.02}$	$\underline{0.74}^{\pm 0.02}$	$\mathbf{0.80}^{\pm 0.02}$
	TWA ↑	$0.56^{\pm 0.05}$	$0.61^{\pm 0.03}$	$\underline{0.63}^{\pm 0.02}$	$0.60^{\pm 0.01}$	$\mathbf{0.67}^{\pm 0.02}$
	DT ↓	$0.41^{\pm 0.05}$	$0.09^{\pm 0.03}$	$0.07^{\pm 0.01}$	$\underline{0.39}^{\pm 0.12}$	$\mathbf{0.30}^{\pm 0.02}$
	TPR ↑	$0.63^{\pm 0.10}$	$1.00^{\pm 0.00}$	$1.00^{\pm 0.00}$	$0.81^{\pm 0.09}$	$0.94^{\pm 0.03}$
	TNR ↑	$0.71^{\pm 0.04}$	$0.31^{\pm 0.08}$	$0.33^{\pm 0.05}$	$0.67^{\pm 0.12}$	$0.66^{\pm 0.02}$
Background	Acc. ↑	$0.68^{\pm 0.04}$	$0.66^{\pm 0.00}$	$0.66^{\pm 0.00}$	$\underline{0.74}^{\pm 0.03}$	$\mathbf{0.77}^{\pm 0.02}$
	TWA ↑	$0.57^{\pm 0.04}$	$\underline{0.61}^{\pm 0.02}$	$0.60^{\pm 0.01}$	$0.58^{\pm 0.03}$	$\mathbf{0.62}^{\pm 0.01}$
	DT ↓	$0.46^{\pm 0.03}$	$0.10^{\pm 0.03}$	$0.12^{\pm 0.03}$	$\underline{0.45}^{\pm 0.02}$	$\mathbf{0.32}^{\pm 0.01}$
	TPR ↑	$0.58^{\pm 0.09}$	$0.99^{\pm 0.01}$	$1.00^{\pm 0.00}$	$0.76^{\pm 0.04}$	$0.94^{\pm 0.02}$
	TNR ↑	$0.78^{\pm 0.03}$	$0.33^{\pm 0.01}$	$0.33^{\pm 0.00}$	$0.73^{\pm 0.05}$	$0.60^{\pm 0.03}$
Lighting	Acc. ↑	$0.69^{\pm 0.04}$	$0.64^{\pm 0.06}$	$0.64^{\pm 0.05}$	$\underline{0.76}^{\pm 0.02}$	$\mathbf{0.80}^{\pm 0.02}$
	TWA ↑	$\underline{0.58}^{\pm 0.03}$	$0.55^{\pm 0.03}$	$0.55^{\pm 0.03}$	$0.56^{\pm 0.02}$	$\mathbf{0.63}^{\pm 0.03}$
	DT ↓	$\underline{0.40}^{\pm 0.07}$	$0.17^{\pm 0.06}$	$0.18^{\pm 0.03}$	$0.49^{\pm 0.09}$	$\mathbf{0.37}^{\pm 0.04}$
	TPR ↑	$0.59^{\pm 0.12}$	$1.00^{\pm 0.00}$	$1.00^{\pm 0.00}$	$0.84^{\pm 0.09}$	$0.94^{\pm 0.02}$
	TNR ↑	$0.79^{\pm 0.03}$	$0.27^{\pm 0.13}$	$0.29^{\pm 0.10}$	$0.68^{\pm 0.12}$	$0.66^{\pm 0.05}$
Layout	Acc. ↑	$0.70^{\pm 0.02}$	$\underline{0.74}^{\pm 0.01}$	$0.71^{\pm 0.02}$	$\underline{0.74}^{\pm 0.04}$	$\mathbf{0.81}^{\pm 0.02}$
	TWA ↑	$0.58^{\pm 0.02}$	$\underline{0.60}^{\pm 0.01}$	$0.58^{\pm 0.03}$	$0.57^{\pm 0.02}$	$\mathbf{0.62}^{\pm 0.02}$
	DT ↓	$0.40^{\pm 0.04}$	$\underline{0.28}^{\pm 0.02}$	$\mathbf{0.26}^{\pm 0.02}$	$0.54^{\pm 0.06}$	$0.48^{\pm 0.03}$
	TPR ↑	$0.63^{\pm 0.06}$	$0.98^{\pm 0.03}$	$0.98^{\pm 0.02}$	$0.65^{\pm 0.11}$	$0.86^{\pm 0.06}$
	TNR ↑	$0.77^{\pm 0.04}$	$0.50^{\pm 0.04}$	$0.43^{\pm 0.03}$	$0.84^{\pm 0.05}$	$0.77^{\pm 0.04}$
Average	Acc. ↑	$0.70^{\pm 0.03}$	$0.68^{\pm 0.02}$	$0.67^{\pm 0.01}$	$\underline{0.75}^{\pm 0.02}$	$\mathbf{0.79}^{\pm 0.00}$
	TWA ↑	$0.59^{\pm 0.02}$	$\underline{0.62}^{\pm 0.01}$	$\underline{0.62}^{\pm 0.01}$	$0.59^{\pm 0.02}$	$\mathbf{0.65}^{\pm 0.00}$
	DT ↓	$\underline{0.35}^{\pm 0.01}$	$0.13^{\pm 0.02}$	$0.11^{\pm 0.00}$	$0.44^{\pm 0.07}$	$\mathbf{0.32}^{\pm 0.02}$
	TPR ↑	$0.65^{\pm 0.09}$	$0.99^{\pm 0.01}$	$0.99^{\pm 0.00}$	$0.75^{\pm 0.08}$	$0.92^{\pm 0.02}$
	TNR ↑	$0.75^{\pm 0.02}$	$0.37^{\pm 0.04}$	$0.35^{\pm 0.02}$	$0.75^{\pm 0.06}$	$0.66^{\pm 0.03}$

Table 14: **Failure detection results in the real world.** We report the mean and standard deviation over three draws of calibration rollouts.

Task	Metric	STAC	logpZO	RND-OE	ACE	**VFD (ours)**
`Place`	Acc. ↑	$0.42^{\pm 0.02}$	$\underline{0.71}^{\pm 0.00}$	$0.63^{\pm 0.02}$	$0.54^{\pm 0.05}$	$\mathbf{0.78}^{\pm 0.02}$
	TWA ↑	$0.39^{\pm 0.01}$	$\mathbf{0.65}^{\pm 0.00}$	$\underline{0.57}^{\pm 0.02}$	$0.51^{\pm 0.04}$	$\underline{0.57}^{\pm 0.01}$
	DT ↓	$0.80^{\pm 0.26}$	$\mathbf{0.12}^{\pm 0.00}$	$0.13^{\pm 0.01}$	$0.32^{\pm 0.07}$	$\underline{0.50}^{\pm 0.05}$
	TPR ↑	$0.10^{\pm 0.08}$	$0.98^{\pm 0.03}$	$1.00^{\pm 0.00}$	$0.21^{\pm 0.12}$	$0.86^{\pm 0.00}$
	TNR ↑	$0.75^{\pm 0.04}$	$0.43^{\pm 0.02}$	$0.27^{\pm 0.05}$	$0.88^{\pm 0.02}$	$0.71^{\pm 0.03}$
`Stack`	Acc. ↑	$0.43^{\pm 0.02}$	$0.68^{\pm 0.10}$	$\underline{0.77}^{\pm 0.05}$	$0.58^{\pm 0.03}$	$\mathbf{0.81}^{\pm 0.01}$
	TWA ↑	$0.43^{\pm 0.02}$	$\underline{0.66}^{\pm 0.10}$	$\mathbf{0.71}^{\pm 0.05}$	$0.53^{\pm 0.01}$	$0.64^{\pm 0.02}$
	DT ↓	$1.00^{\pm 0.00}$	$0.03^{\pm 0.00}$	$\mathbf{0.12}^{\pm 0.02}$	$0.26^{\pm 0.07}$	$\underline{0.54}^{\pm 0.07}$
	TPR ↑	$0.00^{\pm 0.00}$	$1.00^{\pm 0.00}$	$1.00^{\pm 0.00}$	$0.38^{\pm 0.13}$	$0.64^{\pm 0.05}$
	TNR ↑	$0.85^{\pm 0.04}$	$0.36^{\pm 0.20}$	$0.53^{\pm 0.11}$	$0.79^{\pm 0.08}$	$0.99^{\pm 0.02}$
`Drawer`	Acc. ↑	$0.45^{\pm 0.04}$	$\mathbf{0.71}^{\pm 0.02}$	$\underline{0.67}^{\pm 0.05}$	$0.48^{\pm 0.04}$	$0.60^{\pm 0.01}$
	TWA ↑	$0.44^{\pm 0.05}$	$\mathbf{0.61}^{\pm 0.01}$	$\underline{0.56}^{\pm 0.01}$	$0.46^{\pm 0.05}$	$0.53^{\pm 0.01}$
	DT ↓	$0.86^{\pm 0.24}$	$\mathbf{0.26}^{\pm 0.04}$	$0.24^{\pm 0.15}$	$0.49^{\pm 0.09}$	$0.49^{\pm 0.00}$
	TPR ↑	$0.03^{\pm 0.05}$	$0.72^{\pm 0.05}$	$0.94^{\pm 0.10}$	$0.09^{\pm 0.08}$	$0.30^{\pm 0.00}$
	TNR ↑	$0.88^{\pm 0.11}$	$0.69^{\pm 0.09}$	$0.40^{\pm 0.20}$	$0.88^{\pm 0.15}$	$0.90^{\pm 0.02}$
Average	Acc. ↑	$0.43^{\pm 0.02}$	$\underline{0.70}^{\pm 0.03}$	$0.69^{\pm 0.04}$	$0.54^{\pm 0.01}$	$\mathbf{0.73}^{\pm 0.01}$
	TWA ↑	$0.42^{\pm 0.02}$	$\mathbf{0.64}^{\pm 0.03}$	$\underline{0.61}^{\pm 0.02}$	$0.50^{\pm 0.00}$	$0.58^{\pm 0.00}$
	DT ↓	$0.89^{\pm 0.08}$	$\mathbf{0.14}^{\pm 0.01}$	$0.16^{\pm 0.05}$	$0.36^{\pm 0.07}$	$\underline{0.51}^{\pm 0.03}$
	TPR ↑	$0.04^{\pm 0.02}$	$0.90^{\pm 0.02}$	$0.98^{\pm 0.03}$	$0.22^{\pm 0.05}$	$0.60^{\pm 0.02}$
	TNR ↑	$0.83^{\pm 0.05}$	$0.49^{\pm 0.04}$	$0.40^{\pm 0.10}$	$0.85^{\pm 0.05}$	$0.87^{\pm 0.02}$

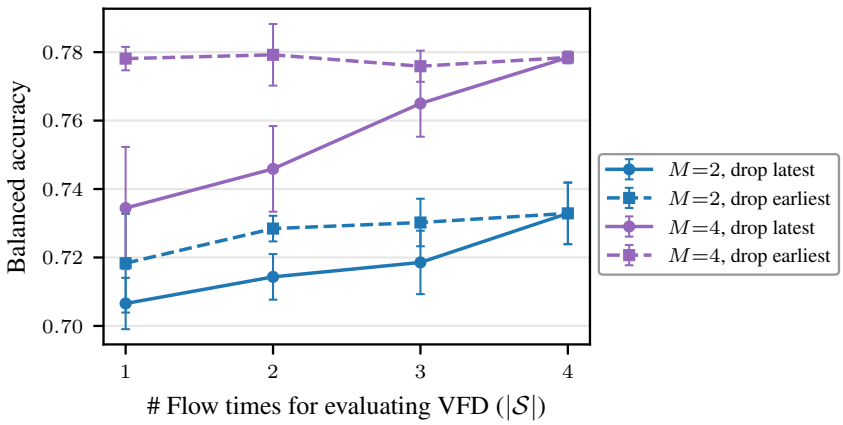

Figure 16: **Impact of the flow times for evaluating VFD**. Dropping later (larger) flow timesteps from the VFD computation (42) reduces performance more than dropping earlier ones, indicating that disagreements in the velocity fields are more indicative of epistemic uncertainty when the action chunks contain less noise.

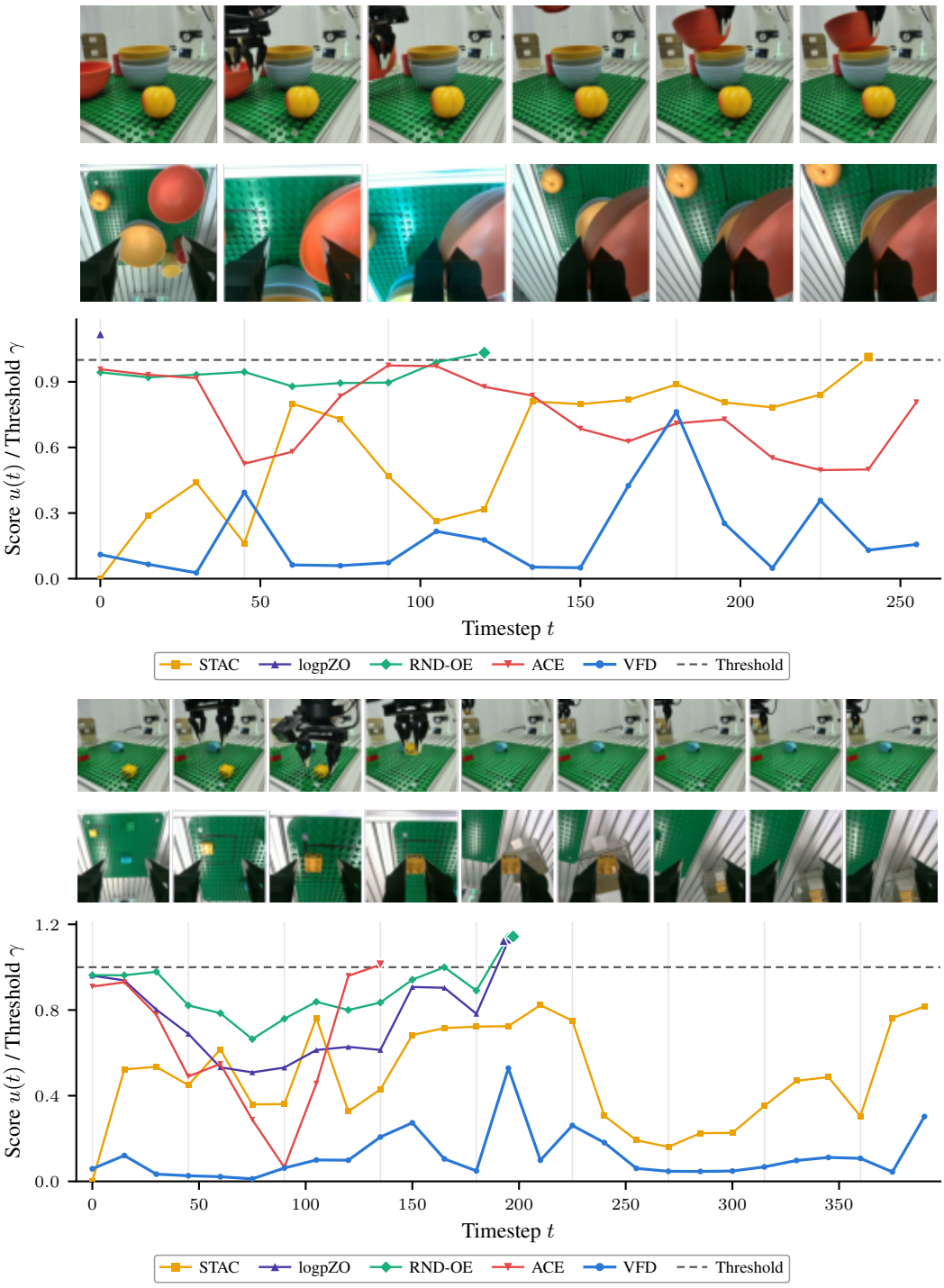

Figure 17: Exemplary successful rollouts that are correctly not flagged as failure by VFD.

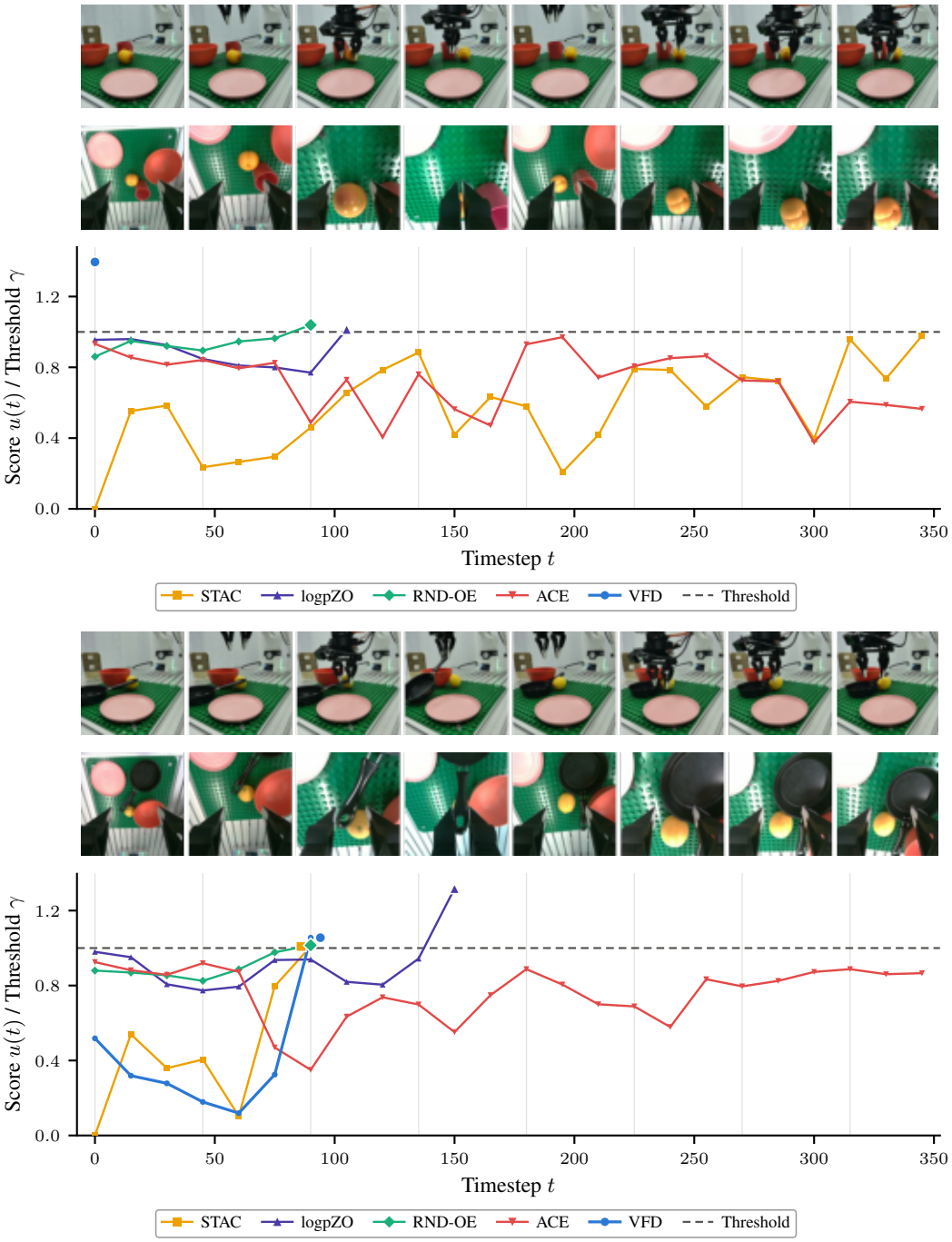

Figure 18: Exemplary failed rollouts that are correctly flagged as failure by VFD.