

# INSIGHT: Inference-time Sequence Introspection for Generating Help Triggers in Vision-Language-Action Models

Ulas Berk Karli<sup>1</sup>, Ziyao Shangguan<sup>1</sup>, and Tesca Fitzgerald<sup>1</sup>

**Abstract**—Recent Vision-Language-Action (VLA) models show strong generalization capabilities, yet they lack introspective mechanisms for anticipating failures and requesting help from a human supervisor. We present INSIGHT, a learning framework for leveraging token-level uncertainty signals to predict when a VLA should request help. Using  $\pi_0$ -FAST as the underlying model, we extract per-token *entropy*, *log-probability*, and Dirichlet-based estimates of *aleatoric and epistemic uncertainty*, and train compact transformer classifiers to map these sequences to help triggers. We explore supervision regimes for strong or weak supervision, and extensively compare them across in-distribution and out-of-distribution tasks. Our results show a trade-off: strong labels enable models to capture fine-grained uncertainty dynamics for reliable help detection, while weak labels, though noisier, still support competitive introspection when training and evaluation are aligned, offering a scalable path when dense annotation is impractical. Crucially, we find that modeling the temporal evolution of token-level uncertainty signals with transformers provides far greater predictive power than static sequence-level scores. This study provides the first systematic evaluation of uncertainty-based introspection in VLAs, opening future avenues for active learning and for real-time error mitigation through selective human intervention.

## I. INTRODUCTION

Vision-language-action (VLA) models offer a promising direction for general-purpose robot policies, using autoregressive token prediction to flexibly map observations and open-ended language instructions to action sequences. However, these models lack mechanisms for introspection [1]–[3]: they do not provide feedback on what training data they need in order to improve their performance, they predict the next token without signaling when they are uncertain or likely to fail, and they do not have the ability to request help from a human operator [4]–[9]. These capabilities are critical for robots to operate safely and reliably in unstructured settings [10], [11].

Our work takes a step toward a **human-in-the-loop, lifelong learning paradigm for VLA models**. Rather than collect training data once, we envision iterative cycles of (i) training the model, (ii) deploying it in the wild, (iii) using introspection to selectively query a human teacher when the model is uncertain, and (iv) incorporating this feedback to improve both immediate task performance and future model updates. In this paper, we focus on the introspection step. While token-level uncertainty metrics have been widely studied in LLMs [12]–[15], their effectiveness in embodied VLA settings remains an open question.

<sup>1</sup>Department of Computer Science, Yale University, New Haven, CT, USA {ulasberk.karli, ziyao.shangguan, tesca.fitzgerald}@yale.edu

**Research Question #1:** Can uncertainty signals extracted from token-level probability distributions at inference time reliably predict when a VLA should request human help?

We address this by introducing and evaluating **INSIGHT** (*Inference-time Sequence Introspection for Generating Help Triggers*). INSIGHT instantiates metrics commonly applied in LLMs (entropy, log-probability, and Dirichlet-based approximations of aleatoric and epistemic uncertainty [14]) and trains a compact transformer for step-by-step prediction of when the robot should request help to avoid an impending failure. To train this model, we need a practical paradigm for obtaining training labels that reflect when the robot should or should not request help. This motivates our second research question.

**Research Question #2:** How does the source of training labels affect this capability for within-/out-of-distribution tasks? We compare two supervision strategies. In a *strong labeling* paradigm, an expert annotates the robot’s behavior in each timestep as “needs help” or “no help”, thus supporting binary classification for each action chunk. In a *weak labeling* paradigm, we rely on only episode-level outcomes (success/failure), constituting a multi-instance learning problem where the model learns to localize predictive patterns in timesteps prior to the failure.

We evaluate INSIGHT extensively, comparing models trained under both supervision regimes and on both in-distribution and out-of-distribution tasks. Our results reveal clear tradeoffs: strong labels yield higher fidelity but require costly step-level annotation, while weak labels enable cheaper training at the expense of precision. We make the following contributions:

- 1) Demonstrate that *sequential structure* of token-level uncertainty metrics provide more effective uncertainty quantification for VLAs than single-value thresholds (such as in Conformal Prediction), underscoring the importance of temporal models for reliable help detection.
- 2) Formulate the help prediction problem under two supervision regimes: strong supervision with step-level labels and weak supervision via multi-instance learning from episode outcomes.
- 3) Introduce *INSIGHT*, the first introspective framework for VLA models that leverages token-level uncertainty signals to generate help triggers.
- 4) Extensive evaluations across in-distribution and out-of-distribution tasks, quantifying the tradeoffs between labeling effort, predictive accuracy, and generalization.

## II. RELATED WORK

**Vision-Language-Action models (VLAs)** enable robots to interpret high-level language instructions, perceive the environment through vision, and output low-level control commands to accomplish tasks. VLAs can be divided into two main architectural families: transformers with an autoregressive decoding scheme and a dedicated diffusion based action head (e.g.,  $\pi_0$  [7],  $\pi_{0.5}$  [16], Octo [17]), and purely autoregressive ones (e.g.,  $\pi_0$ -FAST [8], OpenVLA [4], RT-1 [5], RT-2 [6]). We target purely autoregressive models in order to employ token-level uncertainty metrics from the LLM literature [1], [18].

By pre-training on extensive web data and vast amounts<sup>1</sup> of training episodes, VLAs achieve impressive performance across diverse sensory inputs and action spaces. However, even state-of-the-art VLAs degrade when deployed in OOD settings, such as when tasks involve novel objects, altered layouts, or physical conditions not seen during training. In these cases, VLAs may ‘hallucinate’ and fail to advance the task or even cause unsafe behavior. While post-training fine-tuning can partially mitigate these issues [7], it is infeasible to anticipate every possible environment shift in advance.

This gap motivates the need for **introspection**: mechanisms that allow robots to recognize, at inference time, when their predictions are unreliable in order to avoid cascading failures. By querying a human for help, the robot can both resolve the immediate task failure and acquire additional training data, enabling it to improve its long-term robustness. To this end, KnowNo [19] demonstrated that conformal prediction can enable LLM-based planners to identify when they are uncertain about their next action. Subsequent work [1], [3], [20] has extended this idea in multiple directions, including reasoning over sub-goals, integrating knowledge bases, adding action-feasibility metrics, enabling multi-robot planning, and leveraging multimodal LLMs for failure detection. Similarly, Xu et al. [21] proposed a method for failure detection from robot state and observations, even when trained only on successful executions.

**Research Gap.** To date, little work has explored introspection in VLAs. Prior frameworks such as KnowNo [19] have successfully performed introspection for high-level action selection, but these approaches are not directly applicable to VLAs. The key distinction is that LLM-based planners operate over a discrete set of symbolic, high-level actions, whereas VLAs must directly generate low-level continuous control sequences in joint space. This introduces challenges: uncertainty must be inferred not over a handful of symbolic options but across long, variable-length token sequences that map to fine-grained motor commands. Moreover, errors in VLA policies often manifest gradually within an action sequence (e.g., drift, misalignment, or compounding control errors), rather than as a single incorrect high-level choice. Thus, there is a pressing need for uncertainty quantification methods tailored to VLAs.

<sup>1</sup>OpenVLA [4] was trained on 970k real-robot demonstrations drawn from the Open X-Embodiment dataset.

## III. BACKGROUND: UNCERTAINTY METRICS IN LLMs

We first ground our approach in uncertainty estimation techniques developed in the context of LLMs, and then adapt these ideas to the embodied VLA setting.

*Entropy* captures the spread of a model’s output distribution, enabling confidence ranking via differential entropy over the vocabulary [22]. At the token level, high entropy can indicate low-confidence predictions that are likely to be incorrect or hallucinated; for example, Fadeeva et al. [13] use it to detect unreliable spans in LLM outputs.

*Perplexity* is the exponentiated average negative log-likelihood per token, and quantifies model ‘surprise’.

$$\text{PPL}(x) = \exp\left(-\frac{1}{N} \sum_{i=1}^N \log_2 p(x_i | x_{<i})\right) \quad (1)$$

It is directly linked to cross-entropy loss and is a standard proxy for model confidence [23]. Lower perplexity correlates with more coherent outputs and better human-judged quality. It is also effective for OOD detection; Ren et al. [10] show that thresholding sequence perplexity enables abstention from poor outputs and improves summarization and translation performance.

Recent work has highlighted the limitations of probability-based uncertainty estimation. Because probabilities are normalized by softmax, they lose information about the raw evidence contained in logits, which can lead to counterintuitive reliability estimates [14]. To address this, the Logits-induced Token Uncertainty (LogTokU) framework treats logits directly as evidence, modeling them via a Dirichlet distribution. Specifically, the logits of the top- $K$  tokens are used to form evidence parameters  $\alpha_k$  for the Dirichlet:

$$\alpha_k = M(\tau_k | q, a_{t-1}), \quad \alpha_0 = \sum_{k=1}^K \alpha_k, \quad (2)$$

where  $\tau_k$  is the token with the  $k$ -th largest logit, and  $\alpha_0$  is the total mass of evidence. This enables decomposition into *aleatoric uncertainty* (AU), capturing inherent data ambiguity, and *epistemic uncertainty* (EU), capturing model knowledge gaps. Both have closed-form solutions:

$$\text{AU}(a_t) = -\sum_{k=1}^K \frac{\alpha_k}{\alpha_0} [\psi(\alpha_k + 1) - \psi(\alpha_0 + 1)] \quad (3)$$

$$\text{EU}(a_t) = K / \sum_{k=1}^K (\alpha_k + 1) \quad (4)$$

where  $\psi(\cdot)$  is the digamma function, defined as  $\psi(x) = \frac{d}{dx} \log \Gamma(x)$ .

## IV. PROBLEM FORMULATION

While uncertainty metrics have been established for LLMs, they have not yet been adapted for VLAs due to their higher-dimensional inputs and the complex mapping between tokens and actions [24], [25]. In this section, we formalize the help-prediction problem for VLAs. Later, in

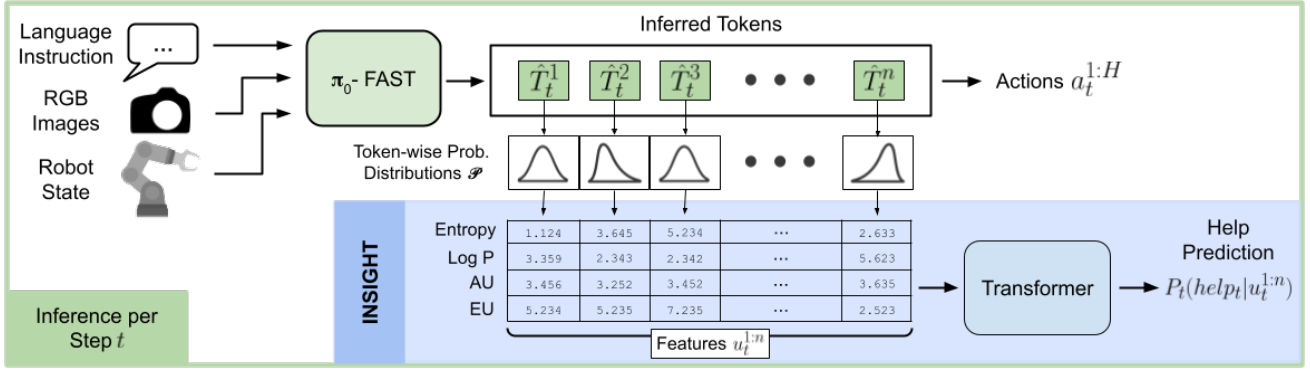


Fig. 1: We use the  $\pi_0$ -FAST model as an underlying policy, translating inputs into autoregressive action tokens  $T_t^1, \dots, T_t^n$ . Our method, INSIGHT, uses the probability distribution that each token is sampled from, and extracts tokenwise uncertainty features  $u_t^{1:n}$ . We train a lightweight transformer to classify these features and predict if help is needed at that step.

Sec. V, we propose our method for bridging the gap between this problem and LLM-inspired uncertainty metrics.

At a high level, an autoregressive VLA model performs inference at each step  $t$  within an episode by taking an observation  $o_t = \langle l, I_t, q_t \rangle$  (consisting of language instruction  $l$ , RGB image(s)  $I_t$ , and robot state  $q_t$ ) and producing a *sequence of tokens*  $\hat{T}$  that represent robot actions in cartesian or joint space. The model autoregressively predicts the probability of tokens from a vocabulary  $\mathcal{V}$  via greedy decoding:

$$\hat{T}_t^i = \arg \max_{T \in \mathcal{V}} \mathcal{P}_t^i(T) \quad \mathcal{P}_t^i(T) = p_\theta(T | \hat{T}_t^{1:i-1}, l, I_t, q_t) \quad (5)$$

where  $\mathcal{P}$  is the probability distribution over all tokens in the vocabulary  $\mathcal{V}$ , as indicated by the VLA using its learned parameters  $\theta$ . This yields a decoded token sequence at episode step  $t$ :  $\hat{T}_t^{1:n} = (\hat{T}_t^1, \dots, \hat{T}_t^n)$ . This token sequence is then de-tokenized (using the action tokenizer  $\mathcal{T}_a$ ) to reconstruct and execute the corresponding continuous action “chunk”:  $\hat{a}_t^{1:H} = \mathcal{T}_a^{-1}(\hat{T}_t^{1:n})S$ .

Importantly, the  $\pi_0$ -FAST tokenizer produces a variable number of tokens  $n$  for a fixed chunk size  $H$ . Thus, individual tokens do not correspond one-to-one with actions in the chunk, and the tokenization length is not static.

A **step** consists of one cycle of collecting an observation, performing inference, decoding the token sequence into one action chunk, and executing it. An **episode** consists of all  $K$  such steps:  $E = (a_1^{1:H}, a_2^{1:H}, \dots, a_K^{1:H})$ . Fig. 2 summarizes how an episode is decomposed into steps, actions, and tokens.

**Our objective** is to determine, at each step, from the token-wise probability distribution  $\mathcal{P}$ , when a VLA is uncertain and should request human help instead of executing its predicted action chunk; i.e.,  $P(\text{help}_t | \mathcal{P}_t^1 \dots \mathcal{P}_t^n)$ . We consider the model successful if these predicted help triggers align with ground-truth indicators of failure—either a lack of progress toward task completion at the current step or eventual failure of the overall episode.

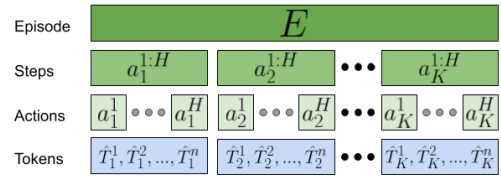


Fig. 2: Hierarchy of episodes, steps, actions, and tokens. Each step involves one round of observation, inference, and action execution, shown in Fig. 1.

## V. APPROACH

Our approach, INSIGHT, addresses the help-prediction problem through the following steps:

- 1) Taking as input the token-wise probability distributions  $\mathcal{P}_t^i$  produced during inference by  $\pi_0$ -FAST (see Eq. 5);
- 2) Computing uncertainty features over each distribution;
- 3) Encoding these features with a transformer;
- 4) Outputting a probability of whether the model should, at that step, (a) execute the actions or (b) request help.

This classifier is deployed in parallel with  $\pi_0$ -FAST and issues binary decisions (*request help* or *proceed*) in real time (Fig. 1). We now address two design questions: (i) how to map uncertainty features to help predictions, and (ii) how to implement practical training paradigms for data labeling.

### A. Uncertainty Features

The INSIGHT pipeline begins with the pretrained  $\pi_0$ -FAST policy, which generates an action sequence as a variable-length token sequence. For each predicted token  $i$  for step  $t$ , we obtain the probability distribution  $\mathcal{P}_t^i$  from which that token was selected, and then extract a feature vector  $u_t^i \in \mathbb{R}^4$  from its predictive distribution and logits:

$$u_t^i = \underbrace{[\mathbb{H}(\mathcal{P}_t^i(\cdot | \hat{T}_t^{1:i-1}, o_t))]}_{\text{Entropy}}, \underbrace{[-\log \mathcal{P}_t^i(\hat{T}_t^i | \hat{T}_t^{1:i-1}, o_t)]}_{\text{Negative log-prob}}, \text{AU}_t^i, \text{EU}_t^i \quad (6)$$

where AU and EU are aleatoric and epistemic uncertainties derived from logits-based Dirichlet evidence (Eqs. 2–4). These token-level features are aggregated into a  $4 \times N$  matrix

$u_t$  representing one step  $t$  where  $N$  is a fixed maximum token length used to ensure consistent input size across steps (shorter sequences are padded as needed), which is processed by a transformer encoder and a prediction head  $g_\psi$  to produce a step-level help score  $r_t \in [0, 1]$ .

### B. Training Paradigms: Strong vs. Weak Supervision

We now consider how to train a model to classify these feature inputs based on practical annotation data. Collecting step-level labels is time-consuming and often subjective, since deciding whether *help is needed* requires an expert to judge if the model’s predicted action meaningfully contributes to task progress. In contrast, episode-level outcomes (success or failure) are easier to obtain and more objective, but noisier, since they do not reveal which specific step should have triggered help. We thus consider two supervision regimes. In **strong supervision**, we annotate each *step*  $t$  with a binary label  $y_t \in \{0, 1\}$  indicating whether help was needed. The classifier is trained with binary cross-entropy:

$$\mathcal{L}_{\text{strong}}(\psi) = - \sum_t [y_t \log r_t + (1 - y_t) \log(1 - r_t)] \quad (7)$$

In **weak supervision**, we annotate each *episode* based on its outcome. Each episode  $E^{(e)}$  receives a binary label  $Y^{(e)} = 0$  or 1 for success and failure, respectively. This assumes that (i) if an episode was successful, the model did not need help during *any* of its steps, and (ii) if an episode ends in failure, the model should have asked for help in at least one step. To train, we pool step logits  $\{\ell_t\}$  into an episode-level logit using log-sum-exp pooling with temperature  $\lambda$ :

$$\tilde{\ell}^{(e)} = \frac{1}{\lambda} \log \sum_{t=1}^{N_e} \exp(\lambda \ell_t), \quad \hat{Y}^{(e)} = \sigma(\tilde{\ell}^{(e)}) \quad (8)$$

Episode prediction is optimized with binary cross-entropy:

$$\mathcal{L}_{\text{weak}}(\psi) = - \sum_e [Y^{(e)} \log \hat{Y}^{(e)} + (1 - Y^{(e)}) \log(1 - \hat{Y}^{(e)})] \quad (9)$$

### C. Classifier Model Architectures

Under strong supervision, we use a compact Transformer that processes the sequence of token-level features within a step. Token embeddings are projected to a  $d_h=64$  hidden space, enriched with sinusoidal positional embeddings, and passed through a Transformer encoder with one self-attention layer and  $n_{\text{head}}=4$  heads. The encoded tokens are aggregated by masked attention pooling and passed through a two-layer feed-forward head (32 hidden units) to produce a step logit. This model has approximately 300k parameters and is trained with step-level binary cross-entropy.

Under weak supervision, we extend this setup to entire episodes. Each step embedding is encoded by the same  $d_h=64$  Transformer encoder (1–2 layers,  $n_{\text{head}}=4$ ), yielding a step logit. Step logits are pooled into an episode-level logit using log-sum-exp pooling with temperature  $\lambda=6.0$ . The pooled logit is sigmoid-activated to predict success or failure and optimized with episode-level binary cross-entropy. This model has approximately 500k parameters

due to the additional pooling operation and episode-level prediction head.

Both architectures are compact ( $\ll 1\text{M}$  params) and leverage self-attention to capture non-local and irregular patterns in uncertainty signals as they evolve across tokens, making them well suited for temporal introspection in VLA models.

## VI. DATA COLLECTION & PRE-TRAINING

Using  $\pi_0$ -FAST as a base policy, we evaluate the efficacy of our approach in monitoring the policy’s rollouts and identifying steps in which the robot should ask for help. Our evaluation requires three types of data: (1) demonstration data to fine tune  $\pi_0$ -FAST for our robot, tasks, and environments; (2) action rollouts (and corresponding uncertainty features) from the fine-tuned  $\pi_0$ -FAST model on within-distribution, shifted-distribution, and simulated OOD settings; and (3) strong and weak labels indicating steps and episodes, respectively, when the robot performed poorly and should have asked for help.

### A. Fine-Tuning $\pi_0$ -FAST

To adapt  $\pi_0$ -FAST to our setting, we used a toy kitchen environment, inspired by prior VLA datasets such as BridgeV2 and Open-X [26], [27]. Our platform is an xArm7 manipulator, which was not part of the original training distribution.

We collected a new demonstration dataset using a custom GELLO controller [28] that supports joystick teleoperation. Our dataset spans five task types: `Lift`, `Put`, `Knock`, `Wipe`, and `Stir`. All demonstrations were recorded at 30 Hz and tokenized using the original  $\pi_0$ -FAST discretization pipeline. In total, our dataset contains 80,419 action steps across 5 task categories and 17 distinct tasks. We performed full-parameter fine-tuning of  $\pi_0$ -FAST on this dataset and selected a checkpoint with low training loss and strong performance in physical robot testing. This checkpoint serves as the foundation for our introspection experiments. For the Sim-OOD setting, we additionally fine-tuned another  $\pi_0$ -FAST on the LIBERO dataset [29] using the exact recipe provided by the Physical Intelligence `openpi`<sup>2</sup> repository, and verified that our checkpoint success rates matched theirs.

### B. Rollout Data for Introspection

We next collected rollout data in which we deploy the fine-tuned policy on four tasks on the real-world setup: `lift the carrot`, `lift the eggplant`, `put the corn in the pot`, and `put the pot in the sink`. We also collected rollout data from the LIBERO 10 dataset.

For our **in-distribution dataset**, we collected 160 policy rollouts (episodes), evenly distributed across tasks, with four distinct start configurations per task. This dataset reflects the policy’s in-distribution performance. Our **distribution-shift dataset** consists of 469 rollouts for the same four tasks, but in novel configurations of the environment including: varied object locations, orientations, and included previously unseen objects. This allows us to test whether introspection methods remain reliable under limited distribution shifts. We

<sup>2</sup><https://github.com/Physical-Intelligence/openpi>

Method	TTFH (fail) ↓	Triggers <sub>succ</sub> ↓	Triggers <sub>fail</sub> ( $\geq 1$ ok)	Trigger Rate (success) ↓	Trigger Rate (fail) ↑
CP-W (Entropy)	6.891 ± 2.257	0.457 ± 0.302	1.721 ± 0.739	0.031 ± 0.020	0.118 ± 0.050
Strong Superv.	<b>5.597 ± 0.809</b>	0.710 ± 0.440	7.062 ± 1.225	0.047 ± 0.029	<b>0.472 ± 0.081</b>
Weak Superv.	7.929 ± 1.867	<b>0.122 ± 0.172</b>	1.566 ± 1.025	<b>0.008 ± 0.011</b>	0.105 ± 0.069

TABLE I: Realtime early & frequency characteristics (mean±std across folds). Lower is better except where noted.

generated a **simulated, highly out-of-distribution (OOD) dataset** via 500 rollouts from a  $\pi_0$ -FAST model fine-tuned on the LIBERO dataset [29]. As the LIBERO dataset contains substantially different task families compared to the in-distribution and OOD settings, this Sim-OOD setting provides a rigorous and challenging testbed for assessing the robustness of the introspection methods that were trained on the (i) in-distribution dataset or (ii) in-distribution + distribution-shift datasets.

### C. Strong and Weak Labels

For strong supervision, we annotated each timestep using the following criterion: *given the observation image and the action inferred by the model, if the action does not contribute to task progress, that step is labeled as **needing help***. We acknowledge that this labeling is inherently subjective and potentially noisy, since “task progress” can depend on the annotator’s interpretation. However, our focus was on applying the criterion consistently across all data, ensuring internal reliability. Importantly, we find that these step-level labels correlate with elevated model uncertainty, suggesting that—even if imperfect—they capture meaningful signals about when the model is likely to fail. For weak supervision, each episode was labeled as a success if the model completed the instructed task within the maximum allowable steps.

## VII. EVALUATION

We evaluate our proposed framework across five settings: (i) in-distribution test, (ii) distribution-shift test, (iii) large-scale combined in-distribution test, (iv) out-of-distribution (OOD) test with simulation data, and (v) real-time test. Within each setting, we further compare different combinations of training and testing under strong and weak labels. For all tests except the real-time evaluation, we perform 10-fold cross-validation and report means and standard deviations across folds. We use complimentary metrics for classification performance: accuracy reflects how reliably the model avoids both unnecessary help and missed helps, while the F1 score emphasizes the model’s ability to identify true help-needed cases despite noise and sparsity in the labels.

### A. Baselines

We benchmark against **Conformal Prediction (CP)** [30], [31] baselines, which provide distribution-free guarantees on the probability of missed help events. For CP, we used data from the same  $\pi_0$ -FAST model and calibrated thresholds on the training set, enforcing  $p(\text{missed help}) \leq \beta$  with  $\beta = 0.2$  under the chosen non-conformity score. We implement two versions of CP with different nonconformity scores: (i)

entropy of the predictive distribution and (ii) sequence-level perplexity. We calibrated CP under two regimes: using strong labels to construct the calibration set, or using weak labels by taking the max uncertainty over all steps within an episode.

### B. Results

1) *In-Distribution*: This setting represents the ideal case where data for (i) tuning the VLA model, (ii) training/calibrating help-prediction models (INSIGHT or CP), and (iii) testing help-prediction models all come from the same distribution, and thus the coverage guarantees of CP formally hold. We report results in Fig. 3a for all combinations of weak and strong labels in training and testing.

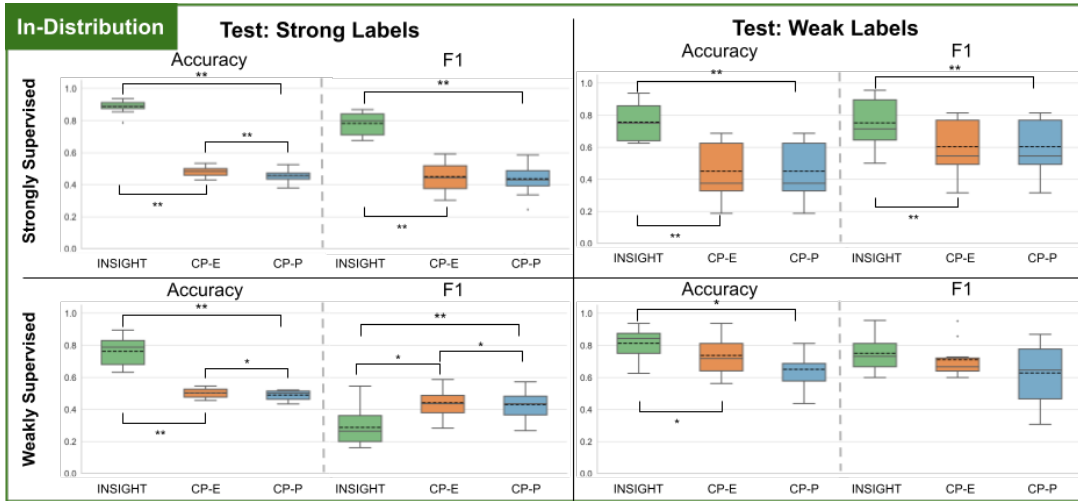
2) *Distribution-Shift*: In this setting, object locations, orientations, and unseen objects differ from those used during training. Since CP requires exchangeability between calibration and test data, its guarantees are no longer valid here; thus these results should be interpreted as diagnostic rather than guaranteed. We report the results in Fig. 3b.

3) *Large In-Distribution Test*: We combine the in-distribution and distribution-shift datasets to create a larger in-distribution dataset to train and evaluate under both strong and weak supervision regimes. This setup increases training diversity while still maintaining exchangeability, allowing us to assess whether scaling data within the same distribution improves performance. We report the results in Fig. 3c.

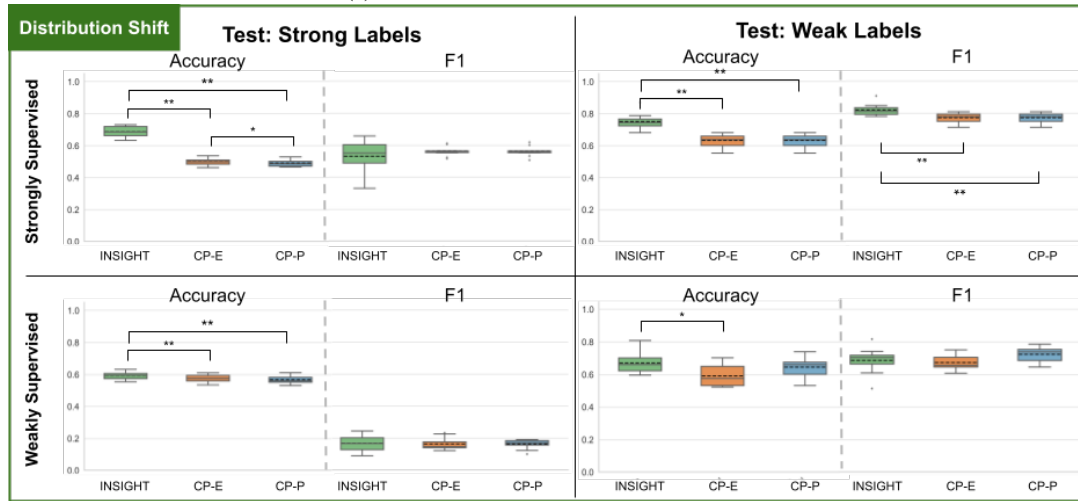
4) *OOD Simulation Test*: In this setting, models are trained on real-world data generated by one  $\pi_0$ -FAST checkpoint, but tested on data produced by a *different*  $\pi_0$ -FAST model fine-tuned on the LIBERO dataset [29]. Because the underlying policies are tuned on different datasets, the trajectories encountered at test time reflect shifts not only in tasks but also in the behavior of the policy itself, making this a highly-OOD case. Fig. 4 summarizes performance.

5) *Timing and Frequency Analysis*: Unlike previous evaluations that assess performance, this analysis focuses on when interventions occur within episodes and how often they are raised, balancing utility (avoiding unnecessary interruptions on successful runs) with responsiveness (timely detection of failures). In Table I, we report the following:

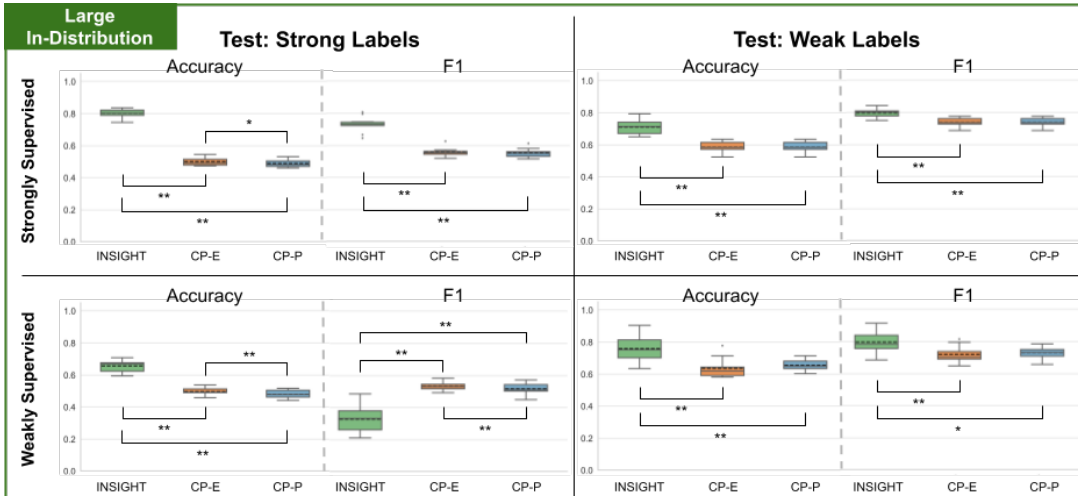
- **Time-to-First-Help (TTFH)**: the index of the first help trigger  $t$  in a failure episode.
- **Trigger Count**: the total number of help triggers raised per episode. We report separate averages for successful episodes (where we ideally minimize triggers) and failed episodes (ideally few triggers but  $\geq 1$ ).
- **Trigger Rate**: the per-step average  $\sum_t f_t/T$ , providing a normalized measure of intervention frequency independent of episode length.



(a) Results for the in-distribution dataset.



(b) Results for the distribution-shift dataset.



(c) Results for the large in-distribution dataset.

Fig. 3: Results for the transformer (INSIGHT) and Conformal Prediction based on entropy (CP-E) and perplexity (CP-P). Each box plot indicates mean (dashed horizontal lines) and median (solid horizontal lines) performance across folds. Error bars indicate 1 standard deviation. Significance by paired Wilcoxon (two-sided) across folds: \*  $p < 0.05$ , \*\*  $p < 0.01$ .

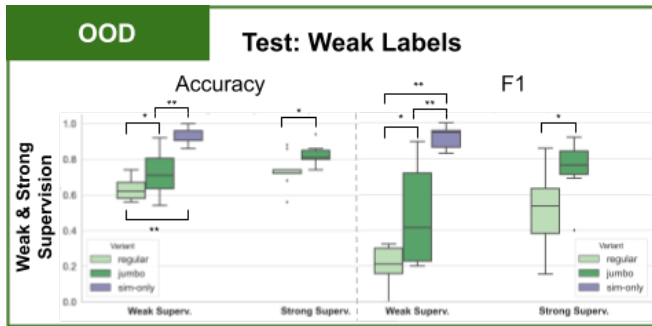


Fig. 4: Simulation-based OOD evaluation. We compare transformer variants under different supervision regimes: regular (trained on the real-world, formerly in-distribution, dataset), jumbo (trained on the combined in-distribution + distribution-shift data), and sim-only (weakly-supervised). Significance by paired Wilcoxon (two-sided) across folds: \*  $p < 0.05$ , \*\*  $p < 0.01$ .

## VIII. DISCUSSION

**Do our uncertainty metrics offer any predictive power for requesting help?** Across all experiments, token-level uncertainty signals (entropy, log-probability, AU, EU) provided predictive signal beyond random guessing (i.e., accuracy and  $F1 > 0.5$ ), with the exception of the weak-training/strong-testing case under distribution shift, where F1 fell below 0.5 due to the mismatch between noisy supervision and strict evaluation. Importantly, there is no prior benchmark for introspection in VLA models; thus, our results establish a first reference point for the field. Even CP, though limited to aggregated sequence-level scores, occasionally reached accuracy and F1 values near 0.7 under weak-label evaluation (see Fig. 3a, weak training and weak testing; Fig. 3b weak testing; Fig. 3c weak testing) reflecting alignment between episode-level calibration and evaluation. While CP alone is not competitive with temporal transformers, our work demonstrates for the first time how such baselines perform in this setting, setting the stage for future comparisons.

**How do aggregate vs sequential use of these metrics affect performance?** CP relies on aggregated sequence-level scores and, while it can occasionally achieve reasonable accuracy or F1 under weak-label evaluation, it typically suffers from poor accuracy and F1 indicating near-chance performance overall (see Fig. 3a strong training with strong testing, Fig. 3b strong testing, Fig. 3c strong testing). By contrast, INSIGHT leverages the sequential structure of token-level signals with transformers, producing consistently higher accuracy and F1. This confirms that uncertainty in VLAs is inherently temporal and must be modeled as such.

**How do models perform when trained on strong vs weak labels?** Strong and weak labels differ not only in quality but also in practicality. Step-level strong labels provide fine-grained supervision but require expert annotation and can be subjective, as deciding whether a step “needs help” is not always clear-cut. Yet, a model that can learn to replicate strong labels is more precise in detecting *when* intervention is

needed, enabling targeted human assistance. Such precision is important for safety-critical settings where unnecessary or missed interventions carry high cost.

Episode-level weak labels, on the other hand, are easy and objective to collect as success/failure outcomes; yet, they offer a noisier signal for training our model because they do not specify *which* step(s) required help. However, replicating weak labels may be sufficient for the objective of avoiding future failures, particularly when scalable supervision is more critical than step-level fidelity.

Our results show that strongly-supervised transformers consistently provide the most reliable performance, achieving the highest F1 scores in nearly all conditions. Weakly-supervised transformers, by contrast, request help more conservatively, leading to low recall and thus lower F1 scores when tested against strong labels (see Fig. 3a, Fig. 3b, Fig. 3c under the combination of weakly-supervised and strong testing). However, when evaluated under weak labels, they remain competitive with strong supervision, showing that weak-label training is viable when dense annotation is infeasible, though at the cost of reduced fidelity. CP calibrated on weak labels performs better in weak-label evaluation than in strong-label evaluation but still lags behind transformers, reinforcing the limits of sequence-level aggregation.

**When and how often does each model trigger help?** The strongly-supervised INSIGHT model triggers help earliest and most frequently, maximizing failure coverage but often over-intervening (see Table I). The weakly supervised model is conservative, rarely interrupting successes but risking late or missed interventions. CP strikes a middle ground with moderate timing and frequency. The optimal choice of model depends on whether deployment prioritizes safety or unobtrusiveness.

**How transferrable are the models under limited distributional shift?** In distribution-shift tests (Fig. 3b), all models degrade, reflecting the difficulty of generalizing across novel configurations. This degradation is minimized for strongly-supervised models. CP achieves high recall under distribution shift, but also incurs numerous false positives and near-chance accuracy, reflecting the challenge of calibrating sequence-level scores when exchangeability assumptions are violated. Compared to CP, INSIGHT generally provides more balanced performance: in strong-label tests, both approaches reach similar F1 scores; but, under weak-label tests, INSIGHT achieves higher accuracy and comparable F1, especially in the strongly-supervised setting. Overall, both methods degrade under a distributional shift, but temporal modeling provides modest gains in balance and reliability.

**How does an increase in data affect the models?** Expanding the in-distribution training dataset with distribution-shift data does not consistently improve robustness. In large in-distribution experiments (Fig. 3c), strongly supervised transformers saw slight decreases in accuracy and F1, likely due to added variability. Weakly supervised transformers benefited from increase in training data more compared to strong transformers when evaluated on weak labels, but overall, label quality proved more important than dataset size.

## How transferrable are the models to significant distribution shifts and changes in the underlying model?

In the simulation-based OOD evaluation, our benchmark is a variant of INSIGHT that is trained directly on the sim-only LIBERO data [29] (i.e., it is within-distribution for the simulated setting). Surprisingly, we find that the INSIGHT models trained on real-world data can still transfer effectively to the simulated OOD setting, despite drastically different simulation environments and task families. Particularly, we find that the strongly-supervised models achieve high accuracy, and that the strongly-supervised jumbo model (trained on the largest amount of real-world data) achieves accuracy and F1 scores that approach that of the sim-only benchmark (see Fig. 4). These findings demonstrate that token-level uncertainty features remain stable across both environments and policy checkpoints. Furthermore, they suggest that strongly-supervised introspection modules offer more robust transfer across OOD settings and VLA model checkpoints, without requiring re-annotation or re-training.

## IX. CONCLUSION

We introduced **INSIGHT**, the first introspective framework for vision–language–action models that leverages token-level uncertainty signals to decide when a robot should request help. Across in-distribution, out-of-distribution, large-scale, and real-time evaluations, our results demonstrate that uncertainty signals extracted at the token level are actionable for introspection, particularly when modeled with temporal transformers. Strong supervision provides the most reliable performance, while weak supervision emerges as a practical alternative when dense annotation is infeasible. Conformal Prediction, though useful as a diagnostic baseline, underscores the need for methods explicitly designed for autoregressive and temporally extended policies.

By establishing these trade-offs, our work makes a first step toward a human-in-the-loop paradigm where VLA models not only act, but also know when to ask for help. More broadly, this opens new research avenues: uncertainty-guided *active learning* to selectively acquire labels, introspection-driven *lifelong learning* where models continuously improve from human feedback, and extension of our approach to *emerging hybrid architectures* that incorporate diffusion-based action experts. Because the uncertainty metrics we exploit are derived from token-level probability distributions, our framework is largely model-agnostic, supporting a future of scalable and introspective VLA systems.

## REFERENCES

- [1] O. Shorinwa, Z. Mei, J. Lidard, A. Z. Ren, and A. Majumdar, “A survey on uncertainty quantification of large language models: Taxonomy, open research challenges, and future directions,” *CoRR*, vol. abs/2412.05563, 2024.
- [2] A. Ciria, G. Schillaci, G. Pezzulo, V. V. Hafner, and B. Lara, “Predictive processing in cognitive robotics: A review,” *Neural Computation*, vol. 33, pp. 1402–1432, Apr. 2021.
- [3] K. Liang, Z. Zhang, and J. F. Fisac, “Introspective planning: Aligning robots’ uncertainty with inherent task ambiguity,” vol. 37, 2024, pp. 71 998–72 031.
- [4] M. J. Kim *et al.*, “Openvla: An open-source vision-language-action model,” in *Conference on Robot Learning*. PMLR, 2024.
- [5] A. Brohan *et al.*, “RT-1: robotics transformer for real-world control at scale,” in *Robotics: Science and Systems*, 2023.
- [6] B. Zitkovich *et al.*, “Rt-2: Vision-language-action models transfer web knowledge to robotic control,” in *Proceedings of The 7th Conference on Robot Learning*. PMLR, 2023.
- [7] K. Black *et al.*, “ $\pi_0$ : A vision-language-action flow model for general robot control,” *CoRR*, vol. abs/2410.24164, 2024.
- [8] K. Pertsch *et al.*, “FAST: efficient action tokenization for vision-language-action models,” *CoRR*, vol. abs/2501.09747, 2025.
- [9] B. Ichter *et al.*, “Do as I can, not as I say: Grounding language in robotic affordances,” in *Conference on Robot Learning*, vol. 205. PMLR, 2022.
- [10] J. Ren, J. Luo, Y. Zhao, K. Krishna, M. Saleh, B. Lakshminarayanan, and P. J. Liu, “Out-of-distribution detection and selective generation for conditional language models,” in *The Eleventh International Conference on Learning Representations ICLR*. OpenReview.net, 2023.
- [11] S. Budd, E. C. Robinson, and B. Kainz, “A survey on active learning and human-in-the-loop deep learning for medical image analysis,” *Medical Image Anal.*, vol. 71, p. 102062, 2021.
- [12] N. Gupta *et al.*, “Language model cascades: Token-level uncertainty and beyond,” in *The Twelfth International Conference on Learning Representations*, 2024.
- [13] E. 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.
- [14] H. Ma, J. Chen, J. T. Zhou, G. Wang, and C. Zhang, “Estimating llm uncertainty with evidence,” 2025. [Online]. Available: <https://arxiv.org/abs/2502.00290>
- [15] U. Arora *et al.*, “Types of out-of-distribution texts and how to detect them,” in *Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2021.
- [16] P. Intelligence *et al.*, “ $\pi_{0.5}$ : a vision-language-action model with open-world generalization,” 2025. [Online]. Available: <https://arxiv.org/abs/2504.16054>
- [17] D. Ghosh *et al.*, “Octo: An open-source generalist robot policy,” in *Robotics: Science and Systems*, 2024.
- [18] A. Sharma and C. David, “Assessing correctness in llm-based code generation via uncertainty estimation,” *CoRR*, vol. abs/2502.11620, 2025.
- [19] A. Z. Ren *et al.*, “Robots that ask for help: Uncertainty alignment for large language model planners,” in *Conference on Robot Learning*, 2023.
- [20] J. Wang, J. Tong, K. Tan, Y. Vorobeychik, and Y. Kantaros, “Conformal temporal logic planning using large language models,” 2024. [Online]. Available: <https://arxiv.org/abs/2309.10092>
- [21] C. Xu *et al.*, “Can we detect failures without failure data? uncertainty-aware runtime failure detection for imitation learning policies,” 2025. [Online]. Available: <https://arxiv.org/abs/2503.08558>
- [22] C. Ling *et al.*, “Uncertainty quantification for in-context learning of large language models,” in *NAACL*. Association for Computational Linguistics, 2024, pp. 3357–3370.
- [23] V. Gangal, A. Arora, A. Einolghozati, and S. Gupta, “Likelihood ratios and generative classifiers for unsupervised out-of-domain detection in task oriented dialog,” in *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.
- [24] R. Firoozi *et al.*, “Foundation models in robotics: Applications, challenges, and the future,” *CoRR*, vol. abs/2312.07843, 2023.
- [25] Y. Hu *et al.*, “Toward general-purpose robots via foundation models: A survey and meta-analysis,” *CoRR*, vol. abs/2312.08782, 2023.
- [26] H. Walke *et al.*, “Bridgedata v2: A dataset for robot learning at scale,” 2024. [Online]. Available: <https://arxiv.org/abs/2308.12952>
- [27] O. X.-E. C. *et al.*, “Open X-Embodiment: Robotic learning datasets and RT-X models,” <https://arxiv.org/abs/2310.08864>, 2023.
- [28] P. Wu, Y. Shentu, Z. Yi, X. Lin, and P. Abbeel, “Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators,” 2023.
- [29] B. Liu *et al.*, “Libero: Benchmarking knowledge transfer for lifelong robot learning,” in *Advances in Neural Information Processing Systems*, 2023.
- [30] V. Vovk, A. Gammerman, and G. Shafer, *Algorithmic Learning in a Random World*. Berlin, Heidelberg: Springer-Verlag, 2005.
- [31] A. N. Angelopoulos and S. Bates, “A gentle introduction to conformal prediction and distribution-free uncertainty quantification,” 2022. [Online]. Available: <https://arxiv.org/abs/2107.07511>