

# ARMADA: Autonomous Online Failure Detection and Human Shared Control Empower Scalable Real-world Deployment and Adaptation

Wenye Yu<sup>1,2</sup>, Jun Lv<sup>1,3</sup>, Zixi Ying<sup>3</sup>, Yang Jin<sup>1</sup>, Chuan Wen<sup>1,†</sup>, Cewu Lu<sup>1,2,3,†</sup>

<sup>1</sup>Shanghai Jiao Tong University <sup>2</sup>Shanghai Innovation Institute <sup>3</sup>Noematrix Ltd.

<sup>†</sup>Equal advising

Project page: <https://virlus.github.io/armada/>

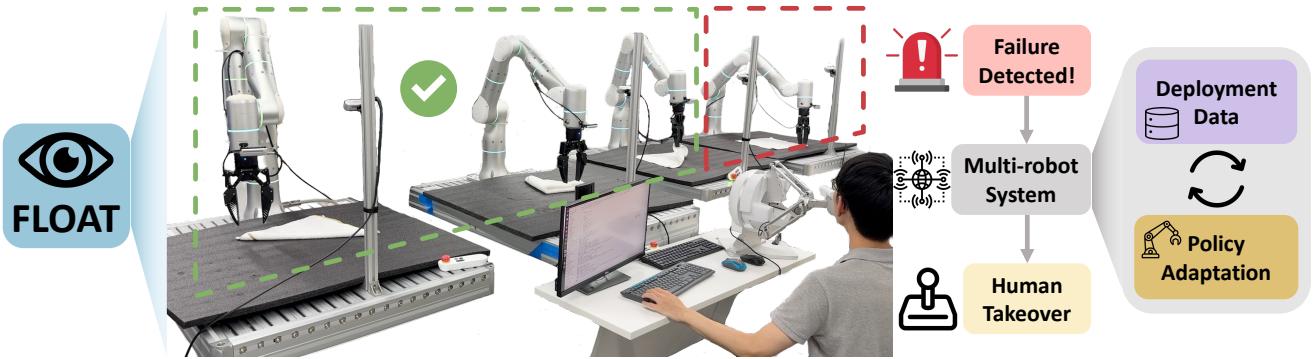


Fig. 1: **Illustration of ARMADA.** ARMADA makes use of FLOAT failure detector and enables paralleled policy rollout on multiple robots, only requesting human intervention when necessary. The deployment data collected online are then utilized for policy improvement, forming a scalable deployment and adaptation loop.

**Abstract**— Imitation learning has shown promise in learning from large-scale real-world datasets. However, pretrained policies usually perform poorly without sufficient in-domain data. Besides, human-collected demonstrations entail substantial labour and tend to encompass mixed-quality data and redundant information. As a workaround, human-in-the-loop systems gather domain-specific data for policy post-training, and exploit closed-loop policy feedback to offer informative guidance, but usually require full-time human surveillance during policy rollout. In this work, we devise ARMADA, a multi-robot deployment and adaptation system with human-in-the-loop shared control, featuring an autonomous online failure detection method named FLOAT. Thanks to FLOAT, ARMADA enables paralleled policy rollout and requests human intervention only when necessary, significantly reducing reliance on human supervision. Hence, ARMADA enables efficient acquisition of in-domain data, and leads to more scalable deployment and faster adaptation to new scenarios. We evaluate the performance of ARMADA on four real-world tasks. FLOAT achieves nearly 95% accuracy on average, surpassing prior state-of-the-art failure detection approaches by over 20%. Besides, ARMADA manifests more than 4× increase in success rate and greater than 2× reduction in human intervention rate over multiple rounds of policy rollout and post-training, compared to previous human-in-the-loop learning methods.

## I. INTRODUCTION

Recent years have witnessed the burgeoning of data-driven approaches in the field of robotic manipulation [8, 54, 20]. In pursuit of policies for real-world deployment, joint efforts have been made to build large-scale datasets for policy learning. Some prior works gather human-collected data from multiple sources and integrate them into large-

scale heterogeneous datasets [34, 19, 11]. Another line of work introduces unified data collection systems, including embodiment-agnostic handheld devices [9, 39, 49] and low-cost exoskeletons [12, 5, 13]. Through pretraining on these datasets, we obtain policies that have rudimentary ability of performing specific tasks in real-world scenarios.

However, pretrained policies usually lack robustness during deployment due to the deficiency of in-domain data. Besides, as indicated in previous works on data curation [16, 6, 52], human-collected demonstrations often contain segments with mixed quality and redundant information, which impedes robots from gaining superior performance. To this end, prior works explore human-in-the-loop systems, where human operators and learned policies control the robot in a shared manner. These systems enable interactive collection of domain-specific data, which are utilized for policy post-training and help adapt the policies to the given scenario. Moreover, by observing robot behaviour during online rollouts, human operators are able to offer informative guidance with closed-loop policy feedback. Some works allow human operators to intervene in policy rollout when the robot fails to accomplish the given tasks, and take advantage of human correction data for policy post-training [23, 25, 43, 18, 46]. Others embrace the idea of shared autonomy and develop joint control between human and robot by a time-varying ratio [28, 51].

Nevertheless, most of these human-in-the-loop systems require full-time surveillance from human operators so that they can spot task failures in time and help robots recover.

This confines the current systems to a one-to-one control setting where each human operator attends to a single robot. In order to further enhance the scalability of human-in-the-loop systems, we hold that there are two desiderata: **1)** A real-time failure detection module that monitors policy rollout can help alert human operators of possible task failures and thus alleviate reliance on full-time human supervision. **2)** A multi-robot shared control system scales up policy deployment and post-training for faster adaptation to new scenarios.

To this end, we propose a scalable real-world robot system featuring autonomous online failure detection and multi-robot shared control, dubbed **ARMADA** (Autonomous Real-world Multi-robot system with human Assistance for Deployment and Adaptation). Concretely, we devise an online failure detection method for visuomotor imitation learning algorithms based on policy embedding. Given a pretrained visuomotor policy and expert demonstrations that constitute the training data, we perform online trajectory matching between current policy rollouts and expert trajectories and compute the “distance” between the matched trajectories in terms of their policy embeddings using Optimal Transport (OT) [14, 35, 10], which we refer to as the **FLOAT** index (FaiLure detection based on OptimAl Transport). Utilizing the FLOAT indices of all successful rollout trajectories, we then define a universal FLOAT threshold, which serves as a real-time and plug-in-and-play approach to online failure detection. More importantly, we implement a multi-robot shared control system that allows for autonomous policy rollouts and timely human intervention when our failure detection model raises warning, significantly improving the efficiency of human-in-the-loop systems. Our system can be easily adapted to various imitation learning policies, human intervention patterns (teleoperation, exoskeleton, etc.), and embodiments. The entire implementation of ARMADA will be open-sourced.

We carry out comprehensive experiments to verify the effectiveness of our novel failure detection approach and multi-robot system. Across four real-world tasks, FLOAT lifts the average accuracy to nearly 95%, which is an improvement of over 20% compared to state-of-the-art approaches. Besides, over several rounds of rollout and fine-tuning, ARMADA achieves more than 4 $\times$  increase in success rate and greater than double reduction in human intervention rate compared to previous human-in-the-loop shared control systems.

In a nutshell, our contributions are three-fold:

- 1) We devise FLOAT, a plug-in-and-play online failure detection system for visuomotor imitation learning methods that achieves nearly 95% accuracy in real world.
- 2) We implement ARMADA, a multi-robot system with human-in-the-loop shared control that enables paralleled and autonomous robot rollouts, empowering scalable real-world deployment and adaptation.
- 3) ARMADA, over multiple rounds of post-training, leads to a more than four-fold increase in success rate and a greater than two-fold decrease in human intervention ratio compared to previous human-in-the-loop learning approaches that require full-time human supervision.

## II. RELATED WORK

### A. Human-in-the-loop learning in robot manipulation

Human-in-the-loop learning exploits interactive human signals to assist policy learning [18, 30, 42]. Utilizing a priori knowledge of human operators, these methods introduce inductive bias that provides strong supervision. Human can be involved in the learning process through various forms. For instance, previous works utilize human preference ranking to align robot behaviour with human preferences through Reinforcement Learning approaches [41, 17, 21, 7, 53]. Besides, some prior works directly steer the learned policy in compliance with human preferences without fine-tuning the policy itself [31, 47, 44]. Shared autonomy also emerges as one of the human-in-the-loop patterns, where human operators share control with the policy by a time-varying ratio, and guide the policy to achieve the given tasks over time [28, 51]. Other works require human supervision during policy rollout, and solicit human intervention in case of potential task failure [23, 46, 27, 25]. The corrective behaviour by human operators is then utilized to fine-tune policies for better performance. The human-in-the-loop system we propose falls into this category. However, thanks to our online failure detection method, we are able to relieve the burden of full-time human surveillance and allow for one-to-multiple shared control system, significantly accelerating real-world deployment and adaptation to new scenarios.

### B. Online failure detection for pretrained policies

Failure detection plays a crucial role during deployment of pretrained policies, especially for imitation learning methods due to their vulnerability to out-of-distribution environment settings [50, 37, 26]. Some prior works adopt an OOD detection perspective on failure detection in robot manipulation tasks. For example, Liu et al. [24] train a failure classifier using data from previous rollouts and detect OOD cases in task execution by K-Means clustering on policy embeddings [25]. However, these methods rely on failure data collected a priori, which are hard to scale up in real-world scenarios. Wong et al. [45] utilize Variational Auto Encoder (VAE) reconstruction error as a metric of OOD detection, but require large quantities of data for VAE training. Xu et al. [48] derive a time-varying Conformal Prediction band with learned failure scores. Nevertheless, these approaches are highly sensitive to changes in environment and susceptible to overfitting issues, as we will empirically validate in our experiments. Besides, some prior works make use of Large Language Models (LLM) and Vision-Language Models (VLM) to identify possible failures. Sinha et al. [38] build a failure classifier based on LLM embeddings, while Sentinel [2] and Genie Centurion [43] directly query VLM for judgement on whether failure occurs. These approaches either depend on the zero-shot ability of LLMs and VLMs or involve fine-tuning them with domain-specific data, which can be strenuous. Agia et al. [2] introduce statistical temporal action consistency (STAC) as a metric for real-time failure detection, which serves as the state-of-the-art method. Our

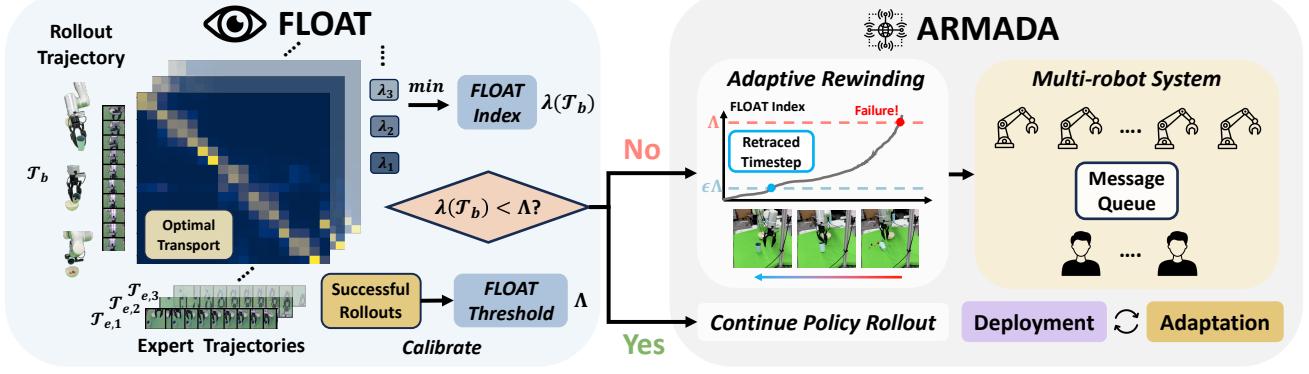


Fig. 2: **Method overview.** FLOAT failure detector conducts real-time OT matching between the policy embeddings of the current rollout and all expert demonstrations, and defines the minimum OT cost as FLOAT index. We thereby calibrate the FLOAT threshold on all successful rollouts. When the FLOAT index of a rollout trajectory exceeds the threshold, we consider it a failure and employ adaptive rewinding based on OT computation, which helps retrace a previous timestep before the scene was disturbed. Our multi-robot system then allocates an idle human operator to the failed robot for intervention.

approach features plug-in-and-play failure detection for visuomotor imitation learning methods, only requiring policy embeddings as input, and achieves empirical performance improvement of over 20% in terms of accuracy compared to the SOTA method, which will be detailed in Sec. V.

### C. Post-training of imitation learning policies

Data quality has a major influence on the performance of policy post-training [4, 22, 29, 36]. However, as pointed out by Zhang et al. [52] and Hejna et al. [16], human-collected demonstrations often suffer from mixed quality and redundant information. In order to filter trajectories that contain undesired behaviour in training data, prior works propose various data curation approaches. SCIZOR [52] adopts a self-supervised approach to removing trajectories containing redundant information and suboptimal behaviour. DemInf [16] evaluates data quality through mutual information estimators. Some other works estimate the influence of each demonstration on policy performance through online rollouts and refine training data accordingly [3, 6]. Hejna et al. [15] perform automated dataset-level mixture optimization for large-scale heterogeneous datasets, while Octo [40] adjusts the weight of each dataset in a heuristic manner. In this work, we improve data quality with our adaptive rewinding mechanism. By retracing a previous timestep in the rollout episode, the robot can recover from failure while human operators help reset the scene, ensuring an informative demonstration with human corrective behaviour.

## III. PRELIMINARIES

### A. Optimal Transport and its application in robotics

Optimal Transport (OT) is a mathematical theory on finding the most efficient way to move a distribution of mass from one location to another, minimizing the total transportation cost. On the other hand, imitation learning methods seek to optimize behaviour policy  $\pi_b$  in order that it stays close to the expert policy  $\pi_e$ , given access to offline expert demonstrations  $\mathcal{D}_e = \{\{(o_{e,t}^n, a_{e,t}^n)\}_{t=1}^{l_n}\}_{n=1}^N$ . Here,  $N$  denotes the number of expert trajectories and  $l_n$

denotes length of the  $n$ -th expert demonstration. OT thereby serves as a non-parametric approach to trajectory matching and reward computation, enabling policy optimization either by imitation learning [35, 32, 1] or reinforcement learning methods [14, 10]. Specifically, given an expert demonstration  $\mathcal{T}_e = \{(o_{e,t}, a_{e,t})\}_{t=1}^{l_e}$  and a policy rollout trajectory  $\mathcal{T}_b = \{(o_{b,t}, a_{b,t})\}_{t=1}^{l_b}$ , OT derives an optimal transport matrix  $(\mu_{ij}^*)^{l_e \times l_b}$  using the optimization objective in 1, where  $c(\cdot, \cdot)$  represents the cost function. For sake of simplicity, we denote  $\mu^* = \text{OT}(\mathcal{T}_e, \mathcal{T}_b)$  in the following sections.

$$\min_{\mu_{ij} \geq 0} \sum_{i=1}^{l_e} \sum_{j=1}^{l_b} c(o_{e,i}, o_{b,j}) \mu_{ij} \quad s.t. \quad \sum_{j=1}^{l_b} \mu_{ij} = \frac{1}{l_e}, \quad \sum_{i=1}^{l_e} \mu_{ij} = \frac{1}{l_b} \quad (1)$$

We can thereby calculate reward function for each rollout timestep through Eq. 2.

$$r_t = - \sum_{i=1}^{l_e} c(o_{e,i}, o_{b,t}) \mu_{i,t}^* \quad (2)$$

In this work, instead of using OT for reward computation, we propose a novel failure detection index based on OT, dubbed FLOAT, which will be expanded on in Sec. IV.

## IV. METHOD

To achieve scalable real-world deployment and adaptation, we propose ARMADA, a multi-robot shared control system equipped with FLOAT, an online failure detection method, and an adaptive rewinding mechanism. In this section, we elaborate on our system design: We first introduce FLOAT in Sec. IV-A. Thereon, we detail the design of ARMADA in Sec. IV-B. Finally, we present our policy architecture and training recipe in Sec. IV-C. We present an overview of our method in Figure 2.

### A. FLOAT: Our failure detector

We devise FLOAT, a novel failure detection approach for visuomotor imitation learning methods. Suppose a pretrained policy  $\pi$  with an observation encoder  $\phi$  has been trained

on expert demonstrations  $\mathcal{D}_e = \{\mathcal{T}_{e,n}\}_{n=1}^N$ , where  $\mathcal{T}_{e,n} = \{(o_{e,t}^n, a_{e,t}^n)\}_{t=1}^{l_n}$ . We first obtain the policy embeddings for all expert demonstrations as shown in 3.

$$\mathcal{F}_e = \{\{\phi(o_{e,t}^n)\}_{t=1}^{l_n}\}_{n=1}^N \quad (3)$$

Subsequently, during rollout, we extract all the policy embeddings up until the current step  $t_0$ , as shown in 4.

$$\mathcal{F}_b = \{\phi(o_{b,t})\}_{t=1}^{t_0} \quad (4)$$

We select cosine similarity as our cost function for OT computation, as aforementioned in Sec. III-A. Namely,  $c(o_{e,i}, o_{b,j}) := \cos(\phi(o_{e,i}), \phi(o_{b,j}))$ . Besides, we compute the OT plan between the current rollout trajectory  $\mathcal{T}_b = \{(o_{b,t}, a_{b,t})\}_{t=1}^{t_0}$  and every expert demonstration  $\mathcal{T}_{e,n}$ , denoted as  $\mu_n^* = \text{OT}(\mathcal{T}_{e,n}, \mathcal{T}_b)$ . Based on that, we define the FLOAT index for  $\mathcal{T}_b$  in Eq. 6, denoted  $\lambda(\mathcal{T}_b)$ .

$$\lambda_n(\mathcal{T}_b) = \sum_{i=1}^{l_n} \sum_{j=1}^{t_0} \mu_{n,i,j}^* c(o_{e,i}^n, o_{b,j}) \quad (5)$$

$$\lambda(\mathcal{T}_b) = \min_n \lambda_n(\mathcal{T}_b) \quad (6)$$

Intuitively, FLOAT traverses all expert demonstrations for the “closest” trajectory to  $\mathcal{T}_b$ , and takes the total OT cost between them as the failure index.

For online failure detection, we derive a universal FLOAT threshold  $\Lambda$  by calibrating on all successful rollouts  $\{\mathcal{T}_{b,k}\}_{k=1}^M$ . Concretely, we define  $\Lambda$  to be the  $1 - \delta$  percentile of  $\{\lambda(\mathcal{T}_{b,k})\}_{k=1}^M$ , where  $\delta$  is a time-varying hyperparameter. If the FLOAT index of a rollout trajectory exceeds  $\Lambda$ , a failure signal is raised. With policy rollout going on,  $\delta$  updates itself adaptively: Whenever the robot fails but FLOAT neglects the failure,  $\delta$  is lowered by a certain value  $\Delta\delta$ ; On the other hand, whenever FLOAT raises a failure signal but the robot is operating normally,  $\delta$  is increased by  $\Delta\delta$ .  $\Lambda$  is thereby updated according to  $\delta$  and all successful rollouts gathered online. It is worth mentioning that FLOAT runs asynchronously with policy rollout to prevent robots from the latency induced by OT computation. Details of FLOAT design are presented in Appendix A.

### B. ARMADA: Our multi-robot shared control system

Empowered by FLOAT, we are able to perform paralleled and autonomous policy rollout on multiple robots. To achieve this, we implement a message queue in charge of communication between robot nodes and human teleoperation nodes. In brief, the robot node puts a message for human intervention into the queue whenever FLOAT raises a failure signal. The teleoperation nodes receive the earliest message in the queue and assign the message to an idle human operator, who then intervenes the target robot rollout. An overview of ARMADA is presented in Algorithm 1.

Nonetheless, there are cases where robots might run into unrecoverable states during rollout. For instance, if the robot aims to grasp a cup of marbles but tips it over, the marbles will roll everywhere. Therefore, we design an adaptive

---

### Algorithm 1: System design of ARMADA

---

```

Input: Robot nodes  $\{R_i\}_{i=1}^m$ , Human teleoperation
nodes  $\{T_j\}_{j=1}^n$ , Message queue  $C$ 
Output: Collected demonstration buffer  $\mathcal{D}$ 
1  $\mathcal{D} \leftarrow \emptyset$ 
2 do in parallel for all  $R_i (1 \leq i \leq m)$ 
3   while True do
4      $R_i$  resets for new rollout episode
5     while  $R_i$ 's episode not finished do
6       if detected failure then
7          $C.put\_wait("R_i needs help")$ 
8       else
9         Take one environment step
10     $\mathcal{D} \leftarrow \mathcal{D} \cup \{R_i\text{'s current episode}\}$ 
11 do in parallel
12   while True do
13     if  $C$  not empty then
14        $i \leftarrow C.get\_robot\_idx(); C.pop()$ 
15        $j \leftarrow \text{Search\_for\_idle\_human\_node}()$ 
16       do in parallel
17          $T_j$  takes over  $R_i$  for failure recovery
18 return  $\mathcal{D}$ 

```

---

rewinding mechanism that allows the robot to retrace a previous timestep while human operators can help reset the scene as it was, thus ensuring an intact and informative demonstration with human corrective behaviour.

Suppose a failure signal is raised at the  $t_0$ -th timestep, and the current rollout trajectory  $(\mathcal{T}_b)_{1:t_0} = \{(o_{b,t}, a_{b,t})\}_{t=1}^{t_0}$ . We thereby derive the retraced timestep through the objective in 7, where  $\epsilon \in (0, 1)$  is a pre-defined hyperparameter. Intuitively, we search for the latest timestep with a corresponding FLOAT index lower than an adaptive threshold, determined by the current FLOAT index.

$$\arg \max_{t \geq 1} \mathbb{1}[\lambda((\mathcal{T}_b)_{1:t}) \leq \epsilon \lambda((\mathcal{T}_b)_{1:t_0})] \cdot t \quad (7)$$

In this way, the robot has a better chance to recover from failure and allows human to help reset the scene. We fix  $\epsilon = 0.2$  for all experiments in this work.

### C. Policy architecture and training recipe

We deploy Diffusion Policy [8] as our imitation learning method, and select transformer architecture as the action generation backbone. Besides, we employ pretrained DINov2 ViT-B/14 model [33] as our visual encoder for richer policy embeddings. To compress the high-dimensional latent produced by DINov2 encoder for OT computation, we append a linear head to the visual encoder. Detailed training recipe can be found in Appendix B.

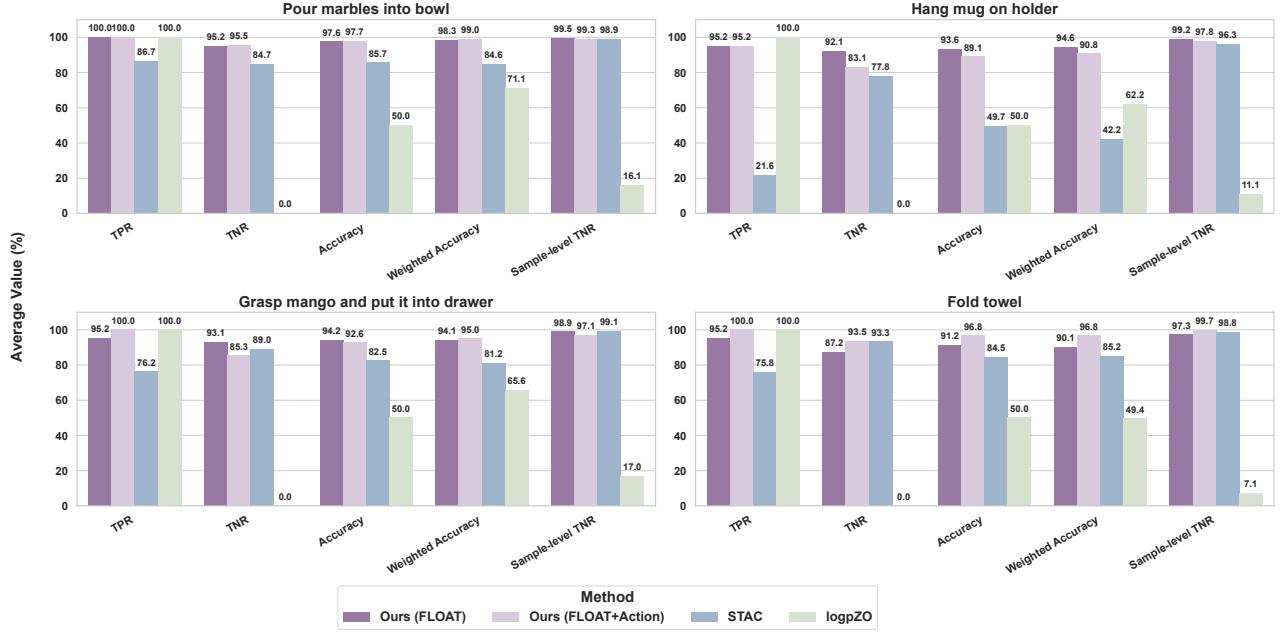


Fig. 3: **Failure detection experiment results.** FLOAT achieves nearly 95% accuracy across four tasks, which is an improvement of over 20% compared to state-of-the-art baseline methods. It manifests comparable performance to its variant which further integrates action inconsistency metric, showcasing the effectiveness of FLOAT in detecting various failures.

## V. EXPERIMENTS

We design the experiments to answer the following questions: (a) How does FLOAT perform compared to previous online failure detection methods? (b) Does our rewinding mechanism help produce demonstrations with better quality? (c) To what extent does ARMADA mitigate the reliance on human intervention over time? (d) How does ARMADA perform in improving scalability of deployment and adaptation to unseen scenarios?

**Task design.** The experiments are conducted on four real-world tasks: *Pour marbles into bowl*, *Hang mug on holder*, *Grasp mango and put it into drawer*, and *Fold towel*, which are illustrated in Figure 4.

**Hardware setup.** We employ an eye-to-hand and an eye-in-hand Intel Realsense D435i camera for image observation. We select Flexiv Rizon4 robot equipped with Robotiq 2F-85 gripper, whose fingers are changed into UMI [9] gripper, as our robot hardware. Besides, we deploy Force Dimension sigma.7 haptic interface as our teleoperation device.

**Evaluation protocol.** We initialize the training data with 50 human-teleoperated demonstrations for each task. During each rollout stage, we collect 30 demonstrations online, during which we conduct the failure detection experiments (namely 30 trials in each rollout stage). After that, we proceed to offline fine-tuning stage where online collected data are merged with initial human demonstrations to form the new training buffer. We carry out two fine-tuning stages and three rollout stages for each task. To reduce the occasionality of real-world evaluation, we strictly align the initial pose of robot and scene configuration for every trial in a certain rollout stage. Detailed evaluation settings can be found in Appendix C.

### A. How does FLOAT perform compared to previous online failure detection methods?

We compare FLOAT with two prominent baseline methods in online failure detection: **STAC** [2] serves as the state-of-the-art method in our experiments. By calculating the statistical distance between temporally overlapping regions of consecutive action chunk predictions, STAC derives a time-invariant threshold for imitation learning approaches. **logpZO** [48] trains a score model on the policy embeddings from successful rollouts, and calibrates time-varying thresholds based on a Conformal Prediction (CP) band.

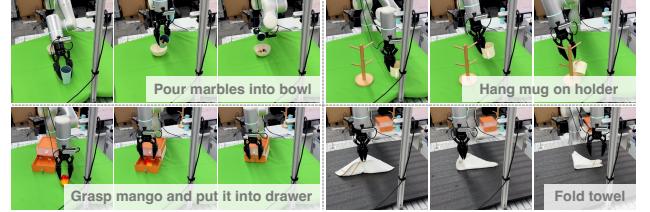


Fig. 4: **Real-world task setup.**

We select five evaluation metrics for failure detection experiments: true positive rate (TPR), true negative rate (TNR), accuracy, weighted accuracy, and sample-level TNR. We count a true positive when the failure detector raises a warning signal during a rollout where the policy fails. Correspondingly, we count a true negative when the failure detector never raises any warning in a rollout where the policy succeeds. The definition of accuracy and weighted accuracy are based on TPR, TNR, and task Success Rate (SR), as shown in 8 and 9. Moreover, we design sample-level TNR, a step-level metric aside from the four episode-level metrics above. It measures the rate of true negative steps in an episode before the policy fails, which is an important

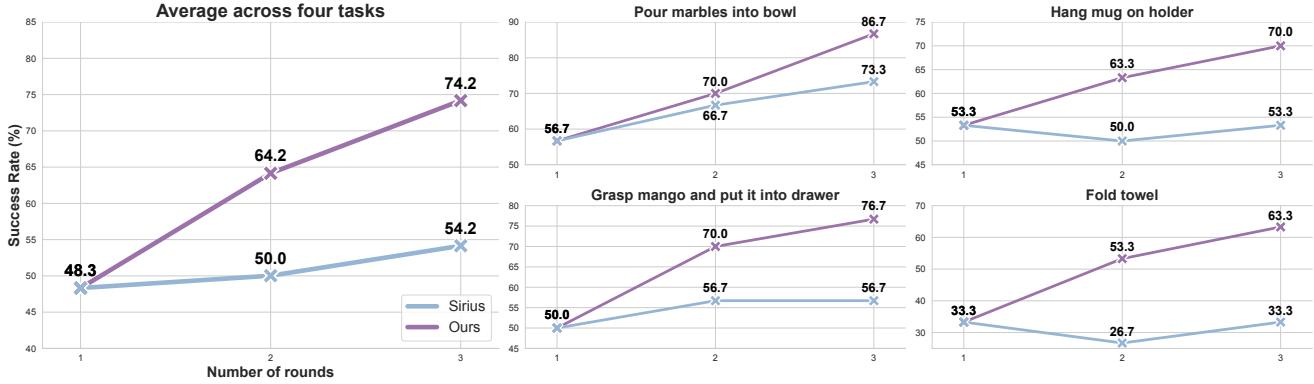


Fig. 5: **Success rate over three evaluation rounds.** ARMADA exhibits stable progress in success rate, with a more than four-fold increase compared to previous human-in-the-loop learning approach, thanks to our adaptive rewinding mechanism.

factor in multi-robot control systems. This is because if the failure detector keeps raising false positive warnings, it will be a heavy burden on system throughput.

$$\text{Accuracy} = \frac{\text{TPR} + \text{TNR}}{2} \quad (8)$$

$$\text{Weighted Accuracy} = \text{TPR} * \text{SR} + \text{TNR} * (1 - \text{SR}) \quad (9)$$

The results of failure detection experiments are presented in Figure 3. FLOAT shows superior performance and achieves nearly 95% accuracy across four real-world tasks, improving the state-of-the-art approach by over 20%. logpZO, on the other hand, suffers from severe overfitting issues and yields a very low TNR. We surmise this is due to its assumption that every rollout should be I.I.D (Independently and Identically Distributed), which is generally not the case in our experiments given the random initial pose of robots and large workspace of the tasks.

To further showcase the effectiveness of FLOAT, we introduce a variant which takes the detection results of FLOAT and STAC both into account, dubbed **FLOAT+Action**, which raises a warning when either of the failure detectors does. FLOAT obtains comparable performance to this variant across all four tasks, manifesting its ability to detect various failure modes without access to action predictions.

#### B. Does our rewinding mechanism help produce demonstrations with better quality?

To demonstrate the effectiveness of our adaptive rewinding mechanism, we compare the performance of policies trained respectively on data collected by ARMADA and the state-of-the-art human-in-the-loop learning method, **Sirius** [23]. Sirius requires full-time human surveillance, and reweights observation-action samples collected online for policy fine-tuning by emphasizing human intervention trajectories. We illustrate the change in success rate for both methods in Figure 5. Over three evaluation rounds, the policy trained with ARMADA yields an improvement in success rate by 25.9% on average, which is more than four times the improvement using Sirius. Specifically, ARMADA shows large progress in tasks such as *Hang mug on holder* and

*Fold towel*, while Sirius hardly makes any difference. This is attributed to the vulnerability of policies to unrecoverable states in these tasks. For instance, if the robot fails to hang the mug on holder, the mug might as well fall on the table and land on its side, making it difficult to accomplish the task even for human operators.

#### C. To what extent does ARMADA mitigate the reliance on human intervention over time?

In a human-in-the-loop learning framework, we expect the reliance on human intervention to decline as the policy gets more capable over post-training stages. To validate this, we inspect the human intervention rate of both ARMADA and Sirius over three evaluation rounds. As illustrated in Figure 7, though ARMADA requires more human intervention due to the adaptive rewinding mechanism in the first evaluation round, it manages to reduce human intervention rate by 23.3% after two fine-tuning stages, which is more than double compared to Sirius. This indicates the potential of ARMADA in facilitating the scalability of real-world deployment by enlarging the number of paralleled robots in the multi-robot shared control system with comparable human effort.



Fig. 6: **Multi-robot experiments on unseen scenarios.** We deploy the pretrained *Fold towel* policy on Scene A, B, and C (**in-domain**) for online data collection, and evaluate the post-trained policy on Scene D (**out-of-distribution**).

#### D. How does ARMADA perform in improving scalability of deployment and adaptation to unseen scenarios?

To verify the effectiveness of ARMADA in facilitating policy adaptation to novel scenarios, we deploy a pretrained *Fold towel* policy on three robots simultaneously for online data collection. It is worth noting that three different sets of distractions are also involved during deployment, as illustrated in Figure 6, so as to improve the generalization of post-trained policies to unseen domains. As an ablation,

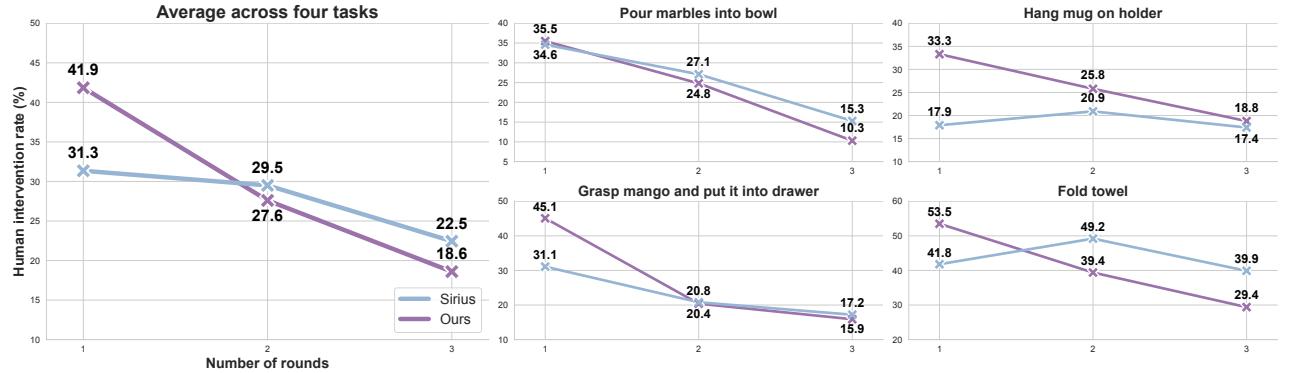


Fig. 7: **Human intervention rate over three evaluation rounds.** ARMADA results in a greater than two-fold reduction in human intervention rate compared to Sirius, showing potential in scalable deployment and adaptation.

we conduct another round of deployment with the same pretrained policy solely on Scene A from Figure 6. Each round of deployment last 20 minutes, ensued by policy post-training on the collected trajectories. The deployment stage and post-training stage alternate three times. Each post-trained policy is evaluated on an unseen scenario, as illustrated in Scene D from Figure 6. The success rates of all policies on the unseen scenario are presented in Figure 8.

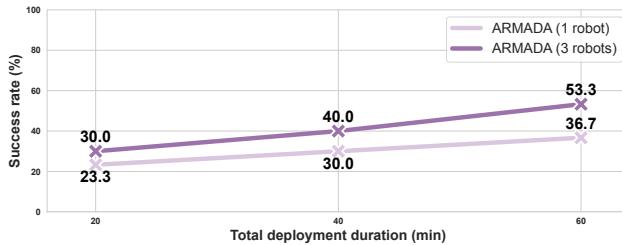


Fig. 8: **Success rate on unseen scenarios.** ARMADA boosts adaptation to unseen scenarios with paralleled policy deployment on multiple robots.

Through paralleled deployment on multiple robots and diverse scenes, ARMADA expedites policy adaptation to unseen scenarios, and manifests steady progress in task success rate with growing deployment duration. This is attributed to the increment in collected human intervention data and the diversity of scenarios as more robots are put into use. To validate ARMADA’s efficiency in yielding human intervention data from various domains, we also examine the occupancy of human operator, measured by the time span of human intervention in a single deployment stage. Intuitively, the more occupied the human operator, the more efficient the system in gathering valuable human intervention data. The results are shown in Figure 9.

With the same time consumption, ARMADA benefits from paralleled policy rollout, acquiring over twice the human intervention samples compared to a single-robot setting. The high-quality human trajectories on various domains naturally promote policy’s robustness to novel scenarios, verifying ARMADA’s practicability in more scalable policy adaptation and better utilization of human guidance.

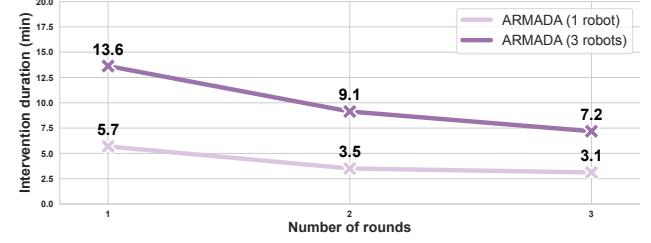


Fig. 9: **Human intervention duration over three deployment rounds.** ARMADA scales up collection of human intervention trajectory with more robots in parallel.

## VI. LIMITATIONS AND DISCUSSION

FLOAT, though effective in single-task settings, still requires human-collected expert demonstrations as references for trajectory matching and cannot extend to novel tasks and embodiments. We expect future work to focus on building general-purpose progress estimators for robot manipulation tasks, which are able to perform online failure detection across various tasks and hardware settings. We believe that this would further enhance the scalability of real-world deployment and adaptation with the help of multi-robot shared control systems such as ARMADA.

## VII. CONCLUSIONS

This paper introduces ARMADA, a scalable multi-robot system for real-world deployment and adaptation, featuring an autonomous online failure detection method, FLOAT, which achieves nearly 95% accuracy in real world. By integrating FLOAT with an adaptive rewinding mechanism, ARMADA significantly reduces the need for human supervision over multiple post-training stages, demonstrating large improvement in task success rate and salient reduction in human intervention ratio.

## APPENDIX

### A. Details of FLOAT design

The main hyperparameters in FLOAT are shown in Table I. We adopt Sinkhorn approximation to the original OT matching objective because solving 1 directly is computationally expensive. Besides, the expert demonstrations  $\mathcal{D}_e = \{\mathcal{T}_{e,n}\}_{n=1}^N$  have varied lengths, which might lead to

poor OT matching between the current rollout and the expert trajectory. As a remedy, we pad all the expert demonstrations by repeating the observations at their last timestep to a unified length, which we select as the maximum length among all expert trajectories, namely  $l_{\max} := \max_{1 \leq i \leq N} l_i$ .  $l_{\max}$  also serves as the upper bound of rollout length in our multi-robot shared control system. In other words, if the current rollout episode exceeds  $l_{\max}$  timesteps without any failure warning, we recognize the episode as successful and the robot will automatically proceed to the next episode.

TABLE I: FLOAT hyperparameters.

Hyperparameter	Value
$N$	50
$\delta$	10
$\Delta\delta$	2.5

### B. Details of policy training

We train the policy on 4 NVIDIA A800 GPUs in parallel. Key hyperparameter choices are detailed in Table II. We also include the end-effector pose as proprioceptive observations. 6D rotation representation is utilized for its continuity in the space of 3D rotations  $SO(3)$  [55].

TABLE II: Policy training hyperparameters.

Batch Size	64
Learning Rate	1e-5 (DINOv2 encoder) 1e-4 (Others)
Training Epochs	500 (Initial) 300 (Fine-tuning)
Image Size	224*224
Optimizer	AdamW
Observation History Length	2
Action Chunk Length	8
DINOv2 linear head hidden layers	[768, 192, 64]

### C. Real-world evaluation settings

We carry out all the experiments in a  $80\text{cm} \times 80\text{cm}$  workspace. Besides, we randomize the initial position of the end-effector in a  $20\text{cm} \times 20\text{cm} \times 20\text{cm}$  space by adding a uniform noise perturbation with a scale of 0.1m. We also apply a uniform noise to the default quaternion of end-effector orientation with a scale of 0.1. In *Pour marbles into bowl*, the cup and the bowl are randomly placed in a  $40\text{cm} \times 80\text{cm}$  region respectively. In *Hang mug on holder*, the mug and the holder are placed in a  $20\text{cm} \times 40\text{cm}$  workspace respectively. The mug and the holder are randomly rotated by  $[0, \pi]$  and  $[0, \frac{\pi}{2}]$  respectively. In *Grasp mango and put it into drawer*,

the mango and the drawer are placed in a  $40\text{cm} \times 80\text{cm}$  area and rotated by  $[0, 2\pi]$  and  $[0, \frac{\pi}{2}]$  respectively. In *Fold towel*, the towel is placed in a  $60\text{cm} \times 60\text{cm}$  area and rotated by  $[0, 2\pi]$  randomly.

## REFERENCES

- [1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.
- [2] Christopher Agia, Rohan Sinha, Jingyun Yang, Zi-ang Cao, Rika Antonova, Marco Pavone, and Jeannette Bohg. Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress. *Conference on Robot Learning*, 2024.
- [3] Christopher Agia, Rohan Sinha, Jingyun Yang, Rika Antonova, Marco Pavone, Haruki Nishimura, Masha Itkina, and Jeannette Bohg. Cupid: Curating data your robot loves with influence functions. *arXiv preprint arXiv:2506.19121*, 2025.
- [4] Suneel Belkhale, Yuchen Cui, and Dorsa Sadigh. Data quality in imitation learning. *Advances in neural information processing systems*, 36:80375–80395, 2023.
- [5] Qingwei Ben, Feiyu Jia, Jia Zeng, Junting Dong, Dahua Lin, and Jiangmiao Pang. Homie: Humanoid loco-manipulation with isomorphic exoskeleton cockpit. *Robotics: Science and Systems*, 2025.
- [6] Annie S Chen, Alec M Lessing, Yuejiang Liu, and Chelsea Finn. Curating demonstrations using online experience. *Robotics: Science and Systems*, 2025.
- [7] Yuxin Chen, Devesh K Jha, Masayoshi Tomizuka, and Diego Romero. Fdpp: Fine-tune diffusion policy with human preference. *arXiv preprint arXiv:2501.08259*, 2025.
- [8] 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.
- [9] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *Robotics: Science and Systems*, 2024.
- [10] Robert Dadashi, Léonard Hussonot, Matthieu Geist, and Olivier Pietquin. Primal wasserstein imitation learning. *arXiv preprint arXiv:2006.04678*, 2020.
- [11] Hao-Shu Fang, Hongjie Fang, Zhenyu Tang, Jirong Liu, Chenxi Wang, Junbo Wang, Haoyi Zhu, and Cewu Lu. Rh20t: A comprehensive robotic dataset for learning diverse skills in one-shot. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 653–660. IEEE, 2024.
- [12] Hongjie Fang, Hao-Shu Fang, Yiming Wang, Jieji Ren, Jingjing Chen, Ruozhang Zhang, Weiming Wang, and Cewu Lu. Airexo: Low-cost exoskeletons for learning whole-arm manipulation in the wild. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15031–15038. IEEE, 2024.
- [13] Hongjie Fang, Chenxi Wang, Yiming Wang, Jingjing Chen, Shangning Xia, Jun Lv, Zihao He, Xiyan Yi, Yunhan Guo, Xinyu Zhan, et al. Airexo-2: Scaling up generalizable robotic imitation learning with low-cost exoskeletons. *arXiv preprint arXiv:2503.03081*, 2025.
- [14] Siddhant Haldar, Vaibhav Mathur, Denis Yarats, and Lerrel Pinto. Watch and match: Supercharging imitation with regularized optimal transport. In *Conference on Robot Learning*, pages 32–43. PMLR, 2023.
- [15] Joey Hejna, Chethan Bhateja, Yichen Jiang, Karl Pertsch, and Dorsa Sadigh. Re-mix: Optimizing data mixtures for large scale imitation learning. *Conference on Robot Learning*, 2024.
- [16] Joey Hejna, Suvir Mirchandani, Ashwin Balakrishna, Annie Xie, Ayzaan Wahid, Jonathan Tompson, Pannag Sanketi, Dhruv Shah, Coline Devin, and Dorsa Sadigh. Robot data curation with mutual information estimators. *Robotics: Science and Systems*, 2025.
- [17] Donald Joseph Hejna III and Dorsa Sadigh. Few-shot preference learning for human-in-the-loop rl. In *Conference on Robot Learning*, pages 2014–2025. PMLR, 2023.
- [18] Hamidreza Kasaei and Mohammadreza Kasaei. Vital: Visual teleoperation to enhance robot learning through human-in-the-loop corrections. *arXiv e-prints*, pages arXiv-2407, 2024.
- [19] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Sriram, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *Robotics: Science and Systems*, 2024.
- [20] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *Conference on Robot Learning*, 2024.
- [21] Sachit Kuhar, Shuo Cheng, Shivang Chopra, Matthew Bronars, and Danfei Xu. Learning to discern: Imitating heterogeneous human demonstrations with preference and representation learning. In *Conference on Robot Learning*, pages 1437–1449. PMLR, 2023.
- [22] Fanqi Lin, Yingdong Hu, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. *arXiv preprint arXiv:2410.18647*, 2024.
- [23] Huihan Liu, Soroush Nasiriany, Lance Zhang, Zhiyao Bao, and Yuke Zhu. Robot learning on the job: Human-in-the-loop autonomy and learning during deploy-

- ment. *The International Journal of Robotics Research*, page 02783649241273901, 2022.
- [24] Huihan Liu, Shivin Dass, Roberto Martín-Martín, and Yuke Zhu. Model-based runtime monitoring with interactive imitation learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4154–4161. IEEE, 2024.
- [25] Huihan Liu, Yu Zhang, Vaarij Betala, Evan Zhang, James Liu, Crystal Ding, and Yuke Zhu. Multi-task interactive robot fleet learning with visual world models. Conference on Robot Learning, 2024.
- [26] Weifeng Lu, Minghao Ye, Zewei Ye, Ruihan Tao, Shuo Yang, and Bo Zhao. Robofac: A comprehensive framework for robotic failure analysis and correction. *arXiv preprint arXiv:2505.12224*, 2025.
- [27] Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *arXiv preprint arXiv:2410.21845*, 2024.
- [28] Shengcheng Luo, Quanquan Peng, Jun Lv, Kaiwen Hong, Katherine Rose Driggs-Campbell, Cewu Lu, and Yong-Lu Li. Human-agent joint learning for efficient robot manipulation skill acquisition. *arXiv preprint arXiv:2407.00299*, 2024.
- [29] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.
- [30] Ajay Mandlekar, Caelan Reed Garrett, Danfei Xu, and Dieter Fox. Human-in-the-loop task and motion planning for imitation learning. In *Conference on Robot Learning*, pages 3030–3060. PMLR, 2023.
- [31] Mitsuhiro Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. *arXiv preprint arXiv:2410.13816*, 2024.
- [32] Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icmi*, volume 1, page 2, 2000.
- [33] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafrańiec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.
- [34] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.
- [35] Georgios Papagiannis and Yunpeng Li. Imitation learning with sinkhorn distances. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 116–131. Springer, 2022.
- [36] Vaibhav Saxena, Matthew Bronars, Nadun Ranawaka Arachchige, Kuancheng Wang, Woo Chul Shin, Soroush Nasiriany, Ajay Mandlekar, and Danfei Xu. What matters in learning from large-scale datasets for robot manipulation. *arXiv preprint arXiv:2506.13536*, 2025.
- [37] Rohan Sinha, Apoorva Sharma, Somrita Banerjee, Thomas Lew, Rachel Luo, Spencer M Richards, Yixiao Sun, Edward Schmerling, and Marco Pavone. A system-level view on out-of-distribution data in robotics. *arXiv preprint arXiv:2212.14020*, 2022.
- [38] Rohan Sinha, Amine Elhafsi, Christopher Agia, Matthew Foutter, Edward Schmerling, and Marco Pavone. Real-time anomaly detection and reactive planning with large language models. *Robotics: Science and Systems*, 2024.
- [39] Tony Tao, Mohan Kumar Srirama, Jason Jingzhou Liu, Kenneth Shaw, and Deepak Pathak. Dexwild: Dexterous human interactions for in-the-wild robot policies. *Robotics: Science and Systems*, 2025.
- [40] Octo Model Team, Dibya Ghosh, Homer Walk, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *Robotics: Science and Systems*, 2024.
- [41] Ran Tian, Yilin Wu, Chenfeng Xu, Masayoshi Tomizuka, Jitendra Malik, and Andrea Bajcsy. Maximizing alignment with minimal feedback: Efficiently learning rewards for visuomotor robot policy alignment. *arXiv preprint arXiv:2412.04835*, 2024.
- [42] Marcel Torne Villasevil, Balsells I Pamies, Zihan Wang, Samedh Desai, Tao Chen, Pulkit Agrawal, Abhishek Gupta, et al. Breadcrumbs to the goal: goal-conditioned exploration from human-in-the-loop feedback. *Advances in Neural Information Processing Systems*, 36:63222–63258, 2023.
- [43] Wenhao Wang, Jianheng Song, Chiming Liu, Jiayao Ma, Siyuan Feng, Jingyuan Wang, Yuxin Jiang, Kylin Chen, Sikang Zhan, Yi Wang, et al. Genie centurion: Accelerating scalable real-world robot training with human rewind-and-refine guidance. *arXiv preprint arXiv:2505.18793*, 2025.
- [44] Yanwei Wang, Lirui Wang, Yilun Du, Balakumar Sundaralingam, Xuning Yang, Yu-Wei Chao, Claudia Perez-D'Arpino, Dieter Fox, and Julie Shah. Inference-time policy steering through human interactions. *arXiv preprint arXiv:2411.16627*, 2024.
- [45] Josiah Wong, Albert Tung, Andrey Kurenkov, Ajay Mandlekar, Li Fei-Fei, Silvio Savarese, and Roberto Martín-Martín. Error-aware imitation learning from teleoperation data for mobile manipulation. In *Conference on Robot Learning*, pages 1367–1378. PMLR, 2022.
- [46] Philipp Wu, Yide Shentu, Qiayuan Liao, Ding Jin, Menglong Guo, Koushil Sreenath, Xingyu Lin, and Pieter Abbeel. Robocopilot: Human-in-the-loop interactive imitation learning for robot manipulation. *arXiv preprint arXiv:2503.07771*, 2025.
- [47] Yilin Wu, Ran Tian, Gokul Swamy, and Andrea Bajcsy. From foresight to forethought: Vlm-in-the-loop policy steering via latent alignment. *arXiv preprint arXiv:2502.01828*, 2025.
- [48] Chen Xu, Tony Khuong Nguyen, Emma Dixon, Christopher Rodriguez, Patrick Miller, Robert Lee, Paarth Shah, Rares Ambrus, Haruki Nishimura, and Masha Itkina. Can we detect failures without failure data? uncertainty-aware runtime failure detection for imitation learning policies. *Robotics: Science and Systems*, 2025.
- [49] Mengda Xu, Han Zhang, Yifan Hou, Zhenjia Xu, Linxi Fan, Manuela Veloso, and Shuran Song. Dexumi: Using human hand as the universal manipulation interface for dexterous manipulation. *arXiv preprint arXiv:2505.21864*, 2025.
- [50] Sizhe Yang, Wenye Yu, Jia Zeng, Jun Lv, Kerui Ren, Cewu Lu, Dahua Lin, and Jiangmiao Pang. Novel demonstration generation with gaussian splatting enables robust one-shot manipulation. *Robotics: Science and Systems*, 2025.
- [51] Takuma Yoneda, Luzhe Sun, Brady Stadie, Matthew Walter, et al. To the noise and back: Diffusion for shared autonomy. *Robotics: Science and Systems*, 2023.
- [52] Yu Zhang, Yuqi Xie, Huihan Liu, Rutav Shah, Michael Wan, Linxi Fan, and Yuke Zhu. Scizor: A self-supervised approach to data curation for large-scale imitation learning. *arXiv preprint arXiv:2505.22626*, 2025.
- [53] Zijian Zhang, Kaiyuan Zheng, Zhaorun Chen, Joel Jang, Yi Li, Siwei Han, Chaoqi Wang, Mingyu Ding, Dieter Fox, and Huaxiu Yao. Grape: Generalizing robot policy via preference alignment. *arXiv preprint arXiv:2411.19309*, 2024.
- [54] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *Robotics: Science and Systems*, 2023.
- [55] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5745–5753, 2019.