ActionComplete: Sample-Efficient Post-Training of VLAs via Residual RL

Saksham Sharma
saksham2
Robotics Institute
Carnegie Mellon University

Abhikhya Tripathy
abhikhyt
Robotics Institute
Carnegie Mellon University

Stephen Oduh
soduh
College of Engineering
Carnegie Mellon University

Abstract: Learning robotic manipulation policies that are both effective and sample-efficient remains a major challenge, particularly for tasks requiring precise physical interactions of the robot with the environment. Although end-to-end reinforcement learning (RL) has shown promise and been widely used for such tasks, it often suffers from slow convergence, sensitivity to reward design, and suboptimal exploration in complex environments. In this work, we investigate a framework that leverages pretrained base policies with rich prior knowledge, often learned via behavior cloning (BC), and adapts them to challenging manipulation tasks through the addition of a residual policy learned via online reinforcement learning. Instead of training from scratch, the residual policy only learns small actions to correct the base policy, resulting in more effective and sample-efficient learning. We specifically use Vision-Language-Action (VLA) models, which are pretrained on large amounts of data, as our base policy. We further highlight limitations of naive exploration and propose more informed exploration strategies beyond low-level action imitation.

1 Introduction

Robotic manipulation is a core problem in robotics, with applications ranging from industrial assembly to household assistance. Many manipulation tasks require precise control, contact-rich interactions, and robustness to uncertainty, making them challenging to solve efficiently with supervised learning methods. Reinforcement learning (RL) provides a general framework for learning manipulation policies through interaction and feedback from the environment. However, end-to-end RL methods are often slow to converge, sample-inefficient, and sensitive to reward design, which is done manually in most cases. Thus, in complex tasks where the agent lacks prior knowledge about the structure and dynamics of the environment, RL methods can fail. A promising alternative is to leverage pretrained large models that already encode rich prior knowledge from large datasets, such as Vision-Language-Action (VLA) models for robotics. These models perform well at predicting coarse robotic actions given visual input of the environment and language instructions, and can generalize across tasks and environments at this coarse level. However, due to limitations in how visual inputs are tokenized and processed, lack of interaction with the environment, and the frequency at which they can generate actions and get feedback from the environment, VLA models struggle with precise, closed-loop manipulation. Thus, VLA models behave well as high-level trajectory planners rather then accurate low-level controllers.

In this work, we aim to combine the strengths of pretrained models and reinforcement learning, by using a VLA model as a base policy and learning a residual reinforcement learning policy on top of it to predict corrective actions. The VLA model provides high-level action predictions using task-relevant priors, and the residual RL predicts small fine-grained actions to add to those predicted by the base policy. The residual RL policy is trained online through interactions with the environment, allowing it to learn precise adjustments that are beyond the scope of the base model. We keep the base VLA policy frozen throughout and train only the RL, leveraging the adaptability and explo-

ration ability of RL, and retaining the rich prior knowledge of the VLA model. Another advantage of keeping the base model frozen is that we don't need large amounts of data for fine-tuning the VLA model through SFT for different environments. The residual policy learns optimal behavior for each environment through interactions and feedback, achieving both sample efficiency and precise control. Crucially, this can enable usage of these specialized policies as expert demonstrators in a self-improving data flywheel [1]: their successful trajectories are aggregated to fine-tune the generalist VLA in a DAgger-like fashion, continuously enhancing the base model's capabilities.

2 Problem Statement

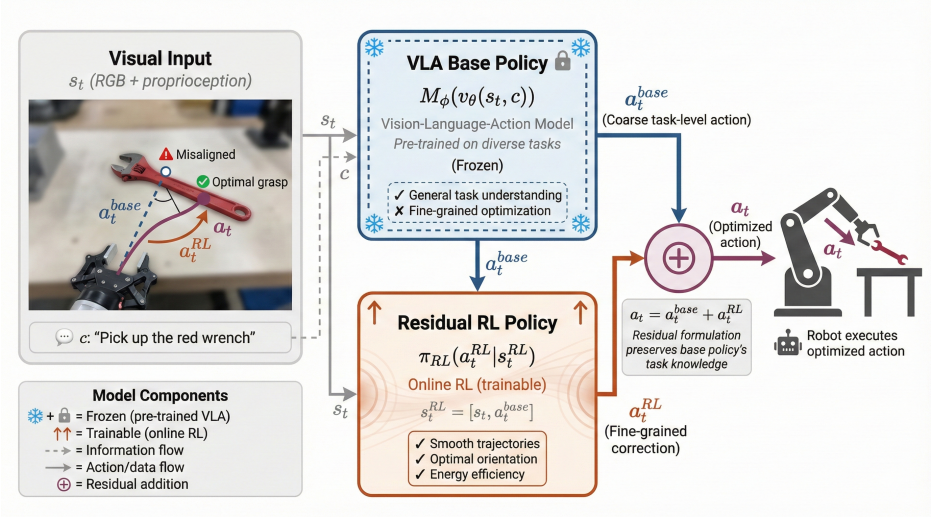

Figure 1: Overall Proposed Architecture

2.1 Residual Reinforcement Learning with Vision-Language-Action Model as Base Policy

Residual reinforcement learning aims to add small corrective actions on top of a pretrained base policy using an online reinforcement learning "residual" policy [2, 3].

In this work (as seen in Fig.1[1]), we choose our base policy to be a pretrained Vision-Language-Action (VLA) model which we keep frozen throughout. The model takes as input the visual (RGB) state of the environment through a specific camera angle, the robot proprioception information (position, velocity, orientation etc.), and a language instruction, and outputs an action a_t^{base} in a given timestep t.

We can formalize this as: $a_t^{base} = M_\phi(v_\theta(s_t, c))$, where v_θ is a vision-language model backbone parametrized by θ and M_ϕ is an action prediction head parametrized by ϕ, s_t is the environment state (visual input + robot proprioception information), and c is the language instruction.

We learn residual policy π_{RL} through online reinforcement learning. The residual policy's observation state is obtained by concatenating the base policy's observation state with its predicted action.

Thus, $s_t^{RL} = [s_t, a_t^{base}]$. In each timestep t, the residual RL policy predicts a corrective action $a_t^{RL} \sim \pi_{RL}(.|s_i^{RL})$, which is added to the base action.

This yields a combined policy:

$$\pi'(a_t|s_t, c) = M_\phi(v_\theta(s_t, c)) \cdot \pi_{RL}(a_t^{RL}|s_t^{RL}), \quad a_t = a_t^{base} + a_t^{RL} \tag{1}$$

In this work, we will implement 3 different reinforcement learning algorithms for the residual policy: On-policy Policy-gradient RL, Off-policy Q-function-based RL, Model-based RL.

[1] Generated using PaperBanana [4]

2.2 Environment

We use the *PegInsertionSide-v1* environment within ManiSkill robot simulator [5] to demonstrate the performance of our algorithms. The goal task in this environment is to pick up an orange-white peg and insert the orange end into a box with a hole in one of its lateral faces. We choose this because it requires detailed perception and precise control.

2.3 Reward Function

We define the reward (per timestep) as a combination of stage-specific components, which are only non-zero when the action is in the corresponding stage. The stages are 1) grasping the peg at its tail (R_{grasp}), 2) orienting the peg axis to align with the hole's normal axis (R_{orient}), and 3) inserting the peg's head into the hole (R_{insert}). On successful insertion, a large reward $R_{success}$ is returned and other components are 0. The stages are identified through the relative positions and orientations of the peg and the hole. The full reward can be written as R where $\lambda_1, \lambda_2, \lambda_3 \in \{0, 1\}$ depending on the stage.

$$R = \lambda_1 R_{\text{grasp}} + \lambda_2 R_{\text{orient}} + \lambda_3 R_{\text{insert}} + R_{\text{success}}$$

$$R_{\text{grasp}} = -\alpha_1 \|p_{\text{gripper}} - p_{\text{peg-tail}}\| - \alpha_2 |\cos^{-1}(v_{\text{gripper}} \cdot v_{\text{peg}})| + \beta_1 I_{contact} - \beta_3 ||g_{width} - g_{optimal}||$$

$$R_{\text{orient}} = -\gamma_1 |\cos^{-1}(v_{\text{peg}} \cdot v_{\text{hole}})|$$

$$R_{\text{insert}} = -\delta_1 \|p_{\text{peg-head}} - p_{\text{hole}}\| + \delta_2 d_{\text{inserted}}$$

p_i denotes the position of the entity i, v_i the orientation of entity i, $I_{contact} = 1$ if gripper makes contact with peg tail and 0 otherwise, g_{width} is the width of the gripper and $g_{optimal}$ its optimal width for stably grasping the peg, and $d_{inserted}$ is the depth of insertion into the hole.
When $d_{inserted} \geq d_{required}$, our task is successfully completed and $R_{success}$ is returned.

3 Proposed Modifications

3.1 Modification 1: Learning Exploration Strategies from Demonstrations

Once we have a combined policy π' that performs well on the standard peg insertion task, this modification extends it to settings with additional environmental complexity. The key idea is that humans often solve complex tasks via trial-and-error (e.g., inserting a plug into a socket behind a door or under a table without seeing it, throwing a lasso around a distant pole, or disentangling a highly coiled wire). However, humans develop strategies to make their trial-and-error, or exploration, more efficient, and apply these to any similar scenario. Our goal is to teach the residual RL policy π_{RL} to learn such *strategies* for efficient exploration from very few demonstrations of humans employing the same strategies for complex tasks [6, 7]. Another benefit is that human strategies often involve inherently safe exploration behaviors, which can be crucial in safety-critical settings.

To concretize, consider peg insertion into a hole located behind an opaque door, on its hidden surface, where our policy cannot "see" it. A human would first establish contact between the peg head and the hidden surface, then explore it systematically in a sweeping manner. Learning this strategy guarantees finding the hole within a bounded time, unlike lifting and re-placing the peg for each failure. Another example for efficient and effective human strategies is spinning a lasso to build momentum before throwing it around a target at a distance.

Concretely, we want the residual RL policy π_{RL} to learn such high-level strategies for peg insertion, rather than low-level actions, from a small number of human demonstrations, improving success rates and convergence in challenging setups. This design also allows incorporating additional sensory inputs, such as force or tactile feedback, into the residual policy without retraining the base VLA policy with new input modalities. By using RL instead of behavioral cloning, the system moves beyond low-level action imitation to learn strategic exploration patterns, thus learning from sparse human demonstrations.

3.2 Modification 2: Learning Human Like Behavior through Affordances

The residual policy π_{RL} explores blindly around the base policy's trajectory, often resulting in inefficient, human-alike movements (holding a knife from blade region instead of handle during exploration) before converging on a solution.

By integrating visual affordance priors derived from egocentric human videos [8] specifically graspable and functional segmentation maps we can shape the exploration of the residual policy. This method acts as a form of implicit imitation learning, where the agent is not just maximizing a sparse task reward but is also incentivized to interact with objects in a human aligned manner. To encourage the robot to interact with the correct part of the object:

$$r_{aff} = \lambda_{aff} \sum_{i \in \text{pixels}} M_{aff}^{(i)} \cdot \exp\left(-\frac{\|p_{ee} - p_i\|^2}{2\sigma^2}\right) \tag{2}$$

Where $M_{aff}^{(i)}$ activation of the affordance map at pixel i, p_{ee} robot's end-effector position in the image plane. This term rewards the end-effector for minimizing distance to high-confidence affordance regions (e.g., moving the gripper towards the knife handle).

So even though model doesn't explicitly "see" the affordances, because of the auxilary reward the policy should implicitly learn visual features that correlate with affordance rewards.

4 Results

4.1 Performance of a Random Agent

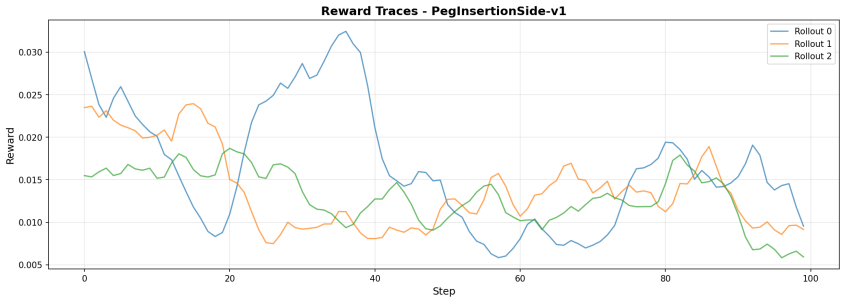

Figure 2: Reward v/s timesteps for three rollouts of a random agent

Table 1: Average Reward per episode for PegInsertionSide-v1 with random actions

Rollout	Mean Reward
Rollout 0	0.017
Rollout 1	0.014
Rollout 2	0.013

From Table 1 we can see that random agent fails to consistently make progress toward successful insertion, highlighting the difficulty of the task and the necessity of structured learning and exploration strategies.

4.2 Project Website

Videos and visualizations of agent behavior in the environment are available on our project website: https://osobotu.github.io/rlproject/.

References

[1] W. Xiao, H. Lin, A. Peng, H. Xue, T. He, Y. Xie, F. Hu, J. Wu, Z. Luo, L. Fan, et al. Self-improving vision-language-action models with data generation via residual rl. *arXiv preprint arXiv:2511.00091*, 2025.

[2] L. Ankile, A. Simeonov, I. Shenfeld, M. Torne, and P. Agrawal. From imitation to refinement-residual rl for precise assembly. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 01–08. IEEE, 2025.

[3] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine. Residual reinforcement learning for robot control. In *2019 international conference on robotics and automation (ICRA)*, pages 6023–6029. IEEE, 2019.

[4] D. Zhu, R. Meng, Y. Song, X. Wei, S. Li, T. Pfister, and J. Yoon. Paperbanana: Automating academic illustration for ai scientists. *arXiv preprint arXiv:2601.23265*, 2026.

[5] S. Tao, F. Xiang, A. Shukla, Y. Qin, X. Hinrichsen, X. Yuan, C. Bao, X. Lin, Y. Liu, T. kai Chan, Y. Gao, X. Li, T. Mu, N. Xiao, A. Gurha, V. N. Rajesh, Y. W. Choi, Y.-R. Chen, Z. Huang, R. Calandra, R. Chen, S. Luo, and H. Su. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *Robotics: Science and Systems*, 2025.

[6] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards, 2018. URL https://arxiv.org/abs/1707.08817.

[7] Y. Shi, Z. Chen, Y. Wu, D. Henkel, S. Riedel, H. Liu, Q. Feng, and J. Zhang. Combining learning from demonstration with learning by exploration to facilitate contact-rich tasks, 2021. URL https://arxiv.org/abs/2103.05904.

[8] G. Li, N. Tsagkas, J. Song, R. Mon-Williams, S. Vijayakumar, K. Shao, and L. Sevilla-Lara. Learning precise affordances from egocentric videos for robotic manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10581–10591, 2025.