MULTI-STUDENT ON-POLICY SELF-DISTILLATION WITH ALTERNATING REINFORCEMENT LEARNING FOR ROBUST VISION-LANGUAGE-ACTION MODELS

Hayeon Kim[1,*] **Byoung Hyun Lee**[1,*] **Jaehyun Cho**[3] **Jason Park**[3]
YongHyeok Seo[4,5] **Yuhyeon Hwang**[5] **Dongin Shin**[5] **Se Young Chun**[1,2,3,†]

[1]Department of Electrical and Computer Engineering, Seoul National University
[2] INMC, [3] IPAI, Seoul National University
[4]Korea University
[5]Korea Electronics Technology Institute

[*]Equal contribution. [†]Corresponding author.

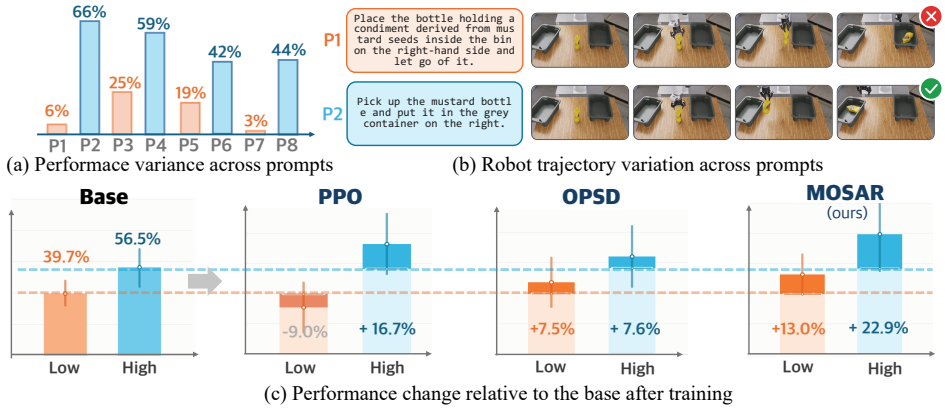
ABSTRACT

Vision-language-action (VLA) models have emerged as a paradigm for general-purpose robotic control by translating visual observations and natural language instructions into executable actions. Recent reinforcement learning (RL) post-training further improves their task performance through online environment interaction and task-completion rewards. However, VLA policies remain sensitive to prompt variation, exhibiting substantially different success rates even across semantically equivalent instructions. For the same task, high-performing prompts can reliably elicit successful behavior, whereas low-performing prompts often provide few successful trajectories for direct RL. To exploit this performance asymmetry, we propose **M**ulti-student **O**n-policy **S**elf-distillation with **A**lternating **R**einforcement learning (**MOSAR**). MOSAR uses a high-performing prompt as a teacher for multiple semantically equivalent student prompts and distills teacher actions at student-visited observations, enabling cross-prompt behavior transfer within a shared VLA policy. Since distillation alone cannot exceed its teacher, MOSAR alternates RL and distillation with separate optimizer states and adaptively weights distillation according to task-specific teacher–student performance gaps, allowing policy improvement and cross-prompt knowledge transfer to reinforce each other. From extensive evaluations on RoboLab, REALM, and a real Franka robot, MOSAR consistently improves success on unseen prompts over RL- and OPSD-based baselines while maintaining strong performance on targeted and unseen tasks. Additional evaluations under visual distractors further demonstrate robustness beyond the prompt variations explicitly addressed during training.

1 INTRODUCTION

Vision-language-action (VLA) models (Zitkovich et al., 2023; Kim et al., 2025; Black et al., 2025b;a; NVIDIA et al., 2026) have emerged as a paradigm for general-purpose robotic control, enabling robots to translate visual observations and natural language instructions into executable actions. Recent advances in reinforcement learning (RL) (Hu et al., 2025; Li et al., 2026a; Chen et al., 2025; Wang et al., 2026b) have further improved VLA performance by optimizing pretrained policies through online environment interaction and task-completion rewards, allowing them to acquire capabilities beyond those provided by supervised fine-tuning (SFT).

Despite these advances, VLA models remain sensitive to prompt variation: semantically equivalent instructions can induce substantially different success rates (Kim et al., 2026a; Dong et al., 2026; Liu et al., 2025b). For the same task, some prompts reliably elicit successful behavior while others yield few successful trajectories, making direct RL under low-performing prompts less informative. (Fig. 1(a),(b)) At the same time, this discrepancy is itself a resource: every phrasing conditions the



(a) Performace variance across prompts (b) Robot trajectory variation across prompts

(c) Performance change relative to the base after training

Figure 1: **Motivation and performance gains of combining PPO and OPSD.** (a) Performance variation across meaning-preserving prompts for the same task. Despite sharing the same goal, different prompt lead to substantial performance differences. (b) Robot trajectories also vary across prompts. (c) PPO primarily benefits high-success-rate prompts, while OPSD improves prompts across the success-rate spectrum. Combining PPO and OPSD shifts both low- and high-success-rate prompts toward higher success rates.

same parameters, so the behavior elicited by a strong phrasing can be transferred to weaker ones without demonstrations or an external teacher. A post-training method should therefore (i) close the success gap among the phrasings of a task, (ii) raise all of them beyond the ceiling of the strongest phrasing, and (iii) preserve performance on tasks it never trained on. Existing approaches address at most one of these: RL fine-tuning raises the ceiling but mostly for prompts that already succeed (Fig. 1(c)), while on-policy self-distillation closes the gap but can never exceed its teacher.

To this end, we propose **M**ulti-student **O**n-policy **S**elf-distillation with **A**lternating **R**einforcement learning (**MOSAR**). MOSAR adaptively performs multi-prompt on-policy self-distillation by treating high-performing prompts as teachers and alternative prompts as students based on their task-specific performance gaps, transferring teacher actions at student-visited observations through on-policy self-distillation (i). We then alternate RL and distillation with separate optimizer states, allowing RL to improve the shared policy (ii) while subsequent distillation propagates the updated teacher behavior across student prompts. By separating and alternating the RL and distillation updates, MOSAR both improves the teacher policy through reward optimization and propagates these improvements across prompt variations (iii) (Fig. 1(c)).

We evaluate MOSAR on RoboLab (Yang et al., 2026), REALM (Sedlacek et al., 2026), and a real Franka robot. MOSAR achieves 53.2% success on unseen prompts in RoboLab, compared with 33.7% for PPO and 41.2% for OPSD-S, and reaches 75.9% on unseen tasks. On REALM, it achieves 61.0% on unseen prompts, compared with 56.8% for both OPSD baselines. Real-robot experiments further confirm improvements across prompt variations, while RoboLab experiments also show improved robustness to visual distractors.

Our contributions are summarized as follows:

- We propose **MOSAR**, a multi-student on-policy self-distillation framework for VLAs, where a high-performing prompt supervises semantically equivalent student prompts at their visited observations. The privileged context is another phrasing, not extra information, enabling cross-prompt transfer within a shared policy without demonstrations or external teachers (i); adding students provably tightens coverage bounds on unseen paraphrases.
- Since distillation alone cannot exceed its teacher, we alternate RL and distillation with separate optimizer states, regenerating teacher targets after each RL step, and adaptively weight distillation according to task-specific teacher–student performance gaps, allowing policy improvement and prompt-level knowledge transfer to complement each other (ii).
- We evaluate MOSAR on RoboLab, REALM, and a real Franka robot, verifying improved robustness to unseen prompt variations, while preserving strong performance on targeted and unseen tasks (iii), with robustness gains under goal swapping and visual distractors.

2 Related Work

Robustness in vision-language-action (VLA) models. Vision-language-action (VLA) models have advanced from RT-1 (Brohan et al., 2023), RT-2 (Zitkovich et al., 2023), and OpenVLA (Kim et al., 2025) to diffusion- and flow-based policies such as π_0 (Black et al., 2025b), $\pi_{0.5}$ (Black et al., 2025a), Cosmos Policy (Kim et al., 2026b), and Cosmos 3 (NVIDIA et al., 2026). Despite their strong performance, VLA models remain sensitive to task-preserving input variations. LIBERO-Para (Kim et al., 2026a) reports substantial degradation under semantically equivalent prompts, and multilingual instructions expose similar sensitivity (Dong et al., 2026). Task-irrelevant visual changes also affect VLA policies, while RL and supervised fine-tuning do not necessarily provide robustness beyond the training conditions (Liu et al., 2025b). Recent work improves instruction robustness through grounded semantic re-binding (Yin & Zhang, 2026).

On-policy self-distillation. On-policy distillation (OPD) mitigates exposure bias by supervising student-generated trajectories, building on interactive imitation learning (Ross et al., 2011) and extending to language models (Agarwal et al., 2024). On-policy self-distillation (OPSD) uses a single model as teacher and student under different contexts, often providing privileged information to the teacher (Zhao et al., 2026; Shenfeld et al., 2026; Hübotter et al., 2026; Li et al., 2026c). Several methods combine dense distillation supervision with RL through adaptive weighting or joint optimization (Tan et al., 2026; Lu et al., 2026; Liu et al., 2026; Pan et al., 2026). For VLAs, VLA-OPD (Zhong et al., 2026) distills an RL-trained expert along student rollouts, while ROAD-VLA (Wang et al., 2026a) addresses the gap between textual guidance and actions by constructing an advantage-guided teacher. Our approach instead dynamically assigns multiple student which share parameters while conditioned on distinct prompts and distills guidance from a strong-prompt teacher. We further integrate RL to improve performance beyond strong-prompt supervision alone.

3 Preliminaries

Flow-based VLA policies and RL fine-tuning. A flow-based VLA policy π_θ generates an action chunk $a \in \mathbb{R}^{H\times D}$ (H control steps, D action dimensions) by integrating a flow-matching velocity field from Gaussian noise to actions (Black et al., 2025b;a). To enable likelihood-based RL, Flow-GRPO (Liu et al., 2025a) converts the deterministic flow into SDE with Gaussian transitions, allowing the denoising process to be treated as an MDP and optimized with PPO (Schulman et al., 2017). From π_{RL} (Chen et al., 2025), we denote the PPO transition loss by $\ell_{\mathrm{R}}(z)$ and defer the sampler, MDP, and objective details to Apps. B.1 and B.3.

From on-policy distillation to self-distillation. On-policy distillation trains a student on its own rollouts with a teacher evaluated at the same states, which avoids the exposure bias of learning from teacher data (Ross et al., 2011; Agarwal et al., 2024). On-policy self-distillation (OPSD) goes one step further. Its teacher is the same model conditioned on privileged context c, such as a reference solution (Zhao et al., 2026; Shenfeld et al., 2026; Hübotter et al., 2026). For a query q and a student response $y \sim \pi_\theta(\cdot \mid q)$ with prefixes $y_{<i}$,

$$\mathcal{L}_{\mathrm{OPSD}}(\theta) = \mathbb{E}_{y\sim\pi_\theta(\cdot|q)}\Big[\sum_i \mathrm{KL}\big(\,\mathrm{sg}[\pi_\theta(\cdot \mid q, c, y_{<i})]\,\big\|\,\pi_\theta(\cdot \mid q, y_{<i})\big)\Big], \tag{1}$$

where sg denotes the stop-gradient. Supervision is dense at exactly the states the student visits, and since both roles share θ, the student improves by learning the trajectories of the privileged context.

Self-distillation for flow policies. For a flow policy the objective lifts to the denoising chain. With the Gaussian transitions of the stochastic sampler (App. B.1), the per-step KL between teacher and student kernels reduces to a squared error between their means (Li et al., 2026b). Motivated by this, we use the following tractable surrogate. Then, at an observation o from the student's rollout, the teacher is the same model with a prompt p^{T}, and its action $a^{\mathrm{T}} = \mathrm{sg}[\pi_\theta(o, p^{\mathrm{T}}; x_1)]$ with $x_1 \sim \mathcal{N}(0, I)$ is taken as a constant. The student policy under its prompt p^{S} is then trained with the flow-matching loss toward a^{T},

$$\ell_{\mathrm{D}}(o, p^{\mathrm{S}}, p^{\mathrm{T}}) = \mathbb{E}_{t,\epsilon}\Big[\frac{1}{HD}\big\|v_\theta\big(t\epsilon + (1-t)a^{\mathrm{T}},\, t \mid o, p^{\mathrm{S}}\big) - (\epsilon - a^{\mathrm{T}})\big\|_2^2\Big], \tag{2}$$

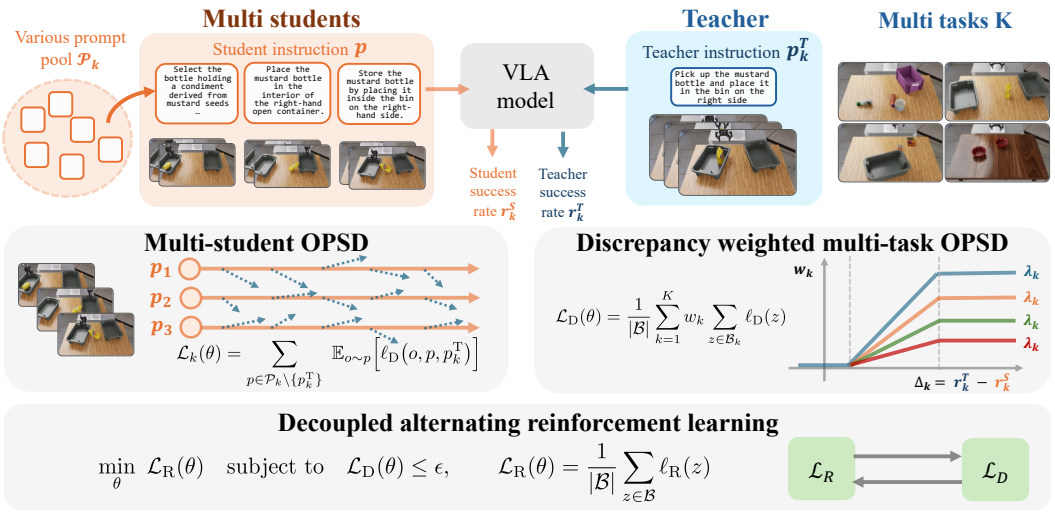


Figure 2: **Overview of MOSAR.** For each sampled task, we select multiple student prompts from a pool of semantically equivalent instructions and use the selected high-performing instruction as the teacher. We then perform multi-student OPSD, weighting the distillation objective based on the teacher–student performance gap across tasks. Finally, the distillation and RL objectives are optimized alternately to improve both prompt robustness and task performance.

with $\epsilon \sim \mathcal{N}(0, I)$ and $t \sim \mathrm{Beta}(1.5, 1)$. This is the loss of the base model with teacher-relabelled actions (Black et al., 2025b;a). We note that it is a guidance of the teacher on the student's observation and no teacher rollout is required in the distillation.

4 METHOD

Let each of the K training tasks k come with a prompt set $\mathcal{P}_k$ of semantically equivalent instructions, and each prompt $p \in \mathcal{P}_k$ induces a *prompt*-conditioned policy $\pi_\theta(\cdot \mid o, p)$ of tasks. The policies share every parameter, yet their success rates can differ widely (Kim et al., 2026a). Recall the three goals of Sec. 1: (i) close the gap among phrasings, (ii) raise the ceiling of the strongest phrasing, and (iii) keep performance on unseen tasks. Sec. 4.1 proposes multi-student OPSD, in which multiple phrasings of the same task act as students of a stronger teacher prompt (i), and theoretically characterizes its prompt robustness and parameter displacement against SFT. Sec. 4.2 proposes discrepancy-weighted multi-task OPSD, which decides how much distillation each task needs when the tasks lag behind unequally, limiting displacement of the shared parameters (iii). Sec. 4.3 proposes alternating RL, which lifts the ceiling of the strongest phrasing (ii). Fig. 2 gives an overview.

4.1 MULTI-STUDENT OPSD FOR ROBUST VLA MODEL

Multi-student distillation. For each task, we fix the prompt with a high initial success rate as the teacher prompt p_k^{T} and use the remaining prompts as student prompts. At each iteration, we sample M student prompts from $\mathcal{P}_k \setminus \{p_k^{\mathrm{T}}\}$ and collect rollouts under the teacher and student prompts. These samples form $\mathcal{B}_k = \mathcal{B}_k^{\mathrm{T}} \cup \mathcal{B}_k^{\mathrm{S}}$, collected under the teacher prompt and the sampled student prompts, respectively, with $\mathcal{B} = \bigcup_k \mathcal{B}_k$ and $\mathcal{B}^{\mathrm{S}} = \bigcup_k \mathcal{B}_k^{\mathrm{S}}$. Each student sample $z \in \mathcal{B}_k^{\mathrm{S}}$ carries its observation o_z, its rollout prompt $p_z^{\mathrm{S}} \in \mathcal{S}_k$, and its task index $k(z) = k$, and is paired with the teacher prompt $p_{k(z)}^{\mathrm{T}}$. The teacher prompt plays the role of the privileged context c in Eq. 1, replacing rather than augmenting the student prompt. Specifically, let $\mathcal{S}_k = \mathcal{P}_k \setminus \{p_k^{\mathrm{T}}\}$, d_p denote the observation distribution of rollouts under a student prompt p. Then the task objective is represented as:

$$\mathcal{L}_k(\theta) = \mathbb{E}_{p \sim \mathrm{Unif}(\mathcal{S}_k)}\, \mathbb{E}_{o \sim d_p}\left[\ell_{\mathrm{D}}\left(o, p, p_k^{\mathrm{T}}\right)\right]. \tag{3}$$

Each iteration estimates this objective using the sampled student prompts and their student-visited observations. The sample loss is $\ell_{\mathrm{D}}(z) = \ell_{\mathrm{D}}(o_z, p_z^{\mathrm{S}}, p_k^{\mathrm{T}})$. Thus, supervision on multiple student

rollouts trains the shared parameters across the task's prompt set, providing dense action targets at observations where successful student trajectories may be scarce.

Analysis of multi-student prompt robustness. Multi-student OPSD is designed to improve robustness by exposing the shared policy to the state distributions induced by multiple semantically equivalent student prompts. Intuitively, increasing the number and diversity of student prompts broadens the coverage of the task's prompt space, reducing the distance between an unseen paraphrase and its nearest covered training prompt. To formalize this intuition, let $\mathcal{S}_k^{(M)} = \{p_1, \ldots, p_M\}$ denote the set of student prompts covered for task k, and define its coverage radius over the set of semantically equivalent prompts $\mathcal{Q}_k$ as

$$\rho(\mathcal{S}_k^{(M)}) = \sup_{q \in \mathcal{Q}_k} \min_{p \in \mathcal{S}_k^{(M)} \cup \{p_k^{\mathrm{T}}\}} d(q, p), \tag{4}$$

where $d(\cdot, \cdot)$ measures prompt distance. Adding a student prompt cannot increase this radius:

$$\mathcal{S}_k^{(M)} \subseteq \mathcal{S}_k^{(M+1)} \quad \Longrightarrow \quad \rho(\mathcal{S}_k^{(M+1)}) \leq \rho(\mathcal{S}_k^{(M)}). \tag{5}$$

Thus, covering multiple student prompts can tighten worst-case prompt coverage, while robust fine-tuning also requires controlling parameter changes that may affect capabilities inherited from the pretrained policy. Our analysis shows that broader prompt coverage tightens the robustness bound on unseen prompts, while limiting parameter displacement $|\theta - \theta_0|$ constrains changes on unseen tasks. Under additional assumptions, OPSD can induce a smaller minimum-norm update than SFT; formal statements and proofs are deferred to App. E.

Remark. *Importantly, this analysis does not explicitly model the shared parameterization of the teacher and student. Because it just bounds their expected performance gap rather than guaranteeing monotonic improvement of each policy, reducing the gap may improve the student while degrading an initially stronger teacher. We examine this empirically in Tab. 1.*

4.2 Discrepancy-Weighted Multi-Task OPSD

In multi-task training, student policies may approach teacher performance on some tasks while remaining weaker on others, making uniform distillation potentially inefficient or disruptive to shared parameters. We therefore extend single-task multi-student OPSD with discrepancy-dependent task weights, emphasizing larger teacher–student gaps and reducing distillation when the gap is small or teacher performance declines.

Specifically, after each rollout phase, we smooth teacher and student success rates over iterations into r_k^{T} and r_k^{S} and define $\Delta_k = r_k^{\mathrm{T}} - r_k^{\mathrm{S}}$. We first assign a weight according to this gap:

$$g(\Delta) = \begin{cases} 0, & \Delta < \tau_{\mathrm{lo}}, \\ g_{\min} + (g_{\max} - g_{\min}) \min\Big\{1, \dfrac{\Delta - \tau_{\mathrm{lo}}}{\tau_{\mathrm{hi}} - \tau_{\mathrm{lo}}}\Big\}, & \Delta \geq \tau_{\mathrm{lo}}. \end{cases} \tag{6}$$

The threshold τ_{lo}, set to one standard error of the estimated gap under the rollout budget, suppresses distillation when the measured difference may reflect sampling noise. For larger gaps, the weight increases from $g_{\min}$ at τ_{lo} to $g_{\max}$ at τ_{hi} and then saturates. This assigns more supervision to tasks whose students remain further behind their teacher.

Since a positive gap can persist even when teacher performance declines, we additionally scale the weight by the teacher's performance relative to its best smoothed success rate $r_k^{\max}$ ($b_k = 1$ while $r_k^{\max} = 0$):

$$\lambda_k^\star = g(\Delta_k)\, b_k, \qquad b_k = \mathrm{clip}\Big(\frac{r_k^{\mathrm{T}} / r_k^{\max} - \zeta}{1 - \zeta}, 0, 1\Big). \tag{7}$$

The factor b_k equals one at the teacher's best performance and decreases to zero when $r_k^{\mathrm{T}} \leq \zeta r_k^{\max}$. It reduces reliance on a deteriorating teacher and limits further distillation updates to the shared parameters. Once per iteration, the weight is updated with gradual increases and immediate decreases,

$$\lambda_k \leftarrow \min\big\{\lambda_k^\star,\ \alpha_\uparrow \lambda_k + (1 - \alpha_\uparrow)\lambda_k^\star\big\}, \tag{8}$$

with rise rate $\alpha_\uparrow \in [0, 1)$, initialized at g_{mid}. We then normalize the weights as $w_k = \lambda_k / g_{\text{mid}}$, where $g_{\text{mid}} = (g_{\min} + g_{\max})/2$, so that a task at the midpoint of the gap interval has unit weight when $b_k = 1$. The multi-task objective is

$$\mathcal{L}_{\text{D}}(\theta) = \frac{1}{|\mathcal{B}^{\text{S}}|} \sum_{k=1}^{K} w_k \sum_{z \in \mathcal{B}_k^{\text{S}}} \ell_{\text{D}}(z), \tag{9}$$

which approximately estimates $\frac{1}{K} \sum_k w_k \mathcal{L}_k(\theta)$ and controls the relative contribution of each task.

4.3 Decoupled Alternating Reinforcement Learning for Policy Improvement

Distillation moves the student policies toward the teacher policy but never raises the teacher policy itself. The teacher's success rate therefore caps what a task can reach, and environment reward is the only signal that lifts it. We cast training as improving return (ii) while keeping the cross-prompt discrepancy small, and realize this objective through alternating updates so that reward-driven improvement and cross-prompt transfer can be optimized with independently controlled dynamics.

$$\min_{\theta} \ \mathcal{L}_{\text{R}}(\theta) \quad \text{subject to} \quad \mathcal{L}_{\text{D}}(\theta) \le \epsilon_{\text{D}}, \qquad \mathcal{L}_{\text{R}}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{z \in \mathcal{B}} \ell_{\text{R}}(z), \tag{10}$$

where $\mathcal{L}_{\text{R}}$ is the PPO loss over the rollouts of both prompt conditions, where the actor term of $\ell_{\text{R}}(z)$ uses $\alpha_{\text{S}} \hat{A}_z$ for $z \in \mathcal{B}^{\text{S}}$ ($\alpha_{\text{S}} < 1$; App. B.3). The two terms play different roles: RL improves the task policy from reward, and distillation transfers that improvement across the prompts for a task.

The usual route to Eq. 10 is a Lagrangian $\mathcal{L}_{\text{R}} + \omega \mathcal{L}_{\text{D}}$ minimized with one optimizer. However, for two signals as different as reward and imitation, this ties both to one step size and one set of optimizer statistics. The larger gradient then dominates, and the balance drifts with the state of training. Thus, we instead treat the constraint by alternation at the optimization level: an RL step moves toward higher return, and a distillation step restores the constraint from the point the RL step reached as

$$\theta_{i+\frac{1}{2}} = U_{\text{R}}\big(\theta_i; \widetilde{\mathcal{B}}_i\big), \qquad \theta_{i+1} = U_{\text{D}}\big(\theta_{i+\frac{1}{2}}; \widetilde{\mathcal{B}}_i^{\text{S}}\big), \tag{11}$$

where $\widetilde{\mathcal{B}}_i \subset \mathcal{B}$ is the i-th mini-batch, $\widetilde{\mathcal{B}}_i^{\text{S}} = \widetilde{\mathcal{B}}_i \cap \mathcal{B}^{\text{S}}$, and U_{R}, U_{D} are AdamW steps on $\mathcal{L}_{\text{R}}$ and $\mathcal{L}_{\text{D}}$ with separate step sizes and optimizer states ψ_{R}, ψ_{D}. Teacher targets are regenerated at $\theta_{i+1/2}$, so each distillation step transfers the improvement of the preceding RL step, and the separate optimizer states let each signal keep its scale and momentum. Alg. 1 (App. B.2) shows one training iteration.

5 Experiments

5.1 Experimental Setup

Evaluation Benchmarks. We conducted training and evaluation in RoboLab (Yang et al., 2026) and REALM (Sedlacek et al., 2026), followed by real-robot experiments using RoboLab-trained checkpoints. In RoboLab, we trained on four tasks with 10 linguistic variations per task, while in REALM, we trained on two tasks using 10 prompts sampled across perturbation types. We evaluate robustness on both trained and unseen prompts (Sec. 5.2), unseen tasks (Sec. 5.3), language binding and visual distractors (Sec. 5.4), and different perturbation types in REALM (Sec. 5.6). Finally, we evaluate sim-to-real transfer under both in-distribution and visually perturbed real-robot settings.

Models and Baselines. We use $\pi_{0.5}$ as the target model for RL fine-tuning and compare against Cosmos3_nano and Cosmos3_edge as additional baselines. All fine-tuned models are initialized from $\pi_{0.5}$, with the SigLIP vision encoder frozen, the action expert fully fine-tuned, and the PaliGemma VLM fine-tuned with LoRA. We consider four training methods (PPO, GRPO, OPSD-S, and OPSD-M). PPO and GRPO follow their respective RL objectives, with prompts randomly sampled from the predefined multi-prompt pool during training. OPSD-S uses fixed teacher and student prompts, where the teacher corresponds to the frozen base model conditioned on a different prompt. OPSD-M extends this to multiple teacher/student prompts by randomly sampling prompts from the pool and assigning the prompt with higher rollout performance as the teacher and the other as the student.

Table 1: **Performance on trained and unseen perturbed prompts.** Success rate (%) evaluated on the 10 prompts seen during training (*Trained*) and 25 held-out perturbed prompts (*Unseen*) per task. Bold indicates the best success rate in each column.

Model	Sauce		Mustard		RaisinBox		BowlStacking		Average	
	Train	Unseen	Train	Unseen	Train	Unseen	Train	Unseen	Train	Unseen
$\pi_{0.5}$	55.1	42.9	49.4	41.0	34.6	34.0	12.5	11.0	37.9	32.2
$\text{Cosmos}_{\text{Edge}}$	67.1	**78.0**	**64.1**	**62.0**	3.8	1.0	12.5	8.0	36.9	37.3
$\text{Cosmos}_{\text{Nano}}$	71.2	73.5	33.8	33.0	12.7	16.5	23.8	27.0	35.4	37.5
PPO	63.8	49.5	48.1	40.4	41.2	40.8	18.8	4.0	43.0	33.7
GRPO	50.8	49.0	38.5	47.5	32.9	29.6	3.8	6.0	31.5	33.0
OPSD-S	59.5	58.0	56.4	50.0	41.8	43.9	16.2	13.0	43.5	41.2
OPSD-M	53.2	38.8	48.7	47.0	38.0	36.0	13.8	16.0	38.4	34.4
MOSAR (Ours)	**75.0**	**78.0**	54.4	**62.0**	**46.2**	**54.1**	**36.2**	**33.0**	**53.0**	**56.8**

Table 2: **OOD instruction generalization across instruction specificity.** Success rate (%) on three OOD tasks, separately evaluated with vague, default, and specific instructions. Each instruction condition contains 12 trials. Overall Avg. aggregates all 108 trials. Bold indicates the best result.

Model	RubiksCube OrBanana			RubiksCube BehindBowl			BowlStacking RightOnLeft			Overall Avg.
	Vague	Default	Specific	Vague	Default	Specific	Vague	Default	Specific	Avg.
$\pi_{0.5}$	**100.0**	**100.0**	91.7	41.7	41.7	16.7	16.7	8.3	25.0	49.1
$\text{Cosmos}_{\text{Edge}}$	66.7	83.3	83.3	25.0	16.7	16.7	0.0	16.7	8.3	35.2
$\text{Cosmos}_{\text{Nano}}$	83.3	83.3	75.0	16.7	0.0	16.7	16.7	33.3	8.3	37.0
PPO	75.0	91.7	75.0	58.3	58.3	8.3	8.3	8.3	0.0	42.6
GRPO	75.0	**100.0**	**100.0**	66.7	33.3	50.0	0.0	8.3	0.0	48.1
OPSD-S	75.0	**100.0**	83.3	66.7	66.7	16.7	16.7	16.7	33.3	52.8
OPSD-M	83.3	91.7	75.0	41.7	50.0	25.0	33.3	8.3	8.3	46.3
MOSAR (Ours)	**100.0**	**100.0**	**100.0**	**83.3**	**75.0**	**66.7**	**41.7**	**66.7**	**50.0**	**75.9**

5.2 RoboLab with trained and unseen prompts results

To evaluate robustness to language variations, we report performance on both trained and unseen prompts for each task in the RoboLab benchmark. The Train setting evaluates the 10 prompts used during training, while the Unseen setting evaluates 25 prompts that were not observed during training, with four trials per prompt. Tab. 1 compares the non-fine-tuned baselines ($\pi_{0.5}$, $\text{Cosmos}_{\text{Edge}}$, and $\text{Cosmos}_{\text{Nano}}$) with the fine-tuned models (PPO, GRPO, OPSD-S, and OPSD-M). Overall, the fine-tuned models show robust performance on both trained and unseen prompts across different tasks, demonstrating generalization to linguistic variations beyond those observed during training.

5.3 RoboLab with out-of-distribution task results

We further evaluate generalization to tasks not seen during training, as reported in Tab. 2. For each task, we use the original vague, default, and specific prompt settings provided by RoboLab. Our method maintains, and in many cases improves upon, the performance of the base model on unseen tasks after fine-tuning. These results suggest that our training does not degrade the model's existing task distribution, but instead improves robustness while preserving its generalization capability.

5.4 RoboLab with language binding, visual perturbation results

To further evaluate robustness, we conduct two additional experiments. First, we evaluate whether the model can follow different goals under the same task and scene configuration (Tab. 3). Our method consistently follows the corresponding instructions when the goal is changed, suggesting that the performance improvement does not result from overfitting specific scenes to particular ac-

Table 3: **Goal-swap evaluation with default instructions.** Each column denotes an original goal → swapped goal evaluated in the identical scene. We report the follow success rate (%) over 24 trials per task. Bold indicates the best result in each column.

Model	BBQ → salad	Right bin → left bin	Larger raisin → smaller butter	Left-on-right → right-on-left	Average
$\pi_{0.5}$	**37.5**	41.7	50.0	20.8	37.5
OPSD-S	16.7	45.8	**66.7**	33.3	40.6
OPSD-M	12.5	4.2	41.7	20.8	19.8
MOSAR (Ours)	**37.5**	**50.0**	**66.7**	**62.5**	**54.2**

Table 4: **Robustness to visual distractors.** Success rate (%) after adding three visual distractors to each task. Each task is evaluated over 24 trials. Bold indicates the best result in each column.

Model	Sauce	Mustard	RaisinBox	BowlStacking	Avg.
$\pi_{0.5}$	58.3	37.5	4.2	0.0	25.0
OPSD-S	62.5	**66.7**	16.7	16.7	40.6
OPSD-M	62.5	33.3	12.5	8.3	29.2
MOSAR	**75.0**	**66.7**	**20.8**	**41.7**	**51.1**

Table 5: **Real-robot evaluation.** Success rate over 20 trials for each real-robot task. Bold indicates the best result in each column.

Model	RaisinBox	Sauce
$\pi_{0.5}$	45.0	30.0
OPSD-S	55.0	30.0
OPSD-M	50.0	30.0
MOSAR	**70.0**	**50.0**

tions, but from improved language-action binding. We further evaluate visual robustness by introducing three different distractor objects into the same scene (Tab. 4). Our method achieves consistently higher performance across different distractor objects, indicating an improved ability to identify the target object and execute the instructed task in visually cluttered scenes.

5.5 Real robot experiments

We further evaluate the RoboLab-trained models on a real Franka robot, with results reported in Tab. 5. We consider two tasks: for the raisin-box task, we introduce a cube as an additional visual distractor, while the sauce-bottle task follows the original simulation setup. For each task, we evaluate five different prompts across four object configurations and report the success rate. Qualitative examples are shown in Fig. 3. In the BBQ-sauce example, our method correctly identifies and grasps the instructed bottle, whereas the baselines tend to grasp the bottles placed closest to the robot. These results indicate that the improved language grounding transfers from simulation to the real robot, including under visual variations.

5.6 REALM benchmark

We further evaluate our method on REALM, a different simulation benchmark. REALM provides five types of prompt perturbations for each task, with 10 prompts per type. During training, we

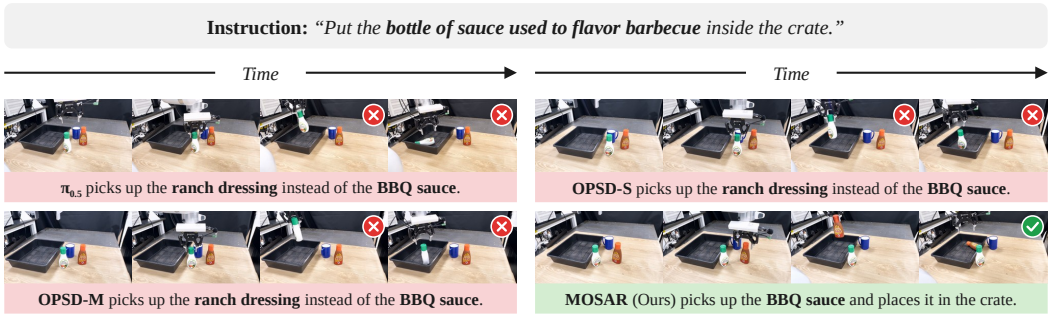


Figure 3: **Real-robot evaluation.** Given the same instruction and scene, $\pi_{0.5}$, OPSD-S, and OPSD-M incorrectly grasp the ranch dressing, whereas MOSAR (ours) correctly identifies the BBQ sauce and places it in the crate.

Table 6: **REALM evaluation by instruction type.** Success rate (%) on the REALM benchmark. Each type-wise result aggregates five prompts per task across both tasks (40 trials). Bold indicates the best result in each column.

	Model	S-PROP	S-LANG	S-MO	S-AFF	S-INT	Average
Trained	$\pi_{0.5}$	68.8	31.2	**100.0**	18.8	25.0	48.8
	OPSD-M	56.2	**37.5**	81.2	**50.0**	**43.8**	**53.8**
	OPSD-S	68.8	25.0	93.8	37.5	25.0	50.0
	MOSAR	**81.2**	**37.5**	68.8	37.5	**43.8**	**53.8**
Unseen	$\pi_{0.5}$	35.0	82.5	72.5	30.0	32.5	50.5
	OPSD-M	47.5	75.0	75.0	37.5	42.5	55.5
	OPSD-S	37.5	80.0	**77.5**	37.5	45.0	55.5
	MOSAR	**50.0**	**87.5**	72.5	**47.5**	**47.5**	**61.0**

Table 7: **Ablation study of our method.** Success rate (%) on unseen prompts and out-of-distribution (OOD) instructions. *w/o Sep. opt.* uses a shared optimizer for all components, *w/o* w removes the weighting term w applied to the OPSD ; and *w/o PPO* removes the PPO objective.

	Unseen Prompts					OOD Instructions			
Method	**Sauce**	**Mustard**	**RaisinBox**	**Bowl**	**Avg.**	**Vague**	**Default**	**Specific**	**Avg.**
MOSAR (full)	**78.0**	62.0	54.1	**33.0**	**56.8**	**75.0**	**80.6**	**72.2**	**75.9**
w/o Sep. opt.	55.6	42.9	56.1	22.0	44.2	66.6	75.0	63.9	68.5
w/o w	73.0	**68.0**	**57.1**	20.0	54.5	44.5	61.1	61.2	55.6
w/o PPO	73.7	50.5	30.6	6.0	40.2	50.0	38.9	41.7	43.5
$\pi_{0.5}$	42.9	41.0	34.0	11.0	32.2	52.8	50.0	44.5	49.1

randomly select two prompts from each type, and evaluate on five unseen prompts per type. The results are reported in Tab. 6. Overall, our method achieves strong performance across most prompt perturbation types, demonstrating robustness to diverse forms of language variation.

5.7 Ablations

We conduct ablation studies in Tab. 7 on unseen prompts and OOD instructions to evaluate robustness. *w/o Sep. opt.* uses a shared optimizer for PPO and OPSD instead of separate optimization, leading to competing gradient objectives and weaker performance on unseen prompts. *w/o* w removes the discrepancy-dependent task weighting for OPSD and applies distillation uniformly across tasks. While it maintains ID performance, the improvement on OOD tasks is limited, highlighting the benefit of adjusting distillation according to teacher–student performance gaps. Finally, *w/o PPO* performs OPSD alone and falls below OPSD-S on unseen prompts (40.2 vs. 41.2): its self-teacher, unlike the frozen teacher of OPSD-S, drifts without RL, which is why lifting the ceiling (ii) is a prerequisite for closing the gap (i). Overall, these results show that separate optimization, task-dependent weighting, and PPO provide complementary benefits for robustness and generalization.

6 Conclusion

We introduced MOSAR, a framework for improving robustness in vision-language-action models through multi-student on-policy self-distillation and alternating reinforcement learning. MOSAR transfers behavior from a high-performing teacher prompt to multiple semantically equivalent student prompts at student-visited observations (i), while discrepancy-aware task weighting focuses distillation on prompts and tasks with the greatest performance gaps (iii). By repeatedly refreshing teacher targets after RL updates, this alternating procedure enables performance improvement (ii) and cross-prompt knowledge transfer to reinforce each other throughout training. Experiments on RoboLab, REALM, and a real Franka robot demonstrate that MOSAR consistently improves robustness to unseen instructions while preserving strong generalization to unseen tasks. In RoboLab,

MOSAR achieves 56.8% average success on unseen prompts and 75.9% on OOD tasks, outperforming RL and OPSD baselines; on REALM, it achieves 61.0% average success on unseen prompt perturbations. The results under goal swapping, visual distractors, and direct sim-to-real deployment further suggest that MOSAR improves language grounding rather than merely memorizing prompt–action associations. Overall, MOSAR provides an effective post-training approach for building VLA policies that are more reliable under the linguistic variability encountered in real robotic use.

AI USE STATEMENT

Generative AI tools were employed to assist with programming, mathematical verification, experiment management, and manuscript editing. These tools did not contribute to the conception or development of the research ideas and scientific contributions presented in this work. Any AI-assisted content, including code and text, was thoroughly examined and verified by the authors. The authors assume full responsibility for the correctness, integrity, and final presentation of the work, including all claims, code, and materials produced with AI assistance.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes. In *ICLR*, 2024.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. In *CoRL*, 2025a.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. In *RSS*, Los Angeles, CA, USA, June 2025b.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S Ryoo, Grecia Salazar, Pannag R Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan H Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics Transformer for Real-World Control at Scale. In *RSS*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.025.

Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Xiang Li, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, Tiejun Huang, Yu Wang, and Chao Yu. π_{RL}: Online RL Fine-tuning for Flow-based Vision-Language-Action Models. arXiv preprint arXiv:2510.25889, 2025.

Xuan Dong, Zhe Han, Tianhao Niu, Qingfu Zhu, and Wanxiang Che. When does language matter? multilingual instructions reveal step-wise language sensitivity in vision-language-action models. In *ACL*, 2026.

Jiaheng Hu, Rose Hendrix, Ali Farhadi, Aniruddha Kembhavi, Roberto Martin-Martin, Peter Stone, Kuo-Hao Zeng, and Kiana Ehsani. FLaRe: Achieving Masterful and Adaptive Robot Policies with Large-Scale Reinforcement Learning Fine-Tuning. In *ICRA*, 2025.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation. In *ICML*, 2026.

Chanyoung Kim, Minwoo Kim, Minseok Kang, Hyunwoo Kim, and Dahuin Jung. LIBERO-Para: A Diagnostic Benchmark and Metrics for Paraphrase Robustness in VLA Models. arXiv preprint arXiv:2603.28301, 2026a.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An Open-Source Vision-Language-Action Model. In *CoRL*, volume 270, 2025.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. In *ICLR*, 2026b.

Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, Dehui Wang, Dingxiang Luo, Yuchen Fan, Youbang Sun, Jia Zeng, Jiangmiao Pang, Shanghang Zhang, Yu Wang, Yao Mu, Bowen Zhou, and Ning Ding. SimpleVLA-RL: Scaling VLA Training via Reinforcement Learning. ICLR, 2026a.

Quanhao Li, Junqiu Yu, Kaixun Jiang, Yujie Wei, Zhen Xing, Pandeng Li, Ruihang Chu, Shiwei Zhang, Yu Liu, and Zuxuan Wu. Diffusionopd: A unified perspective of on-policy distillation in diffusion models. *arXiv preprint arXiv:2605.15055*, 2026b.

Yijiang Li, Bingyang Wang, Yijun Liang, Yunjie Tian, Di Fu, and Nuno Vasconcelos. On-Policy Self-Distillation without Any Supervision. arXiv preprint arXiv:2608.06296, 2026c.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *NeurIPS*, 38:40783–40818, 2025a.

Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, Yi Wu, Chao Yu, and Yu Wang. What Can RL Bring to VLA Generalization? An Empirical Study. In *NeurIPS*. Curran Associates, Inc., 2025b.

Yifeng Liu, Shiyuan Zhang, Yifan Zhang, and Quanquan Gu. Self-Distilled Policy Gradient. arXiv preprint arXiv:2606.04036, 2026. URL https://arxiv.org/abs/2606.04036.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Self-Distilled Agentic Reinforcement Learning. arXiv preprint arXiv:2605.15155, 2026. URL https://arxiv.org/abs/2605.15155.

NVIDIA et al. Cosmos 3: Omnimodal World Models for Physical AI. arXiv preprint arXiv:2606.02800, 2026. URL https://arxiv.org/abs/2606.02800.

Leyi Pan, Shuchang Tao, Yunpeng Zhai, Lingzhe Zhang, Zhaoyang Liu, Bolin Ding, Aiwei Liu, and Lijie Wen. RLCSD: Reinforcement Learning with Contrastive On-Policy Self-Distillation. arXiv preprint arXiv:2606.11709, 2026. URL https://arxiv.org/abs/2606.11709.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *AISTATS*, volume 15. PMLR, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. arXiv preprint arXiv:1707.06347, 2017. URL https://arxiv.org/abs/1707.06347.

Martin Sedlacek, Pavlo Yefanov, Georgy Ponimatkin, Jai Bardhan, Simon Pilc, Mederic Fourmy, Evangelos Kazakos, Cees GM Snoek, Josef Sivic, and Vladimir Petrik. Realm: A real-to-sim validated benchmark for generalization in robotic manipulation. *IEEE RA-L*, 2026.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv preprint arXiv:2402.03300, 2024. URL https://arxiv.org/abs/2402.03300.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. In *ICML*, 2026.

Qitai Tan, Zefang Zong, Mo Li, Yipeng Shi, Yang Li, and Peng Chen. ATOD: Annealed Turn-Aware On-Policy Distillation for Multi-Turn Agentic Tasks. arXiv preprint arXiv:2606.27814, 2026. URL https://arxiv.org/abs/2606.27814.

Kejing Wang, Toan Nguyen, Minh Hoang Nguyen, Simon Khan, and Flora D. Salim. ROAD-VLA: Robust Online Adaptation via Self-Distillation for Vision-Language-Action Models. arXiv preprint arXiv:2606.25800, 2026a. URL https://arxiv.org/abs/2606.25800.

Siting Wang, Xiaofeng Wang, Zheng Zhu, Minnan Pei, Xinyu Cui, Cheng Deng, Jian Zhao, Guan Huang, Haifeng Zhang, and Jun Wang. π-StepNFT: Wider Space Needs Finer Steps in Online RL for Flow-based VLAs. arXiv preprint arXiv:2603.02083, 2026b. URL https://arxiv.org/abs/2603.02083.

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. In *RSS*, Sydney, Australia, July 2026. doi: 10.15607/RSS.2026.XXII.096.

Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu, Shaojie Yang, Xipeng Wu, Qingwei Guo, et al. Mastering complex control in moba games with deep reinforcement learning. In *AAAI*, volume 34, 2020.

Zhaokai Yin and Zhipeng Zhang. Grounded Semantic Re-Binding for Robust Instruction Generalization in Vision-Language-Action Models. arXiv preprint arXiv:2608.02497, 2026. URL https://arxiv.org/abs/2608.02497.

Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, Si Xu, Hongzhi Zang, Quanlu Zhang, Yongji Wu, Chunyang Zhu, Junhao Hu, Zixiao Huang, Mingjie Wei, Yuqing Xie, Ke Yang, Bo Dai, Zhexuan Xu, Jiakun Du, Xiangyuan Wang, Xu Fu, Letong Shi, Zhihao Liu, Kang Chen, Weilin Liu, Gang Liu, Boxun Li, Jianlei Yang, Zhi Yang, Guohao Dai, and Yu Wang. RLinf: Flexible and efficient Large-Scale reinforcement learning via Macro-to-Micro flow transformation. In *OSDI 26*, pp. 829–846, Seattle, WA, July 2026. USENIX Association. ISBN 978-1-939133-55-7. URL https://www.usenix.org/conference/osdi26/presentation/yu-chao.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. In *ICML*, 2026.

Zhide Zhong, Haodong Yan, Junfeng Li, Junjie He, Tianran Zhang, and Haoang Li. VLA-OPD: Bridging Offline SFT and Online RL for Vision-Language-Action Models via On-Policy Distillation. arXiv preprint arXiv:2603.26666, 2026.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *CoRL*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

A EXPERIMENTAL DETAILS

This section gives the training and evaluation settings that the main text leaves out: the common training setup, the schedule of MOSAR, the baselines, the evaluation protocols with their trial counts, the prompt sets and the real-robot setup.

A.1 COMMON TRAINING SETUP

All RoboLab runs start from the public $\pi_{0.5}$ DROID joint-position checkpoint (Black et al., 2025a) and use the RLinf framework (Yu et al., 2026). The simulator runs physics at 60 Hz and the policy at 15 Hz, so one action chunk of $H = 15$ steps spans one second. The policy receives an external and a wrist camera image rendered at 360×640, the seven joint positions, the gripper position and the instruction, and outputs $D = 8$ targets (seven joints and the gripper).

Each iteration runs 450 control steps (30 chunks) in each of 128 parallel environments, 32 per task; an episode ends at success or at the task's time limit, and the reward is $+1$ once, when the task succeeds. Evaluation episodes use the same time limits, except that SauceBottlesCrate runs for 600 steps instead of 450. Every environment starts from its own layout, with each object displaced by up to 7 cm from the task's default. Each reset adds a further displacement of up to 7 cm and a yaw of up to 0.5 rad. This rule and its training seed are shared by all methods, and evaluation uses the same rule.

The rollout batch of 3,840 chunks is split into two mini-batches of 1,920 and reused for 16 epochs. Every method thus takes 32 optimizer steps per iteration, independent of the number of GPUs. The MOSAR run used four NVIDIA B300 GPUs and took 1,927 s per iteration on average, of which 344 s were rollout. The checkpoint used in Sec. 5 was reached after about 8.2 hours.

A.2 ROLLOUTS AND THE GATE

For each task, half of the environments run the teacher prompt and the other half the M student prompts, each environment collecting one episode of fixed length per iteration. The teacher prompt of a task is its built-in RoboLab *specific* instruction, which is not one of the generated prompts (Tabs. A2 and A4); the three student prompts of an iteration are drawn from the ten training prompts of the task and split over 6, 5 and 5 environments. All three were present in every iteration of the main run.

An episode counts as a success when it receives the sparse task reward; chunks after the success remain in the batch. Success rates are measured on these rollouts, which use the hybrid sampler, while the teacher target and evaluation use the deterministic sampler. Both rates are smoothed by an exponential moving average over iterations with weight α_{ema} on the previous value, initialized with the first measurement. The anchor $r_k^{\max}$ is the running maximum of the smoothed teacher rate, and $b_k = 1$ while $r_k^{\max} = 0$. Before the first measurement every λ_k equals λ_{mid}; an increase of λ_k moves a fraction $1 - \alpha_{\uparrow}$ of the way per iteration, a decrease is immediate, and the weights are held fixed while the current batch is optimized. With 16 episodes per condition and task per iteration and the smoothing, the standard error of Δ_k at a success rate of one half is 0.10, which sets τ_{lo}; τ_{hi} is three times that.

A.3 OPTIMIZERS AND UPDATE SCHEDULE

Both update operators of Eq. 11 are AdamW steps with separate moment estimates, decoupled weight decay and gradient clipping to a maximum norm, applied by each step. One iteration reuses its rollouts for E epochs of mini-batches drawn uniformly at random; each mini-batch step is one RL step followed, when some $w_k > 0$, by the regeneration of the teacher targets with fresh x_1 and one distillation step. On a mini-batch, the gradient of Eq. 9 is over its student samples with $w_{k(z)} > 0$. Because the distillation state normalizes its own gradient, a factor common to all w_k leaves the distillation step unchanged. The weights set the ratio between the tasks distilled in the same iteration and switch a task off at zero, while η_{D} sets the strength of distillation.

The RL step size is $\eta_{\mathrm{R}} = s\,\eta_0$ with a fixed base rate η_0 and a scale s set by a controller. Let f be the fraction of samples whose ratio ρ_θ fell outside the clipping range, averaged over the RL steps

of the iteration and over both conditions. At the end of the iteration, $s \leftarrow s\sqrt{f^\star/f}$ with target $f^\star$, limited to a factor of 1.25 per iteration and to $[0.15, 2.0]$, and applied from the next iteration. The value head has its own rate, scaled by s as well, and receives gradient from the RL step only. The distillation step size is $\eta_{\mathrm{D}} = 0.55\, c_i\, \eta_0$, where c_i is a cosine factor over 200 scheduled iterations that the controller does not touch; over the 20 iterations of the main run c_i stays above 0.98.

A.4 Trained parameters, cost and constants

We fine-tune the action expert fully and the VLM language model with LoRA adapters; the vision encoder is frozen, both optimizer states cover the same parameters, and the value loss backpropagates into the adapters. Each distillation step recomputes the teacher target for every sample, one VLM forward under the teacher prompt and N evaluations of the action expert, with nothing cached across steps. The prompt sets are hand-built, and the gate reads one pooled rate over the student prompts, which change every iteration. Tab. A1 lists the constants.

Table A1: **Constants of the method.** Values used in our experiments; symbols as in Secs. 3–4 and App. A.

Component	Symbol	Value	Note
Denoising steps / stochastic step / noise level	N / J / ν	10 / 3 / 0.5	Eqs. A1, A2
Distillation time distribution	t	$\mathrm{Beta}(1.5, 1)$	Eq. 2
Tasks / environments per task	K	4 / 32	teacher 16, students 6+5+5
Student prompt set size / student prompts per iteration	$\lvert\mathcal{S}_k\rvert$ / M	10 / 3	drawn uniformly
Gap thresholds / brake	τ_{lo}, τ_{hi} / ζ	0.10, 0.30 / 0.75	Eq. 7
Gap-term range	$g_{\min}$, $g_{\max}$	1, 10	$g_{\mathrm{mid}} = 5.5$
Success-rate smoothing / weight rise	α_{ema} / $\alpha_\uparrow$	0.5 / 0.5	per iteration; weight on the previous value
Action chunk / episode length	H, D / L	15 / 8; 30 chunks	7 joints + gripper
PPO clipping / dual clip / value clip	ϵ_{c} / c_{d} / –	0.2 / 3.0 / 0.2	Eq. A4
Discount / GAE	γ / λ_{GAE}	0.99 / 0.95	advantages normalized over $\mathcal{B}$
Student-condition advantage scale	α_{S}	0.1	PPO weight
Epochs / mini-batch size	E / $\lvert\widetilde{\mathcal{B}}\rvert$	16 / 1,920	$\lvert\mathcal{B}\rvert = 3{,}840$: 128 episodes of 30 chunks
Base step size	η_0	2.5×10^{-6}	fixed for the RL state
RL step-size controller	s	$s \leftarrow s\sqrt{f^\star/f}$, $f^\star = 0.08$	factor ≤ 1.25 per iteration; $s \in [0.15, 2.0]$
Distillation step size	η_{D}	$0.55\, c_i\, \eta_0$	c_i: cosine to 0.1 over 200 scheduled iterations
Value-head step size / value-loss weight	–	10^{-4} / 1	RL step only; scaled by s
AdamW / gradient clipping	(β_1, β_2), ε / –	$(0.9, 0.95)$, 10^{-8} / 1.0	decoupled weight decay 0.01; both states, each step; maximum norm
Trained parameters	–	expert full; VLM LoRA rank 32	scale 1, attention and MLP projections; fp32; vision encoder frozen

A.5 Baselines

All trained baselines start from the same $\pi_{0.5}$ checkpoint and update the same parameters as MOSAR. They collect 128 episodes per iteration and take 32 optimizer steps per iteration, with one AdamW state and a cosine learning rate starting from 5×10^{-6}. They are trained for 20 iterations and evaluated at iteration 20.

PPO. PPO (Schulman et al., 2017) uses the denoising MDP and the loss ℓ_{R} of App. B.3 with a value head and GAE. In each iteration, every environment of a task receives the same prompt, drawn at random from the task's ten training prompts.

GRPO. GRPO (Shao et al., 2024) differs from PPO only in the advantage: the episode return is standardized within groups of eight adjacent environments that share a layout, and no value head is trained. The prompts, model, rollout budget and optimizer are those of PPO.

OPSD-S. OPSD-S trains with the distillation loss of Eq. 2 alone. Its teacher is the frozen initial policy conditioned on the task's specific instruction, and its student is the current policy conditioned on the task's vague instruction; the pair is fixed for the whole run.

OPSD-M. OPSD-M uses the loss and optimizer of OPSD-S, but its prompt pair changes every iteration. In each iteration and task it draws one perturbation type, runs the two training prompts of that type on half of the environments each, and makes the prompt with the higher rollout success the teacher and the other the student.

Cosmos$_{\text{Edge}}$ and Cosmos$_{\text{Nano}}$. These are the public Cosmos 3 DROID policies (NVIDIA et al., 2026), evaluated without training. They receive one composite image (the wrist view with the two external views attached, 540×640), the joint positions and the instruction, sample with four UniPC steps, and execute their 32-step chunk open loop. Cells, seeds, layouts, prompts and success criteria are those of the other policies.

A.6 Evaluation protocols

All RoboLab evaluations use the deterministic sampler, the observation and action settings of training and the layout-perturbation rule of App. A.1. Episodes whose policy server did not respond are removed from the denominator, which is why a few cells have 98 or 99 trials instead of 100.

Training and unseen prompts (Tab. 1). The training-prompt protocol evaluates the ten training prompts of each task, each twice per seed, with seeds 7 to 10: eight trials and eight layouts per prompt and 320 trials per policy. The unseen-prompt protocol evaluates 25 held-out prompts per task, five per perturbation type, once per seed with seeds 7 to 10: four trials per prompt and 400 trials per policy. For $\pi_{0.5}$ and OPSD-M, the unseen numbers are the same 100 prompts taken from a larger evaluation of 360 held-out prompts with the same seeds; that evaluation assigns prompts to environments, and hence to layouts, differently. For the policies measured both ways, the two estimates differed by up to 6.3 points.

Unseen tasks (Tab. 2). Each unseen task is evaluated with its vague, default and specific RoboLab instruction, 12 trials per instruction with layout seed 7. We evaluated three unseen tasks; Tab. 2.

Goal swap (Tab. 3). For each training task we built a mirror task with the same scene and objects and the scorer of the swapped goal (SauceBottlesCrate $\rightarrow$ SaladDressingCrate, MustardInRightBin $\rightarrow$ MustardInLeftBin, LargerObjectRaisinBoxInBin $\rightarrow$ SmallerObjectButterInBin, BowlStackingLeftOnRight $\rightarrow$ BowlStackingRightOnLeft). The policy receives the mirror task's default instruction and is scored on the swapped goal, 24 trials per task with layout seed 7. The two bowl-stacking scorers are not mutually exclusive: in the vague-instruction arm, 8 of 24 MOSAR episodes satisfied both, so a success on this pair does not show that the swapped instruction was followed.

Visual distractors (Tab. 4). The four training scenes receive the same three extra objects (a blue block, a spoon and a can of mushrooms), placed in free space away from the target objects and paths; instruction, goal and scorer are unchanged. Each cell has 24 trials with the default instruction and layout seed 7.

REALM (Tab. 6). REALM (Sedlacek et al., 2026) provides, per task, ten prompts for each of five perturbation types. On REALM, MOSAR uses the task's default instruction as the teacher prompt and two student prompts per iteration, with eight teacher and eight student environments per task;

with eight episodes per condition, the gap thresholds are $\tau_{\mathrm{lo}} = 0.15$ and $\tau_{\mathrm{hi}} = 0.45$. MOSAR and OPSD-M train on the same two prompts per type, ten per task, and OPSD-S on one fixed pair per task. All use 32 environments and 32 optimizer steps per iteration and are evaluated at iteration 15. Each prompt is evaluated for four episodes of at most 500 steps, with an open-loop horizon of 8 steps; an episode succeeds when the task progress reaches one. The training-prompt rows of Tab. 6 therefore rest on 16 episodes per type (two prompts, two tasks, four episodes), and the unseen rows on 40 (five prompts, two tasks, four episodes).

Statistics. All policies share prompts, seeds and layouts, so we compare them at the prompt level. For two policies, we take the per-prompt difference in success rate and report its mean, a paired t statistic over prompts, and the number of prompts won and lost. Treating the 400 unseen trials of a policy as independent would understate the uncertainty, because trials of one prompt share its difficulty. Tables in the main text report single evaluation runs.

A.7 PROMPT SETS

For RoboLab, we generated 20 prompts for each perturbation type of each training task, 400 in total. The five types follow REALM: physical properties (S-PROP), paraphrase (S-LANG), spatial relations to other objects (S-MO), function or purpose (S-AFF) and category or content (S-INT). Ten prompts per task, two per type, form the training pool of the methods that train on generated prompts (Tabs. A2 and A3). The 25 unseen prompts per task, five per type, were drawn once at random from the 90 remaining prompts before the MOSAR run (Tabs. A5–A8).

Tab. A4 lists the built-in RoboLab instructions, which serve as teacher prompts, as instructions of the unseen tasks and in the goal swap. The REALM prompts are in Tabs. A9–A11.

A.8 REAL ROBOT EXPERIMENT.

We conduct real-robot experiments using a 7-DoF Franka Research 3 equipped with a Robotiq parallel gripper. The setup includes two external Stereolabs ZED 2i cameras for global views and a wrist-mounted ZED-M camera. The robot-side PC is equipped with an NVIDIA RTX 3090 Ti, while policy inference is performed on an NVIDIA A6000 GPU. An on-site operator is present throughout the experiments for emergency stops. For $\pi_{0.5}$, we use one external camera and one wrist camera, together with joint positions, gripper position, and the language instruction as policy inputs. Each image is resized to 224×224 pixels. The policy outputs an 8-dimensional action consisting of a 7-dimensional joint action and a 1-dimensional gripper-position command.

Table A2: **RoboLab training prompts (SauceBottlesCrate, MustardInRightBin).** OPSD-S is trained on the *Teacher*/*Student* pair. OPSD-M, GRPO and PPO is trained on the remaining ten prompts, randomly sampled two per perturbation type. MOSAR (Ours) uses both the *Teacher* prompt and the same ten prompts.

Type	Prompt
	SauceBottlesCrate
Teacher	Pick up the red BBQ sauce bottle and place it inside the wooden crate
Student	Put the red bottle away
S-PROP	Place the bottle with the red body inside the open purple crate.
S-PROP	Select the red bottle with the rounded shoulders and place it inside the purple crate with the white label.
S-LANG	Move the red BBQ sauce bottle into the open-topped purple crate.
S-LANG	Deposit the red BBQ sauce bottle inside the rectangular purple container.
S-MO	Pick up the red bottle beside the left side of the crate in the image and place it inside the crate to the right of the red bottle in the initial image.
S-MO	Locate the red bottle behind the mug as viewed in the image and transfer it inside the container with the purple walls.
S-AFF	Put the red BBQ sauce bottle away inside the rectangular purple container.
S-AFF	Select the red bottle of sauce for seasoning barbecued food and place it inside the crate with the white label.
S-INT	Place the red bottle containing the condiment known as BBQ sauce inside the container with the purple walls.
S-INT	Lift the red bottle containing a condiment of the barbecue-sauce variety, then set it down inside the open purple crate.
	MustardInRightBin
Teacher	Pick up the mustard bottle and place it in the bin on the right side
Student	Use the right bin for mustard
S-PROP	Pick up the yellow bottle with the pointed nozzle and place it inside the right-side bin.
S-PROP	Move the yellow bottle from its initial position and place it inside the bin on the right-hand side.
S-LANG	Place the mustard bottle inside the container on the right.
S-LANG	Place the mustard bottle in the interior of the right-hand open container.
S-MO	Find the bottle between the two bins and put it inside the open bin on the right.
S-MO	Select the bottle in the gap between the bins and move it so that it rests inside the grey bin on the right.
S-AFF	Store the mustard bottle by placing it inside the bin on the right-hand side.
S-AFF	Move the mustard bottle used for dressing hot dogs inside the bin on the right.
S-INT	Place the bottle containing the seed-based condiment mustard inside the right-hand rectangular bin.
S-INT	Select the bottle holding a condiment derived from mustard seeds and place it inside the right-hand bin.

Table A3: **RoboLab training prompts (LargerObjectRaisinBoxInBin, BowlStackingLeftOnRight).** OPSD-S is trained on the *Teacher*/*Student* pair. OPSD-M, GRPO and PPO is trained on the remaining ten prompts, randomly sampled two per perturbation type. MOSAR (Ours) uses both the *Teacher* prompt and the same ten prompts.

Type	Prompt
	LargerObjectRaisinBoxInBin
Teacher	Compare the objects and put the larger raisin box in the grey bin
Student	Put the larger object away
S-PROP	Select the raisin box that is larger than the other package and place it inside the grey receptacle.
S-PROP	Take hold of the larger rectangular raisin box and set it down inside the grey bin on the table.
S-LANG	Move the raisin box into the grey storage bin.
S-LANG	Set the raisin box inside the grey bin with the rounded rim and release it.
S-MO	Find the larger raisin box to the right of the smaller package in the image and put it inside the open-topped grey container.
S-MO	Select the larger raisin box to the right of the smaller package in the image, then set it down inside the bin with the grey sides.
S-AFF	Move the raisin box inside the bin with the rounded corners to put it away.
S-AFF	Pick up the raisin box holding fruit to eat as a snack and place it inside the grey container.
S-INT	Pick up the larger box containing grapes preserved by drying and place it inside the grey bin with the open top.
S-INT	Please pick up the larger box containing raisins from the dried-fruit category and put it inside the open-topped grey container.
	BowlStackingLeftOnRight
Teacher	Pick up the bowl to the left of the robot and place it on top of the bowl to the right of the robot
Student	Stack the left bowl on the right
S-PROP	Put the red bowl on the left on top of the right bowl.
S-PROP	Move the red bowl on the left from its initial position and place it on top of the red bowl to the right of the robot.
S-LANG	Put the left bowl atop the right-side bowl.
S-LANG	Please stack the left bowl on top of the bowl on the right-hand side.
S-MO	Lift the bowl on the robot's left and lower it onto the bowl on the robot's right.
S-MO	Please pick up the bowl whose initial position is left of the other bowl and put it on top of the right bowl.
S-AFF	Make a stack for storage by putting the left bowl on top of the right bowl with the red interior.
S-AFF	Take the left bowl used for holding a portion of food and put it on top of the red bowl on the right.
S-INT	Take the left container classified as a bowl and put it on top of the bowl initially on the robot's right.
S-INT	Take hold of the left item of dishware that is a bowl and set it down on top of the right-side bowl.

Table A4: **RoboLab evaluation prompts for three levels of explicitness.**

Level	Instruction
TRAINING TASKS	
SauceBottlesCrate	
Vague	Put the red bottle away
Default	Put the red bbq sauce bottle in the crate
Specific	Pick up the red BBQ sauce bottle and place it inside the wooden crate
MustardInRightBin	
Vague	Use the right bin for mustard
Default	Put the mustard in the right bin
Specific	Pick up the mustard bottle and place it in the bin on the right side
LargerObjectRaisinBoxInBin	
Vague	Put the larger object away
Default	Place the larger object in the grey bin.
Specific	Compare the objects and put the larger raisin box in the grey bin
BowlStackingLeftOnRight	
Vague	Stack the left bowl on the right
Default	Stack the left bowl on the right bowl
Specific	Pick up the bowl to the left of the robot and place it on top of the bowl to the right of the robot
OOD TASKS	
Level	Instruction
RubiksCubeOrBanana	
Vague	Choose an object and put it in the bowl
Default	Put the cube or the banana in the bowl
Specific	Choose either the rubiks cube or the yellow banana and place it inside the bowl
RubiksCubeBehindBowl	
Vague	Put the cube behind the bowl
Default	Put the rubiks cube behind the bowl
Specific	Pick up the rubiks cube and place it on the table directly behind the bowl, further from you
BowlStackingRightOnLeft	
Vague	Stack the right bowl on the left
Default	Stack the right bowl on the left bowl
Specific	Pick up the bowl to the right of the robot and place it on top of the bowl to the left of the robot
GOAL-SWAP MIRROR TASKS	
Level	Instruction
SaladDressingCrate	
Vague	Put the salad dressing away
Default	Put the salad dressing bottle in the crate
Specific	Pick up the salad dressing bottle and place it inside the wooden crate
MustardInLeftBin	
Vague	Use the left bin for mustard
Default	Put the mustard in the left bin
Specific	Pick up the mustard bottle and place it in the bin on the left side
SmallerObjectButterInBin	
Vague	Put the smaller object away
Default	Place the smaller object in the grey bin.
Specific	Compare the objects and put the smaller butter in the grey bin
BowlStackingRightOnLeft	
Vague	Stack the right bowl on the left
Default	Stack the right bowl on the left bowl
Specific	Pick up the bowl to the right of the robot and place it on top of the bowl to the left of the robot

Table A5: **RoboLab unseen evaluation prompts: SauceBottlesCrate.**

Type	Prompt
S-PROP	Move the red sauce bottle inside the purple storage crate.
S-PROP	Please place the red condiment bottle inside the purple rectangular crate.
S-PROP	Lift the bottle with a red body and an orange cap, then set it down inside the crate to the right of the red bottle in the initial image.
S-PROP	Please pick up the bottle with the red exterior and put it inside the crate with the open top.
S-PROP	Move the red condiment bottle so that it rests inside the purple crate beside the red bottle in the initial image.
S-LANG	Place the red BBQ sauce bottle inside the purple container.
S-LANG	Transfer the red BBQ sauce bottle into the purple rectangular crate.
S-LANG	Pick up the red BBQ sauce bottle and put it in the purple crate with the white label.
S-LANG	Set the red BBQ sauce bottle inside the purple container with the white label and release it.
S-LANG	Please pick up the red BBQ sauce bottle and set it down in the purple crate.
S-MO	Identify the red bottle behind the mug as viewed in the image and move it inside the purple crate with the white label.
S-MO	Take the red bottle between the green-capped bottle and the crate in the image and put it inside the open purple container.
S-MO	Identify the red bottle to the right of the green-capped bottle in the image, then put it inside the purple container with the white label.
S-MO	Please pick up the red bottle between the green-capped bottle and the crate in the image and put it inside the purple crate.
S-MO	Identify the red bottle beside the left side of the crate in the image; place it inside the purple container and let go of it.
S-AFF	Put the red bottle of sauce used to flavor barbecue inside the crate.
S-AFF	Move the red bottle containing sauce for grilled food inside the open purple crate.
S-AFF	Pick up the red bottle of sauce for seasoning barbecued food and place it inside the purple storage crate.
S-AFF	Take the red bottle of sauce used as a barbecue condiment and put it inside the crate with the purple sides.
S-AFF	Please place the red bottle used to dispense BBQ sauce inside the open-topped purple crate.
S-INT	Please place the red bottle containing the condiment known as BBQ sauce inside the open crate on the right side of the image.
S-INT	Locate the red bottle of sauce in the barbecue category, then put it inside the crate.
S-INT	Take hold of the red bottle containing barbecue sauce and set it down inside the open-topped purple crate.
S-INT	Transfer the red bottle of sauce in the barbecue category inside the crate to the right of the red bottle in the initial image.
S-INT	Finish by placing the red bottle containing a condiment of the barbecue-sauce variety inside the rectangular purple container.

Table A6: **RoboLab unseen evaluation prompts: MustardInRightBin.**

Type	Prompt
S-PROP	Locate the bottle with a yellow body and a yellow nozzle, then put it inside the rectangular bin on the right.
S-PROP	Put the bottle with the yellow body inside the right-hand container, please.
S-PROP	Take hold of the yellow bottle with the pointed nozzle and set it down inside the right-hand rectangular bin.
S-PROP	Please pick up the bottle with the yellow exterior and put it inside the grey bin at the right-hand position.
S-PROP	Place the yellow bottle with the rounded shoulders inside the right-hand storage bin and let go of it.
S-LANG	Take the mustard bottle and place it inside the right bin.
S-LANG	Move the mustard bottle so that it rests inside the bin occupying the right-hand position.
S-LANG	Set the mustard bottle inside the grey receptacle on the right and release it.
S-LANG	Please pick up the mustard bottle and set it down in the right-hand bin.
S-LANG	Transfer the mustard bottle from its starting position into the bin on the right.
S-MO	Identify the bottle flanked by the two bins and move it inside the grey container on the right.
S-MO	Select the bottle between the two bins, then set it down inside the right-hand container.
S-MO	Please find the bottle in the gap between the bins and place it inside the open grey bin on the right.
S-MO	Choose the bottle positioned between the left bin and the right bin and put it inside the grey bin at the right-hand position.
S-MO	Identify the bottle in the gap between the bins, then put it inside the grey receptacle on the right.
S-AFF	Use the open grey bin on the right to store the mustard bottle by placing it inside.
S-AFF	Stow the mustard bottle inside the right-hand storage bin.
S-AFF	Put the mustard bottle used to add flavor to sandwiches inside the open-topped bin on the right.
S-AFF	Place the bottle used to dispense mustard onto food inside the right-hand bin.
S-AFF	Select the bottle of mustard used as a sandwich topping and place it inside the rectangular bin on the right.
S-INT	Pick up the bottle holding a condiment derived from mustard seeds and place it inside the right-hand open container.
S-INT	Please place the bottle containing the seed-based condiment mustard inside the right-hand storage bin.
S-INT	Locate the bottle of prepared mustard, then put it inside the open-topped bin on the right.
S-INT	Take hold of the bottle holding a condiment derived from mustard seeds and set it down inside the right-hand grey bin.
S-INT	Move the bottle containing the seed-based condiment mustard so that it rests inside the grey container on the right.

Table A7: **RoboLab unseen evaluation prompts: LargerObjectRaisinBoxInBin.**

Type	Prompt
S-PROP	Pick up the larger rectangular raisin box and place it inside the open grey bin.
S-PROP	Take the raisin box that is the larger of the two objects and put it inside the rectangular grey bin.
S-PROP	Move the larger raisin box from its initial position and place it inside the grey rectangular container.
S-PROP	Identify the raisin box with the larger overall size, then place it inside the open grey container.
S-PROP	Finish by placing the larger raisin box with the multicolored label inside the bin in front of the two packages in the initial image.
S-LANG	Put the raisin box in the open grey bin.
S-LANG	Transfer the raisin box into the open-topped grey container.
S-LANG	Move the raisin box so that it rests inside the grey container in the foreground of the image.
S-LANG	Take hold of the raisin box and put it inside the grey storage container.
S-LANG	Transfer the raisin box from its starting position into the gray bin.
S-MO	Identify the larger raisin box beside the smaller package and move it inside the grey receptacle.
S-MO	Take the larger raisin box on the far side of the grey bin as viewed in the image and put it inside the grey rectangular container.
S-MO	Choose the larger raisin box to the right of the other package in the image and put it inside the bin with the rounded corners.
S-MO	Take hold of the larger raisin box to the right of the other package in the image and set it down inside the bin in front of the two packages in the initial image.
S-MO	Select the larger raisin box behind the grey bin as viewed in the image and move it so that it rests inside the grey container.
S-AFF	Put the raisin box inside the grey bin in front of the packages in the initial image for storage.
S-AFF	Store the raisin box by placing it inside the grey rectangular container.
S-AFF	Use the open grey container to store the larger raisin box by placing it inside.
S-AFF	Place the raisin box inside the grey bin with the open top as a storage step.
S-AFF	Put the raisin box into storage by setting it down inside the grey container in the foreground of the image.
S-INT	Put the larger box containing dried grapes inside the open grey container.
S-INT	Locate the larger box containing fruit in the dried-grape category, then put it inside the bin in front of the two packages in the initial image.
S-INT	Identify the larger box containing fruit in the dried-grape category, then place it inside the rectangular grey bin.
S-INT	Take hold of the larger box containing grapes preserved by drying and set it down inside the grey storage bin.
S-INT	Transfer the larger box containing fruit in the dried-grape category inside the grey bin in front of the packages in the initial image.

Table A8: **RoboLab unseen evaluation prompts: BowlStackingLeftOnRight.**

Type	Prompt
S-PROP	Set down the left bowl with the circular rim on top of the right-side bowl.
S-PROP	Please place the bowl on the left with the red surface on top of the bowl at the right-hand position.
S-PROP	Take hold of the left bowl with the red interior and set it down on top of the right bowl with the red interior.
S-PROP	Move the bowl on the left with the red surface so that it rests on top of the bowl initially on the robot's right.
S-PROP	Transfer the left bowl with the curved sides on top of the right bowl with the curved sides.
S-LANG	Lift the left bowl and lower it onto the bowl initially to the right of the left bowl.
S-LANG	Take the left bowl and place it atop the red bowl to the right of the robot.
S-LANG	Move the left bowl so that it rests on top of the right bowl with the red interior.
S-LANG	Please pick up the left bowl and set it on the right bowl that will form the base of the stack.
S-LANG	Place the left bowl on the bowl on the right so that the left bowl rests on it.
S-MO	Please find the bowl that starts to the left of the right bowl and place it on top of the bowl that starts on the right.
S-MO	Choose the bowl positioned to the left of its counterpart and put it on top of the right bowl with the circular rim.
S-MO	Take hold of the bowl positioned to the left of its counterpart and set it down on top of the right bowl that will form the base of the stack.
S-MO	Find the bowl on the left side of the pair, then transfer it on top of the red bowl on the right.
S-MO	Identify the bowl positioned to the left of its counterpart; place it on top of the bowl on the robot's right and let go of it.
S-AFF	Put the left bowl on top of the bowl on the right-hand side to form a single stack.
S-AFF	Tidy the bowls by stacking the left bowl on top of the bowl occupying the right-hand position.
S-AFF	For a neat storage arrangement, stack the left bowl on the right bowl with the curved sides.
S-AFF	Place the left bowl used for serving food on top of the right bowl.
S-AFF	Please place the left bowl used for serving food on top of the right-side bowl.
S-INT	Put the left bowl-shaped item of tableware on top of the red bowl on the right, please.
S-INT	Identify the left bowl-shaped vessel, then place it on top of the bowl on the robot's right.
S-INT	Please pick up the left container classified as a bowl and put it on top of the bowl at the right-hand position.
S-INT	Position the left dish in the bowl category on top of the right-hand red bowl and release it.
S-INT	Move the left bowl-shaped item of tableware so that it rests on top of the round bowl on the right.

Table A9: **REALM training prompts.** OPSD-S is trained on the *Teacher*/*Student* pair. OPSD-M is trained on the remaining ten prompts, randomly sampled two per perturbation type. MOSAR (Ours) uses both the *Teacher* prompt and the same ten prompts.

Type	Prompt
	Rotate mug
Teacher	rotate the mug
Student	Adjust the mug closer to the robot than the large chocolate bar.
S-PROP	Change the orientation of the ceramic coffee cup.
S-PROP	Maneuver the uniform gray vessel.
S-LANG	Cause the mug to undergo rotation.
S-LANG	Do a rotation on the coffee cup.
S-MO	Twist the mug that is next to the loose piece of chocolate.
S-MO	Adjust the mug closer to the robot than the large chocolate bar.
S-AFF	Spin the handle of the mug toward you.
S-AFF	Spin the object used to hold a hot liquid.
S-INT	Rotate the common kitchen item.
S-INT	Adjust the position of the mug, which holds a unit of volume.
	Put banana into box
Teacher	put the banana in the box
Student	put the snack in the container
S-PROP	put the banana in the dark brown container
S-PROP	transfer the non-spherical fruit into the rectangular box
S-LANG	set the banana down within the box
S-LANG	have the banana enter the box
S-MO	deposit the banana into the object next to the orange
S-MO	put the banana in the container furthest from the packaged item
S-AFF	put the snack in the container
S-AFF	place the item used for quick energy in the box
S-INT	deposit the typically long yellow object in the rectangular prism
S-INT	deposit the high-fiber item in the storage unit

Table A10: **REALM unseen evaluation prompts: Rotate mug.**

Type	Prompt
S-PROP	Turn the tall, cylindrical drinking vessel.
S-PROP	Pivot the object that is standing upright on the desk.
S-PROP	Revolve the handle-less, tubular item.
S-PROP	Twist the small, light-colored cup.
S-PROP	Rotate the smooth-sided object used for holding liquids.
S-LANG	Spin the mug.
S-LANG	Pivot the mug.
S-LANG	Change the orientation of the mug.
S-LANG	Swivel the mug.
S-LANG	Make the mug turn about its vertical axis.
S-MO	Rotate the mug that is between the two chocolate items.
S-MO	Turn the object to the left of the large yellow chocolate bar.
S-MO	Revolve the mug farthest from the grey chair.
S-MO	Swivel the object to the right of the small wrapper piece.
S-MO	Maneuver the mug in the area with the two chocolate pieces.
S-AFF	Orient the mug to drink from it.
S-AFF	Twist the coffee cup for a better grip.
S-AFF	Rotate the mug to face the label forward.
S-AFF	Turn the mug as you would before pouring a drink.
S-AFF	Pivot the cylindrical object you drink tea from.
S-INT	Turn the object typically made of ceramic or porcelain.
S-INT	Spin the vessel with a thick wall.
S-INT	Turn the object whose internal structure is designed for liquid volume.
S-INT	Pivot the common type of heavy cup.
S-INT	Twist the item of pottery used for drinking.

Table A11: **REALM unseen evaluation prompts: Put banana into box.**

Type	Prompt
S-PROP	put the long yellow fruit in the wooden container
S-PROP	place the long, curved item in the brown tray
S-PROP	deposit the long yellow object in the box that contains the orange
S-PROP	move the banana onto the surface of the wooden box
S-PROP	transfer the fruit onto the tray with the largest item
S-LANG	place the banana inside the box
S-LANG	insert the banana into the container
S-LANG	transfer the banana to the box
S-LANG	convey the banana into the box
S-LANG	place the fruit in the storage container
S-MO	place the banana in the container closest to the camera
S-MO	transfer the banana to the object that is on the bottom left
S-MO	move the banana into the object that is near the table's edge
S-MO	place the banana in the container to the left of the packaged item
S-MO	deposit the banana in the object that is not holding the lemon
S-AFF	store the yellow fruit in the box
S-AFF	place the fresh fruit in the container
S-AFF	store the item often used for smoothies in the box
S-AFF	place the item you peel before eating in the container
S-AFF	put the convenient grab-and-go food in the box
S-INT	put the curved fruit in the box
S-INT	place the potassium source in the box
S-INT	move the crescent-shaped food into the box
S-INT	put the tropical crop in the box
S-INT	transfer the food that grows in bunches into the container

B Details on Preliminaries and Methods

B.1 Flow-matching sampler and ODE-to-SDE conversion

Deterministic sampler. Flow matching trains v_θ to regress $\epsilon - a$ at $x_t = t\epsilon + (1-t)a$ (cf. Eq. 2), so $t = 1$ corresponds to noise and $t = 0$ to the action. Starting from $x_1 \sim \mathcal{N}(0, I)$, the sampler integrates the field from $t = 1$ to $t = 0$ with N Euler steps of size $\delta = 1/N$,

$$x_{t-\delta} = x_t - \delta\, v_\theta(x_t, t \mid o, p), \qquad a = x_0, \tag{A1}$$

which defines the action $a = \pi_\theta(o, p)$ of Sec. 3. Given x_1 the map is deterministic, so the likelihood of an action is not available at a cost that policy-gradient training can afford.

Stochastic sampler. To recover a likelihood, Flow-GRPO (Liu et al., 2025a) converts the ODE into an SDE whose marginals match those of the ODE at every t in continuous time; in the convention of Eq. A1, one Euler–Maruyama step reads

$$x_{t-\delta} = x_t - \delta\Big[v_\theta + \frac{\sigma_t^2}{2t}\big(x_t + (1-t)\, v_\theta\big)\Big] + \sigma_t\sqrt{\delta}\,\xi, \qquad \sigma_t = \nu\sqrt{\tfrac{t}{1-t}}, \quad \xi \sim \mathcal{N}(0, I), \tag{A2}$$

with v_θ evaluated at $(x_t, t \mid o, p)$ and ν the noise level. Each step is therefore a Gaussian transition with mean $\mu_\theta(x_t, t) = x_t - \delta\big[v_\theta + \frac{\sigma_t^2}{2t}\big(x_t + (1-t)v_\theta\big)\big]$ and covariance $\delta\sigma_t^2 I$, whose log-likelihood is available in closed form,

$$\log \mathcal{N}\big(x_{t-\delta};\, \mu_\theta(x_t, t),\, \delta\sigma_t^2 I\big) = -\frac{\|x_{t-\delta} - \mu_\theta(x_t, t)\|_2^2}{2\delta\sigma_t^2} - \frac{HD}{2}\log\big(2\pi\delta\sigma_t^2\big). \tag{A3}$$

Since σ_t does not depend on θ, the second term cancels in the likelihood ratio $\rho_\theta(z)$ of App. B.3. The denoising chain can thus be treated as a Markov decision process and optimized with PPO (Schulman et al., 2017; Chen et al., 2025); the two-layer MDP, the hybrid ODE–SDE sampler used for rollouts, and the loss ℓ_{R} are given in App. B.3.

B.2 Algorithm of MOSAR

Algorithm 1 One training iteration of MOSAR

Require: policy parameters θ; optimizer states $\psi_{\mathrm{R}}, \psi_{\mathrm{D}}$; prompt sets $\{\mathcal{P}_k\}_{k=1}^K$ with teacher prompts $\{p_k^{\mathrm{T}}\}_{k=1}^K$ and student prompt sets $\mathcal{S}_k = \mathcal{P}_k \setminus \{p_k^{\mathrm{T}}\}$; running statistics $\{r_k^{\mathrm{T}}, r_k^{\mathrm{S}}, r_k^{\max}, \lambda_k\}_{k=1}^K$

for $k = 1, \dots, K$ **do**
  sample M student prompts from $\mathcal{S}_k$
  collect rollouts under p_k^{T} and under the sampled student prompts, forming $\mathcal{B}_k^{\mathrm{T}}$ and $\mathcal{B}_k^{\mathrm{S}}$
  $\mathcal{B}_k \leftarrow \mathcal{B}_k^{\mathrm{T}} \cup \mathcal{B}_k^{\mathrm{S}}$
  update r_k^{T}, r_k^{S}, Δ_k, and $r_k^{\max}$
  $\lambda_k^\star \leftarrow g(\Delta_k)\, b_k$ ▷ Eqs. 6–7
  update λ_k by Eq. 8; $w_k \leftarrow \lambda_k / g_{\mathrm{mid}}$
end for
$\mathcal{B} \leftarrow \bigcup_k \mathcal{B}_k$, $\mathcal{B}^{\mathrm{S}} \leftarrow \bigcup_k \mathcal{B}_k^{\mathrm{S}}$
for E epochs **do**
  for each mini-batch $\widetilde{\mathcal{B}} \subset \mathcal{B}$ **do**
    $\theta \leftarrow U_{\mathrm{R}}(\theta; \widetilde{\mathcal{B}})$ ▷ RL update on Eq. 10; updates ψ_{R}
    $\widetilde{\mathcal{B}}^{\mathrm{S}} \leftarrow \widetilde{\mathcal{B}} \cap \mathcal{B}^{\mathrm{S}}$
    if $\exists\, z \in \widetilde{\mathcal{B}}^{\mathrm{S}}$ with $w_{k(z)} > 0$ **then**
      $a_z^{\mathrm{T}} \leftarrow \mathrm{sg}\left[\pi_\theta(o_z, p_{k(z)}^{\mathrm{T}}; x_1)\right]$ with fresh $x_1 \sim \mathcal{N}(0, I)$, for all $z \in \widetilde{\mathcal{B}}^{\mathrm{S}}$
      $\theta \leftarrow U_{\mathrm{D}}(\theta; \widetilde{\mathcal{B}}^{\mathrm{S}})$ ▷ OPSD update on Eq. 9; updates ψ_{D}
    end if
  end for
end for

B.3 DENOISING MDP AND PPO

Following π_{RL} (Chen et al., 2025), the denoising chain of one action chunk is the inner layer of a two-layer MDP whose outer layer is the environment. The state is $\bar{s} = (o, p, x_t, t)$, the action is the next denoising state $x_{t-\delta}$, and the environment reward is granted when the chunk is complete and executed. To keep the horizon short, rollouts use hybrid ODE–SDE sampling: for each chunk, one denoising step drawn uniformly among the first J of the N steps follows Eq. A2, and the remaining steps follow Eq. A1. The ratio $t/(1-t)$ is undefined at the first grid time $t = 1$; at that step only, the implementation evaluates it with the next grid time in the denominator, so that $\sigma_1 = \nu\sqrt{N}$. Evaluation uses the deterministic sampler of Eq. A1.

A sample z is one chunk's stochastic transition together with its observation and prompt. Let $\rho_\theta(z)$ be the likelihood ratio of that transition between the current and the rollout policy, and $\hat{A}_z$ its GAE advantage, normalized over the whole rollout batch. A value head on the VLM features gives the clipped value loss $\ell_V(z)$. The per-sample RL loss is $\ell_{\mathrm{R}}(z) = \ell_{\mathrm{A}}(z) + \ell_V(z)$ with the dual-clipped surrogate (Ye et al., 2020)

$$\ell_{\mathrm{A}}(z) = \begin{cases} -\min\left\{\rho_\theta(z)\hat{A}_z,\ \mathrm{clip}(\rho_\theta(z), 1-\epsilon_{\mathrm{c}}, 1+\epsilon_{\mathrm{c}})\,\hat{A}_z\right\}, & \hat{A}_z \geq 0, \\ -\max\left\{\min\{\rho_\theta(z)\hat{A}_z,\ \mathrm{clip}(\rho_\theta(z), 1-\epsilon_{\mathrm{c}}, 1+\epsilon_{\mathrm{c}})\,\hat{A}_z\},\ c_{\mathrm{d}}\hat{A}_z\right\}, & \hat{A}_z < 0, \end{cases} \tag{A4}$$

where ϵ_{c} is the clipping range and $c_{\mathrm{d}} > 1$ the dual-clip constant. In $\mathcal{L}_{\mathrm{R}}$ of Eq. 10, the advantage entering the actor term is scaled by $\alpha_{\mathrm{T}} = 1$ for teacher-prompt samples and by $\alpha_{\mathrm{S}} < 1$ for student-prompt samples after normalization; the value loss is not weighted. This down-weights the student half of the batch, whose episodes rarely succeed early in training and whose advantages add mostly noise to the actor step.

Table A12: Relative prompt-level variability on trained and held-out unseen prompts. Mean success and CV are computed over per-prompt success rates, where $\mathrm{CV} = \mathrm{SD}/\mathrm{Mean}$. Lower CV indicates more consistent performance across prompt formulations.

	Trained prompts		Unseen prompts	
Method	Mean success (%) ↑	CV (%) ↓	Mean success (%) ↑	CV (%) ↓
$\pi_{0.5}$	37.8	65.3	32.4	79.8
COSMOS$_{\mathrm{Edge}}$	36.7	93.1	37.3	104.3
COSMOS$_{\mathrm{Nano}}$	35.1	79.1	37.4	83.4
PPO	43.0	57.1	33.7	90.5
GRPO	33.3	74.4	33.0	87.7
OPSD-S	43.4	55.7	41.1	71.5
OPSD-M	38.3	63.8	34.6	77.4
Ours	**51.6**	**51.0**	**53.2**	**55.0**

C Additional Analyses and Results

C.1 Using separate optimizer states.

A joint loss $\mathcal{L}_{\mathrm{R}} + \omega\mathcal{L}_{\mathrm{D}}$ with one adaptive optimizer normalizes the sum of the two gradients by one second-moment estimate per parameter. Measured from the Adam moment estimates stored with the checkpoints of the main run, the root-mean-square gradient of the RL step was four to ten times that of the distillation step, depending on the module. With a shared estimate, the coordinates where one gradient dominates are normalized by that gradient. The share of the other objective is then set per coordinate by ω together with a magnitude ratio that is unobserved and can change over training. A change of a task weight, including its zero, changes the normalization seen by the RL gradient as well. Separate states give each objective its own normalization, momentum and step size, and the resulting ratio $\eta_{\mathrm{D}}/\eta_{\mathrm{R}}$ is one scalar, logged every iteration. The single-optimizer variant of the ablation keeps the two alternating steps but feeds both through one AdamW state, the RL step at η_0 and the distillation step at $0.55\,\eta_0$, with the controller removed.

C.2 Relative Prompt-Level Variability

We further quantify the consistency of each method across held-out prompt formulations using the coefficient of variation (CV). Let $r_i = y_i/n_i$ denote the success rate for prompt i, where y_i is the number of successful rollouts and n_i is the number of evaluation rollouts. For each method, we compute

$$\mathrm{CV} = \frac{s_r}{\bar{r}}, \qquad \bar{r} = \frac{1}{N}\sum_{i=1}^{N} r_i, \qquad s_r = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(r_i - \bar{r})^2}, \tag{A5}$$

Unlike the raw standard deviation, which is expressed in absolute percentage points, the CV measures dispersion relative to a method's own mean performance. This normalization is important when comparing methods with different average success rates: a low raw standard deviation can arise mechanically when a method has a low mean success rate.

Table A12 reports the mean prompt-level success rate, the raw standard deviation, and the resulting CV. Ours attains the highest mean success rate (53.2%) while also having the lowest relative prompt-level variability ($\mathrm{CV} = 55.0\%$). Thus, Ours improves average held-out performance while reducing relative variation across prompt formulations.

The CV should be interpreted as a relative-dispersion measure rather than an absolute measure of uncertainty. In particular, it becomes unstable when the mean approaches zero. All methods in this experiment have mean prompt-level success rates above 32%, so the CV remains well-defined and provides a useful normalized comparison.

D ADDITIONAL QUALITATIVE RESULTS

This section shows the RoboLab scenes used for training and evaluation, one selected rollout per policy, one example of the prompt sensitivity of $\pi_{0.5}$, and the real-robot runs. Selected examples are chosen by the rules stated in each caption, together with the counts over the batch they come from.

D.1 QUALITATIVE VISUALIZATION ON SIMULATION

We summarize the simulation benchmark experiments in Fig. A1 and Fig. A2. Fig. A1 shows qualitative results under the same prompt and scene condition, where Ours successfully completes the task while the other methods incorrectly grasp the ranch dressing instead of the target sauce. Fig. A2 presents the simulation setups used in our experiments, with the left column showing the original settings and the right column showing the corresponding settings with additional visual distractors.

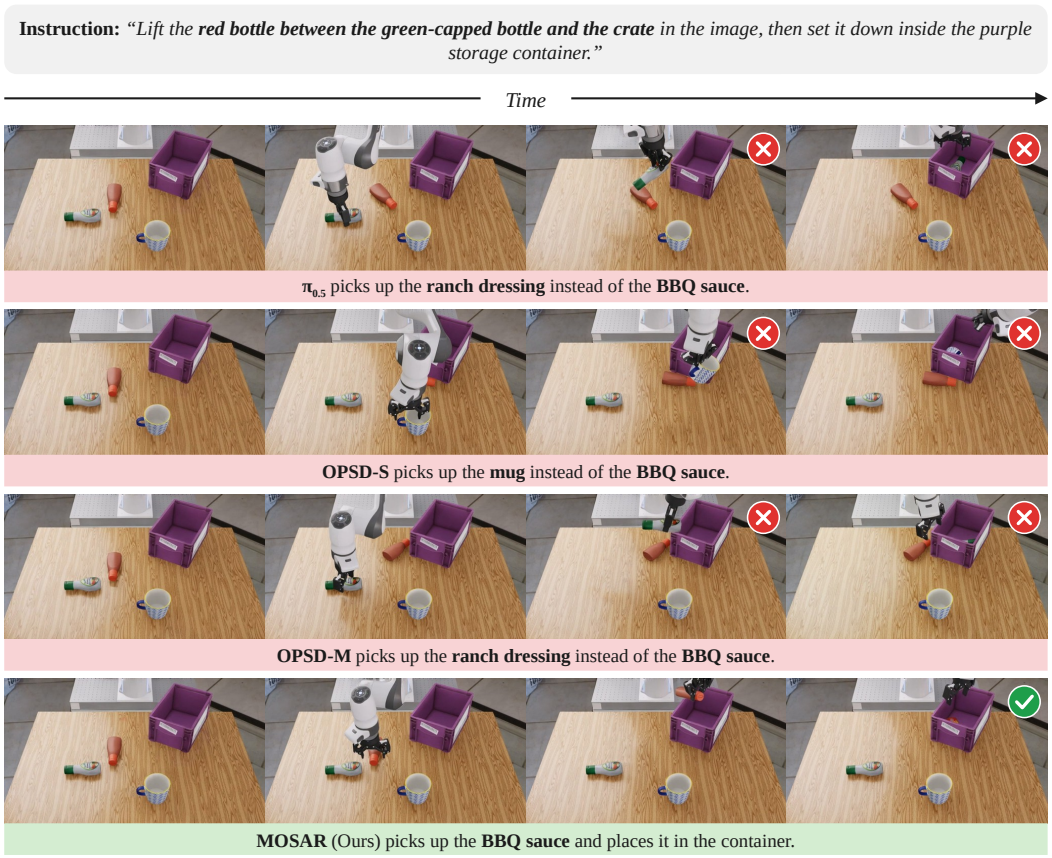


Figure A1: **Simulation evaluation.** We evaluate $\pi_{0.5}$, OPSD-S, OPSD-M, and MOSAR (ours) under the same instruction and scene.

D.2 QUALITATIVE RESULTS ON FRANKA REAL ROBOT

We conduct the real-robot evaluation using the same object configurations across all three baselines. The setups are designed to resemble the SauceBottlesCrate and LargerObjectRaisinBoxInBin tasks, with an additional cube introduced as a visual distractor for LargerObjectRaisinBoxInBin. The qualitative results are shown in Fig. A3–Fig. A6.

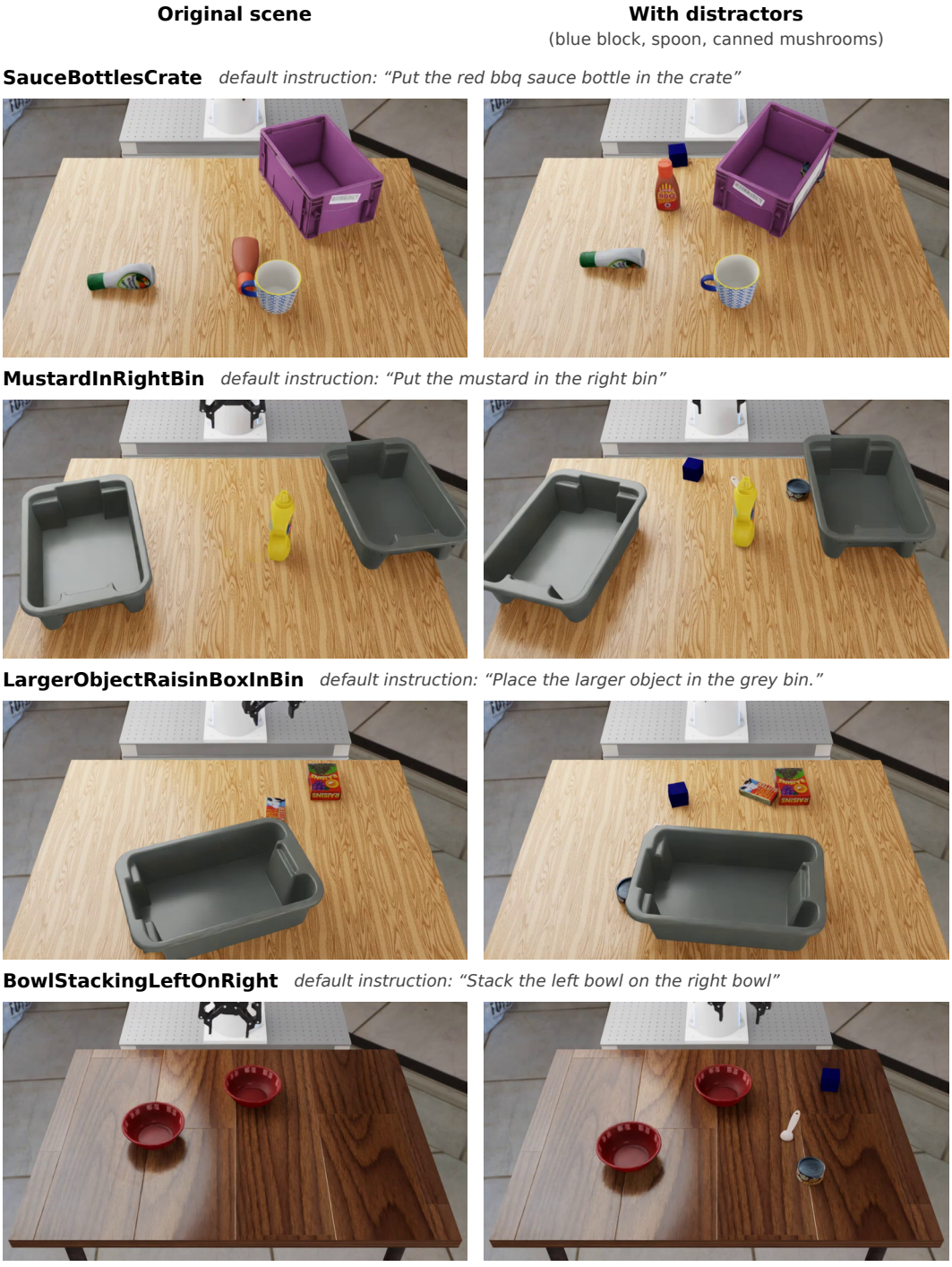


Figure A2: **Visualization of RoboLab tasks**. The left column shows the four original task setups, while the right column shows the corresponding setups with visual perturbations.

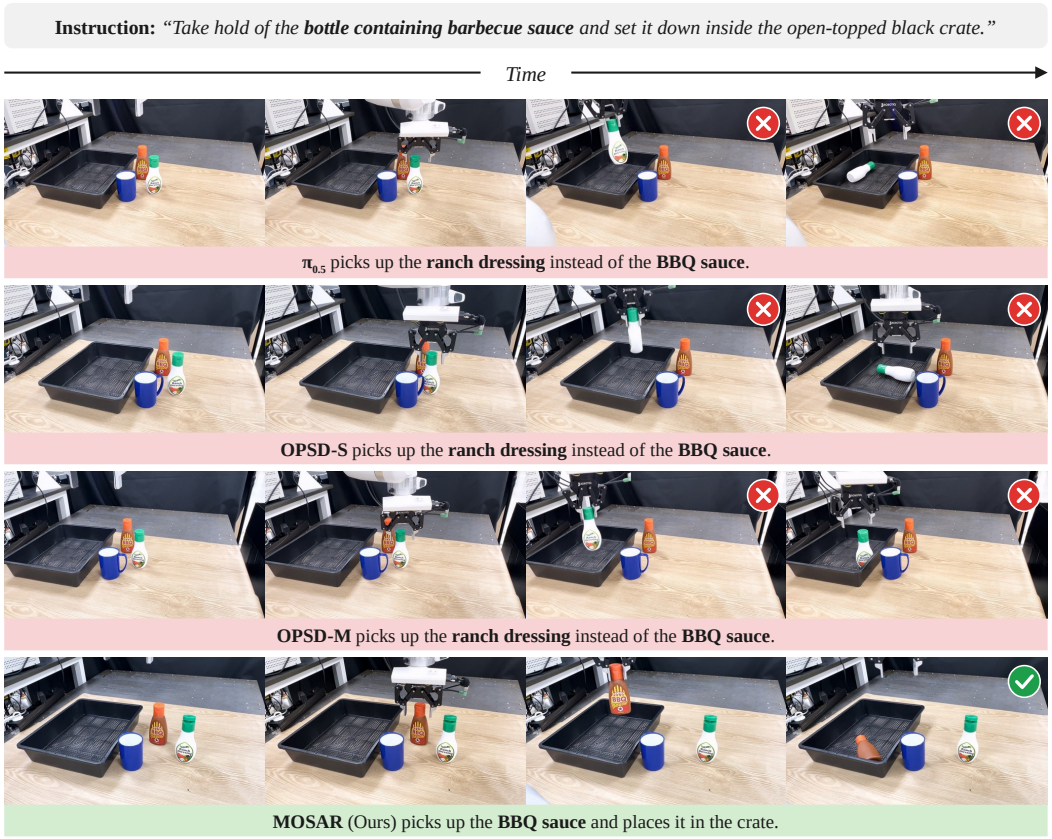


Figure A3: **Real-robot evaluation.** We evaluate $\pi_{0.5}$, OPSD-S, OPSD-M, and MOSAR (ours) under the same instruction and scene.

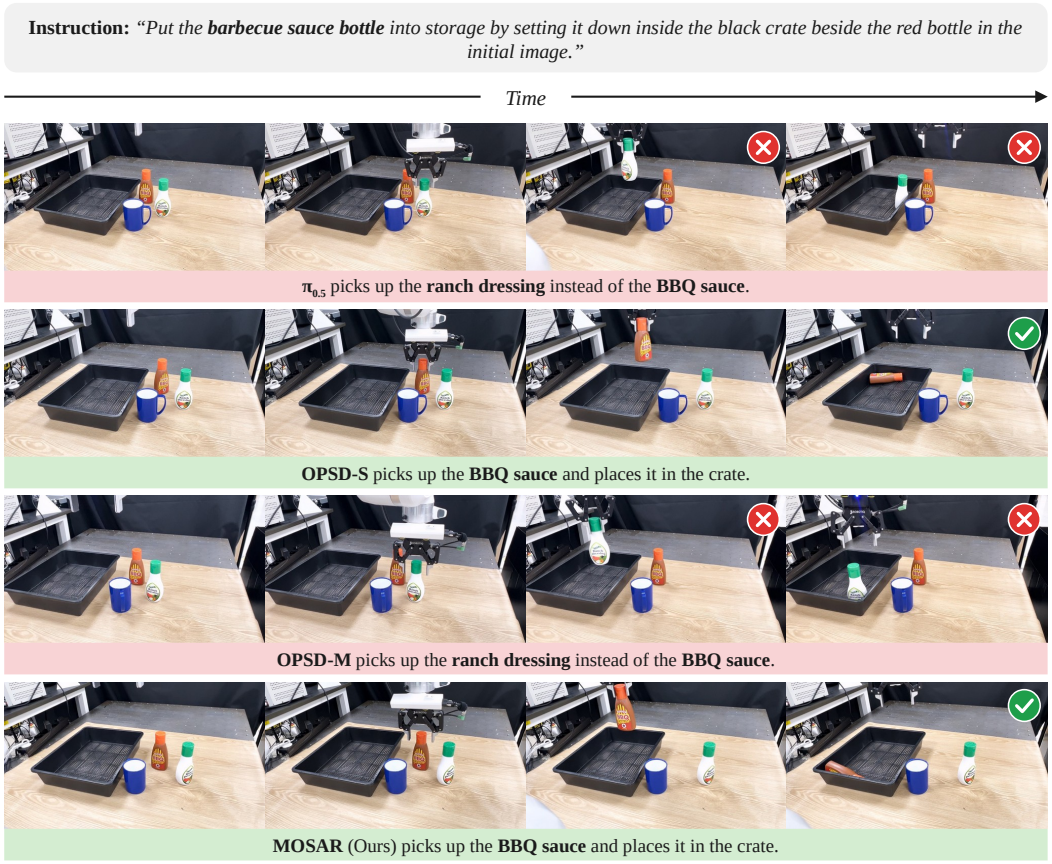


Figure A4: **Real-robot evaluation.** We evaluate $\pi_{0.5}$, OPSD-S, OPSD-M, and MOSAR (ours) under the same instruction and scene.

Figure A5: **Real-robot evaluation.** We evaluate $\pi_{0.5}$, OPSD-S, OPSD-M, and MOSAR (ours) under the same instruction and scene.

Figure A6: **Real-robot evaluation.** We evaluate $\pi_{0.5}$, OPSD-S, OPSD-M, and MOSAR (ours) under the same instruction and scene.

E THEORETICAL ANALYSIS OF MULTI-STUDENT PROMPT ROBUSTNESS

E.1 OVERVIEW

This appendix formalizes the coverage–displacement perspective underlying multi-student OPSD. Our analysis addresses two questions.

First, why can training on multiple semantically equivalent student prompts improve robustness to unseen paraphrases? We show that success on an unseen prompt can be related to success on the covered training prompts through its distance to the covered prompt set. Enlarging the student-prompt set monotonically improves this coverage quantity, and, when student prompts are sampled from a prompt distribution, the expected coverage error decreases with the number of sampled student prompts.

Second, why is parameter displacement relevant when improving prompt robustness? The unseen-prompt bound depends not only on prompt coverage but also on the parameter displacement $\|\theta - \theta_0\|$ from the pretrained policy. The same displacement also controls changes in success on unseen tasks. Together, these results characterize a coverage–displacement trade-off: multi-student training improves coverage of the prompt space, while controlling movement from the pretrained policy limits degradation of previously acquired behavior.

Finally, we analyze on-policy distillation under an explicit strategy-mixture model and a linearized regression approximation. This analysis provides a sufficient condition under which OPSD requires a smaller minimum-norm parameter update than teacher-trajectory SFT. We use this comparison only to illustrate one mechanism by which student-state supervision can improve prompt coverage while controlling parameter displacement; it does not imply unconditional superiority of OPSD over SFT.

The analysis isolates the multi-student OPSD component of MOSAR and does not model discrepancy-dependent task weighting, alternating RL updates, adaptive optimization, or the evolving teacher policy.

E.2 PRELIMINARIES

Rollouts and success rates. An episode contains L action chunks. At chunk $\ell \in \{0, \dots, L-1\}$, a policy observes o_ℓ, samples $a_\ell \in \mathbb{R}^{H \times D}$, and receives the next observation from $P(\cdot \mid o_\ell, a_\ell)$. For the analysis, o_ℓ includes the time index and sufficient information for these transitions to be Markov.

All policies are evaluated from the same initial observation distribution d_0. The episode terminates with a binary success indicator $R(o_L) \in \{0, 1\}$, and we define

$$J(\pi) = \Pr_{\pi}(R = 1), \qquad J(\theta, p) = J(\pi_\theta(\cdot \mid \cdot, p)) \,. \tag{A6}$$

Define

$$Q_\ell^\pi(o, a) = \Pr_{\pi}(R = 1 \mid o_\ell = o, a_\ell = a), \tag{A7}$$

$$V_\ell^\pi(o) = \mathbb{E}_{a \sim \pi(\cdot \mid o)} Q_\ell^\pi(o, a), \tag{A8}$$

with $V_L^\pi(o) = R(o)$. Both are in $[0, 1]$. Let d_ℓ^π denote the observation distribution at chunk ℓ.

Prompt space and covered prompts. Fix a task k and let $\mathcal{Q}_k$ denote the set of semantically equivalent prompts describing its goal. We equip $\mathcal{Q}_k$ with a distance $d(\cdot, \cdot)$.

Let p_k^{T} be the teacher prompt and

$$\mathcal{S}_k^{(M)} = \{p_1^{\mathrm{S}}, \dots, p_M^{\mathrm{S}}\} \tag{A9}$$

be a set of M student prompts used by multi-student training. We define the corresponding covered prompt set as

$$\mathcal{C}_k^{(M)} = \{p_k^{\mathrm{T}}\} \cup \mathcal{S}_k^{(M)}. \tag{A10}$$

For any prompt $q \in \mathcal{Q}_k$, define its distance to the covered prompt set by

$$d(q, \mathcal{C}_k^{(M)}) = \min_{p \in \mathcal{C}_k^{(M)}} d(q, p). \tag{A11}$$

We consider both the worst-case coverage radius

$$\rho_M = \sup_{q \in \mathcal{Q}_k} d(q, \mathcal{C}_k^{(M)}) \tag{A12}$$

and, for an unseen-prompt distribution $\mathcal{D}_k$, the average coverage error

$$\bar{d}_M = \mathbb{E}_{q \sim \mathcal{D}_k} d(q, \mathcal{C}_k^{(M)}). \tag{A13}$$

E.3 Assumptions

We first state the smoothness assumptions used to relate prompt distance and parameter displacement to differences between prompt-conditioned policies.

Assumption A1 (Prompt and parameter smoothness)**.** *For prompts p, p' describing the same task, assume that the pretrained velocity field satisfies*

$$\|v_{\theta_0}(x, t \mid o, p) - v_{\theta_0}(x, t \mid o, p')\| \le L_0 d(p, p') \tag{A14}$$

uniformly over (o, x, t).

We further assume that the change induced by fine-tuning is Lipschitz with respect to the prompt:

$$\begin{aligned} &\|[v_\theta - v_{\theta_0}](x, t \mid o, p) - [v_\theta - v_{\theta_0}](x, t \mid o, p')\| \\ &\qquad \le L_\Phi \|\theta - \theta_0\| d(p, p'). \end{aligned} \tag{A15}$$

Finally, for any fixed prompt p, assume

$$\|v_\theta(x, t \mid o, p) - v_{\theta_0}(x, t \mid o, p)\| \le L_\theta \|\theta - \theta_0\|. \tag{A16}$$

These assumptions separately control the pretrained model's sensitivity to prompt variation, the additional prompt sensitivity introduced by fine-tuning, and the effect of parameter displacement at a fixed prompt.

E.4 Prompt Coverage of Multiple Students

We first isolate the effect of using multiple student prompts independently of the optimization procedure.

Proposition 1 (Monotonicity of multi-student coverage)**.** *Suppose*

$$\mathcal{S}_k^{(M)} \subseteq \mathcal{S}_k^{(M+1)}. \tag{A17}$$

Then

$$\rho_{M+1} \le \rho_M, \qquad \bar{d}_{M+1} \le \bar{d}_M. \tag{A18}$$

Proof. Since $\mathcal{C}_k^{(M)} \subseteq \mathcal{C}_k^{(M+1)}$, for every $q \in \mathcal{Q}_k$,

$$d(q, \mathcal{C}_k^{(M+1)}) = \min_{p \in \mathcal{C}_k^{(M+1)}} d(q, p) \tag{A19}$$

$$\le \min_{p \in \mathcal{C}_k^{(M)}} d(q, p) \tag{A20}$$

$$= d(q, \mathcal{C}_k^{(M)}). \tag{A21}$$

Taking the supremum over q gives $\rho_{M+1} \le \rho_M$, while taking expectation over $q \sim \mathcal{D}_k$ gives $\bar{d}_{M+1} \le \bar{d}_M$. □

The preceding result is deterministic: adding a covered prompt cannot worsen the distance to the covered set. We next quantify the expected coverage when the student prompts themselves are sampled.

Proposition 2 (Expected coverage with M student prompts)**.** *Let $p_1^{\mathrm{S}}, \ldots, p_M^{\mathrm{S}}$ be sampled independently from $\mathcal{D}_k$. Suppose $\mathrm{supp}(\mathcal{D}_k)$ is bounded and admits a partition into N_h cells, each having diameter at most $2h$. Then*

$$\mathbb{E}_{\mathcal{S}_k^{(M)}}[\bar{d}_M] \le 2h + \mathrm{Diam}(\mathrm{supp}\,\mathcal{D}_k)\frac{N_h}{M+1}. \tag{A22}$$

Proof. Let cell c have probability mass q_c under $\mathcal{D}_k$. Consider a test prompt q belonging to cell c. If at least one of the M sampled student prompts also lies in c, then, because the cell has diameter at most $2h$,

$$d(q, \mathcal{C}_k^{(M)}) \le 2h. \tag{A23}$$

The probability that no sampled student prompt lies in the same cell is $(1-q_c)^M$. In this event, we use the trivial bound

$$d(q, \mathcal{C}_k^{(M)}) \le \mathrm{Diam}(\mathrm{supp}\,\mathcal{D}_k). \tag{A24}$$

Averaging over cells therefore gives

$$\mathbb{E}[\bar{d}_M] \le 2h + \mathrm{Diam}(\mathrm{supp}\,\mathcal{D}_k)\sum_{c=1}^{N_h} q_c(1-q_c)^M \tag{A25}$$

$$\le 2h + \mathrm{Diam}(\mathrm{supp}\,\mathcal{D}_k)\frac{N_h}{M+1}, \tag{A26}$$

where the last inequality follows from $q(1-q)^M \le 1/(M+1)$. □

Propositions 1 and 2 formalize the coverage benefit of multi-student training. Increasing the number of covered student prompts cannot increase the distance to the covered prompt set, and under random sampling the expected coverage error decreases with M.

E.5 Prompt Robustness and Parameter Displacement

We next connect prompt coverage to task success.

Theorem 1 (Prompt distance and parameter displacement)**.** *Under Assumption A1 and the stochastic-sampler conditions stated above, for any two prompts p, p' describing the same task,*

$$|J(\theta, p) - J(\theta, p')| \le L\Lambda\left(L_0 + L_\Phi\|\theta - \theta_0\|\right) d(p, p'). \tag{A27}$$

Consequently, for any unseen prompt $q \in \mathcal{Q}_k$,

$$J(\theta, q) \ge \min_{p \in \mathcal{C}_k^{(M)}} J(\theta, p) - L\Lambda\left(L_0 + L_\Phi\|\theta - \theta_0\|\right) d(q, \mathcal{C}_k^{(M)}). \tag{A28}$$

Averaging over $q \sim \mathcal{D}_k$ gives

$$\mathbb{E}_{q\sim\mathcal{D}_k} J(\theta, q) \ge \min_{p \in \mathcal{C}_k^{(M)}} J(\theta, p) - L\Lambda\left(L_0 + L_\Phi\|\theta - \theta_0\|\right) \bar{d}_M. \tag{A29}$$

For any prompt p'' belonging to an unseen task,

$$|J(\theta, p'') - J(\theta_0, p'')| \le L\Lambda L_\theta\|\theta - \theta_0\|. \tag{A30}$$

Proof. For two prompts p and p' describing the same task, decompose the difference between their velocity fields as

$$v_\theta(\cdot \mid p') - v_\theta(\cdot \mid p) = \left[v_{\theta_0}(\cdot \mid p') - v_{\theta_0}(\cdot \mid p)\right] + \left[(v_\theta - v_{\theta_0})(\cdot \mid p') - (v_\theta - v_{\theta_0})(\cdot \mid p)\right]. \tag{A31}$$

Assumption A1 therefore gives

$$\|v_\theta(\cdot \mid p') - v_\theta(\cdot \mid p)\| \le (L_0 + L_\Phi \|\theta - \theta_0\|)\, d(p, p') \tag{A32}$$

uniformly over the observation, denoising state, and time.

Applying the stochastic-chain KL bound to the two model velocity fields, followed by data processing and Pinsker's inequality, gives

$$\begin{aligned} &\mathrm{TV}\left(\pi_\theta(\cdot \mid o, p), \pi_\theta(\cdot \mid o, p')\right) \\ &\qquad \le \Lambda\left(L_0 + L_\Phi \|\theta - \theta_0\|\right) d(p, p'). \end{aligned} \tag{A33}$$

Each term of the finite-horizon performance-difference identity is a difference of expectations of a function taking values in $[0, 1]$ and is therefore bounded by the corresponding total variation distance. Summing over the L action chunks proves Eq. equation A27.

For an unseen prompt q, choose

$$p^\star \in \arg\min_{p \in \mathcal{C}_k^{(M)}} d(q, p). \tag{A34}$$

Applying Eq. equation A27 to $(q, p^\star)$ and using

$$J(\theta, p^\star) \ge \min_{p \in \mathcal{C}_k^{(M)}} J(\theta, p) \tag{A35}$$

gives Eq. equation A28. Taking expectation over $q \sim \mathcal{D}_k$ gives Eq. equation A29.

Finally, at a fixed unseen-task prompt p'', Assumption A1 gives

$$\|v_\theta(\cdot \mid p'') - v_{\theta_0}(\cdot \mid p'')\| \le L_\theta \|\theta - \theta_0\|. \tag{A36}$$

Applying the same stochastic-chain KL, data-processing, Pinsker, and finite-horizon performance arguments proves Eq. equation A30. □

E.6 The Coverage–Displacement Trade-off

Combining the preceding coverage and robustness results makes the role of multiple student prompts explicit.

Corollary 1 (Coverage–displacement bound)**.** *Under the conditions of Theorem 1 and Proposition 2,*

$$\begin{aligned} &\mathbb{E}_{\mathcal{S}_k^{(M)}} \mathbb{E}_{q \sim \mathcal{D}_k} J(\theta, q) \\ &\quad \ge \mathbb{E}_{\mathcal{S}_k^{(M)}} \left[\min_{p \in \mathcal{C}_k^{(M)}} J(\theta, p) \right] \\ &\qquad - L\Lambda\left(L_0 + L_\Phi \|\theta - \theta_0\|\right) \left[2h + \mathrm{Diam}(\mathrm{supp}\, \mathcal{D}_k) \frac{N_h}{M+1} \right]. \end{aligned} \tag{A37}$$

Proof. Apply Eq. equation A29 and take expectation over the sampled student-prompt set. Proposition 2 bounds the expected coverage term, which gives Eq. equation A37. □

Interpretation. The bound separates three quantities: performance on the covered prompts, coverage of the unseen-prompt distribution, and parameter displacement. Increasing the number of student prompts directly improves the coverage term in expectation. This does not imply that the complete lower bound improves monotonically with M, since the minimum success rate over the covered prompts and the learned parameter displacement may themselves change with M. Rather, the result isolates the coverage benefit contributed by multi-student training.

At the same time, $\|\theta - \theta_0\|$ appears in both the unseen-prompt bound and the unseen-task bound. Thus, broad prompt coverage is most useful when it can be obtained without unnecessarily large movement from the pretrained policy. This motivates the second part of our analysis.

E.7 WHY ON-POLICY DISTILLATION CAN CONTROL PARAMETER DISPLACEMENT

The preceding results do not depend on a particular training objective. We now analyze one sufficient setting in which on-policy student-state supervision can require a smaller parameter change than supervision on teacher trajectories. The purpose of this comparison is to characterize the parameter-displacement side of the coverage–displacement trade-off, rather than to establish unconditional superiority over SFT.

Teacher and initial student policies. Assume that the teacher contains G strategies with weights $\varpi_j \geq 0$ and $\sum_j \varpi_j = 1$:

$$\pi^{\mathrm{T}}(\cdot \mid o) = \begin{cases} \sum_{j=1}^{G} \varpi_j \mathcal{N}(\mu_j(o), \varsigma^2 I), & \text{before strategy selection,} \\ \mathcal{N}(\mu_j(o), \varsigma^2 I), & \text{after selecting strategy } j. \end{cases} \tag{A38}$$

For student m, let the initial policy be

$$\pi^{\mathrm{S}}_{m,0}(\cdot \mid o) = \mathcal{N}(\mu^{\mathrm{S}}_m(o), \varsigma^2 I). \tag{A39}$$

Assume

$$\|\mu_j(o) - \mu^{\mathrm{S}}_m(o)\| = d_{j,m} \tag{A40}$$

is independent of o, and define

$$d_{\star,m} = \min_j d_{j,m}, \qquad \bar{d}^2_m = \sum_j \varpi_j d^2_{j,m}, \tag{A41}$$

and

$$d^2_\star = \frac{1}{M} \sum_{m=1}^{M} d^2_{\star,m}, \qquad \bar{d}^2 = \frac{1}{M} \sum_{m=1}^{M} \bar{d}^2_m. \tag{A42}$$

After strategy selection, assume that observations collected by student m follow a nearest teacher strategy, whereas teacher-trajectory observations contain the teacher strategies with frequencies ϖ_j. Let β_b denote the fraction of observations occurring before strategy selection.

For the exact comparison with success-conditioned SFT targets, we additionally assume that conditioning teacher trajectories on success does not change the teacher action distribution on the SFT observations. This assumption is used only for the residual comparison below.

Linearized regression. We linearize the velocity field around θ_0 with fixed training inputs and target velocity fields. Let Φ be the Jacobian of the n scalar outputs across all training inputs and student prompts, and define

$$K = \frac{1}{n} \Phi \Phi^\top. \tag{A43}$$

Assume $K \succ 0$. For a normalized target-minus-initial-output residual vector ϱ, the minimum-norm parameter change matching the targets satisfies

$$\|\theta - \theta_0\|^2 = \varrho^\top K^{-1} \varrho. \tag{A44}$$

Let Γ_{OPSD} and Γ_{SFT} denote the mean normalized squared initial regression residuals for the two objectives, averaged over training observations, noisy actions, denoising time, and all student prompts.

Under the strategy-mixture assumptions above, student-state supervision requires corrections primarily toward the strategy reached by each student, whereas teacher-trajectory supervision contains corrections toward the full mixture of teacher strategies. This yields an upper bound on Γ_{OPSD} and a corresponding lower bound on Γ_{SFT}.

In particular, the residual analysis gives

$$\Gamma_{\mathrm{OPSD}}(t) \leq \frac{\kappa_t^2}{HD} \Big[(1-\beta)(1-\beta_b) d^2_\star + \big(1 - (1-\beta)(1-\beta_b)\big) \bar{d}^2 \Big], \tag{A45}$$

whereas

$$\Gamma_{\mathrm{SFT}}(t) \geq \frac{\kappa_t^2}{HD}(1-\beta_b)\bar{d}^2. \tag{A46}$$

The Gaussian and Gaussian-mixture flow-matching calculations yielding these expressions are given in the following lemmas.

Theorem 2 (Minimum-norm displacement under OPSD)**.** *Under the strategy-mixture and linearized-regression assumptions above, if* $\Gamma_{\mathrm{SFT}} > 0$*, then*

$$\|\theta_{\mathrm{OPSD}} - \theta_0\|^2 \leq C_K \frac{\Gamma_{\mathrm{OPSD}}}{\Gamma_{\mathrm{SFT}}} \|\theta_{\mathrm{SFT}} - \theta_0\|^2, \tag{A47}$$

where

$$C_K = \frac{\lambda_{\max}(K_{\mathrm{SFT}})}{\lambda_{\min}(K_{\mathrm{OPSD}})}. \tag{A48}$$

The kernels and residuals include all student prompts. Consequently, whenever

$$C_K \frac{\Gamma_{\mathrm{OPSD}}}{\Gamma_{\mathrm{SFT}}} < 1, \tag{A49}$$

the minimum-norm OPSD solution has smaller parameter displacement than the corresponding SFT solution.

Proof. For either objective, Eq. equation A44 gives

$$\|\theta - \theta_0\|^2 = \varrho^\top K^{-1} \varrho. \tag{A50}$$

Using the extremal eigenvalues of the corresponding kernels,

$$\|\theta_{\mathrm{OPSD}} - \theta_0\|^2 \leq \frac{\|\varrho_{\mathrm{OPSD}}\|^2}{\lambda_{\min}(K_{\mathrm{OPSD}})}, \tag{A51}$$

$$\|\theta_{\mathrm{SFT}} - \theta_0\|^2 \geq \frac{\|\varrho_{\mathrm{SFT}}\|^2}{\lambda_{\max}(K_{\mathrm{SFT}})}. \tag{A52}$$

Since the normalized squared residual norms correspond to Γ_{OPSD} and Γ_{SFT}, respectively, taking their ratio gives Eq. equation A47. □

E.8 Discussion and Scope of the Analysis

The analysis identifies two complementary effects underlying multi-student OPSD.

First, multiple student prompts improve coverage of the task's prompt space. For a fixed learned policy, enlarging the covered prompt set cannot increase the distance from an unseen prompt to that set, and under random prompt sampling the expected coverage error decreases with the number of student prompts.

Second, the resulting robustness guarantee depends on how far fine-tuning moves the policy from its pretrained parameters. The same parameter displacement also controls changes on unseen tasks. Under the explicit strategy-mixture and linearized-regression assumptions above, OPSD can require a smaller minimum-norm update than teacher-trajectory SFT, providing one sufficient mechanism for controlling this displacement.

These statements should not be interpreted as showing that increasing the number of students necessarily improves final task success monotonically, or that OPSD is unconditionally superior to SFT. The covered-prompt success rates, optimization dynamics, and parameter displacement can themselves change as additional prompts are introduced. The theory instead isolates the coverage benefit of multiple student prompts and the role of parameter displacement in preserving robustness.

Finally, the analysis intentionally abstracts away the other components of MOSAR. In particular, it does not model discrepancy-dependent multi-task weighting, alternating reinforcement-learning and distillation updates, separate optimizer states, or the evolution of the teacher policy during training. These components are evaluated empirically through the ablations in the main paper.

E.9 LIMITATION

While MOSAR demonstrates improved robustness to prompt variations across simulation and real-robot settings, several limitations remain. In particular, the current framework relies on the availability of a sufficiently strong teacher prompt and evaluates language robustness primarily within a predefined set of meaning-preserving prompt variations. These limitations motivate future work toward reducing assumptions on teacher-prompt quality and extending MOSAR to broader and more open-ended forms of linguistic variation.

Dependence on teacher-prompt quality. MOSAR assumes that each task admits a relatively strong prompt that can serve as a teacher. While our discrepancy-aware weighting reduces distillation when teacher performance deteriorates, the effectiveness of cross-prompt transfer may be limited when no reliably strong prompt exists. Automatically discovering or dynamically updating teacher prompts is an interesting direction for future work.

Scope of language variation. Our experiments focus on meaning-preserving prompt variations constructed from predefined perturbation categories. Although MOSAR generalizes to held-out prompts within these categories, we do not exhaustively evaluate more open-ended linguistic shifts, such as conversational instructions, multilingual inputs, or instructions with substantial syntactic and semantic ambiguity.