Uncertainty Comes for Free: Human-in-the-Loop Policies with Diffusion Models

Zhanpeng He[*]
Department of Computer Science
Columbia University
New York, New York 10027
Email: zhanpeng@cs.columbia.edu

Yifeng Cao[*]
Department of Electrical Engineering
Columbia University
New York, New York 10027
Email: yc4317@columbia.edu

Matei Ciocarlie
Department of Mechanical Engineering
Columbia University
New York, New York 10027
Email: matei.ciocarlie@columbia.edu

***Abstract*—Human-in-the-loop (HitL) robot deployment has gained significant attention in both academia and industry as a semi-autonomous paradigm that enables human operators to intervene and adjust robot behaviors at deployment time, improving success rates. However, continuous human monitoring and intervention can be highly labor-intensive and impractical when deploying a large number of robots. To address this limitation, we propose a method that allows diffusion policies to actively seek human assistance only when necessary, reducing reliance on constant human oversight. To achieve this, we leverage the generative process of diffusion policies to compute an uncertainty-based metric based on which the autonomous agent can decide to request operator assistance at deployment time, without requiring any operator interaction during training. Additionally, we show that the same method can be used for efficient data collection for fine-tuning diffusion policies in order to improve their autonomous performance. Experimental results from simulated and real-world environments demonstrate that our approach enhances policy performance during deployment for a variety of scenarios.**

***Index Terms*—Human-in-the-loop policies, policy fine-tuning.**

I. Introduction

Human-in-the-Loop (HitL) robot deployment is a paradigm where a human operator can intervene and assist the robot during deployment. This paradigm is seeing increasing adoption in cases where robots must continue to operate adequately even in corner cases not foreseen before deployment.

In parallel, even as recent advances in policy learning have shown significant improvements in robustness at deployment time, current methods can still fail due to problems such as data distribution shift or incomplete state observability. This issue is particularly pronounced in robotics, where available datasets are far smaller compared to other domains.

To address this issue, HitL methods can be a natural fit for learning agents. The robot can operate autonomously when possible, leveraging the ability of policy learning to execute complex motor control tasks. An expert operator can take over to deal with corner cases, ensuring overall task success. However, deploying human-in-the-loop policies can be labor-intensive if it implies constant monitoring of the robot's behavior by the human operator, or frequent, interruptive interventions. Such limitations can make HitL deployment both impractical and costly in many applications.

[*]Equal contributions.

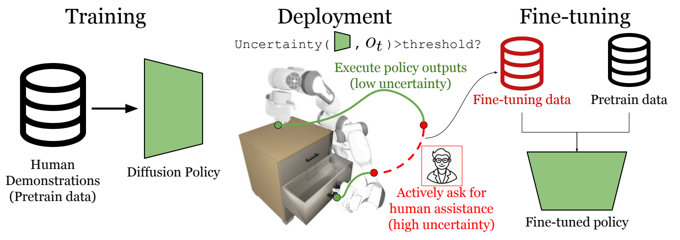
Training
Deployment
Fine-tuning
Uncertainty(▱ , o_t)>threshold?
Human Demonstrations (Pretrain data)
Diffusion Policy
Execute policy outputs (low uncertainty)
Actively ask for human assistance (high uncertainty)
Fine-tuning data
Pretrain data
Fine-tuned policy

Fig. 1. **Human-in-the-loop policies:** We assume a human-in-the-loop robotic agent that actively requests human assistance when necessary. In our method, the robot uses an underlying diffusion policy for autonomous operation. During deployment, we leverage the denoising process inherent to diffusion models in order to identify states with high uncertainty, where the robot seeks operator intervention. We show that this approach allows for task improvement with a small number of such operator calls. Furthermore, the human-operated data can also be used to fine-tune the policy, improving the robot's autonomous execution performance in future deployment.

In this work, we propose a data-driven approach for generating HitL policies. Our method is based on diffusion models, and does not require any additional computation or human intervention during training. Instead, we utilize the intrinsic properties of diffusion models, specifically the denoising process, to allow the agent to compute its internal uncertainty during deployment. When this uncertainty exceeds a certain threshold, the agent proactively seeks human assistance. Furthermore, a key feature of our method is its dual utility: it can be used not only for HitL deployment, but also for collecting additional data to fine-tune the policy.

We test our method on experimental setups where uncertainty can arise from partial observability (where our uncertainty metrics can enable effective supervision), data distribution shifts (where targeted data collection can also improve policy performance) and action multi-modality (where human guidance can help steer the robot into states where mode selection becomes unambiguous). Our main contributions are as follows:

- To the best of our knowledge, we are the first to propose using uncertainty estimation with diffusion policies for efficient HitL deployment of a diffusion-based learning agent. Furthermore, our approach does not require human-robot interactions during training, and incurs minimal computational overhead during deployment.

- We validate our method across multiple types of deployment challenges, in both simulated and real environment. Experimental results show that our approach requires fewer human interventions to and achieves higher task performance compares to alternative learning-based HitL agents.
- We also show that our uncertainty-based state identification method can be utilized to collect targeted fine-tuning data, yielding performance improvements with smaller datasets compared to collecting additional full-trajectory demonstrations.

We see our work as complementary to approaches aiming to improve the robustness of fully autonomous learning-based agents. Until corner cases can be fully eliminated and such agents can generalize to any situation that can be encountered during deployment, HitL methods have the potential to bridge the gap to applications, and help collect critical training data for further policy improvements. We believe this work is a step in this direction.

II. Related Work

Despite recent advances in generalist robot manipulation policies [8, 25], particularly in domestic settings [20, 36], deploying these policies in real-world scenarios remains a significant challenge. Robots often face substantial data distribution shifts when introduced to new environments. In SIMPLER [17], Li et al. show the sensitivity of policies to minor changes, such as variations in robot arm texture, which can cause performance drops exceeding 20%. Additionally, real-world deployment is hampered by incomplete and noisy observations due to environmental factors such as lighting conditions, occlusions, clutter, and weak texture features [9]. The inherent complexity of modeling multi-modal distributions in human demonstrations further exacerbates these issues, particularly in behavior cloning, where stochastic sampling and initialization procedures introduce additional challenges [7, 10, 23, 29]. Our work addresses these deployment difficulties by incorporating human assistance at critical moments. By identifying high-uncertainty states during deployment, our method facilitates timely human intervention, enabling robots to overcome distribution shifts, handle incomplete observations, and navigate complex multi-modal action spaces more effectively.

Human-in-the-loop approaches have been widely explored to improve robot manipulation policies by leveraging human feedback in various forms, including interventions [23, 31], preferences [16], rankings [4], scalar feedback [22], and human gaze [38]. Recent works [19, 21] have demonstrated the effectiveness of human assistance. For example, HIL-SERL achieves state-of-the-art performance in vision-based real-world reinforcement learning, while Sirius integrates human trust signals to enhance behavioral cloning. The HitL systems is also extensively applied in autonomous driving. For example re-routing systems from ZOOX [24] enable human operators to intervene when vehicles encounter challenging scenarios.

However, these approaches often focus on incorporating human feedback during training while neglecting the critical question of when human assistance is most necessary during deployment. They typically require continuous human supervision, which is inefficient and impractical in real-world settings. In contrast, our work introduces an uncertainty-aware diffusion model that actively identifies deployment scenarios where human expertise is most beneficial. By requesting assistance only during high-uncertainty states, we enable a more efficient and scalable HitL framework that minimizes the reliance on constant human oversight. The closest work to ours is HULA [30], a reinforcement learning approach that explicitly models the uncertainty of an agent. However, HULA has limited application in the setting of online reinforcement learning, while our works focus on deriving HitL policies from offline dataset.

In this work, we leverage diffusion models to produce HitL policies. Diffusion models have recently achieved remarkable success in robotics and decision-making tasks [2, 7, 26, 28, 32, 34, 37]. These models are particularly adept at capturing complex behaviors and multi-modal trajectory distributions when trained on high-quality demonstration data. However, collecting perfect demonstration datasets is often infeasible due to limitations in data quality and the prevalence of suboptimal demonstrations. To address this, researchers have proposed methods such as guiding the denoising process with external objectives like reward signals or goal conditioning [1, 5, 13, 18, 33]. Other strategies include integrating Q-learning and weighted regression in offline [6, 35] or online [11, 14, 27] settings. Our work takes a novel approach by leveraging the inherent distribution modeling capabilities of diffusion models to quantify uncertainty in action modes for each state. This allows us to identify high-uncertainty states where human intervention is most impactful. By targeting these critical states, we create a more efficient and focused human-in-the-loop policy deployment framework, enhancing both performance and scalability.

III. Method

We start from the basic premise of HitL robotic agents: the robot is expected to generally act autonomously, but a human operator is available to provide teleoperation commands should the robot require them. Our method is designed to determine when the agent should request such expert assistance, making effective use of a limited number of such calls during deployment. Furthermore, since we use a learning agent, we aim to eliminate the need for expert intervention during the training phase, as that would likely place a large burden on the operator. This means that the agent has no quantitative knowledge about the effect of assistance, except for the assumption that it effective for task completion.

To achieve this, we utilize diffusion models [7] as our policy class. Diffusion policies offer two key advantages: (1) they have demonstrated robust performance in imitation learning tasks, and (2) their generative process involves an iterative denoising mechanism which we can leverage for insight into the agent's decision-making process. Specifically, we use the denoising information to compute an uncertainty metric for the policy, which is then used during deployment to determine when human intervention is most beneficial. Importantly, the policy is trained using an offline dataset and does not rely on human assistance during the training phase. Finally, we show that data collected during the operator interventions can be incorporated back into training through a fine tuning process that further improves policy performance.

In this section, we first re-iterate general diffusion policies concepts, describe how our method utilizes their generative process to compute an uncertainty metric, and finally discuss how this metric is applied to improve policy deployment performance.

A. Background: diffusion policies

Diffusion policies generate actions through an action-denoising process, leveraging denoising diffusion probabilistic models (DDPM). A DDPM models a continuous data distribution $p(x^0)$ as reversing a forward noising process from x^0 to x^K, which is defined as a Markovian chain with Gaussian transition. Here, x^K represents a sample of $\mathcal{N}(0, \sigma^2\mathcal{I})$. To generate actions, the model learns to predict the transition (i.e. noises added during each step) and map the data from x^K back to x^0. Sampling begins with a random Gaussian noise and iteratively refines it to produce a denoised output.

Specifically, the generative process of a diffusion policy $\pi(\mathrm{A}|\mathrm{O})$, where O and A are observations and actions, starts by sampling a random noise a_t^K and iteratively remove noises by:

$$a_t^{k-1} = \beta(a_t^k - \gamma\epsilon_\theta(o_t, a_t^k, k) + \mathcal{N}(0, \sigma^2\mathcal{I})),$$

where β, σ and γ are functions of iteration step k. To train a diffusion policy, we learn a score function ϵ_θ that predicts the noises used during forward noising:

$$\mathcal{L} = MSE(\epsilon^k, \epsilon_\theta(o_t, a_t^k + \epsilon^k, k)) \tag{1}$$

B. Denoising-based uncertainty metric

To estimate the uncertainty of a diffusion-based agent, our method leverages the generative process inherent in the diffusion policy. In particular, we note that, when operating in task-space control (where the action space consists of desired end-effector poses), the noise predicted during the generative process can be interpreted as a vector field pointing toward the target distribution in the next state. We can thus leverage this vector field to analyze whether a diffusion-based agent is confident about its generative target.

We assume that our policy is operating on task space control and a diffusion policy outputs absolute end-effector (e.e.) poses and manipulator state. We also assume that the current e.e. pose is available as an input for action denoising. Our goal is thus to estimate an uncertainty metric $\texttt{Uncertainty}(o_t)$ where o_t is the observation at time step t.

To compute this metric, we first sample a set of points $A_{sampled}$ within a distance r from the current pose. For each sampled point, we feed the diffusion policy forward and predict the noise vectors required to sample actions:

$$v = \epsilon_\theta(o_t, a_{sampled}, 0) \tag{2}$$

These predicted noise vectors represent the directions toward the data distribution that the policy aims to recover. In this work, we use these denoising vectors to estimate uncertainty, defined as $\texttt{Uncertainty}(o_t) = f(V_t)$, where V_t represents the set of denoising vectors at time step t.

A critical characteristic of diffusion policies is their ability to capture multi-modality in the underlying human demonstrations (starting from a given state, there might be multiple distinct action trajectories that accomplish the desired task). Thus, at any given moment, the denoising vector field could reflect the multi-modal nature of the data, and naive variance estimation of the vector field may fail to capture this effect.

To address this, we use Gaussian Mixture Models (GMMs) to capture the potentially multi-modal nature of action generation. Our method, outlined in Algorithm 1, starts by fitting the collected denoising vectors with N GMMs, each using a different number of modes. We then select the best-fit GMM for uncertainty estimation via maximum likelihood estimation:

$$\max_{n, \theta_{\text{gmm}}} P(V_t; n, \theta_{\text{gmm}}),$$

where n is the number of modes and θ_{gmm} is the parameters of a Gaussian mixture model. With the best-fit GMM, we then can estimate an agent's uncertainty at state s. We first evaluate the divergence between each mode:

$$D(V_t) = \frac{1}{n(n-1)} \sum_{i,j} 1 - S_c(g_i, g_j)$$

where,

$$S_c(g_i, g_j) = \frac{g_i \cdot g_j}{||g_i|| \cdot ||g_j||}$$

and g_i represents the mean of the i^{th} mode and S_c represents cosine similarity between two vectors. We also evaluate the GMM variance as part of the uncertainty estimation:

$$\texttt{Var}_g(V_t) = \sum_i p(v_i)\texttt{Var}(v_i) \tag{3}$$

where $\texttt{Var}$ represents the variance of vector data. Putting them together, we can estimate the overall uncertainty as:

$$\texttt{Uncertainty}(o_t) = D(V_t) + \alpha\texttt{Var}_g(V_t), \tag{4}$$

where α is a constant. This uncertainty estimation considers two aspects during denoising: how diverged the target distributions are, and how much entropy there is in each of the modes.

C. Uncertainty-based intervention and policy fine-tuning

Having defined our uncertainty metric, we can use it during deployment by setting a threshold to determine whether we to request human assistance. At every state, the agent computes its own uncertainty and, if the level of uncertainty exceeds the threshold, the agent requests that the operator take control and teleoperate the system for several steps, until uncertainty returns below the threshold.

Algorithm 1 HitL Policy Deployment

```
1:  while rollout not done do
2:      Sample a set of points uniformly within the radius of $r$
3:      Feed forward the diffusion policy $\pi$ to collect a set of denoising vectors $V_t$
4:      Estimate uncertainty in this state using Eq.4
5:      if $\texttt{Uncertainty}(o_t) \geq \texttt{threshold}$ then
6:          Execute $m$ steps of human input actions $a_{human}$
7:          Save intervention data $\{o, a_{human}\}^m$ to $\mathcal{D}_{ft}$
8:      else
9:          Execute an action $a_t$ from the policy $\pi(a_t|o_t)$
10:     end if
11: end while
12: if fine-tune then
13:     while fine-tuning not done do
14:         Sample a batch of data from $\mathcal{D}_{ft}$ and $\mathcal{D}_{train}$
15:         Optimize $\pi$ using Eq. 1
16:     end while
17: end if
```

In addition, our method can also be used to collect data to further fine-tune the policy. This allows for better performance in the next policy execution. To fine-tune a policy, we save the observation and action pairs $\{\mathrm{O}, \mathrm{A}\}$ when a human operator is intervening with the robot and use this data set to fine-tune the underlying diffusion policy. To avoid catastrophic forgetting [3], we sample from both the fine-tuning dataset $\mathcal{D}_{ft}$ and pretraining dataset $\mathcal{D}_{train}$. For each mini-batch, we ensure 50% are from $\mathcal{D}_{ft}$. Our approach implicitly means that this fine-tuning data specifically addresses the areas of state space where the agent's uncertainty is high, since that is where operator assistance is requested.

Putting all components together, the final pipeline contains three main steps: 1. train a diffusion policy; 2. deploy the policy, and request operator control whenever policy uncertainty estimated by our metric exceeds a pre-set threshold; 3. (optional) use the human intervention data to fine-tune the diffusion policy.

To envision the intended applicability of this framework, let us consider three types of deployment issues that typically cause uncertainty for learning-based agents:

- Case 1: Data distribution shift. For example, visual observation distribution shift can be caused by change of lighting condition. A special case for robotics is the change of dynamics that is caused by interacting with novel objects.
- Case 2: Incomplete state observability. The common approach to address this can include redesigning, adding or moving sensors. However, in the general case, it is very difficult to have a sensor setup that guarantees full observations for all the tasks.
- Case 3: Incorrect choice between different action modes. In cases where the agent is presented with a discrete choice between two possible action trajectory modes equally represented during training, diffusion policies are naturally well-equipped to make such choices. However, it is possible that, due to task under-specification, the choice will be incorrect for the given goal.

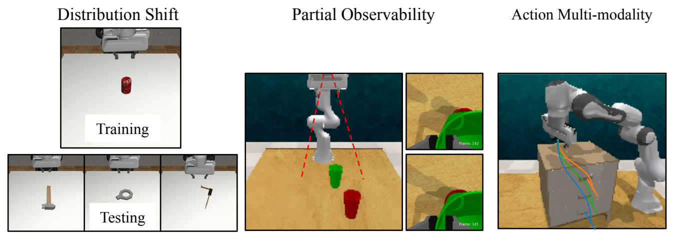

Fig. 2. **Simulated environments.** We considers three major problems during policy deployment. (a) Distribution shift; (b) Partial observability (c) Action multi-modality.

During policy execution, these problems may not be present in all states. In fact, many states are easy to make decisions. For example, moving the arm in free space is often easy and does not require human attention to help the robot. Our first goal is to identify when the issues arise, and request help at that time. We posit that, by combining uncertainty due to variance within a mode with uncertainty due to presence of multiple modes, our framework is positioned to recognize all three cases above. Furthermore, we believe that a HitL framework is particularly well equipped to handle Case 3, since a few steps under teleoperator control can "steer" the policy towards the desired mode. Case 1 lends itself well to fine-tuning based on the novel data collected during teleoperation. Finally, Case 2 is the most difficult, since, by definition, correct decision making is impossible without changing the available observation. We design our experiment set to test a range of scenarios covering these situations.

IV. Experiments

We validate our method on the three types of deployment problems discussed above, conducting experiments in both simulated environments and real-world robotic setups. For most of our test cases, full teleoperation is generally able to successfully complete the task; therefore, with sufficient human intervention and assistance, the success rate can reach 100%. However, a critical aspect of human-in-the-loop deployment is the efficiency of human intervention – the robot should request assistance conservatively, minimizing unnecessary interruptions.

To evaluate the effectiveness of our method, we thus focus on two key aspects across all experiments. First, we assess the efficiency of human-robot interaction by measuring the frequency of human interventions. Second, we evaluate the improvement in task performance achieved through human assistance and policy fine-tuning, quantifying the benefits of integrating human feedback into the system.

We note that most offline policy learning methods do not consider HitL deployment during training, and are thus not comparable to our method. To benchmark our method, we

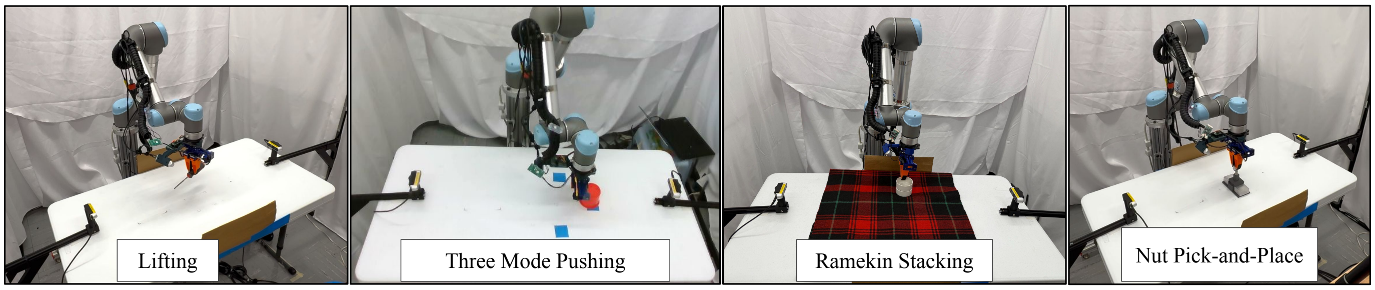

Fig. 3. Real robot experiments.

compare it against two state-of-the-art baselines for incorporating uncertainty estimation into HitL operation. The baselines considered in this work are as follows:

- **Gaussian ResNet18 [12] Multi-Layer-Perception (MLP).** This baseline uses Gaussian MLP as the policy class for imitation learning. While this approach is not necessarily developed with a HitL framework, it is one of the few learning approaches that offers a natural uncertainty metric, which could be used in our contect. During deployment, we use the entropy of the policy output as an uncertainty measurement for HitL polices. We also add an entropy term to regularize the training.
- **HULA-offline [30].** HULA is a HitL policy learning method based on reinforcement learning. It uses an explicit uncertainty metric based on the variance of the estimated reward at any given time step. However, the original HULA method is designed for online RL. In order to make it directly comparable to our method, where only offline datasets are available during training, we adapt it to offline RL by implementing an offline variant using Conservative Q-Learning (CQL) [15]. We also augment the dataset by labeling rewards if they are not available from the original environment: each step receives a reward of 1 if it is the final step of the trajectory and 0 otherwise.

A. Environments

In this section, we briefly summarize the environments that we test our method on in both simulated and real world setting. Details about each environment and statistics of collected data from them are included in the supplementary materials [ref].

Distribution shift: Lift-sim. we first consider the deployment problem of distribution shift. In this task, we ask the robot to grasp and lift objects in a table-top setting. To emulate distribution shift, demonstration data is collected using only a single object (red cube - see Fig.4), while for testing we roll out the pretrained policy to a set of unseen objects (round nuts, hammers, and hooks).

Partial observability: Cup Stacking. we then test our method on problems with partially observable environments. In this task, we ask the robot to grasp a green cup and place it inside a red cup. In this task, we use three views as our visual observation: a front view, a side view and a wrist view. Successful execution requires the robot to infer object alignment based on its observations. Misalignment can lead to unintended collisions, resulting in failures such as the red cup tipping over or becoming unstable. To introduce variability and train a robust policy, cup positions are randomized during data collection.

Incorrect choice of action mode: Open drawer. Here, the robot is tasked with opening one of three drawers in the scene. The collected dataset includes trajectories for opening each drawer, with approximately 33.3% of the data corresponding to each drawer. However, the dataset does not specify which drawer is to be opened in a given trajectory, introducing under-specification in the dataset.

B. Real robot experiments

Finally, we validate our method on real robot data collected using tele-operation. In this work, we use a tele-operation system to collect human demonstration data. The robot is controlled by a trakSTAR electromagnetic 6DoF pose tracker and a gripper control unit that provides continuous commands. To meet the need for real robot deployment, we use denoising diffusion implicit models (DDIM) that allow for high-frequency action generation. Since DDIM can be used with a DDPM, our method can be directly applied to a DDIM model. Unlike DDPM, our vector field sampling can be parallelized and batched feed-forward with the model. Hence, this additional computation does not add a big overhead during policy deployment. Details about hyper-parameters used with DDIM can be found in Appendix.A.

We evaluate our method on 4 real robot tasks (see Fig. 3). As in the simulated experiments, we show an example in each of the deployment problems.

Lift-real. This task extends the lift-sim setup to the real world, where the robot is trained to grasp a set of objects (Fig. 5) and tested on unseen objects to evaluate generalization. Although some objects from train and test set are visually similar, the robot needs to learn different strategies to grasp them. For example, the cup we use for testing has a higher radius than the one in the training set, making it difficult to cage with a gripper. The robot needs to learn to grip the rim of the cup to achieve stable grasps.

3-Mode Pushing. In this task, the robot is required to push a cup to three designated locations on the table, marked with blue tape. The dataset used for training includes equal proportions of pushing trajectories for each of the three locations, with

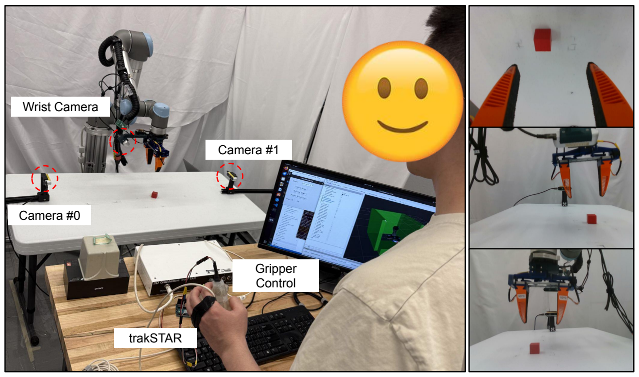

Fig. 4. Data collection pipeline.

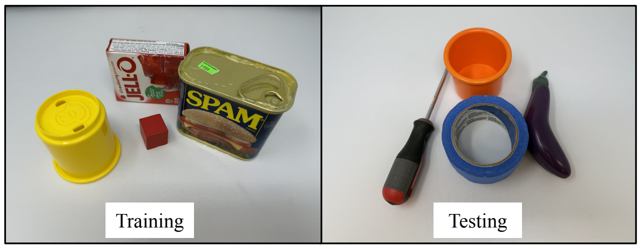

Fig. 5. Objects used for the Lift-real task. The left and right images show training and testing objects respectively.

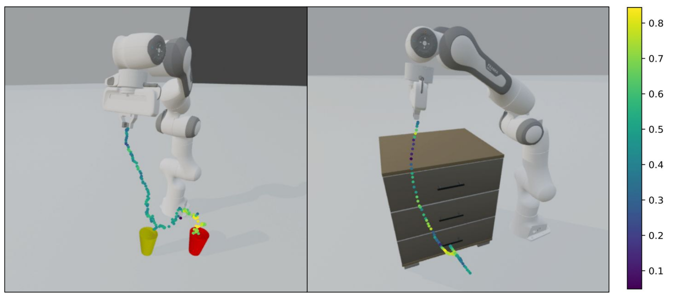

Fig. 6. Qualitative results for uncertainty estimation of the learned policy with the simulated task. The color of the trajectory represents the uncertainty estimation of the agent. Each dot represents an action prediction from the policy.

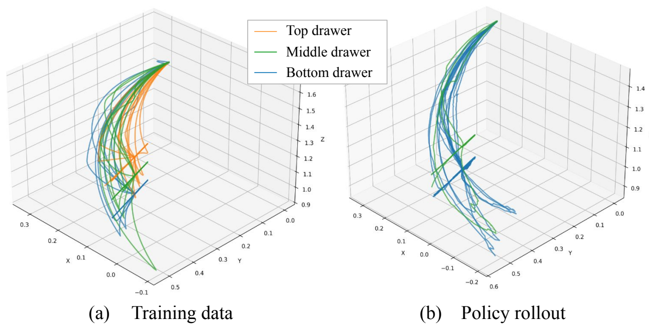

Fig. 7. Multi-modal demonstrations and learned modes from diffusion policy in the Open Drawer task.

each mode a third of the full dataset. Notably, during training, the specific target location (mode) for the push is not explicitly provided to the robot.

Ramekin Stacking. In this task, the robot is required to pick up one ramekin and place it on top of the other. The success criterion for this task is achieving stable placement, where the top ramekin is horizontally aligned with the bottom one. This requires precise alignment between the two ramekins. As shown in Fig. 9, a key challenge arises during stacking because the bottom ramekin becomes occluded in the wrist camera view, making it difficult for the robot to achieve proper alignment for stable placement.

Nut Pick-and-Place. The final task evaluates our method on a pick-and-place scenario where the robot must pick up a nut and place it onto a lug. The success criterion for this task is achieving a stable placement of the nut when the gripper is open, ensuring it is properly aligned for threading. This task demands high placement precision, which is particularly challenging to achieve using visual observations alone.

V. RESULTS

A. *Task performance*

As a sanity check, we first evaluate the task performance of the diffusion policy on each task under the training data distribution. For the Lift-sim task, diffusion policy achieves a 100% success rate with the training object but fails completely (0% success rate) when tested on unseen objects.

In the Cup Stacking task, the robot consistently picks up the first cup (100% grasp success rate) but fails to place it into the second cup due to alignment difficulties, resulting in an overall success rate of 0% without human assistance. We also note that this task is sensitive to observation selection—training with only side and front views causes the robot to fail at grasping the green cup.

For the Open Drawer task, the diffusion policy successfully learns to open a drawer with 100% success if the task description does not specify which particular drawer should be opened. Interestingly, despite the under-specified training (i.e., no conditioning on which drawer to open), the policy captures the multi-modality of the training distribution. As shown in Fig. 7, during 100 rollouts with random sampling, the robot opens the middle and bottom drawers in 15% and 85% of trials, respectively, but never opens the top drawer.

B. *Efficiency of human interaction*

We then evaluate human-in-the-loop deployment performance of these tasks. We note again that 100% success rate is always possible with sufficient human assistance. Thus, we focus here on achieving high success rates with as few human assistance steps as possible, which is a critical aspect for real-world scalability of HitL systems.

As shown in Table I, for all simulated tasks, our method allows the policy to achieve high task success rates with a small number of teleoperation steps. For the Lift-sim task, the robot only seeks human assistance when its gripper is

TABLE I
AVERAGE # OF HUMAN ASSISTANCE STEPS FOR SIMULATED TASKS

	Lift-sim	Cup stacking	Open drawer
Gaussian ResNet18-MLP	74	42	56
HULA-offline	56	54	22
Denoising Uncertainty (ours)	17	5	8
Avg. Full-trajectory length	77	147	115

TABLE II
HITL CONTROL EFFICIENCY FOR REAL ROBOT TASKS: AVERAGE # OF HUMAN ASSISTED STEPS DURING POLICY DEPLOYMENT

	Lift-real	Ramekin stacking	3-Mode Pushing	Nut PnP
Denoising Uncertainty (Ours)	7.2	6.8	6.5	8.4
Avg. Full-trajectory length	80	112	99	49

close to the object, allowing the human to pose the gripper to locations that leads to successful grasp. After the human assists the robot for grasping, the robot can lift the object without any human intervention. In the Cup Stacking task, our method identifies states where the agent fails to align the two cups as having high uncertainty. In contrast, states where the robot needs to pick up the green cup, and can rely on a full observation to complete the grasp process, are marked as having low uncertainty. Finally, for the Open Drawer task, experiments show that our method allow users to choose which drawer to open, or, in other words, which of the learned modes to execute. The policy asks for assistance when it is deciding which drawer to reach to, and, once the human operator steers it to a certain height, the robot autonomously completes the rest of the task. Compared to the baselines, our method can consistently complete the task with fewer human interventions. The Gaussian ResNet18-MLP baseline requires frequent human assistance due to its reliance on entropy minimization for effective action generation. Striking a balance between the entropy term and the log-likelihood objective is challenging, making it difficult to be used as a HitL policy.

C. Fine-tuning performance

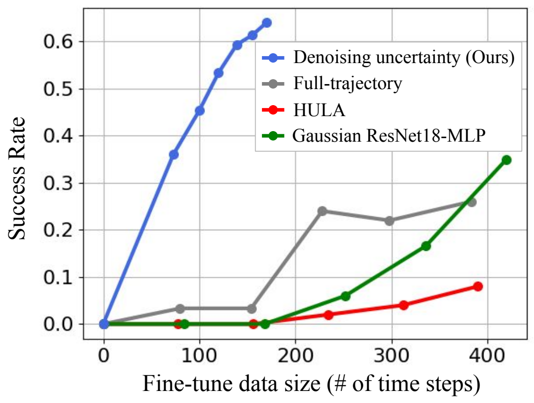

Fig. 8. Average success rate of fine-tuning the Lift-sim task with different number of human demonstration samples.

Our method requests operator assistance in states where the policy exhibits high uncertainty. We posit that these states are particularly valuable as they highlight areas where the policy can benefit from additional data collection for fine-tuning. We test this hypothesis by checking if leveraging our uncertainty metrics reduces the amount of data required for fine-tuning, while still achieving significant performance improvements.

TABLE III
FINE-TUNING PERFORMANCE OF THE LIFT-REAL TASK. RESULTS ARE SUCCESS RATES DERIVED BY 20 POLICY ROLLOUTS PER OBJECT.

	Train	Test (↑)	$\|\|\mathcal{D}_{ft}\|\|$ (↓)
Zero-shot	1	0.16	0
HitL fine-tuning	1	0.63	80
Full-trajectory fine-tuning	1	0.31	132

As shown in Fig. 8, our policy performance improves by 63.3% on average if we fine-tune the policy with 170 teleoperator-led time steps obtained specifically when uncertainty is high. In contrast, the policy success rate only improves by 28% if we fine-tune using full trajectories comprising 383 teleoperator time steps. The HULA-offline baseline only improves its success rate by 8% on average. We also observe that HULA is more prone to over-fitting in both pretraining and fine-tuning. It can complete the task in locations seen in the training set, but fails when the object locations are different, even if close to locations in the training set. This shows that our method can identify states that the policy needs more information.

Compared to the baselines, our method also consistently achieves higher success rates with the similar amount of fine-tuning data. For example, while using approximately 100 time steps of fine-tuning data, our method achieves 45.3% success rate improvement while all baselines achieve less than 10% improvement. We note that the data used for each fine-tuning experiment is collected independently (i.e. the human-in-the-loop fine-tuning data set is not a part of the full-trajectory data set). For the HitL fine-tuning, the fine-tuning dataset only consists of actions when the robot is operated by human operators, instead of full trajectories.

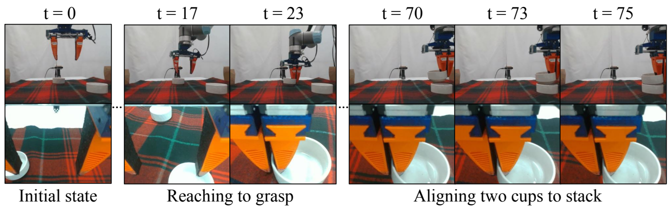

Fig. 9. Examples of observations for Ramekin Stacking in different stages. The top row shows image frames from one of the side cameras and the bottom row shows the view of the wrist camera.

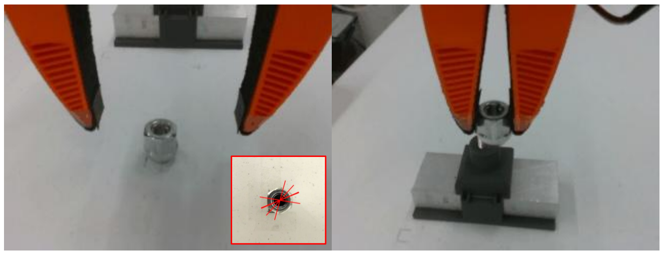

Fig. 10. Two examples of states with high uncertainty for the Nut PnP task: when the robot is posing for grasping the nut (left) and trying to alighn the nut with the lug (right). The image highlighted in red shows rotation of the gripper around z axis while rolling out the policy with the same initial pose of the nut. The background nut image is used only for visualization, not observations.

D. Evaluation on real robot experiments

Finally, our results on real robot experiments shows that our method can be used with policy trained with real world data. Again, with human-assisted deployment, the robot can complete all four tasks. On average, our method only requests help from the human for approximately 8.3% of time steps during policy execution (see Table II). Since we are using action chunking during real robot deployment, we allow the human to control the robot four steps, the same as the diffusion policy. For all tasks, the robot asks for assistance for less than 6 times (each assistance with 3 human control steps). Details about the evaluations protocol (e.g. the number of rollouts for each evaluation) are included in supplementary materials.

Qualitatively, our method identifies crucial states during policy execution. For example, in the Lift-real task, the robot asks for assistance when the gripper is close the the object. Using human-collected data with uncertainty, we can fine-tune the diffusion policy to improve 47% success rate on average (shown in Table III), outperforming fine-tuning with full-trajectories of data. In the 3-Mode Pushing task, the robot autonomously reaches to the side of the object and then transfers control to the human operator, who poses the gripper in the correct location depending on the intended target. Once the pose of the gripper is indicative of the desired target, uncertainty drops, and the robot takes over and completes the task autonomously. In the Ramekin Stacking task, our method identifies high-uncertainty states when the robot is attempting to align the ramekins and the bottom one is visually occluded (see Fig. 9). In contrast, grasping the first ramekin – where visual observations suffice – is marked as low-uncertainty and thus performed autonomously.

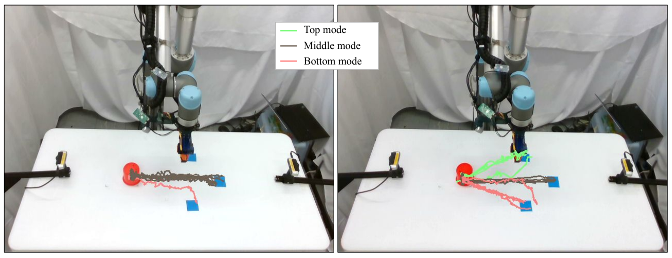

Fig. 11. Object trajectories visualization: policy random rollout vs policy steering.

TABLE IV
EFFECT OF SAMPLING RADIUS ON FINE-TUNING PERFORMANCE OF THE LIFT-SIM TASK.

Radius of sampling	0.01	0.03	0.05	0.1
# of human steps (↓)	60.3	31.6	**20**	46.3
Success rate (↑)	0.46	0.55	**0.63**	0.53

TABLE V
EFFECT OF SCALING CONSTANT α ON FINE-TUNING PERFORMANCE OF THE LIFT-SIM TASK.

α	0.01	0.05	0.1	0.3	0.5
# of human steps	5	5	5	7	7
index of max uncertainty	139	139	139	137	137

Finally, in the Nut Pick-and-Place task, our method assigns high uncertainty to two critical stages of execution. The first stage occurs during the positioning phase for grasping, where the dataset contains diverse strategies for aligning the gripper with the nut's edge. Interestingly, when rolling out the policy multiple times with identical initial nut configurations, the robot exhibits varying rotations around the z-axis to grasp the nut (see Fig. 10). This highlights the ability of our uncertainty metrics to capture critical decision points during execution. The second stage of high uncertainty arises during the placement phase, where precise positioning of the nut is required. The visual observations from the wrist camera and the two side cameras fail to reliably determine the stability of the placement, resulting in elevated uncertainty during this stage. The agent thus requests operator assistance for task completion.

E. Hyperparameter exploration

In this section, we investigate how hyper-parameters affect the performance of our HitL agent. The choice of these hyper-parameters plays an important role of our uncertainty estimation, and hence can affect how the robot ask for human assistance.

1) Sampling Radius: The radius parameter r defines the neighborhood size for collecting denoising vectors in uncertainty estimation. It is crucial to consider the scale of r relative to the action distribution that a diffusion policy aims to recover. If r is too large, the uncertainty metric becomes similar across all states, as it incorporates a broad range of them. Conversely, if r is too small, the metric may be overly influenced by local uncertainty. In this work, we normalize each dimension of the action space to $[0, 1]$, simplifying the selection of r.

In HitL fine-tuning for Lift-sim (see Table V), we test radii ranging from 0.01 to 0.1. A radius of 0.05 achieves the best balance between intervention steps and success rate, requiring only 20 interventions while improving the success rate by 0.63. This efficiency results from accurate uncertainty detection when the gripper approaches unseen targets but fails to grasp them. Smaller radii and larger radii yield less precise estimations, leading to interventions that are either premature or delayed, reducing success rates despite more interventions.

Overall, a radius of 0.05 consistently achieves optimal performance by balancing local and global uncertainty estimations. Smaller radii miss key trajectory patterns, while larger radii incorporate irrelevant vectors, reducing the accuracy of uncertainty estimation in robotic manipulation tasks.

2) Scaling constant α in uncertainty calculation: The alpha parameter serves as a scaling factor in our uncertainty calculation, balancing two components: mode divergence and overall variance. Mode divergence captures directional differences between action modes, while overall variance measures spread within each mode.

Directional differences typically provide stronger uncertainty signals, and small α values (0.01 – 0.1) emphasize mode divergence, effectively identifying critical occlusion phases (e.g., step 139). This highlights mode divergence as a reliable indicator of uncertain states. As α increases, the overall variance term gains more weight, raising both maximum and minimum variance values. However, this added emphasis on within-mode spread does not significantly enhance uncertain state detection, supporting the dominance of mode divergence as the more informative component. The variance term remains essential for distinguishing states with single action mode (e.g. reaching for grasping), where mode divergence alone would yield identical scores. While α influences absolute uncertainty values, it minimally affects the identification of critical steps (consistently around steps 137–139 in cup stacking).

VI. Limitations

Our work investigates methods for seeking human assistance in a non-interruptive manner. We are primarily interested in automatically determining, at deployment time, when such assistance is needed, without overloading the operator with requests. However, we do not tackle other important aspects of a full HitL framework which we consider complementary to our work.

For example, we do not consider methods for conveying to the operator additional information regarding the sources of the uncertainty identified by the agent. In this work, there is no mechanism for identifying which deployment difficulty is affecting the task. We empirically observe that analyzing uncertainty from different dimensions of the denoising vector can sometimes provide insights into the problem's source. For instance, high uncertainty in position versus rotation may indicate distinct underlying challenges. However, we do not have a formal way of conveying such information to the operator.

Furthermore, we do not focus on the specific teleoperation apparatus available to the operator. In this work, we use keyboard control to produce desired changes in end-effector pose and gripper closing commands. We observe that this method requires high expertise on the part of the operator. Future work can investigate the interplay between the method for deciding when to request assistance and the teleoperation tools available to provide such assistance.

VII. Conclusions

In this work, we propose a novel method that enables a robot to actively and efficiently request HitL assistance during deployment. By using an uncertainty metric based on the denoising process inherent to diffusion policies, our method identifies situations where human intervention is most beneficial, thereby reducing unnecessary monitoring and intervention. Experimental results demonstrate the versatility of our method across various deployment scenarios, improving policy performance and thus adaptability in real-world conditions. This work aims to address one of their key challenges in human-in-the-loop robot deployment: minimizing human labor while maximizing robot autonomy and reliability. Additionally, our approach highlights the potential for using such interaction-driven methods to refine and fine-tune policies through targeted data collection.

For future work, we aim to further automate this process by exploring what types of information most effectively facilitate human-robot communication. Specifically, we will investigate how to design interpretable feedback that allows robots to convey their uncertainty and intent in a manner that is intuitive for human operators. Furthermore, we will study advanced control mechanisms that enable humans to seamlessly intervene and guide the robot when necessary. These efforts will help bridge the gap between fully autonomous systems and human-in-the-loop deployment, enabling more efficient and scalable solutions for real-world robotic applications.

References

[1] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

[2] Lars Ankile, Anthony Simeonov, Idan Shenfeld, and Pulkit Agrawal. Juicer: Data-efficient imitation learning for robotic assembly. *arXiv preprint arXiv:2404.03729*, 2024.

[3] Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pages 1577–1594. PMLR, 2023.

[4] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 783–792. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/brown19a.html.

[5] Boyuan Chen, Diego Marti Monso, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *arXiv preprint arXiv:2407.01392*, 2024.

[6] Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. *arXiv preprint arXiv:2209.14548*, 2022.

[7] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[8] Open X-Embodiment Collaboration. Open X-Embodiment: Robotic learning datasets and RT-X models. https://arxiv.org/abs/2310.08864, 2023.

[9] Yang Cong, Ronghan Chen, Bingtao Ma, Hongsen Liu, Dongdong Hou, and Chenguang Yang. A comprehensive study of 3-d vision-based robot manipulation. *IEEE Transactions on Cybernetics*, 53(3):1682–1698, 2021.

[10] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022.

[11] Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

[14] Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[15] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[16] Kimin Lee, Laura Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. *arXiv preprint arXiv:2106.05091*, 2021.

[17] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, et al. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

[18] Zhixuan Liang, Yao Mu, Mingyu Ding, Fei Ni, Masayoshi Tomizuka, and Ping Luo. Adaptdiffuser: Diffusion models as adaptive self-evolving planners. *arXiv preprint arXiv:2302.01877*, 2023.

[19] Huihan Liu, Soroush Nasiriany, Lance Zhang, Zhiyao Bao, and Yuke Zhu. Robot learning on the job: Human-in-the-loop autonomy and learning during deployment. In *Robotics: Science and Systems (RSS)*, 2023.

[20] Peiqi Liu, Yaswanth Orru, Chris Paxton, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Ok-robot: What really matters in integrating open-knowledge models for robotics. *arXiv preprint arXiv:2401.12202*, 2024.

[21] Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning, 2024.

[22] James MacGlashan, Mark K Ho, Robert Loftin, Bei Peng, Guan Wang, David L Roberts, Matthew E Taylor, and Michael L Littman. Interactive learning from policy-dependent human feedback. In *International conference on machine learning*, pages 2285–2294. PMLR, 2017.

[23] Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Yuke Zhu, Li Fei-Fei, and Silvio Savarese. Human-in-the-loop imitation learning using remote teleoperation. *arXiv preprint arXiv:2012.06733*, 2020.

[24] Cade Metz, Jason Henry, Ben Laffin, Rebecca Lieberman, and Yiwen Lu. How self-driving cars get help from humans hundreds of miles away. *New York Times*, 2024. URL https://www.nytimes.com/interactive/2024/09/03/technology/zoox-self-driving-cars-remote-control.html.

[25] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot

policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

[26] Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, et al. Imitating human behaviour with diffusion models. *arXiv preprint arXiv:2301.10677*, 2023.

[27] Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. *arXiv preprint arXiv:2312.11752*, 2023.

[28] Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. *arXiv preprint arXiv:2304.02532*, 2023.

[29] Nur Muhammad Shafiullah, Zichen Cui, Ariuntuya Arty Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *Advances in neural information processing systems*, 35:22955–22968, 2022.

[30] Siddharth Singi, Zhanpeng He, Alvin Pan, Sandip Patel, Gunnar A. Sigurdsson, Robinson Piramuthu, Shuran Song, and Matei Ciocarlie. Decision making for human-in-the-loop robotic agents via uncertainty-aware reinforcement learning. In *International Conference on Robotics and Automation*, pages 7939–7945. IEEE, 2024.

[31] Jonathan Spencer, Sanjiban Choudhury, Matthew Barnes, Matthew Schmittle, Mung Chiang, Peter Ramadge, and Siddhartha Srinivasa. Learning from interventions: Human-robot interaction as both explicit and implicit feedback. In *16th Robotics: Science and Systems, RSS 2020*. MIT Press Journals, 2020.

[32] Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. Nomad: Goal masked diffusion policies for navigation and exploration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 63–70. IEEE, 2024.

[33] Siddarth Venkatraman, Shivesh Khaitan, Ravi Tej Akella, John Dolan, Jeff Schneider, and Glen Berseth. Reasoning with latent diffusion in offline reinforcement learning. *arXiv preprint arXiv:2309.06599*, 2023.

[34] Lirui Wang, Jialiang Zhao, Yilun Du, Edward H Adelson, and Russ Tedrake. Poco: Policy composition from and for heterogeneous robot learning. *arXiv preprint arXiv:2402.02511*, 2024.

[35] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

[36] Jimmy Wu, William Chong, Robert Holmberg, Aaditya Prasad, Yihuai Gao, Oussama Khatib, Shuran Song, Szymon Rusinkiewicz, and Jeannette Bohg. Tidybot++: An open-source holonomic mobile manipulator for robot learning. In *Conference on Robot Learning*, 2024.

[37] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. In *ICRA 2024 Workshop on 3D Visual Representations for Robot Manipulation*, 2024.

[38] Ruohan Zhang, Akanksha Saran, Bo Liu, Yifeng Zhu, Sihang Guo, Scott Niekum, Dana Ballard, and Mary Hayhoe. Human gaze assisted artificial intelligence: A review. In *IJCAI: Proceedings of the Conference*, volume 2020, page 4951. NIH Public Access, 2020.

Appendix

A. Training details and data statistics

In this section, we detail the training and implementation setup for each experiment. The hyperparameters used during pretraining are provided in Table VII. We employ a transformer-based diffusion policy (DP-T) [7], which yields stable performance across the tasks tested. For each experiment, we use the best-performing policy from pretraining. We also note that the selection of specific observations is crucial for task learning. For instance, in the Cup Stacking task, the policy fails to learn how to grasp the green cup without including the wrist view. Additionally, we observe that policies with lower performance typically require more human assistance. Parameters for policy rollouts are also listed in Table VII.

B. Human-in-the-loop policy deployment

In this work, a human operator controls the robot via keyboard inputs during high-uncertainty states. Specifically, the operator provides a delta pose and gripper control, with each assistance event allowing control for 3 or 4 steps. The results presented in this paper reflect human assistance commands rather than the number of expert calls, although these can be easily converted to the amount of expert calls if needed.

C. Instability of HULA

In this work, we extensively test the performance of our baselines to ensure fair comparisons. We find that the training of HULA is sensitive to hyper-parameters. The underlying cause is the requirement of training a good Q-function in the base RL algorithm CQL. For example, Fig. 12 shows an example of training curves of one of our failed HULA training. Since the uncertainty estimation is dependent on the estimated Q function, the uncertainty estimation can be instable too.

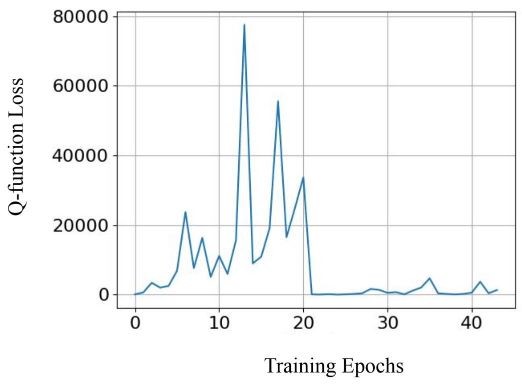

Fig. 12. Example of the training curve for an unstable training with HULA-offline.

	Simulated tasks			Real tasks			
	Lift-sim	Cup stacking	Open drawer	Lift-real	Ramekin stacking	3-mode pushing	Nut PnP
Pretrain dataset size	9666	41670	33617	9600	3360	5940	2450
Observations	WFL	SWFL	FL	Two side cameras views, one wrist view and low-dim obs			
Action chunk Size	1			8			
Obs. history size	1			2			
Image crop size	76 x 76	96 x 96		216 x 288			
Initial object position	randomized		fixed	randomize		fixed	

TABLE VI
TRAINING DETAILS: S, W, F REPRESENTS SIDE, WRIST, FRONT CAMERA VIEWS RECEPTIVELY. L REPRESENTS LOW-DIMENSIONAL OBSERVATIONS THAT CONTAINS END-EFFECTOR POSE AND GRIPPER STATE.