Dissecting Advantage-Guided Post-Training for Vision-Language-Action Policies

Jiahang Cao[1†⋆], Hanye Zhao[1†⋆], Hang Lai[2✉], Shenyu Zhang[2], Xiaoshen Han[1†], Xinghang Li[2], Futeng Liu[2], Wanli Peng[2], Heyun Wang[2], Yunhong Wang[2], Jason Li[2], Yong Yu[1], Weinan Zhang[1✉]

***Abstract*— Advantage-guided reinforcement learning provides a practical way to post-train vision-language-action (VLA) policies using limited robot data. However, its performance depends on several coupled choices, including how critic-derived advantages are constructed, calibrated, and used for policy training. Existing recipes often combine these choices into a single end-to-end procedure, making their individual effects difficult to identify. In this work, we dissect advantage-guided VLA post-training through a controlled empirical study that separates these design choices while accounting for their distinct estimands. We develop stage-specific offline evaluation methods to screen alternative choices efficiently, without requiring extensive real-robot policy evaluations for every possible combination. The staged evaluation identifies a modular recipe that combines temporal-difference advantage construction, group-wise calibration, and continuous advantage weighting. Across four real-world bimanual tasks, the resulting recipe improves mean task progress and success over the SFT initialization by 0.42 and 0.63, respectively. Moreover, the proposed evaluation diagnostics show an overall alignment with downstream real-world performance, supporting their use for interpreting empirical outcomes and selecting advantage-guided post-training designs in practice. Videos are available at our project website dissectvla.github.io.**

I. Introduction

Vision-language-action (VLA) models provide a general interface for language-conditioned robot control [1]–[4], but effectively refining these policies from limited and heterogeneous robot data remains challenging, particularly when feedback is sparse over long-horizon tasks. A post-training dataset may combine demonstrations, autonomous rollouts, and trajectory segments collected around human interventions, all generated by different behavior policies [5]–[7]. Such data contain successful behavior, uncorrected failures, and corrective or recovery actions, making it important to determine how different data samples should contribute to policy learning. Equal weighting may expose the actor to failures as if they were demonstrations, whereas overly aggressive filtering may discard corrective or recovery behavior. These considerations motivate methods that can modulate the contribution of individual samples during policy learning.

As one such approach, advantage-guided reinforcement learning (RL) has attracted growing interest and has been increasingly applied to VLA post-training [8]–[12]. By using critic-derived advantage to reweight or filter sampled action chunks, it can exploit mixed-quality experience without requiring the dataset to consist entirely of expert demonstrations or relying on extensive additional online interaction. Typically, an advantage-guided post-training pipeline involves three stages: advantage construction, advantage calibration, and advantage utilization. These stages address different questions: construction determines what quantity the advantage estimates, calibration determines how the advantage is made comparable across samples, and utilization determines how it shapes policy updates through weighting or filtering. Each stage also admits multiple design choices, yielding a large space of possible post-training recipes.

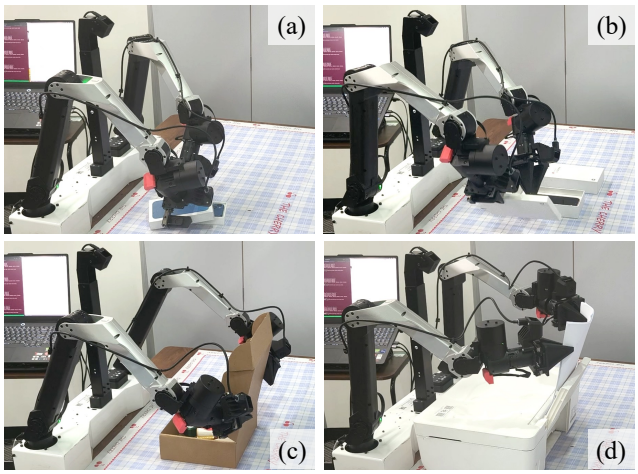

Fig. 1: Representative scenes from the four real-world manipulation tasks in our evaluation. (a) Pack Phone Easy, (b) Pack Phone Hard, (c) Pack Box, and (d) Refill Printer Paper.

Because choices across stages jointly shape how advantage influences policy learning, performance differences between existing end-to-end recipes cannot be reliably attributed to any individual stage. It remains unclear which choices within each stage are most effective, and why these choices are responsible for observed policy improvements. Changing one stage may also alter the sample weights, sample exposure, and effective sample size produced by the complete pipeline. Comparing many such combinations would require substantial policy training and evaluation, which is costly in real-robot settings [13], [14].

To this end, we dissect the post-training pipeline through a sequential three-stage evaluation protocol that screens design choices stage by stage before assessing their combined effect at the policy level. As shown in Fig. 2, stage I compares critic-training and advantage-estimation choices using offline sample-ordering diagnostics. Stage II fixes

⋆Equal contribution. ✉Corresponding author.
†Work done during internship at Xiaomi Robotics.
[1]School of Computer Science, Shanghai Jiao Tong University.
[2]Xiaomi Robotics.

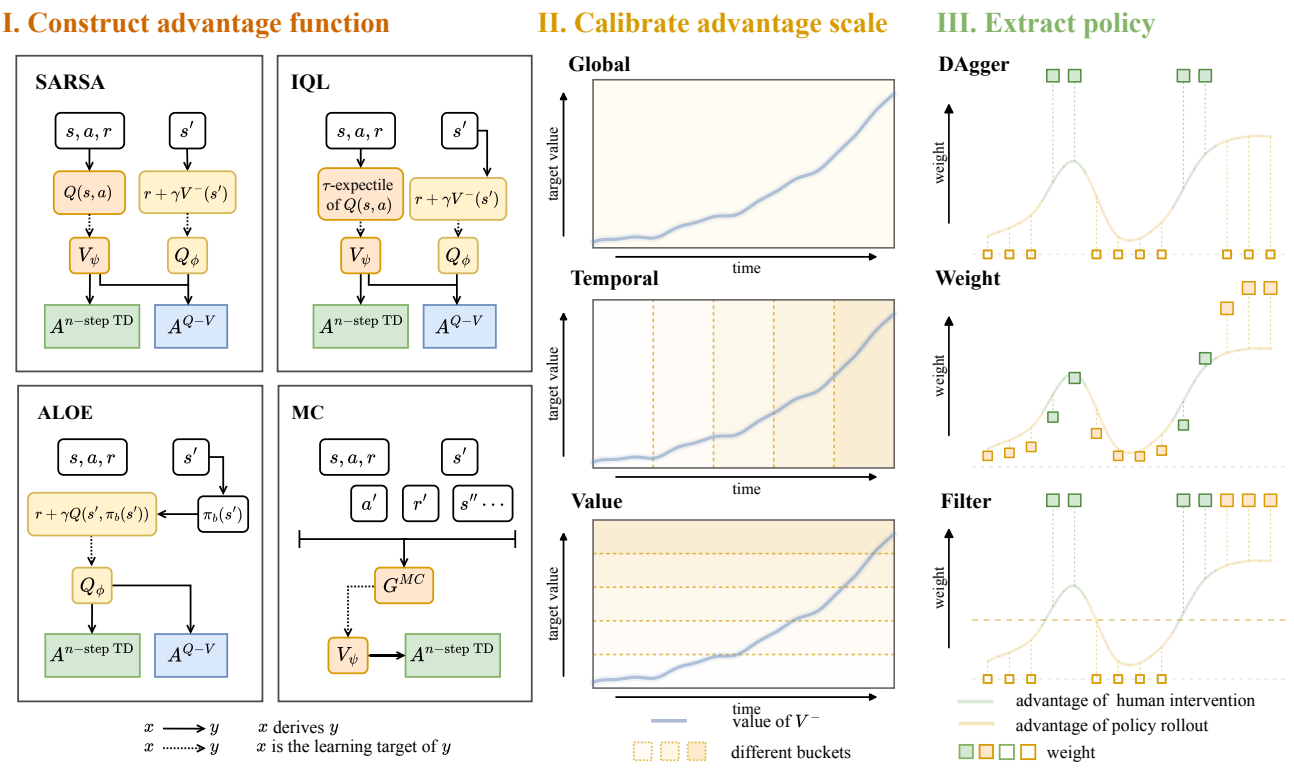

Fig. 2: Overview of our three-stage evaluation of advantage-guided VLA post-training. Stage I compares different advantage construction schemes, Stage II varies the grouping rule in advantage calibration, and Stage III examines three sample-weighting strategies in real-robot policy training.

the selected Stage-I configuration and compares calibration choices using offline distributional diagnostics. Stage III then fixes the selected construction and calibration configurations and evaluates their use in policy training using the $\pi_{0.5}$ flow-matching actor as a representative VLA policy backbone [4]. The first two stages provide offline screening evidence, whereas the third provides the policy-level evidence by comparing complete policy-training conditions. By propagating the selected configuration from one stage to the next, this protocol avoids retraining and evaluating every possible cross-stage combination on real robots. We apply the protocol to offline real-robot datasets and evaluate the resulting policies on four real-world manipulation tasks that span object insertion, long-horizon sequencing, multi-object interaction, and deformable-object handling (Fig. 1). This real-robot study complements prior empirical studies centered on simulation and cost-free settings [15]–[18].

Across the three stages, the results identify a consistent pattern for selecting and using advantages. In Stage I, n-step TD constructions outperform chance on both sample-ordering diagnostics, whereas $Q - V$ and Monte Carlo constructions are generally weaker. In Stage II, value-based calibration produces the largest average reduction in cross-sub-task advantage imbalance. With the selected construction and calibration, continuous advantage weighting performs best on both task progress and final task success. Real-world ablation experiments on alternative construction and calibration choices show an overall alignment between downstream policy performance and the corresponding offline diagnostics, supporting the use of these diagnostics to interpret and select post-training designs without exhaustively evaluating every possible combination.

In summary, this study makes three contributions:

- We develop a sequential evaluation protocol with stage-specific offline diagnostics to screen post-training choices and reduce costly real-robot evaluations.
- Guided by this protocol, we derive a simple modular post-training recipe that combines effective choices for advantage construction, calibration, and utilization.
- We demonstrate the recipe's effectiveness on real-world tasks, and illustrate a consistent relationship between offline diagnostics and real-world policy performance.

II. Related Work

VLA Post-Training. Adapting a pretrained VLA policy to task-relevant robot experience generally follows two complementary paradigms: supervised fine-tuning (SFT) and reinforcement fine-tuning (RFT). SFT methods optimize a supervised objective that directly matches the policy to demonstrated or corrected actions [2], [19], whereas RFT methods optimize the policy toward higher-reward behavior using reward or value feedback [8], [11], [15], [20]–[22]. Empirical comparisons further report improved semantic and execution robustness over SFT [16]. In real-robot settings, post-training data are often limited, and autonomous rollouts may contain failures from which the policy cannot recover without intervention [23]–[25]. Human takeover data are

therefore commonly used as corrective supervision, providing recovery trajectories and examples from task-relevant states. As an interactive SFT method, HG-DAgger [26] extends the dataset-aggregation principle of DAgger [27] to interactive human interventions during autonomous execution. HG-DAgger can provide corrective behavior for states encountered during execution, but its original formulation assigns uniform supervision to collected human-intervention data and does not directly exploit correct actions produced during autonomous execution. Our study also incorporates human-takeover data, but focuses on RFT, where critic-derived advantages can differentiate the training influence of heterogeneous dataset samples [10], [28].

Advantage-Guided Reinforcement Learning. Value or advantage estimates can provide non-uniform supervision for policy improvement [18], [29], [30]. Early offline RL methods mainly differ in how they estimate advantages and translate them into policy updates: IQL fits the state-value function by expectile regression on in-sample action values and performs exponentially advantage-weighted behavioral cloning using $Q - V$ [31], while AWR fits a state-value baseline to empirical Monte Carlo returns and applies exponential advantage weighting [32]. Building on this principle, CRR supports both binary and continuous critic-guided regression [33], and AWAC extends advantage-weighted updates from offline pretraining to online refinement [34]. Recent VLA studies adapt these ideas to high-dimensional, temporally extended control by constructing relative trajectory advantages [8], [9] or aligning temporal-difference learning with action chunking [28]. In particular, ALOE learns a Q-function with policy-sampled chunked TD targets for action-level off-policy evaluation, then derives $Q - V$ advantages for weighted policy improvement [10]. RECAP instead learns a distributional value function fitted to Monte Carlo returns and conditions the policy on a binarized advantage indicator [35]. Despite their effectiveness, these methods couple advantage construction and utilization under different data and interaction settings, making their individual effects difficult to isolate. Our work addresses this gap by separating construction, calibration, and utilization under a shared actor, dataset, and training budget.

III. Preliminaries

VLA policy post-training can be formulated as a Markov Decision Process (MDP), denoted by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{R}$ is the reward function, $\mathcal{P}$ denotes the transition dynamics, and γ is a discount factor. A transition sampled from the dataset is represented as $T = (s_t, \mathbf{a}_t, r_t, s_{t+1}, \mathbf{a}_{t+1})$. s_t denotes the policy input at decision step t, including the natural language instruction, visual observations, and robot state. $\mathbf{a}_t = (a_{t,0}, a_{t,1}, \ldots, a_{t,C-1})$ denotes the action chunk, where $a_{t,i} \in \mathcal{A}$ and C is the chunk length. The C primitive actions are executed consecutively before the next policy query, reducing the decision frequency. Consequently, s_{t+1} denotes the state reached C primitive timesteps after s_t. To avoid task-specific dense reward engineering and facilitate scalable post-training, we adopt a sparse reward such that $r_t = 1$ only for the terminal transition of a successful episode, and $r_t = 0$ otherwise. An indicator $d_t \in \{0, 1\}$ denotes whether the transition terminates an episode.

A. Advantage Construction

Advantage construction consists of critic learning followed by advantage estimation. For TD-based critics, the action-value function Q_ϕ is learned using the Bellman target:

$$\begin{aligned} y_t &= r_t + \gamma(1 - d_t)V_{\psi^-}(s_{t+1}), \\ \mathcal{L}_Q(\phi) &= \mathbb{E}_{T\sim\mathcal{D}}\left[\mathrm{MSE}(Q_\phi(s_t, \mathbf{a}_t), \mathrm{sg}[y_t])\right], \end{aligned} \tag{1}$$

where sg stops gradients and V_{ψ^-} denotes the target value network, whose parameters are updated as an exponential moving average of those of V_ψ. The learning of V_ψ is method-dependent: it may be fitted by expectile regression over Q-values [31] or, in the MC variant, directly to empirical returns G_t^{MC} [35]. Alternatively, the continuation value may be obtained from Q at a policy-sampled action [10].

From the learned values, we consider two advantage estimates used in offline RFT [10], [28]. The $Q-V$ estimate compares the logged action with its state-value baseline:

$$A_t^{Q-V} = Q_\phi(s_t, \mathbf{a}_t) - V_\psi(s_t). \tag{2}$$

The n-step TD estimate compares an n-step bootstrapped return with the current state value:

$$\begin{aligned} A_t^{\text{n-step TD}} &= R_t^n + \gamma^n \mathsf{m}_{t,n} V_{\psi^-}(s_{t+n}) - V_\psi(s_t), \\ R_t^n &= \sum_{j=0}^{n-1} \gamma^j \mathsf{m}_{t,j} r_{t+j}, \quad \mathsf{m}_{t,j} = \prod_{k=0}^{j-1}(1 - d_{t+k}),\ \mathsf{m}_{t,0} = 1. \end{aligned} \tag{3}$$

Since t indexes action chunks, its rewards, discounting, termination masks, and bootstrap state share the same chunk-level time scale [28], [36].

B. Advantage Calibration

Advantages are commonly normalized within a batch or dataset to stabilize policy optimization [37]. Recent RL post-training methods further normalize advantages within structurally homogeneous groups for finer-grained control [38]. We present group-wise normalization as the general form, with standard normalization recovered by assigning all samples to a single group. Let $\mathcal{G}$ denote a grouping rule and let $g_i = \mathcal{G}(i)$ be the group assigned to example i. The corresponding calibration function is

$$\begin{aligned} z_i &= \frac{u_i - \mu_{g_i}}{\max(\sigma_{g_i}, \sigma_{\min})}, \\ \mu_g = \mathbb{E}_{j:g_j=g}[u_j], &\quad \sigma_g^2 = \mathbb{E}_{j:g_j=g}\left[(u_j - \mu_g)^2\right], \end{aligned} \tag{4}$$

where u_i and z_i denote the raw and calibrated advantages, respectively. A scale floor $\sigma_{\min}$ prevents numerical instability when the within-group variance is small.

C. Advantage Utilization

After calibration, each training example i is assigned a calibrated advantage z_i. Let $\ell_i(\theta)$ denote the supervised policy loss for example i, which is the flow-matching loss used by the $\pi_{0.5}$ actor in our study [4], [39]. Given a nonnegative training weight w_i, the weighted actor objective for a training batch $\mathcal{B}$ is

$$\mathcal{L}_\pi(\theta) = \frac{\sum_{i\in\mathcal{B}} w_i \ell_i(\theta)}{\sum_{i\in\mathcal{B}} w_i}, \qquad \sum_{i\in\mathcal{B}} w_i > 0. \tag{5}$$

The weight w_i is determined by the calibrated advantage z_i and the selected utilization rule.

IV. Study Design

We decompose advantage-guided policy post-training into three sequential stages—advantage construction, calibration, and utilization—to isolate the role of each component under a fixed offline dataset and actor setup. Exhaustively training and evaluating all cross-stage combinations on real robots is prohibitively expensive; we therefore screen construction and calibration with offline diagnostics in Stages I and II, and reserve actor training and deployment for Stage III. This ordering follows the pipeline dependency: calibration operates on constructed advantages, and utilization consumes calibrated advantages. We test whether the offline preferences are reflected in downstream policy performance by revisiting alternative construction and calibration choices in real-robot ablations. Fig. 2 summarizes this staged design and the variants considered in each stage. Accordingly, we study the following research questions: **RQ1 (advantage construction):** Which combinations of critic-learning scheme and advantage construction produce advantages that reliably recover the expected quality orderings among dataset actions? **RQ2 (advantage calibration):** How does advantage calibration affect the comparability of advantages across sub-tasks in the fixed dataset? **RQ3 (advantage utilization):** How do different strategies for utilizing advantages affect policy performance across real-robot tasks?

A. Stage I: Advantage Construction

Stage I compares four representative value-learning schemes and evaluates seven advantage construction configurations. The value-learning schemes are:

- **IQL** [31] updates Q_ϕ using the TD loss (Eq. 1) and learns V_ψ by τ-expectile regression on the in-dataset action values $Q(s_t, \mathbf{a}_t)$. We set $\tau = 0.8$ in this study.
- **SARSA** denotes the $\tau = 0.5$ endpoint of IQL, corresponding to the SARSA-style TD backup discussed in the original IQL paper [31]. The resulting V_ψ regresses toward the mean Q-value over in-dataset actions.
- **ALOE** [10] uses a policy-sampled TD target by replacing $V_{\psi^-}(s_{t+1})$ in Eq. 1 with $Q_{\phi^-}(s_{t+1}, \hat{\mathbf{a}}_{t+1})$, where $\hat{\mathbf{a}}_{t+1} \sim \pi(\cdot \mid s_{t+1})$. The corresponding state-value estimate is obtained from policy-sampled Q-values rather than a separate value network.
- **Monte Carlo (MC)** fits V_ψ directly to empirical Monte Carlo return G_t^{MC}, a target choice also used by RECAP [35], and does not learn an action-value function.

IQL and SARSA learn both Q and V, whereas ALOE learns Q and derives a state-value estimate from policy-sampled Q-values. We evaluate all three with both $Q - V$ scoring (Eq. 2) and n-step TD scoring (Eq. 3). Since MC learns only V, we evaluate it only with n-step TD scoring, yielding seven configurations in total.

The estimates are intended to assign higher advantages to actions that contribute more to task completion. Because the compared constructions estimate different quantities, we evaluate the orderings they induce rather than numerical values. Episode-level outcomes are too coarse for this purpose: failed trajectories may contain actions that complete earlier sub-tasks, while successful trajectories may contain suboptimal but recoverable actions. We therefore use two pairwise ordering diagnostics: R_{int} draws on direct but event-limited evidence from human *intervention*, whereas R_{prog} draws on broader but more indirect evidence from automated *progress* segmentation and visual matching. We consider the two diagnostics jointly and treat their agreement as stronger screening evidence than either diagnostic alone.

The intervention diagnostic R_{int} pairs each human-takeover window with the equal-length policy-controlled window immediately preceding it. Because human operators take over when the policy is judged likely to fail, the subsequent takeover behavior is expected to receive higher advantages. We average the estimated advantages within each window and define R_{int} as the fraction of takeover events in which the takeover window receives the higher advantage.

The progress diagnostic R_{prog} compares *advancing* and *stalled* segments within the same sub-task. For each task, we manually define a sequence of 4–6 sub-tasks and provide a textual description for each. The detailed sub-tasks for each task can be found in the supplementary video due to page limitations. GPT-5.6 then segments each trajectory using these descriptions, visual observations, and robot states. All segments in successful trajectories are labeled *advancing*; in failed trajectories, completed segments are labeled *advancing*, while the attempted but incomplete segment is labeled *stalled*. We average the estimated advantages over six one-second windows from each segment and match each stalled segment to an advancing segment using the distance between their three-camera embeddings from SmolVLM2-500M [40], excluding same-trajectory matches and pairs beyond the task-level 95th-percentile distance. R_{prog} is the fraction of matched pairs in which the advancing segment receives the higher advantage, averaged first within each sub-task and then across sub-tasks. For both diagnostics, 0.5 indicates chance ordering. Fig. 3 shows representative pairs.

B. Stage II: Advantage Calibration

Stage II fixes the advantage construction retained from Stage I and varies only the calibration grouping rule $\mathcal{G}$ in Eq. 4. In long-horizon tasks with sparse terminal rewards, sub-tasks differ in state distribution and remaining horizon,

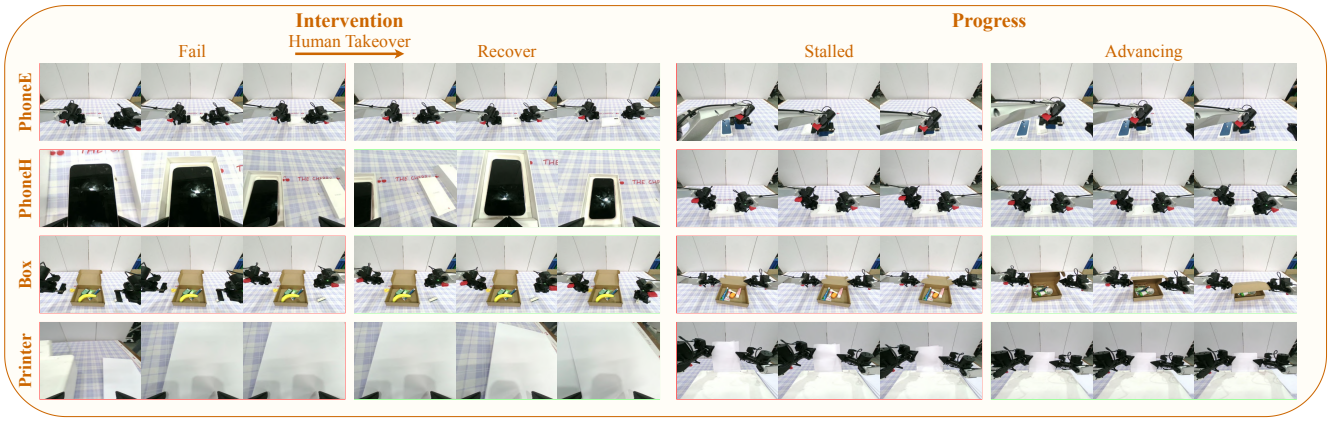

Fig. 3: Representative intervention and progress pairs across four real-world tasks. Each strip contains three frames in temporal order. The intervention examples (left) show failed policy behavior (red) followed by recovery after human takeover (green), whereas the progress examples (right) contrast stalled behavior (red) with advancing behavior (green) within the same sub-task. Both diagnostics assess whether the green examples receive higher estimated advantages than their red counterparts. Images use the base-camera view except for the PhoneH and Printer intervention examples, which use the left- and right-wrist views, respectively.

so raw advantage levels may vary systematically with task progress. Calibration should reduce these context-dependent differences so that calibrated advantages reflect an action's relative quality within its decision context rather than the absolute progress of that context. Actions of comparable relative quality should therefore receive comparable advantages across sub-tasks. We assess this property by measuring systematic differences in calibrated advantage levels across sub-tasks. The sub-task partition established in Stage I is reused only for this offline diagnostic, and sub-task labels are not provided to any calibration operator. Since policy rollouts and expert demonstrations may come from different behavior distributions, normalization statistics are computed separately by source. The comparison includes an uncalibrated baseline together with global and two group-wise normalizations, a distinction also studied in LLM reinforcement learning [38]:

- **Raw** leaves the raw advantages unchanged.
- **Global** normalizes all samples from each source using source-level statistics as Eq. 4.
- **Temporal** further partitions each source into B equal-width bins by episode-relative temporal position and normalizes advantages within each bin, motivated by the step-level relative advantages of TGRPO [9].
- **Value** clips the estimated state values V_{ψ^-} to $[0, 1]$, partitions each source into B equal-width bins over this value, and normalizes advantages within each bin.

For Temporal and Value, we use $B = 8$ bins per source and compute all normalization statistics once from the dataset, holding them constant during Stage III. Bins with fewer than 32 samples use the corresponding source-level Global statistics. For each source, $\sigma_{\min}$ in Eq. 4 is set to 0.1 times its global standard deviation.

We quantify these cross-sub-task differences using η^2, the ratio of variation between sub-task means to total variation in calibrated advantages:

$$\eta^2 = \frac{\sum_k n_k(\bar{z}_k - \bar{z})^2}{\sum_i (z_i - \bar{z})^2}, \tag{6}$$

where z_i is the calibrated advantage of data sample i, with $z_i = u_i$ under Raw. $\bar{z}_k$ is the mean over the n_k samples assigned to sub-task k, and $\bar{z}$ is the mean over all samples. A lower η^2 indicates smaller differences between sub-task means relative to the overall variation and therefore more comparable advantage levels across sub-tasks.

C. Stage III: Advantage Utilization

Stage III fixes the advantage construction and calibration retained from Stages I and II, and varies only whether and how the calibrated advantages are used in the actor update. We compare three post-training conditions, along with their shared SFT initialization as a reference. All three conditions optimize the same $\pi_{0.5}$ flow-matching actor [4] through Eq. 5 using the same dataset and training budget, and differ only in the weight w_i assigned to a data sample.

- **SFT Init** is the shared initialization, trained on the demonstrations alone and evaluated without any further actor update.
- **DAgger** uses an advantage-free binary rule following HG-DAgger [26], [27]: human samples, including both demonstrations and takeover segments, receive unit weight, while autonomous rollouts receive zero weight.
- **Weight** follows the principle of AWR [32] and assigns each sample a continuous exponential weight $w_i = \exp\left(\mathrm{clip}(\beta z_i, -c, c)\right)$, where z_i is the calibrated advantage, with $\beta = 1$ and $c = 2$ in our experiments.
- **Filter** assigns unit weight to the samples whose calibrated advantages are above the q-quantile of the training batch and zero weight to the remaining samples, with $q = 0.5$. This follows the hard-filtering rule of Imit-RECAP baseline in HABC [41].

We do not include RECAP itself as a condition: its advantage conditioning is introduced during pre-training [35],

so a post-training-only re-implementation departs from the original recipe and has been reported to yield more modest gains over supervised fine-tuning [42]. Filter instead serves as a hard approximation that discards low-advantage samples rather than conditioning on them.

Each comparison supports a different attribution. Weight against Filter isolates the weighting form, because both consume the same advantages and differ only in whether the weight is continuous or binary. Weight and Filter against DAgger compare weighting by estimated action quality with weighting by data source, and all three against SFT Init test whether post-training improves over the initialization.

V. Experiment Results

Following the protocol in Sec. IV, we first screen advantage construction and calibration offline, then evaluate policy utilization on four real-world tasks. We also revisit alternative construction and calibration choices in real-world ablations.

A. Experimental Setup

Real-world tasks and platform. We study four bimanual manipulation tasks spanning object insertion, long-horizon sequencing, multi-object interaction, and deformable-object handling (Fig. 1).

- *Pack Phone Easy (PhoneE)* places a phone into its packaging box and closes its lid.
- *Pack Phone Hard (PhoneH)* uses a more complex packaging box and requires an accessory-insertion step, resulting in a longer-horizon manipulation task with greater overall difficulty.
- *Pack Box (Box)* places three objects into a carton, folds the flaps, and presses the side tabs to close the carton.
- *Refill Printer Paper (Printer)* picks up a small stack of paper and loads it into the printer tray.

The bimanual robot platform comprises two 6-DoF arms, each equipped with a gripper, and three RGB fisheye cameras: one mounted on the robot base and one on each wrist. At each policy query, the policy receives the end-effector states and three undistorted camera images, and predicts a chunk of $C = 30$ end-effector actions. The robot executes the chunked actions at 30 Hz, completing the chunk in one second before querying the locally deployed policy again. All compared policies are evaluated using the same robot embodiment, observation and action spaces, deployment interface, and evaluation initial-condition schedule.

Dataset and implementation details. For each task, we construct a fixed offline dataset consisting of approximately 40 hours of expert demonstrations used to train the task-specific SFT initialization, 100 autonomous rollout episodes, and 100 rollout episodes with human takeovers. All critic-learning and advantage-construction experiments are conducted on this fixed dataset. We implement all training procedures based on the LeRobot codebase [43]. Specifically, each learned Q or V function is parameterized by a separate encoder-only Transformer followed by an MLP output head, and reuses the visual embedding of the VLA policy [10]. For n-step TD advantage estimation, we use a per-chunk discount factor of $\gamma = 0.96$ and a backup horizon of $n = 10$ chunks. Because each chunk contains $C = 30$ primitive actions and spans one second, the backup covers $nC = 300$ primitive actions, or 10 seconds of execution.

TABLE I: Stage-II offline calibration results measured by η^2, the proportion of total advantage variance explained by sub-task identity. Lower η^2 indicates less sub-task-dependent variation; Mean is the average across the four tasks.

Calibration	PhoneE	PhoneH	Box	Printer	Mean
Raw	0.072	0.063	0.106	0.127	0.092
Global	0.057	0.051	0.090	0.120	0.080
Temporal	0.071	**0.023**	0.053	0.028	0.044
Value	**0.053**	0.027	**0.013**	**0.010**	**0.026**

B. RQ1: Advantage Construction

We evaluate the seven advantage-construction configurations using the intervention diagnostic R_{int} and progress diagnostic R_{prog}, defined in Sec. IV and illustrated in Fig. 3. All critic checkpoints are evaluated after 10,000 training updates with batch size 2048. The results are shown in Fig. 4. For IQL, SARSA, and ALOE critics, n-step TD yields higher concordance than $Q-V$ on every task and both diagnostics. These three n-step TD variants exceed chance throughout, with R_{int} ranging from 0.639 to 0.773 and R_{prog} from 0.805 to 0.953. In contrast, the $Q-V$ variants and the MC baseline are generally closer to chance.

We hypothesize that the weak $Q - V$ results stem from the limited action diversity, which is common in real-robot datasets collected from a small set of behavior sources. Under narrow action coverage, V is derived from Q evaluated at the same or a similar action, so their subtraction cancels much of the shared value signal and leaves a small residual susceptible to estimation noise. This issue also motivates prior work to deliberately collect diverse failure actions for critic learning [42]. In contrast, n-step TD uses rewards and value changes along logged continuations without relying on action discrimination by Q. MC remains weaker since fitting V to realized returns may leave only small TD residuals along the same logged continuations.

Among the n-step TD variants, IQL achieves the highest four-task mean R_{int} (0.718), while its mean R_{prog} (0.897) nearly matches the best value achieved by SARSA (0.898). We therefore retain IQL with n-step TD, whose upper-expectile regression can emphasize high-value in-dataset actions, a desirable property for heterogeneous datasets [31].

C. RQ2: Advantage Calibration

Stage II evaluates whether calibration reduces systematic differences in advantage levels across sub-tasks, using IQL with the n-step TD construction retained from Stage I. Calibration statistics are computed separately within each data source, following the source-wise grouping rules in Sec. IV. The η^2 diagnostic is then computed accordingly within each task. It measures the proportion of total advantage variance explained by sub-task identity, with lower values indicating greater comparability across sub-tasks.

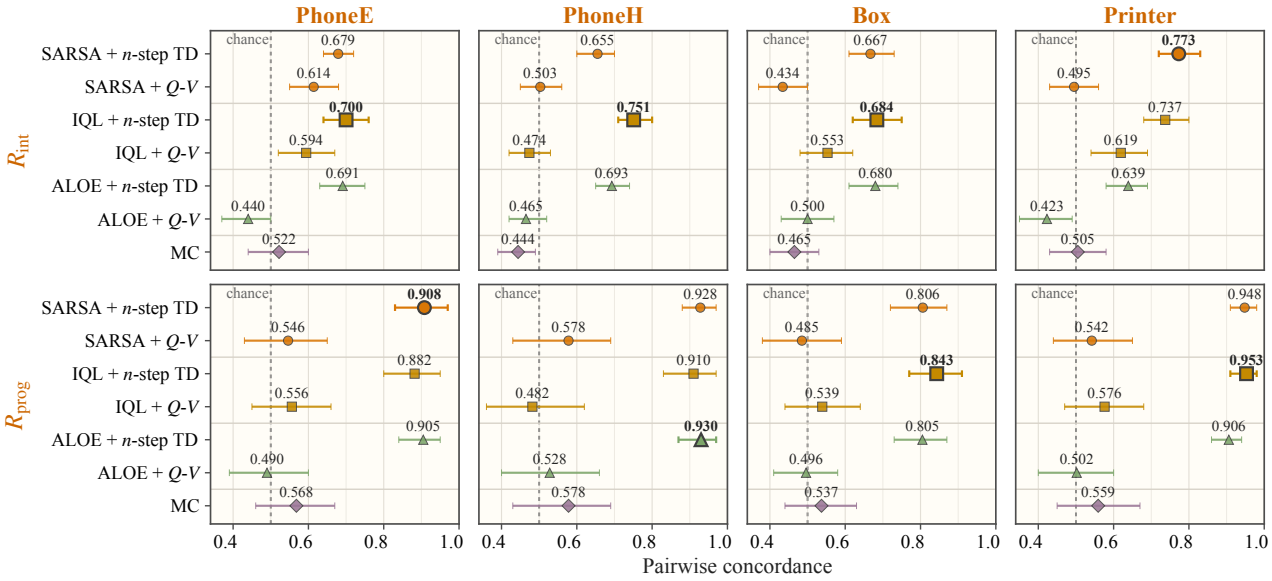

Fig. 4: Advantage-construction diagnostics computed across four real-world tasks. The top row reports intervention ordering R_{int} and the bottom row reports progress ordering R_{prog}. For each task, the intervention and progress diagnostics use approximately 200 and 400 paired comparisons, respectively, after applying their diagnostic-specific filtering rules. Point markers show the ordering diagnostics (higher is better), and horizontal bars show 95% confidence intervals. The dashed line marks chance ordering at 0.5. **Bold** numeric labels with heavier marker outlines identify the within-panel maxima.

TABLE II: Real-world policy evaluation for RQ3. Values in the task columns are averages over 20 real-robot trials per condition; Mean is the average across the four tasks. Higher is better; bold denotes the highest in each column.

Condition	Task progress					Task success				
	PhoneE	PhoneH	Box	Printer	Mean	PhoneE	PhoneH	Box	Printer	Mean
SFT Init	0.50	0.32	0.69	0.24	0.44	0.30	0.00	0.15	0.00	0.11
DAgger	0.65	0.62	0.91	0.66	0.71	0.35	0.30	0.70	0.45	0.45
Weight	**0.88**	**0.81**	**0.95**	**0.79**	**0.86**	**0.80**	**0.70**	**0.80**	**0.65**	**0.74**
Filter	0.78	0.69	0.91	0.60	0.75	0.70	0.40	0.55	0.35	0.50

As shown in Table I, all three calibration methods reduce the mean η^2 from 0.092 for Raw to 0.080, 0.044, and 0.026 for Global, Temporal, and Value, respectively. Compared with Global normalization, the two group-wise normalization variants further reduce the mean η^2 by removing group-specific location and scale differences before the sub-task means are compared. Value achieves a lower mean η^2 than Temporal. One possible explanation is that variations in execution speed, stalls, and recoveries cause temporal bins to mix different progress levels, whereas state value more closely tracks task progress. We therefore carry Value calibration into Stage III and revisit Global and Temporal calibration in the real-world ablations.

D. RQ3: Advantage Utilization

We next evaluate three post-training conditions together with the shared SFT initialization as a reference, using the advantage configuration selected in RQ1 and RQ2. Each post-training condition further trains the SFT checkpoint with 4000 updates and batch size 2048. We report task progress, the fraction of completed sub-tasks at termination, and task success, attained only when all sub-tasks are completed. A trial terminates upon success, an unrecoverable failure, or a task-specific timeout equal to the duration of the longest expert episode. We consider a failure unrecoverable if an object becomes unreachable or the robot begins a later sub-task before completing an earlier one.

Overall performance. Table II reports the real-world task performance. All three post-training conditions yield higher four-task means for both progress and success than SFT Init. Among them, Weight achieves the highest point estimates, gaining 0.42 in progress and 0.63 in success over SFT Init. Weight's finer-grained sample-level supervision may contribute to these gains: it emphasizes higher-scoring actions while retaining nonzero weights for the remaining samples, reducing sensitivity to occasional critic misranking. Qualitatively, Weight exhibits more consistent recovery behaviors in real-robot trials, as shown in the supplementary video.

Ablation study. Keeping Weight fixed, we vary the Stage-I construction and Stage-II calibration. As shown in Fig. 5, replacing the selected n-step TD construction with $Q - V$ generally lowers both metrics, while replacing Value calibration with Global or Temporal also degrades performance in most cases. Across the tested ablations, higher Stage-I ordering concordance and lower Stage-II η^2 generally correspond to higher real-world task progress and success. This overall alignment supports using the diagnostics to screen design choices offline, reducing the need to evaluate

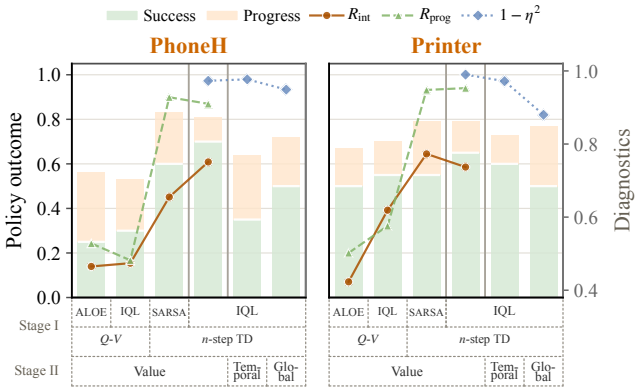

Fig. 5: Ablations of advantage construction and calibration under fixed Weight utilization, with corresponding offline diagnostics overlaid (higher is better). The first four conditions vary Stage-I construction and the last three vary Stage-II calibration; the shared fourth condition is the reference configuration reported in Table II.

every configuration on real robots.

VI. Conclusion

We dissected advantage-guided VLA post-training into construction, calibration, and utilization through a staged empirical study with limited and heterogeneous robot datasets. Stage-specific offline diagnostics selected IQL with chunk-level n-step TD advantages and value-based calibration, while real-robot evaluation favored continuous advantage weighting over source-based DAgger and hard filtering. Across four bimanual tasks, the resulting recipe produced the highest point estimates for both task progress and success on every task, with four-task means of 0.86 and 0.74—gains of 0.42 and 0.63 over SFT Init, respectively. Construction and calibration ablations generally aligned with the offline diagnostics, supporting their use to narrow the post-training design space before costly real-robot evaluation.

References

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar *et al.*, "RT-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[2] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao *et al.*, "OpenVLA: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[3] K. Black, N. Brown, D. Driess, A. Esmail *et al.*, "π_0: A vision-language-action flow model for general robot control," in *RSS XXI*, 2025.

[4] Physical Intelligence, K. Black *et al.*, "$\pi_{0.5}$: A vision-language-action model with open-world generalization," *arXiv preprint arXiv:2504.16054*, 2025.

[5] Z. Zhou, P. Atreya, A. Lee, H. R. Walke, O. Mees, and S. Levine, "Autonomous improvement of instruction following skills via foundation models," in *CoRL*, 2024, pp. 4805–4825.

[6] Y. Jiang, C. Wang, R. Zhang, J. Wu, and L. Fei-Fei, "TRANSIC: Sim-to-real policy transfer by learning from online correction," in *CoRL*, 2024, pp. 1691–1729.

[7] C. Xu, Q. Li, J. Luo, and S. Levine, "RLDG: Robotic generalist policy distillation via reinforcement learning," in *RSS*, 2025.

[8] S. Tan, K. Dou, Y. Zhao, and P. Krähenbühl, "Interactive post-training for vision-language-action models," *arXiv preprint arXiv:2505.17016*, 2025.

[9] Z. Chen, R. Niu, H. Kong, Q. Wang, Q. Xing, and Z. Fan, "TGRPO: Fine-tuning vision-language-action model via trajectory-wise group relative policy optimization," *arXiv preprint arXiv:2506.08440*, 2025.

[10] R. Yang, H. Wang, Z. Wu, C. Liu *et al.*, "ALOE: Action-level off-policy evaluation for vision-language-action model post-training," *arXiv preprint arXiv:2602.12691*, 2026.

[11] J. Shu, Z. Lin, and Y. Wang, "RFTF: Reinforcement fine-tuning for embodied agents with temporal feedback," *arXiv preprint arXiv:2505.19767*, 2025.

[12] Y. Ma, T. Liu, Y. Lan, X. Yin *et al.*, "Diffusion policies with value-conditional optimization for offline reinforcement learning," in *IROS*, 2025, pp. 13 468–13 475.

[13] D. Snyder, A. J. Hancock, A. Badithela, E. Dixon *et al.*, "Is your imitation learning policy better than mine? policy comparison with near-optimal stopping," in *RSS*, 2025.

[14] A. Anwar, R. Gupta, Z. Merchant, S. Ghosh, W. Neiswanger, and J. Thomason, "Efficient evaluation of multi-task robot policies with active experiment selection," in *CoRL*, 2025, pp. 1636–1653.

[15] H. Deng, Z. Wu, H. Liu, W. Guo *et al.*, "A survey on reinforcement learning of vision-language-action models for robotic manipulation," 2025.

[16] J. Liu, F. Gao, B. Wei, X. Chen *et al.*, "What can RL bring to VLA generalization? an empirical study," in *NeurIPS 38*, 2025, pp. 107 724–107 754.

[17] H. Zang, M. Wei, S. Xu, Y. Wu *et al.*, "RLinf-VLA: A unified and efficient framework for reinforcement learning of vision-language-action models," in *RSS*, 2026, listed in the proceedings as "RLux-VLA".

[18] P. Dong, S. Mirchandani, D. Sadigh, and C. Finn, "What matters for batch online reinforcement learning in robotics?" in *ICLR*, 2026.

[19] M. Kim, C. Finn, and P. Liang, "Fine-tuning vision-language-action models: Optimizing speed and success," in *RSS XXI*, 2025.

[20] Y. Chen, S. Tian, S. Liu, Y. Zhou, H. Li, and D. Zhao, "ConRFT: A reinforced fine-tuning method for VLA models via consistency policy," in *RSS XXI*, 2025.

[21] J. Hu, R. Hendrix, A. Farhadi, A. Kembhavi *et al.*, "FLaRe: Achieving masterful and adaptive robot policies with large-scale reinforcement learning fine-tuning," in *2025 ICRA*, 2025, pp. 3617–3624.

[22] Y. Guo, J. Zhang, X. Chen, X. Ji *et al.*, "Improving vision-language-action model with online reinforcement learning," in *2025 ICRA*, 2025, pp. 15 665–15 672.

[23] J. Luo, C. Xu, J. Wu, and S. Levine, "Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning," *Science Robotics*, vol. 10, no. 105, p. eads5033, 2025.

[24] H. Zhao, W. Song, D. Wang, X. Tong *et al.*, "MoRE: Unlocking scalability in reinforcement learning for quadruped vision-language-action models," in *2025 ICRA*, 2025, pp. 11 212–11 218.

[25] K. Huang, R. Scalise, C. Winston, A. Agrawal *et al.*, "Using non-expert data to robustify imitation learning via offline reinforcement learning," in *ICRA*, 2026, arXiv:2510.19495.

[26] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer, "HG-DAgger: Interactive imitation learning with human experts," in *2019 ICRA*. IEEE, 2019, pp. 8077–8083.

[27] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[28] D. Huang, Z. Fang, T. Zhang, Y. Li, L. Zhao, and C. Xia, "CO-RFT: Efficient fine-tuning of vision-language-action models through chunked offline reinforcement learning," *arXiv preprint arXiv:2508.02219*, 2025.

[29] Y. Mao, Z. Yu, W. Mao, Y. Li *et al.*, "ARM: Advantage reward modeling for long-horizon manipulation," *arXiv preprint arXiv:2604.03037*, 2026.

[30] Z. Su, W. Kong, H. Dong, and H. Dong, "IG-RFT: An interaction-guided RL framework for VLA models in long-horizon robotic manipulation," *IEEE T-ASE*, vol. 23, pp. 14 131–14 146, 2026.

[31] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit Q-learning," in *ICLR*, 2022.

[32] X. B. Peng, A. Kumar, G. Zhang, and S. Levine, "Advantage-weighted regression: Simple and scalable off-policy reinforcement learning," *arXiv preprint arXiv:1910.00177*, 2019.

[33] Z. Wang, A. Novikov, K. Zolna, J. S. Merel *et al.*, "Critic regularized regression," in *NeurIPS*, vol. 33, 2020.

[34] A. Nair, A. Gupta, M. Dalal, and S. Levine, "AWAC: Accelerating online reinforcement learning with offline datasets," *arXiv preprint arXiv:2006.09359*, 2020.

[35] P. Intelligence, A. Amin, R. Aniceto, A. Balakrishna *et al.*, "$\pi^*_{0.6}$: a VLA that learns from experience," *arXiv preprint arXiv:2511.14759*, 2025.
[36] Q. Li, Z. Zhou, and S. Levine, "Reinforcement learning with action chunking," in *NeurIPS 38*, 2025, pp. 61 781–61 816.
[37] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[38] M. Zhu, X. Chen, B. Yu, H. Zhao, and J. Jia, "Stratified GRPO: Handling structural heterogeneity in reinforcement learning of LLM search agents," *arXiv preprint arXiv:2510.06214*, 2025.
[39] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *ICLR*, 2023.
[40] A. Marafioti, O. Zohar, M. Farré, M. Noyan *et al.*, "SmolVLM: Redefining small and efficient multimodal models," *arXiv preprint arXiv:2504.05299*, 2025.
[41] T. Fang, S. Huang, N. Fang, G. Zhao *et al.*, "Hierarchical advantage weighting for online RL fine-tuning of VLAs from sparse episode outcomes," *arXiv preprint arXiv:2606.17043*, 2026.
[42] Y. Wang, X. Li, P. Xie, P. Yang *et al.*, "Learning while deploying: Fleet-scale reinforcement learning for generalist robot policies," *arXiv preprint arXiv:2605.00416*, 2026.
[43] R. Cadene, S. Alibert, A. Soare, Q. Gallouedec *et al.*, "LeRobot: State-of-the-art machine learning for real-world robotics in pytorch," https://github.com/huggingface/lerobot, 2024.