TELL ROBOT WHAT NOT TO DO: A NEGATION UNDERSTANDING PERSPECTIVE

Anonymous authors
Paper under double-blind review

ABSTRACT

Instruction following enables robots to perform diverse tasks specified in natural language, making it a fundamental capability for human–robot interaction. Beyond communicating desired outcomes, users also need to specify constraints on *what not to do*. We investigate how to enable vision-language-action models (VLAs) to follow negated instructions, where robots must accomplish task goals while respecting explicit exclusions. To this end, we propose NegaAlign, a parameter-efficient, plug-and-play framework that extends pretrained VLAs to follow negated instructions through image–language supervision alone. Specifically, we introduce Negation Transformation Layers into selected layers of the vision–language backbone to reshape intermediate instruction representations. Meanwhile, teacher-guided alignment mechanism is designed to align the instruction-relevant visual tokens, transferring action-relevant grounding from instructions that satisfy the negated constraint. Training phase uses supervision constructed from existing demonstrations and updates only the inserted layers, keeping all pretrained parameters frozen, including the action generator. We further introduce NegaBench, a simulation benchmark spanning 10 scenarios across five domains for systematically evaluating manipulation under negated constraints. Experiments across GR00T, π_0, and $\pi_{0.5}$ demonstrate consistent improvements in negated instruction following. With 11.6M trainable parameters, NegaAlign increases the negated-instruction success rate of $\pi_{0.5}$ from 2.60% to 88.45% on NegaBench and from 12.4% to 88.8% on real-world tasks, while retaining performance on affirmative instructions.

1 INTRODUCTION

As robots enter daily life, reliable instruction following is crucial for both human and robot safety. Vision Language Action models (VLAs) Zitkovich et al. (2023); Kim et al. (2025); Intelligence et al. (2025); Bjorck et al. (2025) follow natural language instructions and perform manipulation tasks in complex environments. Pretraining on large-scale embodied datasets O'Neill et al. (2024) enables them to complete a wide range of manipulation tasks, including pick-and-place, tool use, and object rearrangement.

However, these existing VLAs are mostly trained on affirmative instructions and struggle to follow negated instructions. This limitation can result in unsafe or undesired actions Chen et al. (2026b); Lyu et al. (2026); Zhang et al. (2025), as the robot may act on the affirmative content of the instruction without considering the

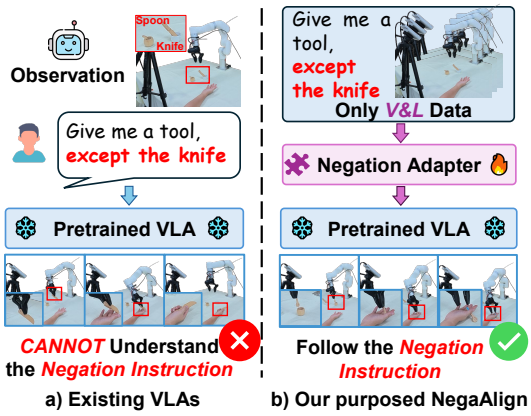

Figure 1: Existing VLAs struggle with negated instructions, risking unsafe actions. Our proposed NegaAlign enables actions under negated constraints.

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

negated constraints. *For example, given "Give me a tool, except the knife", a robot will still hand the knife to the user, which is an extremely dangerous behavior in daily life, as illustrated in Figure 1(a).* A key source of this problem lies in the VLMs backbone, whose difficulty in understanding negation affects the action sequence generation of VLAs further. Existing studies investigate negation understanding in image-text matching Dobreva & Keller (2021); Aggarwal et al. (2026), image editing Bui et al. (2025), and image-grounded object detection Kang et al. (2026); Yang et al. (2026), which could not be applied to the VLAs directly since the VLAs need to ground the negated instruction into the sequence of actions to complete the task. Recent approaches improve VLA instruction following through hierarchical decomposition and fine-grained supervision Zawalski et al. (2024); Zhu et al. (2024). Hi Robot Shi et al. (2025) uses an additional VLM to decompose instructions into subtasks, introducing extra inference computation. APT Xu et al. (2026) adopts two-stage pretraining to better preserve VLM knowledge in the action expert, but requires a dedicated multi-stage training procedure, while Fine-VLA Hu et al. (2026) constructs fine-grained instruction data to strengthen language-action alignment. However, collecting negation-specific robot demonstrations is costly and time-consuming. Hence, how to enable existing VLAs to follow negated instructions without additional negation-specific embodied data remains an unexplored problem.

To address these challenges, we propose NegaAlign, a parameter-efficient, plug-and-play framework for following negated instructions without costly negation-specific embodied data for the reliable instruction following. Our key insight is that when action expert generates the action sequences, most information are provided by the visual tokens, while the vision tokens are grounded by the instruction text representations. Specifically, we firstly construct teacher-student instruction pairs from existing affirmative-task data and hand-crafted negated instructions without additional negation-specific action data. Subsequently, we propose sophisticated Negation Transformation Layers into selected layers of the VLM backbone to revise intermediate instruction representations of negated instructions, which help to modulate the instruction representations to be more compatible with the negated constraints. Meanwhile, we apply teacher-guided alignment to instruction-relevant visual tokens. The alignment propagates the revised instruction representations to the visual tokens which is beneficial for the action generation. Training phase updates only the inserted layers using image–language supervision, while the pretrained backbone and action expert remain frozen.

To evaluate the capability of reliable instruction following, we establish NegaBench, a simulation benchmark spanning 10 scenarios across five domains for systematically evaluating manipulation under negated constraints. Experiments on the NegaBench and real-world robot demonstrate that NegaAlign significantly improves the success rate of negated instruction following across various scenarios with fewer update of parameters. The main contributions of this work are listed as follows:

- We present the first investigation of negated instruction understanding and execution in VLAs, revealing how semantics from language instructions are encoded in internal representations and gradually propagate through the model to shape action generation.
- We propose NegaAlign, a plug-and-play framework which revises instruction representations through Negation Transformation Layers and propagates negation semantics to visual representations through Teacher-Guided Visual Alignment, enabling VLAs to follow negated constraints efficiently by updating only a few number of parameters.
- We establish NegaBench, a benchmark for systematically evaluating robotic manipulation under negated instructions. Experiments in simulation and real-world robot demonstrate that NegaAlign substantially improves negated instruction following across multiple existing VLAs while preserving performance on affirmative instructions.

2 Related Work

2.1 Negation in Language, Vision, and Action

Negation remains challenging for large language models (LLMs) despite their strong general capabilities. Evaluations show that LLMs degrade on negated prompts, sometimes worsening with scale, and struggle with negated commonsense knowledge Jang et al. (2023); Wang et al. (2026). Mechanistic analyses further reveal that shortcut behaviors can disrupt how LLMs suppress or reconstruct negated meanings Zhou et al. (2026). Similar challenges arise in vision-language models (VLMs),

where negated semantics must be grounded in visual content. Pretrained VLMs degrade significantly on negated expressions Dobreva & Keller (2021); Singh et al. (2024). NeIn Bui et al. (2025) shows that image-editing models struggle to remove or exclude specified content under negative instructions, while NegBench Alhamoud et al. (2025) finds that modern VLMs struggle to distinguish retained from excluded content in negated retrieval and reasoning. Thus, visual negation requires both semantic understanding and correct grounding of exclusion relations.

VLAs such as OpenVLA Kim et al. (2025), $\pi_{0.5}$ Intelligence et al. (2025), and GR00T N series Bjorck et al. (2025) map language-conditioned visual observations to robot actions. Recent work has begun to address negative and corrective language: Hi Robot Shi et al. (2025) handles negative feedback through hierarchical reasoning, PGPS Kim et al. (2026) reduces forbidden-region grasp violations through inference-time steering, and LILAC Cui et al. (2023) enables online language correction in shared autonomy. However, these methods depend on hierarchical reasoning, external grounding, or online interaction. How negated semantics alter visual grounding within an end-to-end VLA and propagate to action generation remains underexplored.

2.2 Language Conditioning and Instruction Following in VLAs

Natural language serves as a primary mechanism for task specification and generalization in VLAs Yao et al. (2023). Nevertheless, task success alone does not establish effective linguistic grounding Dong et al. (2026): VLAs may exploit visual shortcuts Zhao et al. (2026), lose linguistic competence during downstream fine-tuning Zhang & Bisk (2026), or execute visually plausible actions under semantically contradictory instructions Zhang et al. (2026). Reliable instruction following in VLAs therefore requires the policy to remain sensitive to linguistic semantics, rather than being dominated by visual cues and degenerating into a language-agnostic imitation policy.

Effective language grounding must ultimately propagate from linguistic semantics through visual representations to action generation. Recent work strengthens this coupling by using language and execution history to reweight task-relevant visual tokens Xiao et al. (2026). Motivated by this requirement, we propose NegaAlign, a parameter-efficient, plug-and-play framework for following negated instructions without costly negation-specific robot demonstrations.

3 Method

As shown in Figure 2, we present our NegaAlign framework for understanding negation, enabling reliable instruction following. We first define the problem in Section 3.1 and introduce how to construct negation supervision from existing affirmative demonstrations in Section 3.2, then we introduce the Negation Transformation Layer for transferring semantic information of instructions in Section 3.3 and the Teacher-Guided Visual Alignment for aligning visual representations with negated instructions in Section 3.4. Finally, we present the training recipe to make the VLAs adapt to the negated instructions efficiently in Section 3.5.

3.1 Problem Formulation

We assume that a vision–language–action model (VLA) has been pretrained on large-scale robot demonstrations and has acquired the affirmative skills required by the target tasks. Our goal is to rapidly adapt it to a given set of target negated instructions and their constraint relations without additional negated action demonstrations.

Following $\pi_{0.5}$ Intelligence et al. (2025), let f_θ denotes the pretrained VLA. Given a visual observation $\mathbf{o}_t$, robot state $\mathbf{s}_t$, and language instruction ℓ, the model predicts an action sequence of horizon H:$\mathbf{a}_{t:t+H-1} = f_\theta(\mathbf{o}_t, \mathbf{s}_t, \ell)$. For a negated instruction ℓ^n, successful execution requires both completing the intended task and satisfying its negated constraints. Let τ denote an execution trajectory, and let $G(\tau, \ell^n)$ and $C(\tau, \ell^n)$ indicate task completion and constraint satisfaction, respectively. The set of valid trajectories is:

$$\mathcal{T}_{\text{valid}}(\ell^n) = \{\tau \mid G(\tau, \ell^n) = 1,\ C(\tau, \ell^n) = 1\}. \tag{1}$$

We formulate this problem as parameter-efficient post-training using existing affirmative demonstrations. Let $\mathbf{x}_0 = (\mathbf{o}_0, \mathbf{s}_0)$ denote the initial observation and state, $\mathcal{P}^n$ the distribution over initial

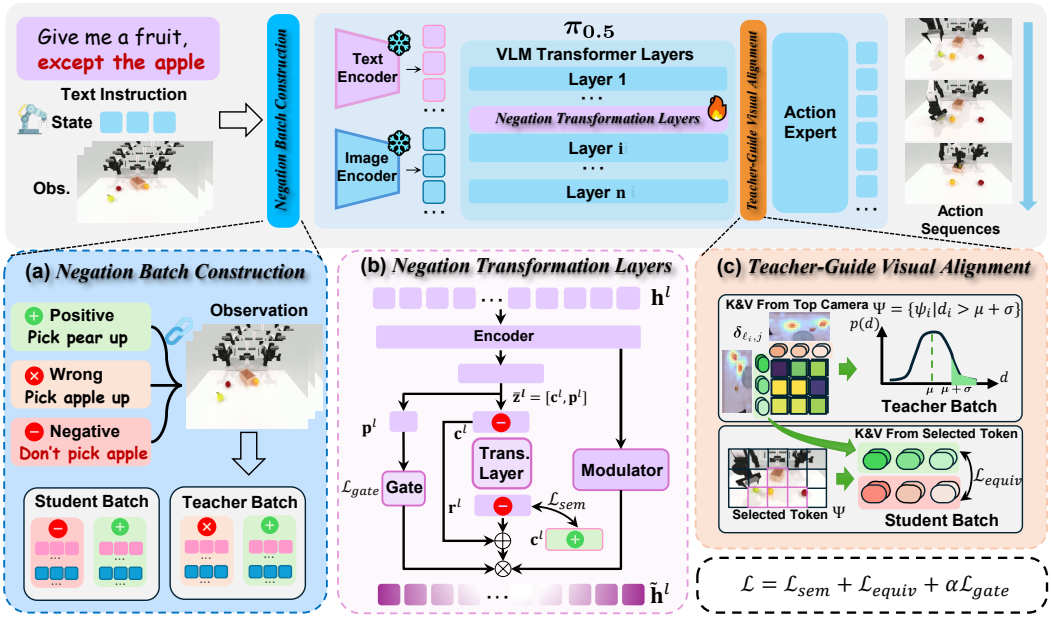

Figure 2: Overview of the purposed NegaAlign framework: (a) shows the construction of only vision and language data, (b) Negation Transformation Layer revises intermediate instruction representations, while (c) Teacher-Guided Visual Alignment transfers the revision of language representations to instruction-relevant visual tokens.

conditions and target negated instructions, and $p_{\theta,\phi}$ the trajectory distribution induced by the adapted model. Its performance is:

$$J^n(\phi) = \mathbb{E}_{\substack{(\mathbf{x}_0,\ell^n)\sim\mathcal{P}^n \\ \tau\sim p_{\theta,\phi}(\cdot|\mathbf{x}_0,\ell^n)}} \left[\mathbb{I}\left\{\tau \in \mathcal{T}_{\text{valid}}(\ell^n)\right\}\right]. \tag{2}$$

We optimize only lightweight adaptation parameters ϕ to maximize $J^n(\phi)$ while preserving affirmative instruction-following capability.

3.2 Negation Supervision Construction

Existing embodied demonstrations encode individual skills with affirmative instructions in different fine-grained levels, telling robots what to do, such as *pick up the hamburger* or *place the fries on the plate*. However, people will specify what not to do in daily life, such as *place any food except the chips on the plate*. Considering the cost of collecting negation-specific robot demonstrations, we construct negation supervision from existing affirmative demonstrations and hand-crafted negated instructions. Specifically, we expose the latent structure of negation constraint as relational supervision, associating each negated instruction with a set of admissible affirmative interpretations and an explicitly excluded one, as illustrated in Figure 2(a). The former defines where the negated semantics should be routed, while the latter provides the contrast needed to identify what should be suppressed. We formalize this relation as $\mathcal{R} = (\ell^n, \mathcal{A}, \ell^e)$, where ℓ^n is a negated instruction, $\mathcal{A} = \{\ell^a_1, \ldots, \ell^a_M\}$ is the set of affirmative instructions satisfying its constraint, and ℓ^e describes the excluded behavior. For example, given *place any foo d except the chips on the plate*, the instructions for the hamburger and fries belong to $\mathcal{A}$, while the chips instruction is ℓ^e.

During the stage of negation adaption, we first sample a tuple, then one satisfying instruction $\ell^a \in \mathcal{A}$, and finally an observation–state pair $(\mathbf{o}, \mathbf{s})$ from its affirmative trajectory. Then we evaluate four language conditions with keeping the scene fixed:

$$\mathcal{I}^a_S = (\mathbf{o}, \mathbf{s}, \ell^a), \mathcal{I}^n_S = (\mathbf{o}, \mathbf{s}, \ell^n), \mathcal{I}^a_T = (\mathbf{o}, \mathbf{s}, \ell^a), \mathcal{I}^e_T = (\mathbf{o}, \mathbf{s}, \ell^e), \tag{3}$$

where S indicates the student branches and T indicates the teacher branches. The student branches use $f_{\theta,\phi}$, whereas the teacher branches use the same frozen f_θ without NTLs. Their outputs are denoted by $\mathcal{O}^a_S$, $\mathcal{O}^n_S$, $\mathcal{O}^a_T$, and $\mathcal{O}^e_T$. Sharing the observation and state across branches isolates the language contrast and turns existing affirmative data into negation supervision.

3.3 NEGATION TRANSFORMATION LAYER

Due to the action generator relies on the attention over visual and language representations, it is difficult for VLAs to learn negation semantics solely from the trajectory that satisfies the instruction. To this end, we purpose Negation Transformation Layers (NTLs) to revise the intermediate instruction representations of negated instructions to distinguish them from affirmative instructions, thereby enabling the VLA to understand negation semantics. Specifically, we insert an NTL before each selected VLM layer l. The NTL first projects the instruction tokens $\mathbf{h}^l$ into a compact space and pools them into content and polarity embeddings $\mathbf{c}^l$ and $\mathbf{p}^l$ respectively. A lightweight Trans. Layer transforms the $\mathbf{c}^l$ into a satisfied affirmative destination $\mathbf{r}^l$, while the polarity branch produces a gate $g^l = \mathrm{Gate}(\mathbf{p}^l)$, with a^l denoting its pre-sigmoid logit used in the gate loss. To retain token-level distinctions, the resulting semantic shift is modulated by each token representation to obtain $\Delta\mathbf{h}_t^l$:

$$\bar{\mathbf{z}}^{\,l} = [\mathbf{c}^l; \mathbf{p}^l] = \mathrm{Pool}\big(\mathrm{Encode}(\mathbf{h}^l)\big)\,, \mathbf{r}^l = \mathrm{TransLayer}(\mathbf{c}^l), g^l = \mathrm{Gate}(\mathbf{p}^l), \tag{4}$$

$$\Delta\mathbf{h}_t^l = \mathrm{Modulator}\big(\mathbf{r}^l - \mathbf{c}^l;\, \mathbf{h}_t^l\big)\,, \qquad \tilde{\mathbf{h}}_t^l = \mathbf{h}_t^l + m_t g^l \Delta\mathbf{h}_t^l, \tag{5}$$

where m_t marks instruction tokens. Meanwhile, we design an alignment loss $\mathcal{L}_{\mathrm{sem}}^l$ to route the revised content toward its paired affirmative representation, and a gate loss $\mathcal{L}_{\mathrm{gate}}^l$ to distinguish affirmative from negated inputs:

$$\mathcal{L}_{\mathrm{sem}}^l = \frac{1}{B}\sum_{b=1}^{B}\Big[1 - \cos\Big(\mathbf{r}_b^{n,l}, \mathrm{sg}(\mathbf{c}_b^{a,l})\Big)\Big]\,, \mathcal{L}_{\mathrm{gate}}^l = \frac{1}{2B}\sum_{b=1}^{2B}\mathrm{BCEWithLogits}(a_b^l, y_b), \tag{6}$$

where $\mathbf{r}_b^{n,l}$ is the negated representation, $\mathbf{c}_b^{a,l}$ is the affirmative representation, a_b^l is the pre-sigmoid logit, and y_b is the ground truth. Since the update is restricted by m_t, image, state, and non-instruction prompt tokens remain unchanged. NTL thus revises how language selects an acquired skill while preserving the frozen VLA that executes it.

3.4 TEACHER-GUIDED VISUAL ALIGNMENT

Related work on negation understanding of VLMs Kang et al. (2026) focus on how to enhance the negation semantics in the language space, however, VLAs not only need to understand the negation semantics, but also need to attend to the visual representations that are used for action generation. Therefore, revising the language representations alone by Negation Transformation Layers (NTLs) is insufficient to reliably alter the generated actions and the language representation revision should also be propagated to the instruction-relevant visual tokens. To achieve this goal, we purpose Teacher-Guided Visual Alignment (TGVA), which guides the revised negation semantics from the language space into the visual representations used for action generation. After each NTLs, we extract the visual keys and values used by the following attention block and denote their mean cosine distance $d_{b,i}^l$ at token i by $D_i^l(\cdot,\cdot)$. For each batch sample and layers, TGVA computes:

$$d_{b,i}^l = D_i^l(\mathcal{O}_{T,b}^a, \mathcal{O}_{T,b}^e), \qquad w_{b,i}^l = \frac{\mathbb{I}[i \in \Psi_b^l](d_{b,i}^l + \epsilon)}{\sum_j \mathbb{I}[j \in \Psi_b^l](d_{b,j}^l + \epsilon)}, \tag{7}$$

where Ψ_b^l is the set of selected visual-token indices, determined by an adaptive threshold with a top-ranked fallback. Then, we minimize the weighted distance between the negated student and the satisfying teacher to align the instruction-relevant visual tokens:

$$\mathcal{L}_{\mathrm{equiv}}^l = \frac{1}{B}\sum_{b=1}^{B}\sum_{i\in\Psi_b^l} w_{b,i}^l\, D_i^l\big(\mathcal{O}_{S,b}^n, \mathrm{sg}(\mathcal{O}_{T,b}^a)\big)\,, \tag{8}$$

where B is the batch size. We apply this alignment to a coordinate-consistent top-camera token and stop gradients through the teacher-derived weights for maintenance of the VLM knowledge.

3.5 TRAINING RECIPE

Nowadays, the usual training recipe of VLAs is to first pretrain with large-scale embodied demonstration data, then fine-tune it on the task- or scenario-specific demonstration data for adaptation and

generalization. Accordingly, NegaAlign follows a two-stage training recipe after the initial pretraining. Stage 1 fine-tunes the pretrained VLA on affirmative robot demonstrations, which is consistent with the standard fine-tuning process. Then during Stage 2, the fine-tuned VLA is frozen, **Negation Transformation Layers (NTLs)** are inserted, and they are trained with **Teacher-Guided Visual Alignment (TGVA)** using vision-language data constructed from the existing affirmative demonstrations. This stage requires neither negated action trajectories nor additional robot demonstrations. The training objective of Stage 2 is defined as follows:

$$\mathcal{L}_{\text{total}} = \frac{1}{|\Omega|} \sum_{l \in \Omega} \left(\mathcal{L}^l_{\text{equiv}} + \mathcal{L}^l_{\text{sem}} + \alpha \mathcal{L}^l_{\text{gate}} \right), \tag{9}$$

where Ω is the set of layers with NTLs, and α is a hyperparameter to balance the three objectives. The objective is optimized to revise the intermediate instruction representations of negated instructions and propagate the negation semantics to the instruction-relevant visual tokens, thereby enabling the VLA to understand negation semantics and follow negated instructions.

3.6 Negation Instruction Following Benchmark Construction

To systematically evaluate whether robots can follow negated constraints during manipulation, we construct NegaBench, a controlled and fine-grained benchmark built upon the RoboTwin simulation environment Chen et al. (2026a). NegaBench comprises 10 manipulation scenarios spanning five representative categories of negation: object exclusion, attribute exclusion, relative comparison, goal-location exclusion, and combinations of multiple constraints. These categories progress from selecting among individual objects to executing complex behaviors that must satisfy several exclusions simultaneously. Each scenario provides semantically paired affirmative and negated instructions together with explicit constraint-aware success criteria: an execution is considered successful only if the robot completes the shared manipulation goal without violating any excluded object, attribute, or outcome specified by the instruction. By evaluating paired instructions under the same scene, manipulation skills, and success conditions, NegaBench controls for basic task competence and scene difficulty, directly measures whether a negated constraint changes the behavioral choice of robot, and systematically exposes the limitations of existing VLA models in constraint-aware instruction following. Complete benchmark construction and instruction relations are provided in Appendix B, and the evaluation protocol is detailed in Appendix C.1.

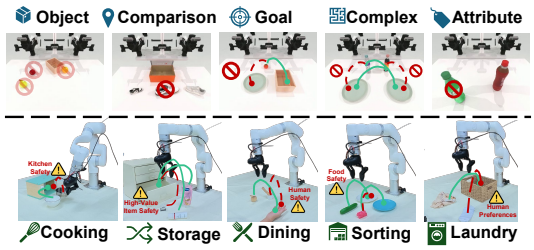

Figure 3: Overview of the NegaBench benchmark. The **green trajectory** indicates a successful execution of the negated instruction, while the **red trajectory** indicates that the robot failed to meet the negated constraints.

4 Experiments

4.1 Experimental Setup and Evaluation Metrics

We evaluate NegaAlign on three VLA models, GR00T, π_0, and $\pi_{0.5}$, under three training settings: *Pos.*, fine-tune the pretrained model with only affirmative instructions; *Pos.+Neg.*, joint training on both affirmative and negated trajectories to enable the capability of negation understanding and prevent the catastrophic forgetting of affirmative instructions; and *Pos.+Ours*, which applies NegaAlign to an affirmative-trained model. We additionally evaluate APT Xu et al. (2026) and Qwen-$\pi_{0.5}$ as baselines. Qwen-$\pi_{0.5}$ uses Qwen3-VL-2B-Instruct Bai et al. (2025) to rewrite negated instruction base on top camera observation to an affirmative command for an affirmative-trained $\pi_{0.5}$ policy. Further implementation details for these baselines are given in Appendix A.1. The training detail is provided in Appendix A. We report success rates separately for affirmative instructions (Pos. SR) and negated instructions (Neg. SR). An episode is considered successful only if the robot completes the intended manipulation while satisfying all negated constraints.

Table 1: Comparison of our method with baselines on NegaBench. The best results are highlighted in **bold**.

Model	Training	Object		Attribute		Comparison		Goal		Complex		Avg.		Trainable Param.	Fine-tuning Time(hours)
		Pos.	Neg.	Pos.	Neg.	Pos.	Neg.	Pos.	Neg.	Pos.	Neg.	Pos.	Neg.		
GR00T	Pos. FT	60.09	1.25	63.75	0.00	70.00	4.00	75.00	0.00	86.63	0.00	71.09	1.05	–	–
π_0	Pos. FT	30.75	14.75	58.50	44.00	75.50	18.00	96.25	0.00	80.00	0.25	68.20	15.40	–	–
$\pi_{0.5}$	Pos. FT	91.59	3.50	88.00	0.50	89.00	7.00	96.00	0.00	99.25	2.00	92.77	2.60	–	–
APT	Pos. FT	58.92	6.00	75.50	13.50	73.00	5.00	84.75	13.50	85.13	3.00	75.46	8.20	–	–
Qwen-$\pi_{0.5}$	Pos. FT	49.50	52.00	84.25	94.00	56.75	25.50	93.50	60.50	73.88	46.50	71.58	55.70	–	–
GR00T	Pos. & Neg. FT	60.92	44.00	60.00	65.50	71.50	65.00	91.25	77.50	93.63	92.25	75.46	68.85	567.6M/3.2B	87.6
	w/ Ours	59.84	24.75	68.75	68.00	72.75	61.50	76.00	87.50	87.13	47.00	72.89	**57.75**	**4.35M/3.2B**	**2.7**
π_0	Pos. & Neg. FT	29.92	26.75	74.75	83.00	77.75	72.50	98.50	94.00	77.50	80.00	71.68	71.25	468.0M/3.6B	251.4
	w/ Ours	29.42	24.75	59.50	59.50	76.50	59.50	96.00	96.50	80.63	76.25	68.41	**63.30**	**11.6M/3.6B**	**3.3**
$\pi_{0.5}$	Pos. & Neg. FT	85.67	77.75	90.50	93.00	85.75	81.00	96.75	92.50	98.38	98.75	91.41	88.60	467.0M/3.7B	179.9
	w/ Ours	87.67	77.50	91.75	94.50	89.50	78.00	94.25	93.50	99.38	98.75	92.51	**88.45**	**11.6M/3.7B**	**4.6**

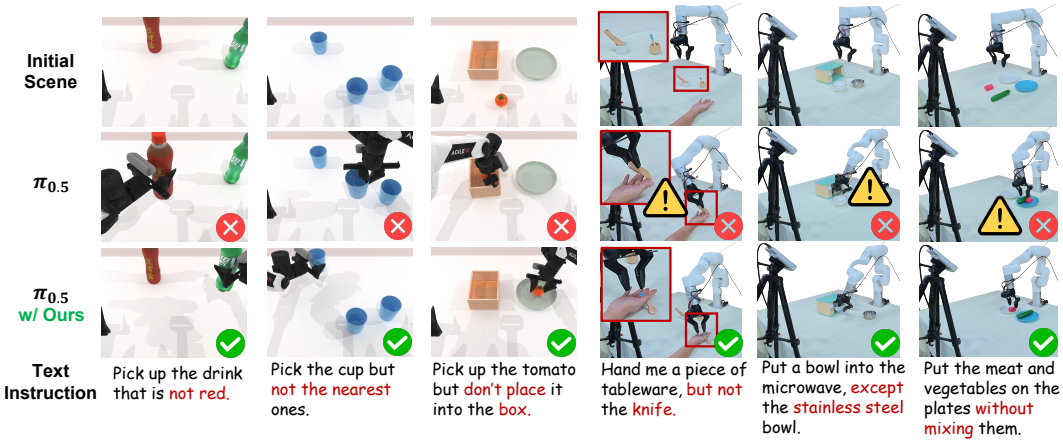

Figure 4: The visualization of the simulation and real-world experiments. The robot is required to follow negated instructions in various scenarios.

4.2 Comparison Results on NegaBench

As shown in Table 1, affirmative-only training yields strong affirmative performance but generalizes poorly to negated instructions: GR00T, π_0, and $\pi_{0.5}$ achieve average negated success rates of 1.05%, 15.40%, and 2.60%, respectively. Explicit negated-trajectory training raises these rates to 68.85%, 71.25%, and 88.60%. This contrast motivates NegaAlign to ground negated constraints in existing manipulation skills rather than relearn the skills themselves.

APT reaches 75.46% affirmative success but only 8.20% negated success with affirmative-only training, indicating that preserving the capabilities of VLM in a VLA is insufficient to solve the negated instruction follow problem. Qwen-$\pi_{0.5}$ directly rewrites negated instructions into affirmative commands, yet achieves only 55.70% negated success and 71.58% affirmative success. This approach adds an upstream VLM inference step at every action-chunk request, increasing inference latency and making the action depend on the VLM visual grounding and instruction rewrite capabilities. Its substantially lower negated success than 88.45% of our purposed NegaAlign suggests that direct instruction rewriting is not an effective solution to negated instruction following in this setting.

Without using any negated action trajectories, NegaAlign reaches 57.75%, 63.30%, and 88.45% negated success on GR00T, π_0, and $\pi_{0.5}$, gains of 56.70, 47.90, and 85.85 percentage points over their corresponding *Pos.* baselines. Affirmative success is 72.89%, 68.41%, and 92.51%, respectively, compared with 71.09%, 68.20%, and 92.77% for *Pos. FT*. Averaged across the three models, it achieves 69.83% negated success, reaching 91.6% of the performance obtained through direct negated-trajectory training. On $\pi_{0.5}$, NegaAlign achieves 88.45%, nearly matching the 88.60% of

Pos.& Neg. These results demonstrate that semantic supervision constructed from existing affirmative data enables effective negated instruction following without additional robot demonstrations.

4.3 Ablation Studies and Mechanism Analysis

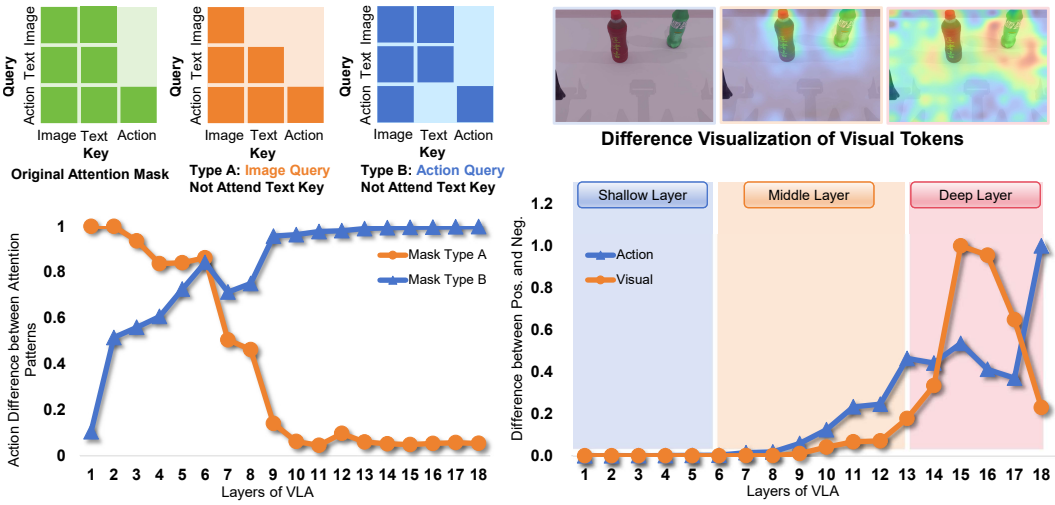

Figure 5: Left: Normalized instruction-induced action separation under attention masking. Right: Layer-wise differences between satisfying and excluded instructions.

How do language semantics influence action generation? We first examine how language information affects the action generator. Figure 5 in left reports the action separation between satisfying and excluded instructions under two attention masks, normalized by the separation under full attention, where 1 indicates full preservation. When action queries cannot attend to text keys, the separation remains near 1 in deeper layers, showing that direct language access has little effect on the final actions. In contrast, blocking image queries from text keys causes the separation to approach zero, leaving the policy unable to distinguish the two instructions at the action level. Language semantics therefore primarily condition visual representations, which subsequently guide the action expert to generate actions. This observation of visual mediation motivates the design of Teacher-Guide Visual Alignment (TGVA).

Table 2: Ablation study on the effect of TGVA and different layer configurations of NTL.

Setting	Layer	TGVA	NTL	Success Rate (%)	
				Pos.	Neg.
Baseline	–	✗	✗	92.77 (+0.26)	2.60 (-85.85)
Exp. 1	Shallow	✓	✓	92.26 (-0.25)	85.10 (-3.35)
Exp. 2	Deep	✓	✓	91.92 (-0.59)	62.20 (-26.25)
Exp. 3	Middle	✗	✓	91.27 (-1.24)	21.35 (-67.10)
Exp. 4	Middle	✓	✗	40.52 (-51.99)	86.65 (-1.80)
Ours	Middle	✓	✓	**92.51**	**88.45**

Why is Teacher-Guide Visual Alignment(TGVA) effective? Based on the above analysis, we visualize the layer-wise differences of visual tokens between satisfying and excluded instructions in Figure 5 right. The visual differences between instructions are negligible in shallow layers, emerge and concentrate on instruction-relevant objects in middle layers, and spread toward background and irrelevant regions in deeper layers. Therefore, TGVA is designed to align the negated branch with the satisfying teacher in middle layers, where semantic visual grounding has formed but has not yet become diffuse. Removing TGVA reduces negated success from 88.45% to 21.35%, while affirmative success remains relatively high at 91.27%, as Table 2 shows. It indicates that NTL could preserve the original manipulation capability but cannot reliably propagate the revised semantics into action generation without TGVA, confirming the necessity of visual alignment and the effectiveness of the visual mediation mechanism identified in Figure 5 left.

What is the optimal intervention depth for Negation Transformation Layers(NTL)? As shown in Table 2, inserting NTLs into middle layers of the VLM backbone yields the best perfor-

mance in both affirmative and negated instruction following. Meanwhile, moving NTL to shallow or deep layers reduces negated success to 85.10% and 62.20%, respectively, while affirmative performance remains close to 92%. It indicates that the instruction-dependent visual representations have different characteristics in different layers, and the middle layers provide the best intervention point for NTLs, after semantic grounding emerges but before it becomes diffuse, which is consistent with the analysis in Figure 5 right.

Why is the NTL design necessary? As shown in Table 2, retaining TGVA while removing NTL yields 86.65% negated success but collapses affirmative performance from 92.51% to 40.52%. In this variant, the proposed NTL is replaced by a trainable ungated revision layer at the same insertion positions, while all pretrained VLA parameters remain frozen. The replacement applies revision to both affirmative and negated instructions and is optimized using the semantic alignment and TGVA objectives. The decrease in performance highlights the importance of NTL in maintaining the performance of affirmative instructions. Specifically, visual alignment can induce constraint-satisfying behavior, but without controlled semantic transformation it also disrupts affirmative instructions. NTL separates semantic content from polarity and gates revision only when needed, enabling the complete model to achieve 88.45% negated success while retaining 92.51% affirmative success. Thus, NTL prevents this revision from compromising the original capabilities of policy while TGVA propagates revised semantics through action-relevant visual tokens.

4.4 Real-World Robot Experiments

We further evaluate NegaAlign on a physical xArm robot in five safety-critical household scenarios, as illustrated in Figure 3. A detailed description of the real-world task setting is provided in Appendix C.2. Violating these negated constraints can cause property loss or damage, and potentially serious injury, making correct exclusion-aware behavior essential. Each negated instruction is evaluated over 50 trials. We compare the affirmative-only $\pi_{0.5}$ policy with the same policy adapted by NegaAlign, as shown in Table 3.

Table 3: Real-world success counts on five xArm manipulation tasks. The best results are highlighted in **bold**.

Task	$\pi_{0.5}$		$\pi_{0.5}$ w/ Ours	
	Pos.	Neg.	Pos.	Neg.
Sorting	46/50	9/50	42/50	**40/50**
Laundry	46/50	4/50	49/50	**47/50**
Storage	45/50	1/50	44/50	**50/50**
Cooking	49/50	0/50	48/50	**48/50**
Dining	43/50	17/50	44/50	**37/50**
Avg. SR (%)	91.6	12.4	90.8	**88.8**
Avg. Inference Latency (ms)	**481.6**		494.3	

The affirmative-only policy achieves a high affirmative success rate of 91.6%, but succeeds in only 12.4% of negated trials. This large discrepancy shows that robust manipulation alone does not enable a VLA to reliably respect exclusion constraints. With our method, the average negated success rate rises to 88.8%, a gain of 76.4 percentage points, while affirmative performance remains essentially unchanged at 90.8%. The consistent improvement across diverse household scenarios demonstrates that our approach effectively transfers negation understanding into real-world robot behavior without compromising the original manipulation capabilities of policy.

5 Conclusion

We present NegaAlign, a parameter-efficient framework that combines Negation Transformation Layers and Teacher-Guided Visual Alignment to enable negated instruction following in pretrained VLAs without additional negated action trajectories. We also introduce NegaBench to evaluate this capability in robotic manipulation. Simulation and real-world experiments demonstrate substantial improvements in negated instruction following while preserving affirmative performance.

The limitations of this work include the reliance on existing pretrained VLAs, which may not cover all possible negated scenarios, and the potential for performance degradation in highly complex or ambiguous negated instructions that leads our direction of future work.

AI USE STATEMENT

In this work, we used generative AI tools to assist with language editing, figure presentation, checking mathematical notation, and checking the consistency between the manuscript and existing code. We did not use generative AI to generate research data, formulate or prove mathematical claims, design experiments, or analyze or interpret experimental results. The authors reviewed all AI-assisted text and suggestions and verified their accuracy against the original code, experimental records, and relevant literature. The authors take full responsibility for the final text, claims, and research artifacts in this work.

REFERENCES

Bhuvan Aggarwal, Amit More, Mudit Soni, and Srinivasa Divakar Bhat. Seeing what's not there: negation understanding needs more than training. In *International Conference on Learning Representations*, volume 2026, pp. 69200–69217, 2026.

Kumail Alhamoud, Shaden Alshammari, Yonglong Tian, Guohao Li, Philip HS Torr, Yoon Kim, and Marzyeh Ghassemi. Vision-language models do not understand negation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 29612–29622. IEEE, 2025.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL https://arxiv.org/abs/2511.21631.

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

Nhat-Tan Bui, Dinh-Hieu Hoang, Quoc-Huy Trinh, Minh-Triet Tran, Truong Nguyen, and Susan Gauch. Nein: Telling what you don't want. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 3132–3140, June 2025.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=itonej9GIV.

Zixing Chen, Yifeng Gao, Li Wang, Yunhan Zhao, Yi Liu, Jiayu Li, Xiang Zheng, Zuxuan Wu, Cong Wang, Xingjun Ma, et al. Hazardarena: Evaluating semantic safety in vision-language-action models. *arXiv preprint arXiv:2604.12447*, 2026b.

Yuchen Cui, Siddharth Karamcheti, Raj Palleti, Nidhya Shivakumar, Percy Liang, and Dorsa Sadigh. No, to the right: Online language corrections for robotic manipulation via shared autonomy. In *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*, pp. 93–101, 2023.

Radina Dobreva and Frank Keller. Investigating negation in pre-trained vision-and-language models. In *Proceedings of the fourth blackboxnlp workshop on analyzing and interpreting neural networks for nlp*, pp. 350–362, 2021.

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 Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 44615–44629, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.2066. URL https://aclanthology.org/2026.acl-long.2066/.

Xintong Hu, Xuhong Huang, Jinyu Zhang, Yutong Yao, Yuchong Sun, Qiuyue Wang, Mingsheng Li, Sicheng Xie, Yitao Liu, Junhao Chen, Yixuan Chen, Yingming Zheng, Shuai Bai, and Tao Yu. Finevla: Fine-grained instruction alignment for steerable vision-language-action policies, 2026. URL https://arxiv.org/abs/2605.27284.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

Joel Jang, Seonghyeon Ye, and Minjoon Seo. Can large language models truly understand prompts? a case study with negated prompts. In Alon Albalak, Chunting Zhou, Colin Raffel, Deepak Ramachandran, Sebastian Ruder, and Xuezhe Ma (eds.), *Proceedings of The 1st Transfer Learning for Natural Language Processing Workshop*, volume 203 of *Proceedings of Machine Learning Research*, pp. 52–62. PMLR, 03 Dec 2023. URL https://proceedings.mlr.press/v203/jang23a.html.

Inha Kang, Youngsun Lim, Seonho Lee, Jiho Choi, Junsuk Choe, and Hyunjung Shim. What "not" to detect: Negation-aware vlms via structured reasoning and token merging. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.13232.

Geonho Kim, SooGon Kim, and Jongmin Lee. Potential-guided particle steering for negation-constrained dexterous grasping. *arXiv preprint arXiv:2609.00555*, 2026.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Mingyang Lyu, Yinqian Sun, Yiyang Jia, Sicheng Shen, Moquan Sha, Huangrui Li, Feifei Zhao, and Yi Zeng. Foresightsafety-vla: A unified diagnostic safety benchmark for vision-language-action models. *arXiv preprint arXiv:2606.27079*, 2026.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guanzhi Wang, Hao Su, Hao-Shu

Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi Jim Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick Tree Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, and Zipeng Lin. Open x-embodiment: Robotic learning datasets and rt-x models : Open x-embodiment collaboration0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903, 2024. doi: 10.1109/ICRA57147.2024.10611477.

Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. Hi robot: Open-ended instruction following with hierarchical vision-language-action models, 2025. URL https://arxiv.org/abs/2502.19417.

Jaisidh Singh, Ishaan Shrivastava, Mayank Vatsa, Richa Singh, and Aparna Bharati. Learn" no" to say" yes" better: Improving vision-language models via negations. *arXiv preprint arXiv:2403.20312*, 2024.

Zijie Wang, MohammadHossein Rezaei, Farzana Rashid, and Eduardo Blanco. Commonsense knowledge with negation: A resource to enhance negation understanding. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 11915–11934, 2026.

Lei Xiao, Jifeng Li, Juntao Gao, Feiyang Ye, Yan Jin, Jingjing Qian, Jing Zhang, Yong Wu, and Xiaoyuan Yu. Ava-vla: improving vision-language-action models with active visual attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13453–13463, 2026.

Kechun Xu, Zhenjie Zhu, Anzhe Chen, Rong Xiong, and Yue Wang. Apt: Action expert pretraining improves instruction generalization of vision-language-action policies, 2026. URL https://arxiv.org/abs/2606.12366.

Zesheng Yang, Xi Jiang, Bingzhang Hu, Weili Guan, Runmin Cong, Guo-Jun Qi, and Feng Zheng. Mastering negation: Boosting grounding models via grouped opposition-based learning. *arXiv preprint arXiv:2603.12606*, 2026.

Xiangtong Yao, Hongkuan Zhou, Oier Mees, Yuan Meng, Ted Xiao, Yonatan Bisk, Jean Oh, Edward Johns, Mohit Shridhar, Dhruv Shah, et al. Bridging language and action: A survey of language-conditioned robot manipulation. *The International Journal of Robotics Research*, pp. 02783649261468360, 2023.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=S70MgnIA0v.

Borong Zhang, Yuhao Zhang, Jiaming Ji, Yingshan Lei, Josef Dai, Yuanpei Chen, and Yaodong Yang. SafeVLA: Towards safety alignment of vision-language-action model via constrained learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=dt940loCBT.

Haochen Zhang and Yonatan Bisk. Flatness preserves instruction following in vision-language-action models. *arXiv preprint arXiv:2606.23641*, 2026.

Ninghao Zhang, Bin Zhu, Shijie Zhou, and Jingjing Chen. Restoring linguistic grounding in vla models via train-free attention recalibration. *arXiv preprint arXiv:2603.06001*, 2026.

Mengao Zhao, Ziang Li, Chaodong Huang, Mengchen Ma, Haoyi Jiang, Yiwei Jin, Xinjie Wang, Yun Du, Xuewu Lin, Taojun Ding, et al. Instructmove: A text-indispensable benchmark for instruction-following manipulation. *arXiv preprint arXiv:2608.22990*, 2026.

Zhejian Zhou, Tianyi Zhou, Robin Jia, and Jonathan May. How language models process negation. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=8DLW34pkNi.

Minjie Zhu, Yichen Zhu, Jinming Li, Junjie Wen, Zhiyuan Xu, Zhengping Che, Chaomin Shen, Yaxin Peng, Dong Liu, Feifei Feng, and Jian Tang. Language-conditioned robotic manipulation with fast and slow thinking. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4333–4339, 2024. doi: 10.1109/ICRA57147.2024.10611525.

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 *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=XMQgwiJ7KSX.

Table 4: Examples of Qwen-$\pi_{0.5}$ instruction rewrites on NegaBench. The generated commands are shown as produced, including cases where a negated constraint is retained or omitted.

Type	Original instruction	Qwen-generated command
Attribute	pick up the drink that is not red	pick up the green drink
	pick the cup that is not full	pick the blue cup
Comparison	pick the cup but not the nearest one	pick the middle cup
	put one shoe in the shoebox, don't choose the biggest one	put the white shoe in the shoebox
Complex	pick up the bottle but not the water and place it on the plate but not the left one	pick up the bottle and place it on the right plate
	pick up the bottle but not the soy sauce and place it on the plate but not the right one	pick up the water bottle and place it on the left plate
	pick up a dumbbell, not the red one and not the biggest one	pick up a dumbbell, not the red one and not the biggest one
	pick up a dumbbell, not the black one and not the smallest one	pick up a red dumbbell
Goal	pick up the tomato but don't place it into the box	pick up the tomato and place it on the table
	pick up the bread but don't place it on the table	pick up the bread and place it on the tray
Object	pick up one fruit but not apple into the box	pick up the orange into the box
	place any food except the chips on the plate	place any food on the plate
	place any food except the fries on the plate	place any food on the plate

A Implementation and Training Details

A.1 Backbones, Trainable Modules, and Branch Execution

We instantiate NegaAlign from an affirmative-only checkpoint of each VLA. During negation post-training, all parameters inherited from the checkpoint are frozen, including the visual encoder, the vision–language backbone, the action expert or action head, and any adapters learned during affirmative training. The only trainable parameters are the inserted Negation Transformation Layers (NTLs). For π_0 and $\pi_{0.5}$, we insert NTLs before VLM layers 6–13 (one-based indexing). For GR00T, whose vision–language interface contains four Vision–Language Self–Attention (VLSA) blocks, we insert NTLs before the first three blocks and use the following block to obtain the Key and Value supervision signal. These configurations introduce 11.6M trainable parameters for π_0 and $\pi_{0.5}$ and 4.35M for GR00T.

The additional APT baseline uses its affirmative-trained checkpoint and executes dual-arm end-effector pose and gripper actions. Qwen-$\pi_{0.5}$ pairs the affirmative-trained $\pi_{0.5}$ checkpoint with an unmodified Qwen3-VL-2B-Instruct model. At each action-chunk request, Qwen receives the current top-camera image and original instruction and generates a command for $\pi_{0.5}$; both affirmative and negated instructions pass through this rewriting step. Table 4 shows examples of the generated commands.

The configured batch size B counts independently sampled affirmative scene–instruction pairs. Each pair is expanded into four logical language-conditioned branches over the same observation and robot state: the affirmative student S^a, negated student S^n, satisfying teacher T^a, and excluded teacher T^e. Consequently, a training step with $B = 4$ evaluates 16 branch instances, but this should not be interpreted as either four model replicas or an optimization batch size of 16. All branches share one set of VLA weights. The two teachers are stateless, no-gradient forward branches of the same frozen checkpoint with the NTLs bypassed; they are neither separately trained networks nor exponential-moving-average copies. The number of independent scenes contributing to the stochastic gradient remains four.

For π_0 and $\pi_{0.5}$, the two student branches are concatenated and executed as one gradient-carrying batch of size $2B$. The satisfying and excluded teachers are then evaluated as two sequential no-

gradient batches of size B. Thus, when $B = 4$, the largest forward that retains activations for backpropagation contains eight samples, while each teacher forward contains four. Frozen image encoding and prompt embedding are also evaluated without constructing an autograd graph. This scheduling provides the four required representations without storing the activations of all $4B$ branches simultaneously.

GR00T further reuses computation across branches. Its frozen Qwen backbone jointly processes the satisfying, negated, and excluded language variants as a batch of size $3B$; the satisfying hidden states are shared by T^a and S^a. The subsequent VLSA computation consists of two no-gradient teacher branches of size B and one gradient-carrying student branch of size $2B$. Images are preprocessed once per sampled scene and repeated only after preprocessing. Hence, with $B = 4$, Qwen receives 12 language variants, while the downstream VLSA blocks realize the same 16 logical branch instances required by the objective.

Post-training executes only the visual–language prefix and the selected VLM blocks required to construct the intermediate supervision. It does not execute the action expert and does not consume action targets. This substantially reduces memory relative to running four complete VLAs, although compute still scales with the teacher and student branch forwards described above. At inference time, all teacher branches are discarded. The input instruction is revised by the learned NTLs at the same insertion sites, after which the frozen action-generation procedure of the original VLA is executed unchanged.

A.2 NTL Architecture

Each selected VLM layer has an independently parameterized NTL; parameters are not shared across insertion sites. Let $\mathbf{h}_t^l \in \mathbb{R}^d$ be hidden state of token t at site l, and let $m_t \in \{0, 1\}$ indicate whether it belongs to the natural-language instruction. The tokenizer-derived mask excludes image and state tokens, special prompt tokens, and padding. The NTL first normalizes and compresses the selected tokens to a 256-dimensional bottleneck, then pools them into an instruction-level representation:

$$\mathbf{z}_t^l = W_{\text{down}}^l \,\text{LN}(\mathbf{h}_t^l), \qquad \bar{\mathbf{z}}^l = \frac{\sum_t m_t \mathbf{z}_t^l}{\max(1, \sum_t m_t)}, \qquad [\mathbf{c}^l; \mathbf{p}^l] = \bar{\mathbf{z}}^l, \tag{10}$$

where $\mathbf{c}^l \in \mathbb{R}^{224}$ is the content code and $\mathbf{p}^l \in \mathbb{R}^{32}$ is the polarity code. The split reserves most of the bottleneck for task content while exposing a compact subspace from which the module can decide whether semantic revision is required.

The transformation layer is a $224 \to 896 \to 224$ MLP, while a separate linear predictor maps the polarity code to a scalar gate:

$$\mathbf{r}^l = W_2^l \,\text{GELU}(W_1^l \mathbf{c}^l), \qquad a^l = W_g^l \mathbf{p}^l, \qquad g^l = \sigma(a^l). \tag{11}$$

Here, $\mathbf{r}^l$ is the satisfying affirmative destination predicted from the instruction content. A single global displacement $\mathbf{r}^l - \mathbf{c}^l$ would update all words identically, so we additionally compute a token-specific modulation vector $\mathbf{u}_t^l = \tanh(W_m^l \mathbf{z}_t^l) \in \mathbb{R}^{224}$. The complete update is

$$\Delta\mathbf{h}_t^l = W_{\text{up}}^l\big(\mathbf{u}_t^l \odot (\mathbf{r}^l - \mathbf{c}^l)\big), \qquad \tilde{\mathbf{h}}_t^l = \mathbf{h}_t^l + m_t g^l \Delta\mathbf{h}_t^l. \tag{12}$$

The $224 \to d$ up-projection returns the residual to the host-model width. The token modulator allows object, attribute, relation, and action words to respond differently to the same instruction-level revision. The mask scatters the revised states back only to their original instruction positions; all other tokens remain bitwise unchanged at the NTL output. Gate supervision drives g^l toward zero for affirmative inputs and toward one for negated inputs, so the same module can preserve the original affirmative pathway while activating the learned revision when needed. Each NTL contributes approximately 1.45M parameters for the backbones used in our experiments, yielding 11.6M parameters over eight sites for π_0 and $\pi_{0.5}$ and 4.35M over three sites for GR00T.

A.3 Teacher-Guided Visual Alignment

TGVA uses two frozen forward branches to determine both *where* negation changes visual grounding and *which* affirmative grounding the negated student should approach. For every sampled scene, the satisfying teacher $\mathcal{O}_T^a$ receives the selected admissible affirmative instruction and the excluded

teacher $\mathcal{O}_T^e$ receives the prohibited affirmative instruction. They share exactly the same image and robot state. Both branches bypass all NTLs and run without gradients, ensuring that their difference is caused only by the language condition rather than scene variation or teacher adaptation.

An NTL is applied immediately before selected layer l. We execute that layer and observe the visual keys and values projected at the input of layer $l+1$; hence the observer measures the first visual representation that has incorporated the revised language through a complete attention block. For π_0 and $\pi_{0.5}$, image tokens form equal contiguous camera blocks, and we use the first, top-camera block. For GR00T, we use the 88 tokens identified by the processor as the coordinate-consistent `cam_high` block. Keys are measured after positional rotation and values after their value projection. The fourth GR00T VLSA block is not revised and serves only as the observer for the NTL before block 3.

For a candidate visual token i, we average cosine distance over KV heads and over keys and values:

$$D_i^l(A,B) = \frac{1}{2H_{\text{kv}}} \sum_{h=1}^{H_{\text{kv}}} \left[1 - \cos(\mathbf{k}_{A,i,h}^l, \mathbf{k}_{B,i,h}^l) + 1 - \cos(\mathbf{v}_{A,i,h}^l, \mathbf{v}_{B,i,h}^l)\right]. \tag{13}$$

For sample b, we compute $d_{b,i}^l = D_i^l(\mathcal{O}_{T,b}^a, \mathcal{O}_{T,b}^e)$ for every top-camera token. Let μ_b^l and s_b^l be the mean and standard deviation of these N scores at layer l, and let $k = \lceil 0.1N \rceil$. We select the token indices

$$\Phi_b^l = \{i \mid d_{b,i}^l > \mu_b^l + s_b^l\}, \qquad \Psi_b^l = \begin{cases} \Phi_b^l, & |\Phi_b^l| \geq k, \\ \text{TopK}_k(d_{b,:}^l), & |\Phi_b^l| < k, \end{cases} \tag{14}$$

where TopK_k returns the indices of the k largest scores. The fallback prevents weak teacher contrast from producing an empty or nearly empty training signal. Figure 6 visualizes the resulting teacher-induced Key and Value differences and retained visual tokens across layers. Within the selected set, weights are proportional to the teacher contrast,

$$w_{b,i}^l = \frac{\mathbb{I}[i \in \Psi_b^l](d_{b,i}^l + 10^{-6})}{\sum_j \mathbb{I}[j \in \Psi_b^l](d_{b,j}^l + 10^{-6})}. \tag{15}$$

The set, distances, and normalized weights are detached; consequently, neither teacher receives gradients and the model cannot reduce the loss by manipulating which tokens are selected.

The excluded teacher is used only to expose instruction-sensitive locations. The alignment target itself is the satisfying teacher. At every selected layer, the equivalence loss aligns the negated student visual Key and Value with $\mathcal{O}_T^a$ on the selected tokens. In parallel, the semantic loss aligns the negated resolver output $\mathbf{r}^{n,l}$ with the detached affirmative student content $\mathbf{c}^{a,l}$, and the gate loss classifies the B affirmative and B negated student inputs with targets zero and one, respectively. These are precisely the three terms in Eq. 9. Losses are averaged uniformly over insertion sites. Gradients pass through the negated student Key and Value and the frozen VLM operations into the NTLs, but the original VLA weights, both teachers, and the teacher-derived selection remain fixed. Thus, TGVA supplies object-level spatial guidance without object masks, action targets, or trainable teacher networks.

A.4 Optimization

Objective and optimizer. Only the NTL parameters are passed to AdamW; all pretrained VLA parameters remain frozen throughout post-training. We optimize Eq. 9 with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, weight decay 10^{-10}, and clip the global norm of all trainable NTL gradients to 1.0. These settings are shared by all three backbones.

Learning-rate schedule. Let T be the total number of optimization steps and W the number of warmup steps. The learning rate at step t is

$$\eta_t = \begin{cases} \eta_0 + (\eta_{\max} - \eta_0)t/W, & t < W, \\ \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})\left[1 + \cos\left(\pi \frac{t-W}{T-W}\right)\right], & t \geq W, \end{cases} \tag{16}$$

where $\eta_{\max} = 10^{-4}$, $\eta_{\min} = 10^{-5}$, and $\eta_0 = \eta_{\max}/(W+1)$. For simulation, we train for $T = 20{,}000$ steps with $W = 2{,}000$ warmup steps. Real-world $\pi_{0.5}$ adaptation uses the same optimizer and endpoint learning rates but trains for $T = 5{,}000$ steps with $W = 500$.

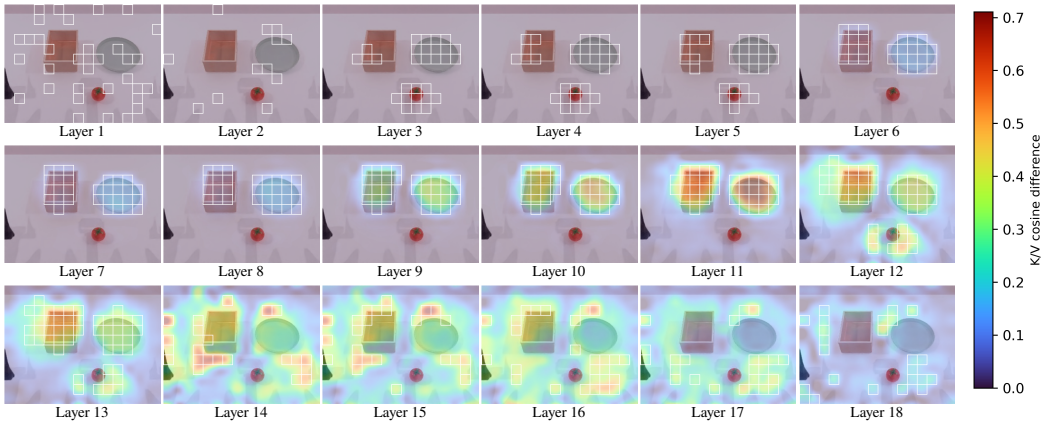

Figure 6: Layer-wise teacher-induced visual Key and Value differences and tokens retained by TGVA. Warmer colors indicate larger differences between satisfying and excluded instructions, and white boxes mark the adaptively selected tokens. Middle-layer selections concentrate on task-relevant objects, whereas deeper-layer selections become spatially diffuse.

Table 5: NegaAlign post-training configurations. Layer indices are one-based.

Backbone	Insertion sites	Trainable param.	Steps	Batch size	Warmup	Precision
GR00T	VLSA 1–3	4.35M	20,000	4	2,000	bfloat16
π_0	VLM 6–13	11.6M	20,000	4	2,000	bfloat16
$\pi_{0.5}$	VLM 6–13	11.6M	20,000	4	2,000	bfloat16
$\pi_{0.5}$ (real)	VLM 6–13	11.6M	5,000	4	500	bfloat16

Distributed batching. The reported batch size of 4 is the *global* number of independent scene–instruction examples per update. Under distributed data parallel training, it is divided evenly across workers, giving a per-worker batch size of $4/N$ for N workers; no gradient accumulation is used. The affirmative and negated student branches and the two frozen teacher branches are generated internally from each example and therefore do not change this optimization-batch definition.

Precision. Model forward passes use bfloat16, while cosine distances, gate binary cross-entropy, loss aggregation, and other reduction-sensitive calculations are evaluated in FP32. Table 5 summarizes the principal model-specific settings.

B Data Collection and Post-Training Pipeline

B.1 Simulation Data Collection and Composition

We construct the simulation data in RoboTwin 2.0 using SAPIEN and the dual-arm ALOHA-AgileX embodiment. The benchmark contains two scenes for each of five negation categories—object, attribute, comparison, goal, and complex—giving 10 scenarios in total. Across their affirmative and negated semantic variants, the environment defines 38 language-conditioned tasks. Objects, attributes, relative relations, and target placements are varied through seeded scene initialization.

Demonstrations are generated by the task-specific scripted expert using motion-planned grasping and placement primitives. A trajectory is retained only when motion planning succeeds and the binary success predicate of task is met. Each retained episode stores the seed, instruction, joint trajectory, end-effector state, and synchronized RGB observations. The raw HDF5 files contain four 320×240 camera streams (front, head, left wrist, and right wrist) and a 14-dimensional action vector ordered as the six left-arm joints, left gripper, six right-arm joints, and right gripper. The models use the head and two wrist views after conversion to the common LeRobot format.

The collected training corpus contains 2,360 episodes and 402,241 frames at 30 FPS. It comprises 38 task groups: 25 affirmative groups with 1,560 episodes and 13 explicitly negated groups with 800 episodes. Groups contain either 40 or 80 successful episodes. The explicit negated trajectories are used only by the fully supervised Pos.+Neg. baseline. Stage 1 affirmative training and NegaAlign Stage 2 do not use their observations, states, or actions.

B.2 Real-World Data Composition

The real-world dataset is collected on a UFACTORY xArm6 and contains 500 affirmative demonstrations, totaling 208,070 frames (approximately 1.93 hours) at 30 FPS. It covers 10 tasks with 50 episodes per task. The tasks form five semantic contrast pairs: spoon versus knife, separated versus mixed food, shirt versus socks, leaving versus storing a diamond, and ceramic versus stainless-steel bowls. No negated robot demonstration is included in this dataset.

The data follow the LeRobot v2.1 organization, with one Parquet trajectory and two synchronized MP4 streams per episode. The overhead camera is recorded at 1920×1080 and the wrist-mounted tool camera at 640×480. Each frame contains a 7-dimensional state and action vector ordered as 3-D position, 3-D rotation, and gripper position. The 10 tasks are balanced by episode count, although their durations differ: the dataset has a mean of 416.14 frames per episode, with lengths ranging from 229 to 1,027 frames. For Stage 1, all 500 affirmative episodes are available to train the base policy.

B.3 Construction of Instruction Relations

We manually organize the available affirmative skills into the relation $\mathcal{R}_q = (\ell_q^n, \mathcal{A}_q, \ell_q^e)$. The simulation configuration contains 13 relations, listed in Table 6. Most have one satisfying interpretation; the two food-exclusion relations retain both valid alternatives so that the supervision does not impose an arbitrary single behavior.

For real-world adaptation, the 10 affirmative tasks yield the five relations in Table 7. Each negated instruction is authored to preserve the common task while excluding one member of its affirmative pair. Consequently, the relation changes skill selection but does not require a new low-level manipulation primitive.

Before training, whitespace is normalized and all members of a relation are required to be nonempty and distinct. The episode manifest verifies that every satisfying and excluded affirmative instruction exists in the affirmative dataset and, critically, that no episode carries the negated instruction. The simulation manifest then loads all 840 episodes associated with 14 unique satisfying prompts; the real-world manifest loads 250 episodes associated with five satisfying prompts. Excluded-task episodes are checked for availability but are not loaded into the Stage 2 sampling pool. They define a language contrast, not an additional source of visual observations.

B.4 End-to-End Post-Training Pipeline

The Stage 2 loader first samples a relation uniformly, then samples a satisfying instruction uniformly from its set, and finally samples a frame uniformly from all episodes associated with that instruction. This hierarchical procedure prevents relations with more satisfying alternatives, longer trajectories, or more frames from dominating the objective. Simulation observations are repacked as the three ALOHA policy views and a 14-dimensional state; real-world observations are repacked as overhead/tool views and a 7-dimensional state. We apply the same input transforms and quantile normalization used by the corresponding Stage 1 policy, resize images to 224×224, tokenize the prompt, and explicitly mark only the instruction-token span. Actions may be read by the common dataset interface, but they are never supplied as a Stage 2 target and do not contribute to any NegaAlign loss.

For every sampled $(\mathbf{o}, \mathbf{s}, \ell^a)$, the pipeline retrieves its relation and re-tokenizes the same observation–state pair with ℓ^n and ℓ^e. It constructs a joint $2B$ student batch containing (ℓ^a, ℓ^n) and two B-sized, no-gradient teacher branches containing (ℓ^a, ℓ^e). The affirmative and negated student samples receive polarity targets 0 and 1, respectively. The satisfying teacher provides the semantic and visual destination, whereas the difference between satisfying and excluded teachers

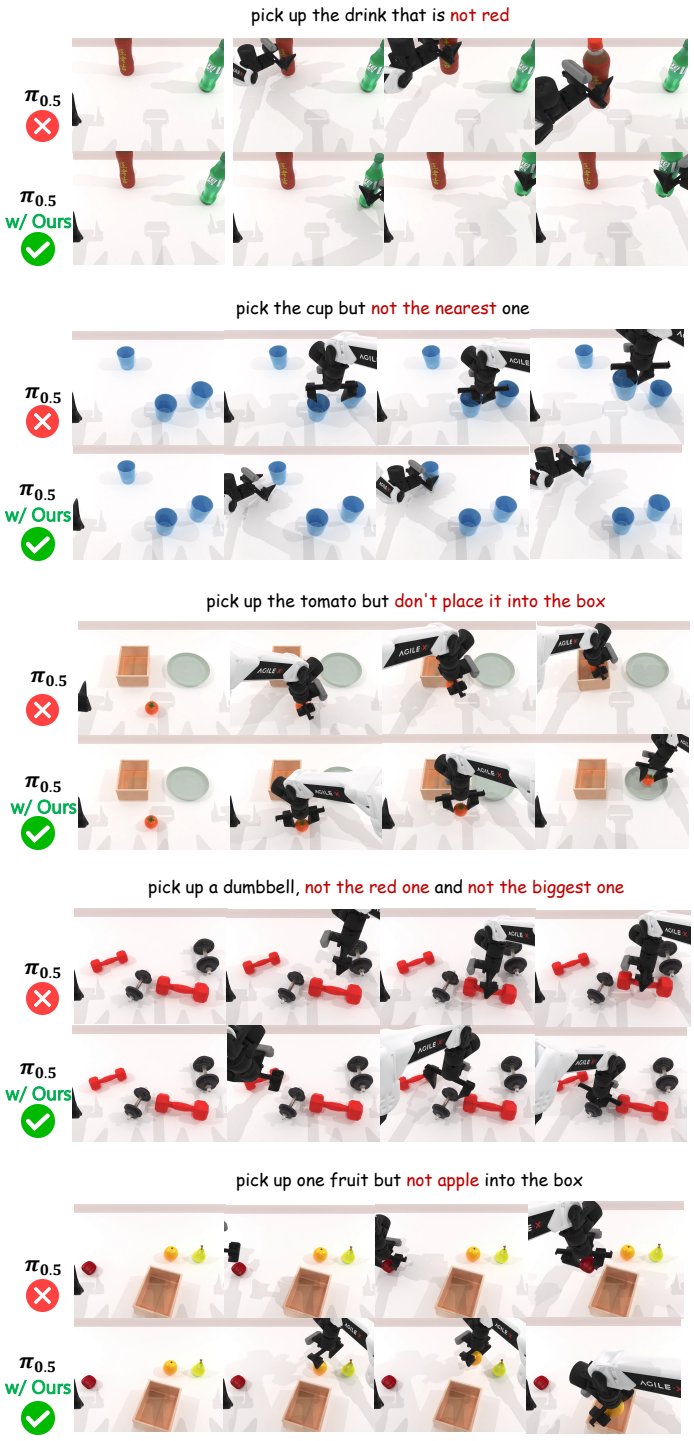

Figure 7: Qualitative evaluation results of affirmative-only $\pi_{0.5}$ and NegaAlign-adapted $\pi_{0.5}$ on five representative NegaBench scenarios. For each negated instruction, the top row shows the affirmative-only baseline violating the constraint, while the bottom row shows NegaAlign satisfying it. The columns show successive stages of each rollout.



Figure 8: Qualitative evaluation results of affirmative-only $\pi_{0.5}$ and NegaAlign-adapted $\pi_{0.5}$ on five real-world xArm6 scenarios. For each negated instruction, the top row shows the affirmative-only baseline violating the constraint, while the bottom row shows NegaAlign satisfying it. The columns show successive stages of each rollout.

Table 6: Complete instruction relations used for simulation post-training. For relations with multiple valid satisfying instructions, all alternatives are retained in $\mathcal{A}$.

Type	Instruction relation
Attribute	**Negated:** Pick up the drink that is not red. **Satisfying:** Pick up the drink that is green. **Excluded:** Pick up the drink that is red.
Attribute	**Negated:** Pick the cup that is not full. **Satisfying:** Pick the cup that is empty. **Excluded:** Pick the cup that is full.
Comparison	**Negated:** Pick the cup but not the nearest one. **Satisfying:** Pick the farthest cup. **Excluded:** Pick the nearest cup.
Comparison	**Negated:** Put one shoe in the shoebox, but do not choose the biggest one. **Satisfying:** Put one shoe in the shoebox; choose the smallest one. **Excluded:** Put one shoe in the shoebox; choose the biggest one.
Complex	**Negated:** Pick up the bottle but not the water, and place it on the plate but not the left one. **Satisfying:** Pick up the soy sauce and place it on the right plate. **Excluded:** Pick up the water and place it on the left plate.
Complex	**Negated:** Pick up the bottle but not the soy sauce, and place it on the plate but not the right one. **Satisfying:** Pick up the water and place it on the left plate. **Excluded:** Pick up the soy sauce and place it on the right plate.
Complex	**Negated:** Pick up a dumbbell, not the red one and not the biggest one. **Satisfying:** Pick up the black and smallest dumbbell. **Excluded:** Pick up the red and biggest dumbbell.
Complex	**Negated:** Pick up a dumbbell, not the black one and not the smallest one. **Satisfying:** Pick up the red and biggest dumbbell. **Excluded:** Pick up the black and smallest dumbbell.
Goal	**Negated:** Pick up the tomato but do not place it into the box. **Satisfying:** Pick up the tomato and place it on the plate. **Excluded:** Pick up the tomato and place it into the box.
Goal	**Negated:** Pick up the bread but do not place it on the table. **Satisfying:** Pick up the bread and place it on the plate. **Excluded:** Pick up the bread and place it on the table.
Object	**Negated:** Pick up one fruit but not the apple and put it into the box. **Satisfying:** Pick up the orange and put it into the box. **Excluded:** Pick up the apple and put it into the box.
Object	**Negated:** Place any food except the chips on the plate. **Satisfying:** Place the fries on the plate; or place the hamburger on the plate. **Excluded:** Place the chips on the plate.
Object	**Negated:** Place any food except the fries on the plate. **Satisfying:** Place the chips on the plate; or place the hamburger on the plate. **Excluded:** Place the fries on the plate.

selects the TGVA visual tokens. The three losses in Eq. 9 update only the NTL parameters. The teacher branches and all original VLA components remain frozen, and the action expert is not executed during post-training. Architecture and optimizer details are provided in Appendix A.

At inference time, relation lookup and both teacher branches are removed. A single observation, state, and user instruction pass through the frozen VLA augmented with the trained NTLs, after which the original action generator produces the action chunk. Thus, the additional training structure has no teacher-model cost at deployment.

Table 7: Complete instruction relations used for real-world post-training.

Scenario	Instruction relation
Dining	**Negated:** Hand me a piece of tableware, but not the knife. **Satisfying:** Hand me the spoon. **Excluded:** Hand me the knife.
Sorting	**Negated:** Put the meat and vegetables on the plates without mixing them. **Satisfying:** Put the meat and vegetable on separate plates. **Excluded:** Put the meat and vegetable on the plates, mixing them together.
Laundry	**Negated:** Put the clothes in the basket, but not the socks. **Satisfying:** Put the shirt in the basket. **Excluded:** Put the socks in the basket.
Storage	**Negated:** Clear the table and put everything into the cabinet except the diamond. **Satisfying:** Clear the table and put everything into the cabinet, leaving the diamond on the table. **Excluded:** Clear the table and put everything into the cabinet, including the diamond.
Cooking	**Negated:** Put a bowl into the microwave, except the stainless-steel bowl. **Satisfying:** Put the ceramic bowl into the microwave. **Excluded:** Put the stainless-steel bowl into the microwave.

C Evaluation Protocol and Detailed Results

C.1 NegaBench Evaluation Protocol

All simulation policies are evaluated through the same RoboTwin client–server pipeline. The client owns environment initialization, instruction selection, rollout, success checking, and result logging, while a model-specific server only converts observations to an action chunk. This separation permits each VLA to use its own software environment without changing the benchmark logic.

For every evaluated task variant, we run 100 episodes. Candidate initialization seeds are first executed by the scripted expert of task; a seed is admitted only when motion planning succeeds and the expert satisfies the same task-specific success predicate used for policy evaluation. The resulting task/config-specific seed list is cached and reused across policies. We fix the task manifest, task configuration, seed group, checkpoint, normalization statistics, and action execution horizon for each comparison.

At each closed-loop step, the policy receives the instruction, a head-view image, two wrist-view images, and its model-specific proprioceptive state. The π_0, $\pi_{0.5}$, and GR00T adapters execute 14-dimensional dual-arm joint-position actions. APT instead executes 16-dimensional dual-arm end-effector pose and gripper actions. A predicted chunk is truncated to the minimum of the configured execution horizon (32), the returned chunk length, and the remaining episode budget. The environment checks success throughout low-level action execution and terminates on the first successful state or after 300 high-level actions.

Each episode produces a binary outcome. For affirmative instructions, success requires completing the requested manipulation. For negated instructions, it additionally requires respecting every exclusion constraint. For task k with $N = 100$ episodes, we report

$$\mathrm{SR}_k = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}[\text{episode } i \text{ succeeds}]. \tag{17}$$

Category and overall values in the main paper are macro-averages of the corresponding task-level success rates, so tasks with longer trajectories do not receive additional weight. Representative evaluation rollouts of affirmative-only $\pi_{0.5}$ and NegaAlign-adapted $\pi_{0.5}$ across the five NegaBench categories are shown in Figure 7.

C.2 Real-World Evaluation Protocol

We evaluate $\pi_{0.5}$ and its NegaAlign-adapted counterpart on the five xArm6 scenarios in Table 7. The negated constraints respectively exclude a knife, mixed food placement, socks, moving a diamond,

Table 8: Detailed NegaBench success rates (%). Pos. FT uses only affirmative trajectories, Pos.+Neg. FT additionally uses explicit negated trajectories, and NegaAlign uses affirmative trajectories only during post-training.

Model	Category	Task	Pos. FT		Pos.+Neg. FT		w/ Ours	
			Pos.	Neg.	Pos.	Neg.	Pos.	Neg.
$\pi_{0.5}$	Object	1	88.50	2.00	83.00	71.00	83.00	79.00
		2	94.67	5.00	88.33	84.50	92.33	76.00
	Attribute	1	99.00	1.00	97.50	98.00	99.00	97.00
		2	77.00	0.00	83.50	88.00	84.50	92.00
	Comparison	1	96.00	12.00	84.00	84.00	94.00	98.00
		2	82.00	2.00	87.50	78.00	85.00	58.00
	Goal	1	92.00	0.00	93.50	85.00	89.50	87.00
		2	100.00	0.00	100.00	100.00	99.00	100.00
	Complex	1	99.25	4.00	98.25	99.00	100.00	98.50
		2	99.25	0.00	98.50	98.50	98.75	99.00
	Average		92.77	2.60	91.41	88.60	92.51	88.45
GR00T	Object	1	51.50	2.00	55.50	36.00	52.00	26.00
		2	68.67	0.50	66.33	52.00	67.67	23.50
	Attribute	1	84.50	0.00	83.50	90.00	87.50	77.00
		2	43.00	0.00	36.50	41.00	50.00	59.00
	Comparison	1	82.50	8.00	92.00	82.00	88.00	84.00
		2	57.50	0.00	51.00	48.00	57.50	39.00
	Goal	1	53.50	0.00	85.50	59.00	56.00	79.00
		2	96.50	0.00	97.00	96.00	96.00	96.00
	Complex	1	79.50	0.00	90.50	90.00	79.75	33.00
		2	93.75	0.00	96.75	94.50	94.50	61.00
	Average		71.09	1.05	75.46	68.85	72.89	57.75
π_0	Object	1	40.50	14.00	30.50	27.00	39.50	35.00
		2	21.00	15.50	29.33	26.50	19.33	14.50
	Attribute	1	65.00	36.00	92.00	96.00	64.00	58.00
		2	52.00	52.00	57.50	70.00	55.00	61.00
	Comparison	1	90.00	15.00	87.00	87.00	92.00	88.00
		2	61.00	21.00	68.50	58.00	61.00	31.00
	Goal	1	92.50	0.00	98.00	88.00	92.00	93.00
		2	100.00	0.00	99.00	100.00	100.00	100.00
	Complex	1	99.50	0.00	96.25	97.50	99.25	95.50
		2	60.50	0.50	58.75	62.50	62.00	57.00
	Average		68.20	15.40	71.68	71.25	68.41	63.30

and a stainless-steel bowl. A trial is successful only when the shared manipulation goal is completed and the excluded object or outcome is avoided.

Each negated instruction is evaluated for 50 trials. Each scenario also has two affirmative counterparts; we run 25 trials for each and pool their outcomes, producing the same 50-trial denominator for affirmative and negated evaluation. We report both the integer success count and success rate for every scenario, followed by the unweighted mean across the five scenarios. This pairing separates failure to perform the underlying manipulation from failure to obey the negated constraint. Representative evaluation rollouts of affirmative-only $\pi_{0.5}$ and NegaAlign-adapted $\pi_{0.5}$ across the five real-world scenarios are shown in Figure 8.

C.3 Detailed Simulation Results

Table 8 reports the task-level success rates underlying Table 1. When a task contains multiple instruction variants, we first average those variants within the task; category and overall scores are then macro-averaged across tasks. The final row of each model block therefore follows exactly the aggregation protocol used in the main paper.

Affirmative-only fine-tuning fails systematically on negated instructions: the task-level negated success rates are at most 12.00% for $\pi_{0.5}$ and 8.00% for GR00T, while π_0 remains at or below 52.00% on every task. NegaAlign improves negated success on all 30 backbone–task combinations without using negated trajectories. The gains are not confined to a single linguistic pattern: $\pi_{0.5}$ reaches at least 76.00% on every object task, at least 92.00% on every attribute task, and at least 98.50% on every complex task. Its overall negated score of 88.45% nearly matches the 88.60% obtained with explicit negated demonstrations.

Table 9: Instruction-level real-world success counts. The satisfying and excluded affirmative controls are evaluated separately and pooled only in the final Pos. row.

Scenario	Instruction	$\pi_{0.5}$	$\pi_{0.5}$ w/ Ours
Sorting	**Negated:** Put the meat and vegetables on the plates without mixing them.	9/50	40/50
	Satisfying: Put the meat and vegetable on separate plates.	22/25	23/25
	Excluded: Put the meat and vegetable on the plates, mixing them together.	24/25	19/25
Laundry	**Negated:** Put the clothes in the basket, but not the socks.	4/50	47/50
	Satisfying: Put the shirt in the basket.	23/25	25/25
	Excluded: Put the socks in the basket.	23/25	24/25
Storage	**Negated:** Clear the table and put everything into the cabinet except the diamond.	1/50	50/50
	Satisfying: Clear the table and put everything into the cabinet, leaving the diamond on the table.	22/25	22/25
	Excluded: Clear the table and put everything into the cabinet, including the diamond.	23/25	22/25
Cooking	**Negated:** Put a bowl into the microwave, except the stainless-steel bowl.	0/50	48/50
	Satisfying: Put the ceramic bowl into the microwave.	24/25	25/25
	Excluded: Put the stainless-steel bowl into the microwave.	25/25	23/25
Dining	**Negated:** Hand me a piece of tableware, but not the knife.	17/50	37/50
	Satisfying: Hand me the spoon.	24/25	23/25
	Excluded: Hand me the knife.	19/25	21/25
All	**Negated:** All five negated instructions	31/250 (12.4%)	222/250 (88.8%)
	Affirmative: All ten affirmative instructions	229/250 (91.6%)	227/250 (90.8%)

The task breakdown also identifies the remaining backbone-dependent failure modes. GR00T is weakest on object exclusion and on the first complex task, whereas π_0 retains larger gaps on object and complex constraints. Despite these differences, affirmative performance is preserved: relative to Pos. FT, NegaAlign changes the overall affirmative score by $+1.80$, $+0.21$, and -0.26 percentage points for GR00T, π_0, and $\pi_{0.5}$, respectively. The improvement therefore reflects constraint-sensitive behavior rather than a trade-off that sacrifices the original manipulation skills.

C.4 Detailed Real-World Results

Table 9 reports the success count for every real-world instruction. Each negated instruction uses 50 trials, while each of its satisfying and excluded affirmative controls uses 25 trials. The pooled rows exactly recover the affirmative and negated success rates reported in Table 3.

The affirmative-only policy completes 229 of 250 affirmative trials but only 31 of 250 negated trials, confirming that task competence alone does not yield exclusion-aware behavior. NegaAlign raises negated success in every scenario: Sorting improves from 9/50 to 40/50, Laundry from 4/50 to 47/50, Storage from 1/50 to 50/50, Cooking from 0/50 to 48/50, and Dining from 17/50 to 37/50.

The gain therefore spans constraints on objects, attributes, and outcomes rather than being driven by one task family.

At the same time, the adapted policy succeeds in 227 of 250 affirmative trials, only two fewer than the base policy. In particular, it remains able to execute the explicitly excluded affirmative alternative when that behavior is directly requested, such as placing the stainless-steel bowl in the microwave or handing over the knife. This rules out a trivial strategy that globally suppresses the excluded object and shows that the behavioral change is conditioned on the negated instruction. Dining remains the most challenging negated scenario at 37/50, suggesting that open-ended object selection leaves more residual ambiguity than the more structured placement tasks.