

---

# RL Token Reproduction: Efficient and Accurate VLA Control via RL Token Representations

---

**Jiayu Yang\***

School of Informatics  
Xiamen University  
Xiamen, Fujian 361102  
jiayuyangai@stu.xmu.edu.cn

## Abstract

Achieving fast and precise robotic task execution remains a major challenge in embodied intelligence. While large pretrained Visual-Language-Action models provide strong perceptual grounding and task generalization, they often suffer from low execution speed and sample inefficiency in critical stages of manipulation tasks. Recent work has proposed token-based online reinforcement learning as a lightweight mechanism for improving the execution efficiency and success rate of frozen VLA policies, which extracts a compact representation RL token from a frozen model and trains a lightweight actor-critic online. This architecture allows the robot to refine actions efficiently while retaining the semantic and visuomotor knowledge of the underlying model. Original experiments demonstrate that RLT increases task success rates by 5-35% and execution speed by 3-4× compared to baseline policies within minutes to a few hours of online practice, even surpassing human teleoperation speed in some scenarios. Official ablation studies reveal that both the RL token and reference-action regularization are critical for these gains. In this paper, we present a reproduction study of the RL Token framework and evaluate its effectiveness on the LIBERO-10 benchmark using a lightweight SmolVLA backbone, observing improvements in success rate and average episode duration compared to the frozen fine-tuned SmolVLA baseline. Overall, these findings highlight that compact token-based online RL, combined with structured action representations and reference-action guidance, is essential for improving both the speed and accuracy of robotic manipulation.

## 1 Introduction

Achieving fast, reliable, and precise robotic manipulation remains a central challenge in embodied intelligence, especially when robots must operate under limited data, sparse feedback, and complex task dynamics Wen et al. [2025], Kim et al. [2024], Black et al. [2026], Liu et al. [2024], Lu et al. [2025]. Vision-Language-Action (VLA) models have recently shown strong potential for addressing this challenge by integrating visual perception, language-conditioned reasoning, and action generation into a unified policy Brohan et al. [2023], Zhao et al. [2025], Wang et al. [2025]. Through large-scale pretraining on diverse robotic and multimodal data, these models acquire rich semantic and visuomotor priors, enabling flexible task specification, broad generalization, and strong zero-shot or few-shot performance across manipulation scenarios Team et al. [2024], Collaboration et al. [2025], Li et al. [2024], Wu et al. [2023]. However, despite these advantages, pretrained VLA policies still face important limitations in practical robotic execution. In particular, frozen VLA models often struggle with the speed, precision, and sample efficiency required in the critical stages of manipulation

---

\*Reinforcement Learning Course Project

tasks, such as contact-rich alignment, insertion, or fine motor adjustment, where small errors can accumulate into task failure Xu et al. [2026], Kim et al. [2025], Yu et al. [2025], Zhang et al. [2026], Pei et al. [2025].

One key challenge lies in the limited speed and precision of pretrained VLA policies during critical manipulation stages Kim et al. [2025], Wang et al. [2026], Wen et al. [2025], Pei et al. [2025], Yu et al. [2025]. Although imitation-learned VLA models provide strong semantic grounding and broad generalization, their behavior is often bounded by the quality and coverage of demonstration data. As a result, they may become slow, hesitant, or overly conservative in precision-sensitive phases such as alignment, insertion, fastening, or contact-rich adjustment Tan et al. [2025], Li et al. [2025], Chen et al. [2025]. Small errors in pose, force, or timing can lead to repeated probing, increased execution time, and eventual task failure. Therefore, pretrained VLAs still require online adaptation to achieve reliable fine-grained manipulation.

Another major challenge is the difficulty of adapting large VLA models with online reinforcement learning Guo et al. [2025], Liu et al. [2026], Pan et al. [2026]. Directly fine-tuning the entire VLA policy could improve task-specific behavior, but modern VLA models contain large vision-language backbones and high-dimensional action modules, making full-model RL computationally expensive and sample inefficient Jiang et al. [2026], Omaisani and Mohamed [2025], Mitra et al. [2025]. This is especially problematic in real-world robotics, where each rollout costs physical time, supervision, and potential recovery effort. Thus, effective adaptation should preserve pretrained VLA knowledge while enabling lightweight and sample-efficient policy improvement.

A third challenge is that lightweight real-world RL methods often fail to exploit the pretrained knowledge of VLA models Yu et al. [2026]. Conventional RL systems usually train small policies from scratch or use standard visual encoders, which improves efficiency but discards the semantic and visuomotor priors learned by large-scale VLA pretraining Moroncelli et al. [2026]. In sparse-reward, precision-critical tasks, this can make exploration unstable and inefficient, causing slow learning, unsafe behaviors, and poor generalization Neary et al. [2025]. Therefore, an effective solution should combine the adaptability of lightweight online RL with the perceptual and behavioral priors of pretrained VLAs.

To address these limitations, Xu et al. from Physical Intelligence introduce RL Token (RLT), a lightweight online reinforcement learning framework for improving pretrained VLA policies Xu et al. [2026]. RLT freezes the VLA backbone and exposes a compact RL token as an efficient interface for downstream RL. A small actor-critic module then refines VLA-proposed action chunks, while reference-action regularization keeps the learned policy close to the original VLA behavior. This design preserves pretrained semantic and visuomotor priors while enabling sample-efficient adaptation for precision-critical manipulation.

RLT also adopts a staged and data-efficient training pipeline. It first adapts the VLA with task-specific demonstrations to expose the RL token, then performs off-policy online RL on top of the frozen model. By using VLA warmup rollouts, optional human interventions, and critical-phase-focused training, RLT reduces unnecessary exploration and improves learning under sparse rewards. These properties make it a practical approach for improving both execution speed and task success in robotic manipulation.

The main contributions of this paper are summarized as follows:

**1. RLT-style adaptation of a frozen SmolVLA backbone.** We reproduce the high-level interface of RL Token (RLT) in an accessible SmolVLA-LIBERO setting. Instead of updating the full VLA backbone, our method keeps the task fine-tuned SmolVLA policy frozen and uses it as both a representation provider and a behavioral prior. This allows a lightweight actor-critic module to refine task-specific actions while preserving the semantic and visuomotor knowledge learned by the base policy.

**2. Construction of a staged replay-buffer and feature-extraction pipeline.** We implement a staged pipeline for RLT-style policy improvement in simulation. We first collect rollouts from the frozen fine-tuned SmolVLA policy on LIBERO-10 and convert them into transition buffers. We then run the frozen SmolVLA policy on the buffered observations to extract reference actions and compact action-latent features. These cached features provide an efficient interface for training a lightweight residual actor-critic without repeatedly running the full VLA during RL updates.

**3. Evaluation of reference-guided single-step residual action refinement.** We evaluate whether reference-guided residual learning can improve the success rate and execution efficiency of a frozen SmolVLA policy. Unlike the original RLT system, which operates over action chunks and constructs RL tokens from full VLA token embeddings, our implementation uses a pooled SmolVLA action-head latent as an action-latent proxy token and refines single executed 7-dimensional LIBERO actions. This provides a practical partial reproduction and adaptation of the RLT idea under limited computational resources.

## 2 Background

### 2.1 Vision-Language-Action models

Vision-Language-Action models unify visual perception, language understanding, and robot control by mapping multimodal observations and task instructions into executable actions Ma et al. [2026], Sapkota et al. [2026], Din et al. [2025], Shao et al. [2025], Guan et al. [2025]. Recent systems such as SmolVLA,  $\pi_0$ , and  $\pi_{0.5}$  show that large-scale vision-language pretraining can provide strong semantic grounding, flexible task specification, and cross-task generalization for robotic manipulation Black et al. [2026], Shukor et al. [2025], Intelligence et al. [2025]. These models often rely on imitation learning and generate continuous actions through autoregressive, chunked, flow-matching or diffusion-based action prediction, enabling them to perform diverse manipulation skills from visual observations and language commands Brohan et al. [2023], Zhao et al. [2023], Black et al. [2026], Chi et al. [2024], Kim et al. [2025]. However, despite their strong general-purpose capabilities, VLA policies are still largely limited by the quality and coverage of demonstration data Zhong et al. [2025], Zhang et al. [2025]. During precision-critical stages or contact-rich adjustment, frozen VLA models may produce slow, hesitant, or overly conservative behaviors, and small errors in pose or timing can accumulate into repeated probing or task failure Huang et al. [2026], Ye et al. [2026]. Moreover, directly adapting large VLA backbones with online reinforcement learning is often computationally expensive and sample inefficient Yang et al. [2025], Lu et al. [2026]. These limitations motivate lightweight adaptation methods that preserve pretrained VLA priors while improving execution speed and reliability in task-specific manipulation.

### 2.2 Reinforcement learning

Reinforcement learning (RL) provides a natural mechanism for improving robotic policies beyond the limitations of imitation learning Gao et al. [2019], Vecerik et al. [2018], Zhu et al. [2018], Jing et al. [2019], Gu et al. [2016]. By interacting with the environment and optimizing task rewards, an RL agent can discover faster, more precise, or more robust behaviors that may not appear in demonstration data Stranghöner et al. [2026], Lei et al. [2026], Luo et al. [2025]. In robotic manipulation, off-policy actor-critic methods are commonly used because they can reuse past experience through replay buffers and improve sample efficiency under limited interaction budgets. However, applying RL to real-world robot control remains challenging: rewards are often sparse, rollouts are costly, and exploration in high-dimensional continuous action spaces can be unstable or unsafe Haarnoja et al. [2018], Kalashnikov et al. [2018]. These issues become more severe in precision-critical tasks, where small errors during contact-rich phases can lead to failure Meng et al. [2026]. Therefore, effective RL-based adaptation should not learn from scratch, but should instead leverage strong pretrained priors while refining task-specific behaviors through efficient online interaction.

## 3 Method

We first review the original RLT formulation, and then describe how we adapt its main interface to the SmolVLA-LIBERO setting. As the official framework illustrated in Fig. 1, we reproduce an RL Token-based adaptation pipeline for improving frozen VLA policies on robotic manipulation tasks. The pretrained VLA first encodes visual observations and language instructions, producing both a compact RL-token representation and a reference action chunk. A lightweight actor-critic module then takes the RL token, robot state, and VLA reference action as input, and learns to generate refined action chunks through off-policy reinforcement learning. Experience collected from VLA warmup rollouts and online interactions is stored in a replay buffer for iterative policy updates. By

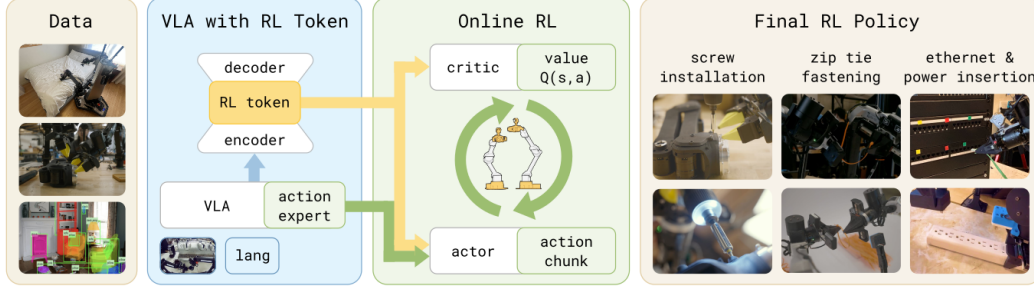


Figure 1: Official overview of the RL Token (RLT) framework. RLT freezes a pretrained VLA and exposes a compact RL token from the VLA’s internal representations. A lightweight actor-critic policy then uses this token together with the VLA reference action to perform online reinforcement-learning-based refinement. In our implementation, we follow the same high-level interface but replace the original full VLA token representation with a pooled SmoVLA action-head latent and evaluate the method in the LIBERO simulator.

combining frozen VLA priors, reference-guided action refinement, and sample-efficient online RL, the framework improves task execution speed and success without updating the full VLA model.

### 3.1 Preliminaries

Modern Vision-Language-Action (VLA) models commonly combine a large vision-language backbone with a continuous action-generation module. Representative models such as  $\pi_0$  and  $\pi_{0.5}$  encode visual observations, language instructions, and robot proprioceptive states into a shared multimodal representation, which is then used by an action expert to generate executable robot actions. Formally, given an observation  $o_t$ , a language instruction  $l$ , and proprioceptive state  $s_t^p$ , the VLA backbone produces a latent representation

$$h_t = f_{\text{VLM}}(o_t, l, s_t^p), \quad (1)$$

where  $f_{\text{VLM}}$  denotes the pretrained vision-language backbone. Instead of predicting a single action, these models usually generate an action chunk  $\hat{\mathbf{a}}_{t:t+H-1} \sim \pi_{\text{VLA}}(\cdot | h_t)$ , which contains a sequence of future actions for temporally coherent execution.

In VLM-flow-based VLA architectures, the action expert generates continuous action chunks through a flow-matching process conditioned on the multimodal representation. Let  $\mathbf{a}$  denote the ground-truth action chunk and let  $\epsilon \sim \mathcal{N}(0, I)$  be Gaussian noise with the same shape as  $\mathbf{a}$ . For a flow time  $\tau \in [0, 1]$ , the chunk state is defined as

$$\mathbf{z}^\tau = (1 - \tau)\epsilon + \tau\mathbf{a}. \quad (2)$$

The action expert learns a conditional velocity field  $v_\theta(\mathbf{z}^\tau, h_t, \tau)$ , which predicts the direction that transports the intermediate action state toward the target action chunk. The flow-matching objective is

$$\mathcal{L}_{\text{fm}} = \mathbb{E}_{\mathbf{a}, \epsilon, \tau} \left[ \|v_\theta(\mathbf{z}^\tau, h_t, \tau) - (\mathbf{a} - \epsilon)\|_2^2 \right]. \quad (3)$$

At inference time, the model starts from random noise and integrates the learned velocity field to obtain an executable action chunk. This formulation allows VLA policies to model complex and multimodal action distributions, making them suitable for long-horizon robotic manipulation.

However, since these policies are mainly learned from demonstration data, their execution quality is often bounded by the coverage, consistency, and precision of the demonstrations. In precision-critical stages such as insertion, alignment, fastening, or contact-rich adjustment, the generated action chunks may become slow, hesitant, or insufficiently accurate. Therefore, while VLM-flow-based VLA models provide strong semantic and visuomotor priors, additional online adaptation is still needed to improve task-specific speed and reliability.

### 3.2 RL Token-Based Adaptation Architecture

We adopt the official RLT architecture as the basis of our reproduction. The key idea of RLT is to keep the pretrained VLA model frozen and expose a compact representation, called the RL token, as the interface between the VLA backbone and the downstream reinforcement learning policy. Given the internal token embeddings of the VLA,

$$z_{1:M} = f_{\theta_{\text{VLA}}}(o_t, l, s_t^p), \quad (4)$$

RLT appends a learnable RL token embedding  $e_{\text{rl}}$  to the token sequence and processes it with a lightweight encoder transformer:

$$z_t^{\text{rl}} = g_{\phi}([z_{1:M}, e_{\text{rl}}])_{M+1}. \quad (5)$$

The resulting representation  $z_t^{\text{rl}}$  serves as a compact state feature for online RL, preserving task-relevant visual, linguistic, and action-related information from the frozen VLA.

On top of this RL-token representation, RLT trains a lightweight actor-critic module instead of updating the full VLA model. The RL state is defined as  $x_t = [z_t^{\text{rl}}, s_t^p]$ , where  $s_t^p$  denotes the robot proprioceptive state. The frozen VLA also provides a reference action chunk

$$\tilde{\mathbf{a}}_{t:t+C-1} \sim \pi_{\text{VLA}}(\cdot \mid o_t, l, s_t^p), \quad (6)$$

which guides the RL policy. The actor then predicts a refined action chunk conditioned on both the RL state and the VLA reference action:

$$\mathbf{a}_{t:t+C-1} \sim \pi_{\theta}(\cdot \mid x_t, \tilde{\mathbf{a}}_{t:t+C-1}), \quad (7)$$

while the critic estimates the value of the state-action pair:  $Q_{\psi}(x_t, \mathbf{a}_{t:t+C-1})$ . In this design, the VLA provides semantic grounding, visual understanding, and an initial action proposal, while the lightweight RL module locally refines the action to improve task-specific execution.

This architecture turns online reinforcement learning from unconstrained exploration in a high-dimensional action space into reference-guided refinement around a strong VLA prior. In our implementation, we preserve the high-level RLT interface but replace the original RL-token reconstruction module with a pooled SmolVLA action-head latent due to computational constraints.

### 3.3 Training Procedure

The training process consists of two main stages: RL-token adaptation and online actor-critic learning. We distinguish between two data sources used in RLT. The first one is a task-specific demonstration dataset  $\mathcal{D}$ , which is used to adapt the VLA and train the RL-token representation. In the original RLT system,  $\mathcal{D}$  is collected through task-specific teleoperation demonstrations. In our reproduction,  $\mathcal{D}$  is constructed from the demonstration trajectories provided by the LIBERO benchmark. Each trajectory contains visual observations, language instructions, proprioceptive states, and expert action sequences  $\mathcal{D} = \{(o_t, l, s_t^p, \mathbf{a}_t)\}_{t=1}^T$ . For chunk-based training, consecutive expert actions are grouped into action chunks  $\mathbf{a}_{t:t+H-1}$ , which serve as supervised targets for the base VLA policy and provide the trajectories from which internal VLA embeddings are extracted.

In the first stage, the pretrained VLA is adapted on  $\mathcal{D}$  to expose an RL-token representation. Given the internal VLA embeddings  $z_{1:M}$ , the RL-token encoder produces a compact representation  $z^{\text{rl}}$ , and a decoder is trained to reconstruct the original embeddings:

$$\mathcal{L}_{\text{ro}} = \mathbb{E}_{\mathcal{D}} \left[ \sum_{i=1}^M \left\| h_{\phi} \left( d_{\phi}([z^{\text{rl}}, \bar{z}_{1:i-1}]) \right)_i - \bar{z}_i \right\|_2^2 \right], \quad (8)$$

where  $\bar{z}_i = \text{sg}(z_i)$  denotes the stop-gradient VLA embedding. Optionally, this reconstruction objective can be combined with supervised VLA fine-tuning on the same demonstration dataset:

$$\phi, \theta_{\text{VLA}} = \arg \min_{\phi, \theta_{\text{VLA}}} \mathcal{L}_{\text{ro}}(\phi) + \alpha \mathcal{L}_{\text{VLA}}(\theta_{\text{VLA}}). \quad (9)$$

After this stage, both the VLA backbone and the RL-token module are frozen.

The second data source is the online replay buffer  $\mathcal{B}$ , which is used for reinforcement learning. Unlike  $\mathcal{D}$ , the replay buffer is not a fixed demonstration dataset. It is built during interaction with the

environment. Following the official RLT pipeline,  $\mathcal{B}$  is first initialized by rolling out the base VLA policy for  $N_{\text{warm}}$  steps. This warmup stage provides initial transitions generated by a competent policy, allowing the critic to start learning from meaningful behavior instead of random exploration. In our LIBERO reproduction, these warmup transitions are collected by executing the frozen SmolVLA policy in the simulator.

During online rollout, the frozen VLA produces a reference action chunk  $\tilde{\mathbf{a}}_{t:t+C-1}$  and the RL state is formed as  $x_t = (z_t^{\text{rl}}, l, s_t^p)$ . The actor then generates a refined action chunk conditioned on both the RL state and the VLA reference action  $\mathbf{a}_{t:t+C-1}$ . The refined action chunk is executed in the environment, producing the next observation and reward. The resulting transition is stored in the replay buffer  $\langle x_t, \mathbf{a}_{t:t+C-1}, \tilde{\mathbf{a}}_{t:t+C-1}, r_t, x_{t+1} \rangle \in \mathcal{B}$ . In the official real-robot setting, human intervention can optionally replace the actor output during unsafe or difficult behaviors, and the teleoperated action is then stored as the executed action and reference action. In our LIBERO reproduction, the reward is obtained from the simulator success signal. When only terminal success is available, intermediate rewards are set to zero and the terminal reward indicates whether the task is completed successfully.

The critic is trained with off-policy temporal-difference learning over action chunks:

$$\mathcal{L}_Q = \mathbb{E}_{(x, \mathbf{a}, x') \sim \mathcal{B}} \left[ \left( \hat{Q} - Q_\psi(x, \mathbf{a}) \right)^2 \right], \quad (10)$$

where

$$\hat{Q} = \sum_{i=1}^C \gamma^{i-1} r_i + \gamma^C \mathbb{E}_{\mathbf{a}' \sim \pi_\theta} [Q_\psi(x', \mathbf{a}')]. \quad (11)$$

The actor is optimized to maximize the critic value while staying close to the VLA reference action:

$$\mathcal{L}_\pi = \mathbb{E}_{\mathbf{a} \sim \pi_\theta} \left[ -Q_\psi(x, \mathbf{a}) + \beta \|\mathbf{a} - \tilde{\mathbf{a}}\|_2^2 \right]. \quad (12)$$

This reference-action regularization prevents the RL policy from drifting too far from the pretrained VLA behavior, turning online RL into local refinement rather than unconstrained action search.

During training, RLT alternates between online data collection and replay-based actor-critic updates. Therefore, the replay buffer  $\mathcal{B}$  gradually contains three types of data: warmup transitions generated by the base VLA, refined rollouts generated by the RL policy, and optional intervention transitions. This construction allows the actor-critic module to reuse all collected experience off-policy while preserving the behavior prior provided by the frozen VLA. In our reproduction, we follow this overall training structure while adapting the implementation to the LIBERO benchmark and the SmolVLA backbone.

### 3.4 Execution

During execution, RLT uses the frozen VLA policy and the learned RL actor in a closed-loop manner. At each decision step, the current visual observation, language instruction, and proprioceptive state are first passed into the frozen VLA backbone. The VLA produces both the RL-token representation  $z_t^{\text{rl}} = g_\phi(f_{\theta_{\text{VLA}}}(o_t, l, s_t^p))$  and a reference action chunk  $\tilde{\mathbf{a}}_{t:t+C-1} \sim \pi_{\text{VLA}}(\cdot \mid o_t, l, s_t^p)$ . The RL state is then formed as

$$x_t = (z_t^{\text{rl}}, s_t^p), \quad (13)$$

and the trained actor generates a refined action chunk conditioned on both the RL state and the VLA reference action:

$$\mathbf{a}_{t:t+C-1} = \pi_\theta(x_t, \tilde{\mathbf{a}}_{t:t+C-1}). \quad (14)$$

The robot executes this refined chunk for a short horizon before receiving a new observation and replanning. In this way, the policy preserves the high-level semantic and visuomotor priors of the VLA while allowing the RL actor to make task-specific corrections during execution.

Following the official RLT design, execution can also be focused on the precision-critical phase of a task. The base VLA policy handles easier stages such as reaching, grasping, or coarse alignment, while the RL policy is activated during difficult phases such as insertion, fastening, or contact-rich adjustment. This targeted execution strategy avoids unnecessary RL control over the entire task and concentrates the learned policy on the stage where speed and precision matter most. In our

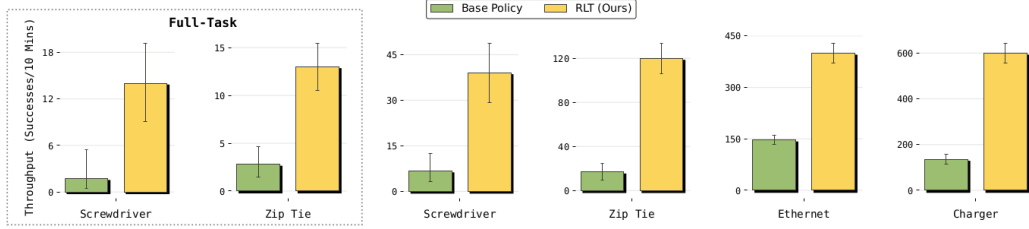


Fig. 4: RLT increases throughput significantly over the base VLA policy, improving both the speed and consistency of the critical phase of each task. The improvement is especially pronounced for the harder tasks where the VLA policy is prone to making mistakes.

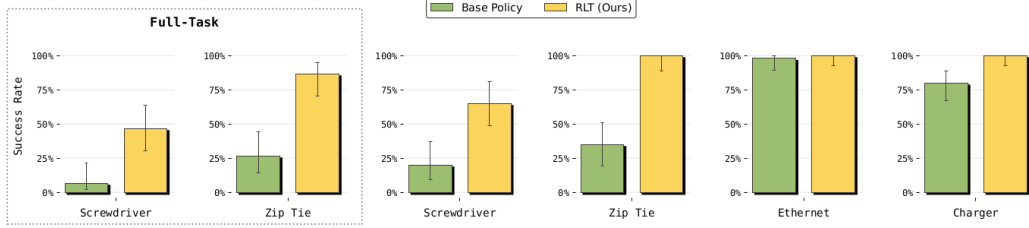


Figure 2: Overview of the Official Experimental Results.

reproduction, we follow the same execution logic on LIBERO tasks, using SmolVLA as the frozen backbone and applying the learned RL policy to refine the VLA-proposed single-step actions in our implementation during task execution.

## 4 Experimentations

### 4.1 Official Experimental Results

As illustrated in Fig. 2, the original RLT paper evaluates the method on four real-world precision manipulation tasks: screw installation, zip tie fastening, Ethernet insertion, and charger insertion. These tasks focus on difficult contact-rich stages such as alignment, insertion, fastening, and rotation, where pretrained VLA policies often become slow or unreliable. The official system uses  $\pi_{0.6}$  as the frozen VLA backbone, which takes multi-view camera observations, language instructions, and proprioceptive states as input and outputs continuous action chunks. After collecting 1-10 hours of task-specific teleoperated demonstrations, the authors train the RL-token interface, freeze the VLA, and perform online actor-critic learning for 400-1000 episodes, corresponding to roughly 15 minutes to 5 hours of actual robot interaction.

The official evaluation considers both critical-phase and full-task settings. In the critical-phase setting, evaluation starts directly before the most difficult stage of the task, isolating the part where online RL is expected to improve performance. In the full-task setting, the base VLA handles easier stages such as reaching and grasping, while the RL policy is activated during the precision-critical phase. The paper compares RLT with the frozen base VLA, HIL-SERL, PLD, DSRL, and Dagger. It also reports ablations that remove the RL token, action chunking, reference-action regularization, or reference-action pass-through.

The results show that RLT improves both task success and execution speed over the base VLA. In critical phases, it increases throughput across all four tasks and achieves up to about  $3\times$  faster execution on easier insertion tasks such as Ethernet and charger. On harder tasks such as screw installation and zip tie fastening, RLT improves both success rate and throughput, and in full-task evaluation it still provides clear gains over the base policy. Compared with other baselines, RLT achieves stronger throughput improvement, while ablation results show that the RL token, chunked action prediction, and reference-action regularization are all important for stable and sample-efficient learning. These official results motivate our reproduction study on LIBERO, where we evaluate whether a SmolVLA-based RLT pipeline can provide similar improvements in a benchmark simulation setting.

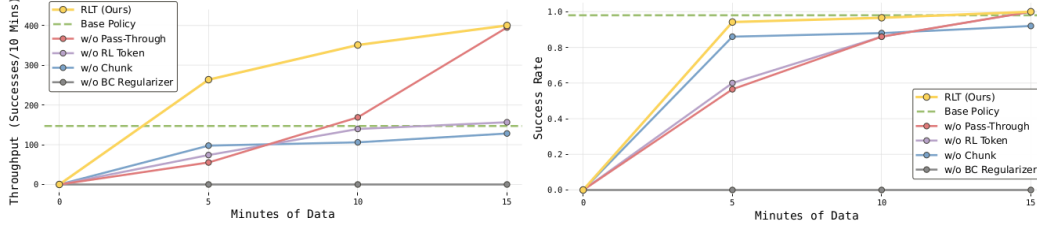


Figure 3: Overview of the Official Ablations.

## 4.2 Official Ablation Studies

The original RLT paper also conducts ablation studies to analyze the contribution of each component in the full RLT framework. These ablations are performed in the official real-robot setting and are not part of our own LIBERO reproduction. We include them here to better understand which design choices are most important in the original method.

The official study considers four ablated variants. First, **w/o RL Token** replaces the learned RL-token representation with a frozen ImageNet-pretrained ResNet-10 encoder. This variant evaluates whether the compact token extracted from the VLA backbone provides more useful task-relevant information than a standard visual encoder. Second, **w/o Chunk** changes the RL policy from chunk-level action prediction to single-step action prediction with  $C = 1$ . This tests the importance of action chunking for reducing the effective decision horizon in long-horizon manipulation. Third, **w/o BC Regularizer** removes the reference-action regularization term by setting  $\beta = 0$ , so the actor is trained only through the Q-function. Fourth, **w/o Pass-Through** removes the VLA reference action  $\tilde{a}$  from the actor input, forcing the RL policy to generate actions only from the state representation.

As illustrated in Fig 3, the official results show that all four components contribute to RLT’s performance. Removing the RL token clearly reduces throughput, indicating that the learned token captures manipulation-relevant structure that is not provided by a generic visual encoder. Removing action chunking makes the task harder because the critic must propagate sparse rewards over a much longer horizon, and the single-step variant cannot reliably match the base policy. The largest degradation comes from removing the BC-style reference-action regularizer. Without this constraint, the actor must explore the full continuous action space using only gradients from the critic, which makes learning unstable and causes the learned policy to drift away from the pretrained VLA behavior.

Among the ablations, removing reference-action pass-through has the smallest effect on final performance, but it still slows down learning and leads to more failures during training. This suggests that the policy can sometimes recover without directly conditioning on the VLA action proposal, especially on simpler tasks, but the reference action remains important for sample-efficient and stable learning. Overall, the official ablation study supports the central design of RLT: the RL token provides an effective state interface, action chunks reduce the effective horizon, and reference-action guidance stabilizes online actor-critic learning around the pretrained VLA policy.

## 4.3 Experimental Setup

Our experiments evaluate whether the core idea of RL Token (RLT) can be transferred to an accessible simulated VLA setting. Specifically, we study whether a frozen, task fine-tuned SmolVLA policy can be further improved by a lightweight reinforcement-learning module that conditions on a compact VLA-derived representation and the VLA reference action. We focus on two questions: (1) whether RLT-style adaptation improves the success rate of a frozen SmolVLA policy on LIBERO tasks, and (2) whether reference-guided actor-critic learning can reduce slow or hesitant behaviors during precision-critical execution Shukor et al. [2025].

Shown in Fig 5, we conduct our experiments on the LIBERO-10 benchmark through the LeRobot platform Liu et al. [2023], Cadene et al. [2026]. The language instructions for all LIBERO-10 tasks are listed in Appendix C. LIBERO provides standardized language-conditioned manipulation tasks, making it suitable for controlled evaluation of policy adaptation in simulation. Shown in Fig 4, We

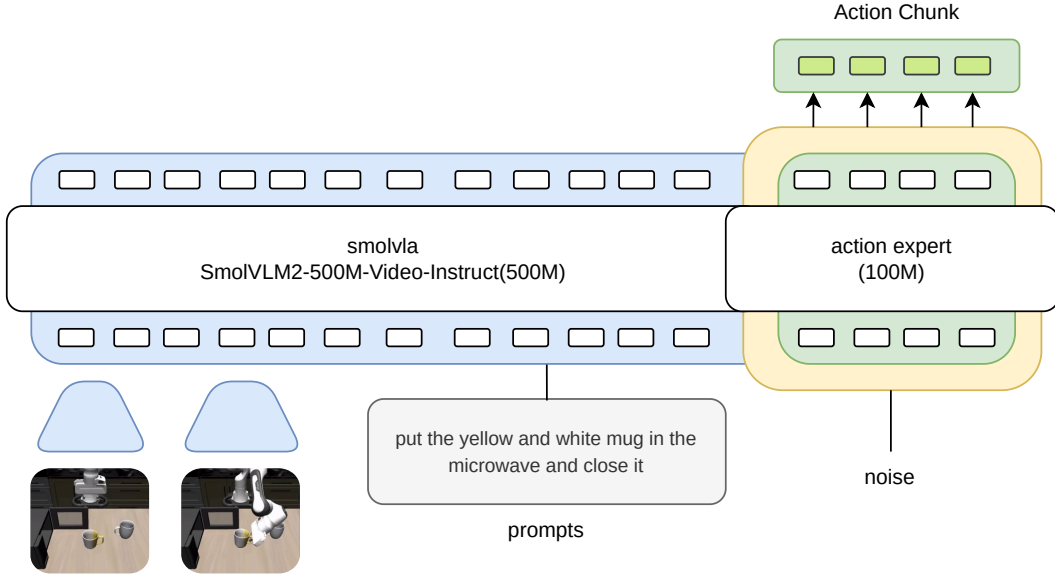


Figure 4: Overview of the SmolVLA.

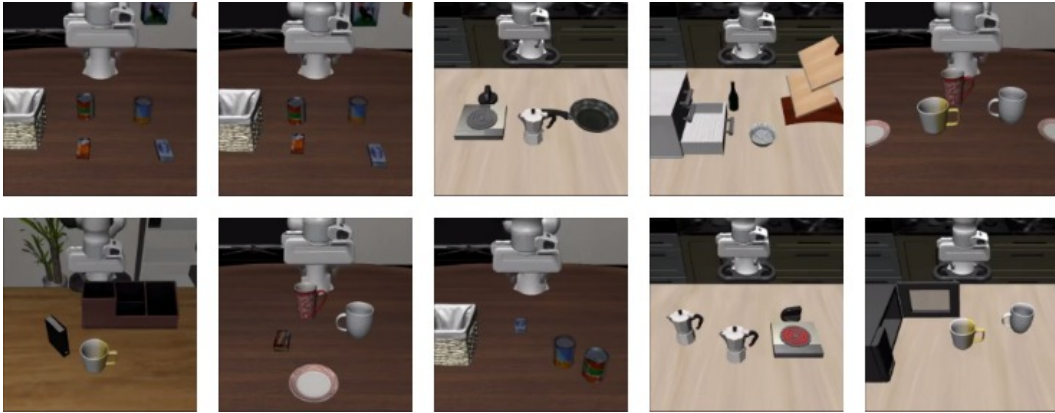


Figure 5: Overview of the LIBERO-10 Benchmark.

use SmolVLA as the backbone policy instead of the original large-scale VLA used in RLT, because SmolVLA is publicly accessible and can be fine-tuned and evaluated on consumer-level hardware. The backbone policy takes visual observations, proprioceptive states, and language instructions as input, and outputs continuous robot actions. All experiments are implemented in LeRobot and run on a single NVIDIA RTX 5070 Ti GPU.

Our implementation preserves the main RLT interface: a frozen VLA backbone, a compact VLA-derived state representation, VLA reference actions, and a lightweight actor-critic module for policy improvement. However, our implementation differs from the original RLT system in three important ways. First, the original method constructs an RL token from full VLA token embeddings, while our SmolVLA interface uses a pooled action-head latent as an action-latent proxy token. Second, the original RLT actor operates over action chunks, whereas our current transition buffer stores single executed 7-dimensional LIBERO actions. Third, the original system performs online real-robot adaptation, while our experiments are conducted in the LIBERO simulator. If additional rollouts are collected using the updated residual policy and appended to the replay buffer, our setting corresponds

to batched online adaptation in simulation; otherwise, the first training round should be interpreted as offline RLT-style residual learning from a fixed replay buffer.

We first collect rollouts using the task fine-tuned SmolVLA policy and construct a transition buffer containing observations, actions, rewards, success labels, and next-state information. We then run the frozen SmolVLA policy on the buffered observations to extract reference actions and compact action-latent features. The residual actor-critic is trained off-policy from this replay buffer. Further implementation details, including the mathematical comparison between the original RLT formulation and our SmolVLA adaptation, are provided in Appendix D. We train the residual actor-critic using the cached action-latent features, frozen SmolVLA reference actions, robot states, executed actions, rewards, and next-state features. The actor predicts a residual correction on top of the frozen VLA reference action, while the critic estimates the value of the executed action under the compact action-latent representation. Unless otherwise specified, the VLA backbone and the feature extractor are kept frozen throughout RL training. The main training hyperparameters are summarized in Appendix E.

For evaluation, each LIBERO-10 task is tested over 50 episodes. Success rate is computed as the fraction of successful episodes according to the LIBERO environment success signal. Each episode is capped at a maximum horizon of 520 environment steps. We also report average episode duration to measure execution efficiency. Episode duration is measured in environment seconds and averaged over all evaluation episodes, including both successful and failed trials. Failed episodes are counted as failures for success-rate computation and are also included in the duration average, typically with longer durations because they often terminate at the 520-step horizon. The full evaluation protocol is described in Appendix E.

Table 1: Per-task success rates on the ten LIBERO-10 tasks.

| Task    | SmolVLA (Fine-tuned) | SmolVLA (RLT, Ours) |
|---------|----------------------|---------------------|
| 10-0    | 16.0%                | 28.0%               |
| 10-1    | 46.0%                | 62.0%               |
| 10-2    | 48.0%                | 56.0%               |
| 10-3    | 84.0%                | 88.0%               |
| 10-4    | 14.0%                | 20.0%               |
| 10-5    | 64.0%                | 74.0%               |
| 10-6    | 66.0%                | 74.0%               |
| 10-7    | 30.0%                | 42.0%               |
| 10-8    | 12.0%                | 24.0%               |
| 10-9    | 44.0%                | 58.0%               |
| Average | 42.4%                | 52.6%               |

Table 2: Per-task average episode duration in seconds on the ten LIBERO-10 tasks. Each episode is capped at 520 environment steps, and failed episodes are included in the average.

| Task    | SmolVLA (Fine-tuned) | SmolVLA (RLT, Ours) |
|---------|----------------------|---------------------|
| 10-0    | 116.6                | 104.5               |
| 10-1    | 97.8                 | 89.4                |
| 10-2    | 89.2                 | 85.1                |
| 10-3    | 69.4                 | 66.7                |
| 10-4    | 114.0                | 110.0               |
| 10-5    | 78.0                 | 72.9                |
| 10-6    | 83.3                 | 76.2                |
| 10-7    | 108.8                | 103.4               |
| 10-8    | 118.4                | 109.6               |
| 10-9    | 100.5                | 91.3                |
| Average | 97.6                 | 90.9                |

#### 4.4 Main Results

Table 1 and Table 2 summarize the main results on LIBERO-10. Compared with the fine-tuned SmolVLA baseline, our RLT-style residual adaptation improves the average success rate from 42.4%

to 52.6%, corresponding to a gain of +10.2 percentage points. It also reduces the average completion time from 97.6s to 90.9s, indicating that the adapted policy completes episodes more efficiently.

These results support the central hypothesis of RLT in our SmolVLA-LIBERO setting: a frozen VLA policy can be improved by training a lightweight actor-critic module on top of compact VLA-derived representations and VLA reference actions. Rather than replacing or fine-tuning the full VLA backbone, our method refines the actions proposed by the VLA while keeping the learned policy close to the supervised VLA prior. This design allows task-specific improvement through reinforcement learning without updating the full VLA model.

The improvement over the fine-tuned SmolVLA baseline indicates that supervised imitation alone does not fully optimize execution quality. In particular, the RLT-style residual policy can correct small action errors, reduce hesitation near contact-rich or precision-critical states, and improve the timing of execution. These effects are reflected in both higher success rates and shorter completion times.

## 5 Discussion and Limitations

Our results suggest that RLT-style adaptation can improve a frozen SmolVLA policy by using the VLA as both a behavioral prior and a representation provider. The improvements in success rate and completion time indicate that supervised fine-tuning alone does not fully optimize task execution. In particular, the residual policy appears useful for correcting small action errors, reducing hesitation near contact-rich states, and improving the timing of precise manipulation actions. This supports the main intuition behind RLT: lightweight RL can refine a strong VLA policy without updating the full backbone. However, our implementation also has several limitations. First, our token is not the same as the original RLT token. The original method constructs an RL token from full VLA token embeddings, while our implementation uses a pooled SmolVLA action-head latent as an action-latent proxy. This representation is compact and efficient, but it may discard visual and semantic information needed for correcting high-level perception or planning errors. As a result, our method is likely better suited for local action refinement than for recovering from wrong object selection or long-horizon reasoning failures. Second, our current actor operates on single executed LIBERO actions rather than action chunks. This differs from the original RLT formulation, where chunked actions reduce the effective horizon for sparse-reward RL. The single-step formulation is simpler and compatible with our collected transition buffer, but it may make credit assignment harder and limit improvement on long-horizon tasks. Third, our experiments are conducted in simulation and use a limited amount of rollout data. The initial dataset contains uneven success coverage across tasks, so low-success tasks may provide insufficient positive examples for stable critic learning. A closer reproduction would require repeated collect-train cycles using the updated residual policy. Due to computational constraints, we do not include additional ablations over the latent representation, reference-action regularization, or residual actor design; these remain important directions for future validation.

## 6 Conclusions

We presented a reproduction study of RL Token-based online reinforcement learning for improving frozen VLA policies in robotic manipulation. By using SmolVLA as the backbone and replacing the original RL-token representation with a compact action-latent proxy, our reproduced framework preserves the main reference-guided actor-critic adaptation idea of RLT while remaining feasible under limited computational resources. Experiments on LIBERO-10 compare our method against a task fine-tuned SmolVLA baseline in task success rate and average episode duration. These results and official experiments both suggest that lightweight online RL on top of pretrained VLA representations is an effective strategy for improving execution efficiency and reliability. Future work will explore stronger VLA backbones, broader real-world tasks, and more autonomous reward and intervention mechanisms.

## Acknowledgments and Disclosure of Funding

We thank the course instructors and teaching assistants for their helpful guidance and feedback. We are grateful to the authors of RL Token for their original contribution, which served as the basis of this reproduction study. We also acknowledge the developers of LIBERO, LeRobot, and SmolVLA for providing the benchmark, implementation framework, and model resources used in our experiments. All experiments were conducted using local computational resources.

## References

- Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky.  $\pi_0$ : A vision-language-action flow model for general robot control, 2026. URL <https://arxiv.org/abs/2410.24164>.
- Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023. URL <https://arxiv.org/abs/2307.15818>.
- Remi Cadene, Simon Alibert, Francesco Capuano, Michel Aractingi, Adil Zouitine, Pepijn Kooijmans, Jade Choghari, Martino Russi, Caroline Pascal, Steven Palma, Mustafa Shukor, Jess Moss, Alexander Soare, Dana Aubakirova, Quentin Lhoest, Quentin Gallouédec, and Thomas Wolf. Lerobot: An open-source library for end-to-end robot learning, 2026. URL <https://arxiv.org/abs/2602.22818>.
- 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, 2025. URL <https://arxiv.org/abs/2502.05450>.
- 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, 2024. URL <https://arxiv.org/abs/2303.04137>.
- Embodiment Collaboration, Abby O’Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlikar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Felipe Vieira Fruejri, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guangwen Yang, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homanga Bharadhwaj, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jay Vakil, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Boher, Jonathan

- Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrrel Pinto, Li Fei-Fei, Liam Tan, Linxi "Jim" Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minh Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Muhammad Zubair Irshad, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick "Tree" Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundareshan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Rohan Bajjal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shubham Tulsiani, Shuran Song, Sichun Xu, Siddhant Halder, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vikash Kumar, Vincent Vanhoucke, Vitor Guizilini, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiangyu Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yansong Pang, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yongqiang Dou, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, Zipeng Fu, and Zipeng Lin. Open x-embodiment: Robotic learning datasets and rt-x models, 2025. URL <https://arxiv.org/abs/2310.08864>.
- Muhyay Ud Din, Waseem Akram, Lyes Saad Saoud, Jan Rosell, and Irfan Hussain. Vision language action models in robotic manipulation: A systematic review, 2025. URL <https://arxiv.org/abs/2507.10672>.
- Yang Gao, Huazhe Xu, Ji Lin, Fisher Yu, Sergey Levine, and Trevor Darrell. Reinforcement learning from imperfect demonstrations, 2019. URL <https://arxiv.org/abs/1802.05313>.
- Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates, 2016. URL <https://arxiv.org/abs/1610.00633>.
- Weifan Guan, Qinghao Hu, Aosheng Li, and Jian Cheng. Efficient vision-language-action models for embodied manipulation: A systematic survey, 2025. URL <https://arxiv.org/abs/2510.17111>.
- Yanjiang Guo, Jianke Zhang, Xiaoyu Chen, Xiang Ji, Yen-Jen Wang, Yucheng Hu, and Jianyu Chen. Improving vision-language-action model with online reinforcement learning, 2025. URL <https://arxiv.org/abs/2501.16664>.
- Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL <https://arxiv.org/abs/1801.01290>.
- Yuzhe Huang, Pei Lin, Wanlin Li, Daohan Li, Jiajun Li, Jiaming Jiang, Chenxi Xiao, and Ziyuan Jiao. Taf-vla: Tactile-force alignment in vision-language-action models for force-aware manipulation, 2026. URL <https://arxiv.org/abs/2601.20321>.
- Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z.

- Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky.  $\pi_{0.5}$ : a vision-language-action model with open-world generalization, 2025. URL <https://arxiv.org/abs/2504.16054>.
- Yuhua Jiang, Junjie Lu, Xinyao Qin, Xiaoyu Chen, Kaixin Wang, Feifei Gao, and Li Zhao. Vla-gse: Boosting parameter-efficient fine-tuning in vla with generalized and specialized experts, 2026. URL <https://arxiv.org/abs/2605.06175>.
- Mingxuan Jing, Xiaojian Ma, Wenbing Huang, Fuchun Sun, Chao Yang, Bin Fang, and Huaping Liu. Reinforcement learning from imperfect demonstrations under soft expert guidance, 2019. URL <https://arxiv.org/abs/1911.07109>.
- Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation, 2018. URL <https://arxiv.org/abs/1806.10293>.
- 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, 2024. URL <https://arxiv.org/abs/2406.09246>.
- Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success, 2025. URL <https://arxiv.org/abs/2502.19645>.
- Kun Lei, Huanyu Li, Dongjie Yu, Zhenyu Wei, Lingxiao Guo, Zhennan Jiang, Ziyu Wang, Shiyu Liang, and Huazhe Xu. RL-100: Performant robotic manipulation with real-world reinforcement learning, 2026. URL <https://arxiv.org/abs/2510.14830>.
- Hengtao Li, Pengxiang Ding, Runze Suo, Yihao Wang, Zirui Ge, Dongyuan Zang, Kexian Yu, Mingyang Sun, Hongyin Zhang, Donglin Wang, and Weihua Su. Vla-rft: Vision-language-action reinforcement fine-tuning with verified rewards in world simulators, 2025. URL <https://arxiv.org/abs/2510.00406>.
- Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, Hang Li, and Tao Kong. Vision-language foundation models as effective robot imitators, 2024. URL <https://arxiv.org/abs/2311.01378>.
- Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning, 2023. URL <https://arxiv.org/abs/2306.03310>.
- Jiaming Liu, Mengzhen Liu, Zhenyu Wang, Pengju An, Xiaoqi Li, Kaichen Zhou, Senqiao Yang, Renrui Zhang, Yandong Guo, and Shanghang Zhang. Robomamba: Efficient vision-language-action model for robotic reasoning and manipulation, 2024. URL <https://arxiv.org/abs/2406.04339>.
- Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, Yi Wu, Chao Yu, and Yu Wang. What can rl bring to vla generalization? an empirical study, 2026. URL <https://arxiv.org/abs/2505.19789>.
- Chengxuan Lu, Shukuan Wang, Yanjie Li, Wei Liu, Shiji Jin, Fuyuan Qian, Peiming Li, Baigui Sun, and Yang Liu. Accerl: A distributed asynchronous reinforcement learning and world model framework for vision-language-action models, 2026. URL <https://arxiv.org/abs/2603.18464>.
- Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning, 2025. URL <https://arxiv.org/abs/2505.18719>.

- Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning, 2025. URL <https://arxiv.org/abs/2410.21845>.
- Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision-language-action models for embodied ai, 2026. URL <https://arxiv.org/abs/2405.14093>.
- Xinpan Meng, Siyao Huang, JingPu Yang, Muyuan Ma, Zhenghua Ma, Lijun Han, Gao Yuan, Houcheng Li, and Long Cheng. From reach to insert: Tactile-augmented precision assembly under sub-millimeter tolerances, 2026. URL <https://arxiv.org/abs/2605.04649>.
- Chancharik Mitra, Yusen Luo, Raj Saravanan, Dantong Niu, Anirudh Pai, Jesse Thomason, Trevor Darrell, Abrar Anwar, Deva Ramanan, and Roei Herzig. Mechanistic finetuning of vision-language-action models via few-shot demonstrations, 2025. URL <https://arxiv.org/abs/2511.22697>.
- Angelo Moroncelli, Roberto Zanetti, Marco Maccarini, and Loris Roveda. Jump-start reinforcement learning with vision-language-action regularization, 2026. URL <https://arxiv.org/abs/2604.13733>.
- Cyrus Neary, Omar G. Younis, Artur Kuramshin, Ozgur Aslan, and Glen Berseth. Improving pre-trained vision-language-action policies with model-based search, 2025. URL <https://arxiv.org/abs/2508.12211>.
- Abdullah Yahya Abdullah Omaisan and Ibrahim Sheikh Mohamed. Towards accessible physical ai: Lora-based fine-tuning of vla models for real-world robot control, 2025. URL <https://arxiv.org/abs/2512.11921>.
- Mingjie Pan, Siyuan Feng, Qinglin Zhang, Xinchun Li, Jianheng Song, Chendi Qu, Yi Wang, Chuankang Li, Ziyu Xiong, Zhi Chen, Yi Liu, and Jianlan Luo. Sop: A scalable online post-training system for vision-language-action models, 2026. URL <https://arxiv.org/abs/2601.03044>.
- Xiaohuan Pei, Yuxing Chen, Siyu Xu, Yunke Wang, Yuheng Shi, and Chang Xu. Action-aware dynamic pruning for efficient vision-language-action manipulation, 2025. URL <https://arxiv.org/abs/2509.22093>.
- Ranjan Sapkota, Yang Cao, Konstantinos I. Roumeliotis, and Manoj Karkee. Vision-language-action (vla) models: Concepts, progress, applications and challenges, 2026. URL <https://arxiv.org/abs/2505.04769>.
- Rui Shao, Wei Li, Lingsen Zhang, Renshan Zhang, Zhiyang Liu, Ran Chen, and Liqiang Nie. Large vlm-based vision-language-action models for robotic manipulation: A survey, 2025. URL <https://arxiv.org/abs/2508.13073>.
- Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. Smolvla: A vision-language-action model for affordable and efficient robotics, 2025. URL <https://arxiv.org/abs/2506.01844>.
- Jannick Stranghöner, Philipp Hartmann, Marco Braun, Sebastian Wrede, and Klaus Neumann. Share-rl: Structured, interactive reinforcement learning for contact-rich industrial assembly tasks, 2026. URL <https://arxiv.org/abs/2509.13949>.
- Shuhan Tan, Kairan Dou, Yue Zhao, and Philipp Krähenbühl. Interactive post-training for vision-language-action models, 2025. URL <https://arxiv.org/abs/2505.17016>.
- Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy, 2024. URL <https://arxiv.org/abs/2405.12213>.
- Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards, 2018. URL <https://arxiv.org/abs/1707.08817>.

- Hanzhen Wang, Jiaming Xu, Yushun Xiang, Jiayi Pan, Yongkang Zhou, Yong-Lu Li, and Guohao Dai. Specprune-vla: Accelerating vision-language-action models via action-aware self-speculative pruning, 2026. URL <https://arxiv.org/abs/2509.05614>.
- Yuqi Wang, Xinghang Li, Wenxuan Wang, Junbo Zhang, Yingyan Li, Yuntao Chen, Xinlong Wang, and Zhaoxiang Zhang. Unified vision-language-action model, 2025. URL <https://arxiv.org/abs/2506.19850>.
- Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, Feifei Feng, and Jian Tang. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation, 2025. URL <https://arxiv.org/abs/2409.12514>.
- Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation, 2023. URL <https://arxiv.org/abs/2312.13139>.
- Charles Xu, Jost Tobias Springenberg, Michael Equi, Ali Amin, Adnan Esmail, Sergey Levine, and Liyiming Ke. RL token: Bootstrapping online rl with vision-language-action models, 2026. URL <https://arxiv.org/abs/2604.23073>.
- Rushuai Yang, Hangxing Wei, Ran Zhang, Zhiyuan Feng, Xiaoyu Chen, Tong Li, Chuheng Zhang, Li Zhao, Jiang Bian, Xiu Su, and Yi Chen. Beyond human demonstrations: Diffusion-based reinforcement learning to generate data for vla training, 2025. URL <https://arxiv.org/abs/2509.19752>.
- Guo Ye, Zexi Zhang, Xu Zhao, Shang Wu, Haoran Lu, Shihan Lu, and Han Liu. Learning to feel the future: Dreamtacvla for contact-rich manipulation, 2026. URL <https://arxiv.org/abs/2512.23864>.
- Dongjie Yu, Kun Lei, Zhennan Jiang, Jia Pan, and Huazhe Xu. Beyond action residuals: Real-world robot policy steering via bottleneck latent reinforcement learning, 2026. URL <https://arxiv.org/abs/2605.19919>.
- Jiawen Yu, Hairuo Liu, Qiaojun Yu, Jieji Ren, Ce Hao, Haitong Ding, Guangyu Huang, Guofan Huang, Yan Song, Panpan Cai, Cewu Lu, and Wenqiang Zhang. Forcevla: Enhancing vla models with a force-aware moe for contact-rich manipulation, 2025. URL <https://arxiv.org/abs/2505.22159>.
- Dapeng Zhang, Jing Sun, Chenghui Hu, Xiaoyan Wu, Zhenlong Yuan, Rui Zhou, Fei Shen, and Qingguo Zhou. Pure vision language action (vla) models: A comprehensive survey, 2025. URL <https://arxiv.org/abs/2509.19012>.
- Yike Zhang, Yaonan Wang, Xinxin Sun, Kaizhen Huang, Zhiyuan Xu, Junjie Ji, Zhengping Che, Jian Tang, and Jingtao Sun. Craft: Adapting vla models to contact-rich manipulation via force-aware curriculum fine-tuning, 2026. URL <https://arxiv.org/abs/2602.12532>.
- Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, Ankur Handa, Ming-Yu Liu, Donglai Xiang, Gordon Wetzstein, and Tsung-Yi Lin. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models, 2025. URL <https://arxiv.org/abs/2503.22020>.
- Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. URL <https://arxiv.org/abs/2304.13705>.
- Yifan Zhong, Fengshuo Bai, Shaofei Cai, Xuchuan Huang, Zhang Chen, Xiaowei Zhang, Yuanfei Wang, Shaoyang Guo, Tianrui Guan, Ka Nam Lui, Zhiquan Qi, Yitao Liang, Yuanpei Chen, and Yaodong Yang. A survey on vision-language-action models: An action tokenization perspective, 2025. URL <https://arxiv.org/abs/2507.01925>.
- Yuke Zhu, Ziyu Wang, Josh Merel, Andrei Rusu, Tom Erez, Serkan Cabi, Saran Tunyasuvunakool, János Kramár, Raia Hadsell, Nando de Freitas, and Nicolas Heess. Reinforcement and imitation learning for diverse visuomotor skills, 2018. URL <https://arxiv.org/abs/1802.09564>.

## A Open-Source Artifacts and Supplementary Materials

To further support reproducibility and help readers better understand the project, we provide the related open-source artifacts and supplementary materials in a Hugging Face repository:

[https://huggingface.co/Joeyfully/smolvla\\_rlt\\_libero\\_10](https://huggingface.co/Joeyfully/smolvla_rlt_libero_10)

This repository contains the model-related artifacts for our SmolVLA-RLT LIBERO-10 project, including the trained residual policy components and supporting materials for inspecting the implementation. In addition, the repository includes the project presentation slides, which provide a more accessible overview of the motivation, system design, implementation pipeline, experimental setup, result analysis, and additional qualitative observations.

The slides also contain several case-study videos that are not included in the main paper due to space and formatting constraints. These videos illustrate different closed-loop behaviors, including cases where the frozen SmolVLA policy fails while the RLT-style residual policy succeeds, cases where both policies fail but the residual policy makes more task progress, and cases where the residual branch locally corrects the base policy but cannot fully override an incorrect high-level intention.

While the main paper focuses on the technical method and quantitative results, the Hugging Face repository provides additional implementation artifacts, qualitative demonstrations, and open-source evidence for the reported system.

## B Official RLT Algorithms

---

### Algorithm 1 Training Procedure of RLT

---

**Require:** Pretrained VLA  $\pi_{\text{VLA}}$ , demo dataset  $\mathcal{D}$ , replay buffer  $\mathcal{B}$

**Require:** Chunk length  $C$ , warmup steps  $N_{\text{warm}}$ , update ratio  $G$

**Require:** Regularization weight  $\beta$

**Ensure:** RL token module  $\phi$ , actor  $\pi_\theta$ , critic  $Q_\psi$

```
1: /* Stage I: RL Token Adaptation */
2: for iterations do
3:   Sample demonstrations from  $\mathcal{D}$ 
4:   Extract VLA embeddings  $z_{1:M}$ 
5:   Obtain RL token  $z^{\text{rl}} \leftarrow g_\phi([z_{1:M}, e_{\text{rl}}])_{M+1}$ 
6:   Train  $\phi$  with reconstruction loss  $\mathcal{L}_{\text{ro}}$ 
7: end for
8: Freeze  $\pi_{\text{VLA}}$  and  $\phi$ 
9: /* Stage II: Online RL Fine-Tuning */
10: Initialize actor  $\pi_\theta$  and critic  $Q_\psi$ 
11: Fill  $\mathcal{B}$  with  $N_{\text{warm}}$  VLA rollouts
12: for environment steps do
13:   Observe state  $s_t$  and form  $x_t = (z^{\text{rl}}(s_t), s_t^p)$ 
14:   Sample VLA reference chunk  $\tilde{\mathbf{a}}_{t:t+C-1} \sim \pi_{\text{VLA}}(\cdot|s_t, l)$ 
15:   Sample refined chunk  $\mathbf{a}_{t:t+C-1} \sim \pi_\theta(\cdot|x_t, \tilde{\mathbf{a}}_{t:t+C-1})$ 
16:   Execute  $\mathbf{a}_{t:t+C-1}$  and store transition in  $\mathcal{B}$ 
17:   for  $g = 1$  to  $G$  do
18:     Sample batch  $(x, \mathbf{a}, \tilde{\mathbf{a}}, r, x')$  from  $\mathcal{B}$ 
19:     Update critic with TD loss  $\mathcal{L}_Q$ 
20:     Update actor with

$$\mathcal{L}_\pi = -Q_\psi(x, \mathbf{a}) + \beta \|\mathbf{a} - \tilde{\mathbf{a}}\|_2^2$$

21:   end for
22: end for
```

---

---

**Algorithm 2** Execution Procedure of RLT

---

**Require:** Frozen VLA policy  $\pi_{\text{VLA}}$ , trained RL token module  $g_\phi$ , trained actor  $\pi_\theta$

**Require:** Language instruction  $l$ , chunk length  $C$ , maximum steps  $T$

**Require:** Critical-phase indicator  $\text{IsCritical}(\cdot)$

**Ensure:** Executed robot trajectory

- 1: Initialize environment and observe initial state  $s_0$
  - 2: **for**  $t = 0$  to  $T$  with step size  $C$  **do**
  - 3:   Observe visual input  $o_t$  and proprioceptive state  $s_t^p$
  - 4:   Extract VLA embeddings:
$$z_{1:M} \leftarrow f_{\theta_{\text{VLA}}}(o_t, l, s_t^p)$$
  - 5:   Obtain RL token:
$$z_t^{\text{rl}} \leftarrow g_\phi([z_{1:M}, e_{\text{rl}}])_{M+1}$$
  - 6:   Form RL state:
$$x_t \leftarrow (z_t^{\text{rl}}, s_t^p)$$
  - 7:   Sample VLA reference action chunk:
$$\tilde{\mathbf{a}}_{t:t+C-1} \sim \pi_{\text{VLA}}(\cdot \mid o_t, l, s_t^p)$$
  - 8:   **if**  $\text{IsCritical}(s_t)$  **then**
  - 9:     Refine the VLA action chunk with the RL actor:
$$\mathbf{a}_{t:t+C-1} \leftarrow \pi_\theta(x_t, \tilde{\mathbf{a}}_{t:t+C-1})$$
  - 10:   **else**
  - 11:     Use the base VLA action chunk:
$$\mathbf{a}_{t:t+C-1} \leftarrow \tilde{\mathbf{a}}_{t:t+C-1}$$
  - 12:   **end if**
  - 13:   Execute  $\mathbf{a}_{t:t+C-1}$  in the environment
  - 14:   Receive new observation  $s_{t+C}$
  - 15:   **if** task succeeds or fails **then**
  - 16:     **break**
  - 17:   **end if**
  - 18: **end for**
- 

## C LIBERO-10 Task Instructions

Table 3 lists the language instructions used for the LIBERO-10 evaluation tasks. These instructions are used both by the frozen SmolVLA policy and by our RLT-style feature extraction pipeline to condition the policy on the target manipulation task.

Table 3: LIBERO-10 task instructions used in our experiments.

| Task ID | Language instruction                                                                    |
|---------|-----------------------------------------------------------------------------------------|
| 0       | put both the alphabet soup and the tomato sauce in the basket                           |
| 1       | put both the cream cheese box and the butter in the basket                              |
| 2       | turn on the stove and put the moka pot on it                                            |
| 3       | put the black bowl in the bottom drawer of the cabinet and close it                     |
| 4       | put the white mug on the left plate and put the yellow and white mug on the right plate |
| 5       | pick up the book and place it in the back compartment of the caddy                      |
| 6       | put the white mug on the plate and put the chocolate pudding to the right of the plate  |
| 7       | put both the alphabet soup and the cream cheese box in the basket                       |
| 8       | put both moka pots on the stove                                                         |
| 9       | put the yellow and white mug in the microwave and close it                              |

## D Implementation Details of the RLT-style SmolVLA Adaptation

This appendix describes the mathematical differences between the original RLT formulation and our SmolVLA-based implementation.

In the original RLT formulation, the VLA produces final-layer token embeddings  $z_{1:M}$ , illustrated in the original paper as  $N \times 2048$  VLA/image embeddings. RLT appends a learned special token  $e_{\text{rl}}$  to this sequence and processes the augmented sequence with a lightweight encoder transformer:

$$z_{\text{rl}} = g_\phi([z_{1:M}, e_{\text{rl}}])_{M+1}.$$

A decoder transformer is then trained to reconstruct the original VLA embeddings from  $z_{\text{rl}}$ :

$$\mathcal{L}_{\text{ro}} = \mathbb{E}_{\mathcal{D}} \left[ \sum_{i=1}^M \|h_\phi(d_\phi([z_{\text{rl}}, \bar{z}_{1:i-1}]))_i - \bar{z}_i\|_2^2 \right],$$

where  $\bar{z}_i = \text{sg}(z_i)$  denotes a stop-gradient VLA embedding. The original method may optionally combine this stage with supervised VLA fine-tuning. After this stage, both the VLA and the RL-token module are frozen, and the lightweight actor-critic operates on the extracted RL token.

Our SmolVLA implementation does not expose the same full VLA token sequence used in the original RLT system. We therefore extract a compact action-head latent representation from the frozen SmolVLA policy:

$$h_t = \text{Pool}(H_t^{\text{act}}) \in \mathbb{R}^{480},$$

where  $H_t^{\text{act}}$  denotes the hidden representation immediately before the SmolVLA action output projection. In our implementation, this representation is obtained by hooking the input to the SmolVLA action output projection and applying pooling. We use this vector directly as a pseudo RL token, or, in the learned-token variant, as the input to a lightweight bottleneck autoencoder:

$$z_t = E_\phi(h_t), \quad \hat{h}_t = D_\phi(z_t),$$

with reconstruction loss

$$\mathcal{L}_{\text{rec}} = \|D_\phi(E_\phi(h_t)) - h_t\|_2^2.$$

Thus, our token should be interpreted as an action-latent proxy for the original RLT token, rather than a full VLA image-token or multimodal-token representation.

The original RLT actor outputs a Gaussian distribution over action chunks and is conditioned on the VLA reference chunk  $\tilde{a}_{1:C}$ :

$$\pi_\theta(a_{1:C} \mid x, \tilde{a}_{1:C}) = \mathcal{N}(\mu_\theta(x, \tilde{a}_{1:C}), \sigma^2 I),$$

where  $x = (z_{\text{rl}}, s^p)$  combines the RL token and proprioceptive state. The actor is trained to maximize the critic while staying close to the VLA reference chunk:

$$\mathcal{L}_\pi(\theta) = \mathbb{E} [-Q_\psi(x, a_{1:C}) + \beta \|a_{1:C} - \tilde{a}_{1:C}\|_2^2].$$

The original method additionally applies reference-action dropout during actor training, replacing the reference chunk with zeros for a subset of samples to prevent the actor from merely copying the VLA proposal.

Our current transition buffer stores single executed LIBERO actions rather than  $C$ -step action chunks. We therefore instantiate the same reference-guided refinement principle as an explicit residual policy over the executed 7-dimensional action:

$$a_t = \text{clip}(a_t^{\text{ref}} + \Delta_\theta(z_t, s_t, a_t^{\text{ref}})).$$

This makes our method a single-step residual approximation of the original chunked RLT actor.

The original RLT system alternates real-robot data collection and off-policy actor-critic updates. Our experiments perform this idea in the LIBERO simulator. We first collect a replay buffer using the fine-tuned SmolVLA policy:

$$\mathcal{D}_0 = \text{Collect}(\pi_{\text{VLA}}).$$

We then train the actor and critic off-policy:

$$\theta_k, \psi_k = \text{Train} \left( \bigcup_{i=0}^{k-1} \mathcal{D}_i \right).$$

The resulting policy is

$$\pi_k(o_t) = \text{clip}(a_t^{\text{ref}} + \Delta_{\theta_k}(z_t, s_t, a_t^{\text{ref}})).$$

If the updated policy is deployed to collect additional rollouts,

$$\mathcal{D}_k = \text{Collect}(\pi_k),$$

the procedure becomes batched online adaptation in simulation. If only  $\mathcal{D}_0$  is used, the experiment should be interpreted as offline RLT-style residual learning.

## E Implementation, Training, and Evaluation Details

This appendix provides additional implementation details for our SmolVLA-based RLT-style adaptation pipeline, including replay-buffer statistics, feature-cache statistics, actor-critic training configuration, and the evaluation protocol.

### E.1 Dataset and Feature Cache Statistics

Table 4 summarizes the replay buffer and feature cache used in our experiments. We first collect rollouts from the task fine-tuned SmolVLA policy on LIBERO-10. The collected rollouts are converted into transition shards, and the frozen SmolVLA policy is then used to extract reference actions and compact action-latent features for each transition. Each feature transition stores the executed action, the frozen SmolVLA reference action, robot state, reward, success signal, next-state information, and a 480-dimensional pooled action-head latent feature. The final feature cache occupies 434 MB across 307 shards.

Table 4: Replay buffer and feature cache statistics.

| Statistic                     | Value                             |
|-------------------------------|-----------------------------------|
| Benchmark suite               | LIBERO-10                         |
| Number of tasks               | 10                                |
| Rollout episodes              | 500                               |
| Successful episodes           | 212                               |
| Total frames                  | 209,612                           |
| Total transitions             | 209,112                           |
| Overall baseline success rate | 42.4%                             |
| Action dimension              | 7                                 |
| State dimension               | 16                                |
| Feature dimension             | 480                               |
| Feature dtype                 | bfloat16                          |
| Feature source                | pooled SmolVLA action-head latent |
| Pooling method                | mean pooling                      |
| Saved next features           | yes                               |
| Number of feature shards      | 307                               |
| Feature cache size            | 434 MB                            |

Table 5 reports the per-task rollout statistics. The dataset is not uniformly balanced in terms of successful episodes: some tasks, such as task 3, task 5, and task 6, have relatively high baseline success rates, while task 0, task 4, and task 8 contain fewer successful rollouts. This imbalance is important for interpreting the learning results, since low-success tasks provide fewer positive examples for critic learning.

### E.2 Actor-Critic Training Details

Table 6 summarizes the actor-critic training configuration. Since the cached features are compact, actor-critic training is lightweight compared with VLA inference or feature extraction. During this stage, the SmolVLA backbone and feature extractor are kept frozen. The actor receives the compact action-latent feature, robot state, and frozen SmolVLA reference action as input, and predicts a bounded residual correction. The critic receives the compact action-latent feature, robot state, and an action, and predicts the corresponding action value.

Table 5: Per-task rollout statistics for LIBERO-10.

| Task ID | Transitions | Success Episodes | Success Rate |
|---------|-------------|------------------|--------------|
| 0       | 24,694      | 8 / 50           | 16.0%        |
| 1       | 20,985      | 23 / 50          | 46.0%        |
| 2       | 19,383      | 24 / 50          | 48.0%        |
| 3       | 14,719      | 42 / 50          | 84.0%        |
| 4       | 24,396      | 7 / 50           | 14.0%        |
| 5       | 16,717      | 32 / 50          | 64.0%        |
| 6       | 17,761      | 33 / 50          | 66.0%        |
| 7       | 23,467      | 15 / 50          | 30.0%        |
| 8       | 25,526      | 6 / 50           | 12.0%        |
| 9       | 21,464      | 22 / 50          | 44.0%        |

Table 6: Actor-critic training configuration.

| Hyperparameter       | Value                             |
|----------------------|-----------------------------------|
| RL algorithm         | offline residual actor-critic     |
| Actor input          | $z_t, s_t, a_t^{\text{ref}}$      |
| Critic input         | $z_t, s_t, a_t$                   |
| Token representation | pooled SmolVLA action-head latent |
| Token dimension      | 480                               |
| State dimension      | 16                                |
| Action dimension     | 7                                 |
| Optimizer            | Adam                              |
| Actor learning rate  | $3 \times 10^{-4}$                |
| Critic learning rate | $3 \times 10^{-4}$                |
| Replay buffer size   | 209,112 transitions               |
| VLA backbone         | frozen                            |
| Feature extractor    | frozen                            |
| Training device      | NVIDIA RTX 5070 Ti                |

The actor is trained as a reference-guided residual policy. Given the frozen SmolVLA reference action  $a_t^{\text{ref}}$ , the actor predicts a residual correction  $\Delta_\theta(z_t, s_t, a_t^{\text{ref}})$ . The final policy action is computed as

$$a_t = \text{clip} \left( a_t^{\text{ref}} + \Delta_\theta(z_t, s_t, a_t^{\text{ref}}) \right),$$

where clipping is applied to keep the action within the valid action range. The residual clipping range limits the magnitude of the learned correction and prevents the adapted policy from drifting too far from the frozen VLA proposal.

The actor objective follows the reference-guided refinement principle used in RLT. In addition to maximizing the critic value, we regularize the learned action toward the behavior action and penalize overly large residual corrections:

$$\mathcal{L}_\pi = \mathbb{E} \left[ -Q_\psi(z_t, s_t, a_t) + \beta \|a_t - a_t^{\text{data}}\|_2^2 + \lambda_\Delta \|\Delta_\theta(z_t, s_t, a_t^{\text{ref}})\|_2^2 \right],$$

where  $a_t^{\text{data}}$  is the action stored in the replay buffer and  $\lambda_\Delta = 0.05$  is the residual regularization weight. This design preserves the supervised VLA prior while allowing reinforcement learning to correct local action errors.

### E.3 Evaluation Protocol

Each LIBERO-10 task is evaluated over 50 episodes. We use the task success signal returned by the LIBERO environment to compute binary episode success. For task  $i$ , the success rate is computed as

$$\text{SuccessRate}_i = \frac{N_i^{\text{success}}}{N_i^{\text{eval}}},$$

where  $N_i^{\text{eval}} = 50$  for each task. The overall success rate is computed as the average success rate across all ten LIBERO-10 tasks:

$$\text{SuccessRate}_{\text{avg}} = \frac{1}{10} \sum_{i=0}^9 \text{SuccessRate}_i.$$

We also report average episode duration as a measure of execution efficiency. Each evaluation episode is capped at a maximum horizon of 520 environment steps. An episode terminates early if the task succeeds; otherwise, failed episodes may run until the 520-step limit. Therefore, the duration metric is averaged over all evaluation episodes, including both successful and failed trials:

$$T_i = \frac{1}{N_i^{\text{eval}}} \sum_{j=1}^{N_i^{\text{eval}}} T_{i,j}.$$

Here,  $T_{i,j}$  denotes the recorded duration of episode  $j$  for task  $i$ . Failed episodes are counted in the success-rate calculation and are also included in the duration average. Since failed episodes often terminate only at the maximum horizon, this metric reflects both execution speed and failure behavior. The average duration is computed by averaging the per-task durations across all ten LIBERO-10 tasks.

During evaluation, the SmolVLA backbone remains frozen and no actor–critic updates are performed. The baseline SmolVLA policy is evaluated under the same LIBERO task definitions, trial count, and 520-step episode horizon as our RLT-style residual policy. This ensures that differences in success rate and average duration are attributable to the residual adaptation module rather than to changes in the backbone VLA model or evaluation protocol.

## F Rebuttal

No additional changes are required because the comments are based on incorrect premises or have already been addressed.

Figure 6 illustrates the reviewers’ comments on the paper. Below is our rebuttal.

Reviewer **Xingyi Liu**:

### Comment 1:

Thanks for the reviewer’s advice. We respectfully disagree that the NeurIPS 9-page limit should be applied as a strict evaluation criterion for this course project unless explicitly required by the course. Our paper documents a adaptive RLT reproduction. These details are necessary for reproducibility and technical clarity. The length itself should not be considered a technical weakness.

### Comment 2:

The citation style used in the paper follows a common NeurIPS convention. Unless the course specifies a mandatory citation format, this should be considered a minor formatting preference rather than a scientific limitation.

### Comment 3:

The introduction is only slightly longer than one page, which is within the normal range for a technical paper. Therefore, we do not consider its length to be excessive. The background is necessary because the project combines important issues of VLA and a brief introduction to the architecture. Removing this context would make the paper less accessible rather than more concise.

Reviewer **Siyuan Wu**:

### Comment 1:

Thanks for the reviewer’s advice. But this comment is based on a misunderstanding of our evaluation protocol. LeRobot evaluation uses seeded environment rollouts, and different evaluation episodes correspond to different environment initializations. Thus, our reported success rates already aggregate performance over many evaluation episodes rather than a single deterministic rollout. In this sense, the evaluation is effectively performed over multiple evaluation seeds. You probably mean you want

me to use several fixed seeds for evaluation. However, given the time and compute constraints of this course project, this requirement is not realistic and is not necessary for supporting our main claim.

**Comment 2:**

This point has already been addressed in the paper. We explicitly discuss the differences between the original RLT system and our SmolVLA-LIBERO adaptation. In particular, the original RLT paper uses a different VLA backbone, different robots, different real-world precision manipulation tasks, full VLA-token representations, chunk-level action refinement, and online actor-critic training. In contrast, our work uses SmolVLA, LIBERO-10 simulation, pooled action-head latent features, offline residual actor-critic learning, and single-step residual action correction.

The two experiments are not conducted on the same robot platform, benchmark, VLA backbone, action representation, or training regime. Because of these fundamental differences, a direct numerical comparison between the official RLT results and our results would not be scientifically meaningful. The original RLT results serve as conceptual motivation and methodological reference, while our experiments evaluate whether the same high-level idea can be adapted to an accessible SmolVLA-LIBERO setting.

Reviewer **Kai Yun**:

**Comment:**

Thank you for the advice. However, this comment is based on an incorrect premise. Our evaluation is not performed on a single task. We evaluate on LIBERO-10, which contains ten manipulation tasks, and each task is evaluated over 50 closed-loop episodes. Since both the frozen SmolVLA baseline and our SmolVLA-RLT policy are evaluated under this protocol, one full comparison contains 1000 closed-loop rollout episodes. The concern that the robot may be initialized in an unusually easy position is therefore already addressed by the evaluation design. Different rollout episodes use different evaluation seeds and environment initializations, and both policies are tested under the same task suite and evaluation protocol. Thus, the reported improvement is not the result of one lucky initialization.

分数: 92

总结: 对一篇NIPS2026  
的复现实验, 可以看出做了  
很多的工作; 由于代码没开源  
自己搭建的, 并且也用到了  
自己的数据

改进: 格式有些微问题:

建议 1. 正文最好按NIPS规定 $\leq 9$ 页,  
其余可放附录

2. 参考文献规范  
按照模版

3. 引言部分可以精简一些

姓名: 刘行逸

本文在LIBERO-10上复现并适配了RL Token框架, 使用SmoVLA作为骨干,  
通过Actor-Critic将成功率从42.4%提升至52.6%, 实现效果, 细节丰富

改进建议: 结果仅基于50 episode/任务评估, 建议至少运行3~5个不同  
种子报告均值和标准差, 并附上P值和置信区间。

缺少与原始RLT结果的比较, 建议在Discussion中加一段定性的  
比较说明

打分: 95

吴思源

韵悦, 98

这份报告实现了对RL Token (RLT) 框架的复现, 以SmoVLA为基础, 验证“稠密强化学习”  
优化预训练VLA模型的可行性。在复现过程中, 使用“冻结预训练+动态修正”的策略, 同  
时也有显存限制条件下的创新性改进, 结果上也验证了其优越性。在LIBERO-10基准  
测试中, 成功率提升, 任务耗时减少, 从而提升了机器人的操作精度与执行速度。

在改进建议方面, 我在表1和表2中注意到其只进行了单个任务的实现, 能否进行“多随机  
种子”并行实验, 这能确保成功率的提升不是因为机器人恰好在某次实验中初始代到  
一个较易完成的位置。

我认为这篇报告结构完整, 并且有根据自身设备受限进行改进创新, 也结合了RL与机  
器人, 选题较好, 能给分为98。

Figure 6: Overview of the reviewers' comments.