MAXIMIZING HUMAN EFFICIENCY IN LARGE-SCALE ROBOT POST-TRAINING VIA VLAC-CUT GUIDED PIPELINE

Shaopeng Zhai[*†], **Qi Zhang**[*], **Tianyi Zhang**[*], **Haoran Zhang**[*], **Fuxian Huang**[*]
Zijun Xu, Zhanhui Lin
Continuous Learning Team, Shanghai AI Lab
[*]Equal contribution. [†]Project Lead.

ABSTRACT

When adapting Vision Language Action (VLA) models to downstream tasks, multiple rounds of post training are required because a single round of data cannot resolve all issues, making continuous iterations necessary to progressively address the weaknesses exposed in previous rounds. In this report, we aim to maximize human efficiency during post-training, defined as the policy improvement and task throughput achieved per unit of human labor and time.

We propose a human-efficient post-training pipeline that enables a small number of human operators to supervise multiple robots. The pipeline is built around a specialized division of labor: a trained Teleoperator focuses on high-value remote interventions and recovery demonstrations, while a Floor Operator monitors multiple robots, triggers takeovers, and performs physical resets. This role specialization reduces task switching, lowers operator training costs, and allows limited human labor to supervise more robot interaction across a larger fleet. To improve data utilization efficiency, we introduce VLAC-CUT as an automatic rollout curation tool. It segments autonomous robot trajectories into progress-making, idle, failure-inducing, and recovery portions, preserving useful segments while filtering harmful or uninformative ones. The curated rollout data are combined with Human-in-the-Loop data for the next post-training round. We validate the proposed pipeline on four real-world manipulation tasks. Across iterative post-training rounds, the final policies achieve 80%–95% success rates and improve task throughput by 1.7×–4.2× over the base model. Under the same human-intervention budget, VLAC-CUT guided rollout reuse outperforms HITL-only training in both success rate and throughput.

1 INTRODUCTION

After pre-training, generalist policies like Vision Language Action (VLA) models require post training to achieve reliable performance on downstream tasks. This phase typically begins with collecting task specific data to fine tune the VLA. However, because human operators cannot foresee all potential edge cases during the initial data collection, the models fine tuned on this preliminary data inevitably exhibit flaws. Consequently, post training inherently involves multiple iterations. In each round, the policy is evaluated, and its specific failure modes are explicitly recorded to guide targeted data collection in the subsequent phase. This iterative process is crucial for progressively mitigating weaknesses and improving the actual task success rate. However, each iteration also requires humans to evaluate failures, intervene in difficult states, reset physical scenes, and collect or curate new training data. As these costs accumulate across repeated post-training rounds, human labor becomes a central bottleneck for scaling real-world VLA adaptation. Therefore, the key question is not only how to improve the policy, but how much policy improvement and task throughput can be obtained per unit of human labor and time.

Currently, real world post training paradigms generally fall into three categories. The first is real world Reinforcement Learning (RL), for example, []. This encompasses both fully online methods,

where the boundaries between training iterations are entirely blurred, and near online methods with clearly defined iterations, such as the approach used in pi0.6. These approaches require the model to autonomously collect data in the real world, generating a mixture of successful and failed rollouts. Unlike traditional supervised learning that relies purely on positive samples, RL methods can effectively extract useful training signals from these mixed trajectories. The second paradigm focuses on targeted human data collection, for example, []. Here, human operators conduct specific data collection to address the policy weaknesses identified in the previous round. This process yields a substantial volume of recovery data, helping the policy learn to recover from common errors. The third and most prevalent approach integrates the former two, for example, [***]. It begins with an offline dataset, and as the model generates autonomous real world rollouts, human operators can intervene at any moment to provide Human in the Loop (HITL) recovery data. These diverse data sources are subsequently combined, allowing the policy to be trained through a hybrid strategy of RL and Supervised Fine Tuning (SFT).

However, constrained by the limited exploration capabilities of current pre trained policies, VLAs often struggle to autonomously execute correct manipulation behaviors when facing unfamiliar corner cases. Consequently, Human in the Loop (HITL) intervention remains an absolute necessity. While some existing works attempt to learn a better policy from limited static datasets, our objective is fundamentally different. We do not aim to optimize sample efficiency on fixed data. Instead, we focus on maximizing how effectively limited human resources can be deployed to actively collect data that directly addresses policy weaknesses and resolves task bottlenecks. We define human efficiency as the post training policy performance gain, specifically in success rate and system throughput, achieved per unit of human labor and time. Therefore, the core challenge lies in optimizing human roles and pairing them with an appropriate system framework and a dedicated generalist model. By lowering operator training costs and ensuring singular task assignments, this model assisted paradigm enables humans to focus exclusively on the most critical tasks and the most urgently needed data.

To address this challenge, we propose a VLAC-Cut guided post-training pipeline that treats human labor as the primary bottleneck in real-world VLA adaptation. Unlike standard HITL pipelines that mainly use human interventions as additional correction demonstrations, our framework amplifies each unit of human effort by coupling targeted human takeover with large-scale autonomous rollout reuse. The system produces high-throughput real-world interaction through concurrent robot execution, while VLAC-Cut converts noisy robot rollouts into curated training segments that can be safely reused in the next post-training round.

Our contributions are threefold:

- **Human-efficiency-centered post-training formulation:** We identify human labor, rather than only robot interaction count or offline sample efficiency, as a central bottleneck in real-world VLA post-training. We therefore formulate human efficiency as the policy improvement, measured by success rate and task throughput, obtained per unit of human labor and time.
- **Large-scale HITL data collection pipeline:** We build a distributed multi-robot framework that enables a small team of operators to supervise many physical robots concurrently. By separating high-skill teleoperation from on-site monitoring and reset, the system reduces task switching, lowers operator training barriers, and allows human effort to focus on high-value takeover and recovery data.
- **VLAC-Cut guided rollout reuse:** We introduce VLAC-Cut as a process-level multimodal trajectory critic for post-training data curation. VLAC-Cut segments autonomous rollouts into progress-making, failure-inducing, idle, and recovery segments, allowing the pipeline to preserve useful robot-collected data while filtering behaviors that would otherwise teach the policy inefficient trial-and-error patterns.

We validate the proposed pipeline in real-world multi-robot post-training experiments. Under the same human-intervention budget, VLAC-Cut guided data curation provides additional rollout-derived training data and improves policy success rate and task throughput over standard HITL training. Overall, our work shifts the focus of robot post-training from raw sample efficiency to systemic human efficiency, establishing a practical framework for high-throughput, human-robot collaborative learning at scale.

2 RELATED WORK

Real-world post-training for robot policies has increasingly moved from static imitation learning toward iterative interaction, where autonomous rollouts expose policy weaknesses and subsequent data collection or optimization improves downstream performance. Real-world RL has demonstrated strong potential in locomotionSmith et al. (2024; 2023) and dexterous or general manipulationHu et al. (2023), but deploying such methods on physical robots remains expensive because each trial consumes real time, hardware lifetime, and human supervision. Many online or near-online systems therefore reuse previously collected trajectories through self-imitation, hindsight relabeling, or offline-to-online updatesKumar et al. (2024); Zhou et al. (2024a); Luo et al. (2024). For smaller manipulation policies, human-in-the-loop pipelines commonly begin with a small set of expert demonstrations and then combine robot interaction with human corrections to improve sample efficiencyKang et al. (2025); Zhou et al. (2024b); Li et al. (2025). Recent VLA-oriented post-training methods further benefit from pretrained multimodal priors, which increase the chance of discovering useful behaviors and make it possible to learn from mixed successful, failed, and recovered rolloutsChen et al. (2025b); Lv et al. (2025); Park et al. (2025). This trend has accelerated over the past year: $\pi^*_{0.6}$ studies how VLAs can improve from real-world deployment data that combine demonstrations, on-policy rollouts, and expert teleoperated interventionsPhysical Intelligence (2025); ROVE targets humanoid VLA post-training from imperfect human interventions and prioritizes high-value behaviors within mixed-quality trajectoriesXiao et al. (2026); SOP and Learning While Deploying scale online post-training to robot fleets that stream on-policy experience and human intervention signals to a centralized learnerPan et al. (2026); Wang et al. (2026). Complementary work reduces the cost of physical interaction through world-model or sim-real post-training, using virtual environments, simulated RL, or latent world-model scoring to improve VLAs under limited real-world dataXiao et al. (2025); Shi et al. (2026); Sun et al. (2026).

Despite this progress, real-world VLA post-training still depends critically on how raw interaction data are converted into reliable supervision. VLA policies differ substantially in their action interfaces: some produce discrete action tokens or structured textual actionsKim et al. (2024); Pertsch et al. (2025); Zhai et al. (2024), whereas others output continuous actions through diffusion or flow-based decodersBlack et al. (2024); Chi et al. (2023); Kim et al. (2025). This heterogeneity makes it difficult to apply a single token-centric RL recipe across models, and existing methods often resort to value-guided sample selection, behavior-cloning regularization, conservative value learning, or trajectory filtering to stabilize policy improvementHe et al. (2024); Lv et al. (2025); Park et al. (2025); Ding & Jin (2024); Physical Intelligence (2025); Wang et al. (2026). However, these strategies usually treat each rollout as a full episode or rely on coarse success/failure labels, while real robot trajectories often contain standard execution, failure-inducing detours, repeated trial-and-error, and useful recovery behaviors within the same episode. In this setting, blindly training on unedited rollouts can teach the policy to imitate avoidable failures, whereas discarding entire imperfect trajectories wastes valuable recovery data. VLAC-Cut addresses this data curation bottleneck directly: it performs process-level trajectory segmentation, removes failure-inducing portions, and preserves both correct execution and recovery segments. This trajectory-centric formulation is especially important for high-throughput multi-robot post-training, where manual inspection does not scale and human effort should be concentrated on high-value intervention rather than offline data cleaning.

Learned reward and progress models for embodied AI. A central challenge in embodied learning is replacing hand-crafted task rewards with scalable reward, value, or progress estimators defined over vision and language. Early approaches derived proxy measures of task advancement from human videos, trajectories, or goal-conditioned visual embeddings (e.g., VIP, LIV, R3M) (Sermanet et al., 2016; Shao et al., 2020; Chen et al., 2021; Ma et al., 2022; 2023a; Nair et al., 2022; Yang et al., 2023; Sontakke et al., 2024), but these are often task-specific and lack fine-grained language grounding. Subsequent foundation-model methods sought to shape or synthesize rewards directly from language and vision (Ma et al., 2023b; Rocamonde et al., 2023; Cui et al., 2025; Lin et al., 2024), framing reward estimation as binary success prediction, image comparison, temporal reordering, or relative progress scoring (Du et al., 2023; Wang et al., 2024; Venkataraman et al., 2024; Luu et al., 2025; Singh et al., 2025; Alakuijala et al., 2024; Ma et al., 2024; Chen et al., 2025a). While substantially improving scalability, these formulations highlight a persistent trade-off: success classifiers are often too coarse for dense shaping, and embedding distances struggle to capture complex intermediate progress (Du et al., 2023; Lynch et al., 2020; Andrychowicz et al., 2017; Escontrela

et al., 2023; Lee et al., 2021; Ma et al., 2024). To address this gap, recent work has shifted toward general-purpose, step-aware process reward modeling (PRM) to provide dense progress supervision. Methods such as VLAC, Robo-Dopamine, and Rewarding DINO extract dense progress deltas and step-aware signals to overcome the limitations of sparse or uniform reward designs (Zhai et al., 2025; Tan et al., 2025; Krack et al., 2026). Concurrently, VLM-based estimators like RoboReward, ProgressLM, and Large Reward Models explicitly frame progress tracking as a long-horizon reasoning or sequence modeling problem, providing discretized progress labels and contrastive signals for closed-loop refinement (Lee et al., 2026; Zhang et al., 2026; Wu et al., 2026). Further extending these paradigms, Robometer incorporates inter-trajectory preference learning to resolve ambiguities in suboptimal demonstrations (Liang et al., 2026), complementing a broader effort to leverage step decomposition, distance-to-goal estimation, and binary completion queries as robust interfaces for scalable reward supervision (Chen et al., 2025a; Ghasemipour et al., 2025; Yu et al., 2026; Mao et al., 2026; Intelligence et al., 2025; Chen et al., 2026).

Benchmarks for robot progress evaluations. Reward models have become increasingly important in modern foundation-model training, especially in post-training and reinforcement learning for large language models (Shao et al., 2024; DeepSeek-AI et al., 2025). Accordingly, benchmarks such as RewardBench, RewardBench 2, VLRewardBench, and Multimodal RewardBench have been introduced to evaluate reward models in language and general multimodal settings (Lambert et al., 2024; Malik et al., 2025; Li et al., 2024; Yasunaga et al., 2025). In embodied settings, however, benchmark construction remains limited and fragmented. OpenGVL evaluates temporal progress prediction from shuffled trajectory frames using a Value-Order Correlation metric, with a particular emphasis on automated data curation and filtering (Budzianowski et al., 2025). ManiRewardBench advances reward benchmarking on real-world manipulation with subtask-level temporal annotations, enabling evaluation of progress sensitivity, completion detection, and cross-embodiment robustness (Chen et al., 2026). RoboRewardBench broadens reward evaluation to diverse real-robot tasks and explicitly includes unsuccessful and near-miss trajectories, but is primarily framed around short-horizon, end-of-episode coarse progress rewards (Lee et al., 2026). Robometer further introduces held-out evaluation sets that probe reward alignment, trajectory ranking, and success–failure separation across unseen institutions, embodiments, camera viewpoints, and scenes (Liang et al., 2026). Adjacent to reward modeling, RoboFAC provides a failure-centric benchmark for task understanding, failure diagnosis, and hierarchical correction on simulated and real-world robot videos (Ye et al., 2025). Overall, existing embodied benchmarks capture complementary aspects of temporal progress, reward calibration, and failure understanding, yet still leave room for evaluating long-horizon, process-level reward and value estimation with richer temporal structure and finer-grained supervision on realistic robotic trajectories.

3 FRAMEWORK

3.1 SYSTEM ARCHITECTURE

To maximize human efficiency in scenarios where a small team of operators controls a large scale robotic fleet during real world robot post training, our system architecture is explicitly designed around the operational demands and collaborative workflows of the two specialized human roles.

First, the Teleoperator undergoes extensive specialized training and incurs high training costs. To maximize their productivity, the system must allow them to concentrate exclusively on high quality data collection without wasting time moving around or physically resetting scenes. This necessitates a seamless and stationary remote control interface. Second, the Floor Operator simultaneously manages multiple physical robots. Because they spend minimal time at any single robotic station while constantly observing the policy in action, they develop a profound understanding of the model capabilities and specific failure cases. Their cognitive effort must be concentrated entirely on issuing takeover signals and manually resetting scenes to setups where the policy frequently fails. Consequently, the architecture must provide dedicated and lightweight communication modules for the Floor Operator to instantly broadcast reset and takeover commands. Finally, to sustain this continuous collaboration, our framework must seamlessly interleave Human in the Loop (HITL) data with autonomous robot rollouts to support both online and offline training. This holistic pipeline is heavily supported by VLMs tasked with processing trajectory data, assessing task completion, and proactively predicting the necessity of takeovers. Driven by these human centric requirements, we

logically partition our distributed software architecture into three primary layers, as illustrated in Fig. 1.

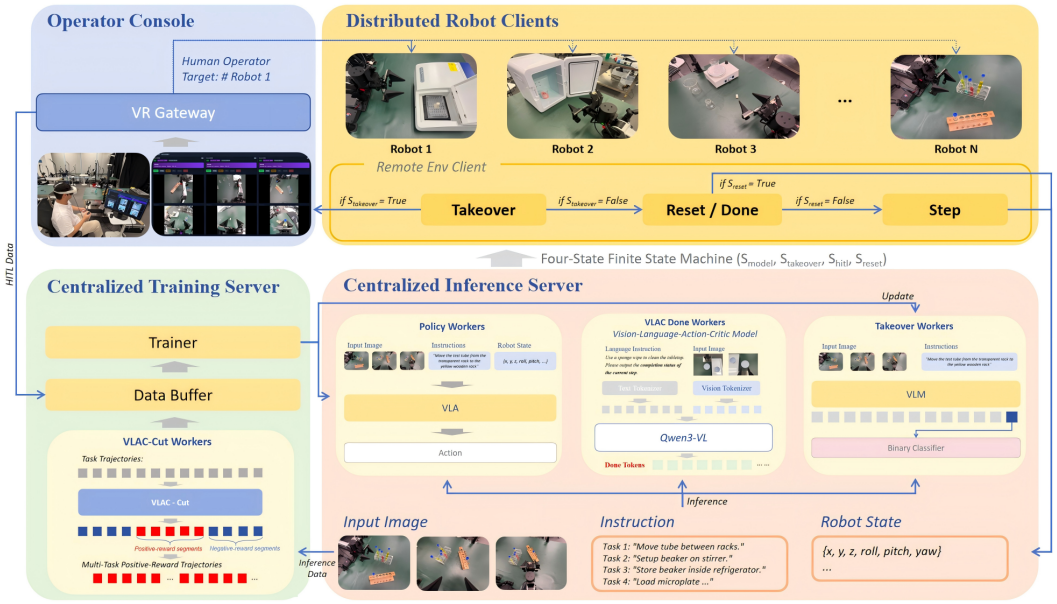

Figure 1: System architecture of the proposed distributed hierarchical HITL data collection framework.

- **Centralized GPU workers:** Deployed on a centralized server, these GPU workers manage parallel inference and continuous policy optimization using the Ray distributed framework. At each control step, an asynchronous coordination module aggregates multi modal observations from the distributed robotic fleet and routes them to parallel inference workers through dynamic load balancing. These workers execute the concurrent inference of the VLA policy π_θ, a predictive takeover model ϕ, and a task termination model ψ:

$$\mathbf{a}_t^i = \pi_\theta(s_t^i), \quad \gamma_t^i = \phi(s_t^i), \quad d_t^i = \psi(s_t^i) \tag{1}$$

  where $\mathbf{a}_t^i$ is the action chunk, $\gamma_t^i \in \{0, 1\}$ indicates the model prediction on whether a takeover is currently required at step t, and $d_t^i \in \{0, 1\}$ represents the model estimated task completion state. Following this parallel computation, the server bundles these outputs and directly transmits them back to the respective robotic client. The client then utilizes these flags to determine its execution state, where a takeover signal ($\gamma_t^i = 1$ or a manual override) or a reset signal ($d_t^i = 1$) will preempt the standard action execution. Once an episode concludes, the autonomous rollout trajectories are processed and filtered by the learned VLAC-Cut model. The preserved high quality data is then integrated into a centralized data service managed by a dedicated Ray worker. Concurrently, an asynchronous training pipeline samples from this data service to update θ and ϕ via gradient descent, asynchronously broadcasting the updated parameters to the inference workers to eliminate training induced latency.

- **Distributed Robot Clients:** Running on each onboard computer, the Robot Clients interface directly with the robot and manage real time data routing. On one hand, the client continuously captures robot proprioceptive data and camera images, streaming them to the Centralized GPU Workers for parallel inference, while simultaneously executing the predicted action chunks returned by the server. On the other hand, it serves as the direct communication endpoint, receiving and executing intervention signals routed through the server.

  To enable a flexible and timely takeover experience for the human operators, the client utilizes a dual process and asynchronous multi threaded design. This architecture explicitly isolates the hardware control loop from the overhead of network I/O and VR telemetry streams. By leveraging ZeroMQ for robust asynchronous command fetching and local inter process communication to manage incoming payloads, the system structurally decouples communication from robot execution.

Driven by these incoming streams, the client finite state machine transitions dynamically via a priority hierarchy:

$$\mathcal{S}^i_{\text{client}} \leftarrow \begin{cases} \mathcal{S}_{\text{takeover}} & \text{if } u^i_t = \text{takeover}, \\ \mathcal{S}_{\text{hitl}} & \text{if } \mathcal{S}^i_{\text{client}} = \mathcal{S}_{\text{takeover}} \text{ and VR streaming is active}, \\ \mathcal{S}_{\text{reset}} & \text{if } u^i_t = \text{reset}, \\ \mathcal{S}_{\text{model}} & \text{if } u^i_t = \text{step}(\mathbf{a}^i_t). \end{cases} \tag{2}$$

In the autonomous execution state $\mathcal{S}_{\text{model}}$, the client unrolls the received action chunks. When a takeover signal arrives from the Centralized GPU Workers, the client instantly purges stale actions and transitions through the standby state $\mathcal{S}_{\text{takeover}}$ into the active human control state $\mathcal{S}_{\text{hitl}}$ as soon as the VR stream synchronizes, guaranteeing a rapid and unimpeded handover for the Teleoperator. If a reset signal is issued, the client immediately aborts execution and notifies the Centralized GPU Workers to terminate the current episode recording, prompting the server to save and process the trajectory data. The client then enters the suspension state $\mathcal{S}_{\text{reset}}$, securely locking the robotic arm to allow the Floor Operator to safely perform the scene reset.

- **Operator Console:** Serving as the unified human interface, the Operator Console consists of a web frontend, a VR application, a hardware interface for the Floor Operator, and a centralized backend service. The Teleoperator utilizes the web frontend to monitor real time visual feeds from individual robots and to explicitly dictate the routing destination for the VR action commands. Once a target is assigned, the Teleoperator directly controls the robot to execute manipulation tasks through the VR application. Concurrently, the interface for the Floor Operator pairs with a portable Bluetooth keyboard, enabling them to rapidly force any specific robot into $\mathcal{S}_{\text{takeover}}$ while monitoring the active robots on site. Underpinning these interactive modules is the backend service which manages the critical communication middleware. For VR control, it employs a dynamic UDP routing gateway to seamlessly redirect high frequency telemetry streams to the designated robot. For event handling, it captures the Bluetooth keyboard inputs and leverages ZeroMQ sockets to instantly trigger $\mathcal{S}_{\text{takeover}}$ for the corresponding robot.

3.2 PIPELINE

In our established highly concurrent data collection pipeline, we structure the overall human workload into two distinct roles: the Teleoperator and the Floor operator. In a typical operational configuration, this strategic division allows just two personnel to effectively oversee six robotic arms. By ensuring that each person focuses exclusively on a single designated responsibility, our framework completely eliminates the cognitive overhead and efficiency degradation typically caused by task switching. Furthermore, isolating these duties drastically simplifies the onboarding process, making it highly efficient to conduct targeted, specialized training for each individual role independently.

- **Teleoperator:** The teleoperator dedicates their entire cognitive focus exclusively to remote manipulation. A proficient data collector must exhibit both optimal behavioral patterns and highly consistent operational habits, as these factors directly determine the quality of the collected data, which in turn dictates the efficiency of post training. For instance, in a test tube insertion task, the operator must deliberately reduce the manipulation speed during the high precision insertion phase and appropriately elevate the robotic arm to increase the clearance between the tube base and the target hole. Training a skilled teleoperator to internalize these specific manipulation patterns often requires weeks or even months. By keeping the teleoperator responsibilities strictly singular, we can concentrate our training efforts on these essential skills while significantly lowering their overall cognitive workload. During operation, the teleoperator remains stationary and uses a VR interface to monitor real time camera feeds. When a robot is identified for a takeover by either the scene floor operator or the predictive model, a notification is transmitted to the teleoperator. The system then seamlessly routes the corresponding camera feed to the VR interface, enabling the operator to quickly transmit action commands back to the robot. This remote operational paradigm completely eliminates the physical locomotion and time wasted walking between different robots. By comprehensively isolating this role from manual scene resetting and physical movement, the teleoperator achieves maximum data collection efficiency without any physical or cognitive interruptions.
- **Floor operator:** The Floor Operator is responsible for executing resets and continuously monitoring the policy execution on site, selectively forcing specific robots into $\mathcal{S}_{\text{takeover}}$. Because they

observe the autonomous rollouts over extended periods, the Floor Operator develops a highly intuitive and profound understanding of the policy behavior. They know exactly when the model is likely to fail, making them the ideal judge for determining the precise moment for a robot to enter the takeover state. Furthermore, their deep awareness of the policy weaknesses allows them to manually reset the environment to states where the policy frequently fails, thereby maximizing the collection of high value data in the exact scenarios that expose model limitations. In practice, their operational duties remain tightly focused. They actively monitor the active robots and use a portable Bluetooth keyboard to instantly force a specific robot into $\mathcal{S}_{\text{takeover}}$. When a robot halts and awaits a reset, they perform these targeted resets. By keeping their responsibilities strictly singular, the system empowers the Floor Operator to fully leverage their cognitive understanding of the policy behavioral characteristics. As the individual who most comprehensively observes the model in action, they play a critical role in creating the conditions to collect targeted data that addresses the policy shortcomings.

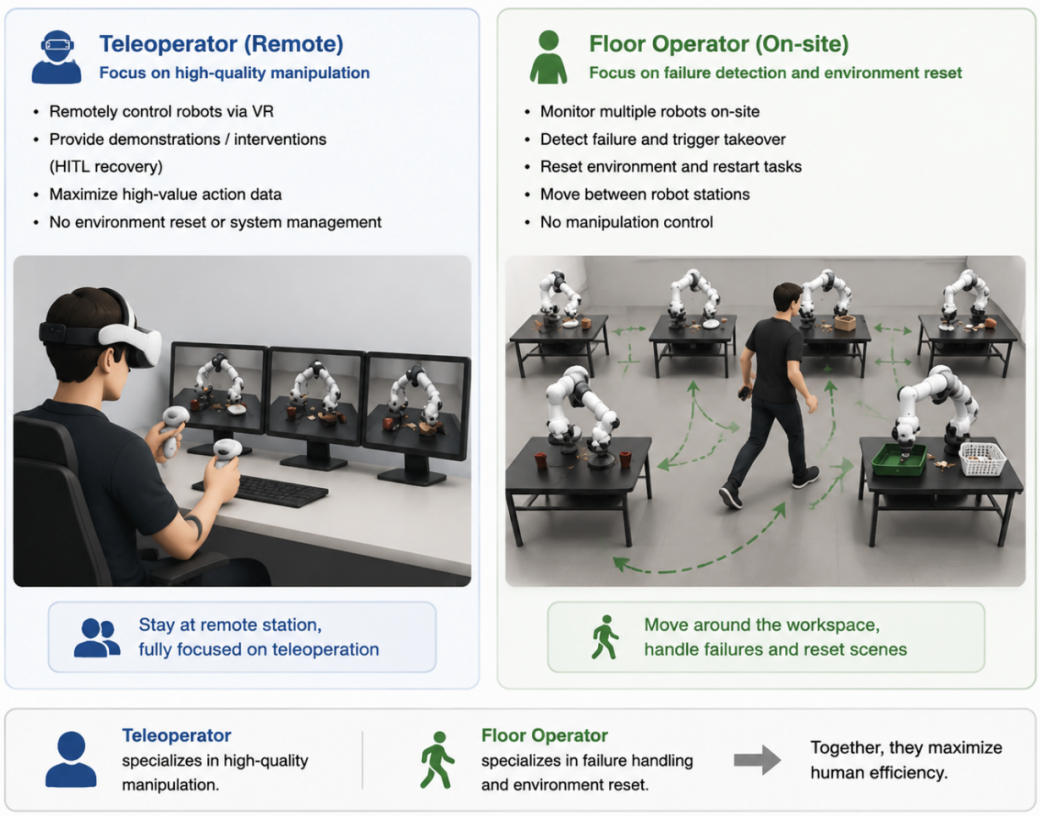

Figure 2: Role division between the Teleoperator and the Floor Operator in the proposed data collection pipeline.

3.3 Method

3.3.1 Data processing by VLAC-Cut

Under this paradigm, the system accumulates two distinct categories of interaction data. The first is rollout data generated entirely by the VLA policy. The second is human in the loop data, which is collected when either the predictive takeover model or the Floor Operator forces the robot into $\mathcal{S}_{\text{takeover}}$, with the subsequent actions executed by the Teleoperator. Consequently, a complete trajectory within our framework exhibits one of two structural configurations: a homogeneous trajectory composed exclusively of VLA rollouts, or a hybrid trajectory consisting of an initial VLA rollout prefix followed by a human in the loop recovery suffix. In addition to serving as recovery demonstrations for policy post-training, the manually triggered HITL events also provide supervision for

the takeover prediction model. Specifically, the observation at which the Floor Operator issues a takeover command is labeled as a positive takeover example, while observations sampled from normal autonomous rollouts are used as negative examples to train a 2B Qwen-VL based takeover predictor.

As the policy is continuously trained on the accumulated recovery data, it gradually develops the capability to recover from failure states. Consequently, the raw VLA rollouts generated during continuous data collection frequently contain repetitive trial and error, where the policy continuously generates errors and then recovers from them. While this continuous trial and error maintains a high task success rate, directly incorporating these unedited trajectories into the training set would teach the policy to replicate these repetitive actions, ultimately reducing its overall task execution throughput.

To prevent this issue, we employ the learned VLAC-Cut model to filter the VLA rollout trajectories prior to training. Trained specifically on failure modes and successful recovery sequences, VLAC-Cut partitions a rollout trajectory into standard execution segments, failure inducing segments, and recovery segments. Following this trajectory level decomposition, the failure inducing segments are purged. The extracted standard segments effectively reinforce the existing correct behaviors of the policy, while the recovery segments are combined with the human in the loop data to construct the refined training set for the subsequent optimization cycle.

3.3.2 ALGORITHM

Training a unified policy on data from a distributed multi robot fleet introduces inherent data imbalance and distribution shift challenges. First, a distribution mismatch exists between the standard rollout data and the human in the loop recovery data, because the action distribution required to correct a failure differs significantly from that of standard execution. Second, the physical heterogeneity across different environments, including variations in object layouts, camera viewpoints, and task definitions, induces cross environment interference. Consequently, optimizing the policy exclusively on newly collected continuous streams often degrades its performance on other environments and causes catastrophic forgetting of previously learned capabilities.To mitigate these distribution shifts and manage the continuous learning process, we formalize two distinct optimization strategies across training cycles:

- **Fully Online Incremental Optimization:** As data collection proceeds continuously, the server initiates an immediate incremental training update once the accumulated data reaches a specified threshold. To mitigate catastrophic forgetting and cross environment interference inherent to this streaming continual learning setup, we integrate the ConSFT algorithm. This approach establishes a self regulating learning dynamic that scales the optimization gradients based on per sample model confidence, eliminating the need for parallel reference networks or historical data buffers. The optimization objective is formulated as:

$$\mathcal{J}_{\mathrm{ConSFT}}(\theta) = \mathrm{sg}\left[\exp\left(-\frac{\mathcal{L}_{\mathrm{SFT}}(\theta)}{\tau}\right)\right] \cdot \mathcal{L}_{\mathrm{SFT}}(\theta) \tag{3}$$

  where $\tau > 0$ is a temperature parameter regulating the conservative scaling sensitivity, and $\mathrm{sg}[\cdot]$ denotes the stop gradient operator. During the initial phases of incremental adaptation, out of distribution behaviors, such as some recovery actions, generate large losses. This objective exponentially suppresses their gradient contributions to prevent disruptive parameter updates. As the policy gradually masters these novel behaviors, its confidence increases, causing the associated loss to decrease and consequently amplifying the update weights. This continuous feedback loop ensures a progressive and stable learning of new out of distribution behaviors while preserving previously learned capabilities.

- **Periodic Batched Optimization:** This configuration operates with discrete training intervals. Following each policy update, the system accumulates a new batch of interaction trajectories. Once a predefined dataset size is reached, this current batch is combined with the entire historical dataset from all preceding rounds for joint training. By aggregating and mixing historical data across all robots and task configurations, this approach resolves distribution shifts and mitigates catastrophic forgetting, enabling standard supervised fine tuning via flow matching.

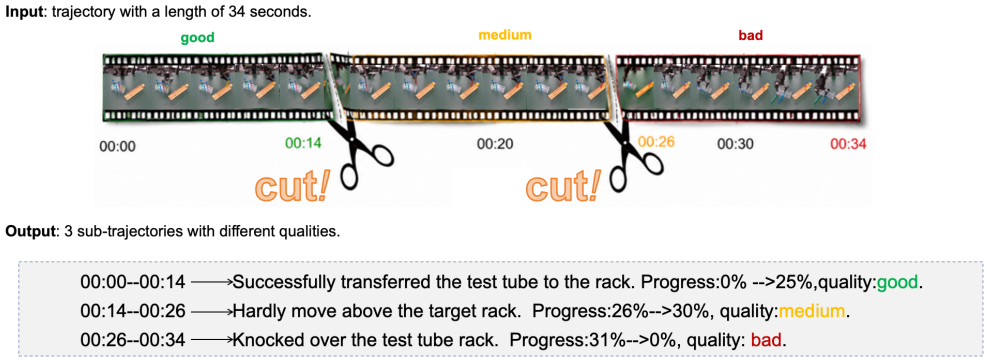

Figure 3: Overview of the VLAC-CUT rollout segmentation pipeline. Given a task-conditioned autonomous rollout trajectory, VLAC-CUT estimates process-level progress, diagnoses local state changes, and segments the trajectory into training-useful and training-harmful portions.

4 VLAC-CUT

4.1 VLAC-CUT AS A ROLLOUT SEGMENTATION CRITIC

The framework in Section 3 relies on large-scale autonomous rollout reuse to amplify limited human labor. However, a raw autonomous rollout trajectory is rarely uniformly useful for policy post-training. It may contain correct progress, idle motion, failure-inducing detours, repeated trial-and-error, and recovery behavior within the same episode. Training on the entire trajectory can therefore teach the policy to imitate avoidable failures, while discarding the entire trajectory wastes useful robot-collected progress and recovery data.

We introduce VLAC-CUT as a process-level multimodal trajectory critic for post-training data curation. Given a task instruction and a rollout trajectory, prefix, or candidate segment, VLAC-CUT estimates signed task progress, explains the visual and action-level causes of progress changes, and predicts cut points that separate segments with different training value. We use four segment types throughout the paper: *progress-making* segments correspond to standard execution that should reinforce correct behavior; *idle* segments contain little task-relevant progress; *failure-inducing* segments move the task state away from the goal or create avoidable errors; and *recovery* segments bring the system back from an error state toward task completion. The post-training data service keeps progress-making and recovery segments, removes failure-inducing and idle portions, and combines the preserved rollout-derived segments with HITL recovery data for the next optimization cycle.

This formulation differs from endpoint success filtering. End-state labels can identify whether an episode eventually succeeds, but they cannot distinguish an efficient execution from a trial-and-error success, nor can they identify which intermediate actions caused stagnation, regression, or recovery. VLAC-CUT instead treats task progress as a temporal process variable, making autonomous rollout reuse compatible with the human-efficiency goal of the overall pipeline.

4.2 PROCESS SUPERVISION FOR VLAC-CUT

Training such a critic requires supervision beyond demonstrations and final success labels. We therefore construct a Progress Annotation Dataset in which each annotated episode contains a task instruction, a task-level plan, sparse timestamped progress points, diagnostic language, and optional grounding/action annotations. The progress labels are signed: positive values indicate movement toward task completion, zero denotes the initial state, and negative values represent regressive states worse than the initial setup. This schema directly supervises the distinctions needed by VLAC-CUT: partial progress, stagnation, regression, failure, and recovery.

The dataset combines public robot sources with an in-house real-world ARX-data collection. Public datasets provide broad task and scene coverage, while ARX-data specifically increases coverage of physically executed non-monotonic behavior such as grasp failures, object drops, wrong-object interactions, inaccurate placements, re-grasping, and pose correction. All selected videos

are re-annotated using the same process-level schema; progress labels are not inherited from source datasets. The curated training/evaluation dataset contains 28,167 video-level records, 22,978 episodes, 15,206 task units, and 375,172 progress points. We reserve 3,515 records for held-out VPB evaluation and use the remaining training split to construct instruction-tuning data. Appendix A.1 provides the full source inventory, annotation protocol, split table, and dataset visualizations.

4.3 INSTRUCTION TUNING

We convert the training split into multimodal conversations that preserve the process structure of the annotations. Inputs may be a single image, an image pair, a sampled video clip, a rollout prefix, or optional robot/action context. Targets include progress values, state and action descriptions, progress explanations, success/failure analysis, correction plans, grounding labels, and relative action deltas.

The annotation-derived instruction data are organized around four abilities. **Task decomposition and grounding** connects language goals with semantic milestones, task-relevant objects, gripper positions, and keypoints. **Temporal progress prediction** teaches the model to estimate timestamp-aligned progress from task-conditioned visual evidence rather than elapsed time. **Diagnostic reasoning** turns scalar progress supervision into explanations of why a segment is progress-making, idle, failure-inducing, or recoverable. **In-context and action-conditioned supervision** prepares VLAC-CUT for rollout curation by conditioning on reference episodes and, when robot trajectories are available, predicting relative action deltas over progress-point or kinematic-keyframe intervals. VLAC-CUT remains a critic and data curation model rather than a closed-loop control policy.

We additionally use two targeted augmentations. Counterfactual reverse-progress augmentation constructs regression-and-recovery examples from pairs of annotated progress points, discouraging the shortcut that later frames are always more complete. Grounding-based rationale generation verbalizes geometric evidence from object boxes and gripper keypoints when it is consistent with the annotated progress direction. The supervised fine-tuning corpus is centered on this annotation-derived robot data and complemented with public multimodal, robotics, progress-reasoning, and spatial/video reasoning datasets. Implementation details and the full data mixture are provided in Appendix A.4.

4.4 THE VIDEO PROGRESS BENCHMARK

We introduce the Video Progress Benchmark (VPB) to evaluate whether a video-language model can recover task-conditioned progress on held-out robot trajectories. VPB uses the held-out split described above and reserves all progress annotations strictly for evaluation. The split is organized along two axes that match the intended use of VLAC-CUT: semantic task familiarity, separating seen and unseen task units, and progress-pattern type, separating expert-like trajectories from non-expert trajectories with regressive or recovery behavior. This design tests whether a model can estimate progress under both familiar and held-out task semantics, and whether it remains robust to non-monotonic executions such as failed grasps, wrong-object interactions, object drops, and recovery attempts.

VPB evaluates progress understanding at three complementary scopes. **Global progress metrics** measure whether the model recovers the continuous progress trajectory, using Progress Rank Correlation (PRC), Value-Order Correlation (VOC) on expert trajectories, and Mean Absolute Error (MAE). **Terminal-state metrics** evaluate whether the model recognizes whether the final state reaches near-completion, using terminal-state accuracy and macro F1. **Local direction metrics** evaluate whether adjacent annotated keyframes are correctly classified as improvement, stagnation, or regression. Appendix A.5 gives the formal task definition, interpolation rule, and metric equations.

5 EXPERIMENTS

This section first evaluates VLAC-CUT as a process-level rollout segmentation critic on VPB, and then evaluates the full VLAC-Cut guided post-training pipeline in real-world robot tasks.

Method	Overall		Expert Seen			Expert Unseen			Non-Expert Seen		Non-Expert Unseen	
	MAE ↓	PRC ↑	MAE ↓	PRC ↑	VOC ↑	MAE ↓	PRC ↑	VOC ↑	MAE ↓	PRC ↑	MAE ↓	PRC ↑
VLAC-CUT	**7.7177**	**0.9192**	**6.3122**	**0.9902**	**0.9910**	**6.3170**	**0.9858**	**0.9865**	**9.7914**	**0.8124**	**9.7403**	**0.8251**
ProgressLM-RL	30.4390	0.2004	29.9053	0.2096	0.2092	29.9537	0.2103	0.2094	30.7117	0.1978	31.6518	0.1754
Robometer	21.4711	0.6434	20.2519	0.7427	0.7422	21.0415	0.7075	0.7067	22.8503	0.5241	22.4994	0.5238
RoboReward	21.0776	0.7316	20.8477	0.8342	0.8338	22.4818	0.7961	0.7956	20.1678	0.6053	20.2730	0.6071
Robo-Dopamine	28.9723	0.5370	26.6553	0.6341	0.6342	28.7088	0.5922	0.5923	29.9523	0.4334	31.7554	0.4182
TOPReward	32.6231	0.2549	31.5370	0.2710	0.2701	31.1984	0.2755	0.2750	34.4395	0.2447	34.4720	0.2117
GVL-GPT-5.5	25.1813	0.4028	24.3283	0.4363	0.4356	24.4238	0.4314	0.4309	26.6008	0.3409	26.1135	0.3735
GVL-Gemini-3.1-Pro	25.2983	0.4602	22.5425	0.5215	0.5208	23.6559	0.4879	0.4863	28.6759	0.3886	28.3418	0.4012
GVL-Gemini-3.5-Flash	22.8257	0.5367	19.7337	0.6091	0.6082	21.3340	0.5570	0.5558	25.9830	0.4759	26.3585	0.4619
Chrono-GVL-GPT-5.5	12.4817	0.8760	10.1806	0.9770	0.9770	10.8201	0.9789	0.9793	15.5005	0.7192	15.2481	0.7338
Chrono-GVL-Gemini-3.1-Pro	12.8464	0.8991	8.9649	0.9818	0.9818	9.4779	0.9869	0.9868	17.9769	0.7618	18.2982	0.7814
Chrono-GVL-Gemini-3.5-Flash	12.8448	0.8955	8.9995	0.9868	0.9872	9.2861	0.9849	0.9852	18.1443	0.7625	18.3528	0.7625

Table 1: Global-level VPB results on the overall split and four evaluation buckets. Lower MAE is better; higher PRC and VOC are better. VOC is reported only for expert buckets, following the benchmark definition.

5.1 VLAC-CUT EVALUATION ON VPB

5.1.1 EXPERIMENTAL SETUP

We evaluate VLAC-CUT on the formal VPB evaluation split described in Section 4.4, which contains 3,515 held-out robot trajectory records and 49,501 annotated progress points. All methods are evaluated under the same official VPB protocol, using the three metric families defined in Section 4.4: global progress prediction, terminal-state recognition, and local progress direction.

We compare VLAC-CUT against representative progress- and reward-estimation baselines for robotic video understanding. ProgressLM-RL is included as a single-observation progress-reasoning baseline: each VPB query point is paired with a task demonstration and evaluated as an image-text progress estimation problem, after which the pointwise predictions are assembled into trajectory progress curves. Robometer and RoboReward represent video-based robotic reward models. For VPB, they are evaluated on trajectory prefixes so that each prefix yields a current progress estimate, which is then aligned with the official annotation points. Robo-Dopamine is included as a process reward modeling baseline based on relative progress estimation between earlier and later states; its pairwise progress predictions are converted into the same signed progress representation used by VPB. TOPReward is used as a zero-shot token-probability reward baseline: we query each trajectory prefix and convert its completion evidence into a normalized progress curve for evaluation. We also evaluate Generative Value Learning (GVL) style baselines, which use a VLM as an in-context progress estimator over selected keyframes. Since GVL is a prompting framework rather than a single fixed model, we instantiate it with three VLM backbones: GPT-5.5, Gemini-3.1-Pro, and Gemini-3.5-Flash. We report two variants for each backbone. The first, denoted GVL, follows the original shuffled-frame formulation, where keyframes are presented in a randomized order and the model predicts their task-completion percentages. The second, denoted Chrono-GVL, uses nearly identical keyframes and prompt structure but presents frames in chronological order. In both cases, predictions on selected keyframes are interpolated back to the full VPB timeline before computing metrics. This design lets us compare the original temporal-ordering formulation of GVL with a chronological variant while keeping the underlying VLM backbone explicit.

5.1.2 EVALUATION RESULTS

Global-Level Progress Prediction Table 1 reports global progress prediction results on the full VPB split and on the four evaluation buckets. VLAC-CUT achieves the best MAE and PRC on the overall split and in every bucket. Overall, VLAC-CUT obtains an MAE of 7.7177 and a PRC of 0.9192. The strongest non-VLAC-CUT method under MAE is Chrono-GVL-GPT-5.5, with 12.4817, while the strongest non-VLAC-CUTmethod under PRC is Chrono-GVL-Gemini-3.1-Pro, with 0.8991. Thus, the gains are not confined to either value calibration or temporal ordering: VLAC-CUT more accurately recovers both the absolute signed progress values and the shape of the progress trajectory.

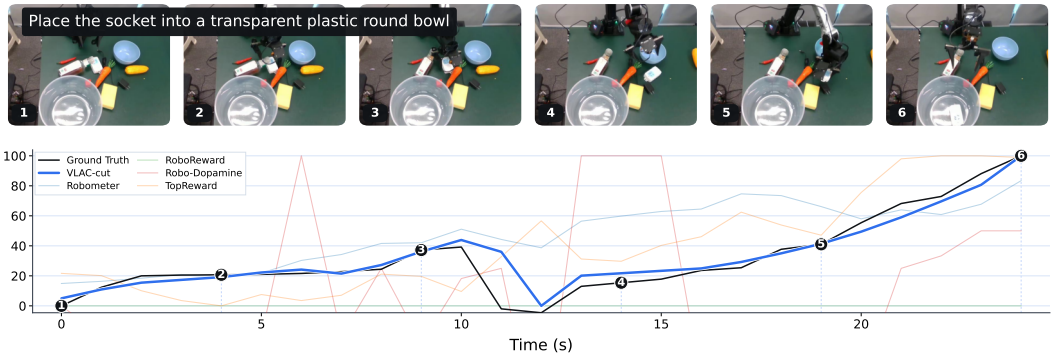

Figure 4: Qualitative comparison of signed progress prediction on a representative VPB trajectory. The top row shows sampled video frames, and the numbered markers link each frame to the corresponding timestamp on the progress curves. The ground-truth trajectory contains an early regression followed by gradual recovery and final task completion.

Method	Overall				Seen				Unseen			
	TSA ↑	$F1_S$ ↑	$F1_F$ ↑	$MacroF1_T$ ↑	TSA ↑	$F1_S$ ↑	$F1_F$ ↑	$MacroF1_T$ ↑	TSA ↑	$F1_S$ ↑	$F1_F$ ↑	$MacroF1_T$ ↑
VLAC-CUT	**90.84**	**94.72**	**65.23**	**79.98**	**91.97**	**95.39**	**68.87**	**82.13**	**89.71**	**94.06**	**61.73**	**77.89**
ProgressLM-RL	18.44	4.40	28.88	16.64	17.26	4.47	27.02	15.75	19.61	4.33	30.69	17.51
Robometer	26.23	21.26	30.61	25.94	26.37	23.08	29.38	26.23	26.09	19.35	31.79	25.57
RoboReward	41.93	47.65	34.81	41.23	42.71	49.50	33.82	41.66	41.16	45.73	35.75	40.74
Robo-Dopamine	46.77	56.32	31.89	44.10	46.98	57.12	30.57	43.85	46.56	55.49	33.14	44.32
TOPReward	60.20	71.54	33.85	52.69	61.22	72.57	33.82	53.20	59.18	70.48	33.89	52.18
GVL-GPT-5.5	52.06	62.50	33.58	48.04	51.88	62.43	33.10	47.76	52.25	62.57	34.07	48.32
GVL-Gemini-3.1-Pro	51.49	62.67	30.78	46.72	52.51	63.93	30.50	47.21	50.48	61.37	31.04	46.21
GVL-Gemini-3.5-Flash	56.10	67.79	31.09	49.44	57.29	69.06	31.07	50.06	54.92	66.50	31.10	48.80
Chrono-GVL-GPT-5.5	67.77	77.83	40.96	59.40	69.76	79.54	42.09	60.82	65.78	76.07	39.92	58.00
Chrono-GVL-Gemini-3.1-Pro	86.32	92.09	49.42	70.75	87.98	93.08	54.23	73.66	84.65	91.08	44.90	67.99
Chrono-GVL-Gemini-3.5-Flash	86.80	92.54	42.43	67.49	88.84	93.72	50.00	71.86	84.76	91.37	35.27	63.32

Table 2: Terminal-state recognition results on VPB. Seen and unseen denote the merged task-familiarity partitions, each computed before metric aggregation. All values are reported in percent (%).

The expert buckets isolate progress estimation under trajectories that are expected to be chronologically ordered. In this setting, VLAC-CUT remains highly consistent across task familiarity: it obtains MAE 6.3122, PRC 0.9902, and VOC 0.9910 on expert seen tasks, and MAE 6.3170, PRC 0.9858, and VOC 0.9865 on expert unseen tasks. The Chrono-GVL variants also preserve strong expert ordering, but their MAE remains substantially higher, indicating that temporal ordering alone is insufficient for calibrated signed-progress prediction. The non-expert buckets are more challenging because trajectories may stagnate, regress, or recover after failed intermediate actions. VLAC-CUT obtains MAE 9.7914 and PRC 0.8124 on non-expert seen tasks, and MAE 9.7403 and PRC 0.8251 on non-expert unseen tasks. These results show that the model's progress estimates remain robust under both held-out task semantics and non-monotonic execution patterns. Figure 4 illustrates the same behavior qualitatively: VLAC-CUT follows the annotated regression and recovery pattern, whereas several baselines either produce unstable estimates or fail to capture the nuances of the regressive dynamics.

Terminal-State Recognition Table 2 reports terminal-state recognition on the full VPB split and on the merged seen and unseen partitions. Although VPB evaluates dense progress rather than endpoint success alone, terminal recognition is an important diagnostic of whether a model's final progress estimate is calibrated to task completion. VLAC-CUT achieves the strongest performance across all reported terminal metrics.

On the overall split, VLAC-CUT reaches 90.84% TSA, 94.72% $F1_S$, 65.23% $F1_F$, and 79.98% $MacroF1_T$. The strongest non-VLAC-CUT TSA is 86.80% from Chrono-GVL-Gemini-3.5-Flash, while the strongest non-VLAC-CUT macro F1 is 70.75% from Chrono-GVL-Gemini-3.1-Pro. The gap is especially clear for the failed or incomplete terminal class: VLAC-CUT improves $F1_F$ from

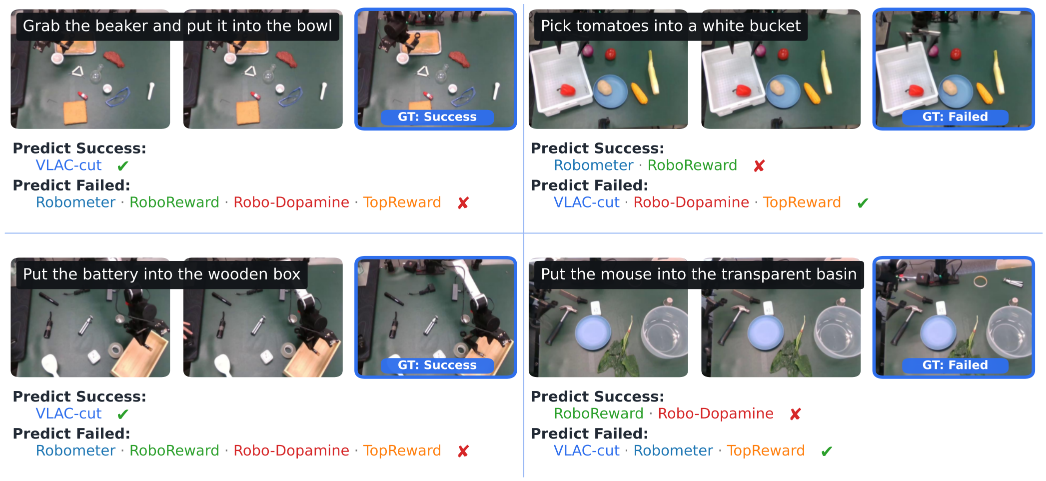

Figure 5: Qualitative examples of terminal-state recognition on VPB trajectories. Each panel shows a task instruction, sampled visual observations, and the final frame annotated with the ground-truth terminal outcome. The predicted success and failure groups indicate how different reward and progress models classify the final state. VLAC-CUT correctly recognizes both completed and failed executions in these examples, while several baselines confuse visually plausible but incomplete states with success, or fail to recognize completed terminal states.

Method	Overall					Seen					Unseen				
	$\text{Acc}_D \uparrow$	$\text{F1}_+ \uparrow$	$\text{F1}_0 \uparrow$	$\text{F1}_- \uparrow$	$\text{MacroF1}_D \uparrow$	$\text{Acc}_D \uparrow$	$\text{F1}_+ \uparrow$	$\text{F1}_0 \uparrow$	$\text{F1}_- \uparrow$	$\text{MacroF1}_D \uparrow$	$\text{Acc}_D \uparrow$	$\text{F1}_+ \uparrow$	$\text{F1}_0 \uparrow$	$\text{F1}_- \uparrow$	$\text{MacroF1}_D \uparrow$
VLAC-CUT	**76.07**	**54.19**	**84.21**	**37.77**	**58.72**	**75.80**	**53.07**	**84.06**	**39.75**	**58.96**	**76.33**	**55.23**	**84.36**	**35.92**	**58.50**
ProgressLM-RL	49.33	24.87	64.97	6.33	32.06	49.98	24.45	65.56	6.75	32.25	48.71	25.25	64.41	5.95	31.87
Robometer	68.50	29.43	80.41	8.16	39.33	68.64	28.73	80.52	8.60	39.28	68.36	30.09	80.29	7.77	39.38
RoboReward	66.86	31.94	78.77	7.00	39.24	66.68	31.99	78.61	7.26	39.29	67.03	31.90	78.92	6.76	39.20
Robo-Dopamine	53.11	30.11	67.36	6.00	34.49	52.86	29.70	67.04	5.76	34.16	53.35	30.51	67.67	6.23	34.80
TOPReward	49.15	29.09	64.14	6.25	33.16	48.45	28.44	63.45	6.11	32.67	49.82	29.73	64.79	6.38	33.63
GVL-GPT-5.5	51.79	31.54	66.08	6.41	34.68	51.35	30.81	65.59	6.93	34.44	52.21	32.24	66.55	5.92	34.90
GVL-Gemini-3.1-Pro	44.27	31.05	57.52	7.33	31.97	43.70	30.43	56.85	7.75	31.68	44.82	31.65	58.16	6.92	32.24
GVL-Gemini-3.5-Flash	48.15	32.79	61.33	8.03	34.05	47.83	32.40	60.85	9.05	34.10	48.45	33.17	61.80	7.08	34.02
Chrono-GVL-GPT-5.5	71.15	38.78	81.54	5.08	41.80	70.48	37.96	81.03	6.86	41.95	71.78	39.59	82.03	3.40	41.67
Chrono-GVL-Gemini-3.1-Pro	66.50	36.63	77.74	10.43	41.60	65.72	35.99	77.13	10.58	41.23	67.24	37.25	78.32	10.29	41.95
Chrono-GVL-Gemini-3.5-Flash	64.83	36.08	76.24	15.29	42.54	63.80	34.73	75.46	15.58	41.92	65.82	37.40	76.98	15.01	43.13

Table 3: Local progress direction results on official annotation-point transitions. F1_+, F1_0, and F1_- are one-vs-rest F1 scores for improvement, stagnation, and regression, respectively. All values are reported in percent (%).

the best non-VLAC-CUT value of 49.42% to 65.23%. This indicates that dense signed progress estimation improves final-state discrimination beyond recognizing visually plausible successful endings.

The seen and unseen partitions show the same trend. VLAC-CUT obtains $\text{MacroF1}_T = 82.13\%$ on seen tasks and 77.89% on unseen tasks, remaining ahead of all baselines in both partitions. Figure 5 provides qualitative examples of this behavior: VLAC-CUT identifies completed executions while also rejecting incomplete terminal states that can appear visually close to success.

Local Progress Direction Table 3 reports local progress direction recognition under the official annotation-point protocol. Each adjacent keyframe transition is categorized as positive, neutral, or negative using the VPB threshold $\tau = 10$, and predictions are evaluated with transition accuracy and class-wise F1 scores. This metric directly tests whether a method can distinguish genuine improvement from stagnation and regression.

VLAC-CUT achieves the best local direction performance across all overall metrics, with $\text{Acc}_D = 76.07\%$ and $\text{MacroF1}_D = 58.72\%$. The strongest non-VLAC-CUT accuracy is 71.15% from Chrono-GVL-GPT-5.5, while the strongest non-VLAC-CUT macro F1 is 42.54% from Chrono-GVL-Gemini-3.5-Flash. The largest difference appears on the regression class: VLAC-CUT obtains $\text{F1}_- = 37.77\%$, compared with 15.29% for the strongest non-VLAC-CUT result. This shows that

VLAC-CUT is not only more accurate on frequent neutral transitions, but also substantially more sensitive to regressive state changes.

The seen and unseen results further indicate that the local progress signal generalizes across task semantics. VLAC-CUT obtains $\mathrm{MacroF1}_D = 58.96\%$ on seen tasks and 58.50% on unseen tasks, with nearly identical direction accuracy. This consistency suggests that the model's ability to distinguish improvement, stagnation, and regression is not limited to familiar task units, but transfers to held-out task semantics and non-monotonic execution patterns.

5.2 REAL-WORLD POST-TRAINING EXPERIMENTS

5.2.1 EVALUATION TASKS

To evaluate policy performance, we conducted a comprehensive evaluation across four diverse real-world tasks: **Refrigerator**, **Microplate**, **Test Tube**, and **Stirrer**. We summarize the tasks below, with progress illustrations in Figure 6:

Refrigerator: In this task, the robot is required to open the refrigerator door, grasp a reagent-filled beaker with the gripper, place it stably inside the refrigerator, and then close the door. For quantitative evaluation, we count a trial as successful only when the robot completes the full task within 100 seconds without dropping the beaker or spilling any liquid during transfer. Although the policy may still recover by picking up the beaker after a drop, such cases are counted as failures to ensure consistent evaluation and reflect practical use. We use the following prompt as the instruction input to the VLA models for this task: *Open the refrigerator door, grasp the beaker with the gripper, place it inside the refrigerator, and then close the door.*

Microplate: This task requires the robot to open the lid of the microplate reader, pick up a microplate from an arbitrary position on the table, transfer it into the reader, and close the lid. To be counted as successful, the robot must finish the full sequence within 100 seconds without dropping the microplate during transfer. Prompt: *Turn on the microplate reader, place the microplate inside, then close the microplate reader.*

Test Tube: We evaluate our policies on the challenging long-horizon task of transferring four reagent-filled test tubes from a transparent plastic rack to a yellow wooden rack. Since the policy may succeed after multiple attempts during test-tube insertion, we set the time limit to 200 seconds for quantitative evaluation. Dropping any test tube during the transfer is also counted as a failure. Prompt: *Move the test tube from the transparent rack to the yellow wooden rack*

Stirrer: This task involves grasping a magnetic stir bar from an arbitrary position on the table, placing it into a beaker containing liquid, and then transferring the beaker onto a magnetic stirrer. A trial is considered successful only if the full task is completed within 60 seconds and no liquid is spilled during the transfer. Prompt: *Grasp the magnetic stir bar from the desktop, place it into the beaker, and then put the beaker onto the magnetic stirrer.*

5.2.2 QUANTITATIVE RESULTS

We use three metrics to evaluate the performance of models across different tasks: throughput, success rate, and failure progress. Throughput is defined as the number of tasks successfully completed per hour, measuring both execution efficiency and task completion capability. Success rate measures the proportion of episodes in which the policy successfully completes the task. We further evaluate failure progress, defined as the proportion of the task successfully completed before failure. This metric reflects a policy's potential to complete the task as well as its robustness, providing additional performance information beyond the binary success/failure outcome of each episode.

Policy improvement across post-training iterations. First, we describe how the VLAC-Cut guided pipeline improves policies through multiple iterations of data collection and training. As described in Section 3, each iteration uses the best-performing model from the previous iteration to collect two types of data: HITL recovery data and autonomous rollout trajectories. The autonomous rollout trajectories are filtered by VLAC-Cut to extract curated rollout data for training. We then construct a new training dataset by combining the curated rollout data, the HITL recovery data, and the data used to train the base model, approximately balancing them at a 1:1:1 ratio to achieve strong

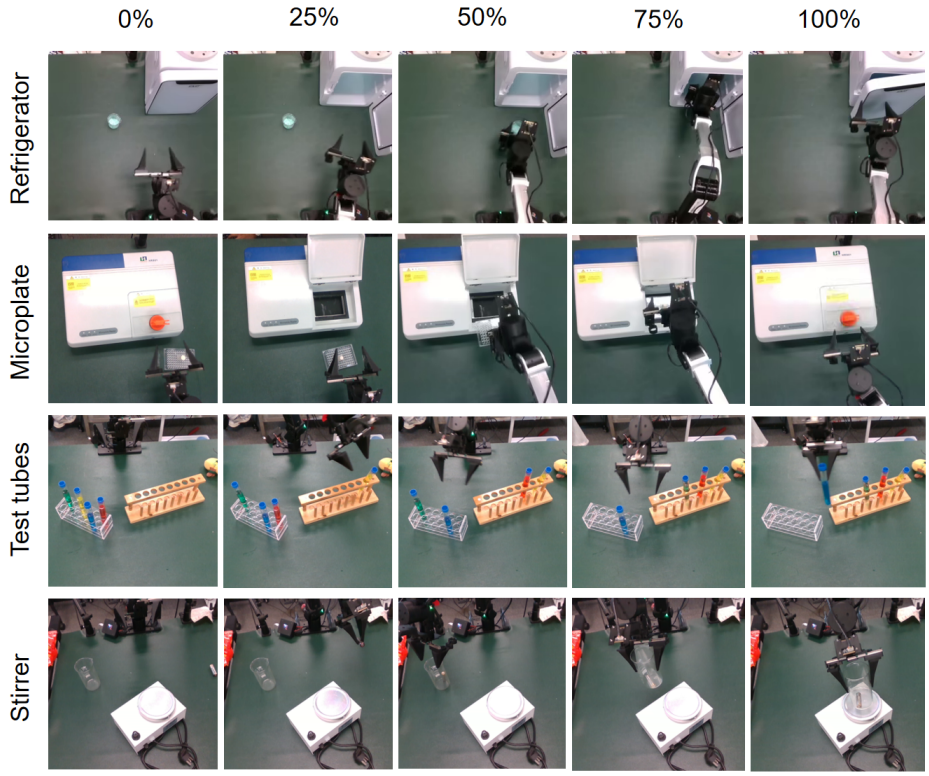

Figure 6: **Task execution progress.** Each row shows execution states for tasks, and each column corresponds to a normalized task-completion progress from 0% to 100%.

model performance. Finally, we train the current inference model on this combined dataset to obtain an updated policy, and the best-performing checkpoint is used as the inference model for the next round of data collection and training.

For the Refrigerator, Microplate, and Test Tube tasks, we conducted two iterations of data collection and training to improve the policy. In contrast, the Stirrer task was relatively simple, with the policy reaching a 90% success rate after a single iteration; therefore, no additional iteration was performed for this task.

Table 4 summarizes the policy evaluation results. After multiple iterations of training, the VLAC-Cut guided pipeline achieves substantial improvements in both throughput and success rate across the four evaluation tasks. Compared with the base model, the final policy improves these metrics by more than two times in most cases and by nearly three times in some tasks. In the final iteration, the pipeline achieves a success rate above 90% in all tasks, except for refrigerator opening. The success rate of the refrigerator-opening task saturates at 80%, likely because further improvement is constrained by the camera viewpoint and the mechanical structure. These results demonstrate that the VLAC-Cut guided pipeline can effectively improve policy performance.

A closer analysis of the same results indicates that the throughput improvement is primarily driven by the increase in success rate, while the execution time first increases and then decreases over the two optimization iterations. For the three tasks with a second round of optimization, the first iteration substantially improves the success rate over the base model, but it also introduces a noticeable increase in execution time. This is mainly because the added HITL recovery data teaches the policy to recover from states that previously led to failures. When the policy enters these error-prone states, it often performs additional recovery actions before returning to the correct execution trajectory. As a result, the initial gain in success rate is achieved largely by correcting erroneous steps, at the cost of longer execution time.

However, in the final iteration, throughput improves more substantially than in the first iteration. For the refrigerator task, throughput increases from 10 to 19 after the first iteration and further rises to

42 after the second iteration, corresponding to gains of 90% and 120%. Similar trends are observed in the Microplate task, where throughput increases from 13 to 18 and then to 42, with gains of 38% and 133%, respectively. In the Test Tube task, throughput increases from 14 to 20 and then to 37, corresponding to gains of 43% and 83%. This larger second-iteration improvement occurs because the stronger policy produces more successful autonomous rollout trajectories, which provide higher-quality data for the next round of training. Training on the curated rollout data enables the policy to execute correct actions more directly, rather than frequently entering error-prone states and relying on recovery behaviors as in the first iteration. Consequently, both execution time and success rate improve, leading to a larger throughput gain in the second iteration.

The quantitative definition of task progress for each task is illustrated in Figure 6. The trend in failure progress provides further evidence of improved policy robustness, as the policy is able to complete a larger portion of each task even when it eventually fails. In the final Microplate iteration, the remaining failures are mainly concentrated at the most difficult step, opening the microplate reader lid. Once this step is completed, the subsequent actions are typically successful. This bottleneck explains the relatively low average failure progress of 20%, since failed trials tend to terminate at the same early stage.

Overall, these results show that the VLAC-Cut guided pipeline substantially improves both success rate and throughput across most tasks, demonstrating its effectiveness in improving policy performance through iterative data collection and training.

Human-efficiency comparison with HITL. We compare the VLAC-Cut guided pipeline with a multi-iteration HITL-only training approach under the same human-labor budget. Both methods use the same amount of HITL recovery data, while the VLAC-Cut guided pipeline additionally records autonomous rollout trajectories during the correction process. The data used in each iteration are summarized in Table 5. To ensure a fair comparison, both methods start each training iteration from the same checkpoint. For example, the initial model for the second iteration is selected as the best-performing checkpoint from the first iteration; in our experiments, this is the best checkpoint from the first VLAC-Cut guided iteration. As shown in Table 4, the VLAC-Cut guided pipeline achieves higher success rate and throughput than HITL-only training in each training iteration. In the refrigerator task, for instance, throughput increases from 10 to 14 with HITL-only training and from 10 to 19 with the VLAC-Cut guided pipeline in the first iteration, corresponding to gains of 40% and 90%, respectively. In the second iteration, throughput increases from 19 to 33 with HITL-only training and from 19 to 42 with the VLAC-Cut guided pipeline, corresponding to gains of 74% and 120%. The VLAC-Cut guided pipeline also achieves shorter execution time in the final iteration. In real-robot experiments, we observe that the VLAC-Cut guided pipeline produces more concise and efficient action sequences.

These results demonstrate that, under the same human-labor budget, the VLAC-Cut guided pipeline can automatically extract additional curated rollout data, leading to substantial improvements in policy performance and human efficiency.

6 Author Contribution Statement

The authors confirm their contribution as follows:

- Shaopeng Zhai (zsp1197@163.com):
- Qi Zhang (zhangqi.fqz@gmail.com):
- Tianyi Zhang (tianyizhang0729@163.com):
- Fuxian Huang (hfuxian@zju.edu.cn):
- Haoran Zhang (zhrecnucee@gmail.com):
- Zijun Xu (25113070118@m.fudan.edu.cn):
- Zhanhui Lin (zhanhuilin@link.cuhk.edu.cn):

Task	Model	Throughput	Success rate	Execution Time (s)	Failure progress
Refrigerator	Base model	10	20%	72.3	40%
	HITL(i=1)	14	35%	89	42.3%
	Ours(i=1)	19	45%	85	49%
	HITL(i=2)	33	60%	65.6	56%
	Ours(i=2)	42	80%	67.2	62.5%
Microplate	Base model	13	30%	83.3	30%
	HITL(i=1)	16	40%	88.35	27.5%
	Ours(i=1)	18	45%	91.4	34.5%
	HITL(i=2)	25	60%	85.1	51.3%
	Ours(i=2)	42	90%	77.7	20%
Test Tube	Base model	14	35%	93	38.9%
	HITL(i=1)	19	45%	83	41.2%
	Ours(i=1)	20	55%	105	45%
	HITL(i=2)	28	70%	90.7	65%
	Ours(i=2)	37	95%	93.4	76%
Stirrer	Base model	66	55%	30	33.3%
	HITL(i=1)	83	70%	30.5	35%
	Ours(i=1)	112	90%	29	45%

Table 4: **Policy evaluation results across tasks and iterations.** We report the throughput, success rate, average execution time, and failure progress for the base model, HITL-only training, and the VLAC-Cut guided pipeline. The VLAC-Cut guided pipeline consistently outperforms HITL-only training across tasks and iterations, demonstrating its significant performance improvements.

Task	Model	Base demos.	HITL recovery	Curated rollout
Refrigerator	Base model	94	-	-
	HITL(i=1)	94	201	-
	Ours(i=1)	94	201	100
	HITL(i=2)	94	242	-
	Ours(i=2)	94	242	117
Microplate	Base model	112	-	-
	HITL(i=1)	112	168	-
	Ours(i=1)	112	168	173
	HITL(i=2)	112	134	-
	Ours(i=2)	112	134	116
Test Tube	Base model	150	-	-
	HITL(i=1)	150	352	-
	Ours(i=1)	150	352	142
	HITL(i=2)	150	300	-
	Ours(i=2)	150	300	141
Stirrer	Base model	101	-	-
	HITL(i=1)	101	230	-
	Ours(i=1)	101	230	334

Table 5: **Training data composition for each task and iteration.** Each entry denotes the number of training trajectories or curated rollout segments in the dataset. Since HITL recovery trajectories are generally shorter, more HITL recovery episodes are collected to maintain data balance. Curated rollout data can be extracted automatically and adjusted according to the training demand, enabling more effective policy optimization.

REFERENCES

Minttu Alakuijala, Reginald McLean, Isaac Woungang, Nariman Farsad, Samuel Kaski, Pekka Marttinen, and Kai Yuan. Video-language critic: Transferable reward functions for language-conditioned robotics. *arXiv preprint arXiv:2405.19988*, 2024. URL https://arxiv.org/abs/2405.19988.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *NeurIPS*, 2017.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Paweł Budzianowski, Emilia Wiśnios, Gracjan Góral, Igor Kulakov, Viktor Petrenko, and Krzysztof Walas. Opengvl–benchmarking visual temporal progress for data curation. *arXiv preprint arXiv:2509.17321*, 2025.

Annie S Chen, Suraj Nair, and Chelsea Finn. Learning generalizable robotic reward functions from" in-the-wild" human videos. *RSS*, 2021.

Qianzhong Chen, Justin Yu, Mac Schwager, Pieter Abbeel, Yide Shentu, and Philipp Wu. Sarm: Stage-aware reward modeling for long horizon robot manipulation. *arXiv preprint arXiv:2509.25358*, 2025a.

Shirui Chen, Cole Harrison, Ying-Chun Lee, Angela Jin Yang, Zhongzheng Ren, Lillian J. Ratliff, Jiafei Duan, Dieter Fox, and Ranjay Krishna. Topreward: Token probabilities as hidden zero-shot rewards for robotics. *arXiv preprint arXiv:2602.19313*, 2026.

Yuhui Chen, Shuai Tian, Shugao Liu, Yingting Zhou, Haoran Li, and Dongbin Zhao. Conrft: A reinforced fine-tuning method for vla models via consistency policy. *arXiv preprint arXiv:2502.05450*, 2025b.

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*, pp. 02783649241273668, 2023.

Jieming Cui, Tengyu Liu, Ziyu Meng, Jiale Yu, Ran Song, Wei Zhang, Yixin Zhu, and Siyuan Huang. Grove: A generalized reward for learning open-vocabulary physical skill. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 15781–15790, 2025.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Zihan Ding and Chi Jin. Consistency models as a rich and efficient policy class for reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Yuqing Du, Ksenia Konyushkova, Misha Denil, Akhil Raju, Jessica Landon, Felix Hill, Nando de Freitas, and Serkan Cabi. Vision-language models as success detectors. *arXiv preprint arXiv:2303.07280*, 2023.

Alejandro Escontrela, Ademi Adeniji, Wilson Yan, Ajay Jain, Xue Bin Peng, Ken Goldberg, Youngwoon Lee, Danijar Hafner, and Pieter Abbeel. Video prediction models as rewards for reinforcement learning. *Advances in Neural Information Processing Systems*, 36:68760–68783, 2023.

Seyed Kamyar Seyed Ghasemipour, Ayzaan Wahid, Jonathan Tompson, Pannag Sanketi, and Igor Mordatch. Self-improving embodied foundation models. *arXiv preprint arXiv:2509.15155*, 2025.

Longxiang He, Li Shen, Junbo Tan, and Xueqian Wang. Aligniql: Policy alignment in implicit q-learning through constrained optimization. *arXiv preprint arXiv:2405.18187*, 2024.

Zheyuan Hu, Aaron Rovinsky, Jianlan Luo, Vikash Kumar, Abhishek Gupta, and Sergey Levine. Reboot: Reuse data for bootstrapping efficient real-world dexterous manipulation. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 1930–1949. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/hu23a.html.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: A vla that learns from experience. *arXiv:2511.14759*, 2025.

Zilin Kang, Chenyuan Hu, Yu Luo, Zhecheng Yuan, Ruijie Zheng, and Huazhe Xu. A forget-and-grow strategy for deep reinforcement learning scaling in continuous control. *arXiv preprint arXiv:2507.02712*, 2025.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

Pierre Krack, Tobias Jülg, Wolfram Burgard, and Florian Walter. Rewarding dino: Predicting dense rewards with vision foundation models. *arXiv preprint arXiv:2603.16978*, 2026.

Nishanth Kumar, Tom Silver, Willie McClinton, Linfeng Zhao, Stephen Proulx, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Jennifer Barry. Practice makes perfect: Planning to learn skill parameter policies. In *Robotics: Science and Systems (RSS)*, 2024.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, L. J. Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*, 2024. URL https://arxiv.org/abs/2403.13787.

Tony Lee, Andrew Wagenmaker, Karl Pertsch, Percy Liang, Sergey Levine, and Chelsea Finn. Roboreward: General-purpose vision-language reward models for robotics. *arXiv preprint arXiv:2601.00675*, 2026.

Youngwoon Lee, Andrew Szot, Shao-Hua Sun, and Joseph J Lim. Generalizable imitation learning from observation via inferring goal proximity. *Advances in Neural Information Processing Systems*, 34, 2021.

Lei Li, Yuancheng Wei, Zhihui Xie, Xuqing Yang, Yifan Song, Peiyi Wang, Chenxin An, Tianyu Liu, Sujian Li, Bill Yuchen Lin, Lingpeng Kong, and Qi Liu. Vlrewardbench: A challenging benchmark for vision–language generative reward models. *arXiv preprint arXiv:2411.17451*, 2024. URL https://arxiv.org/abs/2411.17451.

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. *arXiv preprint arXiv:2507.07969*, 2025.

Anthony Liang, Yigit Korkmaz, Jiahui Zhang, Minyoung Hwang, Abrar Anwar, Sidhant Kaushik, Aditya Shah, Alex S Huang, Luke Zettlemoyer, Dieter Fox, et al. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons. *arXiv preprint arXiv:2603.02115*, 2026.

Jingli Lin, Runsen Xu, Shaohao Zhu, Sihan Yang, Peizhou Cao, Yunlong Ran, Miao Hu, Chenming Zhu, Yiman Xie, Yilin Long, et al. Mmsi-video-bench: A holistic benchmark for video-based spatial intelligence. *arXiv preprint arXiv:2512.10863*, 2025.

Muhan Lin, Shuyang Shi, Yue Guo, Behdad Chalaki, Vaishnav Tadiparthi, Ehsan Moradi Pari, Simon Stepputtis, Joseph P Campbell, and Katia P Sycara. Navigating noisy feedback: Enhancing reinforcement learning with error-prone language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 16002–16014, 2024.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36: 44776–44791, 2023a.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023b.

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

Tung M. Luu, Younghwan Lee, Donghoon Lee, Sunho Kim, Min Jun Kim, and Chang D. Yoo. Erl-vlm: Enhancing rating-based reinforcement learning to effectively leverage feedback from large vision–language models. *arXiv preprint arXiv:2506.12822*, 2025. URL https://arxiv.org/abs/2506.12822.

Lei Lv, Yunfei Li, Yu Luo, Fuchun Sun, Tao Kong, Jiafeng Xu, and Xiao Ma. Flow-based policy for online reinforcement learning. *arXiv preprint arXiv:2506.12811*, 2025.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *CoRL*, 2020.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. In *The Eleventh International Conference on Learning Representations*, 2022.

Yecheng Jason Ma, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. Liv: Language-image representations and rewards for robotic control. In *International Conference on Machine Learning*, pp. 23301–23320. PMLR, 2023a.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023b.

Yecheng Jason Ma, Joey Hejna, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, et al. Vision language models are in-context value learners. In *The Thirteenth International Conference on Learning Representations*, 2024.

Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah A. Smith, Hannaneh Hajishirzi, and Nathan Lambert. Rewardbench 2: Advancing reward model evaluation. *arXiv preprint arXiv:2506.01937*, 2025. URL https://arxiv.org/abs/2506.01937.

Yiming Mao, Zixi Yu, Weixin Mao, Yinhao Li, Qirui Hu, Zihan Lan, Minzhao Zhu, and Hua Chen. Arm: Advantage reward modeling for long-horizon manipulation. *arXiv preprint arXiv:2604.03037*, 2026.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. In *CoRL*, 2022.

Chao Pan et al. SOP: A scalable online post-training system for vision-language-action models, 2026. URL https://arxiv.org/abs/2601.03044.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. *arXiv preprint arXiv:2502.02538*, 2025.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

Physical Intelligence. $\pi^*_{0.6}$: A VLA that learns from experience, 2025. URL https://arxiv.org/abs/2511.14759.

Juan Rocamonde, Victoriano Montesinos, Elvis Nava, Ethan Perez, and David Lindner. Vision-language models are zero-shot reward models for reinforcement learning. *arXiv preprint arXiv:2310.12921*, 2023.

Pierre Sermanet, Kelvin Xu, and Sergey Levine. Unsupervised perceptual rewards for imitation learning. *arXiv preprint arXiv:1612.06699*, 2016.

Pierre Sermanet, Tianli Ding, Jeffrey Zhao, Fei Xia, Debidatta Dwibedi, Keerthana Gopalakrishnan, Christine Chan, Gabriel Dulac-Arnold, Sharath Maddineni, Nikhil J Joshi, et al. Robovqa: Multimodal long-horizon reasoning for robotics. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 645–652. IEEE, 2024.

Lin Shao, Toki Migimatsu, Qiang Zhang, Karen Yang, and Jeannette Bohg. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2020.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Hao Shi et al. Beyond imitation: Reinforcement learning-based sim-real co-training for vision-language-action models, 2026. URL https://arxiv.org/abs/2602.12628.

Anukriti Singh, Amisha Bhaskar, Peihong Yu, Souradip Chakraborty, Ruthwik Dasyam, Amrit Bedi, and Pratap Tokekar. Varp: Reinforcement learning from vision–language model feedback with agent-regularized preferences. *arXiv preprint arXiv:2503.13817*, 2025. URL https://arxiv.org/pdf/2503.13817.

Laura Smith, Ilya Kostrikov, and Sergey Levine. Demonstrating a walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. *Robotics: Science and Systems (RSS) Demo*, 2(3):4, 2023.

Laura Smith, Yunhao Cao, and Sergey Levine. Grow your limits: Continuous improvement with real-world rl for robotic locomotion. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10829–10836. IEEE, 2024.

Sumedh Sontakke, Jesse Zhang, Séb Arnold, Karl Pertsch, Erdem Bıyık, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. Roboclip: One demonstration is enough to learn robot policies. *Advances in Neural Information Processing Systems*, 36, 2024.

Xiaoyu Sun et al. AtomVLA: Scalable post-training for robotic manipulation via predictive latent world models, 2026. URL https://arxiv.org/abs/2603.08519.

Huajie Tan, Sixiang Chen, Yijie Xu, Zixiao Wang, Yuheng Ji, Cheng Chi, Yaoxu Lyu, Zhongxia Zhao, Xiansheng Chen, Peterson Co, Shaoxuan Xie, Guocai Yao, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. Robo-dopamine: General process reward modeling for high-precision robotic manipulation. *arXiv preprint arXiv:2512.23703*, 2025.

Sreyas Venkataraman, Yufei Wang, Ziyu Wang, Zackory Erickson, and David Held. Real-world offline reinforcement learning from vision language model feedback. *arXiv preprint arXiv:2411.05273*, 2024. URL https://arxiv.org/abs/2411.05273.

Yixuan Wang et al. Learning while deploying: Fleet-scale reinforcement learning for generalist robot policies, 2026. URL https://arxiv.org/abs/2605.00416.

Yufei Wang, Zhanyi Sun, Jesse Zhang, Zhou Xian, Erdem Biyik, David Held, and Zackory Erickson. Rl-vlm-f: Reinforcement learning from vision–language foundation model feedback. *arXiv preprint arXiv:2402.03681*, 2024. URL https://arxiv.org/abs/2402.03681.

Yanru Wu, Weiduo Yuan, Ang Qi, Vitor Guizilini, Jiageng Mao, and Yue Wang. Large reward models: Generalizable online robot reward generation with vision-language models. *arXiv preprint arXiv:2603.16065*, 2026.

Xingjian Xiao et al. World-Env: Leveraging world model as a virtual environment for VLA post-training, 2025. URL https://arxiv.org/abs/2509.24948.

Xingjian Xiao et al. ROVE: Unlocking human interventions for humanoid manipulation via reinforcement learning, 2026. URL https://arxiv.org/abs/2606.17011.

Yuechen Xie, Xiaoyan Zhang, Yicheng Shan, Hao Zhu, Rui Tang, Rong Wei, Mingli Song, Yuanyu Wan, and Jie Song. Spatialqa: A benchmark for evaluating spatial logical reasoning in vision-language models. *arXiv preprint arXiv:2602.20901*, 2026.

Jingyun Yang, Max Sobol Mark, Brandon Vu, Archit Sharma, Jeannette Bohg, and Chelsea Finn. Robot fine-tuning made easy: Pre-training rewards and policies for autonomous real-world reinforcement learning, 2023.

Michihiro Yasunaga, Luke Zettlemoyer, and Marjan Ghazvininejad. Multimodal rewardbench: Holistic evaluation of reward models for vision–language models. *arXiv preprint arXiv:2502.14191*, 2025. URL https://arxiv.org/abs/2502.14191.

Zewei Ye, Weifeng Lu, Minghao Ye, Tao Lin, Shuo Yang, Junchi Yan, and Bo Zhao. Robofac: A comprehensive framework for robotic failure analysis and correction. *arXiv preprint arXiv:2505.12224*, 2025.

Checheng Yu, Chonghao Sima, Gangcheng Jiang, Hai Zhang, Haoguang Mai, Hongyang Li, Huijie Wang, Jin Chen, Kaiyang Wu, Li Chen, et al. χ_0: Resource-aware robust manipulation via taming distributional inconsistencies. *arXiv preprint arXiv:2602.09021*, 2026.

Shaopeng Zhai, Jie Wang, Tianyi Zhang, Fuxian Huang, Qi Zhang, Ming Zhou, Jing Hou, Yu Qiao, and Yu Liu. Building open-ended embodied agent via language-policy bidirectional adaptation, 2024. URL https://arxiv.org/abs/2401.00006.

Shaopeng Zhai, Qi Zhang, Tianyi Zhang, Fuxian Huang, Haoran Zhang, Ming Zhou, Shengzhe Zhang, Litao Liu, Sixu Lin, and Jiangmiao Pang. A vision-language-action-critic model for robotic real-world reinforcement learning. *arXiv preprint arXiv:2509.15937*, 2025.

Jiahui Zhang, Yusen Luo, Abrar Anwar, Sumedh A. Sontakke, Joseph J. Lim, Jesse Thomason, Erdem Bıyık, and Jesse Zhang. Rewind: Language-guided rewards teach robot policies without new demonstrations. *arXiv preprint arXiv:2505.10911*, 2025a. URL https://arxiv.org/abs/2505.10911.

Jianshu Zhang, Chengxuan Qian, Haosen Sun, Haoran Lu, Dingcheng Wang, Letian Xue, and Han Liu. Progresslm: Towards progress reasoning in vision-language models. *arXiv preprint arXiv:2601.15224*, 2026.

Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, et al. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11142–11152, 2025b.

Zhiyuan Zhou, Pranav Atreya, Abraham Lee, Homer Walke, Oier Mees, and Sergey Levine. Autonomous improvement of instruction following skills via foundation models. *arXiv preprint arXiv:407.20635*, 2024a.

Zhiyuan Zhou, Andy Peng, Qiyang Li, Sergey Levine, and Aviral Kumar. Efficient online reinforcement learning fine-tuning need not retain offline data. *arXiv preprint arXiv:2412.07762*, 2024b.

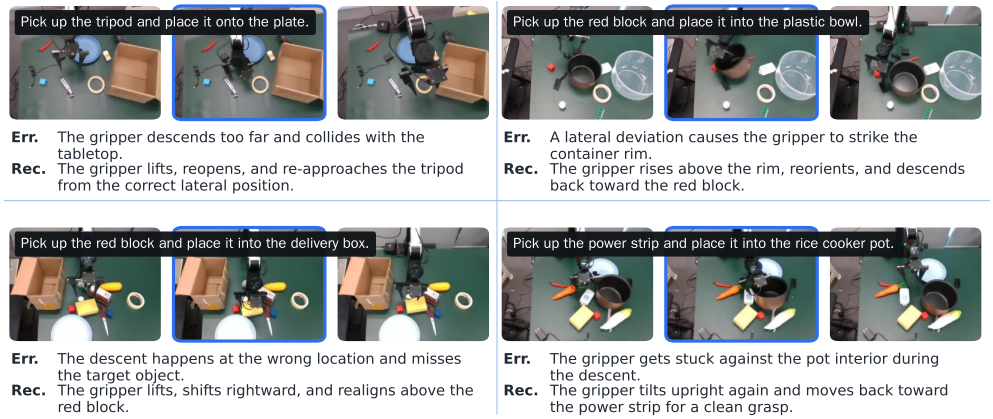

Figure 7: Representative failure (Err.) and recovery modes (Rec.) in ARX-data. These examples provide visual evidence for annotating stagnation, regression, and recovery in signed progress trajectories.

A Additional Details for VLAC-Cut

A.1 Progress Annotation Dataset Details

The full annotation inventory is constructed from public robot datasets and a targeted in-house real-world collection. DROID (Khazatsky et al., 2024) contributes diverse real-world manipulation videos, while LIBERO (Liu et al., 2023a) and VLABench (Zhang et al., 2025b) provide simulated task variation and benchmark-style manipulation scenarios. From these datasets, we extract task instructions, camera videos, trajectory metadata, and semantic task information, and then re-annotate selected videos with our progress labels.

Although these public datasets expand task and scene coverage, they are largely demonstration-centric: most trajectories correspond to successful or near-expert executions. This distribution is insufficient for learning a general-purpose progress critic that must evaluate policy behavior beyond nominal demonstrations. A critic trained primarily on expert-like trajectories may implicitly assume that execution quality improves monotonically with time and fail to distinguish partial progress, stagnation, regression, recoverable errors, and unrecoverable failure. Reverse or rewound video augmentation can expose models to synthetic decreasing-progress examples (Zhang et al., 2025a), but it cannot fully substitute for physically executed non-expert robot rollouts with realistic contact dynamics and out-of-expert-distribution failure modes.

To address this limitation, we collect ARX-data, a targeted in-house real-world robot dataset designed to increase coverage of physically executed non-monotonic progress. ARX-data is collected on a single-arm real-world platform built around an ARX-5 robotic arm under master–slave teleoperation, with synchronized multi-view videos. In addition to successful demonstrations, the collection includes deliberately induced deviations from nominal execution, covering grasp failures, workspace collisions, unstable contacts, wrong-object interactions, inaccurate placements, object drops, deviations from the intended motion, and recovery behaviors such as re-grasping or correcting an object pose after failure, as displayed in Figure 7. These trajectories provide physically grounded examples in which task progress may increase, stagnate, decrease, or later recover.

Overall, the source design prioritizes real-world robot behavior. DROID provides large-scale real-world manipulation diversity, while ARX-data complements it with physically executed non-monotonic progress trajectories. LIBERO and VLABench are retained as supplementary simulated sources for controlled task variation and additional manipulation coverage, rather than as the main basis of the dataset. Progress labels are not inherited from any source dataset; all selected videos are re-annotated using our process-level schema to produce view-conditioned signed progress supervision.

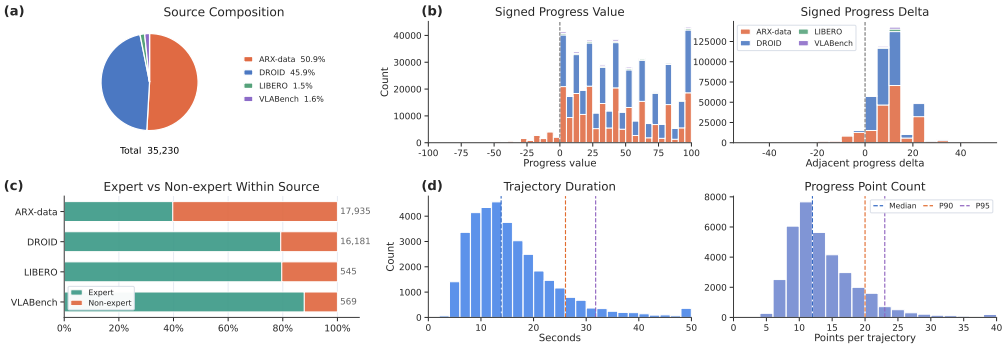

Figure 8: Full annotation inventory statistics. (a) Source composition and execution-quality distribution of the annotated robot video corpus. ARX-data and DROID dominate the real-world portion, while LIBERO and VLABench provide supplementary simulated coverage. (b) Signed progress statistics across data sources. The distribution of progress values and adjacent progress deltas shows that the corpus contains not only forward progress but also stagnation, regression, and recovery behavior. (c) Temporal characteristics of the progress annotations. Most trajectories are short manipulation episodes, while each episode contains multiple progress points that provide dense process-level supervision.

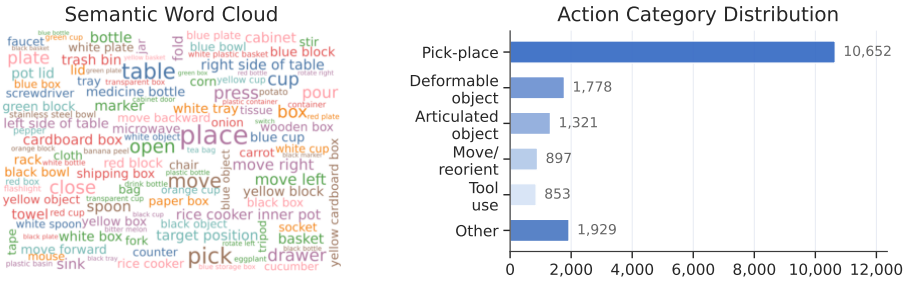

Figure 9: Semantic and action diversity of the full annotation inventory. The corpus covers a broad range of manipulation objects, spatial relations, and action categories, with pick-and-place tasks forming the largest group.

The full annotation inventory contains 35,230 records, 26,615 episodes, 15,206 task units, and 464,446 progress points. A record denotes one annotated video-level sample and is the basic unit of progress annotation. An episode denotes one physical task execution and may correspond to multiple camera- or view-specific records. A task unit denotes a semantic manipulation task used for measuring task coverage and organizing benchmark splits. For progress-pattern analysis, a non-expert record is defined as one whose ordered progress points contain at least one adjacent decrease in signed progress; otherwise it is expert. Under this annotation-derived criterion, the full inventory contains 20,885 expert records and 14,345 non-expert records. Figures 8 and 9 summarize source composition, execution-quality distribution, signed progress statistics, temporal annotation density, and semantic/action diversity.

A.2 Annotation Schema and Protocol

Each progress point is task-conditioned rather than time-conditioned: the same visual motion may indicate progress for one instruction and irrelevant or harmful behavior for another. Annotators therefore mark sparse keyframes where task state changes, including forward progress, stagnation, regression, and recovery. For each episode, the annotator first reviews the task instruction and creates a task-level plan. The plan decomposes the instruction into semantic milestones that describe the nominal progression of the task, typically spanning the 0–100% completion range. This plan provides a task-conditioned reference for assigning signed progress values, but pointwise progress labels themselves are not restricted to monotonic completion. Progress points may take negative values when the execution moves away from the goal or interacts with an incorrect object.

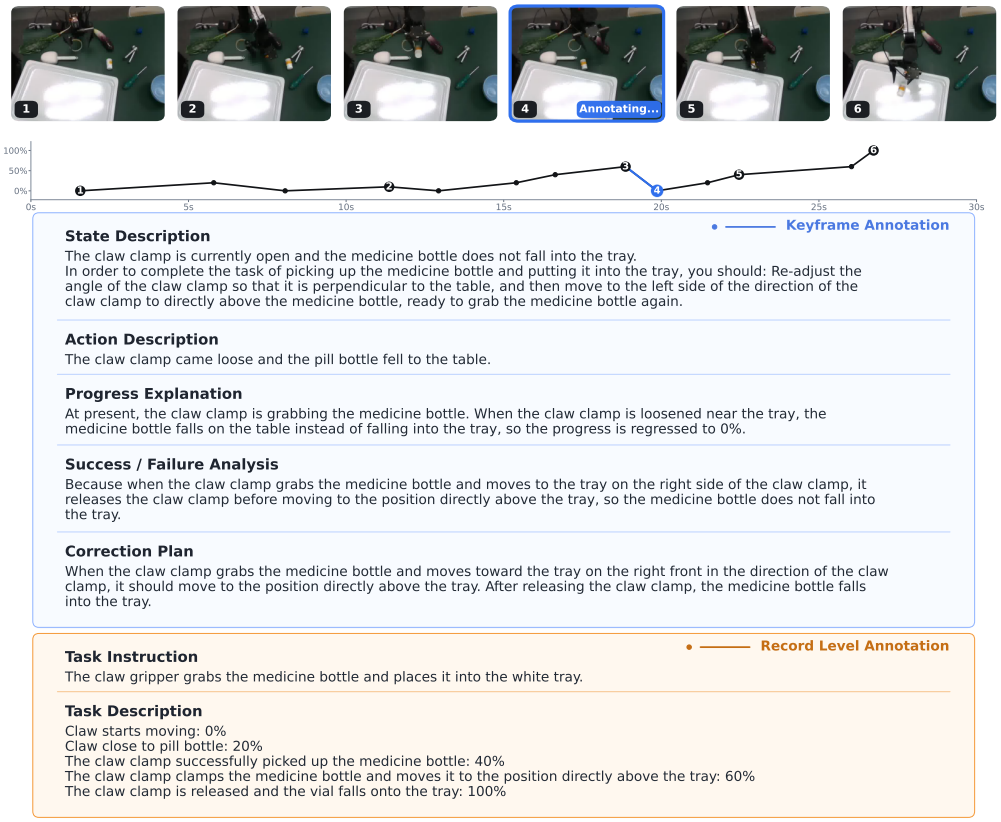

Figure 10: Example of sparse progress annotation for a robot manipulation trajectory. Only selected keyframes are manually annotated with signed progress values and structured diagnostic supervision, including state description, action description, progress explanation, success/failure analysis, and correction planning. Intermediate video frames are shown as unannotated temporal context, while the current keyframe corresponds to the active annotation target.

The annotator then marks progress points at meaningful timestamps. A progress point is not a uniformly sampled frame; it is a semantic event where the task state changes, a subgoal is achieved, an error occurs, the execution stagnates, or the trajectory reaches an inflection point. Annotators are instructed to identify moments where progress reverses, such as failed grasps, object drops, incorrect placements, or recovery attempts. These inflection points define the temporal boundaries at which a trajectory can transition from improvement to stagnation, regression, or recovery.

Each progress point aligns visual time with structured language supervision and optional spatial grounding. The textual annotation contains five complementary components: state description, action description, progress explanation, success/failure analysis, and correction plan. Optional grounding annotations localize the gripper end-effector and task-relevant objects on the keyframe corresponding to each progress point. This grounding links language explanations to visual evidence and supports models that reason jointly about object state, robot motion, and task progress. Figure 10 summarizes the annotation schema.

We use an interactive annotation interface to support process-level labeling at scale. The interface allows annotators to inspect multi-view videos, edit the task plan, mark progress points, assign signed progress values, and provide corresponding state, action, explanation, success/failure, correction, and grounding annotations. For datasets that provide robot trajectories or action information, the system can initialize candidate progress points using kinematic analysis. The kinematic mode loads robot positions and gripper states, rescales position, orientation, and gripper dimensions, unwraps rotations, filters trajectories, computes velocity and acceleration, extracts candidate keyframes, and refines them under weighted position, rotation, and gripper costs. The extracted trajectory indices are then mapped to video timestamps and frame numbers, producing candidate annotation anchors

Split	Records	Episodes	Tasks	Points	Expert / Non-exp.	ARX / DROID / Sim.
Train	24,652	20,479	13,996	331,762	15,467 / 9,185	10,357 / 13,457 / 838
Expert seen	1,043	1,033	1,043	11,769	1,043 / 0	356 / 641 / 46
Expert unseen	1,043	1,035	1,043	12,333	1,043 / 0	356 / 641 / 46
Non-expert seen	713	706	713	9,458	0 / 713	540 / 165 / 8
Non-expert unseen	716	706	716	9,850	0 / 716	540 / 168 / 8
Curated total	28,167	22,978	15,206	375,172	17,553 / 10,614	12,149 / 15,072 / 946

Table 6: Training and held-out evaluation split summary for the curated dataset. Records are video-level annotation units, episodes are physical task executions that may contain multiple view-specific records, and tasks denote semantic manipulation task units. The execution-quality column reports expert / non-expert records. The source column reports ARX-data / DROID-derived / simulated records, where simulated records combine LIBERO and VLABench. Counts of records and progress points are additive across splits, whereas episode and task-unit counts are reported as unique counts within each row and may not be additive when semantic task units are intentionally shared between the training split and the seen evaluation buckets.

in the visual stream. These automatically extracted keyframes are not treated as final progress labels; annotators can add, remove, or modify points according to task semantics.

A.3 Curated Split and Data Usage

After constructing the full annotation inventory, we curate a train–evaluation dataset for supervised training and diagnostic evaluation. The split is designed to evaluate two complementary forms of generalization: semantic task familiarity and progress-pattern recognition. Semantic task familiarity is measured by separating evaluation records into seen and unseen task units, while progress-pattern recognition is measured by separating expert-like trajectories from trajectories that contain non-monotonic or regressive progress patterns.

Seen and unseen are defined at the semantic task-unit level with respect to the annotation-derived instruction data used for supervised training. A seen evaluation record belongs to a task unit that appears in the training split, but the evaluated video record and its progress annotations are held out. An unseen evaluation record belongs to a task unit excluded from training instruction construction. Progress-pattern type is defined by the annotated progress trajectory: expert records correspond to trajectories without annotated progress regression, while non-expert records contain at least one regressive progress transition. Combining the semantic and progress-pattern axes yields four held-out evaluation buckets: expert seen, expert unseen, non-expert seen, and non-expert unseen.

To prevent leakage between training and evaluation, all progress annotations in the held-out split are excluded from prompt generation, data augmentation, in-context extension, and model fine-tuning. Records from the same physical execution, including different camera views of the same episode, are assigned to the same split. Within each evaluation bucket, each record corresponds to a distinct task unit, which prevents the benchmark from being inflated by repeated records of the same semantic task within a bucket. The resulting curated split is summarized in Table 6.

A.4 VLAC-Cut Training Details

The annotation-derived instruction data are complemented by targeted augmentations. Robotic executions are often non-monotonic: the robot may lose contact, move the object away from the goal, undo partial success, or recover after failure. To expose the model to such cases, we construct counterfactual reverse-progress examples from pairs of annotated progress points. An augmented clip traverses the visual segment from a later state back to an earlier state and then returns forward, while the target progress curve decreases during the reversed portion and increases during the recovery portion. This creates explicit regression-and-recovery supervision without additional robot rollouts and discourages the shortcut that later frames are always more complete.

Data group	Role in training	Used amount
Annotation-derived robot data	Robotic process understanding, failure diagnosis, correction, and action-aware reasoning	4.76M selected samples
LLaVA-style instruction data (Liu et al., 2023b)	General visual instruction following	200K converted samples
RoboReward (Lee et al., 2026)	Robot reward and trajectory-quality judgment	54,135 examples
RoboVQA (Sermanet et al., 2024)	Robot-centered VQA and long-horizon reasoning	829,502 video-text pairs
Spatial QA data (Xie et al., 2026)	Spatial relation and logical reasoning	250K converted samples
ProgressLM CoT (Zhang et al., 2026)	Progress reasoning from partial observations	25K CoT SFT samples
MMSI-Video-Bench (Lin et al., 2025)	Video-based spatial intelligence	1,106 QA samples / 1,278 clips

Table 7: Training data mixture used for supervised fine-tuning. Annotation-derived robot data form the core of the corpus, while auxiliary public datasets preserve general multimodal reasoning, robotics reasoning, progress estimation, and spatial/video understanding.

When object boxes and gripper keypoints are available at consecutive progress points, we generate grounding-based rationales as additional training targets. The procedure compares object centers, gripper positions, and task-relevant distances across time, then verbalizes geometric evidence that supports increasing, stagnant, or decreasing progress. We retain rationales only when the geometric evidence is consistent with the annotated progress direction, so this component aligns numerical progress prediction with visible spatial evidence rather than introducing a separate model module.

For episodes with robot trajectory information, we derive action-conditioned samples from kinematic keyframes or adjacent progress points. The input may contain one to four selected camera views, wrist views when available, the task instruction, the current robot state, and an optional action-intent sentence. The target is a relative action delta between the start and end states of the interval, expressed in millimeters for translation, degrees for rotation, and percentage units for gripper motion. Intervals with negligible motion are filtered out. This supervision helps the critic relate progress diagnosis and correction plans to executable robot motion, while the benchmark and downstream use still evaluate VLAC-CUT as a video-language critic rather than a policy.

The supervised fine-tuning corpus is centered on annotation-derived robot data and complemented with public multimodal, robotics, progress-reasoning, and spatial/video reasoning datasets. The annotation-derived component supplies the process supervision needed by VLAC-CUT: task planning, progress estimation, state/action description, progress explanation, failure diagnosis, correction planning, grounding-aware reasoning, and action-conditioned prediction. It also contains both expert-like executions and non-expert trajectories with stagnation, regression, failed attempts, and recovery behavior, which is essential for training a critic rather than a final-success classifier. Auxiliary datasets are included to preserve broad visual instruction following and robot-centered reasoning capabilities. Table 7 summarizes the mixture.

All components are converted into the same conversation format. This unified format allows image, video, language, grounding, and action-conditioned examples to be optimized under a single supervised fine-tuning interface. VLAC-CUT uses Qwen3-VL-30B-A3B-Instruct as the base backbone and is fine-tuned as a multimodal instruction-following progress critic. Training samples are formatted as multimodal conversations containing task instructions, task plans, images or video frames, and optional robot/action context. The model is trained to generate structured textual responses that include progress estimates, state and action descriptions, progress explanations, success/failure analysis, correction plans, grounding labels, and relative action deltas. We use supervised fine-tuning with an autoregressive language-modeling objective over assistant responses, so all prediction and explanation tasks share a unified generative interface. During training, we use an effective global batch size of 2304, a maximum sequence length of 15,240 tokens, and a peak learning rate of 3×10^{-5} with cosine decay. Training is conducted with MS-SWIFT and DeepSpeed ZeRO-3 on 144 NVIDIA A800-SXM4-80GB GPUs, and the full run takes 16 days.

A.5 VPB TASK DEFINITION AND METRICS

VPB evaluates video-language progress prediction over a dataset $\mathcal{D} = \{(l_i, \mathbf{o}_i, \mathbf{P}_i^{\text{key}}, \mathbf{p}_i)\}_{i=1}^{N}$, where l_i is the natural language instruction describing the task, $\mathbf{o}_i = (o_{i,1}, o_{i,2}, \ldots, o_{i,T_i})$ is the observation video consisting of T_i frames, $\mathbf{P}_i^{\text{key}} = \{(t_{i,m}, p_{i,m}^{\text{key}})\}_{m=1}^{K_i}$ is a set of K_i keyframe annotations, where $t_{i,m} \in \{1, \ldots, T_i\}$ denotes a frame index and $p_{i,m}^{\text{key}} \in [-100, 100]$ is the corresponding progress value, and $\mathbf{p}_i = (p_{i,1}, \ldots, p_{i,T_i})$ is the dense ground-truth progress trajectory. The progress value $p \in [-100, 100]$ represents the task completion status: $p = 100$ indicates full completion, $p = 0$ denotes the initial state, and $p < 0$ represents regressive states worse than the initial setup. Given sparse keyframe annotations $\mathbf{P}_i^{\text{key}}$ with $t_{i,1} < t_{i,2} < \cdots < t_{i,K_i}$, the dense trajectory $\mathbf{p}_i$ is obtained via linear interpolation between adjacent keyframes:

$$p_{i,t} = p_{i,m}^{\text{key}} + \frac{t - t_{i,m}}{t_{i,m+1} - t_{i,m}} \left(p_{i,m+1}^{\text{key}} - p_{i,m}^{\text{key}} \right), \quad t_{i,m} \le t \le t_{i,m+1},\ m \in \{1, \ldots, K_i - 1\}. \tag{4}$$

Given an instruction l_i and video $\mathbf{o}_i$, a method should output a progress sequence $\hat{\mathbf{p}}_i = (\hat{p}_{i,1}, \ldots, \hat{p}_{i,T_i})$. The predicted trajectory $\hat{\mathbf{p}}_i$ is evaluated against the ground-truth $\mathbf{p}_i$ across multiple metrics.

All metrics are formally defined with respect to an arbitrary evaluation subset $\mathcal{S} \subseteq \mathcal{D}$. Progress Rank Correlation (PRC) measures whether the predicted sequence preserves the temporal ordering of the annotated progress states using Spearman rank correlation ρ_S:

$$\text{PRC}_i = \rho_S \left(\hat{\mathbf{p}}_i, \mathbf{p}_i \right), \qquad \text{PRC}(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \text{PRC}_i. \tag{5}$$

Value-Order Correlation (VOC) follows the rank-correlation evaluation introduced by GVL for value prediction on expert videos (Ma et al., 2024). It assesses whether the predicted values increase monotonically with the chronological progression of frames and is evaluated exclusively on expert data:

$$\text{VOC}_i = \rho_S \left(\hat{\mathbf{p}}_i, (1, 2, \ldots, T_i) \right), \qquad \text{VOC}(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \text{VOC}_i. \tag{6}$$

Mean Absolute Error (MAE) evaluates absolute calibration error on the VPB scale:

$$\text{MAE}_i = \frac{1}{T_i} \sum_{t=1}^{T_i} |\hat{p}_{i,t} - p_{i,t}|, \qquad \text{MAE}(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \text{MAE}_i. \tag{7}$$

Terminal-state metrics evaluate whether a method reliably discriminates the ultimate task outcome. We use 90% as a near-completion threshold:

$$y_i^T = \mathbb{I}\left[p_{i,T_i} \ge 90 \right], \qquad \hat{y}_i^T = \mathbb{I}\left[\hat{p}_{i,T_i} \ge 90 \right]. \tag{8}$$

Aggregating these binary predictions over $\mathcal{S}$ yields true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). Terminal-State Accuracy is:

$$\text{TSA}(\mathcal{S}) = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}. \tag{9}$$

The class-wise F1 scores for successful states (F1_S) and failed/incomplete states (F1_F) are:

$$\text{F1}_S(\mathcal{S}) = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}, \qquad \text{F1}_F(\mathcal{S}) = \frac{2\text{TN}}{2\text{TN} + \text{FP} + \text{FN}}, \tag{10}$$

and their macro average is:

$$\text{MacroF1}_T(\mathcal{S}) = \frac{1}{2} \left(\text{F1}_S(\mathcal{S}) + \text{F1}_F(\mathcal{S}) \right). \tag{11}$$

Local direction metrics evaluate sensitivity to improvement, stagnation, and regression at the keyframe level. For evaluating transition dynamics, predictions are sampled at official annotation frames $t_{i,m}$, yielding $\hat{\mathbf{P}}_i^{\text{key}} = \{(t_{i,m}, \hat{p}_{i,m}^{\text{key}})\}_{m=1}^{K_i}$ where $\hat{p}_{i,m}^{\text{key}} = \hat{p}_{i,t_{i,m}}$. For any adjacent keyframe

pair $(m, m+1)$, the local progress difference $\Delta p^{\text{key}}_{i,m} = p^{\text{key}}_{i,m+1} - p^{\text{key}}_{i,m}$ and its prediction $\Delta \hat{p}^{\text{key}}_{i,m}$ are mapped to a direction class $c(\Delta) \in \{+, 0, -\}$ using a margin threshold $\tau = 10$:

$$c(\Delta) = \begin{cases} +, & \Delta > \tau, \\ 0, & |\Delta| \le \tau, \\ -, & \Delta < -\tau. \end{cases} \tag{12}$$

Let $d_{i,m} = c(\Delta p^{\text{key}}_{i,m})$ and $\hat{d}_{i,m} = c(\Delta \hat{p}^{\text{key}}_{i,m})$ denote the ground-truth and predicted direction labels. For subset $\mathcal{S}$, define $\mathcal{M}(\mathcal{S}) = \{(i, m) \mid i \in \mathcal{S}, 1 \le m < K_i\}$. Local Direction Accuracy is:

$$\text{Acc}_D(\mathcal{S}) = \frac{1}{|\mathcal{M}(\mathcal{S})|} \sum_{(i,m)\in\mathcal{M}(\mathcal{S})} \mathbb{I}\left[\hat{d}_{i,m} = d_{i,m}\right]. \tag{13}$$

The class-wise F1 for direction class $c \in \{+, 0, -\}$ is:

$$\text{F1}_c(\mathcal{S}) = \frac{2\text{TP}_c}{2\text{TP}_c + \text{FP}_c + \text{FN}_c}. \tag{14}$$

The local direction macro F1 is:

$$\text{MacroF1}_D(\mathcal{S}) = \frac{1}{3}\left(\text{F1}_+(\mathcal{S}) + \text{F1}_0(\mathcal{S}) + \text{F1}_-(\mathcal{S})\right). \tag{15}$$

This metric penalizes models that hallucinate progress during stagnation or fail to detect regression in non-expert executions.