SURPASSING VLA PERFORMANCE: ONLINE RL TRAINING OF LIGHTWEIGHT POLICIES WITH MINIMAL HUMAN INTERVENTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Pre-trained vision-language-action (VLA) models often exhibit low success rates and task completion speeds when deployed without fine-tuning in new environments. The standard approach to improve the performance of a pre-trained VLA is to collect new demonstrations and fine-tune the VLA for every new task encountered during deployment. However, collecting human demonstrations during deployment can be costly and time-consuming, which is a barrier to the widespread adoption of VLAs in industry and daily life. In this work, we propose a framework based on reinforcement learning for surpassing the performance of pre-trained VLAs by learning a lightweight policy (LWP) with no human demonstrations and minimal human intervention. By reinforcement learning only from online experience, our proposed framework *RELAY* transfers task-specific robot behavior from the VLA to the LWP and improves the behavior over time. Our experiments on three real-robot manipulation tasks show that RELAY-based agents autonomously improve their performance and achieve higher success rates and task completion speeds than the pre-trained VLA within six hours of interaction time.

1 INTRODUCTION

The common approach for improving the performance of a pre-trained vision-language-action (VLA) model on a task involves spending substantial human effort to collect new demonstrations and fine-tune the VLA. Large billion-parameter VLAs (Kim et al., 2024; Black et al., 2025; NVIDIA et al., 2025) are most often initialized from a vision-language model (Radford et al., 2021) backbone trained on web-scale image and text data. VLAs are subsequently pre-trained with supervised learning on large datasets of human demonstrations, such as Open-X Embodiment (Open X-Embodiment Collaboration, 2024) and DROID (Khazatsky et al., 2024), which contain various manipulation tasks on real robots. VLM training on web-scale image-text data and pre-training on vast amounts of human demonstrations grants VLAs the ability to generalize across different object types and tasks, such as picking and placing a marker, folding a towel, and wiping a table. Despite their generalization ability, pre-trained VLAs often suffer from moderate success rates and task completion speeds when deployed in new environments (Xu et al., 2026; Kim et al., 2024). Figure 1 shows some suboptimal behaviors found when deploying π_0 (Black et al., 2025) in three manipulation tasks.

To improve the performance of a VLA, humans commonly need to collect demonstrations for every new task and fine-tune the VLA. Requiring ongoing access to human demonstrators for every task encountered during deployment remains a barrier to the widespread adoption of VLAs in industry and daily life. VLA-based embodied agents have the potential to significantly increase productivity in performing physical tasks in warehouses, factories, and personal homes. Embodied agents deployed in such dynamic and unstructured environments routinely encounter and are expected to perform well on tasks that were not well represented in the pre-training dataset. However, as adoption scales to tens of millions of embodied agents, collecting human demonstrations for fine-tuning becomes economically infeasible, as the number of available humans cannot scale with the required amount of demonstrations. Thus, the ability to improve performance on a task autonomously without the need for human demonstrations is a prerequisite for the widespread adoption of VLA-based agents.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

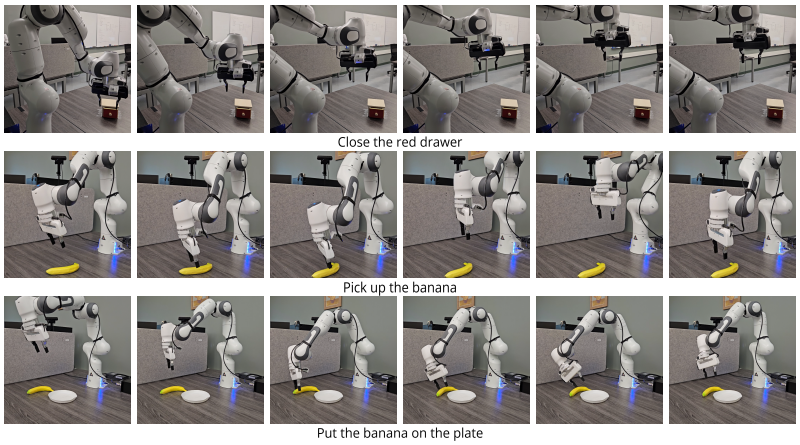

Figure 1: π_0 failure mode examples: backing away from the object (first row), closing the gripper above the object (second row), and failing to close the gripper (third row).

In this work, we propose *RELAY*, a framework that combines a model architecture with online reinforcement learning (Sutton & Barto, 2018) to surpass pre-trained VLAs without collecting any human demonstrations. RELAY's model architecture pairs a frozen pre-trained VLA with a lightweight policy (LWP) built from the VLA's shared frozen image encoder and a randomly initialized feedforward network. Both process the shared image embeddings independently, each outputting a chunk of eight actions to be executed in sequence. The LWP has an order of magnitude fewer parameters than the billion-parameter VLA, and only a fraction of them are learnable. This can speed up inference and task completion, and lets inference and learning updates run simultaneously on a single RTX 3090 GPU. Online learning has two stages that both update the LWP with online RL: Stage 1 collects experience mostly with the VLA and Stage 2 with the LWP, while mini-batches gradually shift toward LWP-generated experience. By learning from both VLA- and LWP-generated experience, RELAY transfers the VLA's task-specific behavior and improves on it over time.

On three real-robot object manipulation tasks, we show that our RELAY-based agent autonomously improves task performance from online experience with minimal human intervention and achieves higher success rates and task completion speeds than the pre-trained VLA. We instantiate an agent from the RELAY framework using the publicly-available pre-trained π_0 VLA (Black et al., 2025) that has been fine-tuned on the DROID dataset (Khazatsky et al., 2024) and use calibrated Q-learning (Nakamoto et al., 2023) as the RL algorithm. We chose three tasks that are representative of common object manipulation tasks in the DROID dataset and on which the π_0 VLA has a moderate but non-zero success rate, so that successful episodes are available to learn from. To greatly reduce the need for human interventions during experiments, we automated task success detection and environment resets with the help of LLM coding agents. On a Franka Panda robot arm with a different gripper than what π_0 was trained on, our RELAY-based agent roughly triples the success rate of the π_0 VLA and completes tasks 2.1 to 3.4 times faster, without collecting any human demonstrations and in less than six hours of interaction time. A human only provides minimal interventions on rare occasions when an object or the robot enters an unrecoverable state.

Our contributions are: (1) RELAY, a framework that transfers task-specific behavior from a frozen VLA to a lightweight policy through online RL and improves upon the behavior. (2) We demonstrate real-robot training with automated success detection and environment resets without human demonstrations or success labels. (3) On three real-robot manipulation tasks, we show that our RELAY-based agent surpasses the success rate and task completion speed of the pre-trained π_0 VLA, and that at least 29% of the increase in task completion speed comes from more efficient behavior learned through RL rather than from faster inference alone.

2 Problem Setup

Our goal in this paper is to surpass the performance of a pre-trained VLA without collecting any human demonstrations. We aim to achieve our goal by learning from online experience through trial and error without using any reward models or complex engineered reward functions. We therefore use a task-agnostic reward: a time penalty until success and generic safety penalties. To ensure that an informative learning signal exists, we choose tasks on which the pre-trained VLA has non-zero success rate. Learning to perform tasks that the pre-trained VLA cannot perform is out of the scope of this work. Additionally, we define surpassing the performance of a pre-trained VLA as achieving a higher success rate or lower success duration than the pre-trained VLA. *Success rate* is defined as the proportion of successful episodes within a total number of episodes. *Success duration* is defined as the duration of a successful episode in seconds.

We use the reinforcement learning framework (Sutton & Barto, 2018), in which an agent interacts with an environment through a sequence of time steps $t \geq 0$. At each time step t, the agent observes the current state s_t, applies an action $a_t \sim \pi(\cdot \mid s_t)$ to the environment, receives the next state s_{t+1} and reward r_{t+1}, and the cycle repeats until episode termination at time step T. The goal of the RL agent is to maximize the expected value of the undiscounted episodic return $\int_s d_0(s)\mathbb{E}_\pi[\sum_{t=0}^{T} r_t \mid s_0 = s]$.

3 The RELAY Framework

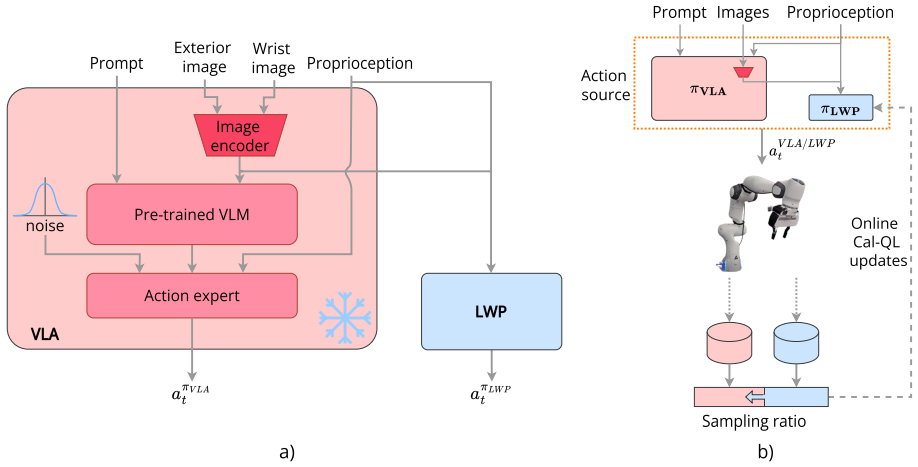

Figure 2: The RELAY framework. a) Model architecture. It comprises a flow-matching VLA that shares its image encoder with a LWP. b) Learning process. In stage 1, the action source is the VLA for the first 30 episodes, then the LWP every 5 VLA episodes. In stage 2, the LWP is the default action source, except during recovery. After choosing an actor and executing its action chunk, the transitions get stored in the corresponding replay buffer. Then, the transitions are sampled from the VLA and LWP buffers to perform online Cal-QL updates. In stage 1 and the beginning of stage 2, the sampling ratio is 50/50. It decays to 100% LWP (except VLA recovery transitions) in stage 2.

In this section, we describe the model architecture and online learning process of the RELAY framework. Although large pre-trained VLAs can identify various common objects and perform manipulation tasks out of the box, having billions of parameters significantly increases the inference time of such models. For example, the JAX implementation of the π_0 VLA takes at least 140ms to produce an output on an RTX 3090 GPU when the GPU compute resources are not shared by any other processes such as learning updates or VLM inferences. Since producing and executing a single action once every 140ms is undesirable for controlling real robots, for each inference, VLAs most often output an action chunk, which is a list of actions to be executed in sequence. With action chunking, however, the VLA receives stimuli from and can react to the environment only when an entire chunk has been executed, which can take more than half a second with π_0 on the Franka arm. Thus, action

chunking can impair the reactiveness and precision of the VLA behavior despite increasing the robot control frequency.

To maintain high robot control frequencies and allow for reactive behavior, the RELAY framework augments a pre-trained VLA with a lightweight policy (LWP) that can produce actions independently without waiting for a VLA inference. The LWP has far fewer parameters than the VLA, resulting in significantly faster inference. We instantiate an agent from the RELAY framework by augmenting the π_0 VLA pre-trained on the DROID dataset with an LWP that shares the image encoder of π_0. The LWP inputs one image from each of the wrist-mounted camera and the fixed exterior camera, which are processed by the frozen SigLip (Zhai et al., 2023) image encoder of π_0 to produce the image embeddings. The image embeddings are then concatenated with the proprioception vector and fed to a feedforward network to produce a chunk of actions. Although outputting a single action for each inference allows for reactive behavior, we implement the LWP to output an action chunk instead to speed up learning and leave a single-action to future work. With only a few million learnable parameters, the LWP enables computationally cheaper learning updates compared to low-rank adaptation of the VLA and can learn using a single GPU to perform manipulation tasks in less than six hours of interaction time. The VLA policy in the RELAY framework can continue to output action chunks without any modifications, independently of the LWP.

3.1 RELAY Learning Process

We now turn our attention to the online learning process of RELAY. The desired outcome of the learning process is an LWP that can perform a given manipulation task continuously and can improve online performance on the task indefinitely. However, parts of the LWP network are randomly initialized, leading to random exploratory LWP-generated actions and zero LWP task success rate in the beginning. Without achieving any task success, the randomly-initialized LWP does not receive a learning signal to differentiate between good and bad behavior. We use the non-zero success rate of the VLA policy on the task as a source of exploratory behavior to provide a reliable learning signal for the LWP.

RELAY uses the soft actor-critic (SAC, Haarnoja et al. 2018) family of algorithms to learn the LWP in a time-efficient manner on real robots. SAC learns the actor parameters by maximizing a policy objective composed of an action-value and an entropy term. However, SAC suffers from extrapolation error (Fujimoto et al. 2019), which results in erroneously-learned action values when the learning data distribution differs from that of the current policy. Hence, maximizing an action-value function learned using SAC from VLA-generated transitions results in an LWP that exhibits random, often unsafe, robot movements and fails to learn the task due to unrealistic action values assigned to non-task-related states and actions. To mitigate extrapolation error, our RELAY-based agent employs the Calibrated Q-learning algorithm (Cal-QL, Nakamoto et al. 2023). Although Cal-QL was introduced as a solution for reducing extrapolation error in offline RL primarily, the RELAY framework does not use offline learning, and our agent performs Cal-QL learning updates only online, that is, while interacting with the environment.

We implement the Q function similarly to the LWP with a feedforward network that inputs the image embeddings produced by the frozen Siglip image encoder concatenated with the proprioception vector and outputs a scalar. In addition, we use an ensemble of ten Q functions (Chen et al. 2021).

Since the pre-trained VLA has a non-zero initial success rate and we want an LWP that surpasses it and performs the task continuously, RELAY splits learning into two stages. Both stages use Cal-QL to learn the LWP and the critics and differ mainly in the ratio of VLA- and LWP-generated episodes and mini-batch samples. Stage one gathers mostly VLA-generated episodes (those with actions computed by the VLA) and learns an LWP with a success rate comparable to the VLA's. It collects VLA episodes for the first 30 episodes, then one LWP episode for every five consecutive VLA episodes. It ends when the LWP's average success rate over its last ten episodes is at least 30% and exceeds the VLA's average success rate over all episodes minus 10%. In stage two, all actions come from the LWP until an episode timeout, after which VLA recovery mode alternates between VLA and LWP actions for a few consecutive steps each, helping the robot recover from LWP mistakes.

The learning process keeps separate replay buffers for VLA and LWP episodes. In stage one, mini-batches are 100% VLA samples for the first 30 episodes and 50% VLA and 50% LWP samples

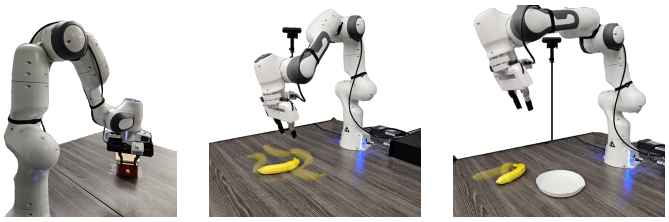

Figure 3: Real-world manipulation tasks. From left to right: close the red drawer, pick up the banana, and put the banana on the plate. Banana positions are randomized for the two banana tasks.

thereafter. In stage two, the ratio starts at 50% VLA and 50% LWP and decays linearly to 100% LWP over the first 100 episodes. Learning updates run in an asynchronous thread that does not block action inference or robot control. Pseudocode of the learning process is given in the algorithm below

Algorithm 1: RELAY Learning Process

Input: task prompt l, image encoder weights θ_{enc}, non-encoder LWP weights θ_{LWP}, non-encoder VLA weights θ_{VLA}, weights for N critics $\{\phi\}_{n=1}^{N}$ and N target critics $\{\overline{\phi}\}_{n=1}^{N}$, fixed temperature α, learning rates η_θ and η_ϕ, initial-state distribution d_0, transition probabilities p, timeout steps T_{timeout}, smoothing parameter τ

$j \leftarrow 0$, sampling ratio $r \leftarrow 0.5$, $\mathcal{D}_{\text{VLA}} \leftarrow \emptyset$, $\mathcal{D}_{\text{LWP}} \leftarrow \emptyset$

$\boldsymbol{\xi} \doteq d_0, p, \theta_{\text{enc}}, \theta_{\text{VLA}}, l, T_{\text{timeout}}$

$\boldsymbol{\Theta} \doteq \theta_{\text{LWP}}, \{\phi\}_{n=1}^{N}, \{\overline{\phi}\}_{n=1}^{N}, \alpha, \eta_\theta, \eta_\phi, \tau$

`// stage one`

for *episode* $i = 0, 1, 2, \cdots$ **do**

  if $i \leq 30$ ***or*** $i \bmod 5 \neq 0$ **then**

    $\text{succ}_{\text{VLA},i} \doteq$ `collect_rollout`$(\theta_{\text{VLA}}, \mathcal{D}_{\text{VLA}}, 0, \boldsymbol{\xi})$

  else

    $\text{succ}_{\text{LWP},i} \doteq$ `collect_rollout`$(\theta_{\text{LWP}}, \mathcal{D}_{\text{LWP}}, 0, \boldsymbol{\xi})$

    $j \leftarrow j + 1$

  $\text{succ_rate}_{\text{VLA,i}} \doteq \mathbb{E}_{k\sim\{0,\cdots,i\}}\left[\text{succ}_{\text{VLA},k}\right]$

  $\text{succ_rate}_{\text{LWP,i}} \doteq \mathbb{E}_{k\sim\{j-9,\cdots,j\}}\left[\text{succ}_{\text{LWP},k}\right]$

  if $\text{succ_rate}_{\text{LWP},i} \geq \max(30,\ \text{succ_rate}_{\text{VLA},i} - 10)$ **then**

    break

`// stage two`

for *episode* $k = 0, 1, 2, \cdots$ **do**

  `collect_rollout`$(\theta_{\text{LWP}}, \mathcal{D}_{\text{LWP}}, 1, \boldsymbol{\xi})$

  $r \leftarrow \min(1,\ 0.5 + \frac{0.5k}{100})$

`// asynchronous thread`

`Cal_QL_update`$(\mathcal{D}_{\text{LWP}}, \mathcal{D}_{\text{VLA}}, \boldsymbol{\Theta})$

4 Experiments

We evaluate RELAY on real-world manipulation tasks using a Franka Emika Panda arm equipped with a wrist-mounted camera and an exterior camera. Our experiments address the following questions: (Q1) Does RELAY increase success rate and decrease success duration during online training; (Q2) At deployment, does the LWP, running without the VLA, address the VLA's failure modes (Figure 1) and outperform it in both metrics?; and (Q3) How much of the reduction in success duration comes from faster inference, and how much from better behavior learned through RL?

4.1 Experimental Setup

We selected three manipulation tasks, shown with their initial states in Figure 3. For each task, we implement an automated success detector and an automated reset procedure.

Close the drawer. The robot must close a small red drawer. We detect success by segmenting the drawer with SAM3 (Carion et al., 2026) and comparing the centroid of its mask with that of a reference image of the closed drawer. Because the arm can occlude the drawer, success is registered only after the robot has closed the drawer and moved away from it. The reset script fully closes the drawer and then reopens it to a fixed position.

Pick up the banana. The robot must grasp and lift a banana whose position is randomized within a 23×32 cm region. An episode is successful when the gripper is closed on an object, as indicated by the gripper width exceeding 3 mm, and the end effector is more than 8 cm above the table. The reset script raises the arm to segment and locate the banana, positions the gripper above it and refines the grasp position, picks the banana up, places it at a random position, and returns the arm to its home configuration. A reset verifier segments the banana and checks that the new scene differs sufficiently from the one after the previous reset, retrying the reset otherwise.

Put the banana on the plate. The robot must place the banana on a plate. An episode is successful when more than 50% of the banana mask overlaps the plate mask. The reset follows the same procedure as in the picking task, with a smaller randomization area of 10×20 cm.

Success detectors and reset procedures were implemented with the help of large language model (LLM) coding agents before running any experiments, to minimize human involvement. None of our experiments collected or used human demonstrations or corrective trajectories. Experiments were run with minimal human intervention, with occasional human help when the robot or the objects entered a rare unrecoverable state.

Reward. All tasks share the same reward function, and only the success detector differs between them:

$$r_t = -1 \; - \; 10\,\mathbb{1}[\text{table collision}] \; - \; \frac{1-\gamma^T}{1-\gamma}\,\mathbb{1}[\text{unsafe}],$$

where γ is the discount factor and T is the maximum episode duration, which is 200, 240, and 400 timesteps for the drawer, banana-picking, and banana-on-plate tasks, respectively. An episode terminates upon success, upon unsafe behavior, or after T timesteps. Unsafe behavior is defined as exceeding a safe boundary defined at the beginning of the experiments, hitting a joint limit, or risking self-collision. We found that this reward term quickly mitigated unwanted behaviors as training progressed. Because the agent receives -1 at every timestep until termination, maximizing the return is equivalent to succeeding as quickly as possible, which directly optimizes both success rate and success duration. Unsafe behavior is defined as leaving a predefined workspace boundary, reaching a joint limit, or risking self-collision. The unsafe-behavior penalty equals the discounted return of an episode that times out, which ensures that ending an episode early through unsafe behavior is never preferable to failing.

4.2 Results

Online improvement (Q1). Figure 4 shows the success rate and success duration throughout training. During Stage 1, most episodes are collected by the VLA, together with periodic LWP evaluation episodes, and performance stays close to the VLA baseline. Once the LWP's success rate approaches that of the VLA and training switches to Stage 2, performance improves rapidly on all three tasks. On the drawer task, the success rate rises from below 0.5 to about 0.9 at the beginning of Stage 2, while success duration drops from about 18 s to below 8 s. On the banana-picking task, the success rate increases steadily to about 0.65, and success duration falls from about 21 s to about 11 s. The banana-on-plate task, which has the longest horizon, improves more slowly and with higher variance between windows, while its success duration drops from about 45 s to about 20 s. Training took 2.8, 3.6, and 5.8 hours of interaction time for the drawer, banana-picking, and banana-on-plate tasks, respectively. Because Stage 2 episodes use stochastic actions and include VLA recovery segments, these curves do not measure the LWP's standalone performance, which we evaluate next.

Deployment performance (Q2). We evaluate the final LWP on its own, without VLA recovery and using the mean action, for 30 episodes per task (Figure 5). The LWP outperforms the VLA on every task and on both metrics. It raises the success rate from 0.32 to 1.00 on the drawer task, from 0.25 to 0.77 on the banana-on-plate task, and from 0.27 to 0.77 on the banana-picking task. Figures 1 and 6 show that these gains come from resolving the VLA's failure modes: the LWP reaches directly

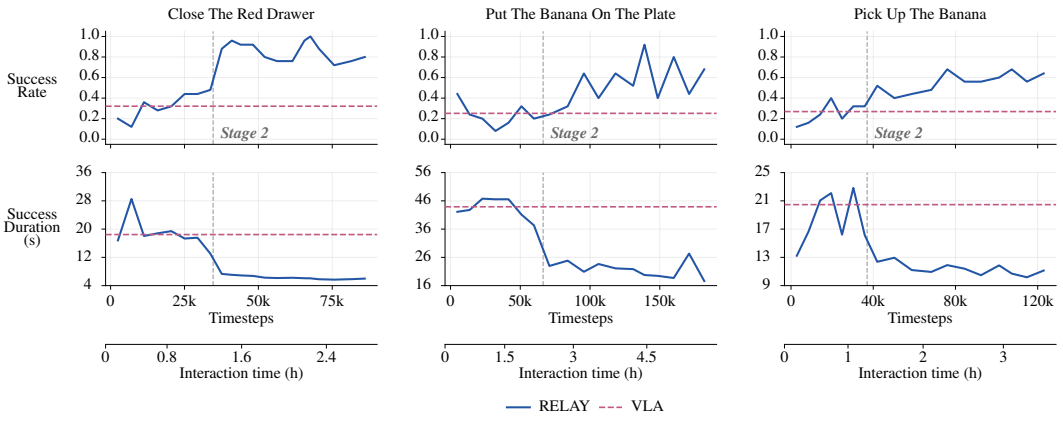

Figure 4: Online performance during training. Top: success rate. Bottom: duration of successful episodes, in seconds. Both are averaged over non-overlapping 25-episode windows. The dashed vertical line marks the start of Stage 2. The horizontal axis shows timesteps, and the second axis shows cumulative interaction time, meaning the time spent executing actions, including recovery but excluding environment resets. The dashed horizontal line shows the VLA baseline. Its success rate and success duration are measured over the entire training.



Figure 5: Performance comparison between the VLA and the LWP after training. Top: mean success rate. Bottom: mean duration of successful episodes, in seconds and steps. The LWP was evaluated over 30 episodes per task using the mean (deterministic) action. The VLA values come from all of its episodes collected during training. Bars show the mean, and error bars show the standard error.

Table 1: Share of the reduction in success duration from the VLA to the LWP attributable to faster inference per timestep and to fewer timesteps per successful episode, computed with Equation 1. The timestep share, which reflects behavior learned through RL, is a lower bound.

TASK	INFERENCE	TIMESTEPS
Close the drawer	45%	55%
Put the banana on the plate	64%	36%
Pick up the banana	71%	29%

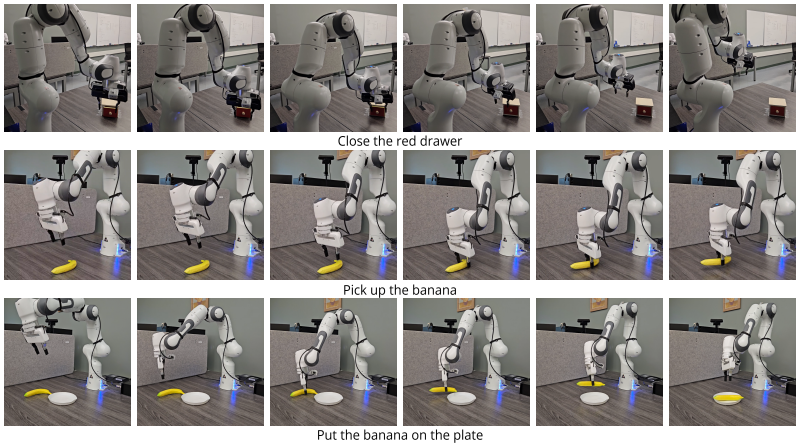

Figure 6: LWP rollouts. RELAY-trained policies address the failure modes observed in π_0: they reach directly for the drawer and close it (top row), pick up the banana by grasping it on the correct spot (middle row), and put the banana on the plate (last row).

for the drawer and closes it, grasps the banana at a stable point, and places it on the plate. Success duration also improved, decreasing from 18.5 s to 5.4 s, from 43.9 s to 18.1 s, and from 20.5 s to 9.6 s, respectively.

Sources of speedup (Q3). The LWP can complete a task faster for two reasons: its smaller network runs inference faster, so each timestep takes less time, and RL can teach it to complete the task in fewer timesteps. Let $\bar{T}_{\mathrm{V}}$ and $\bar{T}_{\mathrm{L}}$ denote the mean number of timesteps in successful episodes of the VLA and the LWP, and let Δt_{V} and Δt_{L} denote their mean duration per timestep in seconds. The reduction in success duration then decomposes exactly as

$$\bar{T}_{\mathrm{V}}\Delta t_{\mathrm{V}} - \bar{T}_{\mathrm{L}}\Delta t_{\mathrm{L}} = \underbrace{\bar{T}_{\mathrm{V}}(\Delta t_{\mathrm{V}} - \Delta t_{\mathrm{L}})}_{\text{faster inference}} + \underbrace{(\bar{T}_{\mathrm{V}} - \bar{T}_{\mathrm{L}})\,\Delta t_{\mathrm{L}}}_{\text{fewer timesteps}}. \tag{1}$$

Table 1 shows that faster inference accounts for most of the reduction on the two banana tasks, whereas fewer timesteps account for most of it on the drawer task. Across tasks, fewer timesteps account for 29–55% of the reduction, or 3.2 to 7.2 s per successful episode. The LWP therefore does not merely execute the VLA's behavior faster, but learns a more efficient behavior through RL. Because this decomposition attributes the interaction term $(\bar{T}_{\mathrm{V}} - \bar{T}_{\mathrm{L}})(\Delta t_{\mathrm{V}} - \Delta t_{\mathrm{L}})$ entirely to faster inference, these values are a lower bound on the contribution of RL.

5 Related Work

Fine-tuning VLAs. VLAs have achieved remarkable results in challenging manipulation tasks such as folding laundry, preparing espresso drinks, and assembling boxes (Physical Intelligence et al., 2025). However, zero-shot performance frequently has a large margin for improvement. To close this gap, many works fine-tune the entire VLA on tasks where the policy initially struggles with human interventions (Chen et al., 2025; Han et al., 2026) or autonomously (Lu et al., 2025; Tan et

al., 2025; Li et al., 2026; Bai et al., 2025). However, fine-tuning a multi-billion parameter model can be expensive and impractical. Many of these works were tested only in simulation, highlighting the difficulty of applying them in the real world (Lu et al., 2025; Bai et al., 2025; Tan et al., 2025).

Training a separate module on top of the VLA. Another line of work avoids fine-tuning the VLA by keeping it frozen and adding additional components on top of it. Value-based methods (Giridhar et al., 2026; Nakamoto et al., 2024) train a critic that assigns value to N action chunks generated by the VLA. Then, based on their value, the weighted average (Giridhar et al., 2026) or the highest-ranked action (Nakamoto et al., 2024) gets executed. However, behavior will be limited by the actions that the VLA produces. Additionally, the VLA must still be queried at every timestep. Given the large size of VLAs, inference can be slow, hindering the optimization of success duration. An additional approach is to train not only a critic but also a lightweight policy (Wagenmaker et al., 2025; Xiao, Lin, et al., 2026; Xu et al., 2026; Hu et al., 2024; Dong, Hung, Gao, et al., 2026; Dong, Hung, Sadigh, et al., 2026; Yuan et al., 2025). This modification yields more expressive action distributions than with value-based methods, but these methods still query the VLA backbone at every timestep. Our method, RELAY, trains an independent lightweight policy module that requires only image embeddings, rather than a full VLA forward pass.

Improving VLA performance with a trained lightweight policy and limited human intervention. Current methods for improving VLA performance still rely on varying degrees of human intervention. Xu et al. (2026) relies on access to a human who classifies episodes as successful or not; Wagenmaker et al. (2025)'s public implementation reveals the same dependency. Other approaches require collecting demonstrations to train a success classifier (Xiao, Lin, et al., 2026). Our work automates this by using rule-based success detection systems based on the SAM3 segmentation model (Carion et al., 2026). Other works require collecting costly demonstrations (Hu et al., 2024), interventions during training (Dong, Hung, Gao, et al., 2026; Xu et al., 2026), or manual resets (Dong, Hung, Gao, et al., 2026; Dong, Hung, Sadigh, et al., 2026). Our method requires no demonstrations and fully automates resets for each task. Recently, agentic frameworks have achieved substantial degrees of automation (Xiao, Xie, et al., 2026), where agents design their own success detectors and resets. Nevertheless, humans still need to provide feedback during the environment construction phase and collect a small dataset of labeled demonstrations.

Our work combines an independent lightweight policy plus an episode-level automated relay mechanism. No human is needed to classify successes, intervene during rollouts, provide demonstrations at any stage, or reset the environment (excluding rare cases where the robot is in an unrecoverable state). Because the lightweight policy operates on image embeddings rather than a full VLA forward pass, this also resolves the inference-cost and success-duration problem that persists across the value-based and actor-based methods above.

6 Conclusion

The standard approach for improving the performance of a pre-trained vision-language-action model on a task during deployment relies on collecting human demonstrations and fine-tuning the VLA, which can be costly and time-consuming. In this work, we proposed a framework named RELAY for surpassing the performance of a pre-trained VLA on a task using online reinforcement learning with no human demonstrations. Our experiments on three real-robot manipulation tasks showed that our RELAY-based agent achieved higher success rates and lower success durations than the pre-trained π_0 VLA and that having fewer lightweight policy parameters than the VLA and online RL contribute to the reduction in success duration. The RELAY framework provides a practical step toward widespread adoption of autonomous self-improving embodied agents.

AI USE STATEMENT

In this work, we used generative AI tools to design or provide feedback on experiments and implement methods. We have not used AI tools to help develop theoretical models or conceptual frameworks, propose or refine hypotheses, assist with translation, clean and reformat datasets, support qualitative and thematic data analysis, or interpret results. The tasks of generating synthetic data sets, formulating mathematical claims, providing critical ingredients for proving mathematical claims, and assisting in the writing of proofs do not apply to this work. Additionally, we used generative AI tools to aid and polish writing and to identify relevant literature. We reviewed all AI-assisted work. Before executing code written by LLMs, we verified that it matched its intended goal and validated it again after execution. The authors reviewed text written with the aid of AI tools before including it in the article. The authors read the works identified in the literature by AI to assess their relevance. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Bai, Z., Gao, C., Shou, M. Z. (2025). EVOLVE-VLA: Test-time training from environment feedback for vision-language-action models. *arXiv preprint arXiv:2512.14666*.

Black, K., Brown, N., Driess, D., Esmail, A., Equi, M. R., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., Jakubczak, S., Jones, T., Ke, L., Levine, S., Li-Bell, A., Mothukuri, M., Nair, S., Pertsch, K., Shi, L. X., Smith, L., Tanner, J., Vuong, Q., Walling, A., Wang, H., Zhilinsky, U. (2025). π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems XXI*.

Carion, N., Gustafson, L., Hu, Y.-T., Debnath, S., Hu, R., Suris, D., Ryali, C., Alwala, K. V., Khedr, H., Huang, A., Lei, J., Ma, T., Guo, B., Kalla, A., Marks, M., Greer, J., Wang, M., Sun, P., Rädle, R., Afouras, T., Mavroudi, E., Xu, K., Wu, T.-H., Zhou, Y., Momeni, L., Hazra, R., Ding, S., Vaze, S., Porcher, F., Li, F., Li, S., Kamath, A., Cheng, H. K., Dollár, P., Ravi, N., Saenko, K., Zhang, P., Feichtenhofer, C. (2026). SAM 3: Segment anything with concepts. In *Proceedings of the 14th International Conference on Learning Representations*.

Chen, X., Wang, C., Zhou, Z., Ross, K. W. (2021). Randomized ensembled double Q-learning: Learning fast without a model. In *Proceedings of the 9th International Conference on Learning Representations*.

Chen, Y., Tian, S., Liu, S., Zhou, Y., Li, H., Zhao, D. (2025). ConRFT: A reinforced fine-tuning method for VLA models via consistency policy. In *Proceedings of Robotics: Science and Systems XXI*.

Dong, P., Hung, K.-H., Gao, T., Sadigh, D., Finn, C. (2026). EXPO-FT: Sample-efficient reinforcement learning finetuning for vision-language-action models. *arXiv preprint arXiv:2605.25477*.

Dong, P., Hung, K.-H., Sadigh, D., Finn, C. (2026). Reinforcement learning for real-time vision-language-action policies. *arXiv preprint arXiv:2609.18207*.

Fujimoto, S., Meger, D., Precup, D. (2019). Off-policy deep reinforcement learning without exploration. In *Proceedings of the 36th International Conference on Machine Learning*.

Giridhar, V., Khandelwal, A., Collins, J. A., Georgiev, I., Garg, A. (2026). Beyond imitation: Self-improving robot policies via off-policy Q-planning. *arXiv preprint arXiv:2608.21204*.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., Levine, S. (2018). Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

Han, Y., Chen, Z., Zhao, Y., Xu, C., Shao, Y., Peng, Y., Mu, Y., Lian, W. (2026). DexHiL: A human-in-the-loop framework for vision-language-action model post-training in dexterous manipulation. *arXiv preprint arXiv:2603.09121*.

Hu, H., Mirchandani, S., Sadigh, D. (2024). Imitation bootstrapped reinforcement learning. In *Proceedings of Robotics: Science and Systems XX*.

Khazatsky, A., Pertsch, K., Nair, S., Balakrishna, A., Dasari, S., Karamcheti, S., Nasiriany, S., Srirama, M. K., Chen, L. Y., Ellis, K., Fagan, P. D., Hejna, J., Itkina, M., Lepert, M., Ma, Y. J., Miller, P. T., Wu, J., Belkhale, S., Dass, S., Ha, H., Jain, A., Lee, A., Lee, Y., Memmel, M., Park, S., Radosavovic, I., Wang, K., Zhan, A., Black, K., Chi, C., Hatch, K. B., Lin, S., Lu, J., Mercat, J., Rehman, A., Sanketi, P. R., Sharma, A., Simpson, C., Vuong, Q., Walke, H. R., Wulfe, B., Xiao, T., Yang, J. H., Yavary, A., Zhao, T. Z., Agia, C., Baijal, R., Castro, M. G., Chen, D., Chen, Q., Chung, T., Drake, J., Foster, E. P., Gao, J., Herrera, D. A., Heo, M., Hsu, K., Hu, J., Jackson, D., Le, C., Li, Y., Lin, R., Ma, Z., Maddukuri, A., Mirchandani, S., Morton, D., Nguyen, T., O'Neill, A., Scalise, R., Seale, D., Son, V., Tian, S., Tran, E., Wang, A. E., Wu, Y., Xie, A., Yang, J., Yin, P., Zhang, Y., Bastani, O., Berseth, G., Bohg, J., Goldberg, K., Gupta, A., Gupta, A., Jayaraman, D., Lim, J. J., Malik, J., Martín-Martín, R., Ramamoorthy, S., Sadigh, D., Song, S., Wu, J., Yip, M. C., Zhu, Y., Kollar, T., Levine, S., Finn, C. (2024). DROID: A large-scale in-the-wild robot manipulation dataset. In *Proceedings of Robotics: Science and Systems XX*.

Kim, M. J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E. P., Sanketi, P. R., Vuong, Q., Kollar, T., Burchfiel, B., Tedrake, R., Sadigh, D., Levine, S., Liang, P., Finn, C. (2024). OpenVLA: An open-source vision-language-action model. In *Proceedings of the 8th Conference on Robot Learning*.

Li, H., Zuo, Y., Yu, J., Zhang, Y., Yang, Z., Zhang, K., Zhu, X., Zhang, Y., Chen, T., Cui, G., Wang, D., Luo, D., Fan, Y., Sun, Y., Zeng, J., Pang, J., Zhang, S., Wang, Y., Mu, Y., Zhou, B., Ding, N. (2026). SimpleVLA-RL: Scaling VLA training via reinforcement learning. In *Proceedings of the 14th International Conference on Learning Representations*.

Lu, G., Guo, W., Zhang, C., Zhou, Y., Jiang, H., Gao, Z., Tang, Y., Wang, Z. (2025). VLA-RL: Towards masterful and general robotic manipulation with scalable reinforcement learning. *arXiv preprint arXiv:2505.18719*.

Nakamoto, M., Mees, O., Kumar, A., Levine, S. (2024). Steering your generalists: Improving robotic foundation models via value guidance. In *Proceedings of the 8th Conference on Robot Learning*.

Nakamoto, M., Zhai, Y., Singh, A., Mark, M. S., Ma, Y., Finn, C., Kumar, A., Levine, S. (2023). Cal-QL: Calibrated offline RL pre-training for efficient online fine-tuning. In *Advances in Neural Information Processing Systems*.

NVIDIA, Bjorck, J., Castañeda, F., Cherniadev, N., Da, X., Ding, R., Fan, L., Fang, Y., Fox, D., Hu, F., Huang, S., Jang, J., Jiang, Z., Kautz, J., Kundalia, K., Lao, L., Li, Z., Lin, Z., Lin, K., Liu, G., Llontop, E., Magne, L., Mandlekar, A., Narayan, A., Nasiriany, S., Reed, S., Tan, Y. L., Wang, G., Wang, Z., Wang, J., Wang, Q., Xiang, J., Xie, Y., Xu, Y., Xu, Z., Ye, S., Yu, Z., Zhang, A., Zhang, H., Zhao, Y., Zheng, R., Zhu, Y. (2025). GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*.

Open X-Embodiment Collaboration (2024). Open X-Embodiment: Robotic learning datasets and RT-X models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903.

Physical Intelligence, Amin, A., Aniceto, R., Balakrishna, A., Black, K., Conley, K., Connors, G., Darpinian, J., Dhabalia, K., DiCarlo, J., Driess, D., Equi, M., Esmail, A., Fang, Y., Finn, C., Glossop, C., Godden, T., Goryachev, I., Groom, L., Hancock, H., Hausman, K., Hussein, G., Ichter, B., Jakubczak, S., Jen, R., Jones, T., Katz, B., Ke, L., Kuchi, C., Lamb, M., LeBlanc, D., Levine, S., Li-Bell, A., Lu, Y., Mano, V., Mothukuri, M., Nair, S., Pertsch, K., Ren, A. Z., Sharma, C., Shi, L. X., Smith, L., Springenberg, J. T., Stachowicz, K., Stoeckle, W., Swerdlow, A., Tanner, J., Torne, M., Vuong, Q., Walling, A., Wang, H., Williams, B., Yoo, S., Yu, L., Zhilinsky, U., Zhou, Z. (2025). $\pi^*_{0.6}$: A VLA that learns from experience. *arXiv preprint arXiv:2511.14759*.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I. (2021). Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*.

Sutton, R. S., Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.

Tan, S., Dou, K., Zhao, Y., Krähenbühl, P. (2025). Interactive post-training for vision-language-action models. *arXiv preprint arXiv:2505.17016*.

Wagenmaker, A., Nakamoto, M., Zhang, Y., Park, S., Yagoub, W., Nagabandi, A., Gupta, A., Levine, S. (2025). Steering your diffusion policy with latent space reinforcement learning. In *Proceedings of the 9th Conference on Robot Learning*.

Xiao, W., Lin, H., Peng, A., Xue, H., He, T., Xie, Y., Hu, F., Wu, J., Luo, Z., Fan, L., Shi, G., Zhu, Y. (2026). Self-improving vision-language-action models with data generation via residual RL. In *Proceedings of the 14th International Conference on Learning Representations*.

Xiao, W., Xie, J., Zhang, T., Lin, H., Fu, L., Xue, H., Lu, J., Yang, Y., Dai, C., Wang, Z., Wu, J., Wang, G., Sastry, S. S., Goldberg, K., Fan, L., Zhu, Y., Shi, G. (2026). ENPIRE: Agentic robot policy self-improvement in the real world. *arXiv preprint arXiv:2606.19980*.

Xu, C., Springenberg, J. T., Equi, M., Amin, A., Esmail, A., Levine, S., Ke, L. (2026). RL token: Bootstrapping online RL with vision-language-action models. *arXiv preprint arXiv:2604.23073*.

Yuan, X., Mu, T., Tao, S., Fang, Y., Zhang, M., Su, H. (2025). Policy decorator: Model-agnostic online refinement for large policy model. In *Proceedings of the 13th International Conference on Learning Representations*.

Zhai, X., Mustafa, B., Kolesnikov, A., Beyer, L. (2023). Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11975–11986.

A Appendix

A.1 Pseudocodes for Rollout Collection and Cal-QL Updates

Algorithm 2: Rollout Collection

Function `collect_rollout` $(\theta, \mathcal{D}, \text{use_rec}, \boldsymbol{\xi})$ **:**

- $d_0, p, \theta_{\text{enc}}, \theta_{\text{VLA}}, l, T_{\text{timeout}} \doteq \boldsymbol{\xi}$
- initial image and proprioception $I_0, q_0 \sim d_0(\cdot, \cdot)$
- recovery $\leftarrow 0$
- **for** *time step* $t = 0, 1, 2, \cdots, T_{\text{timeout}} + 49$ **do**
  - $z_t \doteq f_{\theta_{\text{enc}}}(I_t)$
  - $\tilde{\theta}_t \doteq \begin{cases} \theta_{\text{VLA}}, & \text{if recovery} = 1, \\ \theta, & \text{otherwise} \end{cases}$
  - $a_t \sim \pi_{\tilde{\theta}_t}(\cdot \mid z_t, q_t, l)$
  - $I_{t+1}, q_{t+1}, r_{t+1} \sim p(\cdot, \cdot, \cdot \mid I_t, q_t, a_t)$
  - $\mathcal{D} \leftarrow \mathcal{D} \cup (I_t, q_t, a_t, r_{t+1}, I_{t+1}, q_{t+1})$
  - **if** I_{t+1}, q_{t+1} *is success* **then**
    - **return** 1
  - recovery $\leftarrow 0$
  - **if** use_rec $= 1$ ***and*** $t - T_{\text{timeout}} \in [0, 3] \cup [19, 26] \cup [42, 49]$ **then**
    - recovery $\leftarrow 1$
- **return** 0

Algorithm 3: Cal-QL Update

Function `Cal_QL_update` $(\mathcal{D}_{\text{LWP}}, \mathcal{D}_{\text{VLA}}, \boldsymbol{\Theta})$ **:**

shared memory parameters $\theta_{\text{LWP}}, \{\phi\}_{n=1}^{N}, \{\overline{\phi}\}_{n=1}^{N}, \alpha, \eta_\theta, \eta_\phi, \tau \doteq \boldsymbol{\Theta}$

for $i = 0, 1, 2, \cdots$ **do**

$\mathcal{M}_i \doteq \{x_j \sim \mathcal{D}_{\text{LWP}}\}_{j=1}^{\lceil 256r \rceil} \cup \{x_j \sim \mathcal{D}_{\text{VLA}}\}_{j=1}^{\lfloor 256(1-r) \rfloor}$

$\phi_{n,i+1} \doteq \phi_{n,i} - \eta_\phi \nabla_{\phi_n} J_Q^{\text{cal}}(\phi_{n,i}, \mathcal{M}_i) \quad \text{for } n \in \{1, \cdots, N\}, \qquad \text{using (2)}$

$\theta_{\text{LWP},i+1} \doteq \theta_{\text{LWP},i} + \eta_\theta \nabla_{\theta_{\text{LWP}}} J_\pi(\theta_{\text{LWP},i}, \mathcal{M}_i), \qquad \text{using (3)}$

$\overline{\phi}_{n,i+1} \doteq (1-\tau)\overline{\phi}_{n,i} + \tau\phi_{n,i} \quad \text{for } n \in \{1, \cdots, N\}$

$\theta_{\text{LWP}} \leftarrow \theta_{\text{LWP},i+1}, \phi_n \leftarrow \phi_{n,i+1}, \overline{\phi}_n \leftarrow \overline{\phi}_{n,i+1} \quad \text{for } n \in \{1, \cdots, N\}$

A.2 CAL-QL WITH Q ENSEMBLE

The Q function objective of Cal-QL $J_Q^{\text{cal}}(\phi_n, \mathcal{M}_i)$ is composed of two terms:

$$J_Q^{\text{cal}}(\phi_n, \mathcal{M}_i) \doteq J_Q^1(\phi_n, \mathcal{M}_i) + J_Q^2(\phi_n, \mathcal{M}_i). \tag{2}$$

The first term $J_Q^1(\phi_n, \mathcal{M}_i)$ is the squared semi-gradient TD error borrowed from SAC without entropy backup defined as

$$J_Q^1(\phi_n, \mathcal{M}_i) \doteq \mathbb{E}_{(s,a,r,s') \sim \mathcal{M}_i} \left[\left[r + \gamma \min_{j \in \mathcal{I}} Q_{\overline{\phi}_j}(s', a') - Q_{\phi_n}(s, a) \right]^2 \right],$$

where $\mathcal{I} \sim \text{Uniform}\left(\binom{[N]}{2} \right)$, $[N] \doteq \{1, \cdots, N\}$ and $a' \sim \pi_{\theta_{\text{LWP}}}(\cdot \mid s')$.

The second term $J_Q^2(\phi_n, \mathcal{M}_i)$ is the additional objective of Cal-QL to penalize Q extrapolation. We have modified the Cal-QL implementation to incorporate an ensemble of $N = 10$ critics. The second term is calculated as

$$\widetilde{Q}_{\phi_n}^{\text{rand}}(s) \doteq \left[Q_{\phi_n}\left(s, a_i^{\text{rand}}\right) - \ln\left(0.5^{|\mathcal{A}|}\right) \right]_{i=1}^{N}, \quad a_i^{\text{rand}} \sim \mathcal{U}\left([-1, 1]^{|\mathcal{A}|}\right),$$

$$\widetilde{Q}_{\phi_n}^{\text{curr}}(s) \doteq [\max(\widehat{Q}^\mu(s,a), Q_{\phi_n}(s, a_i^{\text{curr}})) - \ln \pi_{\theta_{\text{LWP}}}(a_i^{\text{curr}} \mid s)]_{i=1}^{N}, \quad a_i^{\text{curr}} \sim \pi_{\theta_{\text{LWP}}}(\cdot \mid s),$$

$$\widetilde{Q}_{\phi_n}^{\text{next}}(s) \doteq [\max(\widehat{Q}^\mu(s,a), Q_{\phi_n}(s, a_i^{\text{next}})) - \ln \pi_{\theta_{\text{LWP}}}(a_i^{\text{next}} \mid s')]_{i=1}^{N}, \quad a_i^{\text{next}} \sim \pi_{\theta_{\text{LWP}}}(\cdot \mid s),$$

$$\widetilde{Q}_{\phi_n}^{\text{OOD}}(s) \doteq \ln \sum_{g \in G} \sum_{k=1}^{N} \exp\left(\widetilde{Q}_{\phi_n,k}^{g}(s) \right), \qquad \text{with } G \doteq \{\text{rand}, \text{curr}, \text{next}\}$$

$$J_Q^2(\phi_n, \mathcal{M}_i) \doteq 0.5\, \mathbb{E}_{(s,a,r,s') \sim \mathcal{M}_i} \left[\widetilde{Q}_{\phi_n}^{\text{OOD}}(s) - Q_{\phi_n}(s, a) \right],$$

where $\widehat{Q}^\mu(s,a)$ is the discounted Monte Carlo return from the state-action pair (s,a) until the end of the corresponding episode and $\mathcal{A}$ is the action space.

Finally, the policy objective of Cal-QL is defined as

$$J_\pi(\theta_{\text{LWP}}, \mathcal{M}_i) \doteq \mathbb{E}_{(s,a,r,s') \sim \mathcal{M}_i} \left[\frac{1}{N} \sum_{n=1}^{N} Q_{\phi_n}(s, a^{\text{curr}}) - \alpha \ln \pi_{\theta_{\text{LWP}}}(a^{\text{curr}} \mid s) \right], \tag{3}$$

where $a^{\text{curr}} \sim \pi_{\theta_{\text{LWP}}}(\cdot \mid s)$. We do not use automatic tuning of the temperature α and use a fixed $\alpha = 0.01$ instead.

A.3 HYPERPARAMETERS

The hyperparameters used across all three tasks are shown in Table 2

Table 2: Hyperparameters

HYPERPARAMETER	VALUE
Discount factor (γ)	0.99
Actor learning rate (η_θ)	3×10^{-4}
Critic learning rate (η_ϕ)	1×10^{-4}
Fixed temperature (α)	0.01
Soft target update (τ)	0.005
Target entropy	-32
Learning starts	1500 steps
Critic warm-up	5000 steps
Image projection size	384
Hidden width	512
Hidden layers	3
Critic ensemble size (N)	10
Batch size	256
Actor updates per step	5
Critic updates per step	20
Action chunk size	8