AIDA: Action Inquiry DAgger for Interactive Imitation Learning

Anonymous authors
Paper under double-blind review

Abstract

Human teaching effort is a significant bottleneck for the broader applicability of interactive imitation learning. To reduce the number of required queries, existing methods employ active learning to query the human teacher only in uncertain, risky, or novel situations. However, during these queries, the novice's planned actions are not utilized despite containing valuable information, such as the novice's capabilities, as well as corresponding uncertainty levels. To this end, we allow the novice to say: "*I plan to do this, but I am uncertain.*" We introduce the Action Inquiry DAgger (AIDA) framework, which leverages teacher feedback on the novice plan in three key ways: (1) Sensitivity-Aware Gating (SAG), which adjusts the query threshold to track a desired sensitivity level; (2) Foresight Interactive Experience Replay (FIER), which recasts valid and relabeled novice action plans into demonstrations; and (3) Prioritized Interactive Experience Replay (PIER), which prioritizes replay based on uncertainty, novice success, and demonstration age. Together, these components balance query frequency with failure incidence, reduce the number of required demonstration annotations, improve generalization, and speed up adaptation to changing domains. We validate the effectiveness of AIDA through language-conditioned manipulation tasks in both simulation and real-world environments. Code, data, and videos are available at `https://aida-paper.github.io`.

1 Introduction

The promise of imitation learning is to enable individuals to teach robots to perform desired tasks, all without the need for specialized knowledge in coding or robotics. This learning paradigm is beneficial when humans possess the knowledge to solve a task but prefer not to do it themselves due to its repetitive, risky nature or when automation is more efficient. Specifically, demonstrating correct robot behavior in unstructured environments can be simpler than engineering a controller. Imitation learning has demonstrated success across various domains, such as autonomous driving (Pomerleau, 1988), helicopter aerobatics (Abbeel et al., 2010), language-conditioned robotic manipulation (Jang et al., 2022; Shridhar et al., 2021), and generalist robot policies (Brohan et al., 2022; Reed et al., 2022; Octo Model Team et al., 2024). Despite these successes of *behavioral cloning* (BC), it can suffer from *covariate shift*. This issue arises when imitation learning is naively posed as a standard supervised learning problem. In imitation learning, the data is not independent and identically distributed because past predictions can influence future states (Ross et al., 2011). As a result, a prediction error can lead to encountering states unseen in the training data, causing a cascade of mistakes since errors in these unfamiliar states are even more likely.

Covariate shift issues can be alleviated with Interactive Imitation Learning (IIL) methods (Celemin et al., 2022). These methods involve obtaining human demonstrations, corrections, and reinforcements interactively. In a seminal work, Ross et al. (2011) introduced the Dataset Aggregation (DAgger) algorithm. This approach alleviates the covariate shift problem by aggregating human input while executing the novice policy. This enables the novice to learn to recover from failures, for instance. While having favorable performance guarantees, the DAgger algorithm requires continuous teacher input and can have safety issues as the novice policy is executed while learning to perform the task.

To overcome these limitations of the DAgger algorithm, various extensions allow the novice to *actively* query the teacher in risky (Hoque et al., 2022) or uncertain situations (Menda et al., 2019; 2017; Zhang & Cho, 2017; Hoque et al., 2023). We refer to these data aggregation methods as active DAgger approaches, as they integrate data aggregation with active learning. The benefit of this active learning strategy is twofold. First, possible failures can be prevented during the interactive training phase since uncertainty is assumed to be correlated with failures. Second, this strategy minimizes the number of demonstrations needed by maximizing their meaningfulness. That is to say, it is a waste of resources if humans demonstrate behaviors already mastered by novices. Instead, the teacher should only demonstrate what the robot novice can not do to enable learning from as few demonstrations as possible.

Existing active DAgger methods hand over control when querying the human teacher. Instead, we allow the novice to also communicate their planned action when they are uncertain. This allows the teacher to validate or correct the novice plan, providing valuable feedback that can be leveraged in several ways. First, it reveals the levels of uncertainty where the novice succeeds or fails. This information can be used to improve gating by allowing dynamic threshold adjustments to maintain a desired sensitivity. This extends existing methods, which either require constant supervision (Kelly et al., 2019), rely on heuristics (Zhang & Cho, 2017), or use a fixed query rate independent of novice performance (Hoque et al., 2022). Second, validated novice actions can be aggregated into the demonstration dataset, reducing the need for teacher demonstrations. Additionally, invalid plans may be useful demonstrations for alternative goals if the teacher relabels them accordingly. Third, since not all demonstrations are created equally, we can prioritize replay by considering the validity of the novice plan, the corresponding uncertainty level, and the demonstration age. For example, the novice might learn more from a recent demonstration in a situation where they failed rather than one where they acted successfully.

To this end, we introduce the Action Inquiry DAgger (AIDA) framework, a novel IIL method where the robot novice actively communicates its planned actions when uncertain. An overview of AIDA is shown in Fig. 1, and it is built on three key contributions: i) Sensitivity-Aware Gating (SAG): Adjusts the gating threshold to maintain a desired sensitivity level (true positive rate). ii) Foresight Interactive Experience Replay (FIER): aggregates valid and relabeled novice action plans into the demonstration dataset. iii) Prioritized Interactive Experience Replay (PIER): prioritizes replay based on uncertainty, novice success, and demonstration age.

Since AIDA relies on the novice communicating its planned actions for teacher feedback, we focus on learning mid- to high-level control tasks. AIDA is best suited for scenarios where a robot has access to predefined skills such as grasping, walking, pushing, door opening, screwing, or inserting. When querying the teacher, the robot novice can specify which skill they plan to use along with the parameterization of that skill. If the teacher deems the novice's plan invalid, they can provide a demonstration by annotating the appropriate skill and its parameters. For example, a pick skill can be parameterized by a 3D Cartesian position.

We make the following claims, considering an active data aggregation setting where the teacher can validate novice action plans:

- **C1** SAG balances query count and system failures by tracking a desired sensitivity.
- **C2** FIER reduces the number of annotations needed to achieve a given success rate by recasting novice actions to demonstrations.
- **C3** FIER enhances generalization to unseen scenarios by recasting failures to demonstrations.
- **C4** PIER improves the success rate and reduces the required annotations under domain shift compared to uniform sampling.

The remainder of this paper is structured as follows. Sec. 2 reviews related work. The problem formulation is presented in Sec. 3. Our method is introduced in Sec. 4, followed by its experimental evaluation in Sec. 5. Sec. 6 discusses the results and limitations, and Sec. 7 concludes the paper.

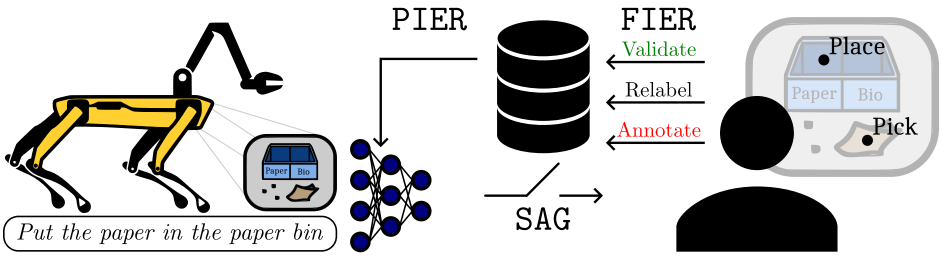


Figure 1: The Action Inquiry DAgger (AIDA) framework consists of three main components: Sensitivity-Aware Gating (SAG, detailed in Sec. 4.1), Foresight Interactive Experience Replay (FIER, detailed in Sec. 4.2), and Prioritized Interactive Experience Replay (PIER, detailed in Sec. 4.3). In this interactive imitation learning framework, we allow the novice to say: "*I plan to do this, but I am uncertain.*" The uncertainty gating threshold is set by SAG to achieve a desired sensitivity level, facilitating the trade-off between queries and failures. Teacher feedback is obtained with FIER, enabling demonstrations through validation, relabeling, or annotation demonstrations. Lastly, PIER prioritizes replay based on novice success, uncertainty, and demonstration age.

2 Related Work

In this section, we will review related work on uncertainty-aware IIL. In a seminal work on IIL with active learning, Chernova & Veloso (2007) introduced the Confidence-Based Autonomy (CBA) algorithm that combined the prediction confidence of a Gaussian Mixture Model (GMM) with the nearest neighbor distance from demonstration data to quantify the confidence of the novice policy. Based on this confidence measure, control is then gated between the novice policy and the human expert. Several related strategies exist, primarily as safety- and/or uncertainty-aware variants of the DAgger algorithm (Ross et al., 2011), which we refer to as active DAgger approaches. Like CBA, these methods apply a form of active learning, i.e., actively querying in situations deemed informative and/or risky. Such techniques include confidence measures based on prediction confidence (Grollman & Jenkins, 2007), maximum mean discrepancy (Kim & Pineau, 2013; Laskey et al., 2016), predicted proximity of novice actions to expert actions (Zhang & Cho, 2017), Monte Carlo dropout (Menda et al., 2017; Cui et al., 2019), ensembles (Menda et al., 2019; Li & Silver, 2023; Li & Zhang, 2023), variational autoencoder reconstruction error (Liu et al., 2024; Wong et al., 2021), value estimates (Hoque et al., 2022; Gokmen et al., 2023), ambiguity (Franzese et al., 2020; Luijkx et al., 2022), divergence (Datta et al., 2023), and diffusion policy training loss (Lee & Kuo, 2024). Outside the scope of IIL, robot-gating based on conformal prediction theory was introduced as well (Ren et al., 2023). In contrast to these robot-gated techniques, human-gated methods have also been proposed, requiring continuous human supervision, where the teacher actively intervenes (Spencer et al., 2020; Kelly et al., 2019; Luo et al., 2024). There are also combinations of robot and human gating (Celemin & Kober, 2023; Hoque et al., 2022). Our approach differs from existing methods by considering the novice's actions during active queries, which we leverage in three ways. First, it enables a sensitivity-aware gating strategy to balance query frequency with error incidence while maintaining the desired sensitivity level. Second, it allows novice actions to be recast as demonstrations by validating the novice's plan or relabeling the goal, inspired by Hindsight Experience Replay (HER) (Andrychowicz et al., 2017). Third, it enables replay prioritization based on novice success, drawing inspiration from Prioritized Experience Replay (PER) (Schaul et al., 2015).

3 Problem Statement

We consider an IIL problem, where a (robot) novice is learning interactively from (human) teacher feedback. The novice and teacher are denoted with subscripts N and T, respectively. The novice learns a policy $\pi_{\mathrm{N}} : \mathcal{O} \times \mathcal{G} \to \mathcal{A}$ that maps observations $\boldsymbol{o}_t^k \in \mathcal{O}$ and goals $g^k \in \mathcal{G}$ to actions $\boldsymbol{a}_t^k \in \mathcal{A}$ during episode k at time step t. The novice learns from a demonstration dataset $\mathcal{D} = \{\boldsymbol{\tau}^k\}_{k=0}^{K}$, consisting of trajectories $\boldsymbol{\tau}$. These are provided by the teacher. Contrasting to existing works, we let the teacher optionally provide a reward r_t^k indicating whether the novice's actions were appropriate considering a goal g^k and the observation $\boldsymbol{o}_t^k$.

Algorithm 1: Action Inquiry DAgger (AIDA)

```
Input: BC dataset $\mathcal{D}_{\text{BC}}$, BC policy $\pi_{\text{N}}^0$, teacher policy $\pi_{\text{T}}$
Parameters: Desired sensitivity $\sigma_{\text{des}}$, random query rate $p_{\text{rand}}$, maximum number of episodes $k_{\max}$
Output: $\pi_{\text{N}}^{k_{\max}}$
$\boldsymbol{u} \leftarrow [\,], \boldsymbol{r} \leftarrow [\,], \boldsymbol{k} \leftarrow [\,], \mathcal{D} \leftarrow \mathcal{D}_{\text{BC}}$
for episode $k = 0 : k_{\max} - 1$ do
    $\boldsymbol{\tau}^k \leftarrow \emptyset$, $\boldsymbol{o}_0^k \leftarrow$ observe(), $g^k \leftarrow$ command(), done $\leftarrow$ False, $t \leftarrow 0$
    while not done do
        $\boldsymbol{a}_t^k \leftarrow \pi_{\text{N}}^k(\boldsymbol{o}_t^k, g^k)$
        $u_t^k \leftarrow$ quantify_uncertainty($\pi_{\text{N}}^k, \boldsymbol{o}_t^k, g^k$)
        $\gamma \leftarrow$ SAG($\boldsymbol{u}, \boldsymbol{r}, \boldsymbol{k}, \sigma_{\text{des}}, p_{\text{rand}}$)    // Set threshold to track sensitivity (Alg. 2)
        $\epsilon \sim U_{[0,1)}$
        if $u_t^k \geq \gamma$ or $\epsilon < p_{\text{rand}}$ then    // Query actively and with probability $p_{\text{rand}}$
            $\boldsymbol{a}_t^k, \boldsymbol{\tau}^k, r_t^k \leftarrow$ FIER($\boldsymbol{o}_t^k, \boldsymbol{a}_t^k, \pi_{\text{T}}, \boldsymbol{\tau}^k, g^k$)    // Collect demonstration (Alg. 3)
            $\boldsymbol{o}_{t+1}^k$, done $\leftarrow$ act($\boldsymbol{a}_t^k$)
        else
            $\boldsymbol{o}_{t+1}^k$, done $\leftarrow$ act($\boldsymbol{a}_t^k$)
        $t \leftarrow t + 1$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \boldsymbol{\tau}^k$
    $P(i), \boldsymbol{w} \leftarrow$ PIER($\boldsymbol{u}, \boldsymbol{r}, \boldsymbol{k}$)    // Prioritize replay (Alg. 4)
    $\pi_{\text{N}}^{k+1} \leftarrow$ update_model($\pi_{\text{N}}^k, \mathcal{D}, P(i), \boldsymbol{w}$)
return $\pi_{\text{N}}^{k_{\max}}$
```

Therefore, a trajectory consists of tuples $\boldsymbol{\tau}^k = \{(\boldsymbol{o}_t^k, \boldsymbol{a}_t^k, g^k, r_t^k)\}_{t=0}^{T_k}$, where reward

$$r_t^k = \begin{cases} 1 & \text{if the teacher validates novice action;} \\ -1 & \text{if the teacher rejects novice action and provides an annotation;} \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

It is worth noting that r_t^k is a teacher reward obtained during queries related to the *novice*'s actions, these actions may be different than $\boldsymbol{a}_t^k$. In our interactive approach, we collect data while executing the novice policy π_{N}^k and iteratively update it with the dataset $\mathcal{D}$, aggregating new demonstrations. Optionally, the policy can be pre-trained with a BC dataset $\mathcal{D}_{\text{BC}}$. During this update, we aim to find the policy π^* within policy space Π that minimizes a loss measure $\mathcal{L}$ between the novice's actions and the teacher's actions, given the distribution of observations in $\mathcal{D}$: $\pi^* = \operatorname{argmin}_{\pi \in \Pi} \mathcal{L}(\pi, \mathcal{D})$. Since we generally do not have full state information, we consider $\boldsymbol{o}_t^k$ to result from an observation mapping $O : \mathcal{S} \to \mathcal{O}$ and we observe the state $\boldsymbol{o}_t^k = O(\boldsymbol{s}_t^k)$. We define the goal state set $\mathcal{S}^g \subset \mathcal{S}$ to be the set of states that satisfy the constraints of g. Therefore, an action $\boldsymbol{a}_t^k$ leads to `success` if $\boldsymbol{s}_{t+1}^k \in \mathcal{S}^g$. The set of states that result in achieving some goal is the union of all possible goal sets, i.e., $\mathcal{S}^G = \bigcup_{g \in \mathcal{G}} \mathcal{S}^g$. We assume the goal g^k is constant throughout an episode, i.e., independent of the dynamics and actions taken. Therefore, a failure described by transition $(\boldsymbol{s}_t^k, \boldsymbol{a}_t^k, \boldsymbol{s}_{t+1}^k, g^k, \texttt{success} = 0)$ can be "relabeled" to success $(\boldsymbol{s}_t^k, \boldsymbol{a}_t^k, \boldsymbol{s}_{t+1}^k, g', \texttt{success} = 1)$ if $\boldsymbol{s}_{t+1}^k \in \mathcal{S}^G$, i.e., if the action resulted in achieving some other goal $g' \in \mathcal{G}$. So, if the teacher can observe s_t^k and (predict) s_{t+1}^k, and can infer whether $s_{t+1}^k \in \mathcal{S}^G$, the teacher can relabel failure transitions to successes. Furthermore, we consider an active approach based on the policy's prediction uncertainty. Therefore, we require an uncertainty operator $U : \Pi \times \mathcal{O} \times \mathcal{G} \to \mathbb{R}_{[0,1]}$ that provides the prediction uncertainty of the novice policy, given the current observation and goal, i.e., $u_t^k = U(\pi_{\text{N}}, \boldsymbol{o}_t^k, g^k)$. Optionally, one can also take $\mathcal{D}$ into account when quantifying uncertainty, e.g., to quantify the proximity of an observation to those in $\mathcal{D}$.

4 Action Inquiry DAgger (AIDA) Framework

AIDA is an interactive imitation learning framework where the teacher is actively queried based on Sensitivity-Aware Gating (SAG). The teacher can provide feedback through Foresight Interactive Experience Replay (FIER) in three modalities: validation, relabeling, or annotation demonstrations. We update the novice policy using the demonstration dataset while we perform Prioritized Interactive Experience Replay (PIER). The main training procedure of AIDA, summarized in Alg. 1, follows these steps: In each episode k, at every time step t, the novice policy π_{N}^{k} selects an action $\boldsymbol{a}_t^k$ based on observation $\boldsymbol{o}_t^k$ and goal g^k. Besides inferring its policy, the novice also quantifies the corresponding uncertainty u_t^k (Alg. 1, lines 2-6). SAG then sets a gating threshold γ to track the user-defined sensitivity level σ_{des} (line 7). The teacher is queried if the novice uncertainty exceeds the gating threshold. Additional queries are obtained with probability p_{rand} to enhance the performance of the SAG algorithm (detailed in Sec. 4.1). During these queries the novice presents its planned action $\boldsymbol{a}_t^k$, allowing the teacher to validate, relabel, and/or provide an annotation demonstration (lines 9-11). If the teacher is not queried, the novice acts autonomously (lines 12-13). When the episode is `done`, e.g., because the goal constraints are satisfied or a time-out is reached, the demonstration trajectory $\boldsymbol{\tau}^k$ is added to $\mathcal{D}$. Finally, a model update can be performed using PIER (lines 16-17). Note that we keep track of the update/episode counts $\boldsymbol{k} = [0, \dots, K_{T^K}]$, uncertainties $\boldsymbol{u} = [u_0^0, \dots, u_{T^K}^K]$, and rewards $\boldsymbol{r} = [r_0^0, \dots, r_{T^K}^K]$ during training with AIDA. The next sections will provide detailed descriptions of the subroutines from Alg. 1, i.e., SAG for gating (Alg. 2), FIER for demonstration collection (Alg. 3) and PIER for replay prioritization (Alg. 4).

4.1 Sensitivity-Aware Gating (SAG)

At each time step, the uncertainty of the novice policy determines whether it should act autonomously or request teacher feedback. We introduce SAG for this gating problem. This is an intuitive method for balancing query frequency and system failures. SAG dynamically adjusts the gating threshold γ to maintain a user-prescribed sensitivity level. In this context, queries are considered positives, and autonomous actions are considered negatives. A false positive occurs when the teacher is queried despite the novice's action being valid since we want the novice to be as autonomous as possible. Conversely, a false negative occurs when the teacher is not queried despite an invalid novice action since it leads to system failure. We formalize the gating problem as a semi-supervised logistic regression, using uncertainty u as the independent variable and reward r as the indicator variable. The logistic regression assumption—that the log-likelihood ratio of class distributions is linear in the observations—holds for various exponential distributions, such as normal, beta, and gamma distributions (Amini & Gallinari, 2002).

We summarize SAG in Alg. 2 and provide a visualization for more intuitive understanding in Fig. 2. For computing the gating threshold γ, we maintain a window of the most recent values in $\boldsymbol{u}_W, \boldsymbol{r}_W$, and $\boldsymbol{k}_W$ We do this because, with each model update, the uncertainty and reward information becomes more outdated (line 1 of Alg. 2). The window size is adjusted adaptively to ensure that the window contains at least N_{min} novice failures. These known failures (true positives) are essential for approximating sensitivity (true positive rate) accurately. Since the failure distribution over uncertainty shifts over time due to model updates, we normalize $\boldsymbol{u}_W$ (lines 2-4 of Alg. 2) to match the expected uncertainty at the current episode K. This is achieved by performing linear regression on $\boldsymbol{k}_W$ and $\boldsymbol{u}_W$, then adjusting for the expected difference in uncertainty between episode k and K. This is visualized in Fig. 2 A. Next, we fit a logistic function to $-\boldsymbol{r}_W$ and $\boldsymbol{u}_W$ (line 5 of Alg. 2 and Fig. 2 B-C). We negate the rewards because, in the context of sensitivity, positive cases typically represent costly events—in this case, novice failures. We capture these failures by negating $\boldsymbol{r}_W$. If the teacher was not queried, we do not know whether the novice acted successfully at (k, t). Since the uncertainty u_t^k is known, we can obtain a pseudo-label by sampling from the fitted logistic model at u_t^k (line 11 and Fig. 2 D). This enables us to approximate true and false positive rates for different threshold values.

Queries are not only made when uncertainty exceeds the threshold but also with probability p_{rand}. These random queries ensure that labels are collected across the entire uncertainty range, not just in the high-uncertainty regime. If we do not account for these random queries when setting the threshold, the resulting threshold will be too conservative, leading to a sensitivity higher than σ_{des}. Since p_{rand} is user-defined, we can incorporate these random queries when determining the gating threshold. The total sensitivity σ can be computed by considering both the true positives TP_γ and false negatives FN_γ resulting from active gating, along with the true positives $\mathrm{TP}_{\mathrm{rand}}$ from random gating:

$$\sigma = \frac{\mathrm{TP}_\gamma + \mathrm{TP}_{\mathrm{rand}}}{\mathrm{TP}_\gamma + \mathrm{FN}_\gamma}. \tag{2}$$

Algorithm 2: Sensitivity-Aware Gating (SAG)

```
    Input: Uncertainties $\boldsymbol{u}$, rewards $\boldsymbol{r}$, update counts $\boldsymbol{k}$
    Parameters: Desired sensitivity $\sigma_{\text{des}}$, random query rate $p_{\text{rand}}$, minimum number of negative labels $N_{\text{min}}$, number of
                imputation repetitions $N_{\text{rep}}$
    Output: Gating threshold $\gamma$
 1  $\boldsymbol{u}_W, \boldsymbol{r}_W, \boldsymbol{k}_W \leftarrow$ get_window($\boldsymbol{u}, \boldsymbol{r}, \boldsymbol{k}, N_{\text{min}}$)
 2  $w_{\text{lin}}, b_{\text{lin}} \leftarrow$ LinRegres().fit($\boldsymbol{k}_W, \boldsymbol{u}_W$)             // Linear regression for normalizing $u$
 3  for $u_t^k \in \boldsymbol{u}_W$ do
 4      $u_t^k \leftarrow u_t^k + w_{\text{lin}}(K - k)$                                        // Visualized in Fig. 2 A
 5  $w_{\text{log}}, b_{\text{log}} \leftarrow$ LogRegres().fit($\boldsymbol{u}_W, -\boldsymbol{r}_W$)   // Logistic regression for imputation where $r_t^k = 0$
 6  $\boldsymbol{\gamma} \leftarrow [\,]$
 7  for $i = 0 : N_{\text{rep}} - 1$ do
 8      $\boldsymbol{f}_W \leftarrow -\boldsymbol{r}_W$                                              // Visualized in Fig. 2 B
 9      for $r_t^k \in \boldsymbol{r}_W$ do
10          if $r_t^k == 0$ then                                              // If teacher was not queried
11              $f_t^k \sim \{-1, 1\}, \quad P(f_t^k = 1) =$ sigmoid($w_{\text{log}} u_t^k + b_{\text{log}}$)    // Visualized in Fig. 2 C
12      $\boldsymbol{\sigma}_\gamma, \boldsymbol{\theta} \leftarrow$ get_roc($\boldsymbol{u}_W, \boldsymbol{f}_W$)     // True positive rates and corresponding thresholds
13      $\gamma_i \leftarrow$ interpolate($\sigma_{\text{des}}, \boldsymbol{\sigma}_\gamma + p_{\text{rand}}(1 - \boldsymbol{\sigma}_\gamma), \boldsymbol{\theta}$)   // Visualized in Fig. 2 D
14  $\gamma \leftarrow$ median($\boldsymbol{\gamma}$)
15  return $\gamma$
```

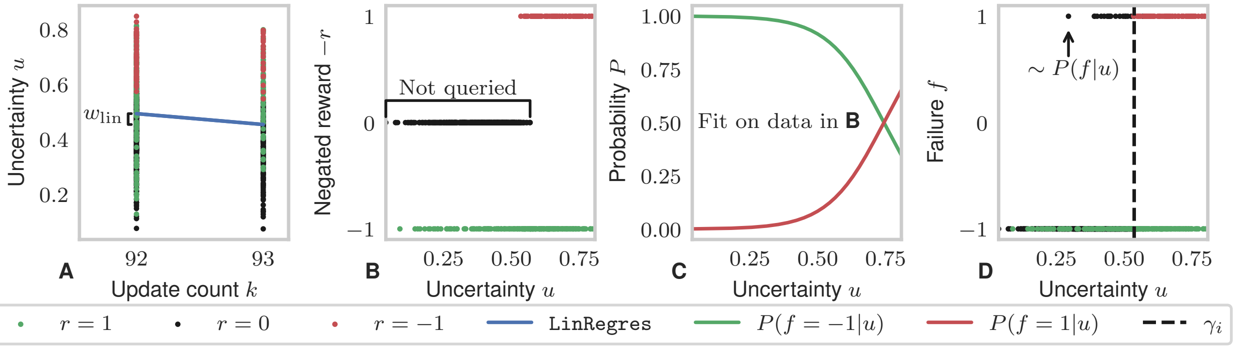


Figure 2: Visualization of the SAG algorithm with experimental data from Sec. 5.1. First, we normalize the uncertainty values in $\boldsymbol{u}_W$ using linear regression, as the uncertainty distribution shifts with the number of updates **(A)**. By negating the rewards $\boldsymbol{r}_W$, we obtain labels for novice failures, denoted as $\boldsymbol{f}_W$ **(B)**. Labels are unavailable when the teacher was not queried ($r = 0$). However, since uncertainties at those time steps are known, we generate pseudo labels by sampling from a logistic distribution fit to $\boldsymbol{u}_W$ and $\boldsymbol{f}_W$ **(C)**. We then compute a gating threshold γ_i using both the labels and pseudo labels **(D)**. This sampling and threshold calculation process is repeated N_{rep} times.

The number of TP_{rand} are determined by which of FN_γ are queried. These queries are based on whether occur with probability p_{rand} (see lines 8-9 of Alg. 1). Therefore, its expected value is $\text{FN}_\gamma \cdot p_{\text{rand}}$. This leads to:

$$\begin{aligned}\mathbb{E}_{\epsilon \sim U_{[0,1)}}[\sigma] &= \frac{\text{TP}_\gamma}{\text{TP}_\gamma + \text{FN}_\gamma} + \frac{\text{FN}_\gamma \cdot p_{\text{rand}}}{\text{TP}_\gamma + \text{FN}_\gamma} && (3)\\ &= \sigma_\gamma + p_{\text{rand}}(1 - \sigma_\gamma), && (4)\end{aligned}$$

where σ_γ is the sensitivity for gating disregarding the random queries. Thus, we interpolate between threshold values that best satisfy the desired sensitivity level $\sigma_{\text{des}} = \sigma_\gamma + p_{\text{rand}}(1 - \sigma_\gamma)$ (line 13 of Alg. 2 and Fig. 2 D). The process of sampling pseudo labels is repeated N_{rep} times, and the final threshold γ is set as the median of the thresholds obtained across repetitions.

4.2 Foresight Interactive Experience Replay (FIER)

When the novice's uncertainty exceeds the threshold set by SAG, feedback is requested from the teacher through FIER. This method enhances demonstration collection by considering the novice's planned actions during queries and presenting them to the human teacher. The FIER procedure is summarized in Alg. 3. FIER reduces the number of required teacher annotations by enabling the human teacher to provide two additional feedback modalities. First, we allow the teacher to validate the proposed actions. If the teacher considers the plan valid ($r == 1$ in line 2 in Alg. 3), we execute and add the novice's planned actions to the demonstration dataset. The novice plan can be valuable even if it is invalid. Inspired by HER Andrychowicz et al. (2017), we can utilize invalid novice plans as demonstrations if they achieve another goal (line 7 in Alg. 3). In this case, we can obtain a new demonstration by letting the teacher relabel the goal. The benefit of relabeling demonstrations is twofold. First, there can be situations where relabeling the goal is less demanding for the teacher than providing an annotation. In that case, it is possible to collect additional demonstrations at a reduced cost Secondly, it allows for the collection of demonstrations for goals induced by the novice policy instead of collecting demonstrations only for goals induced by the distribution of commands. This way, the novice can learn to perform tasks beyond the instructed commands and possibly generalize better to novel scenarios. After providing the option to relabel the novice's actions, the teacher is asked to provide an annotation demonstration (line 6). To summarize, we can collect three types of demonstrations with FIER: validation, relabeling, and annotation demonstrations.

4.3 Prioritized Interactive Experience Replay (PIER)

At the end of the episode, the demonstrations collected through FIER are aggregated into the demonstration dataset, allowing the novice policy to be updated. Efficiently performing policy updates is particularly important in interactive imitation learning, as this paradigm involves a human teacher providing online feedback. Taking inspiration from PER (Schaul et al., 2015), we introduce an interactive equivalent, which we call PIER (summarized in Alg. 4). PIER prioritizes the replay of the demonstration dataset based on uncertainty, novice success, and demonstration age. We prioritize demonstrations where the novice fails over successes. Those with low uncertainty receive the highest priority among failures, as they suggest confident yet mistaken actions. While among successes, those with high uncertainty are prioritized to reduce the novice's uncertainty for those situations. Successes with low uncertainty, indicating proficient performance, are given the lowest priority. Since the uncertainty and novice success information become outdated with each model update, we diminish the prioritization based on the age of the demonstration. Alg. 4 shows how these desired properties are integrated into our prioritized replay scheme. Similar to PER (Schaul et al., 2015), we define the probability of sampling demonstration tuple of episode k at timestep t to be:

$$P(k,t) = \frac{(p_t^k)^\alpha}{\sum_i \sum_j (p_j^i)^\alpha}. \tag{5}$$

Here p_t^k is the priority of the demonstration tuple from episode k at time step t and $\alpha \geq 0$. Increasing α results in more prioritization, while $\alpha = 0$ corresponds to uniform sampling. We define the prioritization exponent as a linear combination of the uncertainty and the number of model updates since the demonstration was added to the dataset, i.e., $c_t^k = \lambda u_t^k + (1-\lambda)(\frac{K-k}{K})$. Here, $0 \leq \lambda \leq 1$ scales prioritization based on the uncertainty versus novelty of the sample. Finally, the priorities are:

$$p_t^k = 1 - r_t^k \frac{b^{1-c_t^k} - 1}{b - 1}, \tag{6}$$

where $b > 1$ is the base. In this way, the priorities of old demonstrations with high novice uncertainty depend little on novice success. In contrast, the priorities for recent demonstrations with low novice uncertainty depend greatly on novice success. Having defined the priorities, we need to compensate for the bias that prioritization introduces when minimizing the expectation of a loss function l, i.e., $\mathbb{E}_{k,t\sim P(k,t)} \mathcal{L}\left(\pi_{\mathrm{N}}, (\boldsymbol{o}_t^k, \boldsymbol{a}_t^k)\right)$, instead of sampling k, t according to the distribution of the dataset—i.e., $k, t \sim D(k,t)$. Again, following PER (Schaul et al., 2015), we can mitigate the bias by introducing importance-sampling weights $\boldsymbol{w} = [w_0^0, \dots]$ (see line 7 of Alg. 4). Here, β determines the level of bias compensation. The weights are normalized to prevent numerical instabilities.

Algorithm 3: Foresight Interactive Experience Replay (FIER)

```
Input: Observation $\boldsymbol{o}$, novice action $\boldsymbol{a}$, teacher
       policy $\pi_{\mathrm{T}}$, trajectory $\boldsymbol{\tau}$, goal $g^k$
Parameters: Goal set $\mathcal{G}$
Output: Action $\boldsymbol{a}$, trajectory $\boldsymbol{\tau}$
$r, g' \leftarrow$ query($\boldsymbol{o}, \boldsymbol{a}$)
if $r == 1$ then                                   // Validation tuple
    $\boldsymbol{\tau} \leftarrow \boldsymbol{\tau} \cup (\boldsymbol{o}, \boldsymbol{a}, g = g^k, r = 1)$
else
    $\boldsymbol{a} \leftarrow \pi_{\mathrm{T}}(\boldsymbol{o})$                   // Annotation tuple
    $\boldsymbol{\tau} \leftarrow \boldsymbol{\tau} \cup (\boldsymbol{o}, \boldsymbol{a}, g = g^k, r = -1)$
    if $g' \in \mathcal{G}$ then                   // Relabeled tuple
        $\boldsymbol{\tau} \leftarrow \boldsymbol{\tau} \cup (\boldsymbol{o}, \boldsymbol{a}, g = g', r = 0)$
return $\boldsymbol{a}$, $\boldsymbol{\tau}, r$
```

Algorithm 4: Prioritized Interactive Experience Replay (PIER)

```
Input: Uncertainties $\boldsymbol{u}$, rewards $\boldsymbol{r}$, update counts $\boldsymbol{k}$
Parameters: Scale $\lambda$, base $b$, exponents $\alpha, \beta$
Output: Sampling priorities $\boldsymbol{p}$, weights $\boldsymbol{w}$
$\boldsymbol{w} = [\,]$
for $k = 0 : K$ do
    for $t = 0 : T^k$ do
        $c_t^k = \lambda u_t^k + (1 - \lambda) \frac{K-k}{K}$
        $p_t^k \leftarrow 1 - r_t^k \frac{b^{1-c_t^k} - 1}{b-1}$
        $P(k,t) \leftarrow \frac{(p_t^k)^\alpha}{\sum_i \sum_j (p_j^i)^\alpha}$
        $w_t^k = (|\boldsymbol{k}| \cdot P(k,t))^{-\beta} \,/\, \max_{i,j} w_j^i$
return $P(k,t), \boldsymbol{w}$
```

5 Experimental Evaluation

To support claims **C1-4** from Sec. 1, we evaluate AIDA and its components in four sets of experiments. First, we performed active dataset aggregation on the MNIST dataset (LeCun et al., 1998) using TorchUncertainty (Lafage & Laurent, 2024) to validate SAG extensively. Second, we interactively trained CLIPort agents on simulated language-conditioned tabletop manipulation tasks (Shridhar et al., 2021). Third, we conducted experiments on a real-world assembly setup to demonstrate that these claims extend beyond simulation. Finally, we showcase AIDA's applicability by integrating it with built-in primitive actions on a Spot robot to perform a sorting task.

5.1 MNIST Dataset Aggregation

To support claim **C1**—*SAG balances query count and system failures by tracking a desired sensitivity*—we conducted experiments in which we interactively trained digit classification models on the MNIST dataset (LeCun et al., 1998) [1]. We selected this setup due to its low computational requirements, enabling extensive ablations and easy reproducibility. Additionally, existing packages such as TorchUncertainty (Lafage & Laurent, 2024) facilitate uncertainty quantification in this setting. Since we focus on the SAG component, we follow the procedure described in Alg. 1, but without demonstration collection via relabeling or replay prioritization. To validate whether SAG can track a desired sensitivity, we performed interactive training for nine different sensitivity values, i.e., $\sigma_{\text{des}} \in \{0.1i\}_{i=1}^{9}$, repeating the procedure ten times for each value of σ_{des}.

These experiments proceed as follows. We sample a batch of 128 handwritten digit images from the MNIST dataset at each timestep without replacement. This allows for 468 timesteps, as the dataset contains 60,000 samples. Although MNIST provides ground truth labels, we simulate an active learning scenario where labels are queried if the prediction uncertainty exceeds the threshold γ set by SAG or randomly with probability $p_{\text{rand}} = 0.1$. Uncertainty quantification is performed using Monte Carlo Dropout (MC Dropout) (Gal & Ghahramani, 2016) with a dropout rate of 0.4 and 16 stochastic forward passes, forming an ensemble $\mathcal{C} = \{h_1, \ldots, h_{16}\}$. For a sample x with label y, prediction uncertainty is computed as $u = 1 - \max_y P_{\mathcal{C}}(y|x)$, where $P_{\mathcal{C}}(y|x) = \frac{1}{16} \sum_{i=1}^{16} P_i(y|x)$. Ground truth labels are obtained for the queried samples and added to the training dataset. The model is updated every five timesteps using the aggregated dataset.

The results of these experiments are summarized in Fig. 3. The sensitivity plots in Fig. 3 A show that SAG successfully tracks the desired sensitivity level for all nine values of σ_{des}. Fig. 3 B reveals a clear trade-off between sensitivity and specificity: higher values of σ_{des} lead to lower specificity. Additionally, Fig. 3 B and Fig. 3 C show that, for most values of σ_{des}, specificity increases over time, while the query rate decreases over time. This shows that as the uncertainty quantification's informedness (Youden's J statistic) improves,

---

[1] The code and data from these experiments are available at `https://github.com/aida-paper/aida_mnist`.

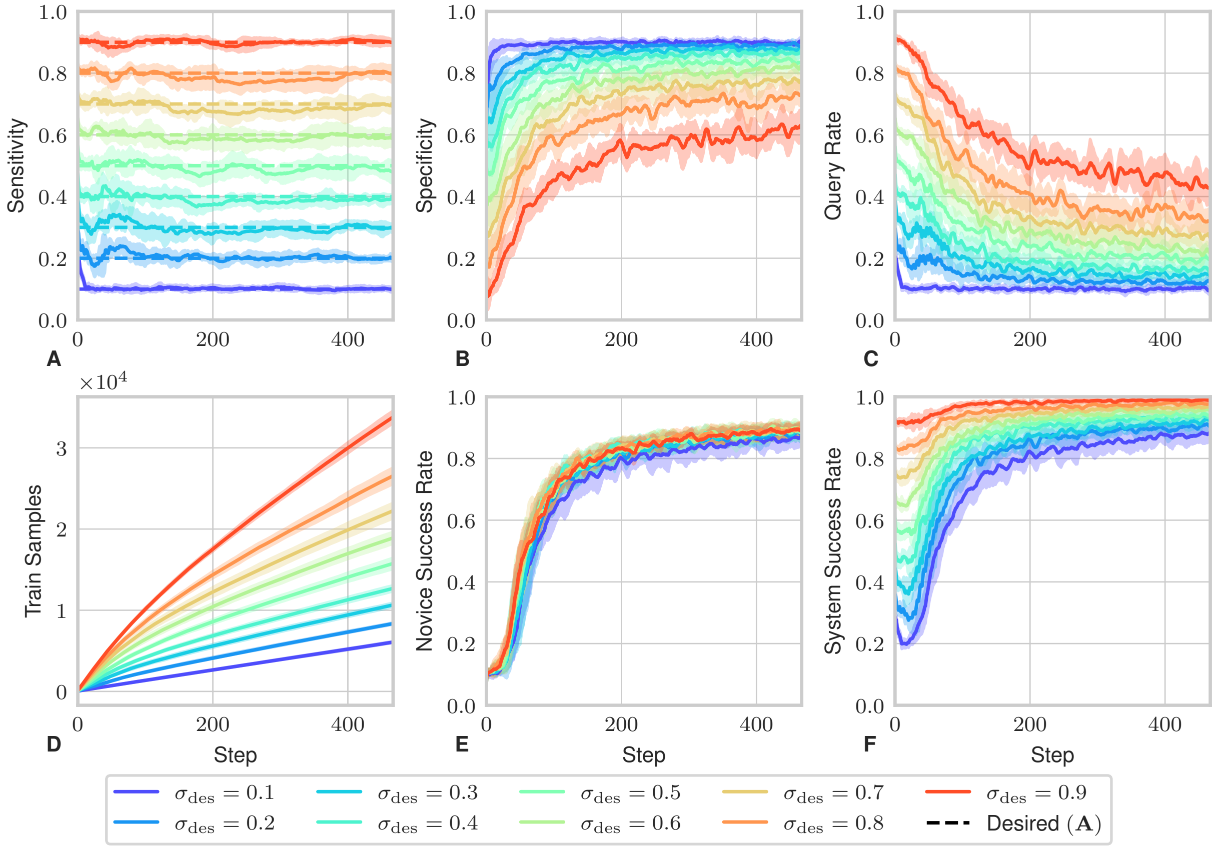


Figure 3: Results for various levels of desired sensitivity, σ_{des}, in active dataset aggregation with SAG on the MNIST (LeCun et al., 1998) dataset. Sensitivity in **A** and specificity in **B** are calculated over a moving window of 1000 failures and successes, respectively. Novice and system success rates in E and F are calculated over a window of 1000 samples (approximately eight steps). Mean and standard deviation are shown for ten repetitions.

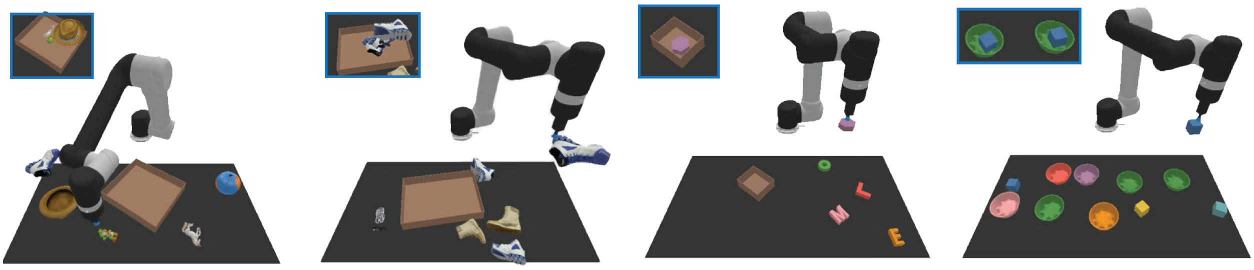
SAG adapts by reducing the query rate to maintain sensitivity tracking. Fig. 3 D and Fig. 3 E show that while fewer training samples are collected for lower values of σ_{des}, the novice still converges to similar success rates over time. However, the convergence rate is slightly slower for $\sigma_{\text{des}} = 0.1$. Finally, Fig. 3 F shows that higher values of σ_{des} lead to a better system success rate, as more failures are prevented through querying.

5.2 CLIPort Benchmark Tasks

To support claims **C1-3**, we conducted experiments using AIDA to train CLIPort (Shridhar et al., 2021) agents interactively [2]. CLIPort is a language-conditioned imitation-learning agent that leverages the CLIP (Radford et al., 2021) foundation model and sample-efficient Transporter Networks (Zeng et al., 2021) for vision-based manipulation. We selected this setup because it allows novices to communicate their actions by indicating planned pick-and-place locations on an image alongside a language command, making it well-suited for AIDA. We compared AIDA's performance against an active DAgger baseline without both PIER and FIER to provide evidence for claims **C2-3**. Examples of such active DAgger approaches that request feedback when the policy is considered unsafe, risky, or uncertain are Zhang & Cho (2017); Menda et al. (2019; 2017); Hoque et al. (2022); Datta et al. (2023); Lee & Kuo (2024); Luijkx et al. (2022). We also performed ablations with AIDA without PIER and AIDA without FIER to isolate the effects of the individual components. All methods use SAG for gating and rely on prediction entropy to quantify uncertainty for a fair comparison. The comparison was conducted across the subset of tasks by Shridhar et al. (2021). These tasks

---

[2] The code, data, videos, and a notebook are available at `https://github.com/aida-paper/aida_cliport`.



“pack the yoshi figure in the brown box”

“pack all the blue and black sneaker objects in the brown box”

“pack the hexagon in the brown box”

“put the blue blocks in a green bowl”

Figure 4: The CLIPort benchmark tasks from left to right: packing-google-objects-seq, packing-google-objects-group, packing-shapes and put-blocks-in-bowls. Adapted from image by Shridhar et al. (2021).

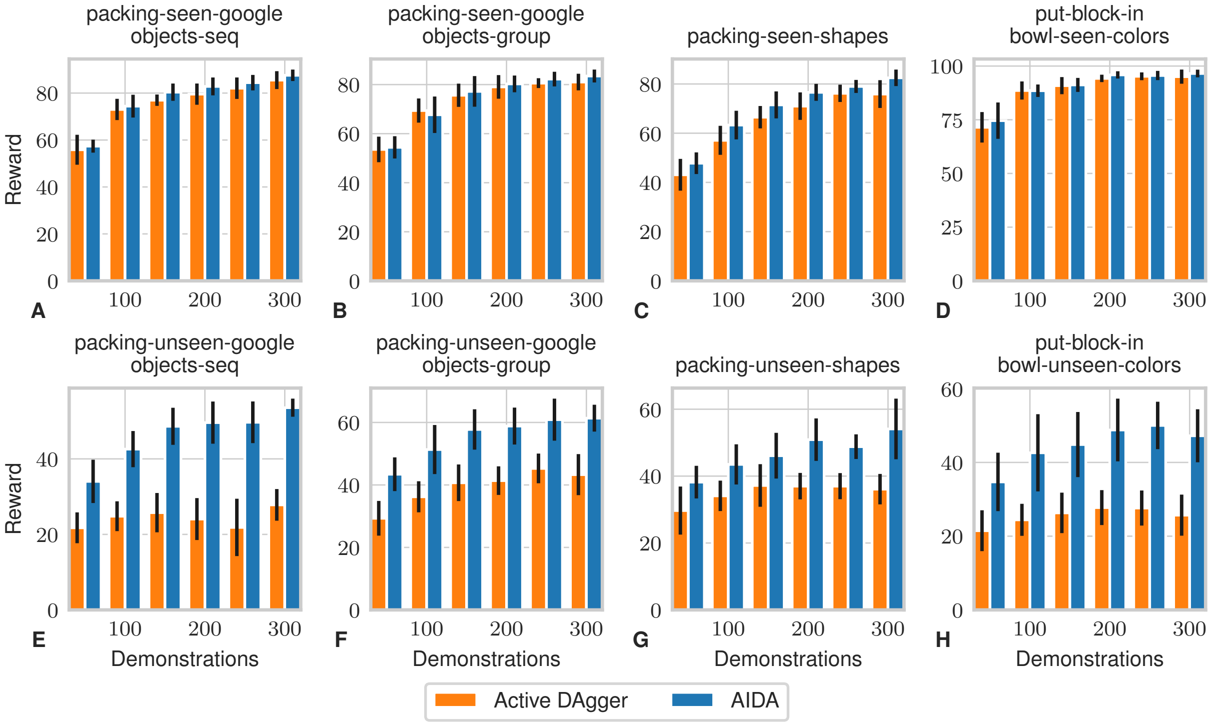


Figure 5: Cumulative rewards for evaluating checkpoints over 100 episodes on tasks with seen and unseen objects. Mean and standard deviation are shown for ten policies using a moving window of 50 episodes. The results show clear improvements with AIDA on the unseen scenarios.

are visualized in Fig. 4. To highlight scenarios where relabeling demonstrations is beneficial, we modified the tasks involving Google objects and shapes so that objects from the unseen set appear as distractors during training. The following hyperparameters were used for both AIDA and the active DAgger baseline if applicable. For SAG we used $\sigma_{\text{des}} = 0.9, N_{\text{min}} = 15, p_{\text{rand}} = 0.2$ and for PIER $\alpha = 1.5, b = 10, \beta = 1$ and $\lambda = 0.5$. Each setting involved training ten CLIPort agents without BC pretraining, collecting 300 interactive demonstrations, and evaluating checkpoints every 50 demonstrations. Model updates occur at the end of an episode if a demonstration is collected. This way, we ensure that AIDA and the active DAgger baseline had the same number of updates.

The cumulative rewards for evaluating checkpoints on tasks with seen and unseen objects are shown in Fig. 5. While AIDA performs equally or better on all tasks, the number of teacher annotations is significantly lower, as shown in Fig. 6. The performance gain of AIDA can be attributed to the composition of the demonstration dataset. For the active DAgger baseline, all demonstrations consist of annotation tuples, whereas AIDA collects many through validation and relabeling. The relabeled demonstrations explain AIDA's superior performance on unseen tasks: agents sometimes obtained demonstrations by relabeling novice failures, where the intended pick was a distractor object from the unseen set. Fig. 6 also shows that AIDA requires fewer annotation demonstrations than the active DAgger baseline. This is further reflected in the training curves in Fig. 7, where the novice achieves higher success rates for the same number of annotation tuples. The corresponding sensitivity curves for AIDA in Fig. 8 confirm that SAG maintains the desired sensitivity level across all tasks.

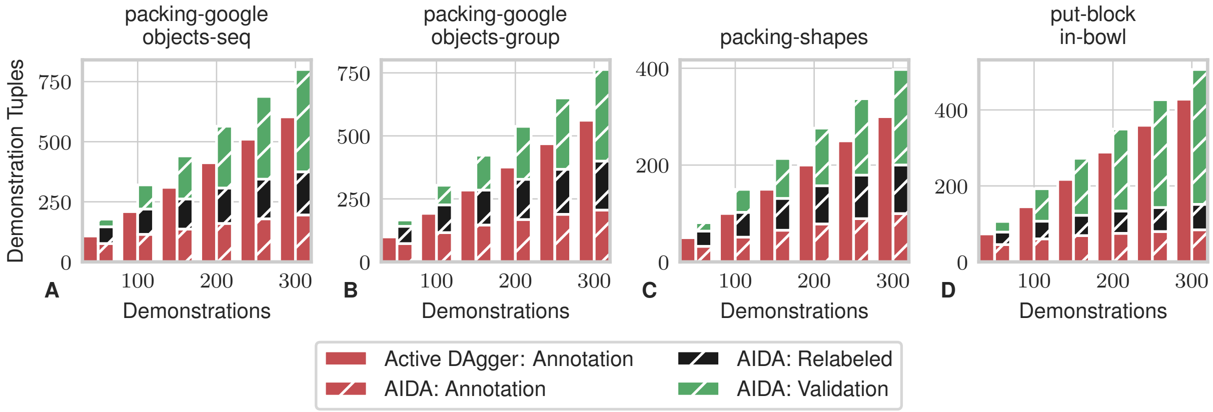


Figure 6: Composition of the demonstration datasets, showing mean values for ten policies. AIDA relies on fewer annotation demonstrations and benefits from relabeling and validation demonstrations.

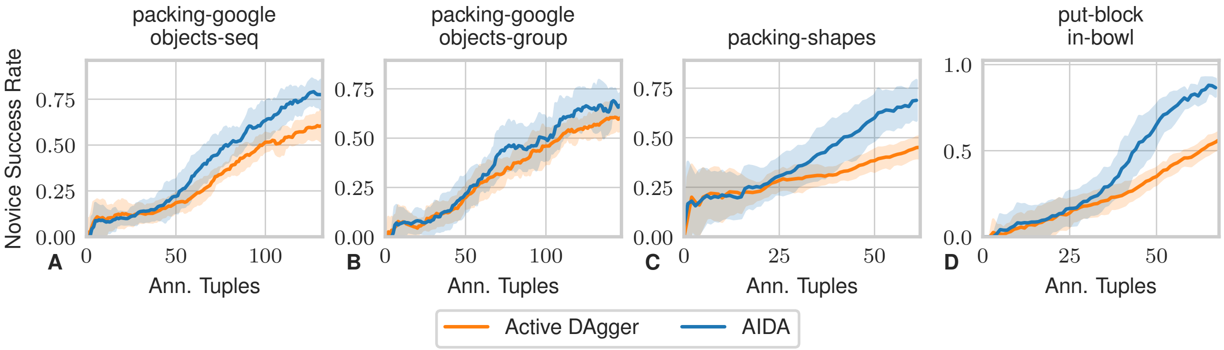


Figure 7: Novice success rate during training as a function of the number of collected annotation tuples. Mean and standard deviation are shown for ten policies, with a moving window of 50 episodes.

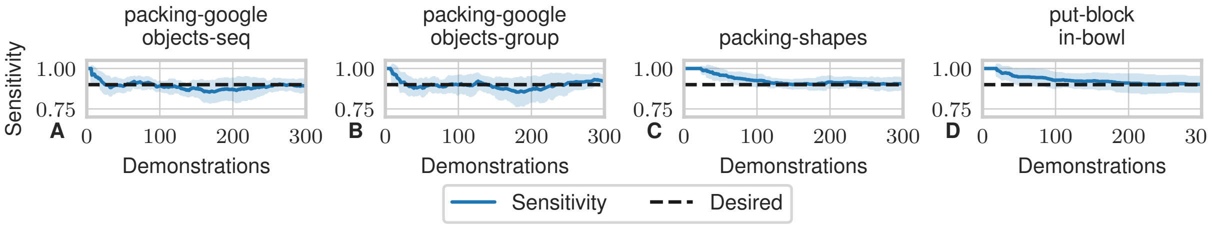


Figure 8: Sensitivity during training for AIDA on the CLIPort tasks. Sensitivity is calculated over a moving window of 50 failures. Mean and standard deviation are shown for ten policies.

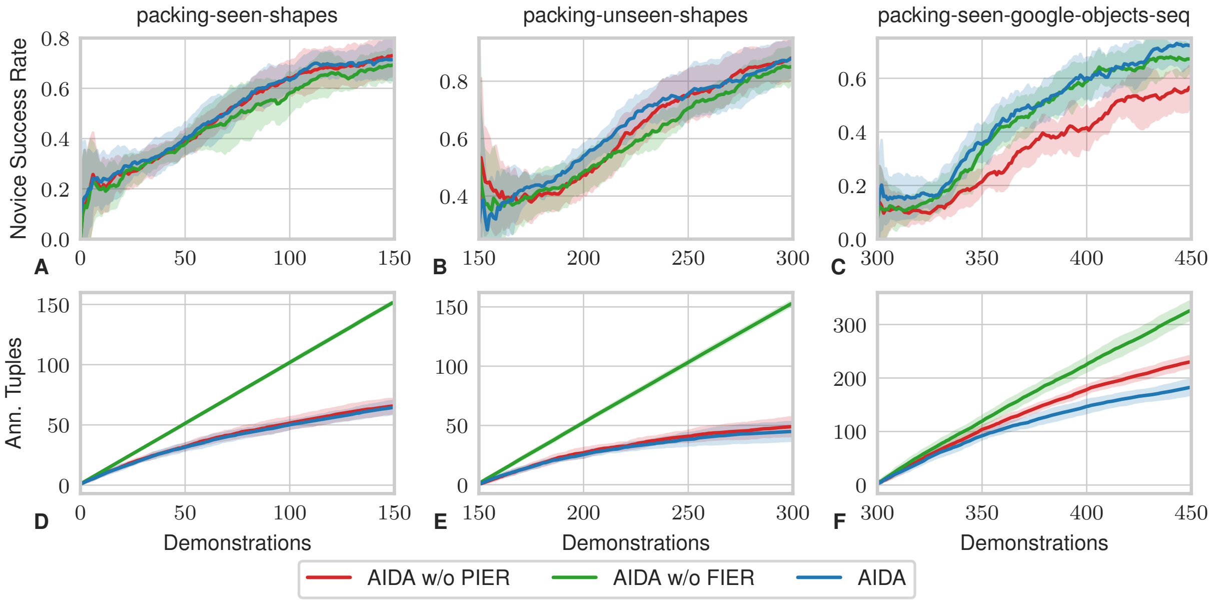


Figure 9: Novice success rate during training (**A-C**) and the number of collected annotation tuples (**D-F**) under domain shifts. The success rate is computed over a moving window of 50 episodes, reinitialized after each domain shift. The first 150 demonstrations are collected on packing-seen-shapes, the next 150 on packing-unseen-shapes, and the final 150 on packing-seen-google-objects-seq. Mean and standard deviation are shown for ten policies.

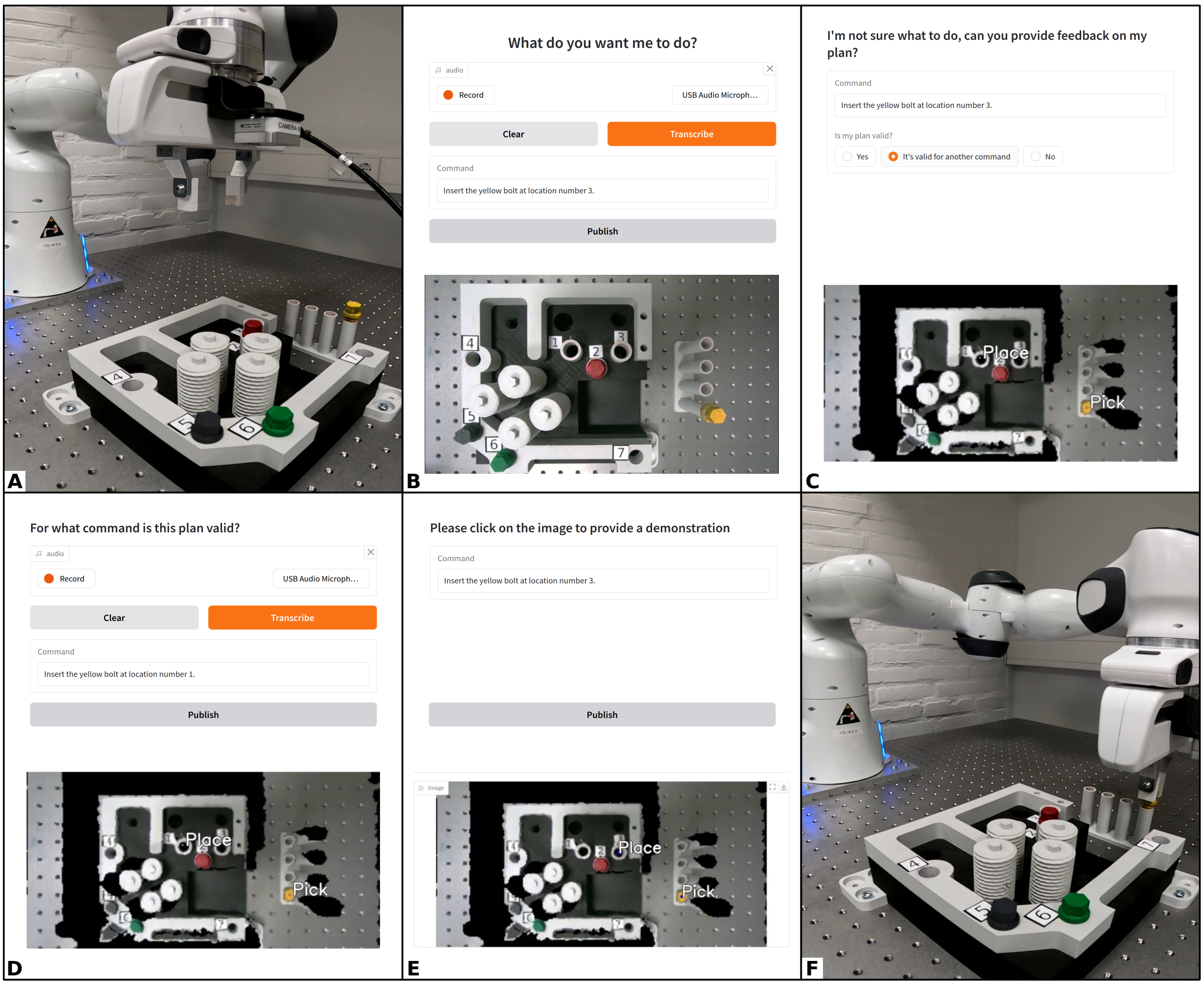
To support **C4**, we performed ablations with AIDA under domain shifts. We trained agents using AIDA, AIDA without PIER (w/o PIER), and AIDA without FIER (w/o FIER) on a sequence of tasks with increasing domain shifts: packing-seen-shapes, packing-unseen-shapes, and packing-seen-google-objects-seq. As shown in Fig. 9, AIDA and AIDA w/o PIER perform similarly on the initial task, while AIDA w/o FIER performs slightly worse, likely due to the benefits of relabeling demonstrations. Moreover, AIDA w/o FIER requires more annotation demonstrations (Fig. 9 D). After transitioning to packing-unseen-shapes, AIDA adapts slightly faster to the domain shift initially, though performance is similar after 300 demonstrations across all settings. This improved adaptation likely stems from FIER's relabeling and PIER's replay prioritization. Upon shifting to packing-seen-google-objects-seq, AIDA and AIDA w/o FIER outperform AIDA w/o PIER, highlighting the benefits of replay prioritization with PIER. This effect is more pronounced after the second transition, as the shift from unseen-shapes to seen-google-objects-seq is larger than from seen to unseen shapes. Additionally, as the demonstration dataset grows, the probability of sampling a specific demonstration decreases for uniform sampling, further increasing PIER's effect. Finally, AIDA's improved performance on the third task requires fewer annotation demonstrations, as more validation demonstrations are collected (Fig. 9 F).

5.3 Real-World Engine Assembly

We conducted experiments on a real-world assembly task to demonstrate that our claims extend beyond simulation and showcase AIDA's applicability in real-world settings [3]. This task is a simplified version of a diesel engine assembly using 3D-printed models. The procedure is illustrated in Fig. 10. As shown in Fig. 10 A, the setup includes a Franka Panda robot equipped with an in-hand RealSense D405 RGB-D camera and a Franka hand with custom-printed fingers for grasping bolts. The control scheme is implemented using the EAGERx framework (van der Heijden et al., 2024). The objective is to pick bolts from a holder and insert them into specific locations on the engine block. We use pick-and-place primitives that rely on 2D Cartesian positions, assuming a given height for picking and placing.

---

[3] A video of this experimental evaluation is available at `https://aida-paper.github.io`.



What do you want me to do?

Command: Insert the yellow bolt at location number 3.

I'm not sure what to do, can you provide feedback on my plan?

Command: Insert the yellow bolt at location number 3.

Is my plan valid?
○ Yes ● It's valid for another command ○ No

For what command is this plan valid?

Command: Insert the yellow bolt at location number 1.

Please click on the image to provide a demonstration

Command: Insert the yellow bolt at location number 3.

Figure 10: Real-world implementation of AIDA on an engine assembly task. **(A)** The setup includes a Franka Panda robot, an RGB-D camera, and 3D-printed parts. **(B)** The interface allows the operator to issue commands. **(C)** When queried, the operator can validate, relabel, or reject the plan. **(D)** The operator relabels the plan. **(E)** The operator provides an annotation demonstration. **(F)** The robot executes the demonstration.

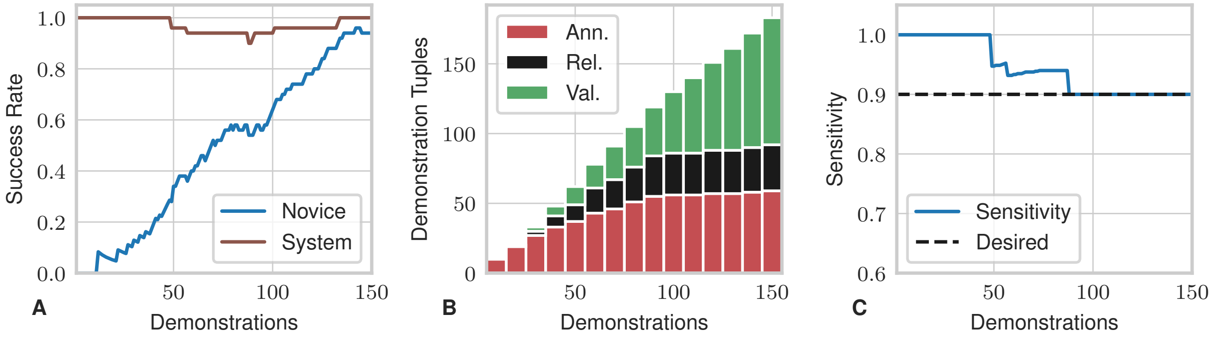


Figure 11: Results for the engine assembly task. **(A)** Success rates calculated over a window of 50 episodes. **(B)** Composition of the demonstration dataset. **(C)** Sensitivity calculated over a moving window of 50 failures.

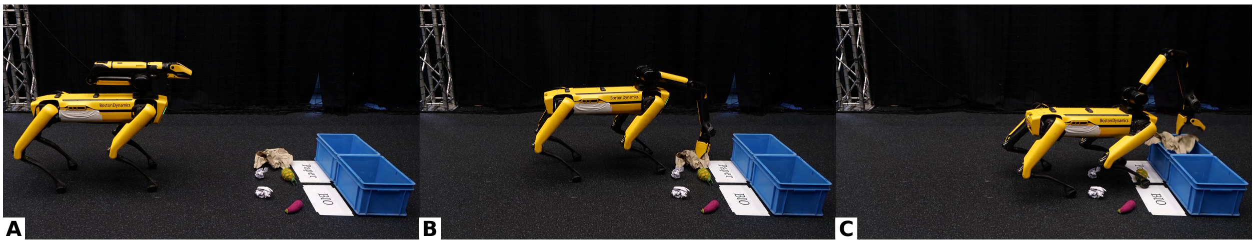


Figure 12: Real-world implementation of AIDA on a sorting task with Spot **(A)**. This demonstrates AIDA's applicability in scenarios where a robot has access to one or more skills, such as grasping **(B)**, walking, and placing **(C)**.

The task involves four bolt colors (red, yellow, green, and black) and seven insertion locations. The bolts are randomly ordered and placed in a holder. The human operator interacts with the robot via the interface shown in Fig. 10 B. This Gradio (Abid et al., 2019) interface allows command input via speech or text. In our experiments, we generate random commands in the form: "*Insert the* [color] *bolt at location number* [location number]." Upon receiving a command, a top-down RGB-D reconstruction is obtained following Zeng et al. (2021), and the novice policy is evaluated. We use a CLIPort agent with the same hyperparameters as in Sec. 5.2, but with a smaller batch size of 3 due to GPU memory constraints on our laptop. If the prediction uncertainty exceeds the threshold γ set by SAG, or randomly with probability $p_{\text{rand}} = 0.2$, the robot queries the human operator. This is shown in Fig. 10 C. The robot's planned action is visualized in the interface, allowing the operator to validate, relabel, or reject the plan. If validated, the robot executes the plan and it is added to the dataset. If relabeling is required, as in Fig. 10 C, the operator is prompted to provide corrections by selecting the correct bolt and location. This is shown in Fig. 10 D. The robot then executes the annotation demonstration (Fig. 10 F) and aggregates the relabeling and annotation demonstrations into the dataset. When a demonstration is collected, a model update is performed while the robot executes the demonstration.

The results for collecting 150 demonstrations are summarized in Fig. 11. The novice and system success rates in Fig. 11 A show that while the novice is learning to perform the task, the system can maintain a high success rate by querying based on SAG. The composition of the dataset in Fig. 11 B shows a similar trend as in the simulated experiments. This figure shows that AIDA can learn from mostly validation demonstrations in the later stages of training. Also, it shows that relabeling demonstrations is possible not only in simulation but also in real-world scenarios. The sensitivity in Fig. 11 C shows that SAG can track the desired sensitivity level across the task.

5.4 Real-World Sorting Task

To further demonstrate AIDA's applicability in real-world scenarios, we integrated it with Spot's built-in primitive skills to perform a sorting task [4]. Since AIDA is designed to work with any robot with one or more skills, we selected Spot for its built-in grasping, walking, and placing capabilities. The task involves sorting objects into paper and organic waste bins, as illustrated in Fig. 12. For this demonstration, we trained a CLIPort agent using an interface similar to Fig. 10 with the command format: "*Put the* [object] *in the* [bin type] *bin.*" Since CLIPort requires a top-down RGB-D projection, we use the in-hand RGB-D camera to scan the environment from different perspectives. With these images we can obtain a top-down projection using the robot's joint states, odometry, and camera intrinsics. The teacher issues the command through a Gradio (Abid et al., 2019) interface at the beginning of an episode. Similar to Sec. 5.3, the teacher is queried if the novice's uncertainty exceeds the gating threshold, and the interface shows the planned pick and place locations. The teacher can choose to validate or relabel the novice's actions. The teacher can provide annotation demonstrations by clicking where to pick and where to place in the top-down projected image. Demonstrations are aggregated to the demonstration dataset at the end of the episode, and the policy is updated. In the meantime, the robot walks back to its initial position and rescans its environment.

---

[4] A video of this demonstration is available at `https://aida-paper.github.io`.

6 Discussion

This section discusses the results from Sec. 5 by revisiting the claims in Sec. 1. We also examine the limitations of the AIDA framework and its experimental evaluation.

6.1 Claims

Our experiments were designed to provide evidence for claims **C1-4** from Sec. 1. First, we claim that *SAG balances query count and system failures by tracking a desired sensitivity* (**C1**). We provide evidence for this claim by showing accurate sensitivity tracking and the consequences for the query and system success rate for multiple values of σ_{des} on an MNIST dataset aggregation task in Fig. 3. Moreover, we extend these results to simulated and real-world robot tasks in Sec. 8 and Fig. 11 C. Secondly, we claim that *FIER reduces the number of annotations needed to achieve a given success rate* (**C2**). We trained CLIPort agents on a set of benchmark tasks in simulation for 300 demonstrations and evaluated the performance of checkpoints saved every 50 demonstrations. The results from these experiments support this claim. The evaluation results in Fig. 5 show that AIDA performs equivalent or better compared to the active DAgger baseline, while Fig. 6 shows that AIDA requires significantly fewer teacher annotations. Further evidence for this claim is provided by the results in Fig. 7, showing that training with AIDA results in higher success rates for the same number of annotations. The results in Fig. 11 provide evidence that these results generalize to real-world tasks, as we see a similar composition of the dataset and similar performance improvements. Thirdly, we claim that *FIER enhances generalization to unseen scenarios by recasting failures to demonstrations* (**C3**). The results in Fig. 5 provide evidence for this claim, as the AIDA checkpoints outperform the active DAgger baseline on the unseen scenarios. This can be attributed to relabeling failures involving the distractor objects, resulting in demonstrations for the unseen scenarios. Finally, we claim that PIER improves the success rate and reduces the required annotations under domain shift compared to uniform sampling. Evidence for this claim is provided by the results in Fig. 9, where PIER outperforms uniform sampling under domain shift regarding success rate while requiring fewer annotations.

6.2 Limitations

AIDA is designed for tasks with sparse rewards and learning mid- to high-level control. While this covers many problems, it does not extend to applications requiring high-rate feedback from the teacher. Thus, AIDA is best suited for scenarios where a robot has access to predefined skills and can learn higher-level plans for these skills. Additionally, AIDA can be integrated with methods that are better suited for learning low-level control or longer-horizon reasoning. While FIER significantly improves performance, it relies on recasting failures as successes, which can be challenging in some applications. Additionally, although AIDA reduces the teacher's workload, it assumes the teacher can validate or relabel actions before execution. This may not always be feasible. In such cases, relabeling and validation can still occur post-execution. Tuning hyperparameters in real-world settings is challenging. However, our experiments show that a single set of hyperparameters, without extensive tuning, generalizes effectively across different tasks, suggesting that hyperparameter sensitivity is not strongly dependent on task descriptions. Finally, our evaluations primarily used policies with a CLIPort (Shridhar et al., 2021) architecture, but AIDA is not limited to this choice. It can be applied to any policy architecture where a teacher can determine the success of actions and provide demonstrations.

7 Conclusion

We have introduced the AIDA framework, which consists of three components: a gating procedure SAG that can track a desired sensitivity level; FIER for recasting novice actions to demonstrations; and PIER to prioritize replay experience based on uncertainty, novice success, and age. Our experiments show that AIDA allows for an effective balance between queries and failures based on a user-specified sensitivity level. It can significantly enhance task performance while also improving its generalization capabilities to unseen scenarios. By utilizing demonstrations acquired through interactive relabeling and validation, AIDA can reduce the number of annotation demonstrations needed. Additionally, it improves adaptation to domain shifts by prioritizing replay.

For future work, several promising directions emerge for further enhancing the capabilities of AIDA. Extending this methodology to scenarios involving longer horizons and non-sparse rewards could improve its applicability. Leveraging generative/foundation models and simulators for visualizing action plans and generating artificial demonstrations could also expand the application domain of AIDA, potentially improving training efficiency and outcome predictability. Moreover, conducting participant studies would be vital to assess the mental load associated with AIDA and refine user interfaces accordingly. Finally, adapting AIDA to environments characterized by heterogeneous or imperfect teachers could widen its applicability and effectiveness in real-world settings.

References

P. Abbeel, A. Coates, and A. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *Intl. Journal of Robotics Research (IJRR)*, 29(13):1608–1639, 2010.

A. Abid, A. Abdalla, A. Abid, D. Khan, A. Alfozan, and J. Zou. Gradio: Hassle-free sharing and testing of ML models in the wild. *arXiv preprint*, 2019.

M.-R. Amini and P. Gallinari. Semi-supervised logistic regression. In *Proc. of the Europ. Conf. on Artificial Intelligence (ECAI)*, pp. 11, 2002.

M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, OpenAI, and W. Zaremba. Hindsight experience replay. In *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, 2017.

A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. RT-1: Robotics transformer for real-world control at scale. *arXiv preprint*, 2022.

C. Celemin and J. Kober. Knowledge-and ambiguity-aware robot learning from corrective and evaluative feedback. *Neural Computing and Applications*, pp. 1–19, 2023.

C. Celemin, R. Pérez-Dattari, E. Chisari, G. Franzese, L. de Souza Rosa, R. Prakash, Z. Ajanović, M. Ferraz, A. Valada, J. Kober, et al. Interactive imitation learning in robotics: A survey. *Foundations and Trends in Robotics*, 10(1-2):1–197, 2022.

S. Chernova and M. Veloso. Confidence-based policy learning from demonstration using gaussian mixture models. In *Proc. of the Intl. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1–8, 2007.

Y. Cui, D. Isele, S. Niekum, and K. Fujimura. Uncertainty-aware data aggregation for deep imitation learning. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 761–767. IEEE, 2019.

G. Datta, R. Hoque, A. Gu, E. Solowjow, and K. Goldberg. IIFL: Implicit interactive fleet learning from heterogeneous human supervisors. In *Proc. of the Conf. on Robot Learning (CoRL)*, volume 229, pp. 2340–2356. PMLR, 2023.

G. Franzese, C. Celemin, and J. Kober. Learning interactively to resolve ambiguity in reference frame selection. In *Proc. of the Conf. on Robot Learning (CoRL)*. PMLR, 2020.

Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proc. of the Intl. Conf. on Machine Learning (ICML)*, pp. 1050–1059. PMLR, 2016.

C. Gokmen, D. Ho, and M. Khansari. Asking for help: Failure prediction in behavioral cloning through value approximation. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 5821–5828, 2023.

D. H. Grollman and O. C. Jenkins. Dogged learning for robots. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 2483–2488. IEEE, 2007.

R. Hoque, A. Balakrishna, E. Novoseller, A. Wilcox, D. S. Brown, and K. Goldberg. ThriftyDAgger: Budget-aware novelty and risk gating for interactive imitation learning. In *Proc. of the Conf. on Robot Learning (CoRL)*, volume 164 of *Proceedings of Machine Learning Research*, pp. 598–608. PMLR, 2022.

R. Hoque, L. Y. Chen, S. Sharma, K. Dharmarajan, B. Thananjeyan, P. Abbeel, and K. Goldberg. Fleet-DAgger: Interactive robot fleet learning with scalable human supervision. In *Proc. of the Conf. on Robot Learning (CoRL)*, pp. 368–380. PMLR, 2023.

E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. BC-Z: Zero-shot task generalization with robotic imitation learning. In *Proc. of the Conf. on Robot Learning (CoRL)*, pp. 991–1002. PMLR, 2022.

M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer. HG-DAgger: Interactive imitation learning with human experts. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 8077–8083, 2019.

B. Kim and J. Pineau. Maximum mean discrepancy imitation learning. In *Proc. of Robotics: Science and Systems (RSS)*, 2013.

A. Lafage and O. Laurent. TorchUncertainty. `github.com/ENSTA-U2IS-AI/torch-uncertainty`, 2024.

M. Laskey, S. Staszak, W. Y.-S. Hsieh, J. Mahler, F. T. Pokorny, A. D. Dragan, and K. Goldberg. SHIV: Reducing supervisor burden in dagger using support vectors for efficient learning from demonstrations in high dimensional state spaces. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 462–469. IEEE, 2016.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, 1998.

S.-W. Lee and Y.-L. Kuo. Diff-DAgger: Uncertainty estimation with diffusion policy for robotic manipulation. *arXiv preprint*, 2024.

A. Li and T. Silver. Embodied active learning of relational state abstractions for bilevel planning. In *Proc. of the Conf. on Lifelong Learning Agents (CoLLAs)*, pp. 358–375. PMLR, 2023.

Y. Li and C. Zhang. Ensemble-based interactive imitation learning. *Computing Research Repository (CoRR)*, 2023.

H. Liu, S. Dass, R. Martín-Martín, and Y. Zhu. Model-based runtime monitoring with interactive imitation learning. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2024.

J. Luijkx, Z. Ajanović, L. Ferranti, and J. Kober. PARTNR: Pick and place ambiguity resolving by trustworthy interactive learning. In *NeurIPS Robot Learning Workshop: Trustworthy Robotics*, 2022.

J. Luo, P. Dong, Y. Zhai, Y. Ma, and S. Levine. RLIF: Interactive imitation learning as reinforcement learning. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2024.

K. Menda, K. Driggs-Campbell, and M. Kochenderfer. DropoutDAgger: A bayesian approach to safe imitation learning. *arXiv preprint*, 2017.

K. Menda, K. Driggs-Campbell, and M. J. Kochenderfer. EnsembleDAgger: A bayesian approach to safe imitation learning. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 5041–5048, 2019.

Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y.-L. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. In *Proc. of Robotics: Science and Systems (RSS)*, 2024.

D. A. Pomerleau. ALVINN: An autonomous land vehicle in a neural network. *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, 1, 1988.

A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *Proc. of the Intl. Conf. on Machine Learning (ICML)*, pp. 8748–8763. PMLR, 2021.

S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, et al. A generalist agent. *arXiv preprint*, 2022.

A. Z. Ren, A. Dixit, A. Bodrova, S. Singh, S. Tu, N. Brown, P. Xu, L. Takayama, F. Xia, J. Varley, et al. Robots that ask for help: Uncertainty alignment for large language model planners. In *Proc. of the Conf. on Robot Learning (CoRL)*, pp. 661–682. PMLR, 2023.

S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proc. of the Intl. Conf. on Artificial Intelligence and Statistics (AISTATS)*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635. PMLR, 2011.

T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *arXiv preprint*, 2015.

M. Shridhar, L. Manuelli, and D. Fox. CLIPort: What and where pathways for robotic manipulation. In *Proc. of the Conf. on Robot Learning (CoRL)*, 2021.

J. Spencer, S. Choudhury, M. Barnes, M. Schmittle, M. Chiang, P. Ramadge, and S. Srinivasa. Learning from interventions: Human-robot interaction as both explicit and implicit feedback. In *Proc. of Robotics: Science and Systems (RSS)*, 2020.

B. van der Heijden, J. Luijkx, L. Ferranti, J. Kober, and R. Babuska. Engine Agnostic Graph Environments for Robotics (EAGERx): A graph-based framework for sim2real robot learning. *IEEE Robotics and Automation Magazine (RAM)*, pp. 2–15, 2024.

J. Wong, A. Tung, A. Kurenkov, A. Mandlekar, L. Fei-Fei, S. Savarese, and R. Martín-Martín. Error-aware imitation learning from teleoperation data for mobile manipulation. In *Proc. of the Conf. on Robot Learning (CoRL)*, 2021.

C. Wu. CLIPort-Batchify. `github.com/ChenWu98/cliport-batchify`, 2024.

A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. In *Proc. of the Conf. on Robot Learning (CoRL)*, pp. 726–747. PMLR, 2021.

J. Zhang and K. Cho. Query-efficient imitation learning for end-to-end simulated driving. In *Proc. of the Conference on Advancements of Artificial Intelligence (AAAI)*, 2017.

A Appendix

A.1 Sensitivity-Aware Gating (SAG) Ablations

Sensitivity-Aware Gating (SAG) involves two regression steps, as shown in Alg. 2 and Fig. 2. The first step is linear regression to normalize $\boldsymbol{u}_W$, and the second is logistic regression to impute pseudo labels for cases without teacher feedback ($r = 0$). To evaluate the impact of these steps, we performed ablations in the same experimental setup as described in Sec. 5.1. We conducted dataset aggregation on the MNIST handwritten dataset under two conditions: one without normalization of $\boldsymbol{u}_W$ (w/o Normalization) and one without pseudo-label imputation (w/o Imputation). Fig. 13 and Tab. 1 compare these results to SAG with both normalization and imputation.

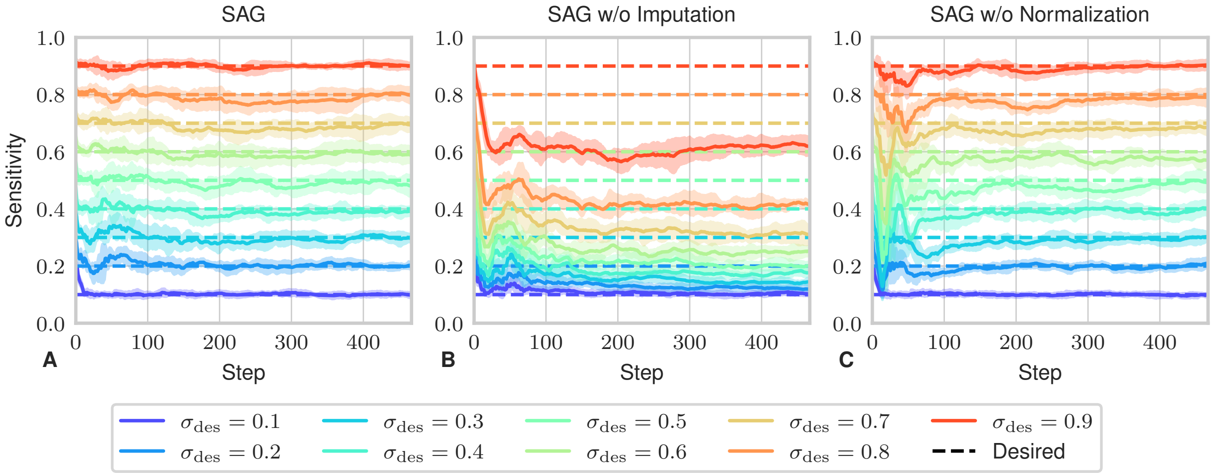


Figure 13: Comparison of SAG, SAG without pseudo-label imputation (w/o Imputation), and SAG without uncertainty normalization (w/o Normalization). Sensitivity is calculated over a moving window of 1000 failures. Mean and standard deviation are shown for ten repetitions. The plots show that SAG w/o Imputation sets the threshold too high, as it primarily uses labels from high-uncertainty regions. SAG w/o Normalization performs poorly in the early training stages due to rapidly decreasing uncertainty, which also results in an overly high threshold. A too high threshold will result in too low sensitivity, as not enough failures are prevented through querying.

Table 1: Comparison of SAG, SAG without pseudo-label imputation (w/o Imputation), and SAG without uncertainty normalization (w/o Normalization). Sensitivity is calculated over the complete training duration. Mean and standard deviation are shown for ten repetitions. Boldface is used to highlight the values corresponding to the best sensitivity tracking.

σ_{des}	SAG	SAG w/o Imputation	SAG w/o Normalization
0.1	0.104 ± 0.003	0.114 ± 0.003	$\mathbf{0.103 \pm 0.002}$
0.2	$\mathbf{0.207 \pm 0.005}$	0.141 ± 0.005	0.190 ± 0.004
0.3	$\mathbf{0.304 \pm 0.009}$	0.171 ± 0.003	0.272 ± 0.006
0.4	$\mathbf{0.397 \pm 0.007}$	0.200 ± 0.005	0.365 ± 0.010
0.5	$\mathbf{0.497 \pm 0.008}$	0.236 ± 0.005	0.448 ± 0.010
0.6	$\mathbf{0.598 \pm 0.007}$	0.283 ± 0.009	0.548 ± 0.012
0.7	$\mathbf{0.695 \pm 0.005}$	0.350 ± 0.009	0.655 ± 0.013
0.8	$\mathbf{0.793 \pm 0.010}$	0.447 ± 0.008	0.764 ± 0.005
0.9	$\mathbf{0.899 \pm 0.004}$	0.628 ± 0.008	0.880 ± 0.007

The sensitivity plots and table show a clear performance degradation for SAG w/o Imputation. Without imputing pseudo labels where success is unknown, the threshold is set too high, as gating results in labels for only the high-uncertainty regions. This leads to overly low sensitivity values. Tab. 1 also indicates performance degradation when $\boldsymbol{u}_W$ is not normalized. Fig. 13 suggests that this primarily results from poor sensitivity tracking in the early training stages. Uncertainty drops rapidly during early training. Therefore, failing to normalize $\boldsymbol{u}_W$ causes the threshold to be set too high. As a result, sensitivity remains too low in these early stages.

We also performed an ablation to study the influence of the value of the random query probability p_{rand}. We compared the experiments from Sec. 5.1 with $p_{\text{rand}} = 0.1$ to training with $p_{\text{rand}} = 0.05$ and $p_{\text{rand}} = 0.2$. The results of this comparison are shown in Fig. 14, Tab. 2 and Tab. 3. The sensitivity plots in Fig. 14 A-C show that SAG can track multiple sensitivity levels for different values of p_{rand}. However, for $p_{\text{rand}} = 0.2$ and $\sigma_{\text{des}} = 0.1$ it fails. This results from SAG being unable to track sensitivities lower than p_{rand}. This follows from Eq. (3):

$$\begin{aligned} \mathbb{E}_{\epsilon \sim U_{[0,1)}}[\sigma] &= \sigma_\gamma + p_{\text{rand}}(1 - \sigma_\gamma) && (7) \\ &\geq p_{\text{rand}}, && (8) \end{aligned}$$

since $0 \leq \sigma_\gamma \leq 1$. The specificity plots in Fig. 14 D-F show that increasing p_{rand} results in lower specificity values. This is also reflected in Tab. 3. This table shows the informedness values for the different values of p_{rand}. The informedness is also known as the Youden's J statistic and is calculated as:

$$\text{informedness} = \text{sensitivity} + \text{specificity} - 1. \tag{9}$$

It quantifies the performance of a dichotomous diagnostic test:

- below zero means worse than random;
- 0 equals random performance;
- Above zero indicates some degree of informed decision-making.

In table Tab. 3, we see that increasing p_{rand} results in lower informedness, as a higher percentage of queries results from random querying, rather than from querying based on exceeding the uncertainty threshold.

Table 2: Comparison of SAG with different values of p_{rand}. Sensitivity is calculated over the complete training duration. Mean and standard deviation are shown for ten repetitions. Boldface is used to highlight the values corresponding to the best sensitivity tracking.

σ_{des}	sensitivity $p_{\text{rand}} = 0.05$	sensitivity $p_{\text{rand}} = 0.1$	sensitivity $p_{\text{rand}} = 0.2$
0.1	0.113 ± 0.006	$\mathbf{0.104 \pm 0.003}$	0.200 ± 0.003
0.2	$\mathbf{0.198 \pm 0.012}$	0.207 ± 0.005	$\mathbf{0.202 \pm 0.003}$
0.3	$\mathbf{0.298 \pm 0.010}$	0.304 ± 0.009	0.309 ± 0.005
0.4	$\mathbf{0.399 \pm 0.006}$	0.397 ± 0.007	0.406 ± 0.007
0.5	0.493 ± 0.009	$\mathbf{0.497 \pm 0.008}$	0.507 ± 0.007
0.6	0.591 ± 0.006	0.598 ± 0.007	$\mathbf{0.599 \pm 0.005}$
0.7	0.696 ± 0.010	0.695 ± 0.005	$\mathbf{0.699 \pm 0.005}$
0.8	$\mathbf{0.799 \pm 0.010}$	0.793 ± 0.010	$\mathbf{0.799 \pm 0.006}$
0.9	0.897 ± 0.003	0.899 ± 0.004	$\mathbf{0.900 \pm 0.006}$

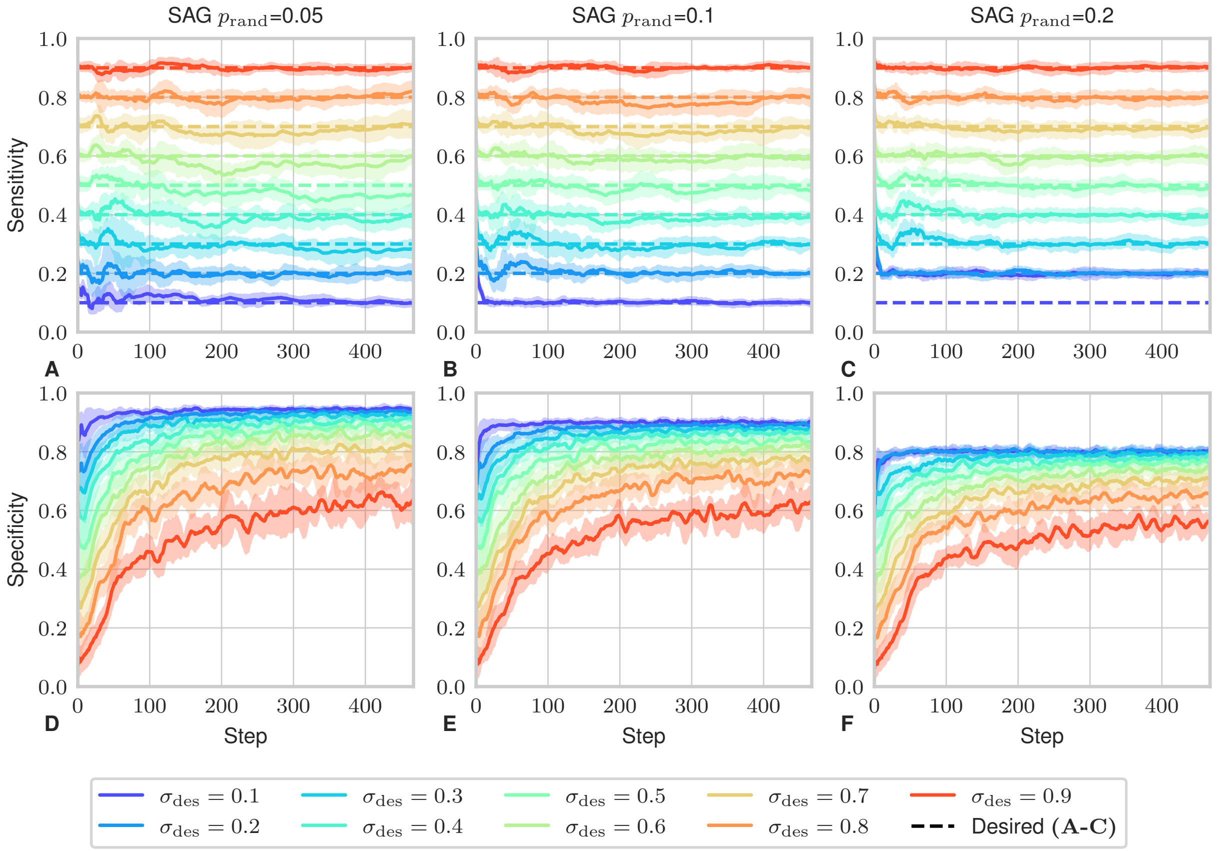


Figure 14: Comparison of SAG with different values of p_{rand} (0.05 shown, 0.1 and 0.2). Sensitivity in **A-C** and specificity in **E-F** are calculated over a moving window of 1000 failures and successes, respectively. Mean and standard deviation are shown for ten repetitions. SAG can track a desired sensitivity for different values of p_{rand}. However, it fails for $\sigma_{\mathrm{des}} = 0.1$ when $p_{\mathrm{rand}} = 0.2$. This happens because the minimum trackable sensitivity is equal to p_{rand}. The plots **E-F** show that increasing p_{rand} leads to lower specificity values.

Table 3: Comparison of SAG with different values of p_{rand}. Informedness (Youden's J statistic = sensitivity + specificity − 1) is calculated over the complete training duration. Mean and standard deviation are shown for ten repetitions. Boldface is used to highlight the values corresponding to the best informedness.

σ_{des}	informedness $p_{\mathrm{rand}} = 0.05$	informedness $p_{\mathrm{rand}} = 0.1$	informedness $p_{\mathrm{rand}} = 0.2$
0.1	$\mathbf{0.055 \pm 0.006}$	0.003 ± 0.004	-0.000 ± 0.004
0.2	$\mathbf{0.126 \pm 0.011}$	0.092 ± 0.004	0.002 ± 0.004
0.3	$\mathbf{0.209 \pm 0.012}$	0.170 ± 0.010	0.094 ± 0.007
0.4	$\mathbf{0.284 \pm 0.007}$	0.245 ± 0.007	0.171 ± 0.010
0.5	$\mathbf{0.349 \pm 0.013}$	0.311 ± 0.015	0.249 ± 0.010
0.6	$\mathbf{0.410 \pm 0.010}$	0.375 ± 0.013	0.308 ± 0.010
0.7	$\mathbf{0.464 \pm 0.011}$	0.427 ± 0.014	0.365 ± 0.011
0.8	$\mathbf{0.483 \pm 0.025}$	0.460 ± 0.019	0.407 ± 0.015
0.9	$\mathbf{0.456 \pm 0.027}$	0.445 ± 0.014	0.403 ± 0.016

A.2 Prioritized Interactive Experience Replay (PIER) Intuition

Prioritized Interactive Experience Replay (PIER) prioritizes the replay of demonstrations based on novice success, uncertainty, and demonstration age. In Fig. 15, we visualize Eq. (6) to provide a more intuitive understanding of the prioritization scheme. The red lines correspond to novice failures ($r = -1$), while the green lines correspond to novice successes ($r = 1$). The black lines indicate $r = 0$, which occurs, for example, for relabeled demonstrations where the goal was relabeled, and we do not know whether the novice would act correctly. In this figure, we observe that for values of b close to 1, the priorities converge almost linearly to 1, while for higher values of b, they converge exponentially.

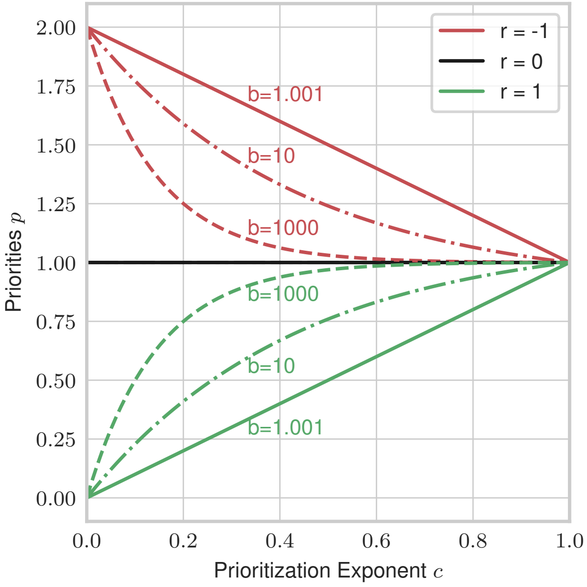


Figure 15: Visualization of Eq. (6) showing priorities p over prioritization exponents c for various values of base b.

In summary, we prioritize demonstrations in the following order (from highest to lowest):

1. Novice failure demonstrations that are recent and have low uncertainty.
2. Novice failure demonstrations that are either recent or have low uncertainty.
3. Novice failure demonstrations that are neither recent nor have low uncertainty.
4. Offline or relabeled demonstrations.
5. Novice success demonstrations that are neither recent nor have low uncertainty.
6. Novice success demonstrations that are either recent or have low uncertainty.
7. Novice success demonstrations that are both recent and have low uncertainty.

A.3 Foresight Interactive Experience Replay (FIER) Oracle

The oracle used in the experiments from Sec. 5.2 is based on Shridhar et al. (2021). We extend this oracle to provide relabeling demonstrations. Instead of querying a human teacher about the validity of the novice's planned actions, we execute these actions in simulation to verify the plan and then reset the environment to its previous state.

A.4 CLIPort Implementation

Our implementation of CLIPort agents builds on Shridhar et al. (2021) with modifications for interactive imitation learning. Efficient model updates are essential in this setting, as saving multiple checkpoints and performing evaluation rollouts after each update is impractical. Thus, training stability is a priority. We replace rectified linear units (ReLUs) with leaky ReLUs to prevent vanishing gradients. We also use a larger batch size (8 in simulation, 3 in real-world experiments) than the original (1) (Shridhar et al., 2021). Our batch training implementation is based on Wu (2024). Since batch training increases memory requirements, we reduce the model size by removing the two middle layers from the ResNet streams with lateral connections, the first language fusion layer, and its associated upscaling and lateral fusion layers.

A.5 Compute Resources

The MNIST experiments (Sec. 5.1) and real-world experiments (subsection 5.3 and subsection 5.4) were performed using an RTX 3080 Mobile graphics card. The CLIPort simulation benchmark experiments (Sec. 5.2) were performed using multiple A40 graphics cards on a high-performance computing cluster.