LAYERED VALUE FORECASTS FOR ONLINE RL FINE-TUNING OF VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models provide strong priors for robot manipulation, but adapting them to a new task with online reinforcement learning (RL) is costly, since every episode has to be run on a real robot. Part of this cost comes from the reward, which is sparse and arrives only when the task succeeds. A decision such as closing the gripper on an object is credited only through the value of the states that follow it, and that value stays near zero until the policy completes the task from them, so the critic separates a good decision from a bad one only once success is already frequent. Intermediate events, in contrast, occur in failed episodes as well as successful ones. To exploit this, we propose Layered Value Forecasts (LVF), which augments the critic of an off-policy actor-critic with event-based general value functions (GVFs). Each GVF predicts the discounted probability that one event occurs, such as a grasp or a collision, and is learned from binary event labels with minimal supervision. Their prediction at the chosen action enters the critic's TD target as a cumulant alongside the terminal reward, so the critic receives credit at the steps leading to each event. Across three real-robot manipulation tasks, LVF improves most where success is rare, reaching 95.0% and 70.0% on the two tasks where the initial policy succeeds in 32.5% of trials.

1 INTRODUCTION

Vision-language-action models (VLAs) can perform a wide range of manipulation tasks, but on any given task they are only as good as the demonstrations they were trained on (Brohan et al., 2023; Kim et al., 2025; Black et al., 2024). Reinforcement learning can take them further, but the models are large, and RL needs more rollouts than a physical robot can provide (Li et al., 2026; Chen et al., 2025a; Wang et al., 2026). A group of methods avoid this cost by keeping the VLA frozen and training only a small actor-critic on top of it. Some learn a residual that is added to the VLA's predicted action (Yuan et al., 2025; Xiao et al., 2026), some fine-tune a lightweight action head (Chen et al., 2025b), some act in the latent noise space of a flow-based action expert to steer which action it generates (Wagenmaker et al., 2025), and some read a compact token out of the VLA's internal features and use it as the state representation for RL (Xu et al., 2026). In all of them the VLA supplies perception and a strong action prior, and the small module makes the fine adjustments. This keeps the sample budget low enough so that a robot can improve on a high-precision task in a few hours of practice. Appendix A discusses these methods and other related work in more detail.

These approaches still rely on receiving a single sparse reward at the end of each episode. Consider picking up a book and sliding it into a rack. The grasp is credited only through the value of the next states, and that value is nonzero only once the policy succeeds from there, so while success is rare, the TD target at the grasp is near zero regardless of the quality of the grasp. Existing methods address this by putting a person in the loop, who either designates the critical segment of the task (Xu et al., 2026) or corrects the robot mid-episode (Luo et al., 2025) so that more trajectories succeed. In both cases the operator acts on the robot during training. Human object manipulation, in contrast, relies on anticipating contact events such as a grasp or a collision before they happen and on adjusting when a prediction fails (Johansson & Flanagan, 2009). The success of putting the book in the rack depends on its grip and whether it avoids the divider. Both factors can be predicted by the robot's sensory input before the episode ends.

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.

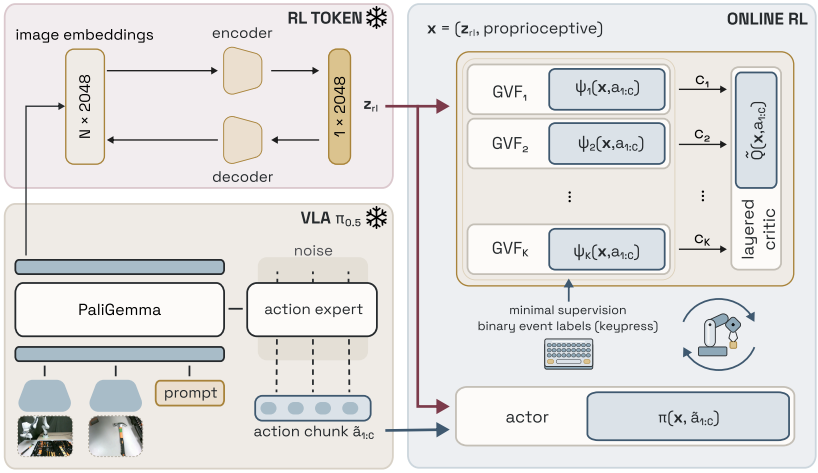


Figure 1: **Overview of LVF.** A frozen VLA ($\pi_{0.5}$) encodes images and the task prompt, and an encoder-decoder bottleneck compresses its final-layer embeddings into a single RL token that serves as the state for online RL (Xu et al., 2026). On top of this state we train a set of event GVFs $\psi_1, \ldots, \psi_K$, each predicting the discounted probability that one task event occurs under the current policy, such as a grasp or a collision. Their weighted predictions form an additional cumulant in the TD target of the task critic $\tilde{Q}$, so the critic's target carries a nonzero signal at every transition and not only at the end of the episode.

Reinforcement learning has a standard form for such predictions, the general value function (GVF) (Sutton et al., 2011), which predicts what the agent will observe if it follows a given policy from its current state. Formally, a GVF is the discounted sum of a chosen signal, the cumulant, under that policy up to a termination condition. The probability that the current grasp will hold is one such GVF, and so is the probability that the book will hit the rack. Each terminates when its event occurs. Because a GVF itself is a signal that the agent can observe, one prediction can serve as the cumulant of another (Sutton & Tanner, 2004; Schlegel et al., 2021; Ring, 2021). The task can then be represented by a set of forecasts rather than a single value.

We propose LVF, which adds a GVF for every intermediate event the task depends on. These GVFs learn from the same transitions as the task critic, and their predictions enter the critic's target along with the terminal reward. Because an event can be observed in both failed and successful trajectories, its GVF can learn over a shorter horizon and from every episode. The critic therefore receives credit at the event even when few episodes succeed. A lightweight actor, conditioned on the VLA's action chunk, is trained against this critic.

We evaluate LVF on three real-world and eleven LIBERO tasks. In both settings, the benefit of event forecasts is greatest when the initial policy is weak and decreases as the base success increases, with LVF showing improvement over the same actor-critic without forecasts for the less successful policies and matching its performance for the stronger ones. Using the same labels directly as rewards does not achieve this gain. In the real-robot setting, the cumulants of the event GVFs are provided by an operator, who presses a key when an event occurs and never takes control of the robot. Discovering events from the agent's own experience is left to future work.

2 Preliminaries

Vision-language-action models. We build on $\pi_{0.5}$ (Black et al., 2025), which is a VLA with a vision-language backbone following PaliGemma (Beyer et al., 2024) that encodes a sequence of camera images $I_t^1, \ldots, I_t^n$, a language instruction ℓ, and the robot's tokenized proprioceptive state q_t into a shared $N \times 2048$ token sequence. The action expert is a separate 300M-parameter transformer which attends unidirectionally to this backbone sequence, and is trained via flow matching to map a chunk of partially-denoised actions to the corresponding vector field. At inference time, the action expert outputs continuous action chunks $a_{t:t+H-1}$.

Markov decision process and off-policy actor-critic. We formulate robot control as an MDP $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$. At each timestep t, the agent observes a state $s_t \in \mathcal{S}$ and selects an action $a_t \in \mathcal{A}$, where $\mathcal{A}$ is a continuous action space. The environment transitions to s_{t+1} according to $p(s_{t+1} \mid s_t, a_t)$ and gives a reward $r(s_t, a_t)$, and the goal of learning is to adjust π to maximize the expected return $\mathbb{E}_\pi[G_t \mid s_t = s, a_t = a] = \mathbb{E}_\pi\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots \mid s_t = s, a_t = a\right]$, where $\gamma \in [0, 1)$ is the discount factor. We assume access only to a sparse binary terminal reward, with $r_T = 1$ on success, $r_T = 0$ otherwise, and $r_t = 0$ at every other timestep.

We use TD3 (Fujimoto et al., 2018), an off-policy actor-critic method, to learn π. Transitions (s_t, a_t, r_t, s_{t+1}) are stored in a replay buffer $\mathcal{B}$, from which mini-batches are sampled to update the actor and critic. The critic $Q_\varphi(s_t, a_t) \approx Q^\pi(s_t, a_t) \doteq \mathbb{E}_\pi[G_t \mid s_t = s, a_t = a]$ is fit by minimizing the squared temporal-difference error on transitions sampled from $\mathcal{B}$,

$$\mathcal{L}_Q(\varphi) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim \mathcal{B}}\Big[\big(r_t + \gamma\, \mathbb{E}_{a' \sim \pi_\theta(\cdot \mid s_{t+1})}\big[Q_{\varphi'}(s_{t+1}, a')\big] - Q_\varphi(s_t, a_t)\big)^2\Big], \tag{1}$$

where $Q_{\varphi'}$ is a target copy of the critic. The actor π_θ is optimized to select actions that maximize the critic's estimated values,

$$\mathcal{L}_\pi(\theta) = -\,\mathbb{E}_{s_t \sim \mathcal{B},\, a \sim \pi_\theta(\cdot \mid s_t)}\big[Q_\varphi(s_t, a)\big]. \tag{2}$$

In our formulation, the actor and critic operate over action chunks $a_{t:t+C-1} \in \mathbb{R}^{C \times d}$ instead of single-timestep actions.

General value functions. A general value function (GVF) (Sutton et al., 2011; Ring, 2021) consists of two parts. The *behavior* is specified by an option (π, β), where π is a policy and $\beta(s) \in [0, 1]$ is the probability that the option terminates at s. The *outcome* is specified by a pair (c, z), where $c(s, a, s') \in \mathbb{R}$ is a cumulant received on every transition and $z(s) \in \mathbb{R}$ is a terminal value received when the option terminates. Informally, a GVF predicts how much of c the agent will accumulate by following π before the option terminates, and what terminal value z it will receive. It is defined as

$$\psi(s, a) = \mathbb{E}_{s' \sim P(\cdot \mid s, a),\, a' \sim \pi(\cdot \mid s')}\big[c(s, a, s') + \beta(s')\, z(s') + (1 - \beta(s'))\, \psi(s', a')\big]. \tag{3}$$

The action-value function is one such prediction among many the agent could make about its own sensorimotor stream. Under a sparse terminal reward, Q^π is the GVF with option (π, β), where $\beta(s) = 1$ at terminal states and $1 - \gamma$ elsewhere, and outcome $(c = 0, z)$, where $z(s_T) = r_T$ and $z = 0$ elsewhere. The geometric termination $1 - \gamma$ plays the role of discounting, and all of the value comes from the terminal value.

3 Layered Value Forecasts

This section presents LVF. We augment the task critic with one GVF per sub-task event and add their predictions to the critic's cumulant. Section 3.1 formalizes online RL on top of a pretrained VLA on a real robot. In Section 3.2 we define the event GVFs, which are used in Section 3.3 to create the layered critic. Section 3.4 gives the training pipeline. Figure 1 summarizes the method.

3.1 Online RL in the Real World

Online RL on the full VLA is impractical on a real robot (Li et al., 2026; Chen et al., 2025a; Wang et al., 2026). The model has a 3B-parameter vision-language backbone (Beyer et al., 2024) and a 300M-parameter action expert (Black et al., 2025), and its representation, a sequence of $N \times 2048$ embeddings, is too large to learn from with a realistic amount of robot data. We follow RLT (Xu et al., 2026) and run RL on a compressed representation.

The VLM part of the frozen VLA turns each observation into a sequence of embeddings. A small encoder-decoder compresses this sequence into a single vector, $\mathbf{z}_{rl} \in \mathbb{R}^{2048}$, called the RL token. We train it on demonstrations to reconstruct the embedding sequence from $\mathbf{z}_{rl}$. Because the actor and critic use this vector instead of the full sequence, they can be small networks, while the vector retains what the backbone learned from large-scale web and robot data.

After this stage, the VLA and the encoder-decoder are frozen, and only two small networks, an actor and a critic, learn from online interaction. Their input is $\mathbf{x} = (\mathbf{z}_{rl}, \mathbf{s}^p)$, the RL token concatenated

with the proprioceptive state of the robot $\mathbf{s}^p$. Each action is a chunk $a_{1:C}$ of C consecutive low-level actions, executed by the agent in full before the actor is queried again, and we index actions relative to the chunk throughout this section. The reward is given at the end of the episode, $r_T = 1$ for success and $r_T = 0$ for failure, and all previous rewards are zero.

Data collection and storage. During collection, the VLA outputs a chunk of H actions at each chunk boundary, and the actor takes its first $C \leq H$ actions, $\tilde{a}_{1:C}$, as an extra input. After each episode we re-encode the episode offline on a stride-2 grid, recomputing $\mathbf{z}_{rl}$ and querying the VLA for $\tilde{a}_{1:C}$ at every second step, and extract one chunk transition per grid point. A transition stored at step t records the C actions the robot actually executed from t, together with the state it reached at $t + C$. Subsequent transitions therefore overlap by $C - 2$ steps. We write $\mathbf{x}'$ for the state at $t + C$ and $\tilde{a}'_{1:C}$ for the reference chunk stored with it.

The actor is queried only at chunk boundaries, but transitions are stored on a grid finer than the chunk. A grid point away from a boundary falls in the middle of a chunk, so the C actions that follow it are the rest of the chunk the actor sampled at the last boundary and the start of the next chunk. The stored $a_{1:C}$ is therefore a behavior action and not a draw from $\pi_\theta(\cdot \mid \mathbf{x}, \tilde{a}_{1:C})$, since the actor was never queried at $\mathbf{x}$.

Critic. $Q_\varphi(\mathbf{x}, a_{1:C})$ approximates the discounted return of executing $a_{1:C}$ from $\mathbf{x}$ and following π_θ afterward. It is fit by minimizing the squared TD error on chunk-level transitions from the replay buffer $\mathcal{B}$,

$$\mathcal{L}_Q(\varphi) = \mathbb{E}_{(\mathbf{x}, a_{1:C}, r_{1:C}, \mathbf{x}') \sim \mathcal{B}} \Big[\big(\hat{Q} - Q_\varphi(\mathbf{x}, a_{1:C}) \big)^2 \Big], \tag{4}$$

with the target

$$\hat{Q} = \sum_{t'=1}^{C} \gamma^{t'-1} r_{t'} + \gamma^C \, \mathbb{E}_{a' \sim \pi_\theta(\cdot \mid \mathbf{x}', \tilde{a}'_{1:C})} \big[\bar{Q}_{\varphi'}(\mathbf{x}', a') \big], \tag{5}$$

where $Q_{\varphi'}$ is a target copy of the critic, updated after each gradient step as $\varphi' \leftarrow \zeta\varphi + (1 - \zeta)\,\varphi'$ with $\zeta \in (0, 1]$. The target action a' is always drawn from π_θ, so Q_φ still estimates the value of π_θ, and the off-boundary starts widen the action coverage the critic is fit on. To reduce overestimation, the critic is an ensemble of five heads whose target bootstraps from the minimum over a random subset of two (Chen et al., 2021). The heads are fit independently on the same target by Eq. (4). Throughout the paper, an overbar denotes this minimum over an ensemble's heads. For the five-head critics it is taken over a random subset of two heads, and for the GVFs of Section 3.2 it is taken over both heads.

Actor. The actor is Gaussian with a learned mean and a fixed covariance, $\pi_\theta(a_{1:C} \mid \mathbf{x}, \tilde{a}_{1:C}) = \mathcal{N}(\mu_\theta(\mathbf{x}, \tilde{a}_{1:C}), \sigma^2 I)$. We sample from π_θ during data collection and execute the mean μ_θ at evaluation. The actor loss follows RLT (Xu et al., 2026),

$$\mathcal{L}_\pi(\theta) = \mathbb{E}_{(\mathbf{x}, \tilde{a}_{1:C}) \sim \mathcal{B},\, \epsilon \sim \mathcal{N}(0, I)} \Big[- Q_\varphi(\mathbf{x}, \mu_\theta + \sigma\epsilon) + \eta \|\mu_\theta - \tilde{a}_{1:C}\|_2^2 \Big], \tag{6}$$

where μ_θ is short for $\mu_\theta(\mathbf{x}, \tilde{a}_{1:C})$ and the second term is a behavior-cloning (BC) penalty toward the reference chunk with coefficient $\eta > 0$. A larger η keeps the actor closer to the VLA, and a smaller one lets the critic move it further. The actor maximizes the prediction of a single critic head, and its reference-chunk input is dropped with probability 0.5 during training.

3.2 Event Forecasts

Manipulation skills in humans are naturally planned around events such as the fingers closing on an object or the object contacting a surface (Johansson & Flanagan, 2009). The nervous system predicts the tactile signal each event should produce and compares it against the signal that arrives. We use GVFs to make these predictions. Let $\mathcal{E} = \{e_1, \ldots, e_K\}$ be a set of K events for a task, where each event e_i defines a GVF ψ_i that predicts whether e_i will occur if the agent follows π_θ from the current state. Events can be desired, such as a secure grasp, or to be avoided, such as a collision with the book rack. We write $\rho_i = +1$ for a desired event and $\rho_i = -1$ for one to avoid.

Each chunk transition in $\mathcal{B}$ stores a binary mark $m_i(\mathbf{x}') \in \{0, 1\}$ for each event, equal to 1 if e_i occurs within the chunk ending at $\mathbf{x}'$. Each ψ_i is evaluated under the same policy π_θ as the task

critic and differs from it only in when it terminates and what it returns. Each ψ_i has behavior (π_θ, β_i), evaluated at the successor state as in Eq. (3), with

$$\beta_i(\mathbf{x}') = \begin{cases} 1 & \text{if } m_i(\mathbf{x}') = 1 \text{ or } \mathbf{x}' \text{ is terminal,} \\ 1 - \gamma_\psi^C & \text{otherwise,} \end{cases} \tag{7}$$

and outcome $(c_i = 0,\ z_i(\mathbf{x}) = m_i(\mathbf{x}))$. The option ends at the event, returning $z_i = 1$, or through either geometric termination or the end of the episode, returning $z_i = 0$. By Eq. (3), its value is the discounted probability that e_i occurs,

$$\psi_i(\mathbf{x}, a_{1:C}) = \mathbb{E}_{\pi_\theta}\big[(\gamma_\psi^C)^{\,k_i - 1} \mid \mathbf{x}, a_{1:C}\big], \tag{8}$$

where $k_i = \lceil \tau_i / C \rceil \geq 1$ is the index of the action chunk in which e_i first occurs, τ_i is the number of environment steps until e_i, and γ_ψ is the per-step discount of the event GVFs, which can be smaller than the task discount γ. We set $(\gamma_\psi^C)^{\,k_i - 1} = 0$ if e_i does not occur before the episode ends. The GVF predicts at the chunk level, so ψ_i counts chunks to the event rather than steps, and a chunk that makes e_i more likely has a higher value.

Because the marks are recorded in every episode, whether it succeeds or fails, each ψ_i learns from both. On the real robot an operator provides the marks during collection (Section 4.1), whereas in simulation they are read from simulator state (Appendix C). We take the events as given and study how to use their predictions.

Training. Each ψ_i is a separate network with the same architecture and inputs as the task critic, with two heads. It is trained on the same chunk transitions by minimizing $\mathbb{E}_{\mathcal{B}}\big[(\hat{\psi}_i - \psi_i(\mathbf{x}, a_{1:C}))^2\big]$, with the target

$$\hat{\psi}_i = m_i + (1 - m_i)\, \gamma_\psi^C\, \mathbb{E}_{a' \sim \pi_\theta}\big[\bar{\psi}_i'(\mathbf{x}', a')\big], \tag{9}$$

where ψ_i' is a target copy of ψ_i. A mark anywhere in the chunk counts as the event occurring at the chunk boundary, so the target is not discounted within the chunk. As in Eq. (5), the bootstrap uses the same next action a' and is dropped when the episode ends, so $\hat{\psi}_i = 0$ at a terminal transition where e_i is not marked, matching $\beta_i = 1$ with $z_i = 0$ there. The critic gets only the target copies $\bar{\psi}_i'$, which carry no gradient, so Eq. (9) is the only loss that trains ψ_i.

3.3 The Layered Critic

Under sparse reward, all of Q^π's value comes from the terminal value, which reaches the contact states only after many bootstrap steps and only from episodes that succeed. When success is rare, little credit is assigned to the contact states. We therefore keep the behavior and terminal value of Q^π and add a cumulant, so that the critic accumulates event predictions along the way,

$$c(\mathbf{x}, a_{1:C}) = \sum_{i=1}^{K} w_i\, \bar{\psi}_i'(\mathbf{x}, a_{1:C}). \tag{10}$$

This is added once to the target of each stored chunk transition. The weight $w_i = \rho_i \lambda_i$ scales the contribution of ψ_i by a coefficient $\lambda_i > 0$ and signs it by ρ_i. The resulting value function, which we write $\tilde{Q}_\varphi$, is trained by Eq. (4) with $\tilde{Q}_\varphi$ in place of Q_φ and the target

$$\hat{\tilde{Q}} = \sum_{t'=1}^{C} \gamma^{t'-1} r_{t'} \ + \ c(\mathbf{x}, a_{1:C}) \ + \ \gamma^C\, \mathbb{E}_{a' \sim \pi_\theta}\big[\bar{\tilde{Q}}_{\varphi'}(\mathbf{x}', a')\big]. \tag{11}$$

The middle term is the only addition to Eq. (5). The terminal reward and the bootstrap are as in RLT, and the bootstrap is still dropped at terminal transitions. The actor loss is Eq. (6) with $\tilde{Q}$ in place of Q. Unlike the terminal reward, the cumulant is collected at every transition. One event therefore enters many backups, and what it adds over an episode can outweigh the terminal reward. We control this with λ_i and sweep it in Appendix I.

$\tilde{Q}_\varphi$ is not an estimate of task return. Its cumulant is not a difference of potentials, so the policy that maximizes $\tilde{Q}_\varphi$ can differ from the one that maximizes task return (Ng et al., 1999). Each ψ_i terminates when e_i occurs, which changes the objective in two ways. For a desired event, ψ_i rises

as the event becomes more likely, so a chunk that moves toward a grasp receives value without completing it. For an event to be avoided, ψ_i drops to zero once the event has occurred, so the negative contribution ends at the collision rather than persisting after it. Smaller λ_i shrinks both effects. Together with the BC term, which holds the actor near the VLA over a short interaction budget, this keeps the actor improving the reference policy locally rather than converging to the optimum of $\tilde{Q}_\varphi$.

3.4 Implementation

The actor, the task critic, and each ψ_i are two-layer MLPs of width 256. The critic and the GVF heads take $(\mathbf{x}, a_{1:C})$, and the actor takes $(\mathbf{x}, \tilde{a}_{1:C})$. We use $C = 10$, $\gamma = 0.99$, $\gamma_\psi = 0.95$, and $\zeta = 0.005$ for all target copies. The replay buffer holds every chunk transition collected online with its event marks and the episode's success flag. Algorithm 1 gives the training loop and Appendix E the remaining hyperparameters. The GVF heads and the event marks are used only during training. At evaluation the policy runs μ_θ, so the deployed system is the same size as the RLT baseline.

Algorithm 1 Online RL with event forecasts

Freeze the VLA and encoder-decoder.
Initialize θ, φ, $\{\psi_i\}_{i=1}^K$, and target copies.
for each episode **do**
  $\mathcal{T} \leftarrow \emptyset$ ▷ event timestamps
  for each chunk boundary t **do**
    Encode $\mathbf{x}_t$; query VLA for $\tilde{a}_{1:C}$.
    Execute $a_{1:C} \sim \pi_\theta(\cdot \mid \mathbf{x}_t, \tilde{a}_{1:C})$.
    Append event marks in the chunk to $\mathcal{T}$.
  end for
  Observe the terminal reward r_T.
  for each step t on the stride-2 grid **do**
    Encode $\mathbf{x}_t$; query VLA for $\tilde{a}_{1:C}$.
    Set $m_{1:K}$ from $\mathcal{T} \cap [t, t{+}C)$.
    Store $(\mathbf{x}_t, a_{1:C}, r_{1:C}, \mathbf{x}_{t+C}, \tilde{a}'_{1:C}, m_{1:K})$.
  end for
  for each gradient step **do**
    Sample a batch from $\mathcal{B}$.
    Update each ψ_i toward Eq. (9).
    Form the cumulant from ψ'_i, Eq. (10).
    Update $\tilde{Q}_\varphi$, Eq. (11).
    Update π_θ, Eq. (6).
    Soft-update all target copies with rate ζ.
  end for
end for

4 Experiments

4.1 Real-World Experiments

Tasks. We use three tasks (Figure 3), chosen so that the SFT policy is already competent on one and weak on the other two. In *book-short*, the robot picks up a book and inserts it into a narrow slot of a book rack, between two dividers. *Book-long* chains two placements, in which the robot places one book, then picks up a second and inserts it into the adjacent slot. Since the episode succeeds only if both books are placed, the terminal reward arrives only after the second insertion. In *hammer in toolbox*, the robot picks up a hammer, places it into its foam cutout, and pushes it down until it seats, which replaces slot insertion with a press fit. A grasp event fires in all tasks when the object leaves the table in the gripper, and in the book tasks a collision event fires when the book hits the rack. The hammer task has two further events, a collision event when the hammer hits another object and a placement event when the hammer first enters its cutout.

Setup. We run all experiments on a Franka arm with two cameras at a 20 Hz control rate. After fine-tuning $\pi_{0.5}$ on 75 teleoperated demonstrations per task, we run online RL for 100 episodes per task, training each method once. An operator provides the terminal success label and the binary event labels, marking each event with a key press when it occurs, without controlling the robot. We use the same values as in simulation on every task, $\lambda = 0.3$ and $\eta = 0.2$ for LVF and $\eta = 0.1$ for RLT, and evaluate each policy on 40 trials from fixed initial positions, executing the actor mean.

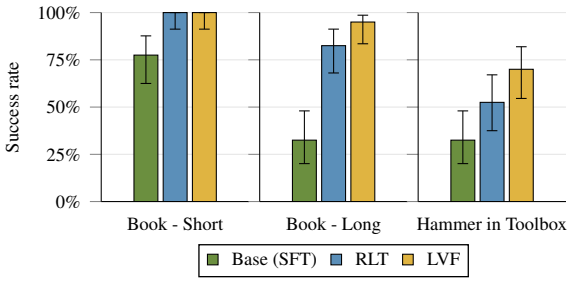


Figure 2: Real-world success rates over 40 evaluation trials per policy, with 95% intervals. Both RL methods run for 100 online episodes with task-independent hyperparameters, with $\eta = 0.2$ for LVF and $\eta = 0.1$ for RLT. Both methods reach 100% on book-short, where the SFT policy is already competent. LVF is ahead on the two tasks where the SFT policy succeeds in a third of trials.

Results. Figure 2 reports success rates with 95% Wilson intervals. Both RL methods improve on the SFT policy on every task. On book-short, where the SFT policy already

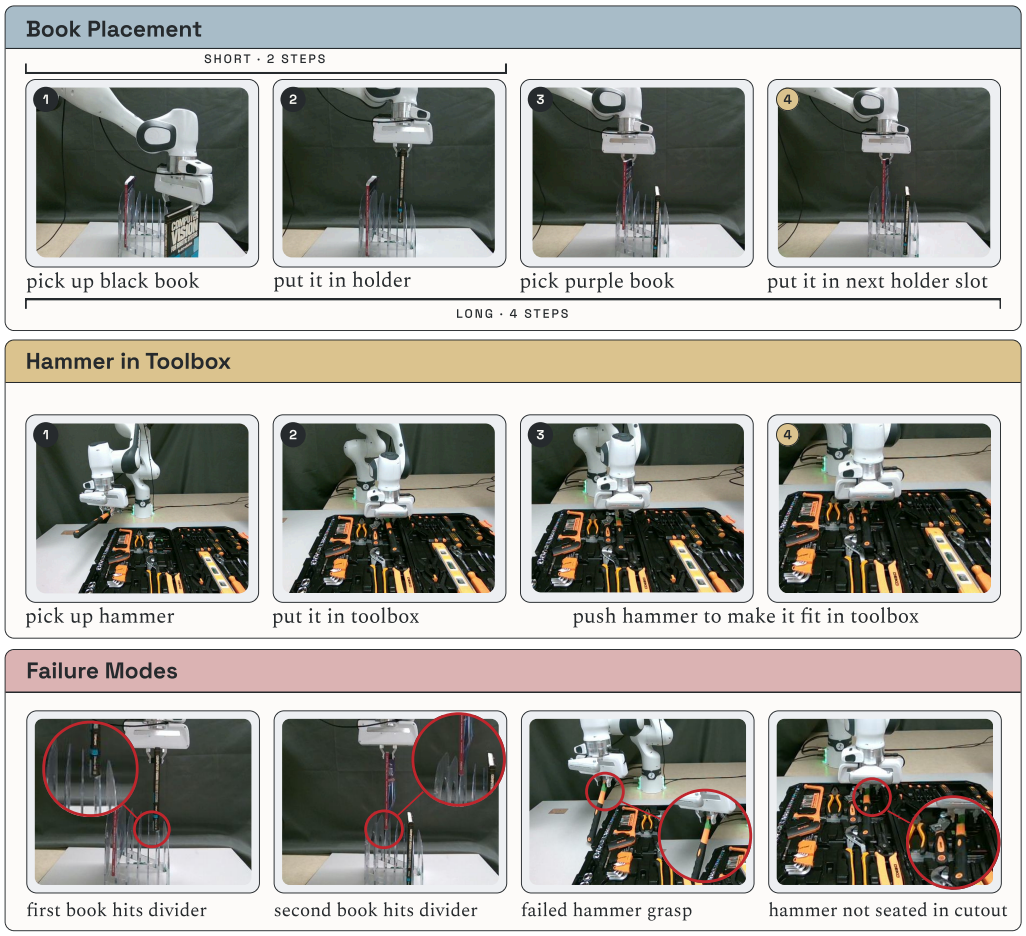


Figure 3: Real-world tasks. *Book-short* requires inserting a book into a narrow slot of a rack. *Book-long* chains two placements, with the terminal reward given only after the second. *Hammer in toolbox* requires placing a hammer into its foam cutout and pressing it down until it seats. The bottom row shows typical failures, a book hitting a divider, a failed hammer grasp, and a hammer that does not seat in its cutout.

succeeds in 77.5% of trials, RLT and LVF both reach 100%. LVF is ahead on the two tasks where the SFT policy succeeds in 32.5% of trials, reaching 95.0% against 82.5% for RLT on book-long and 70.0% against 52.5% on hammer in toolbox. With 40 trials per policy, the intervals on these two tasks overlap. Therefore, we conducted more experiments in simulation, where the same pattern appears over three seeds per task (Section 4.2).

4.2 Simulation Experiments

Setup. A $\pi_{0.5}$ policy fine-tuned on the full LIBERO demonstrations already succeeds on nearly every episode (Physical Intelligence, 2025; Chen et al., 2025a; Xiao et al., 2026), which leaves no room to measure what online RL adds. We therefore fine-tune $\pi_{0.5}$ on only two demonstrations per task and run online RL from the resulting policy, which gives base policies with a wide range of success rates. Task selection (Appendix B) gives eleven tasks across LIBERO-Goal, LIBERO-Spatial and LIBERO-Long. For a fair comparison, all base success rates we report come from evaluating the SFT policy with the same procedure as the RL methods. Events are read from simulator state, giving $K = 4$ channels, namely contact, grasp, open and collision (Appendix C). As on a real robot, where several rollouts cannot run in parallel, each iteration collects only a single episode. We report success after 140 episodes rather than training until the policy is perfect, because the question is whether online RL can improve the policy within a limited interaction budget, and the evaluation is done over 48 trials.

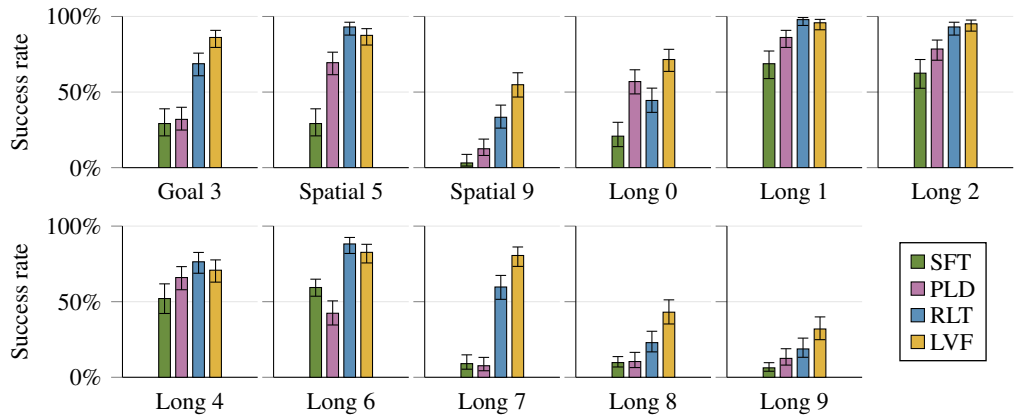


Figure 4: Success rate on LIBERO after 140 episodes. RLT ($\eta = 0.1$) and LVF ($\eta = 0.2$, $\lambda = 0.3$) use the same settings on every task and are pooled over three seeds (per-seed counts in Appendix F). Error bars are 95% intervals over episodes.

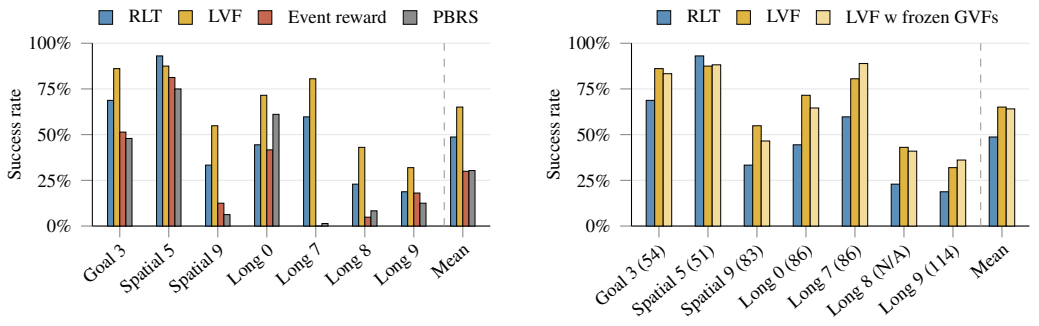
We compare against RLT (Xu et al., 2026) and PLD (Xiao et al., 2026). Our RLT baseline uses the RL token and actor-critic of Xu et al. (2026) without its human-designated critical segment. It shares the backbone, networks and optimizer with LVF and differs only in η and λ. PLD trains a residual RL policy on top of the base policy and distills its data back by supervised fine-tuning. It uses a larger budget of 250 000 online environment steps and additional offline rollouts, against at most 69 600 steps for RLT and LVF. All methods run on three seeds per task. We sweep η per arm on two pilot tasks, Goal 3 and Spatial 5, and use the resulting values on every task, 0.1 for RLT and 0.2 for LVF (Appendix D). We sweep λ on two pilot tasks and use the single value that is better on average across them, $\lambda = 0.3$, on every task and on the real robot (Appendix I).

Results. Figure 4 reports success at 140 episodes. RLT and LVF improve over the base policy on every task, and the gain from event forecasts decreases as base success increases (Spearman $\rho = -0.68$ over the eleven tasks, Figure 6). Where the base policy succeeds in more than half of trials, on four tasks, the two methods are close, 88.9% for RLT and 86.1% for LVF. On the seven tasks where the base SFT policy succeeds in under a third of trials, LVF reaches 65.1% against 48.7% for RLT. PLD, in turn, performs below LVF on all eleven tasks and below RLT on ten, and on Long 6 and Long 7 it falls below the base policy it is initialized from. A policy that rarely succeeds collects mostly unrewarded episodes, yet it still grasps objects in many of them. In that setting the event forecasts provide a learning signal that the terminal reward does not. Mean episode lengths are reported in Appendix H.

Table 1: η sweep at 140 episodes over 48 trials, single seed. Base is the few-shot SFT policy evaluated with the same procedure as RL methods (Table 7). η is swept independently for each arm.

η	Goal 3	Spatial 5
Base $\pi_{0.5}$	0.292	0.292
RLT ($\lambda = 0$)		
0.00	0.000	0.000
0.02	0.750	0.583
0.05	**0.812**	0.729
0.10	0.792	**0.979**
0.20	0.625	0.646
0.30	0.375	0.708
LVF ($\lambda = 0.3$)		
0.00	0.000	0.000
0.05	0.938	0.792
0.10	0.812	**0.979**
0.20	**0.958**	0.958
0.30	0.917	0.917
0.40	0.646	0.854

Do the event labels alone explain the gain? The event labels could be used directly as rewards, and we test two such variants with the same labels as LVF (Figure 5a). *Event reward* adds each mark to the reward with LVF's signs and weights, and *PBRS* uses the marks as a potential-based shaping term (Ng et al., 1999). Its potential and a sweep of its weight are given in Appendix J. Averaged over the seven low-base tasks, both reach about 30% success, below RLT at 48.7% and LVF at 65.1%, and on Long 7 both nearly stop succeeding. PBRS is designed to preserve the optimal policy, yet it fails as well. A change of objective therefore does not explain why event labels fail as rewards. The variants differ from LVF in how the labels enter the critic. An event reward enters the target only at the transition where the event occurs, whereas LVF adds a learned forecast of each event, evaluated at the chosen action, to the target at every transition.



(a) Event forecasts against event rewards. Event reward adds the marks to the reward with LVF's signs and weights; PBRS uses them as a potential-based shaping term.

(b) GVFs frozen after 20 successful episodes, with no event labels afterward. Parentheses give the mean freeze episode. In Long 8, it did not reach 20 successes.

Figure 5: Ablations on how LVF uses event labels, at 140 episodes and pooled over three seeds.

Does LVF need event labels throughout training? Although our supervision is only a binary signal, requiring an operator to mark events during online collection still limits the autonomy that makes RL attractive. We therefore stop event labels once the policy has collected 20 successful episodes. At that point the event GVFs and their target copies are frozen, and the task critic keeps reading their predictions as its cumulant (Figure 5b). The GVFs froze on every task except Long 8, which never reached 20 successes, with the mean freeze episode ranging from 51 (Spatial 5) to 114 (Long 9). Averaged over the seven low-base tasks (Figure 5b), frozen LVF reaches 64.1% success, against 65.1% with labels throughout and 48.7% for RLT. Labels were still needed for a large part of training. Once learned, however, the forecasts no longer needed supervision, since a GVF fit on a limited number of labeled episodes kept supplying a useful cumulant for the rest of training, matching continued labeling. This suggests that the event forecasts are a simpler learning problem than the task value, and the operator's role could be reduced further, which we leave for future study.

How much does LVF rely on the VLA's action chunk? The BC coefficient η sets how strongly the actor is pulled toward the VLA's reference chunk. Because the event cumulant supplies 65% to 74% of the critic's value (Table 8) and changes its scale relative to the BC term, we sweep η separately for each arm on the two pilot tasks, on a single seed (Table 1). The larger critic values under LVF call for a stronger BC term, so its sweep range is shifted toward larger η. Both arms fail with $\eta = 0$, so neither can discard the reference. LVF is, however, less sensitive to the choice of η. On the four nonzero settings swept for both arms (0.05, 0.1, 0.2 and 0.3), it exceeds 0.9 success in six of eight cases, and RLT in one. On Spatial 5, moving one setting away from the best η reduces RLT's success from 0.979 to 0.646 and LVF's from 0.979 to 0.792.

5 Conclusion and Limitations

We proposed LVF, which adds event-based GVFs to the critic of an off-policy actor-critic for online RL fine-tuning of VLAs. Each GVF's prediction at the chosen action enters the critic's TD target, and because events occur in failed episodes too, the GVFs train from every episode rather than from the rare successful ones. On a real robot and eleven LIBERO tasks, LVF matches the same actor-critic when the initial policy often succeeds and improves on it when success is rare. Used as rewards, directly or through potential-based shaping, the same labels fall below the actor-critic without them on average over the seven low-success tasks, and shaping fails on Long 7 and Long 8 at every weight we tried. In simulation, forecasts frozen after 20 successful episodes kept supplying a useful cumulant without further labels.

LVF depends on event labels. We choose the events by hand, and on the real robot an operator marks them in every training episode, which requires knowing which events matter for a task. The event cumulant also changes the objective. It is not potential-based, so a policy that maximizes the layered critic does not necessarily maximize the task success. The weight λ limits this effect, but a large λ can pull the policy away from the task (Appendix I).

AI USE STATEMENT

We used Claude (Anthropic) to implement the baseline methods from their public repositories and papers, to give feedback on the design of our experiments, to write and edit code, to create figures, to draft and edit parts of the paper, to check grammar, and to review the literature and identify relevant work.

We did not use generative AI tools for brainstorming, for the ideation of the method, or to propose or refine hypotheses. We also did not use them to develop the conceptual framework or to formulate mathematical claims. Synthetic data generation, proofs, translation, and qualitative data analysis are not applicable to this work.

The authors reviewed all AI-assisted work. We edited all AI-assisted text and figures. We verified the baseline implementations against the original code and papers, and tested all AI-assisted code before running experiments. The authors made all decisions on experimental design, and read every paper identified with AI assistance before citing it. We take responsibility for the final content of this work produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 3 and Algorithm 1 specify the method, and Appendix E lists all hyperparameters for RLT, LVF and PLD. Appendices B–D describe task selection, the simulation event predicates and the evaluation protocol, and Appendix F gives per-seed results. The simulation experiments use the public LIBERO benchmark and the $\pi_{0.5}$ checkpoint from openpi (Physical Intelligence, 2025), and we will release the full code for both the simulation and real-world experiments after the review period.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement – residual RL for precise assembly. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

Jose A. Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. RUDDER: Return decomposition for delayed rewards. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, Thomas Unterthiner, Daniel Keysers, Skanda Koppula, Fangyu Liu, Adam Grycner, Alexey Gritsenko, Neil Houlsby, Manoj Kumar, Keran Rong, Julian Eisenschlos, Rishabh Kabra, Matthias Bauer, Matko Bošnjak, Xi Chen, Matthias Minderer, Paul Voigtlaender, Ioana Bica, Ivana Balazevic, Joan Puigcerver, Pinelopi Papalampidi, Olivier Henaff, Xi Xiong, Radu Soricut, Jeremiah Harmsen, and Xiaohua Zhai. PaliGemma: A versatile 3b VLM for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

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. *arXiv preprint arXiv:2504.16054*, 2025.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Xiang Li, Bingwen Wei, Jiakai Zhou, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, Tiejun Huang, Yu Wang, and Chao Yu. $\pi_{\texttt{rl}}$: Online rl fine-tuning for flow-based vision-language-action models. *arXiv preprint arXiv:2510.25889*, 2025a.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. Randomized ensembled double Q-learning: Learning fast without a model. In *International Conference on Learning Representations (ICLR)*, 2021.

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. In *Proceedings of Robotics: Science and Systems, RSS 2025, Los Angeles, CA, USA, Jun 21-25, 2025*, 2025b. doi: 10.15607/RSS.2025.XXI.019.

Sam Devlin and Daniel Kudenko. Dynamic potential-based reward shaping. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2012.

Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. Variational inverse control with events: A general framework for data-driven reward definition. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, 2018.

Marek Grześ. Reward shaping in episodic reinforcement learning. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pp. 565–573, São Paulo, Brazil, 2017. International Foundation for Autonomous Agents and Multiagent Systems.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations (ICLR)*, 2017.

Roland S. Johansson and J. Randall Flanagan. Coding and use of tactile signals from the fingertips in object manipulation tasks. *Nature Reviews Neuroscience*, 10(5):345–359, 2009.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R 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. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 06–09 Nov 2025. URL `https://proceedings.mlr.press/v270/kim25c.html`.

Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Yang Zhaohui, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, et al. Simplevla-rl: Scaling vla training via reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 125846–125869, 2026.

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. *arXiv preprint arXiv:2505.18719*, 2025.

Junjie Lu, Xinyao Qin, Yuhua Jiang, Kaixin Wang, Chuheng Zhang, Bin Liang, Jun Yang, Min Xu, and Li Zhao. Unified noise steering for efficient human-guided VLA adaptation. *arXiv preprint arXiv:2605.10821*, 2026.

Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *Science Robotics*, 10(105):eads5033, 2025.

Mingyang Lyu, Yinqian Sun, Erliang Lin, Huangrui Li, Ruolin Chen, Feifei Zhao, and Yi Zeng. Reinforcement fine-tuning of flow-matching policies for vision-language-action models. *arXiv preprint arXiv:2510.09976*, 2025.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: Towards universal visual reward and representation via value-implicit pre-training. In *International Conference on Learning Representations (ICLR)*, 2023.

Yecheng Jason Ma, Joey Hejna, Ayzaan Wahid, Chuyuan Fu, Dhruv Shah, et al. Vision language models are in-context value learners. In *International Conference on Learning Representations (ICLR)*, 2025.

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. In *International Conference on Learning Representations*, volume 2026, pp. 36352–36372, 2026.

Joseph Modayil, Adam White, and Richard S. Sutton. Multi-timescale nexting in a reinforcement learning robot. *Adaptive Behavior*, 22(2):146–160, 2014.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML)*, pp. 278–287, 1999.

Omkar Patil, Ondrej Biza, Thomas Weng, Karl Schmeckpeper, Wil Thomason, Xiaohan Zhang, Robin Walters, Nakul Gopalan, Sebastian Castro, and Eric Rosen. You've got a golden ticket: Improving generative robot policies with a single noise vector. *arXiv preprint arXiv:2603.15757*, 2026.

Physical Intelligence. openpi: Open-source models and code for the π series of vision-language-action models. `https://github.com/Physical-Intelligence/openpi`, 2025.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $\pi^*_{0.6}$: A VLA that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.

Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Van de Wiele, Volodymyr Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing – solving sparse reward tasks from scratch. In *International Conference on Machine Learning (ICML)*, 2018.

Mark Ring. Representing knowledge as predictions (and state as knowledge). *arXiv preprint arXiv:2112.06336*, 2021.

Matthew Schlegel, Andrew Jacobsen, Zaheer Abbas, Andrew Patterson, Adam White, and Martha White. General value function networks. *Journal of Artificial Intelligence Research*, 70:497–543, 2021.

Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

Sumedh Sontakke, Jesse Zhang, Séb Arnold, Karl Pertsch, Erdem Bıyık, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. RoboCLIP: One demonstration is enough to learn robot policies. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Richard S. Sutton and Brian Tanner. Temporal-difference networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2004.

Richard S. Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M. Pilarski, Adam White, and Doina Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 761–768, 2011.

Shuhan Tan, Kairan Dou, Yue Zhao, and Philipp Krähenbühl. Interactive post-training for vision-language-action models. *arXiv preprint arXiv:2505.17016*, 2025.

Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.

Vivek Veeriah, Matteo Hessel, Zhongwen Xu, Janarthanan Rajendran, Richard L. Lewis, Junhyuk Oh, Hado van Hasselt, David Silver, and Satinder Singh. Discovery of useful questions as auxiliary tasks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2025. arXiv:2506.15799.

Siting Wang, Xiaofeng Wang, Zheng Zhu, Minnan Pei, Xinyu Cui, Cheng Deng, Jian Zhao, Guan Huang, Haifeng Zhang, and Jun Wang. π-stepnft: Wider space needs finer steps in online rl for flow-based vlas. *arXiv preprint arXiv:2603.02083*, 2026.

Eric Wiewiora, Garrison Cottrell, and Charles Elkan. Principled methods for advising reinforcement learning agents. In *International Conference on Machine Learning (ICML)*, 2003.

Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Yuqi Xie, Fengyuan Hu, Jimmy Wu, Zhengyi Luo, Linxi Fan, Guanya Shi, and Yuke Zhu. Self-improving vision-language-action models with data generation via residual rl. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2511.00091.

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. *arXiv preprint arXiv:2604.23073*, 2026.

Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Mengke Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. In *International Conference on Learning Representations (ICLR)*, 2025.

A Related Work

A.1 Reinforcement Learning Fine-Tuning of VLAs

Methods for online RL on VLAs differ in which part of the model the RL update changes. We group them by whether RL updates the weights of the VLA or trains a small module on top of a frozen VLA.

A.1.1 Fine-Tuning the VLA Weights

One line of work applies policy-gradient RL to the whole VLA. Early methods target autoregressive VLAs such as OpenVLA, where the likelihood of an action is available from the token distribution (Lu et al., 2025; Tan et al., 2025). Li et al. (2026) scale this approach with success-based rewards and many parallel simulated rollouts. Flow-based VLAs such as π_0 and $\pi_{0.5}$ generate actions by iterative denoising, so the likelihood of an action is intractable. π_{RL} makes it tractable in two ways, either by learning the noise injected at each denoising step or by converting the sampling ODE into an SDE (Chen et al., 2025a). Flow policy optimization replaces the likelihood ratio with the change in the conditional flow-matching loss (McAllister et al., 2026), and Lyu et al. (2025) apply it to π_0. π-StepNFT removes both the critic and the likelihood, and trains the action expert with step-wise supervision (Wang et al., 2026). $\pi^*_{0.6}$ uses a learned value function to decide which of the robot's own experience to imitate, rather than running online policy gradients (Physical Intelligence et al., 2025). These methods update the VLA's parameters and are usually trained on many parallel simulated episodes. A single physical robot cannot supply this many rollouts.

A.1.2 Small Actor-Critics on a Frozen VLA

The second line keeps the VLA frozen and trains a small module with off-policy RL. The VLA supplies perception and an action prior, and the module learns to correct the prior. The methods differ in where the module connects to the VLA.

Steering the input: noise-space RL. A flow or diffusion-based policy maps a noise sample to an action. DSRL trains a small actor that chooses this noise, so RL acts in the latent noise space while the generator stays fixed (Wagenmaker et al., 2025). Every action it produces is therefore one the frozen policy can generate. UniSteer inverts the frozen flow decoder to turn human corrective actions into noise targets, which supervise the same noise actor that RL trains (Lu et al., 2026). Patil et al. (2026) show that a single noise vector can already improve a generative robot policy. The Flow-Noise variant of π_{RL} also learns a noise network, but it updates the weights of the VLA as well, so we list it with the first group.

Correcting the output: residual and action-head RL. Residual RL adds a learned correction to the action of a fixed base policy (Silver et al., 2018). Policy Decorator trains a small residual policy with sparse rewards on top of large imitation policies and restricts its exploration so that the combined policy stays close to the base (Yuan et al., 2025). ResiP applies residual RL to high-precision assembly (Ankile et al., 2025). PLD trains a residual specialist with RL, uses it to collect data from the states the base policy reaches, and distills this data back into the VLA by supervised fine-tuning (Xiao et al., 2026). Chen et al. (2025b) fine-tune a lightweight action head. RLT reads a compact RL token out of the VLA's internal features and trains a small actor that takes the VLA's action chunk as a reference (Xu et al., 2026). LVF uses the same RL token, actor and BC term, and changes only the target of the critic.

All of these methods learn from a sparse terminal reward. When the base policy rarely succeeds, few episodes carry any reward, and improvement is slow. Lu et al. (2026) report this for DSRL, which rarely collects a successful trajectory when initial success is low. Existing methods address this by adding a person who acts on the robot. In HIL-SERL (Luo et al., 2025) and UniSteer, an operator takes over and corrects the robot mid-episode. In RLT, the operator designates the critical segment of the task. In LVF, the operator only marks when an event occurs and never controls the robot.

A.2 General Value Functions

A GVF extends the value function to predictions about any signal the agent observes (Sutton et al., 2011). Horde learns many GVFs in parallel from a single stream of experience, each with its own policy, cumulant and termination. Off-policy TD learning lets all of them learn from the behavior of one agent. On a robot, GVFs have been used to predict sensor readings at several time scales (Modayil et al., 2014). This line of work treats the agent's knowledge of the world as a collection of such predictions (Sutton et al., 2011; Ring, 2021). Our event GVFs are predictions of this kind. Each one predicts whether an event will occur under the current policy, and the signal it predicts is a binary event mark.

Because a GVF's prediction is itself a signal, it can serve as the cumulant of another GVF. TD networks were the first to learn predictions about predictions (Sutton & Tanner, 2004). GVF networks use a set of such predictions as the recurrent state of the agent (Schlegel et al., 2021). Ring (2021) composes forecasts in layers, where the forecasts of one layer are the signals predicted by the next. LVF uses two layers. The event GVFs form the first layer and predict the event marks. The task critic forms the second and takes their predictions as its cumulant. Unlike in GVF networks, the predictions do not form the agent's state, which is the RL token, and the event GVFs do not feed each other.

GVFs as auxiliary tasks. In deep RL, GVF-like predictions are often trained as auxiliary tasks. UNREAL adds auxiliary control and reward-prediction tasks that share the agent's representation (Jaderberg et al., 2017). Veeriah et al. (2019) learn the cumulants and discounts of auxiliary GVFs with meta-gradients. In both, the predictions help only through the shared representation, and their values never enter the main value target. LVF takes the opposite route. Each ψ_i is a separate network, and the critic reads only their target copies, which carry no gradient. The event forecasts therefore affect the critic only through its TD target.

A.3 Reward Shaping and Event-Based Rewards

Reward shaping adds a term to the reward to guide learning. Ng et al. (1999) show that a shaping term of the form $\gamma\Phi(s') - \Phi(s)$ leaves the optimal policy unchanged. Wiewiora et al. (2003) extend this to potentials over state-action pairs, called look-ahead advice, and Devlin & Kudenko (2012) to potentials that change over time. Our PBRS baseline uses this form with a potential built from the event marks (Appendix J). The LVF cumulant is not a difference of potentials, so LVF does not inherit this guarantee (Section 3.3).

Reward machines specify a task as an automaton over high-level events and give reward on its transitions (Toro Icarte et al., 2018). A labeling function maps each state to the events that hold in it, which plays the role of our event marks. Reward machines use the events to define the reward and to decompose the task. They also require the order of events to be specified in advance. LVF takes a set of events without an order and uses them only as prediction targets. The reward remains the terminal success signal.

How LVF differs from shaping. The event-reward and PBRS baselines use the same labels as LVF, and averaged over seven tasks both fall below RLT (Figure 5a). The difference lies in how the label enters the critic. An event reward enters the TD target only at the transition where the event occurs. To reach the chunk that made the grasp likely, it still has to travel back through the bootstrap, as the terminal reward does. LVF instead trains a separate GVF on each event. This GVF terminates at the event, so its target reaches the preceding chunks over a short horizon. Its prediction at the chosen action then enters the critic's target at every transition. PBRS preserves the optimal policy and still fails on the two low-success tasks at every weight we tried (Appendix J), so a change of objective does not explain the gap. LVF gives up this guarantee. Its cumulant changes the objective, and λ limits how far (Appendix I).

A.4 Auxiliary Events and Learned Rewards

SAC-X (Riedmiller et al., 2018) is the closest prior use of events for sparse-reward manipulation. It defines auxiliary tasks from sensor events such as touch, grasp and lift, learns a separate intention policy for each, and schedules among them to collect data from which the sparse main task is learned off-policy. The auxiliary events therefore help through exploration, and the main critic still bootstraps only from the task reward. LVF learns no auxiliary policies and needs no scheduler. It keeps a single actor near the VLA, and uses the events only as prediction targets whose forecasts enter the main critic's target. VICE (Fu et al., 2018) also defines rewards through events, learning a classifier of the probability that an event holds from examples of it. The event then defines the task, whereas in LVF the terminal success signal defines the task and the events only supply a cumulant. Each ψ_i also differs from such a classifier in being conditioned on the policy and the action, so it predicts whether the event will occur if the agent follows π_θ after the chosen chunk, not whether it holds in the current state. A second line learns a dense reward for the task itself. HIL-SERL (Luo et al., 2025) trains a binary success classifier as the reward. VIP (Ma et al., 2023) and RoboCLIP (Sontakke et al., 2023) derive rewards from pretrained visual or video-language representations, GVL (Ma et al., 2025) prompts a VLM to estimate task progress, and $\pi^*_{0.6}$ (Physical Intelligence et al., 2025) learns a value function to select which experience to imitate. These methods estimate progress toward success, and when success is rare they must rely on examples of it or on priors from pretraining. The events in LVF occur in failed episodes as well, so their forecasts can be learned before success is frequent. The two approaches are complementary, since a learned progress reward could replace the terminal reward in Eq. (11) while the event cumulant is kept.

A.5 Credit Assignment under Sparse Reward

Other methods address sparse reward by changing how credit reaches earlier steps. RUDDER (Arjona-Medina et al., 2019) learns to predict the return of an episode and redistributes it to the steps that change the prediction. Hindsight experience replay relabels failed episodes with goals the agent did reach, so that a goal-conditioned agent can learn from them (Andrychowicz et al., 2017). RUDDER still learns from the return, so it can find the decisive steps only once some episodes succeed. Hindsight relabeling needs a goal space in which a failed episode achieved something. LVF

needs neither. An event such as a grasp is observed in failed episodes, so its forecast can learn before success is frequent.

B TASK SELECTION

RL fine-tuning is only informative on tasks where the base policy has room to improve. We therefore select tasks by the success rate of the few-shot SFT policy rather than by hand.

We fine-tune $\pi_{0.5}$ on two demonstrations per task for all forty tasks in LIBERO-Object, LIBERO-Goal, LIBERO-Spatial and LIBERO-Long. Tasks were selected using an initial standalone evaluation over 50 trials per task, keeping every task with success under 80%, since a task the policy already solves cannot measure the gains from RL. We include tasks with very low success, because they test the regime where the terminal reward is rarely observed. This selects eleven tasks, LIBERO-Goal 3, LIBERO-Spatial 5 and 9, and LIBERO-Long 0, 1, 2, 4, 6, 7, 8 and 9. We exclude LIBERO-Long 5, where the SFT policy never succeeds and none of the methods we evaluate obtained a single success during training.

Table 2 is used only for task selection. We later re-evaluated the selected tasks and all LIBERO-Long tasks with the action chunking and episode horizon used for RL, and the table reports these numbers where available. Selection used the initial evaluation. For a fair comparison, every base success rate reported with the RL results instead comes from evaluating the SFT policy with exactly the same procedure as the RL methods (Appendix D). Because the two evaluations use different seeds, they can differ, for example 44% against 59.4% on Long 6.

Table 2: Success rate (%) of the few-shot SFT policy on each LIBERO task over 50 standalone trials. LIBERO-Goal 3, LIBERO-Spatial 5 and 9, and all LIBERO-Long tasks use the action chunking and episode horizon used for RL. The remaining tasks show the initial evaluation. Selected tasks are in bold. This table is used only for task selection.

Suite	0	1	2	3	4	5	6	7	8	9	Mean
LIBERO-Object	100	100	100	100	98	98	100	80	100	100	97.6
LIBERO-Goal	84	98	92	**18**	82	92	98	100	100	88	85.2
LIBERO-Spatial	100	96	100	84	96	**24**	96	100	84	**2**	78.2
LIBERO-Long	**36**	**62**	**76**	98	**70**	0	**44**	**8**	**8**	**8**	41.0

C EVENT CHANNELS IN SIMULATION

In simulation we read events from the simulator state rather than from operator annotation, so the labels are noise-free and the experiments isolate the effect of the cumulant from the effect of label quality. Each task defines predicates over the MuJoCo state. At every step the environment evaluates them and returns a binary vector alongside the observation, which is stored with the transition and used as the event marks m_i in Eq. (9).

Table 3: Event channels used in simulation. Each channel fires on the rising edge of its predicate, at most once per episode. No channel encodes task success.

Channel	Predicate
contact	Any gripper geom in contact with any geom of the object of interest.
grasp	At least one geom in each of the left and right fingerpad groups in contact with the object.
open	The articulated state of a non-movable target, such as a drawer or microwave door, or, failing that, its powered state.
collision	Gripper contact with any fixture the task never references.

The **grasp** predicate is bilateral fingerpad contact, not force closure and not a lift. It fires whenever both pads touch the object, whether or not the object is subsequently held, so a pinch that immediately slips registers as a grasp event. This is a weaker condition than the grasp event an operator marks on the robot.

The predicate set is derived from the parsed task definition alone. The first declared movable object yields **contact** and **grasp**. Each non-movable target yields an **open** candidate, and fixtures the task never references yield **collision**. On tasks with no openable target, **open** is kept as a permanently zero column so that the column layout is the same on every task, and its GVF trains on an all-zero target.

D SIMULATION PROTOCOL DETAILS

η is swept for RLT and for LVF on two pilot tasks, LIBERO-Goal task 3 and LIBERO-Spatial task 5, on a single seed (Table 1). All main results in Figure 4 then use $\eta = 0.1$ for RLT and $\eta = 0.2$ for LVF on every task, including the pilot tasks. No arm is tuned per task. PLD uses the hyperparameters of Xiao et al. (2026), except the residual scale, which we reduce from 0.5 to 0.1 because training collapsed at 0.5 (Appendix E). PLD also receives more interaction than the other methods. It collects up to 200 offline rollouts until it has 50 successes, then trains online for 250 000 environment steps. RLT and LVF run 5 warm-up and 140 online episodes of at most 480 steps each, or at most $69\,600$ steps.

The base policy in Figure 4 is the few-shot SFT policy evaluated with the same procedure as the RL methods, with learning disabled. On the first seven tasks it is pooled over two evaluation runs on one seed ($n = 96$). On Long 6, 8 and 9 it is pooled over two evaluation runs on each of three seeds ($n = 288$), and on Long 7 over one evaluation run on each of three seeds ($n = 144$). We use these numbers, not the standalone evaluation in Table 2, wherever base success is compared with RL results, so that all methods are evaluated identically. The two evaluations use different seeds, and Table 2 serves only to select tasks.

E HYPERPARAMETERS

Table 4 lists the settings shared by RLT and LVF, and Table 5 the settings that differ between arms. The RLT baseline instantiates and trains the same GVF heads with $\lambda = 0$, so the two arms have identical architectures and parameter counts, and differ only in the settings of Table 5. Until the critic warm-up ends, the actor is trained with the BC term only, and the event cumulant enters the critic target after the warm-up. Event labels are taken from the first rising edge of each predicate, with a lookback of two steps. Every task uses an action box fitted to its own demonstrations, from the 0.5 and 99.5 percentiles of each action dimension with a margin of 1.5. Table 6 lists the PLD settings.

Table 4: Hyperparameters shared by RLT and LVF in simulation.

Parameter	Value
Actor, critic and GVF networks	2 layers, width 256
Critic ensemble	5 heads, target minimum over a random subset of 2
GVF heads	2 per event
Optimizer	Adam, no gradient clipping
Actor / critic learning rate	1×10^{-4} / 3×10^{-4}
Batch size	256
Update-to-data ratio	5, at most 1000 updates per iteration
Policy delay	2
Task discount γ	0.99 per environment step
GVF discount γ_ψ	0.95
Target update rate ζ	0.005
Target policy noise	0.1, clipped at 0.3
Exploration noise σ	0.1
Reference chunk dropout	0.5
Action clip	1.0
Chunk length C	10 steps, transitions stored every 2 steps
Replay capacity	200 000
Minimum buffer size before updates	2 000 transitions
RL token width	2048
Critic warm-up	3 000 updates
Maximum episode length	480 steps
Iterations	5 warm-up, 140 online, one episode each
Evaluation	48 episodes at iterations 50, 95 and 140
Seeds	0, 1, 2

Table 5: Settings that differ between arms.

Parameter	RLT	LVF	LVF, frozen GVFs
BC coefficient η	0.1	0.2	0.2
Cumulant weight λ_i	0	0.3	0.3
GVF heads frozen	—	—	after 20 successes

Table 6: PLD hyperparameters. All other settings follow Xiao et al. (2026).

Parameter	Value
Residual scale	0.1 (training collapsed at 0.5)
Cal-QL α	0.5
Networks	3 layers, width 256
Learning rate	3×10^{-4}
Batch size / γ / target update rate	256 / 0.99 / 0.005
Update-to-data ratio / policy delay	4 / 1
Target entropy	0
Offline successes required	50, at most 200 rollouts
Online interaction budget	250 000 environment steps
Environments (offline / online)	8 / 1
Probing	hybrid, $\alpha = 0.6$

F PER-SEED RESULTS

Table 7 gives the successes of RLT and LVF on each seed and the success rate pooled over the three seeds. Tasks are grouped by base success as in Section 4.2. The variance across seeds is largest on the low-success tasks. On Spatial 9, for example, RLT succeeds in 10, 33 and 5 of 48 trials, and LVF in 11, 24 and 44.

Table 7: Successes out of 48 evaluation episodes per seed at 140 episodes, and pooled success over the three seeds ($n = 144$). Base is the SFT policy evaluated with the same procedure as the RL methods (Appendix D). Tasks are grouped by base success as in Section 4.2.

		RLT		LVF	
Task	Base (%)	Seeds 0/1/2	Pooled (%)	Seeds 0/1/2	Pooled (%)
Base success above 50%					
Long 1	68.8	48 / 47 / 46	97.9	48 / 45 / 45	95.8
Long 2	62.5	40 / 48 / 46	93.1	47 / 46 / 44	95.1
Long 4	52.1	37 / 40 / 33	76.4	30 / 32 / 40	70.8
Long 6	59.4	43 / 42 / 42	88.2	44 / 36 / 39	82.6
Base success below one third					
Goal 3	29.2	38 / 28 / 33	68.8	46 / 41 / 37	86.1
Spatial 5	29.2	47 / 42 / 45	93.1	46 / 34 / 46	87.5
Spatial 9	3.1	10 / 33 / 5	33.3	11 / 24 / 44	54.9
Long 0	20.8	21 / 26 / 17	44.4	36 / 34 / 33	71.5
Long 7	9.0	32 / 23 / 31	59.7	41 / 34 / 41	80.6
Long 8	9.7	12 / 11 / 10	22.9	18 / 22 / 22	43.1
Long 9	6.3	11 / 8 / 8	18.8	15 / 15 / 16	31.9

G GAIN AGAINST BASE SUCCESS

Figure 6 plots the difference in success between LVF and RLT against the base success of the SFT policy. The difference is largest where the base policy rarely succeeds and shrinks as base success increases (Spearman $\rho = -0.68$, $p = 0.022$).

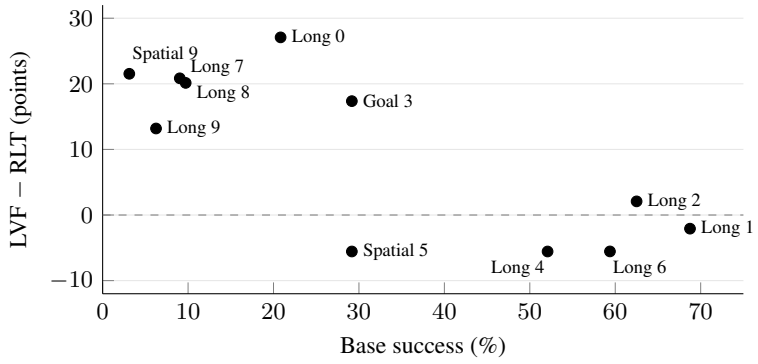


Figure 6: Difference in success between LVF and RLT at 140 episodes against base success of the SFT policy under the RL evaluation procedure, pooled over three seeds. Spearman $\rho = -0.68$ ($p = 0.022$).

H EPISODE LENGTH

Figure 7 reports the mean length of successful episodes for each method after 140 episodes. A shorter episode means the policy completed the task more directly. Only successful episodes are averaged, so the mean is noisier for methods with few successes.

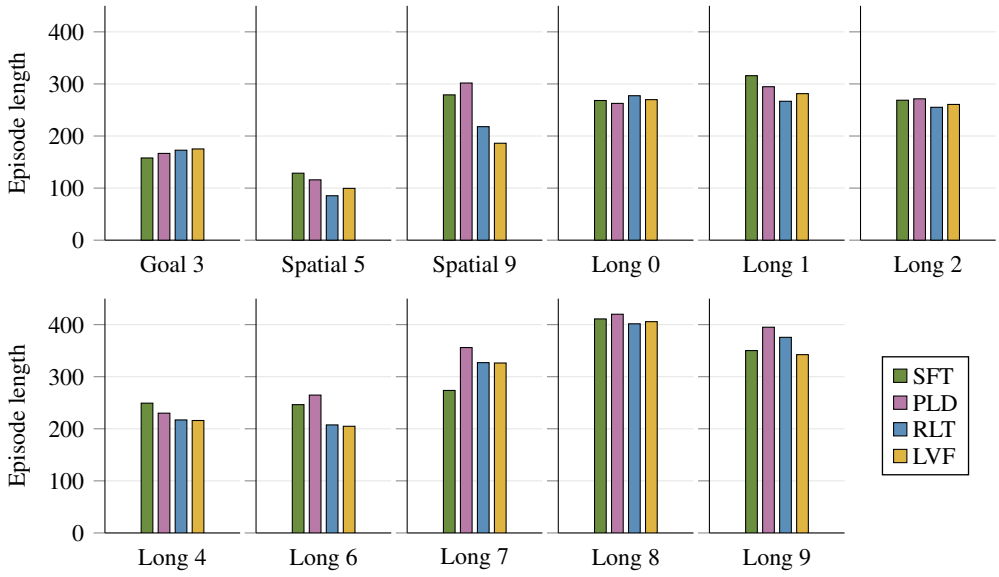


Figure 7: Mean episode length in environment steps on LIBERO after 140 episodes, for the runs in Figure 4. Shorter episodes indicate a policy that completes the task more directly. All are averaged over the episodes that were successful.

I FULL λ SWEEP

Table 8: λ sweep at 140 episodes over 48 trials, single seed, with η fixed at LVF's best value for each task (0.2 on Goal, 0.1 on Spatial). On Spatial this is also RLT's best η; on Goal RLT's own best is $\eta = 0.05$ (Table 1). $\lambda = 0$ is the RLT objective. B is the share of the critic's value from the event cumulant.

	Goal 3 ($\eta = 0.2$)		Spatial 5 ($\eta = 0.1$)	
λ	Succ.	B	Succ.	B
0.0	0.625	0.000	0.979	0.000
0.1	0.833	0.536	0.979	0.390
0.3	**0.958**	0.742	0.979	0.645
0.5	0.812	0.819	0.938	0.753
0.7	0.938	0.855	0.729	0.813

Table 8 fixes η at LVF's best value for each task, 0.2 on Goal and 0.1 on Spatial. On Spatial this is also RLT's best η, so the $\lambda = 0$ entry is RLT at its own optimum. On Goal it is RLT at LVF's η, which is not the value used for RLT in the main results, and should be read against the 0.812 that RLT reaches at $\eta = 0.05$ in Table 1 rather than as that method's best. The $\lambda = 0.3$ Spatial entry, however, is not the configuration behind LVF's main results. Table 8 sweeps λ on Spatial at $\eta = 0.1$ on a single seed, whereas LVF in Figure 4 uses $\eta = 0.2$ over three seeds and reaches 87.5% there.

Spatial is at 0.979 for every $\lambda \leq 0.3$, including $\lambda = 0$. Because RLT already reaches this level, the sweep on Spatial shows only that the cumulant does not reduce success for $\lambda \leq 0.3$. Goal separates, from 0.625 at $\lambda = 0$ to 0.958 at $\lambda = 0.3$.

Beyond the optimum, Goal is not monotone, with 0.812 at $\lambda = 0.5$ and 0.938 at 0.7. At $n = 48$ these differences are within the evaluation interval, so success on Goal does not depend clearly on λ in this range, whereas Spatial degrades at the top of it, reaching 0.729 at $\lambda = 0.7$ where the cumulant supplies 81% of the critic's value. We did not sweep past $\lambda = 0.7$ and so have not located the point where Goal breaks.

J SENSITIVITY OF POTENTIAL-BASED SHAPING TO THE SHAPING WEIGHT

A natural objection to LVF is that the event marks could enter the reward directly through potential-based reward shaping (PBRS) (Ng et al., 1999). PBRS adds $F(x, x') = \gamma^C \Phi(x') - \Phi(x)$ to the task reward. The marks themselves cannot serve as the potential. A mark is 1 only on the chunk where its event occurs, so a potential built from it would pay w at the event and take it back on the next transition. The potential therefore records which events have already occurred. Let $o_i(x) = 1$ if e_i has occurred by x in the current episode, and $o_i(x) = 0$ otherwise. We set $\Phi(x) = w \sum_i \rho_i \, o_i(x)$, with the signs ρ_i of Section 3.2. An event counts only when its predicate changes from false to true, so Φ is zero at reset, and each o_i does not decrease within an episode. An object that is grasped and then dropped still counts as grasped. Following Grześ (2017), Φ is zero at a successful terminal state. The shaping terms then sum to zero over a successful episode, and on a truncated episode the critic's bootstrap from the last state accounts for the remainder. The shaping weight w sets how strongly the events affect the critic.

We run the sweep on three LIBERO-Long tasks that span the range of base success. The base policy succeeds in 20.8% of trials on task 0 and in under 10% on tasks 7 and 8. All other hyperparameters match the RLT arm. Each setting uses three seeds with 48 evaluation episodes per seed, evaluated after 140 episodes of fine-tuning. Table 9 reports per-seed success counts and the success rate pooled over seeds ($n = 144$).

Table 9: PBRS shaping-weight sweep on LIBERO-Long. Entries are successes per seed out of 48, followed by the success rate pooled over three seeds. The $w = 0.3$ anchor is the run reported in Figure 5a. Base is the SFT policy evaluated with the same procedure as the RL methods (Appendix D), as in Table 7.

Task	Base VLA	$w = 0.1$	$w = 0.3$ (anchor)	$w = 1.0$
Long 0	20.8%	14, 15, 13 (29.2%)	[22, 40, 26] (61.1%)	40, 38, 42 (83.3%)
Long 7	9.0%	0, 0, 0 (0.0%)	[0, 1, 1] (1.4%)	0, 0, 0 (0.0%)
Long 8	9.7%	4, 5, 3 (8.3%)	[1, 6, 5] (8.3%)	0, 0, 0 (0.0%)

Whether shaping helps depends on the task. On task 0, shaping helps at every weight, rising with w from 29.2% at $w = 0.1$ to 61.1% at the anchor and 83.3% at $w = 1.0$. Shaping exceeds LVF on this task, 83.3% against 71.5%. On task 7 no weight produces more than two successes in 144 evaluation episodes. On task 8 every weight ends at or below the base policy's 9.7%, reaching 8.3% at $w = 0.1$ and at the anchor and 0.0% at $w = 1.0$.

These results show that the failure of shaping on low-success tasks is not a tuning artifact. No weight in the sweep recovers task 7 or task 8, and the largest weight eliminates task 8's remaining successes. Shaping helps on task 0, where the largest weight reaches 83.3%, and fails on the two tasks where the base policy almost never succeeds, so its effect is task-dependent at every weight we tried, while LVF improves over the base policy on all eleven tasks.