ATOMIC-TO-COMPOSITE DISTILLATION VIA TASK-WISE PROGRESS POTENTIALS AND HIDDEN-STATE ALIGNMENT IN VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) models achieve strong performance on individual manipulation skills, yet their success drops substantially when these skills must be executed sequentially as composite tasks. We identify two key challenges underlying this atomic-to-composite gap. First, composite behaviors are difficult to learn: collecting long sequence demonstrations is costly, while reinforcement learning (RL) with sparse binary rewards provides little learning signal for trajectories that achieve only partial progress. Second, composite instructions do not explicitly indicate which atomic task should be active at each stage, creating ambiguity at subtask transitions. Moreover, the same skill succeeds far less often under a composite instruction than under its own atomic instruction, suggesting that composite conditioning fails to reliably invoke competence the policy already possesses. We propose **ATLAS**, an end-to-end RL framework that leverages the policy's own atomic-task competence to address both challenges. To address sparse credit assignment, *task-wise progress potential shaping* learns reusable progress potentials from successful and failed atomic rollouts and composes them across stages to provide dense, phase-aware rewards while preserving the optimal policy. To bridge the representation gap, *phase-conditioned atomic-to-composite hidden-state alignment* aligns composite-conditioned representations with their corresponding atomic-conditioned representations at each stage, allowing the composite-conditioned policy to directly exploit skills already encoded by the atomic policy without inference-time planners or prompt switching. These complementary mechanisms bridge atomic and composite learning within a unified training framework. Across simulation and real-robot experiments, ATLAS consistently outperforms zero-shot VLA baselines, standard PPO, and existing progress-reward approaches, achieving gains of up to 33pp on RoboLab and 40pp in real-world settings.

1 INTRODUCTION

Generalist robotic policies are increasingly regarded as a foundation for diverse manipulation tasks, enabling multi-task learning across skills such as pick-and-place, drawer opening, and tool use. In real-world settings, however, these skills rarely occur in isolation and must often be composed to accomplish a broader goal. For example, preparing a drink may require retrieving a cup, grasping a bottle, pouring the drink, and delivering the filled cup. Successful execution therefore requires not only mastering each individual skill, but also reliably chaining them into a coherent sequence.

Although recent vision-language-action (VLA) models (Black et al., 2024; Kim et al., 2024; Bjorck et al., 2025; Agarwal et al., 2026) achieve remarkable performance on individual skills, their success rates decrease substantially when these skills must be executed sequentially as part of composite tasks (Yang et al., 2025; Wu et al., 2026; Zhang et al., 2026b; Xu et al., 2026). An atomic task here refers to an individual task, whereas a composite task consists of two or more atomic tasks chained together. We refer to each atomic task comprising a composite task as a subtask, and define the execution stages for these subtasks as Phase A and Phase B.

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.

As shown in Fig. 1, the two atomic skills achieve success rates of 62.5% and 83.3% in isolation. However, naively chaining them yields only an 29.2% composite-task success rate, far below the 52.1% expected under independent execution. This performance gap suggests that learning sequential behaviors requires additional guidance beyond the atomic skills themselves.

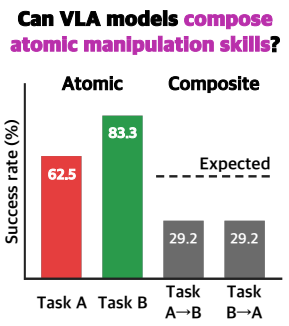

Figure 1: **Decreased composite skill success under naive chaining.** Chaining individually successful atomic skills yields only 29.2% success, far below expectation 52.1%.

Moving from atomic to composite tasks makes policy learning more challenging for several reasons. First, collecting long demonstrations of composite tasks for supervised fine-tuning is costly, making reinforcement learning (RL) an attractive alternative that can also generalize beyond demonstrated behaviors (Ross et al., 2011; Levine et al., 2020). However, as task composition lengthens the decision horizon, the single binary reward typically provided by standard RL gives no credit to failed trajectories, even when they achieve partial progress. To address this, existing methods estimate task advancement using temporal ordering (Yang et al., 2024), goal-conditioned representations (Ma et al., 2023b;a), or frozen VLMs (Ma et al., 2025), while stage-aware approaches use hand-designed potentials or annotated action primitives (Xu et al., 2025; Chen et al., 2026). Although effective, these progress signals either rely on proxies for task completion or require task-specific structure and supervision, making reliable and scalable progress estimation for composite tasks difficult.

Second, composite instructions lack explicit indicators of which subtask is currently active or completed. This creates ambiguity in phase transitions, particularly at subtask boundaries where the model must determine when to switch from one atomic task to the next. Prior methods address this ambiguity using high-level planners that select the next skill or condition the low-level policy on an explicit subtask instruction (Ahn et al., 2022; Shi et al., 2025; Cai et al., 2026; Yang et al., 2026b; Liu et al., 2026a), while SeqVLA (Yang et al., 2025) switches prompts based on a learned completion detector. Despite improving sequential execution, these methods rely on inference-time management, making execution dependent on planner decisions or completion-triggered prompt switching rather than improving the composite-conditioned policy itself.

We propose **ATLAS**, an end-to-end RL framework that jointly optimizes atomic and composite-task learning, enabling improvements in atomic skills to directly support composite execution. ATLAS jointly rolls out composite tasks and their constituent atomic tasks during training, coupling their learning through two complementary mechanisms. First, we learn task-wise progress potentials from successful and failed atomic rollouts to estimate eventual skill completion rather than temporal advancement or goal similarity alone. We then compose these potentials across stages for dense reward shaping that can be reused across composite tasks sharing the same atomic skills. Second, we align composite-conditioned representations with the corresponding atomic-conditioned representations at each stage, directly leveraging the policy's own atomic competence instead of relying on external planners or prompt switching. This phase-conditioned alignment improves the composite-conditioned policy itself without introducing additional inference-time orchestration. Our method achieves performance gains of up to 33pp across several RoboLab (Yang et al., 2026a) tasks, including challenging composite tasks that require sequentially grasping visually similar objects, such as colored blocks and Rubik's cubes, in a specified order. Furthermore, our real-robot experiments demonstrate that the proposed method transfers effectively beyond simulation to physical robotic systems. Our contributions can be summarized as follows:

- We introduce **ATLAS**, an end-to-end RL framework that jointly trains atomic and composite tasks by collecting rollouts from individual atomic skills and their ordered compositions, enabling atomic-task learning to directly support composite execution.
- We propose **task-wise progress potential shaping**, which learns reusable progress potentials from both successful and failed atomic rollouts and composes them across stages to provide dense, phase-aware credit assignment while preserving the optimal policy.
- We introduce **phase-conditioned atomic-to-composite hidden-state alignment**, which leverages the policy's atomic-conditioned representations to improve composite-conditioned execution without inference-time planners or prompt switching. Across Robo-

Lab tasks and real-robot experiments, ATLAS consistently improves composite-task performance and transfers effectively beyond simulation.

2 Related Works

2.1 Vision-Language-Action models

Vision-language-action (VLA) models map visual observations and language instructions to robot actions. Prior work either predicts discretized action tokens autoregressively (Brohan et al., 2023; Kim et al., 2024; Qu et al., 2025; Pertsch et al., 2025), or generates continuous action chunks with diffusion (Team et al., 2024; Li et al., 2024; Liu et al., 2025) or flow matching (Black et al., 2024; Physical Intelligence et al., 2025; Bjorck et al., 2025). Continuous generative policies enable smooth control but have intractable action likelihoods that complicate policy-gradient fine-tuning (Chen et al., 2025a; Lyu et al., 2025). We adopt the flow-matching VLA $\pi_{0.5}$ (Physical Intelligence et al., 2025) and investigate reinforcement learning post-training for robotic manipulation.

2.2 Compositional skill execution in VLAs

Recent benchmarks reveal a gap between atomic skill execution and composition, as policies often struggle with composite tasks (Chen et al., 2025b; Wu et al., 2026). Prior work attributes this failure to skill transitions, where the end state of the preceding skill matters for the next skill (Pang et al., 2026; Liu et al., 2026b). Existing approaches address composition through skill abstractions or experts (Zhang et al., 2026a), hierarchical or modular architectures (Shi et al., 2025; Yang et al., 2026c), and transition-aware conditioning, rewards, or replanning (Fan et al., 2025; Liu et al., 2026b; Pang et al., 2026). However, they largely overlook within-skill progress feedback and reuse of atomic competence under composite instructions. We address both by learning composable progress potentials and aligning composite-conditioned representations with their atomic counterparts.

2.3 Progress-aware credit assignment

Sparse terminal rewards in reinforcement learning create a fundamental credit assignment problem, as they do not reveal which intermediate decisions led to success or failure. In language-model reasoning, process supervision addresses this by providing feedback to individual reasoning steps (Uesato et al., 2022; Lightman et al., 2024), including rewards based on changes in predicted success probability (Setlur et al., 2025) or automatically labeled trajectories (Choudhury, 2025; Xi et al., 2026). Robot manipulation suffers from a similar lack of progress feedback under sparse task rewards. Accordingly, prior work derives continuous progress signals from temporal ordering, goal-conditioned visual embeddings, or pretrained vision-language models (Yang et al., 2024; Ma et al., 2023b;a; 2025). Prior works also assigns intermediate supervision according to predefined or inferred stages, trajectory segments, or action primitives (Xu et al., 2025; Lu et al., 2025; Chen et al., 2026). Our approach instead learns skill-specific progress potentials and composes them, providing dense progress feedback for each skill that transfers across composite tasks.

2.4 Intermediate representation distillation

Knowledge distillation transfers intermediate representations such as feature maps (Romero et al., 2015; Yang et al., 2022), attention patterns (Zagoruyko & Komodakis, 2017; Wang et al., 2020), and hidden states (Sun et al., 2019; Jiao et al., 2020) from teacher to student models. Self-distillation further derives supervisory targets from the model itself under different views or conditioning, removing the need for a separate teacher model (Caron et al., 2021). In robotics, privileged learning distills state-based teachers into policies that act from deployable observations (Chen et al., 2020; Song et al., 2023), while VLA distillation transfers action capabilities (Dong et al., 2025), preserves accuracy in efficient models (Zhang et al., 2026c), and incorporates spatial knowledge (Li et al., 2026). Our approach extends self-distillation by using the policy's own atomic-conditioned representations as phase-specific targets under composite instructions, leveraging its atomic competence.

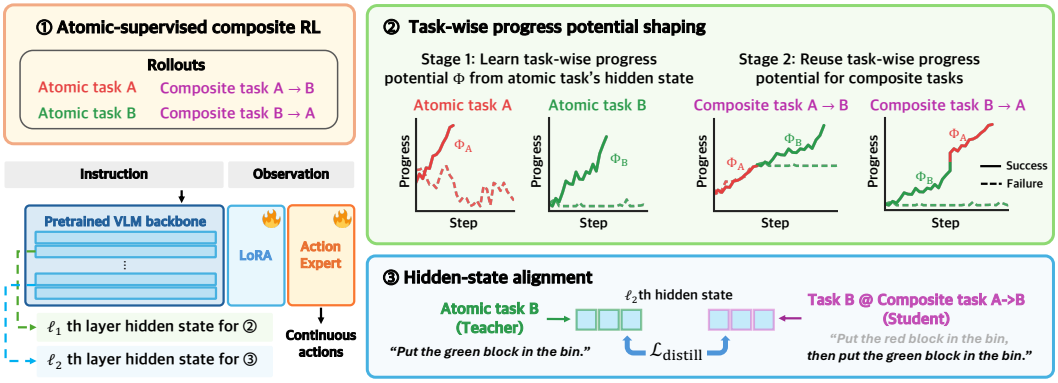

Figure 2: **Overall framework of ATLAS**. ATLAS jointly trains composite and atomic tasks using two mechanisms: task-wise progress potential shaping for dense, phase-aware rewards, and phase-conditioned atomic-to-composite hidden-state alignment for transferring atomic-skill knowledge to composite execution, which are integrated into PPO training.

3 METHOD

3.1 PROBLEM STATEMENT: COMPOSITIONALITY GAP IN SEQUENTIAL EXECUTION

Table 1: **Composition gap in sequential task execution.** Success rates (SR, %) of $\pi_{0.5}$ on atomic and composite tasks over 24 episodes per task. For each ordering, we report task 2 success conditioned on task 1 completion (*e.g.*, $A \mid B_{succ.}$) and overall task success. Expected composite success (A×B) is the product of the two atomic success rates under an independence assumption.

	SR @ Atomic Tasks		**Conditional task 2 SR** @ Composite Tasks		**Full SR** @ Composite Tasks		**Expected SR** @ Composite Tasks
	A	B	$A \mid B_{succ.}$	$B \mid A_{succ.}$	A→B	B→A	A×B
Suite 1	62.5	83.3	35.0	43.8	29.2	29.2	52.1
Suite 2	50.0	25.0	25.0	9.1	8.3	4.2	12.5
Suite 3	58.3	62.5	50.0	18.8	12.5	29.2	36.5
Suite 4	75.0	41.7	60.0	15.0	12.5	12.5	31.2

To test whether competence on individual skills transfers to sequential execution, we compare the performance of $\pi_{0.5}$ (Physical Intelligence et al., 2025) on composite tasks with estimates based on atomic results. Tab. 1 shows that in every task and for both orderings, the success rate on composite tasks falls below the product of the success rates on the two constituent tasks. The shortfall arises in the second stage: conditioned on the completion of task 1, the success rate of task 2 is lower than the success rate of the same skill executed as an atomic task in all cases. Together, these results demonstrate that strong performance on atomic tasks does not translate into reliable sequential execution. The evaluation tasks are detailed in Appendix A.1.

3.2 ATLAS: ATOMIC-SUPERVISED COMPOSITE REINFORCEMENT LEARNING

RL on composite tasks faces two key challenges: sparse rewards hinder credit assignment, and ambiguity about the active skill under a composite instruction makes it hard for the policy to leverage its atomic competence. We address both by jointly optimizing the policy with rollouts from each composite task and its constituent atomic tasks. Atomic rollouts provide task-wise progress potentials that yield dense, phase-aware rewards for composite rollouts (Sec. 3.3), while also serving as phase-specific supervision to align composite-conditioned representations with their atomic counterparts (Sec. 3.4). In this way, atomic-task learning directly supports composite execution.

Composite rollouts are dominated by first-phase chunks, since task B in $A \to B$ begins only after task A is completed. Uniformly pooling all chunks would therefore overemphasize the first phase.

We therefore group chunks by task and phase and weight each chunk by its inverse group frequency, renormalized to unit mean, so that every phase contributes equally regardless of its sample count.

3.3 TASK-WISE PROGRESS POTENTIAL SHAPING

We introduce task-wise progress potential shaping to mitigate reward sparsity in composite tasks while avoiding a key failure of existing progress rewards, which can mistake mere temporal advancement for genuine progress. Specifically, we learn a separate potential function $\Phi_k(o) \in [0, 1]$ for each atomic task k from rollouts of the base policy, including failed ones. Here t indexes action chunks, the unit at which the policy acts and receives credit. Each observation o_t is assigned a discounted progress target

$$y_t = \begin{cases} \gamma_\Phi^{T-t}, & \text{if the current skill is completed at chunk } T, \\ 0, & \text{if it is never completed,} \end{cases} \tag{1}$$

so that $\mathbb{E}[y_t \mid o_t]$ is the value of the current skill under a sparse completion reward discounted at γ_Φ. Since failed rollouts receive a zero target regardless of elapsed time, Φ_k cannot equate temporal advancement with progress. We fit Φ_k using a linear regressor on visual-token features pooled from three intermediate VLA layers. The linear head reduces overfitting to the limited number of successful episodes, while intermediate-layer features preserve visual cues of physical progress that may be attenuated in later, more action-oriented layers.

For a task q with N_q phases, we define the task potential as

$$\Phi_q(s_t) = \frac{n_t + \Phi_{k_t}(o_t)}{N_q}, \qquad \Phi_q \equiv 1 \text{ once all } N_q \text{ phases are complete,} \tag{2}$$

where k_t is the active phase and n_t is the number of completed phases, read from the simulator's subtask counter, which credits a phase only in the instructed order. This formulation provides a continuous estimate of progress within a phase and a discrete increase at each phase completion. For atomic tasks ($N_q = 1$), Φ_q reduces to Φ_k. We then add the potential difference to the environment reward r_t in the potential-based form of Ng et al. (1999):

$$r'_t = r_t + \lambda_{\text{prog}} \left[\gamma \Phi_q(s_{t+1}) - \Phi_q(s_t)\right], \tag{3}$$

where γ is the RL discount factor and λ_{prog} controls the shaping magnitude. The regression head is fit offline and kept fixed, and is used only during training.

3.4 PHASE-CONDITIONED ATOMIC-TO-COMPOSITE HIDDEN-STATE ALIGNMENT

VLAs that execute an atomic skill under its atomic instruction can degrade when the same skill appears as one phase of a composite instruction. We use the same VLA, conditioned on the phase-specific atomic instruction, as a teacher to supervise composite execution. We hypothesize that the composite instruction distorts representations the policy already forms under the atomic one.

For a composite trajectory with active subtask p, let l_{comp} denote the composite instruction and $l_{\text{atom}}^{(p)}$ the atomic instruction corresponding to p. Given the same observation o, we treat the hidden state induced by the atomic instruction as a teacher and the hidden state induced by the composite instruction as a student. Specifically, at a selected intermediate layer L, we minimize

$$\mathcal{L}_{\text{distill}} = \frac{\left\| h_L(o, l_{\text{comp}}) - \text{sg}\left[h_L(o, l_{\text{atom}}^{(p)})\right] \right\|^2}{\text{RMS}\left(\text{sg}\left[h_L(o, l_{\text{atom}}^{(p)})\right]\right)^2}, \tag{4}$$

where h_L denotes the hidden representation at layer L and $\text{sg}[\cdot]$ is the stop-gradient operator. Because the teacher and student differ only in their instructions, their visual-token representations are nearly identical. Averaging over all prefix tokens would therefore dilute the instruction-dependent signal, so we compute the loss over language tokens only.

The normalized distillation objective is combined with PPO as

$$\mathcal{L} = \mathcal{L}_{\text{PPO}} + \lambda_{\text{distill}} \mathcal{L}_{\text{distill}}, \tag{5}$$

where λ_{distill} controls the strength of representation alignment. Atomic-task samples incur zero distillation loss because their teacher and student instructions are identical. This objective is used only during training and preserves the original single-policy inference procedure.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Task suite design. We construct four task suites in RoboLab to evaluate a policy's ability to compose atomic skills into sequential behaviors. RoboLab is a language-conditioned robot-manipulation benchmark built on NVIDIA Isaac Lab (Mittal et al., 2025). Each suite contains two atomic pick-and-place tasks, each targeting a single object, and two composite tasks that execute the same skills in opposite orders. All models are trained and evaluated on the same suites in simulation. Details of the task suites are given in Appendix A.1.

Simulation and real-robot setup. We conduct all simulation experiments in RoboLab built on Isaac Lab 2.3 (Mittal et al., 2025) and Isaac Sim 5.1, using a 7-DoF Franka Panda arm with a Robotiq 2F-85 gripper. The policy receives wrist and over-the-shoulder RGB observations and controls joint positions at 15 Hz. Object positions and orientations are randomized at every episode, with separate seeds for training and evaluation. Further details are provided in Appendix A.2. To evaluate sim-to-real transfer, we deploy the simulation-trained policy on a Franka Research 3 arm with a Robotiq gripper, using wrist and external RGB observations. Without any real-robot RL training, we evaluate two composite tasks from the simulation suites for 10 episodes each under randomized object placements. Further details are provided in Appendix A.3.

Implementation details. We initialize our method from the publicly released $\pi_{0.5}$ checkpoint (Physical Intelligence et al., 2025), which is fine-tuned on the DROID dataset (Khazatsky et al., 2024) with joint-position actions and therefore matches the action interface of RoboLab (Yang et al., 2026a). $\pi_{0.5}$ combines a 3B-parameter PaliGemma vision-language backbone (Beyer et al., 2024) with a 300M-parameter flow-matching action expert. This checkpoint serves both as our zero-shot baseline and as the common initialization for all RL-based methods. Following π_{RL} (Chen et al., 2025a), we inject noise into the flow-matching process to obtain tractable chunk-level log-likelihoods for policy optimization. Additional training details are listed in Appendix A.4.

Evaluation metrics. We report success rates (SR) under two criteria. *Order-preserving SR* counts an episode as successful if the required objects are manipulated in the specified order, regardless of additional intermediate manipulations. *Strict-selection SR* is stricter: it requires manipulating only the specified objects, in the correct order, without placing any irrelevant object in the bin. The gap between the two metrics thus measures task-irrelevant object manipulation that nonetheless satisfies the task-level completion predicate. We use 24 episodes per simulation task and 10 per real-world task. Detailed evaluation protocols are reported in Appendix A.5.

Baselines. We compare ATLAS with two RL baselines initialized from the same $\pi_{0.5}$ (Physical Intelligence et al., 2025): standard PPO (Schulman et al., 2017) with a binary success reward and PPO with a GVL-style dense progress reward (Ma et al., 2025) using Qwen3-VL-32B (Bai et al., 2025) as the estimator. All RL methods use the same initialization, optimization hyperparameters, and rollout budget. For a fair comparison, we train the RL baselines on both composite orderings, $A \rightarrow B$ and $B \rightarrow A$. This allows each atomic skill to appear as the second subtask and receive a reward signal from successful composite rollouts. We also report the pre-trained $\pi_{0.5}$ before fine-tuning. As external references, we evaluate three recent generalist VLA models zero-shot, namely GR00T N1.7 (Bjorck et al., 2025), MolmoAct 2 (Fang et al., 2026), and Cosmos 3 Edge Policy (Agarwal et al., 2026). Further details are provided in Appendix A.6.

4.2 EXPERIMENTAL RESULTS

Simulation results. Tab. 2 summarizes the success rates across the four simulation suites. Among $\pi_{0.5}$ and its fine-tuned variants, ATLAS attains the highest success rate in 20 of the 24 columns. The improvement is most pronounced in Suite 1, where ATLAS performs best in every column, doubles $\pi_{0.5}$ on a composite task under both metrics, and reaches 100% on an atomic task. Across all composite tasks, ATLAS improves over $\pi_{0.5}$ under both metrics, with gains of up to 33.3 percentage points. In contrast, standard PPO with the sparse success reward yields less consistent improvements and even degrades composite-task performance in Suites 1 and 3 by as much as 20.8

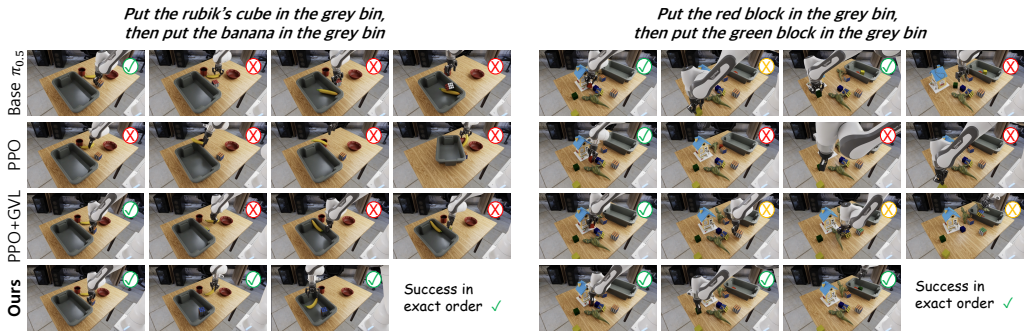

Figure 3: **Qualitative simulation results on RoboLab composite tasks.** The left and right examples correspond to Suite 1 and 2, respectively. Baseline policies often fail by grasping target objects in the wrong order or by interacting with non-target objects, whereas ATLAS reliably follows the sequence and completes the composite task. Red crosses indicate wrong-order grasps of target objects, while yellow crosses indicate grasps of wrong objects that violate the strict sequence criterion.

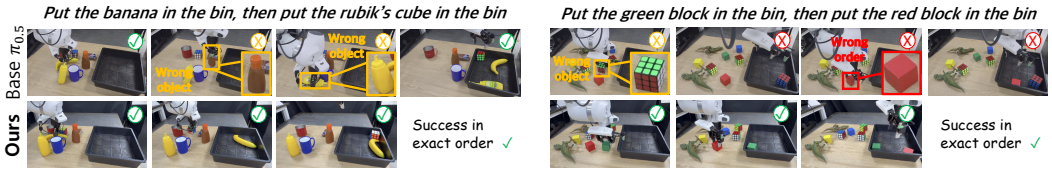

Figure 4: **Qualitative real-world results on composite tasks.** The left and right examples correspond to Suite 1 and 2, respectively. Using the same visual notation as in Fig. 3, the base $\pi_{0.5}$ policy exhibits wrong-order target-object selections and wrong-object selections, whereas ATLAS follows the instructed sequence and achieves strict-selection success in both examples.

percentage points. Using the dense GVL progress reward does not alleviate this issue, reducing the order-preserving success rate of $\pi_{0.5}$ on seven of the eight tasks in Suites 1 and 4. ATLAS further improves seven of the eight atomic tasks, including a 20.8-point gain in Suite 3, highlighting the benefit of atomic co-training. Among the external models, GR00T N1.7 and MolmoAct 2 rarely complete a composite task, while Cosmos 3 Edge Policy is the strongest reference yet ATLAS exceeds it in 12 of the 16 composite columns. These results demonstrate the effectiveness of ATLAS in improving composite task execution over both sparse and dense reward baselines. Fig. 5 shows that ATLAS better translates atomic-skill competence into composite-task performance, substantially narrowing the gap between the expected success rate under independent atomic execution and the observed composite success rate.

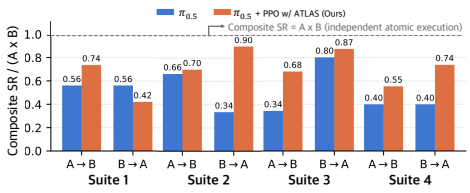

Figure 5: **ATLAS (Ours) better realizes the composite-task performance implied by atomic-skill competence.** The dashed line shows the expected composite success rate under independent atomic execution. ATLAS reduces the gap between this expectation and the observed composite success rate compared with $\pi_{0.5}$.

Fig. 3 compares all methods within the same scenes. For the Suite 1 task requiring the Rubik's cube followed by the banana, the base $\pi_{0.5}$ policy and PPO with the GVL reward fail to execute the required pick-and-place sequence accurately, leading to an order violation. Standard PPO also fails, but through an unsuccessful target object pick-and-place rather than an ordering error. For the Suite 2 task requiring the red block followed by the green block, the base $\pi_{0.5}$ policy and PPO with the GVL reward select wrong objects and consequently reach time-limit termination before completing the instructed sequence. Standard PPO instead fails to complete the required target object pick-and-place. In contrast, ATLAS selects only the target objects and completes both subgoals in the specified order, yielding strict-selection success in both examples. Additional qualitative simulation results for Suite 3 and Suite 4 are provided in Appendix B.1.

Table 2: **Success rates (%) on RoboLab simulation suites.** Each suite consists of two atomic tasks (A, B) and their two composite orderings ($A \to B$, $B \to A$), with 24 evaluation episodes per task. *Order* counts a trial as successful if the target objects are placed in the instructed order; *Strict* additionally counts any placement of a non-target object as a failure. Atomic tasks report *Order* only. Shaded rows are $\pi_{0.5}$ and its fine-tuned variants, whose best result in each column is shown in **bold** and the second best is underlined. Among them, fine-tuning with ATLAS yields the highest success rate in most columns.

Method	Suite 1						Suite 2					
	A	B	A→B		B→A		A	B	A→B		B→A	
			Order	Strict	Order	Strict			Order	Strict	Order	Strict
GR00T N1.7	33.3	29.2	8.3	8.3	8.3	8.3	29.2	12.5	0.0	0.0	0.0	0.0
MolmoAct 2	20.8	12.5	0.0	0.0	0.0	0.0	37.5	0.0	0.0	0.0	0.0	0.0
Cosmos3 Edge Policy	62.5	66.7	37.5	33.3	33.3	16.7	62.5	54.2	12.5	4.2	16.7	12.5
$\pi_{0.5}$	62.5	83.3	29.2	29.2	29.2	29.2	50.0	25.0	8.3	4.2	4.2	4.2
+ PPO	50.0	83.3	29.2	25.0	8.3	8.3	50.0	29.2	12.5	8.3	20.8	20.8
+ PPO w/ GVL reward	54.2	62.5	20.8	12.5	16.7	16.7	45.8	25.0	16.7	**16.7**	4.2	0.0
+ PPO w/ **ATLAS (Ours)**	**79.2**	**100.0**	**58.3**	**58.3**	**33.3**	**33.3**	**83.3**	**50.0**	**29.2**	8.3	**37.5**	**33.3**

Method	Suite 3						Suite 4					
	A	B	A→B		B→A		A	B	A→B		B→A	
			Order	Strict	Order	Strict			Order	Strict	Order	Strict
GR00T N1.7	16.7	50.0	0.0	0.0	0.0	0.0	20.8	12.5	0.0	0.0	0.0	0.0
MolmoAct 2	4.2	8.3	0.0	0.0	0.0	0.0	41.7	0.0	0.0	0.0	0.0	0.0
Cosmos3 Edge Policy	91.7	70.8	37.5	29.2	29.2	29.2	75.0	33.3	8.3	0.0	12.5	4.2
$\pi_{0.5}$	58.3	**62.5**	12.5	8.3	29.2	16.7	75.0	41.7	12.5	0.0	12.5	0.0
+ PPO	54.2	45.8	4.2	0.0	33.3	**33.3**	**87.5**	41.7	16.7	**12.5**	16.7	16.7
+ PPO w/ GVL reward	62.5	50.0	8.3	0.0	29.2	20.8	54.2	37.5	12.5	**12.5**	8.3	8.3
+ PPO w/ **ATLAS (Ours)**	**79.2**	54.2	**29.2**	**25.0**	**37.5**	25.0	83.3	**54.2**	**25.0**	**12.5**	**33.3**	**25.0**

Table 3: **Success rates (%) on real-world compositional tasks.** *Order* and *Strict* are defined as in Tab. 2. Each entry uses 10 trials. Fine-tuning with ATLAS in simulation improves both metrics on both physical tasks. Best results in each column are in **bold**.

Method	Suite 1		Suite 2		Mean	
	Order	Strict	Order	Strict	Order	Strict
$\pi_{0.5}$	80	50	70	20	75	35
+ PPO w/ **ATLAS (Ours)**	**100**	**90**	**80**	**40**	**90**	**65**

Table 4: **Success rates (%) of ATLAS ablations on Suite 1.** Removing any single component lowers all reported success rates on every task. Best results in each column are in **bold**.

Method	A	B	A→B		B→A	
			Order	Strict	Order	Strict
ATLAS (full)	**79.2**	**100.0**	**58.3**	**58.3**	**33.3**	**33.3**
w/o hidden-state alignment	58.3	75.0	41.7	41.7	0.0	0.0
w/o potential shaping	54.2	87.5	37.5	29.2	16.7	8.3
w/o atomic episodes	62.5	45.8	29.2	25.0	16.7	16.7

Real-world results. Tab. 3 reports the order-preserving and strict-selection success rates of $\pi_{0.5}$ on the two real-world tasks before and after ATLAS training in simulation. Fine-tuning with ATLAS raises the strict-selection success rate of $\pi_{0.5}$ from 50% to 90% on the Suite 1 task and from 20% to 40% on the Suite 2 task. The improvement is particularly pronounced on the Suite 1 task, where the fine-tuned policy completes every trial in the instructed order and succeeds in 9 of 10 trials under the strict-selection metric. Zero-shot $\pi_{0.5}$ shows a large gap between the two metrics on both tasks, 30 percentage points on Suite 1 and 50 on Suite 2, indicating that many of its order-preserving successes also place a non-target object in the bin. Fine-tuning with ATLAS narrows this gap while raising task completion, suggesting that it aligns the policy more closely with the instructed sequential behavior. Fig. 4 presents representative real-world rollouts in the same scenes. For the Suite 1 task requiring the banana followed by the Rubik's cube, the base $\pi_{0.5}$ policy eventually completes the two target placements in the instructed order, but additionally selects a wrong object. It therefore attains order-preserving success without satisfying strict-selection success. For the Suite 2 task requiring the green block followed by the red block, the base policy selects a target object in the wrong order and fails the composite task. In contrast, ATLAS selects only the instructed objects and completes both subgoals in the specified order, achieving strict-selection success in both real-world examples.

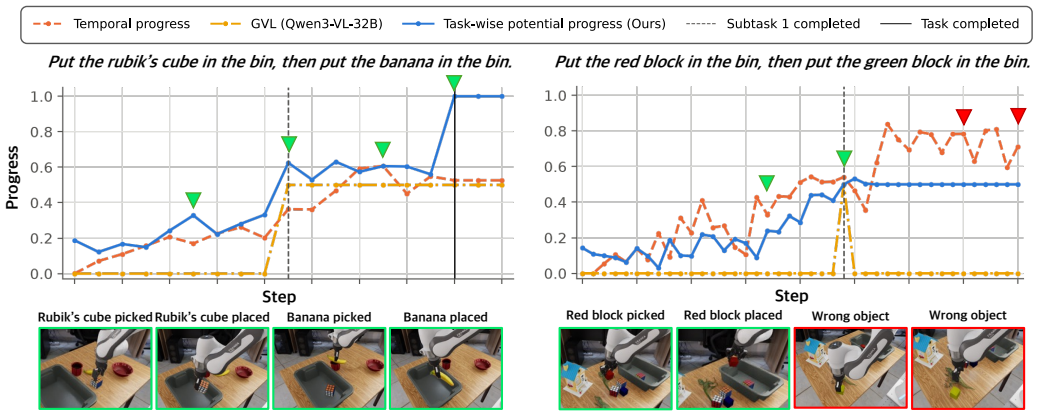

Figure 6: **Comparison of progress rewards.** Our task-wise potential provides dense supervision by predicting discounted eventual skill completion, thereby reflecting actual advancement toward success. Temporal progress can instead increase with elapsed time without further task completion, while VLM-based rewards remain largely sparse around completion events.

4.3 Analysis and Ablation Studies

Progress reward analysis. Fig. 6 compares our task-wise potential with two progress rewards: a temporal-progress model in the style of Rank2Reward (Yang et al., 2024), which regresses normalized elapsed time on successful rollouts using the same policy features as ours, and GVL (Ma et al., 2025), which queries a pretrained VLM (Qwen3-VL-32B) for per-frame completion. Unlike the temporal-progress model, our potential predicts discounted eventual completion of the active atomic skill from the current observation, providing dense within-phase supervision while preserving completed progress across phases. The temporal-progress model keeps increasing with elapsed execution time even after meaningful task progress stops, whereas GVL changes mainly around salient completion events and thus provides a much sparser signal. Overall, our potential better reflects advancement toward successful completion rather than temporal progression alone.

Ablation studies. We conduct ablation studies on Suite 1 to measure the contribution of each component of ATLAS. Specifically, we construct three variants that each remove one component from the full method while sharing all other training and evaluation settings: (i) no concurrent training with atomic tasks, (ii) no task-wise progress potential shaping, and (iii) no phase-conditioned atomic-to-composite hidden-state alignment. As shown in Tab. 4, removing just one component leads to substantial decline in success rates. The drop appears consistently on both atomic and composite tasks and under both the evaluation metrics. These results confirm that all three components are indispensable to ATLAS, demonstrating the effectiveness of atomic-to-composite distillation.

5 Conclusion

We propose **ATLAS**, an end-to-end reinforcement learning framework for addressing the challenges of composite-task execution by jointly leveraging atomic and composite skills during training. Rather than learning composite behaviors in isolation, ATLAS jointly collects atomic and composite trajectories, allowing the policy's existing competence on atomic skills to provide effective supervision for learning their composition. First, we introduce task-wise progress potential shaping, which learns reusable progress potentials from both successful and failed atomic rollouts and composes them across task stages to provide dense, phase-aware credit assignment. We further introduce phase-conditioned atomic-to-composite hidden-state alignment, which reduces ambiguity about the currently active subtask by aligning composite-conditioned representations with the corresponding atomic-conditioned representations at each phase. Across RoboLab simulation benchmarks, ATLAS consistently improves composite-task success over the base policy and alternative RL objectives, while maintaining or improving atomic-task competence. Our real-robot experiments further demonstrate that these gains transfer beyond simulation, highlighting the effectiveness of jointly exploiting atomic-task knowledge for scalable composite-task learning.

AI USE STATEMENT

In this work, we used generative AI tools for method implementation and assistance with translation. We also used generative AI tools to suggest experimental parameters and to summarize and analyze existing literature. We did not use generative AI tools to generate the core research ideas, formulate the main hypotheses, or develop the central methodological concepts of this work. All AI-assisted code and implementation suggestions were independently inspected, verified, and tested by the authors before use. Literature summaries were cross-checked against the original sources. Suggested experimental parameters were treated only as preliminary recommendations and were independently validated through our own experiments. We take responsibility for the final content of this work, including any text, code, analyses, or other artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the task suites, simulation and real-robot setups, training procedure, hyperparameters, and evaluation protocols in the main paper and appendix. We report the model initialization, trainable components, environment settings, optimization details, reward configurations, and evaluation criteria. All RL methods share the same base initialization and matched optimization settings, and simulation evaluations use identical scenes under a fixed evaluation seed.

REFERENCES

Niket Agarwal, Arslan Ali, Jon Allen, Martin Antolini, Adeline Aubame, Alisson Azzolini, Junjie Bai, Maciej Bala, Yogesh Balaji, Josh Bapst, et al. Cosmos 3: Omnimodal world models for physical ai. *arXiv preprint arXiv:2606.02800*, 2026.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Alisson Azzolini, Junjie Bai, Hannah Brandon, Jiaxin Cao, Prithvijit Chattopadhyay, Huayu Chen, Jinju Chu, Yin Cui, Jenna Diamond, Yifan Ding, et al. Cosmos-reason1: From physical common sense to embodied reasoning. *arXiv preprint arXiv:2503.15558*, 2025.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

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.

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.

Xiaowei Cai, Yunuo Cai, Bingao Chen, Jingxiao Chen, Zhi Chen, Siyuan Feng, Tengyu Hou, Jingshun Huang, Han Jiang, Runkun Ju, et al. τ_0-VLA: a Hierarchical Robot Foundation Model with World-Model-Guided Test-Time Computation. *arXiv preprint arXiv:2608.16885*, 2026.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *CoRL*, 2020.

Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, et al. πrl: Online rl fine-tuning for flow-based vision-language-action models. *arXiv preprint arXiv:2510.25889*, 8, 2025a.

Qianzhong Chen, Hau Zheng, Justin Yu, Suning Huang, Jiankai Sun, Ken Goldberg, Chuan Wen, Pieter Abbeel, Yide Shentu, Philipp Wu, et al. Sarm2: Multi-task stage aware reward modeling for self improving robotic manipulation. *arXiv preprint arXiv:2606.10305*, 2026.

Yangtao Chen, Zixuan Chen, Nga Teng Chan, Junting Chen, Junhui Yin, Jieqi Shi, Yang Gao, Yong-Lu Li, and Jing Huo. Robohiman: A hierarchical evaluation paradigm for compositional generalization in long-horizon manipulation. *arXiv preprint arXiv:2510.13149*, 2025b.

Sanjiban Choudhury. Process reward models for llm agents: Practical framework and directions. *arXiv preprint arXiv:2502.10325*, 2025.

Shaoqi Dong, Chaoyou Fu, Haihan Gao, Yi-Fan Zhang, Chi Yan, Chu Wu, Xiaoyu Liu, Yunhang Shen, Jing Huo, Deqiang Jiang, et al. Vita-vla: Efficiently teaching vision-language models to act via action expert distillation. *arXiv preprint arXiv:2510.09607*, 2025.

Yiguo Fan, Shuanghao Bai, Xinyang Tong, Pengxiang Ding, Yuyang Zhu, Hongchao Lu, Fengqi Dai, Wei Zhao, Yang Liu, Siteng Huang, Zhaoxin Fan, Badong Chen, and Donglin Wang. Long-VLA: Unleashing long-horizon capability of vision language action model for robot manipulation. In *CoRL*, 2025.

Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, et al. Molmoact2: Action reasoning models for real-world deployment. *arXiv preprint arXiv:2605.02881*, 2026.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. In *Findings of the association for computational linguistics: EMNLP 2020*, 2020.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Fuhao Li, Wenxuan Song, Han Zhao, Jingbo Wang, Pengxiang Ding, Donglin Wang, Long Zeng, and Haoang Li. Spatial forcing: Implicit spatial representation alignment for vision-language-action model. In *ICLR*, 2026.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *ICLR*, 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICLR*, 2023.

Isabella Liu, An-Chieh Cheng, Rui Yan, Geng Chen, Ri-Zhao Qiu, Xueyan Zou, Sha Yi, Hongxu Yin, Xiaolong Wang, and Sifei Liu. Long-horizon manipulation via trace-conditioned vla planning. *arXiv preprint arXiv:2604.21924*, 2026a.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. In *International Conference on Learning Representations*, volume 2025, pp. 29982–30009, 2025.

Yuhan Liu, Xinyu Zhang, Litao Liu, and Abdeslam Boularias. Foresight residual rl for long-horizon robot manipulation with vision-language-action models. *arXiv preprint arXiv:2607.16506*, 2026b.

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.

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, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. Liv: Language-image representations and rewards for robotic control. In *ICML*, 2023a.

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 *ICLR*, 2023b.

Yecheng Jason Ma, Joey Hejna, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, et al. Vision language models are in-context value learners. In *ICLR*, 2025.

Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Munoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, et al. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, 1999.

Yiwen Pang, Bo Zhou, Changjin Li, Xuanhao Wang, Shengxiang Xu, Deng-Bao Wang, Peng Cheng, Shimin Di, Jingkuan Song, and Min-Ling Zhang. AtomBridge: Agentic VLA inference plugin for long-horizon tasks in scientific experiments. *arXiv preprint arXiv:2602.09430*, 2026.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a Vision-Language-Action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for Thin Deep Nets. In *ICLR*, 2015.

Stéphane Ross, Geoffrey Gordon, and Drew 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*, 2011.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. In *ICLR*, 2025.

Lucy Xiaoyang Shi, brian ichter, Michael Robert Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. In *ICML*, 2025.

Yunlong Song, Kexin Shi, Robert Penicka, and Davide Scaramuzza. Learning perception-aware agile flight in cluttered environments. In *ICRA*, 2023.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for bert model compression. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, 2019.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *NeurIPS*, 2020.

Zenan Wu, Bingqing Wei, Lu Liu, Zheqi He, Xi Wang, Jiakang Liu, Zehui Li, Guocai Yao, Jing-Shu Zheng, Xi Yang, et al. Atom-bench: A real-world benchmark for atomic skills and compositional generalization in manipulation policies. *arXiv preprint arXiv:2606.16826*, 2026.

Zhiheng Xi, Chenyang Liao, Guanyu Li, Zhihao Zhang, Wenxiang Chen, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, Wei Wu, et al. Agentprm: Process reward models for llm agents via step-wise promise and progress. In *Proceedings of the ACM Web Conference 2026*, pp. 4184–4195, 2026.

Bingxin Xu, Yuzhang Shang, and Emilio Ferrara. Don't drop the baton: Long-horizon robot manipulation via agentic subtask exploration and transition-aware memory. *arXiv preprint arXiv:2608.16889*, 2026.

Feng Xu, Guangyao Zhai, Xin Kong, Tingzhong Fu, Daniel FN Gordon, Xueli An, and Benjamin Busam. Stare-vla: Progressive stage-aware reinforcement for fine-tuning vision-language-action models. *arXiv preprint arXiv:2512.05107*, 2025.

Daniel Yang, Davin Tjia, Jacob Berg, Dima Damen, Pulkit Agrawal, and Abhishek Gupta. Rank2reward: Learning shaped reward functions from passive video. In *ICRA*. IEEE, 2024.

Jenai Xuning Yang, Rishit Dagli, Alex Zook, Hugo Hadfield, Ankit Goyal, Stan Birchfield, Fabio Ramos, and Jonathan Tremblay. Robolab: A high-fidelity simulation benchmark for analysis of task generalist policies. *arXiv preprint arXiv:2604.09860*, 2026a.

Ran Yang, Zijian An, Lifeng Zhou, and Yiming Feng. Seqvla: Sequential task execution for long-horizon manipulation with completion-aware vision-language-action model. *arXiv preprint arXiv:2509.14138*, 2025.

Tianshuo Yang, Guanyu Chen, Yutian Chen, Zhixuan Liang, Yitian Liu, Zanxin Chen, Chunpu Xu, Haotian Liang, Jiangmiao Pang, Yao Mu, et al. Hivla: A visual-grounded-centric hierarchical embodied manipulation system. *arXiv preprint arXiv:2604.14125*, 2026b.

Yue Yang, Shuo Cheng, Yu Fang, Homanga Bharadhwaj, Mingyu Ding, Gedas Bertasius, and Daniel Szafir. Lilo-vla: Compositional long-horizon manipulation via linked object-centric policies. *arXiv preprint arXiv:2602.21531*, 2026c.

Zhendong Yang, Zhe Li, Mingqi Shao, Dachuan Shi, Zehuan Yuan, and Chun Yuan. Masked generative distillation. In *ECCV*, 2022.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*, 2017.

Likui Zhang, Tao Tang, Zhihao Zhan, Xiuwei Chen, Zisheng Chen, Jianhua Han, Jiangtong Zhu, Pei Xu, Hang Xu, Hefeng Wu, Liang Lin, and Xiaodan Liang. Atomicvla: Unlocking the potential of atomic skill learning in robots. In *CVPR*, 2026a.

Likui Zhang, Tao Tang, Zhihao Zhan, Xiuwei Chen, Zisheng Chen, Jianhua Han, Jiangtong Zhu, Pei Xu, Hang Xu, Hefeng Wu, et al. Atomicvla: Unlocking the potential of atomic skill learning in robots. *arXiv preprint arXiv:2603.07648*, 2026b.

Rongyu Zhang, Menghang Dong, Yuan Zhang, Liang Heng, Xiaowei Chi, Gaole Dai, Li Du, Dan Wang, Yuan Du, and Shanghang Zhang. Mole-vla: Dynamic layer-skipping vision language action model via mixture-of-layers for efficient robot manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026c.

Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. *NeurIPS*, 2023.

A Experimental Details

A.1 Task suite details

We construct four task suites using the simulation scenes and object assets of RoboLab (Yang et al., 2026a). Each suite consists of two atomic tasks which pick one target object and place it in a container, and two composite tasks which perform both atomic tasks in sequence, in either order. The four tasks within a suite share the same scene, objects, and initial-state distribution, and differ only in the instruction, the success condition, and the episode budget. Tab. A1 lists the target and distractor objects of each suite, and Tab. A2 the instructions, which are given to the policy verbatim. Suites 1 and 3 use two different kitchen-object scenes, while Suites 2 and 4 share the same toy-cleanup scene and differ only in which pair of blocks is the target.

A.2 Simulation setup

We build all simulated tasks in RoboLab on top of Isaac Lab 2.3 (Mittal et al., 2025) and Isaac Sim 5.1. The robot is a 7-DoF Franka Panda arm with a Robotiq 2F-85 parallel jaw gripper as in the DROID platform (Khazatsky et al., 2024) on which $\pi_{0.5}$ was fine-tuned. The policy commands joint positions at 15 Hz while physics is simulated at 60 Hz with four physics steps per control step. Observations come from an over-the-shoulder RGB camera and a wrist RGB camera mounted on the gripper. Both are rendered at 640×360 with the balanced RTX rendering mode and resized with padding to 224×224 before entering the policy. At the start of each episode every object except the robot and the table is placed at a randomized position. Each parallel environment shifts all objects by up to 7 cm from their default layout and every reset adds a further displacement of up to 7 cm with a rotation of up to 0.5 rad, so every episode starts from a different scene. Training and evaluation use different randomization seeds, with the fixed evaluation seed ensuring that every method is evaluated on the same scenes. The simulation results are reported in Tab. 2.

A.3 Physical robot setup

For physical experiments, we use a Franka Research 3 arm fitted with a Robotiq parallel gripper, arranged as in DROID (Khazatsky et al., 2024). The scene is captured by a ZED Mini mounted on the wrist and two external ZED 2i cameras placed around the table. The policy receives the wrist view and the left external view. A workstation with an RTX 3090 Ti runs the DROID client, which records the camera streams and queries a remote inference server equipped with an RTX A6000. The client relays the predicted actions over ZeroRPC to a dedicated control computer that drives the arm and the gripper through Polymetis. Whereas the official $\pi_{0.5}$ checkpoint is trained on real robot data from DROID, our RL training uses only simulated rollouts and no real robot data. We evaluate two composite tasks taken from the simulation suites, placing the banana before the Rubik's cube from Suite 1 and the green block before the red block from Suite 2. For each task, we evaluate $\pi_{0.5}$ and our policy trained on the corresponding suite for 10 episodes each, randomizing the object positions in every episode to assess robustness to the initial scene. The results are reported in Tab. 3.

A.4 Training details

Trainable parameters. We fine-tune $\pi_{0.5}$ (Physical Intelligence et al., 2025) with low rank adaptation (Hu et al., 2021) of rank 32 on the language model of the PaliGemma (Beyer et al., 2024) backbone and train all parameters of the action expert together with its action and timestep projections in full. The vision encoder stays frozen. A value head composed of a 4-layer MLP with ReLU activations reads the mean of the output tokens of the backbone and predicts the expected discounted return from the current state. It is trained from scratch and serves as the critic for PPO (Schulman et al., 2017).

Stochastic denoising for likelihood computation. Flow matching (Lipman et al., 2023) generates actions by integrating a deterministic ODE and therefore provides no tractable action likelihood. Following the Flow SDE formulation of π_{RL} (Chen et al., 2025a), we make one denoising step stochastic by injecting Gaussian noise and integrate the remaining steps deterministically. The log likelihood of an action chunk is computed as the Gaussian log density of the stochastic transition.

Table A1: Target and distractor objects of the four task suites.

Suite	Target objects	Distractor objects
Suite 1	banana, Rubik's cube	bowl, mug
Suite 2	red block, green block	yellow block, blue block, three Rubik's cubes, two lizard figurines, birdhouse
Suite 3	mug, banana	bowl, ketchup bottle, three Rubik's cubes
Suite 4	red block, yellow block	green block, blue block, three Rubik's cubes, two lizard figurines, birdhouse

Table A2: Instructions of all tasks in the four task suites.

Suite	Category	Instruction
Suite 1	Atomic	Put the banana in the grey bin
	Atomic	Put the rubiks cube in the grey bin
	Composite	Put the banana in the grey bin, then put the rubiks cube in the grey bin
	Composite	Put the rubiks cube in the grey bin, then put the banana in the grey bin
Suite 2	Atomic	Put the red block in the grey bin
	Atomic	Put the green block in the grey bin
	Composite	Put the red block in the grey bin, then put the green block in the grey bin
	Composite	Put the green block in the grey bin, then put the red block in the grey bin
Suite 3	Atomic	Put the mug in the grey bin
	Atomic	Put the banana in the grey bin
	Composite	Put the mug in the grey bin, then put the banana in the grey bin
	Composite	Put the banana in the grey bin, then put the mug in the grey bin
Suite 4	Atomic	Put the red block in the grey bin
	Atomic	Put the yellow block in the grey bin
	Composite	Put the red block in the grey bin, then put the yellow block in the grey bin
	Composite	Put the yellow block in the grey bin, then put the red block in the grey bin

We draw the stochastic step uniformly from the first three of the 10 denoising steps, which carry the highest noise level.

Additional optimization settings. We warm up the critic during the first two epochs by training only the value head while the learning rate of the policy is kept at zero. Advantages are normalized per task by subtracting the task mean and dividing by a single standard deviation shared across all tasks. We weight the policy and value losses of each chunk by the inverse frequency of its group in the current rollout, where chunks are grouped by task and number of completed subtasks. We bound these weights by clipping small group counts and large weights, then rescale them to unit mean. Every run uses 128 parallel environments, which ours splits evenly over the four tasks of a suite and the baselines split over the two composite tasks only. All training hyperparameters are listed in Tab. A3.

A.5 Evaluation protocol

We evaluate each trained policy after 10 training epochs without checkpoint selection. Every simulation task is evaluated over 24 episodes that start from the randomized scenes of a fixed evaluation seed, ensuring that every policy faces identical scenes. Each physical task is evaluated over 10 episodes with the object positions randomized in every episode. An episode ends when the task succeeds or when its budget of 450 steps for atomic tasks or 900 steps for composite tasks runs out.

Table A3: Training hyperparameters, shared by all suites.

Hyperparameter	Value
Policy	
Initialization	$\pi_{0.5}$ DROID checkpoint
LoRA rank / scaling α	32 / 32, on the language model of PaliGemma
Vision encoder	frozen
Value head	MLP 2048, 512, 256, 128, 1 with ReLU
Action chunk length	15 steps (1 s at 15 Hz)
Denoising steps	10
Noise level	0.5
Precision	bfloat16 autocast
PPO	
Training epochs	10
PPO passes per epoch	4
Global / micro batch size	1920 / 120 chunks
Optimizer	Adam (β_1=0.9, β_2=0.95, ϵ=10^{-8})
Learning rate (policy / value head)	10^{-4} / 10^{-4}
Learning rate schedule	cosine over 200 steps to $0.1\times$
Weight decay	0.01
Gradient clipping	1.0
Discount γ (per chunk)	0.99
GAE λ	0.95
Clip ratio / dual clip	0.2 / 3.0
Value loss	Huber (δ=10), value clip 0.2
Entropy bonus / KL penalty	0 / 0
Training seed	42
PPO w/ GVL	
Shaping weight	0.2
Value model	zero-shot Qwen3-VL-32B at temperature 0
Frames	one 448×252 over-shoulder frame per chunk
PPO w/ Ours	
Shaping weight λ_{prog}	0.2
Label discount γ_Φ	0.94
Potential features	mean of image tokens at layers 4, 8, 12
Potential model	ridge regression (α=30), output clipped to $[0, 1]$
Distillation weight	0.5
Distillation layer	6
Distillation tokens	language tokens
Distillation samples per micro batch	16 composite chunks
Group weighting	minimum group count 50, maximum weight 10

At evaluation, all policies act with their standard inference procedure without any noise injection used for training.

We report two metrics. *Order-preserving success rate* counts an episode as successful when every target object is inside the bin and the targets of a composite task were placed in the instructed order. *Strict-selection success rate* additionally counts an episode as a failure as soon as a non-target object is placed in the bin, where the second target of a composite task is treated as a non-target until the first target has been placed. An object counts as placed when its centroid stays inside the footprint of the bin and below a height of 2 cm above its rim for at least 10 consecutive steps, thereby excluding objects that are only carried over the bin.

A.6 Baselines

A.6.1 Pre-trained VLA models

$\pi_{0.5}$. $\pi_{0.5}$ (Physical Intelligence et al., 2025) is a vision-language-action model designed for open world generalization. It pairs a 3B parameter PaliGemma backbone with a 300M parameter action expert that generates continuous action chunks by flow matching while attending to the backbone

representation of the images and the instruction. Its pre-training co-trains on data from multiple robot embodiments together with web data and high-level subtask annotations, which transfers semantic knowledge into low-level control. We use the publicly released checkpoint that is further fine-tuned on DROID with joint position actions and therefore matches the Franka robot and action interface of RoboLab. The policy receives the two camera images and the instruction together with the joint and gripper positions of the robot, which it encodes as discretized tokens in its text prompt. It predicts chunks of 15 actions that are executed open loop before the next observation is taken. We evaluate this checkpoint zero-shot under the same protocol as all other policies, and it serves as the initialization of our method and of both RL baselines.

GR00T N1.7. GR00T N1.7 (Bjorck et al., 2025) is an open vision-language-action model for generalist robot manipulation. It couples a Cosmos-Reason2-2B Azzolini et al. (2025) vision-language backbone with a flow-matching action head that produces continuous action chunks conditioned on the backbone representation of the images, the instruction, and the proprioceptive state. It is trained across a large set of robot embodiments under a shared embodiment-tag interface, so a single checkpoint serves many robots and action spaces. We use the publicly released DROID checkpoint, which exposes the joint-position and gripper interface that RoboLab expects. The policy receives the left exterior and left wrist images at 180×320, stretched rather than letterboxed so that no black padding enters the visual field, together with the joint and gripper state. It predicts an action chunk of which the first eight actions are executed open loop before the next observation is taken, and its gripper output is thresholded to a binary open-close command.

MolmoAct 2. MolmoAct 2 (Fang et al., 2026) is an open VLA built on the Molmo family of vision-language models. Its training emphasizes open, fully documented data, so its competence comes from a different pre-training mixture than the robot-centric models above. We use the publicly released MolmoAct2-DROID checkpoint, which matches the Franka robot and the action interface of RoboLab. The policy receives one exterior and one wrist image together with the instruction and the robot state, whose gripper component is rescaled to the convention the checkpoint was trained with. It predicts an action chunk that is executed open loop before the next observation is taken, and its gripper output is thresholded to a binary open-close command.

Cosmos3 Edge Policy. Cosmos 3 (Agarwal et al., 2026) is a family of omnimodal world models that jointly generate video and actions. Cosmos 3 Edge Policy is its variant for on-device deployment, post-trained into a robot policy on DROID (Khazatsky et al., 2024). The policy produces actions with the same diffusion sampler that synthesizes video, which we run with four UniPC (Zhao et al., 2023) steps, a shift of 5.0 and a guidance scale of 3.0. It observes a single composite frame that places the wrist view above two external views from opposite sides of the table, together with the joint and gripper positions of the robot and the instruction. Each call predicts a chunk of 32 joint position actions, about two seconds at 15 Hz, which is executed open loop before the next observation. We use the publicly released DROID checkpoint, whose action space matches the Franka robot and the joint position interface of RoboLab. We evaluate it zero-shot under the same protocol as all other policies.

A.6.2 Reinforcement learning

PPO. Proximal policy optimization (Schulman et al., 2017) updates the policy with a clipped surrogate objective that keeps each update close to the policy that collected the data, using advantages estimated by a learned critic. We fine-tune $\pi_{0.5}$ with PPO under the same training framework and hyperparameters as our method. The reward is one when the task is completed and zero otherwise, so a composite episode is rewarded only after both objects are placed in the instructed order. Because the target capability is composite execution, this baseline trains only on the two composite tasks of each suite and assigns 64 parallel environments to each. It is evaluated on all four tasks of the simulation suites after the same 10 training epochs.

PPO with GVL progress reward. GVL (Generative Value Learning) (Ma et al., 2025) estimates task progress with a pre-trained vision-language model (VLM) without any task-specific training. The model is given the frames of an episode in shuffled order and asked to predict the completion percentage of each frame, thereby forcing it to judge every frame by its content instead of assigning

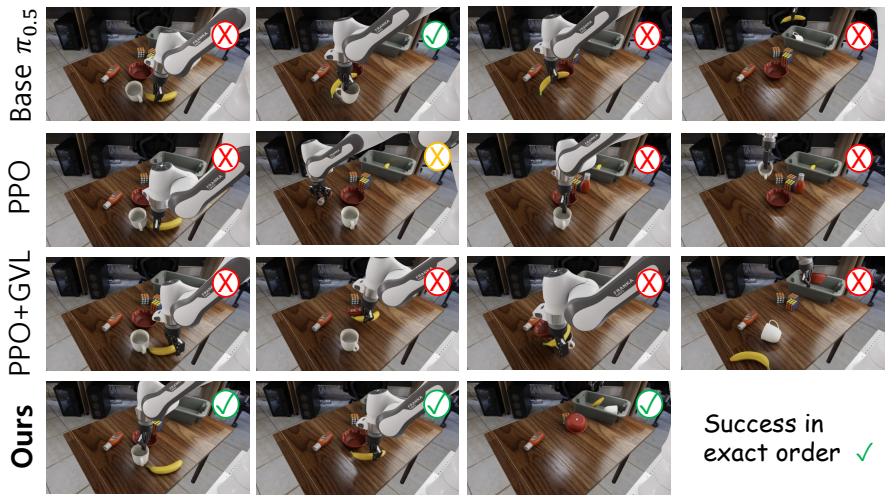

Figure A1: **Qualitative simulation results on RoboLab composite task from Suite 3.** Using the same visual notation as in Fig. 3, we compare all methods in the same scene under the instruction to place the mug before the banana. The baseline policies exhibit wrong-order target-object selections or wrong-object selections, whereas ATLAS follows the instructed sequence and achieves strict-selection success.

increasing values to consecutive frames. We incorporate the GVL progress estimate into PPO as a dense reward, utilizing Qwen3-VL-32B (Bai et al., 2025) with greedy decoding as the VLM in zero-shot manner. We query the model with the task instruction and the external camera image of every action chunk, providing the first frame of the episode as the reference for zero completion. The predicted percentages scaled to $[0, 1]$ serve as the potential Φ and add $\lambda(\gamma\Phi_{t+1} - \Phi_t)$ to the success reward at the end of each chunk with $\lambda = 0.2$ and $\gamma = 0.99$, the same form in which our method adds its learned potential. All other settings follow the PPO baseline, including training only on the two composite tasks of each suite.

A.7 Compute Resources

All experiments were conducted on a single node equipped with eight NVIDIA B300 GPUs (288 GB each), 128 CPU cores, and 4 TB of system memory. Each RL run, including our method and both PPO baselines, uses four GPUs to host 128 parallel RoboLab environments and the PPO learner. Training our method for 10 epochs takes approximately 2.5 hours, and evaluating the final checkpoint with 24 episodes per task takes an additional 20 minutes on the same GPUs, for a total of roughly 12 GPU-hours per run. The PPO baselines incur comparable cost, and the GVL reward baseline additionally serves a 32B vision-language model with vLLM on one of the four training GPUs. To fit the progress potentials, we collect hidden states from 4 to 16 rollout rounds per suite, each requiring about 30 minutes on a single GPU, and fit the ridge regression heads on CPU.

B Additional Experimental Results

B.1 Additional simulation results

Figs. A1 and A2 provide additional qualitative simulation results for Suite 3 and 4, respectively. For the Suite 3 task requiring the mug followed by the banana, the base $\pi_{0.5}$ and PPO with the GVL reward select the target objects in the wrong order. Standard PPO additionally selects wrong objects during execution, preventing completion of the instructed sequence. In contrast, ATLAS selects the mug and banana in the specified order and achieves strict-selection success.

For the Suite 4 task requiring the red block followed by the yellow block, all baselines first interact with wrong objects and subsequently fail to complete the required target-object placements in the instructed sequence. ATLAS instead selects only the red and yellow blocks and completes both subgoals in order, yielding strict-selection success.

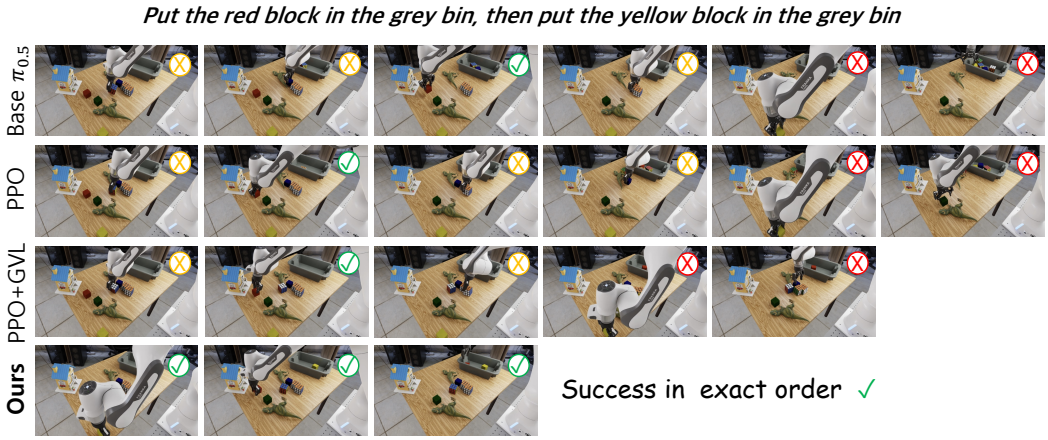

Figure A2: **Qualitative simulation results on RoboLab composite task from Suite 4.** Using the same visual notation as in Fig. 3, we compare all methods in the same scene under the instruction to place the red block before the yellow block. Baseline policies select wrong objects or fail to complete the required target-object placements, whereas ATLAS selects only the instructed objects and achieves strict-selection success.

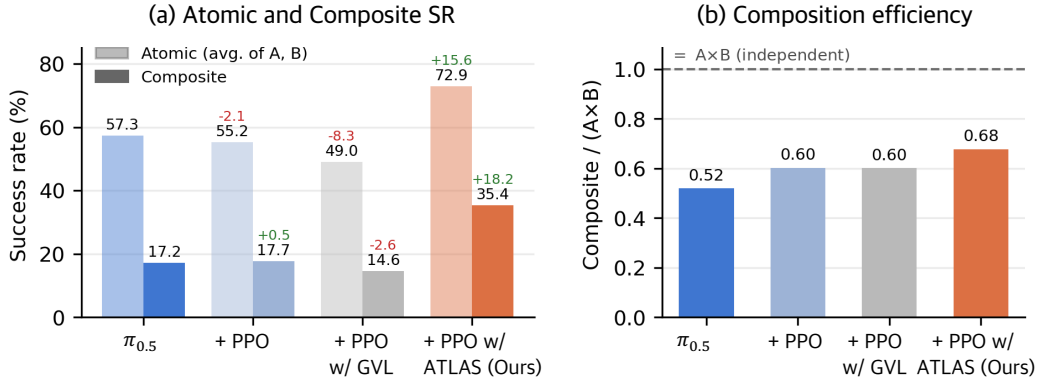

Figure A3: **ATLAS improves both skill learning and the composition of learned skills.** (a) Average success rates on atomic tasks and composite tasks. While standard PPO and GVL yield limited gains, ATLAS substantially improves both atomic and composite success rates. (b) Composition efficiency, defined as the observed composite success rate normalized by the expected rate under independent atomic execution ($A \times B$). The dashed line indicates the independent-execution expectation. ATLAS achieves the highest ratio, indicating that it more effectively translates atomic-skill competence into composite-task success.

As shown in Fig. A3, ATLAS improves both atomic and composite success rates while achieving the highest composition efficiency, suggesting that its gains stem from both stronger skill learning and more effective skill composition.

B.2 Additional analysis on task-wise progress potential shaping

Progress potential analysis. We evaluate whether each progress signal reflects eventual skill completion rather than simply increasing along the trajectory. We use the area under the receiver operating characteristic curve (AUROC), which measures how well a scalar progress score separates observations that eventually lead to successful completion from those that do not; an AUROC of 0.5 corresponds to chance-level discrimination. As shown in Fig. A4, temporal progress is consistently below chance, while the VLM-based signal varies substantially across tasks. In contrast, our task-wise potential achieves consistently high AUROC, indicating that it captures progress that is strongly predictive of eventual skill completion.

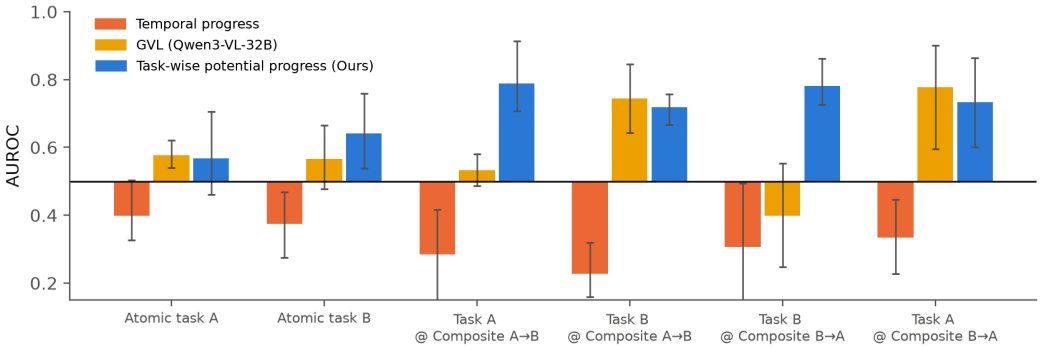

Figure A4: **Success discriminability of progress rewards.** AUROC measures how well each progress signal distinguishes eventual skill success from failure. Our task-wise potential is consistently more predictive of successful completion than temporal progress and GVL.

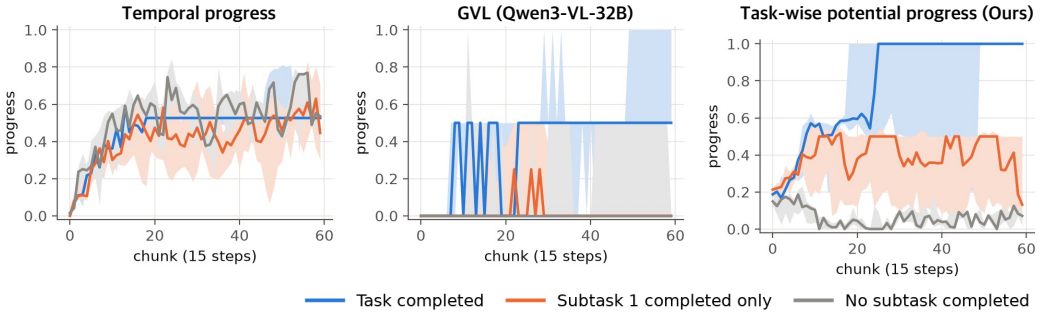

Figure A5: **Comparison of progress rewards over trajectory outcomes.** Our task-wise potential provides dense supervision that separates full success, partial success, and failure trajectories. Temporal progress evolves similarly across outcomes, while VLM-based rewards remain sparse and unstable.

Progress reward behavior across outcomes. Fig. A5 further examines how each signal evolves over trajectories with different final outcomes. Temporal progress increases similarly even when no subtask is completed, while GVL changes mainly around salient completion events and therefore remains relatively sparse. In contrast, our task-wise potential stays low for failed trajectories, reaches an intermediate level when only the first subtask is completed, and continues toward full progress only for successful trajectories. This shows that the proposed potential provides dense within-trajectory supervision while preserving clear separation between failure, partial success, and full success.