NS-VLA: Towards Neuro-Symbolic Vision-Language-Action Models

Ziyue Zhu[1,*] **Shangyang Wu**[1,*] **Shuai Zhao**[1,†] **Zhiqiu Zhao**[1] **Jian Zhang**[2] **Shengjie Li**[1]
Yi Wang[2] **Anh Tuan Luu**[2,3] **Xinliang Zhou**[2] **Fang Li**[2] **Haoran Luo**[2,†]
[1]Beijing University of Posts and Telecommunications
[2]Nanyang Technological University, Singapore [3]VinUniversity
{zyzhu, wushangyang, zhaoshuaiby}@bupt.edu.cn, haoran.luo@ntu.edu.sg

Abstract

Vision-Language-Action (VLA) models ground natural-language instructions in visual observations to generate actions for robotic manipulation. On long-horizon tasks, existing VLA models face structure-blind backbones, backbone-bound generalization, and flat single-objective optimization. To address these challenges, we propose a novel **N**euro-**S**ymbolic **V**ision-**L**anguage-**A**ction (NS-VLA) framework. It introduces a Neuro-Symbolic Encoder that parses the instruction into a primitive plan tracked by a gated monotone stage pointer, a Neuro-Symbolic Solver that conditions a chunked VLA policy on the active sub-instruction, and Hierarchical Joint Policy Optimization, which fits the pointer on demonstration segments and refines the action policy with success-filtered online learning. Even when trained on few demonstrations, NS-VLA clearly leads on LIBERO and the held-out LIBERO-Plus (76.5% and 51.8% against 71.2% and 45.4% for its backbone), exceeds the strongest baseline by 12.7 points on long-horizon tasks, and raises real-robot success from 40.0% to 60.0%. Our code is publicly available.[1]

1 Introduction

Vision-Language-Action (VLA) models [7, 16, 62, 92] map a natural-language instruction and visual observations to low-level robot actions (Figure 1) and are the main route toward generalist robots that follow free-form instructions [50, 55]. Many everyday instructions are long-horizon: "put the black bowl in the bottom drawer of the cabinet and close it" chains primitives that must each succeed in order. Such tasks are costly to demonstrate, so a practical VLA must learn them from a few examples.

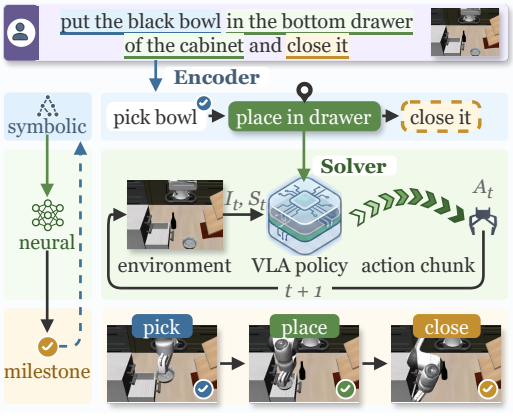


Figure 1: NS-VLA plans, then acts step by step.

Existing VLAs adopt a monolithic design: one backbone fuses vision and language, and a continuous action head decodes actions through diffusion [12, 90], flow matching [5, 6] or chunked regression [35, 79]. Others fine-tune multimodal LLMs [4, 34, 87, 88, 91] or compact backbones [71] to predict tokenized or chunked actions. Training pairs behavior cloning on expert demonstrations [20, 21, 32, 78] with online RL beyond them [39, 45], assuming the backbone internalizes the task decomposition, compositional generalization and training signal of long-horizon tasks.

*Equal contribution. †Corresponding authors. [1]https://github.com/Zuzuzzy/NS-VLA

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

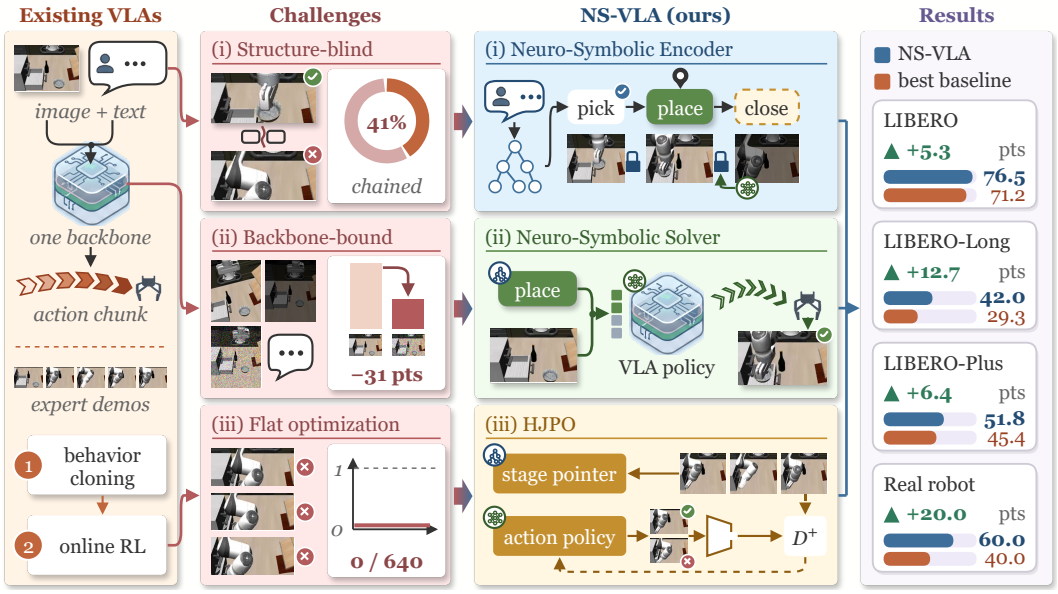


Figure 2: Overview: existing VLAs face three challenges; NS-VLA answers each with its Encoder, Solver and HJPO, and clearly leads on LIBERO, LIBERO-Plus and a real robot by up to 20 points.

However, this assumption raises three challenges on long-horizon tasks. **(i) Structure-blind backbones.** Tasks share reusable primitives [1, 28, 29, 42], yet end-to-end VLAs imitate a task as one undivided trajectory and cannot reuse a primitive in a new chain. **(ii) Backbone-bound generalization.** Generalization rests on one backbone, so paraphrases, new viewpoints, lighting or sensor noise cut success [19] beyond what the demonstrations cover. **(iii) Flat optimization.** Online RL maximizes one trajectory-level return of a flat policy [18, 39, 45], ignoring the task hierarchy [36, 38]; as long tasks rarely succeed early, the binary reward is silent and partial progress unrewarded: flat RL on VLA-Adapter succeeds in none of its 640 online rollouts on the LIBERO-Long suite.

We therefore propose a **N**euro-**S**ymbolic **V**ision-**L**anguage-**A**ction (**NS-VLA**) framework (Figure 2), answering the three challenges one to one. **First**, the **Neuro-Symbolic Encoder** parses the instruction into an ordered plan of primitives like pick, place and close, and tracks it with a monotone stage pointer that advances only when a neural classifier and a symbolic gripper gate agree. **Moreover**, the **Neuro-Symbolic Solver** turns the active primitive into a sub-instruction for a chunked VLA policy, so each demonstration segment teaches a reusable skill in one canonical wording; on single-primitive tasks it equals the backbone. **Furthermore**, **Hierarchical Joint Policy Optimization (HJPO)** fits the pointer on demonstration segments and refines the action policy by advantage-weighted regression on successful rollouts and demonstrations, giving each level its own learning signal.

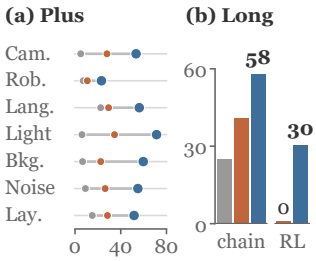
We evaluate NS-VLA on LIBERO [43], the held-out LIBERO-Plus [19] and a real robot arm. Trained on only a few demonstrations, it ranks first or ties for first in every column of Table 2 against ten baselines and its VLA-Adapter backbone. It scores 76.5 against 71.2 on LIBERO, reaches 42.0 on LIBERO-Long, 12.7 points above the strongest baseline, scores 51.8 against 45.4 on LIBERO-Plus, where it wins 77 more of the 1,200 paired variants than its backbone, a significant margin (exact McNemar test), and raises real-robot success from 40.0% to 60.0%. Neuro-symbolic structure [14, 81] thus makes VLAs data-efficient, robust and interpretable on long-horizon manipulation.

2 Related work

Neuro-symbolic learning. Neuro-symbolic methods pair neural perception with symbolic structure for compositional, verifiable reasoning, from concept learners and module networks [25, 48, 75] to probabilistic logic, theorem provers and program synthesis [17, 47, 63]. LLMs extend them with programs, tools and solvers [24, 57, 65, 81, 83, 84], and embodied agents compose skills, code and value maps from language [1, 13, 28, 29, 31, 42]. NS-VLA brings this structure into closed-loop control: a parsed plan and a gated stage pointer drive a chunked VLA policy, one primitive at a time.

Table 1: Related paradigms and our pilot results (colors as the dots): (a) LIBERO-Plus success rate (SR) per perturbation; (b) LIBERO-Long SR after the first primitive (chain) and of RL rollouts.

Paradigm	Core formulation	Ctrl.	Sym.	Comp.	Hier.
• Imitation [12, 46]	$\min_\theta \ell(\pi_\theta(o_t, x), a_t)$	✓	×	×	×
• Generalist VLA [34, 79]	$A_t = f_\theta(I_t, S_t, x)$	✓	×	×	×
• VLA + online RL [39, 45]	$\max_\theta \mathbb{E}[R(\tau)]$, $R \in \{0, 1\}$	✓	×	×	×
Hierarchical RL [2, 52]	$\max \mathbb{E}[e^{\mathrm{adv}/\beta} \log \pi(a\|s, u)]$	✓	×	×	✓
Neuro-symbolic [13, 48]	$y = \mathrm{Exec}(\mathrm{Parse}(x))$	×	✓	✓	×
• **NS-VLA (ours)**	$\pi^{\mathrm{h}}_\phi(u_t\|\mathcal{H}_t)\, \pi^{\mathrm{l}}_\theta(A_t\|I_t, S_t, \tilde{x}_t)$	✓	✓	✓	✓



Vision-language-action models. Following generalist robot policies [7, 50], VLAs fine-tune vision-language backbones [5, 6, 34, 92] with latent, tokenized or world-model actions [9, 11, 60], efficient designs [8, 27, 30, 35, 41, 44] and compact backbones [71, 79], evaluated on LIBERO, LIBERO-Plus and CALVIN [19, 43, 49]. Online RL tunes them beyond demonstrations [3, 18, 39, 40, 45, 74, 85] via policy-gradient and advantage-weighted updates [53, 59, 68], and hierarchical RL and VLAs add temporal abstraction [2, 10, 15, 52, 58, 70]. NS-VLA unites these lines in one hierarchical VLA.

3 Preliminaries

Problem formulation. Manipulation is a POMDP [33]: at decision t a VLA π_θ maps instruction x and observation $o_t = (I_t, S_t)$ (images, proprioception) to an H-step chunk A_t. Given history $\mathcal{H}_t$, a neuro-symbolic VLA factorizes it into a symbolic high level for primitive u_t and a neural low level:

$$\pi_\Theta\big(u_t, A_t \mid \mathcal{H}_t\big) = \pi^{\mathrm{h}}_\phi\big(u_t \mid \mathcal{H}_t\big) \cdot \pi^{\mathrm{l}}_\theta\big(A_t \mid u_t, \mathcal{H}_t\big), \qquad \Theta = (\phi, \theta). \tag{1}$$

Here x is parsed into a primitive plan $p = (u^{(1)}, \ldots, u^{(M)})$ under pointer m_t, $u_t = u^{(m_t)}$; both factors learn from demonstrations, the low level also from rollouts, to maximize $\mathbb{E}[R(\tau)]$, $R(\tau) = \mathbf{1}[\tau \models x]$. This separates what to do from how to move and equals the VLA when $M = 1$.

Pilot study. Table 1 compares paradigms by continuous control (Ctrl.), symbolic structure (Sym.), primitive-level conditioning (Comp.) and hierarchical optimization (Hier.); only NS-VLA covers all four. The pilots confirm the challenges. (i) After the first primitive, imitation and generalist VLAs complete 25.0% and 40.9% of LIBERO-Long versus 57.9% for NS-VLA. (ii) Monolithic families average below 35% per Plus category, while NS-VLA leads all seven, five by over 25 points. (iii) Flat RL on VLA-Adapter succeeds in none of 640 rollouts, a silent reward, versus 195 for HJPO.

4 Method: NS-VLA

As shown in Figure 3, the Encoder realizes the symbolic high level π^{h}_ϕ, the Solver the neural low level π^{l}_θ, and HJPO trains both from demonstrations and the policy's own successes.

4.1 Neuro-Symbolic Encoder

Compositional plan grammar. The Encoder compiles x over the vocabulary $\mathcal{U} = \{\texttt{pick}, \texttt{place}, \texttt{open}, \texttt{close}, \texttt{turn on}, \texttt{turn off}, \texttt{push}\}$. A deterministic grammar $\mathcal{P}$ splits x into clauses $c_1, \ldots, c_C$, resolves shared verbs and anaphora, keeps qualifiers and rewrites clauses by rw:

$$p \;=\; \mathcal{P}(x) \;=\; \bigoplus\nolimits_{i=1}^{C} \mathrm{rw}(c_i), \qquad \mathrm{rw}\big(\texttt{put}(o, \rho, g)\big) \;=\; \big((\texttt{pick}, o), (\texttt{place}_\rho, (o, g))\big), \tag{2}$$

where $\oplus$ concatenates, o and g are the object and its target, ρ is the placement relation, and other clauses map to $(\mathrm{op}_c, \mathrm{arg}_c)$. Each u is verbalized as a sub-instruction $\mathrm{text}(u)$, so $p = (u^{(1)}, \ldots, u^{(M)})$ is executable and human-readable, and an undecomposed instruction runs the one-slot plan $p = (x)$. Parsing runs once per episode, before the first action, and requires no learned parameters.

Symbolic precondition gate. A primitive is complete only once its effect shows in the robot's own state. Let $\kappa_t \in \{0, 1\}$ be the last gripper command (1 = close), w_t the finger separation in S_t and

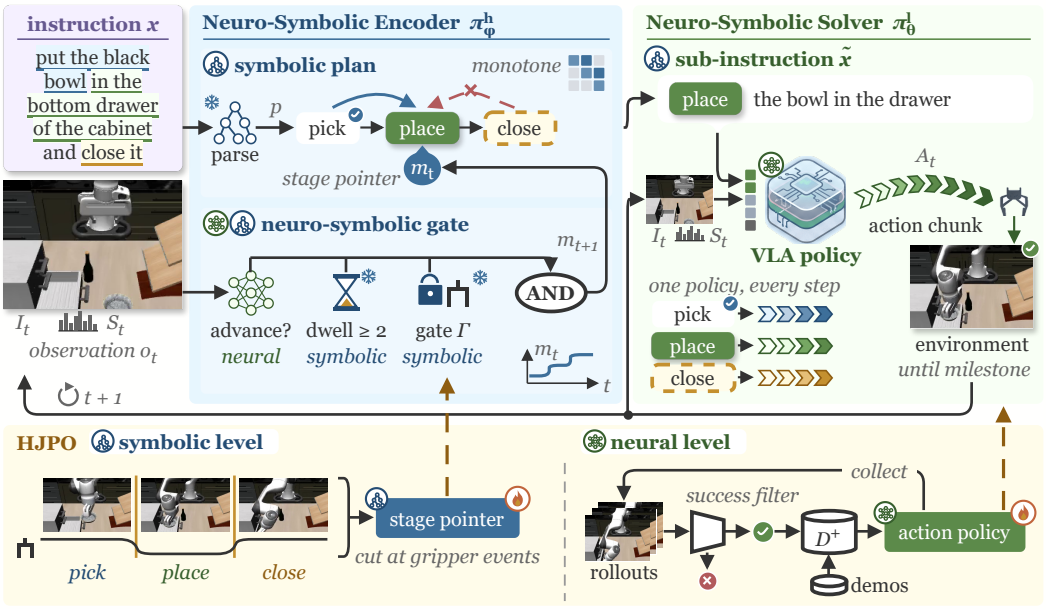


Figure 3: Overview of NS-VLA: the Encoder parses a primitive plan and gates a monotone pointer, the Solver conditions the backbone on the active primitive, and HJPO trains both factors.

$h_t = \kappa_t \mathbf{1}[w_t > w_{\min}]$ the holding indicator with threshold $w_{\min}$; $\mathcal{U}$ splits into grasp $\mathcal{U}_{\mathrm{g}} = \{\texttt{pick}\}$, grasp–release $\mathcal{U}_{\mathrm{r}}$ and gripper-free $\mathcal{U}_{\mathrm{f}}$ primitives, and t_m is the first decision of slot m:

$$\Gamma_t(u) \;=\; \mathbf{1}\big[u \in \mathcal{U}_{\mathrm{f}}\big] \;+\; \mathbf{1}\big[u \in \mathcal{U}_{\mathrm{g}}\big]\, h_t \;+\; \mathbf{1}\big[u \in \mathcal{U}_{\mathrm{r}}\big]\,(1-\kappa_t) \max_{t_{m_{t-1}} \le t' \le t} h_{t'}, \tag{3}$$

so a `pick` is certified only while an object is held, a grasp–release one only after a grasp and a release in its slot, and a gripper-free one is left to the classifier. This closed-form predicate on proprioception and commands makes every transition auditable and independent of the images and the backbone.

Gated monotone stage pointer. A light classifier g_ϕ judges stage completion from features z_t fusing the pooled backbone context $\bar{h}_t$ of observation and instruction, embeddings $e(\cdot)$ of the current and next sub-instruction, the state S_t, the dwell time d_t since the last advance and the progress m_{t-1}/M. The pointer then advances by at most one slot per decision, never skipping or revisiting a step:

$$m_t \;=\; m_{t-1} \;+\; \mathbf{1}\big[m_{t-1} < M\big]\; \mathbf{1}\big[d_t \ge d_{\min}\big]\; \Gamma_t\big(u^{(m_{t-1})}\big)\; \mathbf{1}\big[\sigma\big(g_\phi(z_t)\big) > \tfrac{1}{2}\big], \tag{4}$$

where σ is the logistic function and $d_{\min}$ the minimum dwell. A neural judgment thus moves the pointer only when a symbolic rule agrees, and the high-level factor is the deterministic policy $\pi_\phi^{\mathrm{h}}(u \mid \mathcal{H}_t) = \mathbf{1}[u = u^{(m_t)}]$, whose trace can be read off and verified at every decision.

Proposition 4.1 (Certified execution)**.** *For every ϕ, the pointer splits each episode into $\le M$ ordered contiguous segments, each non-final one lasting $\ge d_{\min}$ decisions and left only once Γ certifies it.*

Proof. See Appendix B.1; Section 5.4 measures how well pointer advances are grounded. □

4.2 Neuro-Symbolic Solver

Primitive-conditioned chunk policy. The Solver grounds the active primitive in an unchanged chunked VLA backbone f_θ. Its bridged input $\tilde{x}^{(m)} = x \oplus \texttt{now:} \oplus \mathrm{text}(u^{(m)})$ keeps x and names the active step ($\tilde{x}^{(1)} = x$ for a one-slot plan), and f_θ is read as the mode of a Laplace chunk policy:

$$\pi_\theta^{\mathrm{l}}\big(A_t \mid u^{(m_t)}, \mathcal{H}_t\big) \;=\; (2b)^{-7H} \exp\!\Big(-\tfrac{1}{b}\,\big\|A_t - f_\theta\big(I_t,\, S_t,\, \tilde{x}^{(m_t)}\big)\big\|_1\Big), \tag{5}$$

where $A_t \in \mathbb{R}^{H\times 7}$ stacks H end-effector deltas with gripper commands and $b > 0$ is a scale. The mode runs open-loop and is re-predicted right after an advance, so every executed chunk is conditioned on a certified primitive; L1 regression is exact maximum likelihood under this policy.

Sub-instruction-augmented imitation. Demonstrations are segmented by gripper events at no annotation cost, so each frame i carries its segment m and chunk $A_i^\star$. A fair coin $\xi \sim \mathrm{Bern}(1/2)$ selects the input x_i^ξ, bridged $\tilde{x}^{(m)}$ if $\xi = 1$ and native x otherwise, and the backbone minimizes

$$\mathcal{L}_{\mathrm{SFT}}(\theta) \;=\; \mathbb{E}_{(x,\, o_i,\, A_i^\star,\, m)\sim\mathcal{D},\ \xi\sim\mathrm{Bern}(1/2)}\Big[\big\|f_\theta\big(I_i,\, S_i,\, x_i^\xi\big) - A_i^\star\big\|_1\Big], \tag{6}$$

which keeps the backbone fluent on native instructions while every segment adds a primitive-specific signal. Because sub-instructions recur across tasks, a primitive learned in one chain is rehearsed whenever it reappears in another, which lets limited demonstrations cover long horizons.

Cross-task primitive sharing. This sharing is exact. Let $\mathcal{L}_{\mathrm{nat}}$ be the imitation loss on native inputs, λ_u the fraction of demonstration frames whose segment executes u ($u = x$ on one-slot plans), and $\mathcal{L}_u$ the mean bridged loss over those frames pooled over tasks; as ξ is independent of the data,

$$\mathcal{L}_{\mathrm{SFT}}(\theta) \;=\; \tfrac{1}{2}\,\mathcal{L}_{\mathrm{nat}}(\theta) \;+\; \tfrac{1}{2}\sum_u \lambda_u\,\mathcal{L}_u(\theta), \qquad \sum_u \lambda_u = 1, \tag{7}$$

so half of the supervision is organized by primitive rather than by task, and each $\mathcal{L}_u$ aggregates every demonstration of u in the suite, whatever task it came from. On a one-slot plan the two halves coincide frame by frame, and the objective is exactly the backbone's imitation loss.

Proposition 4.2 (Graceful equivalence)**.** *If $M = 1$, then for every θ NS-VLA equals its backbone $\pi_\theta(\cdot \mid I_t, S_t, x)$ at every history, with the same trajectory law and training objectives.*

Proof. See Appendix B.2; Section 5.5 studies the Solver on multi-primitive LIBERO-Long plans. □

4.3 Hierarchical Joint Policy Optimization

Factorized hierarchical likelihood. HJPO trains both factors with signals matched to their types, discrete stage transitions for the Encoder and continuous action chunks for the Solver. Viewing each stay/advance decision y_t as a Bernoulli variable with $p_\phi(y_t = 1 \mid z_t) = \sigma(g_\phi(z_t))$, the log-likelihood of a rollout τ under the hierarchical policy separates into a discrete and a continuous part:

$$\log \pi_\Theta(\tau) \;=\; \sum_{t=1}^{T}\Big[\log p_\phi\big(y_t \mid z_t\big) \;+\; \log \pi_\theta^{\mathrm{l}}\big(\hat{A}_t \mid u^{(m_t)}, \mathcal{H}_t\big)\Big] \;+\; \log \mu_{\mathrm{env}}(\tau), \tag{8}$$

where $\hat{A}_t$ is the executed chunk and μ_{env} collects the parameter-free initial-state and transition terms. With z_t as fixed inputs, the gradient in ϕ involves only stage transitions and the gradient in θ only action chunks, so each factor fits the signal suited to it, without a learned critic or value function.

High-level factor. The classifier learns from segments: each frame of segment j yields a stay example under pointer j and, for $j > 1$, an advance example under pointer $j - 1$, fit by class-balanced cross-entropy. With the gate it fixes π_ϕ^{h} and each rollout's sub-instructions for online refinement.

Low-level factor. The action policy is first fit by sub-instruction-augmented imitation and then improved online from its own experience. Each iteration rolls out the current hierarchical policy with exploration noise, stores every executed chunk of each successful rollout with its bridged input in a replay set $\mathcal{D}^+$, and mixes it with the expert demonstrations at a fixed mixing ratio η:

$$\mathcal{L}_{\mathrm{HJPO}}(\theta) \;=\; \eta\,\mathcal{L}_{\mathrm{SFT}}(\theta) \;+\; (1-\eta)\,\mathbb{E}_{(o,\,\tilde{x},\,\hat{A})\sim\mathcal{D}^+}\big\|f_\theta(I, S, \tilde{x}) - \hat{A}\big\|_1. \tag{9}$$

This is advantage-weighted regression (AWR) [59] at zero temperature with task-level credit: the Laplace log-likelihood of the Solver turns its weighted maximum likelihood exactly into L1 regression, and under the binary reward its weights collapse to a hard, parameter-free success filter,

$$w_\beta(\tau) \;=\; \exp\!\Big(\frac{R(\tau) - R_{\max}}{\beta}\Big) \;\xrightarrow[\beta\to 0]{}\; \mathbf{1}\big[R(\tau) = R_{\max}\big], \qquad R_{\max} = 1, \tag{10}$$

with temperature $\beta > 0$, so each update regresses the policy onto its own successes, anchored by the demonstrations, while failed rollouts drop out of the update without any reward shaping. The reported policy $\hat{\theta}$ maximizes the validation success rate $\mathrm{SR}_{\mathrm{val}}$ over the validated iterates, the supervised start θ_0 itself included, so selection never falls below the supervised policy on validation.

Proposition 4.3 (Filtered AWR)**.** *With binary reward and nonempty $\mathcal{D}^+$, as $\beta \to 0$ anchored AWR converges to $\mathcal{L}_{\mathrm{HJPO}}$ and to the success-conditioned law at rate $e^{-1/\beta}$, and $\mathrm{SR}_{\mathrm{val}}(\hat{\theta}) \geq \mathrm{SR}_{\mathrm{val}}(\theta_0)$.*

Proof. See Appendix B.3; Section 5.6 reports the online stage across all 19 online runs. □

Table 2: SR (%) on LIBERO and LIBERO-Plus (Plus Avg pools 1,200 variants); subscripts: standard error (no success: 95% bound u as $u/2 \pm u/2$); **best**, second; Δ: gain over best baseline.

Method	LIBERO					LIBERO-Plus							
	Spat.	Obj.	Goal	Long	Avg	Cam.	Rob.	Lang.	Light	Bkg.	Noise	Lay.	Avg
BC-RNN	$7.0_{\pm 1.8}$	$21.5_{\pm 2.9}$	$13.5_{\pm 2.4}$	$8.5_{\pm 2.0}$	$12.6_{\pm 1.2}$	$0.8_{\pm 0.8}$	$0.5_{\pm 0.5}$	$3.8_{\pm 1.4}$	$0.7_{\pm 0.7}$	$1.1_{\pm 1.1}$	$0.5_{\pm 0.5}$	$2.2_{\pm 1.1}$	$1.2_{\pm 0.3}$
BC-ViLT	$26.0_{\pm 3.1}$	$64.5_{\pm 3.4}$	$35.0_{\pm 3.4}$	$3.5_{\pm 1.3}$	$32.3_{\pm 1.5}$	$0.8_{\pm 0.8}$	$1.1_{\pm 0.8}$	$10.8_{\pm 2.3}$	$6.6_{\pm 2.1}$	$1.6_{\pm 1.1}$	$0.8_{\pm 0.8}$	$3.8_{\pm 1.4}$	$3.3_{\pm 0.5}$
QueST	$46.0_{\pm 3.5}$	$65.5_{\pm 3.4}$	$47.0_{\pm 3.5}$	$16.5_{\pm 2.6}$	$43.8_{\pm 1.6}$	$11.0_{\pm 2.3}$	$10.2_{\pm 2.2}$	$31.9_{\pm 3.4}$	$11.0_{\pm 2.7}$	$14.7_{\pm 3.1}$	$14.7_{\pm 2.6}$	$24.2_{\pm 3.2}$	$17.1_{\pm 1.1}$
BC-T	$48.5_{\pm 3.5}$	$63.0_{\pm 3.4}$	$47.5_{\pm 3.5}$	$16.5_{\pm 2.6}$	$43.9_{\pm 1.6}$	$0.8_{\pm 0.8}$	$4.3_{\pm 1.5}$	$11.4_{\pm 2.3}$	$2.9_{\pm 1.4}$	$3.1_{\pm 1.5}$	$0.5_{\pm 0.5}$	$7.7_{\pm 2.0}$	$4.3_{\pm 0.6}$
VQ-BeT	$44.0_{\pm 3.5}$	$71.5_{\pm 3.2}$	$51.5_{\pm 3.5}$	$15.5_{\pm 2.6}$	$45.6_{\pm 1.6}$	$5.8_{\pm 1.7}$	$7.0_{\pm 1.9}$	$20.5_{\pm 3.0}$	$0.7_{\pm 0.7}$	$8.5_{\pm 2.5}$	$8.9_{\pm 2.1}$	$15.9_{\pm 2.7}$	$10.0_{\pm 0.8}$
Smol.	$57.5_{\pm 3.5}$	$59.0_{\pm 3.5}$	$52.5_{\pm 3.5}$	$27.5_{\pm 3.2}$	$49.1_{\pm 1.7}$	$0.8_{\pm 0.8}$	$0.8_{\pm 0.8}$	$3.2_{\pm 1.3}$	$5.1_{\pm 1.9}$	$5.4_{\pm 2.0}$	$0.5_{\pm 0.5}$	$2.2_{\pm 1.1}$	$2.1_{\pm 0.4}$
Octo	$61.5_{\pm 3.4}$	$66.5_{\pm 3.3}$	$60.5_{\pm 3.5}$	$11.5_{\pm 2.3}$	$50.0_{\pm 1.6}$	$33.0_{\pm 3.4}$	$12.9_{\pm 2.5}$	$36.2_{\pm 3.5}$	$39.7_{\pm 4.2}$	$6.2_{\pm 2.1}$	$35.6_{\pm 3.5}$	$34.6_{\pm 3.5}$	$28.9_{\pm 1.2}$
DP	$59.0_{\pm 3.5}$	$77.5_{\pm 3.0}$	$57.5_{\pm 3.5}$	$29.0_{\pm 3.2}$	$55.8_{\pm 1.6}$	$2.1_{\pm 1.0}$	$8.6_{\pm 2.1}$	$21.1_{\pm 3.0}$	$7.4_{\pm 2.2}$	$8.5_{\pm 2.5}$	$12.0_{\pm 2.4}$	$21.4_{\pm 3.0}$	$11.8_{\pm 0.9}$
BAKU	$38.5_{\pm 3.4}$	$85.5_{\pm 2.5}$	**$81.0_{\pm 2.8}$**	$24.5_{\pm 3.0}$	$57.4_{\pm 1.5}$	$19.9_{\pm 2.9}$	$18.3_{\pm 2.8}$	$39.5_{\pm 3.6}$	$14.7_{\pm 3.0}$	$7.8_{\pm 2.4}$	$23.6_{\pm 3.1}$	$31.3_{\pm 3.4}$	$23.1_{\pm 1.2}$
ACT	$79.5_{\pm 2.9}$	$78.5_{\pm 2.9}$	$62.0_{\pm 3.4}$	$20.0_{\pm 2.8}$	$60.0_{\pm 1.5}$	$1.0_{\pm 0.7}$	$7.0_{\pm 1.9}$	$40.5_{\pm 3.6}$	$4.4_{\pm 1.8}$	$8.5_{\pm 2.5}$	$12.6_{\pm 2.4}$	$14.3_{\pm 2.6}$	$13.1_{\pm 0.9}$
VLA-Ad.	$90.0_{\pm 2.1}$	**$87.0_{\pm 2.4}$**	$78.5_{\pm 2.9}$	$29.3_{\pm 14.7}$	$71.2_{\pm 3.8}$	$50.8_{\pm 3.6}$	$19.4_{\pm 2.9}$	$48.6_{\pm 3.7}$	$58.8_{\pm 4.2}$	$55.8_{\pm 4.4}$	$42.9_{\pm 3.6}$	$48.4_{\pm 3.7}$	$45.4_{\pm 1.2}$
NS-VLA	**$96.0_{\pm 1.4}$**	**$87.0_{\pm 2.4}$**	**$81.0_{\pm 2.8}$**	**$42.0_{\pm 3.5}$**	**$76.5_{\pm 1.3}$**	**$53.4_{\pm 3.6}$**	**$23.1_{\pm 3.1}$**	**$56.2_{\pm 3.6}$**	**$71.3_{\pm 3.9}$**	**$59.7_{\pm 4.3}$**	**$55.0_{\pm 3.6}$**	**$51.6_{\pm 3.7}$**	**$51.8_{\pm 1.4}$**
Δ	**+6.0**	**+0.0**	**+0.0**	**+12.7**	**+5.3**	**+2.6**	**+3.7**	**+7.6**	**+12.5**	**+3.9**	**+12.1**	**+3.2**	**+6.4**

5 Experiments

We study seven research questions. **RQ1:** Does NS-VLA beat policies trained on the same data? **RQ2:** Does each component contribute? **RQ3–RQ5:** How do the Encoder, Solver and HJPO each help? **RQ6:** How does NS-VLA run a long instruction? **RQ7:** Do gains transfer to real robots?

5.1 Setup

Datasets. LIBERO [43] has four ten-task suites: Spatial, Object, Goal and long-horizon Long. Every method trains on 10% of the official demonstrations, the first five per task. LIBERO-Plus [19] is held out for evaluation: 10,030 variants of the same tasks along seven perturbation dimensions with five difficulty levels: camera (Cam.), robot initial state (Rob.), language (Lang.), lighting (Light), background (Bkg.), sensor noise (Noise) and object layout (Lay.). Details are in Appendix E.

Baselines. We compare against ten language-conditioned policies with at most 2B parameters, each trained on the same five demonstrations per task with its official recipe: BC-RNN [46], BC-Transformer (BC-T) and BC-ViLT [43], ACT [89], Diffusion Policy (DP) [12], VQ-BeT [37], QueST [51], BAKU [26], Octo [50] and SmolVLA (Smol.) [71], together with VLA-Adapter (VLA-Ad.) [79], the backbone that NS-VLA builds on. Further details are in Appendix D.

Metrics. Success rate (SR) is the share of successful episodes. LIBERO uses the first 20 official initial states per task and LIBERO-Plus 300 pre-registered variants per suite, shared by all methods; RQ2–RQ5 mainly analyze the 500 LIBERO-Long test episodes, which chain primitives. The Plan-Stage Completion Rate (PSCR) is the share of plan stages whose simulator milestone is reached.

Implementation. NS-VLA wraps VLA-Adapter (Qwen2.5-0.5B with DINOv2 and SigLIP encoders) initialized from the generic vision-language model. Supervised training runs 20k steps on Long and 10k elsewhere, and the online stage runs 8 iterations of 8 rollouts per task. The reported model is selected on validation states, the supervised start included. Hyperparameters are in Appendix F.

5.2 Main results (RQ1)

As shown in Table 2, we compare NS-VLA with ten baselines and its VLA-Adapter backbone on LIBERO and the held-out LIBERO-Plus, and it ranks first or ties for first in all thirteen columns.

The lead is largest where tasks chain primitives. On LIBERO, NS-VLA averages 76.5 against 71.2 for VLA-Adapter and 60.0 for ACT, and on Long it reaches 42.0, 12.7 points above the strongest baseline, while no baseline exceeds 30; this gain is more than twice the gain on any other LIBERO suite. On Spatial it reaches 96.0, six points above VLA-Adapter, the best baseline, and on Object and Goal it matches the strongest baselines, VLA-Adapter and BAKU, at 87.0 and 81.0.

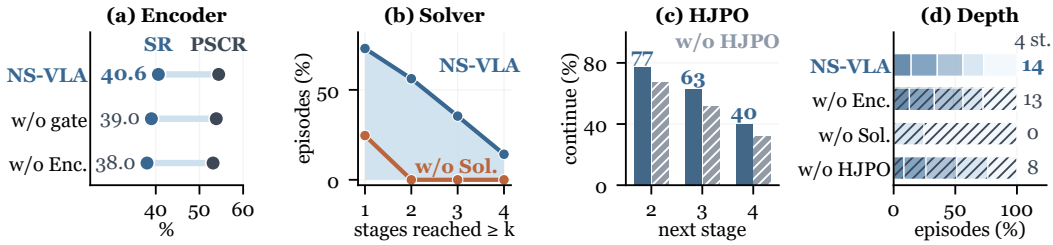


Figure 4: Ablations: (a) full model, without gate and without Encoder; (b) episodes reaching at least k stages; (c) share of episodes continuing to the next stage; (d) plan stages reached per variant.

The lead holds under perturbation. On the held-out LIBERO-Plus, NS-VLA averages 51.8 against 45.4 for VLA-Adapter and 28.9 for Octo and leads all seven dimensions, with the largest margins under lighting (+12.5), sensor noise (+12.1) and language (+7.6). Over the 1,200 paired variants it wins 77 more variants than VLA-Adapter, a significant margin (exact McNemar test, $p = 1.5 \times 10^{-5}$). Its margin over VLA-Adapter thus grows from 5.3 points on LIBERO to 6.4 points under perturbation.

5.3 Ablation study (RQ2)

We remove one component at a time: Table 3 reports SR and PSCR, with drops as subscripts, and Figure 4 shows at which plan stage each component acts.

Table 3: Ablations of each component.

Variant	SR	PSCR
NS-VLA	**40.6**	**54.4**
w/o Encoder	38.0 $_{-2.6}$	53.1 $_{-1.3}$
w/o Solver	0.0 $_{-40.6}$	7.3 $_{-47.1}$
w/o HJPO	28.2 $_{-12.4}$	48.9 $_{-5.5}$

Every component contributes. Removing any component lowers both SR and PSCR. The Solver lifts SR from 0.0 to 40.6, HJPO adds 12.4 points over supervised training, and the Encoder another 2.6 points at inference.

Each component plays a distinct role. The gate and the Encoder each add success, lifting SR from 39.0 and 38.0 to 40.6 (a). With the Solver, 56.2% of episodes complete two stages, which the backbone never does (b). HJPO raises continuation at every later stage to 77, 63 and 40% (c), so the full model reaches four stages in 14.2% of episodes, more often than any ablated variant (d).

5.4 Analysis of the Neuro-Symbolic Encoder (RQ3)

As shown in Figure 5, we test whether the plan and the monotone pointer track execution, measuring how well pointer advances match simulator milestones and how the plan transfers to paraphrases.

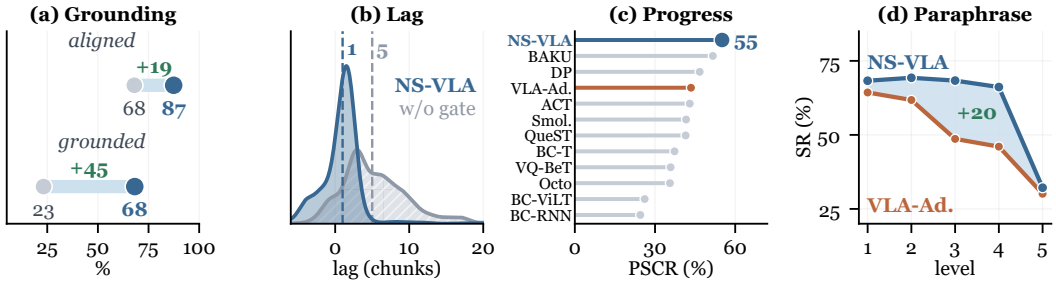


Figure 5: Encoder: (a) the gate grounds pointer advances in milestones; (b) lag from an advance to its milestone; (c) PSCR of all twelve methods; (d) SR on LIBERO-Plus paraphrases by level.

The precondition gate grounds the pointer. Of NS-VLA's advances, 68.2% see their milestone at most three chunks later, against 23.2% without the gate, and the final pointer lies within one stage of the achieved progress in 87.4% of episodes (a); median lag falls from five chunks to one (b).

Structured execution reaches further and absorbs paraphrases. NS-VLA attains the highest PSCR of all twelve methods, 55.1 against 51.6 for BAKU, so it completes the most plan stages (c). On reworded LIBERO-Plus instructions it stays above its VLA-Adapter backbone at every difficulty level, by up to 20.2 points, and keeps its SR at 66–69% through level 4 of the five levels (d).

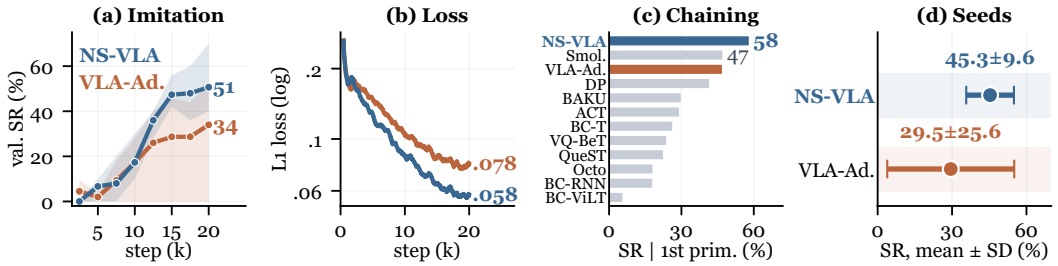


Figure 6: Solver: (a) validation SR (band: range) and (b) training loss over three seeds; (c) task SR once the first primitive is done, for all methods; (d) test SR over three training seeds.

5.5 Analysis of the Neuro-Symbolic Solver (RQ4)

As shown in Figure 6, we test whether sub-instructions make five demonstrations easier to imitate, following imitation and loss over three seeds and each method's success after the first primitive.

The Solver learns faster and further. Over three training seeds, NS-VLA's checkpoints lead from 12.5k steps on and finish at 51% validation SR against 34% (a), with lower training loss after warmup, 0.058 against 0.078 (b), as the Solver pools the segments of each primitive across all ten tasks.

It chains primitives stably. Once the first primitive is done, NS-VLA completes 57.9% of episodes, ahead of SmolVLA (47.0%) and every other compared method (c), and over all three training seeds it averages 45.3 ± 9.6 against 29.5 ± 25.6 for its VLA-Adapter backbone (d), so its standard deviation across these seeds is well under half that of the backbone on the same test episodes.

5.6 Analysis of Hierarchical Joint Policy Optimization (RQ5)

As shown in Figure 7, we test whether the hierarchy turns online interaction into a learning signal, following every online iteration and the validation-based model selection over all online runs.

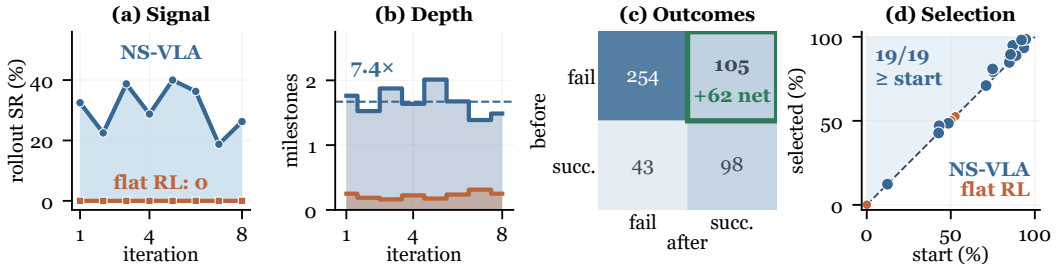


Figure 7: Online stage of HJPO: (a) rollout SR and (b) milestones per rollout in every iteration; (c) test outcomes before and after the stage; (d) selected against starting validation SR of all runs.

Structure makes rewards reachable. NS-VLA succeeds in 18.8–40.0% of its rollouts in every iteration, so each filtered update has successes to learn from, whereas flat RL on VLA-Adapter collects none (a), and NS-VLA's rollouts reach 1.67 milestones on average, $7.4\times$ the flat baseline (b).

The online stage converts failures safely. On the 500 test episodes it turns 105 failures into successes, a net gain of 62 that lifts SR from 28.2 to 40.6 (c), and across all 19 online runs the selected model never falls below its supervised start on validation and gains up to 8 points over it (d).

5.7 Case study (RQ6)

As shown in Figure 8, we replay the long-horizon task "put the black bowl in the bottom drawer of the cabinet and close it" and a LIBERO-Plus paraphrase of it from the same initial state.

NS-VLA executes the plan stage by stage. The plan has three stages, S1 "pick up the black bowl", S2 "place the black bowl in the bottom drawer of the cabinet" and S3 "close the bottom drawer of the cabinet". NS-VLA picks up the bowl, places it in the drawer and closes it, advancing within two chunks of each milestone, and succeeds at step 211, while VLA-Adapter reaches no stage (a).

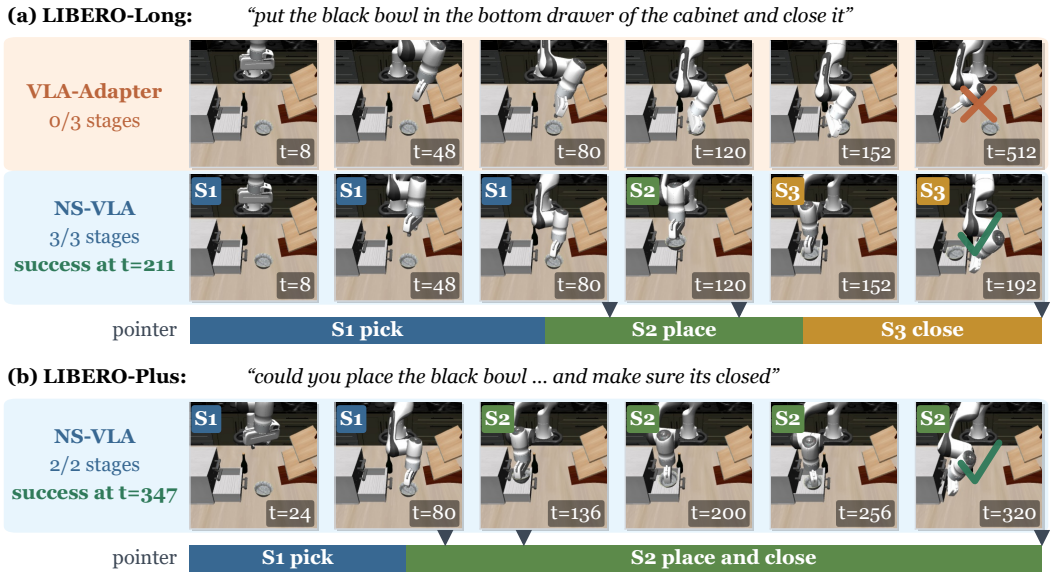


Figure 8: Case study from the same initial state on (a) the original instruction and (b) a paraphrase: badges give the active plan stage Sk, the bar shows the pointer over time and ▼ marks milestones.

The plan carries over to a paraphrase. The reworded instruction "could you place the black bowl … and make sure its closed" is parsed into a pick stage and a place stage. The pointer leaves the pick stage as soon as the gripper holds the bowl, and NS-VLA succeeds at step 347 (b), so the plan absorbs the rewording without any retraining of the grammar parser or the action policy.

5.8 Performance on real-world tasks (RQ7)

As shown in Figure 9, we deploy NS-VLA on a myCobot 320pi arm against VLA-Adapter to test whether the gains we observe in simulation also transfer to real hardware.

The tasks span single and compositional skills. With a wrist and a third-person RealSense D435i camera, both policies run the same three Pick, three Move and four Place tasks, each repeated ten times under randomized object placements, for 100 trials per policy in total (a).

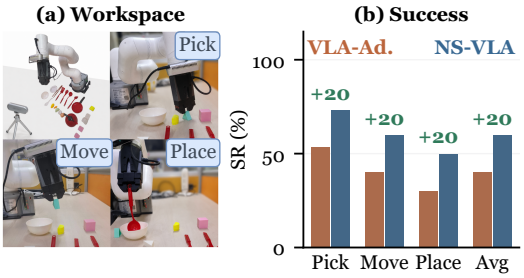


Figure 9: Real robot: (a) setup; (b) success rate.

The gains transfer to the real robot. NS-VLA reaches 73.3 on Pick, 60.0 on Move and 50.0 on Place, ahead of VLA-Adapter by 20.0 points in every category, and averages 60.0 over all 100 trials against 40.0, a relative gain of 50%, with the compositional Place tasks, which chain pick and place, keeping the full margin (b).

6 Conclusion

We presented NS-VLA, a neuro-symbolic vision-language-action framework that exposes the compositional structure of manipulation instructions to a chunked action policy. The Encoder compiles each instruction into a primitive plan tracked by a precondition-gated monotone stage pointer, the Solver conditions an unchanged backbone on the active sub-instruction, and HJPO fits the pointer on demonstration segments and refines the action policy by success-filtered advantage-weighted regression. By construction, every stage transition is certified, one-slot plans reduce exactly to the backbone, and online refinement never falls below its supervised start on validation. Empirically, NS-VLA ranks first or ties for first in every column of LIBERO and LIBERO-Plus, with the largest margin on long-horizon tasks, and raises real-robot success from 40.0% to 60.0%. Explicit symbolic structure thus offers an effective route to data-efficient, robust and interpretable VLA policies.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (Grant No. 62472043). We thank the anonymous reviewers and area chairs for their constructive feedback, which helped improve the clarity and rigor of this paper. The authors declare no competing interests.

References

[1] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 166–175. PMLR, 2017. URL `https://proceedings.mlr.press/v70/andreas17a.html`.

[2] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 1726–1734. AAAI Press, 2017.

[3] Zechen Bai, Chen Gao, and Mike Zheng Shou. EVOLVE-VLA: Test-time training from environment feedback for vision-language-action models. arXiv:2512.14666, 2025.

[4] Suneel Belkhale, Tianli Ding, Ted Xiao, Pierre Sermanet, Quan Vuong, Jonathan Tompson, Yevgen Chebotar, Debidatta Dwibedi, and Dorsa Sadigh. RT-H: Action hierarchies using language. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.049.

[5] 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 Joseph Lim, Shuran Song, and Hae-Won Park, editors, *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 17–40. PMLR, 2025. URL `https://proceedings.mlr.press/v305/black25a.html`.

[6] 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 *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.010.

[7] 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 *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.025.

[8] Qingwen Bu, Hongyang Li, Li Chen, Jisong Cai, Jia Zeng, Heming Cui, Maoqing Yao, and Yu Qiao. Towards synergistic, generalized, and efficient dual-system for robotic manipulation. arXiv:2410.08001, 2025.

[9] Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. UniVLA: Learning to act anywhere with task-centric latent actions. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.014.

[10] Mateo Guaman Castro, Sidharth Rajagopal, Daniel Gorbatov, Matt Schmittle, Rohan Baijal, Octi Zhang, Rosario Scalise, Sidharth Talia, Emma Romig, Celso de Melo, Byron Boots, and Abhishek Gupta. VAMOS: A hierarchical vision-language-action model for capability-modulated and steerable navigation. arXiv:2510.20818, 2025.

[11] Jun Cen, Siteng Huang, Yuqian Yuan, Kehan Li, Hangjie Yuan, Chaohui Yu, Bohan Hou, Yuming Jiang, Jiayan Guo, Xin Li, Hao Luo, Fan Wang, Deli Zhao, and Hao Chen. WorldVLA: A unified vision-language-action and world model. In *Advances in Neural Information Processing Systems*, 2026.

[12] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025. doi: 10.1177/02783649241273668.

[13] Wonje Choi, Jooyoung Kim, and Honguk Woo. NeSyPr: Neurosymbolic proceduralization for efficient embodied reasoning. In *Advances in Neural Information Processing Systems*, volume 38, pages 174484–174504, 2025. doi: 10.52202/085713-5804.

[14] Artur d'Avila Garcez and Luis C. Lamb. Neurosymbolic AI: The 3rd wave. *Artificial Intelligence Review*, 56(11):12387–12406, 2023. doi: 10.1007/s10462-023-10448-w.

[15] Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. In S. Hanson, J. Cowan, and C. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5. Morgan-Kaufmann, 1992. URL `https://proceedings.neurips.cc/paper_files/paper/1992/file/d14220ee66aeec73c49038385428ec4c-Paper.pdf`.

[16] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. PaLM-E: An embodied multimodal language model. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 8469–8488. PMLR, 2023. URL `https://proceedings.mlr.press/v202/driess23a.html`.

[17] Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B. Tenenbaum. DreamCoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, PLDI 2021, pages 835–850. Association for Computing Machinery, 2021. doi: 10.1145/3453483.3454080.

[18] Senyu Fei, Siyin Wang, Li Ji, Ao Li, Shiduo Zhang, Liming Liu, Jinlong Hou, Jingjing Gong, Xianzhong Zhao, and Xipeng Qiu. SRPO: Self-referential policy optimization for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6718–6728, 2026.

[19] Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: A progressive robustness benchmark for visual-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 38574–38583, 2026.

[20] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *Proceedings of the 1st Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 357–368. PMLR, 2017.

[21] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 158–168. PMLR, 2021. URL https://proceedings.mlr.press/v164/florence22a.html.

[22] Scott Fujimoto and Shixiang Gu. A minimalist approach to offline reinforcement learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 20132–20145. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/a8166da05c5a094f7dc03724b41886e5-Paper.pdf.

[23] Chen-Xiao Gao, Chenyang Wu, Mingjun Cao, Chenjun Xiao, Yang Yu, and Zongzhang Zhang. Behavior-regularized diffusion policy optimization for offline reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 18630–18657. PMLR, 2025.

[24] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10764–10799. PMLR, 2023. URL https://proceedings.mlr.press/v202/gao23f.html.

[25] Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. Neural module networks for reasoning over text. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SygWvAVFPr.

[26] Siddhant Haldar, Zhuoran Peng, and Lerrel Pinto. BAKU: An efficient transformer for multi-task policy learning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 141208–141239. Curran Associates, Inc., 2024. doi: 10.52202/079017-4484.

[27] ByungOk Han, Jaehong Kim, and Jinhyeok Jang. A dual process VLA: Efficient robotic manipulation leveraging VLM. arXiv:2410.15549, 2024.

[28] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1769–1782. PMLR, 2022.

[29] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. VoxPoser: Composable 3D value maps for robotic manipulation with language models. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 540–562. PMLR, 2023.

[30] Chia-Yu Hung, Qi Sun, Pengfei Hong, Amir Zadeh, Chuan Li, U-Xuan Tan, Navonil Majumder, and Soujanya Poria. NORA: A small open-sourced generalist vision language action model for embodied tasks. arXiv:2504.19854, 2025.

[31] Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J. Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. Do as I can, not as I say: Grounding language in robotic affordances. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning*

Research, pages 287–318. PMLR, 2022. URL https://proceedings.mlr.press/v205/ichter23a.html.

[32] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. BC-Z: Zero-shot task generalization with robotic imitation learning. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 991–1002. PMLR, 2021. URL https://proceedings.mlr.press/v164/jang22a.html.

[33] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X.

[34] 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, editors, *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR, 2024. URL https://proceedings.mlr.press/v270/kim25c.html.

[35] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.017.

[36] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/f442d33fa06832082290ad8544a8da27-Paper.pdf.

[37] Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 26991–27008. PMLR, 2024.

[38] Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. In *International Conference on Learning Representations*, 2019.

[39] 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. In *International Conference on Learning Representations*, 2026.

[40] Hengtao Li, Pengxiang Ding, Runze Suo, Yihao Wang, Zirui Ge, Dongyuan Zang, Kexian Yu, Mingyang Sun, Hongyin Zhang, Donglin Wang, and Weihua Su. VLA-RFT: Vision-language-action reinforcement fine-tuning with verified rewards in world simulators. arXiv:2510.00406, 2025.

[41] Wei Li, Renshan Zhang, Rui Shao, Jie He, and Liqiang Nie. CogVLA: Cognition-aligned vision-language-action models via instruction-driven routing & sparsification. In *Advances in Neural Information Processing Systems*, volume 38, pages 137646–137675, 2025. doi: 10.52202/085713-4594.

[42] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500, 2023. doi: 10.1109/ICRA48891.2023.10160591.

[43] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 44776–44791. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/8c3c666820ea055a77726d66fc7d447f-Paper-Datasets_and_Benchmarks.pdf.

[44] Jiaming Liu, Mengzhen Liu, Zhenyu Wang, Pengju An, Xiaoqi Li, Kaichen Zhou, Senqiao Yang, Renrui Zhang, Yandong Guo, and Shanghang Zhang. RoboMamba: Efficient vision-language-action model for robotic reasoning and manipulation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 40085–40110. Curran Associates, Inc., 2024. doi: 10.52202/079017-1266.

[45] Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. VLA-RL: Towards masterful and general robotic manipulation with scalable reinforcement learning. arXiv:2505.18719, 2025.

[46] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1678–1690. PMLR, 2021.

[47] Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. DeepProbLog: Neural probabilistic logic programming. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/dc5d637ed5e62c36ecb73b654b05ba2a-Paper.pdf.

[48] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJgMlhRctm.

[49] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022. doi: 10.1109/LRA.2022.3180108.

[50] Oier Mees, Dibya Ghosh, Karl Pertsch, Kevin Black, Homer Rich Walke, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.090.

[51] Atharva Mete, Haotian Xue, Albert Wilcox, Yongxin Chen, and Animesh Garg. QueST: Self-supervised skill abstractions for learning continuous control. In *Advances in Neural Information Processing Systems*, volume 37, pages 4062–4089, 2024. doi: 10.52202/079017-0133.

[52] Ofir Nachum, Shixiang (Shane) Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/e6384711491713d29bc63fc5eeb5ba4f-Paper.pdf.

[53] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. AWAC: Accelerating online reinforcement learning with offline datasets. arXiv:2006.09359, 2020.

[54] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning*, pages 278–287, 1999.

[55] 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. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903, 2024. doi: 10.1109/ICRA57147.2024.10611477.

[56] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc., 2022. URL `https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf`.

[57] Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.248.

[58] Jeongeun Park, Jihwan Yoon, Byungwoo Jeon, Juhan Park, Jinwoo Shin, Namhoon Cho, Kyungjae Lee, Sangdoo Yun, and Sungjoon Choi. Hierarchical vision language action model using success and failure demonstrations. arXiv:2512.03913, 2025.

[59] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. arXiv:1910.00177, 2019.

[60] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision-language-action models. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.012.

[61] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/a85b405ed65c6477a4fe8302b5e06ce7-Paper-Conference.pdf.

[62] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Giménez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *Transactions on Machine Learning Research*, 2022. URL https://openreview.net/forum?id=1ikK0kHjvj.

[63] Tim Rocktäschel and Sebastian Riedel. Learning knowledge base inference with neural theorem provers. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, pages 45–50, 2016.

[64] SAM 3D Team, Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J. Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, Aohan Lin, Jiawei Liu, Ziqi Ma, Anushka Sagar, Bowen Song, Xiaodong Wang, Jianing Yang, Bowen Zhang, Piotr Dollár, Georgia Gkioxari, Matt Feiszli, and Jitendra Malik. SAM 3D: 3Dfy anything in images. arXiv:2511.16624, 2025.

[65] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 68539–68551. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/d842425e4bf79ba039352da0f658a906-Paper-Conference.pdf.

[66] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897. PMLR, 2015. URL https://proceedings.mlr.press/v37/schulman15.html.

[67] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

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

[69] 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:2402.03300, 2024.

[70] Yide Shentu, Philipp Wu, Aravind Rajeswaran, and Pieter Abbeel. From LLMs to actions: Latent codes as bridges in hierarchical robot control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8539–8546, 2024.

[71] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. SmolVLA: A vision-language-action model for affordable and efficient robotics. arXiv:2506.01844, 2025.

[72] Zirui Song, Guangxian Ouyang, Mingzhe Li, Yuheng Ji, Chenxi Wang, Zixiang Xu, Zeyu Zhang, Xiaoqing Zhang, Qian Jiang, Fengxian Ji, Zhenhao Chen, Zhongzhi Li, and Xiuying Chen. ManipLVM-R1: Reinforcement learning for reasoning in embodied manipulation with large vision-language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(22):18558–18566, 2026. doi: 10.1609/aaai.v40i22.38922.

[73] Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1): 181–211, 1999. doi: 10.1016/S0004-3702(99)00052-1.

[74] Shuhan Tan, Kairan Dou, Yue Zhao, and Philipp Krähenbühl. Interactive post-training for vision-language-action models. In *Workshop on Foundation Models Meet Embodied Agents at CVPR 2025*, 2025. URL https://openreview.net/forum?id=ytGamErI9m.

[75] Jidong Tian, Yitian Li, Wenqing Chen, Liqiang Xiao, Hao He, and Yaohui Jin. Weakly supervised neural symbolic learning for cognitive tasks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(5):5888–5896, 2022. doi: 10.1609/aaai.v36i5.20533.

[76] Yang Tian, Sizhe Yang, Jia Zeng, Ping Wang, Dahua Lin, Hao Dong, and Jiangmiao Pang. Predictive inverse dynamics models are scalable learners for robotic manipulation. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=meRCKuUpmc.

[77] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. FeUdal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3540–3549. PMLR, 2017.

[78] Lirui Wang, Xinlei Chen, Jialiang Zhao, and Kaiming He. Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 124420–124450. Curran Associates, Inc., 2024. doi: 10.52202/079017-3952.

[79] Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, Siteng Huang, Yifan Tang, Wenhui Wang, Ru Zhang, Jianyi Liu, and Donglin Wang. VLA-Adapter: An effective paradigm for tiny-scale vision-language-action model. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(22): 18638–18646, 2026. doi: 10.1609/aaai.v40i22.38931.

[80] Eric Wiewiora. Potential-based shaping and Q-value initialization are equivalent. *Journal of Artificial Intelligence Research*, 19:205–208, 2003.

[81] Xiaoqian Wu, Yong-Lu Li, Jianhua Sun, and Cewu Lu. Symbol-LLM: Leverage language models for symbolic system in visual human activity reasoning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 29680–29691. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/5edb57c05c81d04beb716ef1d542fe9e-Paper-Conference.pdf.

[82] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. arXiv:1911.11361, 2019.

[83] Fangzhi Xu, Qiushi Sun, Kanzhi Cheng, Jun Liu, Yu Qiao, and Zhiyong Wu. Interactive evolution: A neural-symbolic self-training framework for large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12975–12993. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.635.

[84] Shunyu Yao, Jeffrey Zhao, Dian Yu, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

[85] Angen Ye, Zeyu Zhang, Boyuan Wang, Xiaofeng Wang, Dapeng Zhang, and Zheng Zhu. VLA-R1: Enhancing reasoning in vision-language-action models. arXiv:2510.01623, 2025.

[86] Yifan Ye, Jun Cen, Jing Chen, and Zhihe Lu. Self-evolved imitation learning in simulated world. *IEEE Robotics and Automation Letters*, 11(6):6967–6974, 2026. doi: 10.1109/LRA.2026.3683594.

[87] Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard, editors, *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 3157–3181. PMLR, 2024. URL `https://proceedings.mlr.press/v270/zawalski25a.html`.

[88] Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, Ankur Handa, Tsung-Yi Lin, Gordon Wetzstein, Ming-Yu Liu, and Donglai Xiang. CoT-VLA: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1702–1713, 2025.

[89] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.016.

[90] Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 3D-VLA: A 3D vision-language-action generative world model. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 61229–61245. PMLR, 2024. URL `https://proceedings.mlr.press/v235/zhen24a.html`.

[91] Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. TraceVLA: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. In *International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=b1CVu9l5GO`.

[92] 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, editors, *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR, 2023. URL `https://proceedings.mlr.press/v229/zitkovich23a.html`.

A Plan grammar and demonstration segmentation

The Encoder maps an instruction to a primitive plan with a deterministic grammar, renders each primitive as a sub-instruction, and segments demonstrations into stages by gripper events. None of the three requires annotation; Spatial, Object and Goal run the one-slot plan (Figure 10).

Encoder: instruction → plan

```
instruction turn on the stove and
            put the moka pot on it
clauses     [turn on the stove]
            [put the moka pot on it]
anaphora    it -> the stove
plan        (turn on, stove)
            (pick, moka pot)
            (place_on, (moka pot, stove))
```

Solver: bridged input per stage

```
[S1] put the black bowl in the bottom
     drawer of the cabinet and close it.
     now: pick up the black bowl
[S2] ... now: place the black bowl in
     the bottom drawer of the cabinet
[S3] ... now: close the bottom drawer
     of the cabinet
```

Rendering of a primitive

```
(pick, o)          pick up the o
(place_rho, (o,g)) place the o rho the g
other primitives   normalized clause
```

One-slot plan (M = 1)

```
S/O/G: pick up the alphabet soup and
place it in the basket -> x itself
```

Figure 10: Inputs of the Encoder and the Solver: a plan compiled from an instruction, the rendering of primitives as sub-instructions, and the bridged input the backbone reads at every stage.

A.1 Grammar and primitive vocabulary

Clause splitting. The parser lower-cases the instruction, splits it at coordinating conjunctions and normalizes each clause to a verb frame. Three rewrites make the clauses self-contained. A shared verb is distributed over coordinated objects, and a plural object is expanded into its instances ("put both moka pots on the stove" yields one clause per pot). A pronoun or elided target takes the most recent compatible referent ("…in the microwave and close it" → "close the microwave"). Spatial qualifiers of object and target are kept verbatim ("the black bowl between the plate and the ramekin").

Vocabulary. A clause $\texttt{put}(o, \rho, g)$ becomes the pair $(\texttt{pick}, o)$, $(\texttt{place}_\rho, (o, g))$, with ρ one of *on*, *in*, *on top of* or a spatial relation such as *to the right of*. Every other clause yields one primitive, `open`, `close`, `turn on`, `turn off` or `push`, whose argument is the articulated object and, for `push`, the goal. The vocabulary is read off the training instructions, and objects remain free-text arguments. It is partitioned by the gripper behavior that the precondition gate of the Encoder checks:

Class	Primitives	Gripper signature
$\mathcal{U}_\text{g}$ (grasp)	`pick`	ends with an object held
$\mathcal{U}_\text{r}$ (grasp–release)	`place`, `turn on`, `turn off`	a grasp followed by a release
$\mathcal{U}_\text{f}$ (gripper-free)	`open`, `close`, `push`	no gripper event

Rendering and bridged input. A primitive is rendered as "pick up the o", "place the o ρ the g" or its normalized clause, and the Solver reads "x. `now:` $\text{text}(u^{(m)})$"; a one-slot plan reads x itself. Table 4 lists every LIBERO-Long plan, fixed across training, rollouts and evaluation.

A.2 Demonstration segmentation

Gripper events. Let $g_1, \ldots, g_n \in \{0, 1\}$ be the gripper channel of a demonstration's actions (1 = open). A state change is registered when the new value persists for three consecutive frames, and the event is time-stamped at the first frame of that run. Pairing every close event at frame c with the next open event at frame r (or $r = n$ if the gripper stays closed) gives candidate grasps, of which we keep only those held long enough to lift and safely transport an object to its target location:

$$\mathcal{G} \;=\; \big\{(c_k, r_k) \;:\; r_k - c_k \geq 30\big\}, \qquad c_1 < r_1 \leq c_2 < r_2 \leq \cdots, \tag{11}$$

Table 4: Plans of the ten LIBERO-Long instructions; "|" separates consecutive sub-instructions.

Task	Instruction	Plan (sub-instructions)
0	put both the alphabet soup and the tomato sauce in the basket	pick up the alphabet soup | place the alphabet soup in the basket | pick up the tomato sauce | place the tomato sauce in the basket
1	put both the cream cheese box and the butter in the basket	pick up the cream cheese box | place the cream cheese box in the basket | pick up the butter | place the butter in the basket
2	turn on the stove and put the moka pot on it	turn on the stove | pick up the moka pot | place the moka pot on the stove
3	put the black bowl in the bottom drawer of the cabinet and close it	pick up the black bowl | place the black bowl in the bottom drawer of the cabinet | close the bottom drawer of the cabinet
4	put the white mug on the left plate and put the yellow and white mug on the right plate	pick up the white mug | place the white mug on the left plate | pick up the yellow and white mug | place the yellow and white mug on the right plate
5	pick up the book and place it in the back compartment of the caddy	pick up the book | place the book in the back compartment of the caddy
6	put the white mug on the plate and put the chocolate pudding to the right of the plate	pick up the white mug | place the white mug on the plate | pick up the chocolate pudding | place the chocolate pudding to the right of the plate
7	put both the alphabet soup and the cream cheese box in the basket	pick up the alphabet soup | place the alphabet soup in the basket | pick up the cream cheese box | place the cream cheese box in the basket
8	put both moka pots on the stove	pick up the moka pot | place the moka pot on the stove | pick up the other moka pot | place the other moka pot on the stove
9	put the yellow and white mug in the microwave and close it	pick up the yellow and white mug | place the yellow and white mug in the microwave | close the microwave

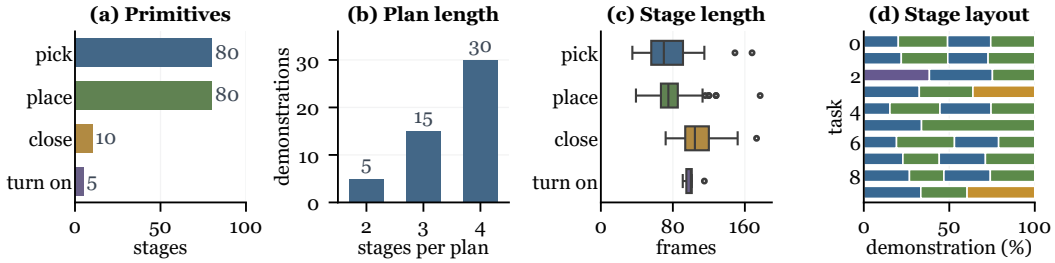


Figure 11: Automatic segmentation of the 50 LIBERO-Long demonstrations: (a) stages per primitive; (b) plan length per demonstration; (c) stage length in frames; (d) mean stage layout of every task.

so brief closures from regrasp adjustments never create spurious stages in the plan.

Stage boundaries. The gripper-consuming clauses of the plan (pick-and-place and `turn on`/`turn off`) are matched in plan order, one-to-one, to the kept grasps in the set $\mathcal{G}$. A cursor $s \leftarrow 1$ then sweeps the plan in a single pass and emits the start frame $s^{(m)}$ of every plan stage:

$$\begin{aligned} &\texttt{put}(o, \rho, g) \text{ with grasp } (c_k, r_k): && s^{(\texttt{pick})} = s, \;\; s^{(\texttt{place})} = c_k, \;\; s \leftarrow r_k; \\ &\texttt{turn on}/\texttt{turn off} \text{ with grasp } (c_k, r_k): && s^{(\texttt{turn})} = s, \;\; s \leftarrow r_k; \\ &u \in \mathcal{U}_{\mathrm{f}}: && s^{(u)} = s, \end{aligned} \tag{12}$$

and stage m covers frames $[s^{(m)}, s^{(m+1)})$, the last stage ending at frame n. A pick segment therefore ends exactly when the grasp closes, a place segment when it opens, and a gripper-free primitive that follows the last grasp starts at the final release. Every LIBERO-Long training demonstration is segmented by this rule; a demonstration whose grasps and gripper-consuming clauses do not match one-to-one enters training with its original instruction, so all demonstrations contribute to imitation.

Supervision from the segments. Frame i of stage m is paired with $\tilde{x}^{(m)}$ for the Solver and yields stay/advance examples for the pointer. The dwell feature under pointer j is $\lfloor (i - s^{(j)})/H \rfloor$, the number of chunks since the queried stage began in the demonstration. All 50 LIBERO-Long training demonstrations are segmented into the stages of their plans without a manual label (Figure 11).

B Proofs

Conventions. Decisions $t = 1, \dots, T$ start at pointer $m_0 = 1$, $d_{\min} \geq 1$, and the dwell obeys

$$d_1 = 0, \qquad d_{t+1} \;=\; \big(d_t + 1\big)\,\mathbf{1}\big[m_t = m_{t-1}\big], \tag{13}$$

so it counts the decisions since the last advance. We write $\Delta_t = m_t - m_{t-1}$ for the pointer increment, $t_m = \min\{t : m_t = m\}$ for the first decision of slot m (so $t_1 = 1$, as $d_1 < d_{\min}$), and $\mathcal{S}_m = \{t : m_t = m\}$ for the segment of slot m. The chunk of decision t is computed after the pointer update, from $\tilde{x}_t = \tilde{x}^{(m_t)}$. The history $\mathcal{H}_t$ holds x, the observations up to t and the earlier chunks.

B.1 Proof of Proposition 4.1

The statement is split into four claims: the pointer is a monotone unit-step counter, its level sets are contiguous segments in plan order, every non-final segment respects the minimum dwell, and every exit is certified by the gate. None of the arguments uses the learned values of ϕ or θ.

Lemma B.1 (Binary gate and increment)**.** *For every t and $u \in \mathcal{U}$, $\Gamma_t(u) \in \{0,1\}$ and $\Delta_t \in \{0,1\}$.*

Proof. The classes $\mathcal{U}_{\mathrm{g}}, \mathcal{U}_{\mathrm{r}}, \mathcal{U}_{\mathrm{f}}$ partition $\mathcal{U}$, so exactly one class indicator of the gate equals one and

$$\Gamma_t(u) \;\in\; \Big\{\, 1,\;\; h_t,\;\; (1-\kappa_t) \max_{t_{m_{t-1}} \leq t' \leq t} h_{t'} \,\Big\} \;\subseteq\; \{0,1\}, \tag{14}$$

because $\kappa_t \in \{0,1\}$ and $h_{t'} = \kappa_{t'}\,\mathbf{1}[w_{t'} > w_{\min}] \in \{0,1\}$, and products and maxima of $\{0,1\}$-valued quantities stay in $\{0,1\}$. By the pointer update, Δ_t is the product of the four $\{0,1\}$-valued factors $\mathbf{1}[m_{t-1} < M]$, $\mathbf{1}[d_t \geq d_{\min}]$, $\Gamma_t(u^{(m_{t-1})})$ and $\mathbf{1}[\sigma(g_\phi(z_t)) > \frac{1}{2}]$, hence $\Delta_t \in \{0,1\}$. □

Claim 1: monotone unit-step counter. By Lemma B.1, $m_{t-1} \leq m_t \leq m_{t-1} + 1$. If $m_{t-1} = M$, the factor $\mathbf{1}[m_{t-1} < M]$ vanishes; otherwise $m_t \leq m_{t-1} + 1 \leq M$. Induction from $m_0 = 1$ gives

$$1 \;\leq\; m_0 \;\leq\; m_1 \;\leq\; \cdots \;\leq\; m_T \;\leq\; M, \qquad m_t - m_{t-1} \in \{0,1\} \quad \text{for all } t. \tag{15}$$

Claim 2: contiguous segments in plan order. Let $t < t' < t''$ with $t, t'' \in \mathcal{S}_m$. Monotonicity gives $m = m_t \leq m_{t'} \leq m_{t''} = m$, so $t' \in \mathcal{S}_m$ and every $\mathcal{S}_m$ is an interval of decisions. If $t \in \mathcal{S}_m$ and $t' \in \mathcal{S}_{m'}$ with $m < m'$, then $t \geq t'$ would imply $m = m_t \geq m_{t'} = m'$, a contradiction, so $t < t'$ and the intervals appear in plan order. Unit steps visit every slot $1, \dots, m_T$ in turn, and

$$\{1, \dots, T\} \;=\; \bigsqcup_{m=1}^{m_T} \mathcal{S}_m, \qquad \mathcal{S}_m = \big[\, t_m,\; t_{m+1} - 1 \,\big]\;\; (m < m_T), \qquad \mathcal{S}_{m_T} = \big[\, t_{m_T},\; T \,\big], \tag{16}$$

a partition into $m_T \leq M$ nonempty contiguous segments. Each chunk of $\mathcal{S}_m$ uses $\tilde{x}^{(m)}$, as advances re-predict with the new pointer, so segment m runs under sub-instruction $\mathrm{text}(u^{(m)})$.

Claim 3: minimum dwell. Take a non-final segment, $m < m_T$; its exit at t_{m+1} has $\Delta_{t_{m+1}} = 1$ and therefore $d_{t_{m+1}} \geq d_{\min}$. Unrolling the dwell recursion over the decisions of the segment gives

$$d_t \;=\; t - t_m - \mathbf{1}[m \geq 2] \quad \big(t_m < t \leq t_{m+1}\big), \qquad \big|\mathcal{S}_m\big| \;=\; d_{t_{m+1}} + \mathbf{1}[m \geq 2], \tag{17}$$

since the advance at t_m resets the dwell at $t_m + 1$. For every $m < m_T$, $|\mathcal{S}_m| \geq d_{t_{m+1}} \geq d_{\min}$, i.e. every non-final segment spans at least $d_{\min}$ decisions and at least $H d_{\min}$ environment steps. With $d_{\min} = 2$ and $H = 8$, every non-final segment thus lasts at least 16 environment steps.

Claim 4: certified exits. The exit at t_{m+1} requires $\Gamma_{t_{m+1}}(u^{(m)}) = 1$, which reads by class as

$$u^{(m)} \in \mathcal{U}_{\mathrm{g}} \;\Rightarrow\; h_{t_{m+1}} = 1, \quad u^{(m)} \in \mathcal{U}_{\mathrm{r}} \;\Rightarrow\; \kappa_{t_{m+1}} = 0 \wedge \exists\, t' \in \big[t_m, t_{m+1}\big] : h_{t'} = 1, \tag{18}$$

so a `pick` slot is left only while the gripper is commanded closed with the fingers held apart by an object, and a grasp–release slot only after an object was held within the slot and the gripper was reopened. A gripper-free slot is certified by the classifier and the dwell. A `pick` slot thus never exits on a close command alone, since fingers closing on air fail the width test. Claims 1–4 hold for every rollout and every parameter value of both factors, which proves the full proposition. □

B.2 Proof of Proposition 4.2

The high-level factor is a point mass. With $M = 1$ the factor $\mathbf{1}[m_{t-1} < 1]$ of the pointer update vanishes for every t, so by induction $m_t = 1$, independently of ϕ, the gate and the features z_t. Hence

$$\pi^{\mathrm{h}}_{\phi}\big(u \mid \mathcal{H}_t\big) \;=\; \mathbf{1}\big[u = u^{(1)}\big] \qquad \text{for every } \phi \text{ and every history } \mathcal{H}_t. \tag{19}$$

The low-level factor is the backbone. For a one-slot plan the bridged input is the instruction, $\tilde{x}^{(1)} = x$, so the Solver is the backbone's Laplace chunk policy and the hierarchical policy splits as

$$\pi_{\Theta}\big(u, A_t \mid \mathcal{H}_t\big) \;=\; \mathbf{1}\big[u = u^{(1)}\big]\; \pi_{\theta}\big(A_t \mid I_t, S_t, x\big), \tag{20}$$

so, summed over u, the chunk marginal of NS-VLA is exactly the backbone policy at every history.

Trajectory laws coincide. Let $\mathcal{K}(s_{t+1}, o_{t+1} \mid s_t, \hat{A}_t)$ be the H-step kernel of open-loop execution of the chunk $\hat{A}_t$ from state s_t, and μ_0 the initial-state law. Any chunk policy π induces the rollout law

$$\mathbb{P}_{\pi}\big(s_{1:T+1}, o_{1:T+1}, \hat{A}_{1:T}\big) \;=\; \mu_0(s_1, o_1) \prod_{t=1}^{T} \pi\big(\hat{A}_t \mid \mathcal{H}_t\big)\, \mathcal{K}\big(s_{t+1}, o_{t+1} \mid s_t, \hat{A}_t\big). \tag{21}$$

NS-VLA and its backbone share μ_0 and $\mathcal{K}$ and place the same conditional law on every chunk, including the exploration noise of the online stage, added to the predicted chunk in the same way. Induction over t shows that the joint laws of $(s_{\le t}, o_{\le t}, \hat{A}_{<t})$ coincide for every t; hence the trajectory laws coincide, and so do the law of the binary return $R(\tau)$ and every success rate.

Training objectives coincide. A one-slot plan has a single segment and $\tilde{x}^{(1)} = x$, so $x_i^{\xi} = x$ for both values of ξ, and in its cross-task decomposition every bridged loss equals the native one:

$$\mathcal{L}_{\mathrm{SFT}}(\theta) \;=\; \tfrac{1}{2}\,\mathcal{L}_{\mathrm{nat}}(\theta) + \tfrac{1}{2}\sum_{u} \lambda_u\, \mathcal{L}_u(\theta) \;=\; \mathbb{E}_{(x,\, o_i,\, A_i^{\star}) \sim \mathcal{D}} \big\| f_{\theta}\big(I_i, S_i, x\big) - A_i^{\star} \big\|_1, \tag{22}$$

the backbone's imitation loss. The pointer is never queried, so the pointer loss plays no role. In the online stage every tuple of $\mathcal{D}^+$ carries the input x and, by the previous step, is drawn from the backbone's own trajectory law, so the HJPO objective is the backbone's filtered regression with the same demonstration anchor, and validation selection compares the very same candidates. □

Derivation of cross-task sharing. Let ν be the uniform law on frames and $\mathrm{type}(i)$ the primitive of frame i (x if one-slot), so $\lambda_u = \nu(\mathrm{type} = u)$. As ξ is independent, the tower rule gives

$$\begin{aligned} \mathcal{L}_{\mathrm{SFT}}(\theta) \;&=\; \tfrac{1}{2}\,\mathbb{E}_{\nu}\big[\ell_{\theta}(i; x)\big] + \tfrac{1}{2}\,\mathbb{E}_{\nu}\big[\ell_{\theta}(i; \tilde{x}^{(m)})\big] \\ &=\; \tfrac{1}{2}\,\mathcal{L}_{\mathrm{nat}}(\theta) + \tfrac{1}{2}\sum_{u} \lambda_u\, \mathbb{E}_{\nu}\big[\ell_{\theta}(i; \tilde{x}^{(m)}) \;\big|\; \mathrm{type}(i) = u\big], \end{aligned} \tag{23}$$

with $\ell_{\theta}(i; x') = \|f_{\theta}(I_i, S_i, x') - A_i^{\star}\|_1$. The conditional expectation is $\mathcal{L}_u(\theta)$, the mean bridged loss over all frames of primitive u, pooled over tasks, which proves the decomposition.

B.3 Proof of Proposition 4.3

Factorized likelihood. Given the stage features as fixed inputs, the pointer is a deterministic function of the stay/advance decisions $y_{1:T}$ and the gate, and the primitive executed at decision t is $u^{(m_t)}$. Writing the likelihood of a rollout with the decision $y_t \sim p_{\phi}(\cdot \mid z_t)$ made before the chunk,

$$\pi_{\Theta}(\tau) \;=\; \mu_0(s_1, o_1) \prod_{t=1}^{T} p_{\phi}\big(y_t \mid z_t\big)\, \pi^{\mathrm{l}}_{\theta}\big(\hat{A}_t \mid u^{(m_t)}, \mathcal{H}_t\big)\, \mathcal{K}\big(s_{t+1}, o_{t+1} \mid s_t, \hat{A}_t\big), \tag{24}$$

and taking logarithms leaves an environment term $\log \mu_{\mathrm{env}}(\tau)$, the sum of $\log \mu_0$ and the kernel terms $\log \mathcal{K}$, that depends on neither ϕ nor θ. The gradient in ϕ thus involves only the decision terms $\log p_{\phi}(y_t \mid z_t)$ and the gradient in θ only the chunk terms $\log \pi^{\mathrm{l}}_{\theta}(\hat{A}_t \mid u^{(m_t)}, \mathcal{H}_t)$, so the factors get disjoint gradients. HJPO holds ϕ fixed online and optimizes the second sum only.

Setup. Fix an iteration and let $\mathcal{C} = \mathcal{D}^+ \cup \mathcal{D}^-$ be the collected chunks, where every chunk c inherits the binary reward $R(c) = R(\tau(c))$ of its rollout, $\mathcal{D}^+$ holds the $N^+ \ge 1$ chunks with $R = 1$ and $\mathcal{D}^-$ the N^- chunks with $R = 0$. Write $\ell_{\theta}(c) = \|f_{\theta}(I_c, S_c, \tilde{x}_c) - \hat{A}_c\|_1$ and $\varepsilon = e^{-1/\beta}$.

AWR reduces to weighted L1 regression. AWR [59] fits the policy by weighted maximum likelihood with the AWR weights w_β of each chunk, self-normalized over the collected data, $\bar{w}_\beta(c) = w_\beta(c)/\sum_{c' \in \mathcal{C}} w_\beta(c')$. Self-normalization removes any constant baseline, since for every $b_0 \in \mathbb{R}$

$$\bar{w}_\beta(c) \;=\; \frac{\exp\big((R(c) - b_0)/\beta\big)}{\sum_{c' \in \mathcal{C}} \exp\big((R(c') - b_0)/\beta\big)} \;=\; \frac{\exp\big(R(c)/\beta\big)}{\sum_{c' \in \mathcal{C}} \exp\big(R(c')/\beta\big)}, \tag{25}$$

so the choice $R_{\max}$ only fixes the scale of the unnormalized weights. Under the Laplace chunk policy, $-\log \pi^{\mathrm{l}}_\theta(\hat{A}_c \mid \cdot) = \ell_\theta(c)/b + 7H \log(2b)$, whose constant does not depend on θ; therefore

$$b\,\mathcal{L}^{\mathrm{AWR}}_\beta(\theta) \;=\; -\,b \sum_{c \in \mathcal{C}} \bar{w}_\beta(c) \log \pi^{\mathrm{l}}_\theta\big(\hat{A}_c \mid \cdot\big) \;=\; \sum_{c \in \mathcal{C}} \bar{w}_\beta(c)\, \ell_\theta(c) \;+\; 7Hb \log(2b). \tag{26}$$

Zero-temperature limit with explicit rate. With $R_{\max} = 1$, $w_\beta = 1$ on $\mathcal{D}^+$ and $w_\beta = \varepsilon$ on $\mathcal{D}^-$, so $\bar{w}_\beta(c) = 1/(N^+ + N^-\varepsilon)$ on $\mathcal{D}^+$ and $\varepsilon/(N^+ + N^-\varepsilon)$ on $\mathcal{D}^-$. Let $L^+(\theta) = \frac{1}{N^+}\sum_{c \in \mathcal{D}^+} \ell_\theta(c)$ be the replay term of $\mathcal{L}_{\mathrm{HJPO}}$ under uniform sampling from $\mathcal{D}^+$ and $L^-(\theta)$ the mean over $\mathcal{D}^-$. Then

$$\begin{aligned} \sum_{c \in \mathcal{C}} \bar{w}_\beta(c)\, \ell_\theta(c) - L^+(\theta) \;&=\; \frac{N^-\varepsilon}{N^+ + N^-\varepsilon}\,\big(L^-(\theta) - L^+(\theta)\big), \\ \Big|\sum_{c \in \mathcal{C}} \bar{w}_\beta(c)\, \ell_\theta(c) - L^+(\theta)\Big| \;&\le\; \frac{N^-}{N^+}\, e^{-1/\beta}\, \big|L^-(\theta) - L^+(\theta)\big|. \end{aligned} \tag{27}$$

The right-hand side vanishes as $\beta \to 0$ at rate $e^{-1/\beta}$, pointwise in θ and uniformly on any set where the losses are bounded. The identity is linear in ℓ_θ, so the same bound holds for the (sub)gradients,

$$\Big\|\nabla_\theta \sum_{c \in \mathcal{C}} \bar{w}_\beta(c)\, \ell_\theta(c) - \nabla_\theta L^+(\theta)\Big\| \;\le\; \frac{N^-}{N^+}\, e^{-1/\beta}\, \big\|\nabla_\theta L^-(\theta) - \nabla_\theta L^+(\theta)\big\|, \tag{28}$$

so the AWR update direction converges to the gradient of the replay term. As the demonstration term is free of β, the anchored objective converges to $\mathcal{L}_{\mathrm{HJPO}}$ up to the constant $(1-\eta)\,7Hb\log(2b)$:

$$\begin{aligned} &\Big|\eta\,\mathcal{L}_{\mathrm{SFT}}(\theta) + (1-\eta)\big(b\,\mathcal{L}^{\mathrm{AWR}}_\beta(\theta) - 7Hb\log(2b)\big) - \mathcal{L}_{\mathrm{HJPO}}(\theta)\Big| \\ &\qquad\le\; (1-\eta)\,\frac{N^-}{N^+}\, e^{-1/\beta}\, \big|L^-(\theta) - L^+(\theta)\big|. \end{aligned} \tag{29}$$

An iteration with an empty $\mathcal{D}^+$ performs no replay update and keeps the current θ unchanged.

Target law. Let μ be the trajectory law of the exploring hierarchical policy and $q = \mu(R = 1) > 0$. The AWR target is the reweighted law $\mu_\beta(\tau) = \mu(\tau)\, w_\beta(\tau)/Z_\beta$ with $Z_\beta = q + (1-q)\varepsilon$, and the success-conditioned law is $\mu^+(\tau) = \mu(\tau)\,\mathbf{1}[R(\tau) = 1]/q$. On successes $\mu_\beta - \mu^+ = \mu\,(1/Z_\beta - 1/q) \le 0$ and on failures $\mu_\beta - \mu^+ = \mu\,\varepsilon/Z_\beta \ge 0$, so the total variation distance is

$$\mathrm{TV}\big(\mu_\beta, \mu^+\big) \;=\; \tfrac{1}{2}\Big[\, q\Big(\tfrac{1}{q} - \tfrac{1}{Z_\beta}\Big) + \tfrac{(1-q)\varepsilon}{Z_\beta}\Big] \;=\; \frac{(1-q)\,e^{-1/\beta}}{q + (1-q)\,e^{-1/\beta}} \;\le\; \frac{1-q}{q}\, e^{-1/\beta}, \tag{30}$$

and the Kullback–Leibler divergence $\mathrm{KL}(\mu^+ \,\|\, \mu_\beta) = \log(Z_\beta/q) \le \frac{1-q}{q} e^{-1/\beta}$ obeys the same rate. As ϕ and z_t are fixed and the environment terms are free of θ, maximum likelihood on μ^+ satisfies

$$\operatorname*{argmax}_\theta\; \mathbb{E}_{\tau \sim \mu^+}\big[\log \pi_\Theta(\tau)\big] \;=\; \operatorname*{argmin}_\theta\; \mathbb{E}_{\tau \sim \mu^+}\Big[\sum_{t=1}^{T} \ell_\theta(c_t)\Big], \qquad c_t = \big(o_t, \tilde{x}_t, \hat{A}_t\big), \tag{31}$$

whose empirical counterpart is the replay term L^+: filtered AWR fits the policy's own successes.

Validation selection is non-degrading. The reported model is $\hat{\theta} = \theta_{\hat{k}}$ with $\hat{k} = \operatorname{argmax}_{k \in \mathcal{Q}} \mathrm{SR}_{\mathrm{val}}(\theta_k)$ over the candidate iterates $\mathcal{Q}$, and $0 \in \mathcal{Q}$ indexes the supervised start. Hence

$$\mathrm{SR}_{\mathrm{val}}(\hat{\theta}) \;=\; \max_{k \in \mathcal{Q}} \mathrm{SR}_{\mathrm{val}}(\theta_k) \;\ge\; \mathrm{SR}_{\mathrm{val}}(\theta_0), \tag{32}$$

and tie-breaking chooses only among maximizers, leaving the value unchanged. The inequality holds deterministically for every online run, as all candidates are scored on the same validation states. □

C Training algorithm

Algorithm 1 gives the full training procedure and Algorithm 2 a rollout of the hierarchical policy. Following the options framework [73], a *decision* t updates the pointer and emits a chunk, which then runs open-loop for $H = 8$ environment steps before the next decision reads a new observation.

Algorithm 1 NS-VLA training: supervised stage, online HJPO stage and validation selection

Require: demonstrations $\mathcal{D}$ (5 per task); parser $\mathcal{P}$; backbone f_θ initialized from the generic vision-language model; validation initial states $\mathcal{V}$ (5 per task); $K = 8$, $N_{\text{roll}} = 8$, $U = 400$, $\eta = 0.5$, $\sigma_{\text{exp}} = 0.03$

Stage I: supervised fitting of both factors

for each demonstration $(x, o_{1:n}, a_{1:n}) \in \mathcal{D}$ **do**
  $p \leftarrow \mathcal{P}(x)$ ▷ plan grammar
  $(s^{(1)}, \ldots, s^{(M)}) \leftarrow$ gripper-event segmentation of $a_{1:n}$ ▷ grasps and stage starts
end for
train θ on $\mathcal{L}_{\text{SFT}}$ ▷ $\xi \sim \text{Bern}(1/2)$
$\theta_0 \leftarrow$ snapshot with the highest SR_{val} ▷ snapshots every 2.5k steps
build stay/advance examples with f_{θ_0} under the queried pointer; fit ϕ on $\mathcal{L}_{\text{ptr}}$ ▷ class-balanced

Stage II: online stage of the low-level factor (filtered AWR)

$\theta \leftarrow \theta_0$; $\mathcal{D}^+ \leftarrow \emptyset$; $\mathcal{Q} \leftarrow \{0\}$
for $k = 1, \ldots, K$ **do**
  for each task, N_{roll} times **do**
    $(\tau, R) \leftarrow \textsc{Rollout}(\theta, \phi, \sigma_{\text{exp}})$ ▷ Algorithm 2
    if $R = 1$ **then** $\mathcal{D}^+ \leftarrow \mathcal{D}^+ \cup \{(o_t, \tilde{x}_t, \hat{A}_t)\}_{t \in \tau}$ ▷ task-level credit
    end if
  end for
  if $\mathcal{D}^+ \neq \emptyset$ **then**
    take U AdamW steps on $\mathcal{L}_{\text{HJPO}}(\theta)$ from the current θ ▷ replay mixed with demos
  end if
  $\theta_k \leftarrow \theta$
  if $k \bmod 2 = 0$ **then**
    evaluate the policy with θ_k on $\mathcal{V}$; $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{k\}$
  end if
end for
return $\hat{\theta} \leftarrow \theta_{\hat{k}}$, $\hat{k} = \text{argmax}_{k \in \mathcal{Q}} \text{SR}_{\text{val}}(\theta_k)$ ▷ ties: more milestones, then larger k

C.1 Implementation of the two stages

Supervised stage. The Solver is trained on the sub-instruction-augmented loss with snapshots every 2.5k steps, and the snapshot with the highest validation SR becomes θ_0. The stage features z_t are then computed by θ_0 under the bridged input of the queried pointer, exactly as at execution time. Each frame i of segment j yields a stay example $(z_i^{(j)}, 0)$, with $z_i^{(j)}$ its stage features under pointer j, and, for $j > 1$, an advance example $(z_i^{(j-1)}, 1)$, and with class weights $\omega_y = N/(2N_y)$ over N_y examples of class y out of N, the classifier g_ϕ minimizes the class-balanced cross-entropy

$$\mathcal{L}_{\text{ptr}}(\phi) \;=\; -\,\mathbb{E}_{(z,y)}\Big[\omega_1\, y \log \sigma\big(g_\phi(z)\big) \;+\; \omega_0\,(1-y)\log\big(1-\sigma(g_\phi(z))\big)\Big], \tag{33}$$

which balances the rare advance decisions against the far more frequent stay decisions.

Online stage. A replayed tuple stores both images, the proprioceptive state, the bridged input under which the chunk ran and the executed chunk with a binarized gripper dimension, so the target is the action that produced the success and the condition is the sub-instruction that the pointer selected. $\mathcal{D}^+$ accumulates over iterations; each sample comes from the demonstrations with probability $\eta = 0.5$ and otherwise uniformly from $\mathcal{D}^+$. Every update starts a fresh AdamW optimizer on the LoRA adapters, the action head and the proprioception projection, while parser, gate and ϕ stay fixed.

Algorithm 2 ROLLOUT of one episode with the hierarchical policy (inference uses $\sigma_{\text{exp}} = 0$)

Require: parameters (θ, ϕ); exploration scale σ_{exp}; $d_{\min} = 2$; $w_{\min} = 0.0025$
reset the environment; read x; $p \leftarrow \mathcal{P}(x)$ ▷ plan, once per episode
$m \leftarrow 1$; $d \leftarrow 0$; *held* $\leftarrow 0$; $\kappa \leftarrow 0$; $h \leftarrow 0$ ▷ pointer, dwell and gripper state
while the episode is not over **do**
  observe $o_t = (I_t, S_t)$; $\tilde{x} \leftarrow$ bridged input of slot m (x itself if $M = 1$)
  $(\bar{A}_t, \bar{h}_t) \leftarrow f_\theta(I_t, S_t, \tilde{x})$ ▷ mode chunk and pooled context, one pass
  if $M > 1$ **then**
    $z_t \leftarrow [\bar{h}_t;\ e(u^{(m)});\ e(u^{(\min(m+1,M))});\ S_t;\ d;\ m/M]$ ▷ stage features
    $\Gamma \leftarrow$ gate of $u^{(m)}$ from $(\kappa, h, \textit{held})$ ▷ precondition gate
    if $m < M$ **and** $d \geq d_{\min}$ **and** $\Gamma = 1$ **and** $\sigma(g_\phi(z_t)) > 1/2$ **then**
      $m \leftarrow m + 1$; $d \leftarrow 0$; *held* $\leftarrow h$ ▷ a grasp held across the boundary counts
      $\tilde{x} \leftarrow$ bridged input of slot m; $(\bar{A}_t, \cdot) \leftarrow f_\theta(I_t, S_t, \tilde{x})$ ▷ re-predict
    else
      $d \leftarrow d + 1$
    end if
  end if
  $\hat{A}_t \leftarrow \bar{A}_t$; $\hat{A}_t[:, 1{:}6] \leftarrow \text{clip}\big(\bar{A}_t[:, 1{:}6] + \zeta, -1, 1\big)$, $\zeta \sim \mathcal{N}(0, \sigma_{\text{exp}}^2 I)$ ▷ arm only
  execute $\hat{A}_t$ open-loop for up to H steps (stop at success)
  $\kappa \leftarrow$ last executed gripper command; $w \leftarrow$ finger separation from S
  $h \leftarrow \kappa\, \mathbf{1}[w > w_{\min}]$; *held* $\leftarrow \max(\textit{held}, h)$
  record $(o_t, \tilde{x}, \hat{A}_t)$ with the gripper dimension binarized
end while
return trajectory τ and task reward $R(\tau) \in \{0, 1\}$

Rollouts and selection. Rollouts start from training initial states, validation uses five held-out initial states per task, and the official test states are reserved for evaluation, so selection never touches test episodes. Simulator milestones serve analysis and the tie-break of the selection only.

D Baselines and ablation variants

All ten baselines and VLA-Adapter train one multi-task model per suite on the same 10% split with seed 7 (VLA-Adapter also 17 and 27 on Long), follow their official recipes from scratch or general-purpose weights, and are evaluated with their final checkpoint (VLA-Adapter, like NS-VLA, with its best-validation snapshot) under the program, initial states, checks and limits of NS-VLA.

- **BC-RNN** [46]: a recurrent behavior-cloning policy over a ten-step observation history with a five-mode Gaussian-mixture action head, ResNet image encoders and FiLM language fusion of a BERT instruction embedding.
- **BC-Transformer (BC-T)** [43]: the LIBERO multi-task policy that replaces the recurrent core of BC-RNN by a temporal Transformer over the same observation history and Gaussian-mixture head.
- **BC-ViLT** [43]: the LIBERO vision-language Transformer that tokenizes image patches and the instruction embedding jointly in a single Transformer.
- **ACT** [89]: a conditional-VAE Transformer that predicts 16-step action chunks.
- **Diffusion Policy (DP)** [12]: a denoising-diffusion policy over 32-step action chunks, with 100 training and 10 DDIM inference steps.
- **VQ-BeT** [37]: a behavior Transformer over residual vector-quantized action codes of five-step blocks with a ten-frame observation stack.
- **QueST** [51]: a two-stage policy that learns a finite-scalar-quantized skill autoencoder over 32-step action blocks and an autoregressive GPT prior over the skill tokens.
- **BAKU** [26]: a compact multi-task Transformer with ResNet encoders, FiLM conditioning on a MiniLM sentence embedding of the instruction, proprioceptive input and ten-step action chunks with temporal aggregation.

- **Octo** [50]: the 93M-parameter Octo-Base generalist Transformer policy with a T5-base instruction encoder and a diffusion action head, fine-tuned from octo-base-1.5 with its official recipe.
- **SmolVLA** [71]: a 0.45B vision-language-action model that pairs the SmolVLM2-500M vision-language model with a flow-matching action expert over 50-step chunks, fine-tuned from smolvla_base.
- **VLA-Adapter** [79]: the backbone of NS-VLA, initialized from the same generic vision-language model and trained with the same supervised recipe on the original instruction.

Codebases. BC-RNN, BC-T and BC-ViLT use the official LIBERO codebase (commit `8f1084e`, with the end-effector input set to the 6-D pose of our state). ACT, DP, VQ-BeT and QueST use the QueST codebase (commit `cc25a83`) with CLIP instruction embeddings. BAKU uses its official repository unmodified (commit `47e3670`), Octo the official repository unmodified (commit `241fb35`) with its fine-tuning script, and SmolVLA the LeRobot package (version 0.4.4). Observations are the third-person and wrist images (128×128 for the LIBERO, QueST and BAKU codebases; 256×256 and 128×128 for Octo) together with the end-effector and gripper state, and actions are the same 7-D end-effector deltas with gripper command used by NS-VLA. Table 5 lists every training configuration.

Table 5: Training configuration; lengths per suite (S/O/G/L), ep.: epochs, AE: autoencoder.

Method	Initialization	Params	Batch	Learning rate	Length (S / O / G / L)
BC-RNN	from scratch	19.10M	32	1e-4, cosine	50 / 50 / 50 / 28 ep.
BC-T	from scratch	5.38M	32	1e-4, cosine	50 / 50 / 50 / 21 ep.
BC-ViLT	from scratch	7.90M	32	1e-4, cosine	30 / 30 / 30 / 15 ep.
ACT	from scratch	–	128	1e-4	100 / 100 / 100 / 100 ep.
DP	from scratch	49.88M	128	1e-4	200 / 197 / 200 / 173 ep.
VQ-BeT	from scratch	1.10M	128	5.5e-5	100 / 99 / 61 / 44 ep. (+100 VQ-VAE)
QueST	from scratch	–	128	1e-4	100 / 100 / 100 / 100 ep. (+100 AE)
BAKU	from scratch	6.53M	64	1e-4	28.3k / 32.8k / 36.6k / 35.4k steps
Octo	octo-base-1.5	93M	48	3e-4, cosine	2.6k / 2.6k / 2.6k / 2.6k steps
SmolVLA	smolvla_base	0.45B	16	1e-4, cosine	4.5k / 4.5k / 4.0k / 4.0k steps
VLA-Adapter	generic VLM	$\sim$1.3B	16	2e-4, cosine	10k / 10k / 10k / 20k steps

The ten baselines train on NVIDIA V100 GPUs, and VLA-Adapter and NS-VLA on NVIDIA H100 GPUs in BF16; every method is evaluated on the same machines with EGL rendering.

Ablation variants. All variants of Table 3 and Figure 4 run on the 500 LIBERO-Long test episodes with seed 7. *w/o Encoder* runs the final checkpoint without plan and pointer, so the backbone reads the original instruction. *w/o gate* sets $\Gamma_t \equiv 1$, so the pointer advances on the classifier and the dwell alone. *w/o Solver* is the seed-7 VLA-Adapter run, trained on the original instructions and given the same online stage without plan or bridged input; its supervised start collapses on LIBERO-Long (0.0) and collects no success, so it is kept. *w/o HJPO* evaluates the supervised start θ_0 with the Encoder.

E Datasets and evaluation protocol

LIBERO. LIBERO [43] has four suites of ten tasks with 50 demonstrations and 50 official test initial states per task. Spatial varies the layout around identical objects, Object the target object, Goal the goal in a fixed scene, and Long chains two to four primitives in a single instruction.

Training split. Every method trains on the first five demonstrations of every task in the fixed record order of the no-op-filtered LIBERO RLDS release, 200 demonstrations in total (33,604 frames). The list is shared by all methods and seeds, and normalization statistics use these demonstrations only.

LIBERO-Plus. LIBERO-Plus [19] is held out from training and selection. Its 10,030 variants (2,402 Spatial, 2,518 Object, 2,591 Goal, 2,519 Long) each perturb one of seven dimensions at a difficulty level from 1 to 5; the language dimension rewrites the instruction, as shown in Figure 8.

Evaluation protocol. LIBERO uses the first 20 official initial states of every task, 200 episodes per suite. LIBERO-Plus draws 300 variants per suite, stratified by dimension: 100 with sampling seed 2026 and a pre-registered extension of 200 with seed 2027, 1,200 variants (129 background,

191 camera, 185 language, 136 lighting, 182 layout, 186 robot-state, 191 sensor-noise). Lists are fixed in advance and shared, so every comparison is paired; the extension runs the same checkpoints on a second machine whose renderings differ by at most two gray levels. VLA-Adapter collapses on LIBERO-Long with seed 7 (0.0), so Table 2 reports its mean over seeds 7, 17 and 27 (0.0, 42.5 and 45.5) with the standard error across seeds. LIBERO-Plus uses the seed-7 checkpoints, so VLA-Adapter's Long variants come from this collapsed model (0 of 300); on the 900 Spatial, Object and Goal variants it solves 545 against 542 for NS-VLA, and with seed 17 it solves 31.2% of all 2,519 Long variants against 47.1%. A cell with no success in n episodes shows the one-sided 95% Clopper–Pearson interval $[0, u]$, $u = 1 - 0.05^{1/n}$, as $u/2 \pm u/2$: 0.8 ± 0.8 for $n = 186$ or 191 and 1.1 ± 1.1 for $n = 129$. RQ2–RQ5 mainly analyze the 500 LIBERO-Long test episodes.

F Implementation details

Table 6 lists the full configuration. The backbone is VLA-Adapter initialized from a generic vision-language model without robot pretraining, with randomly initialized action head and proprioception projection. LIBERO-Long instructions are parsed into the plans of Table 4; on Spatial, Object and Goal the parser is not applied, by design, and every episode there runs the one-slot plan, on which NS-VLA reduces exactly to its backbone trained with the same procedure (Proposition 4.2).

G Experiment details

Three episode sets are used: the 200 episodes per LIBERO suite of Table 2, the 500 LIBERO-Long test episodes of RQ2–RQ5, and the 1,200 LIBERO-Plus variants, with the paraphrase analysis using all 1,537 language variants. On LIBERO-Long, NS-VLA thus reaches 42.0 on the 200 episodes, 40.6 on the 500 test episodes and 45.3 on average over the three training seeds (Figure 6).

G.1 Main results (RQ1)

Setting. A cell of Table 2 is an SR with its binomial standard error, the LIBERO average is the mean of the four suites, the LIBERO-Plus average pools the 1,200 variants, and Figure 12 charts them.

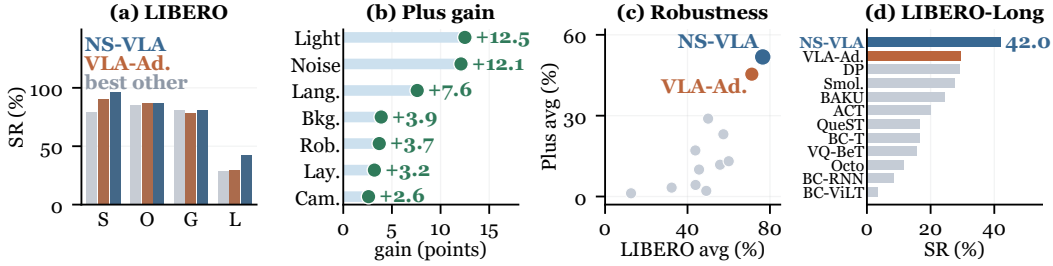


Figure 12: Main results: (a) LIBERO suites (S/O/G/L); (b) gain over the best baseline per LIBERO-Plus dimension; (c) LIBERO against LIBERO-Plus average of every method; (d) LIBERO-Long.

Results. NS-VLA ranks first or ties for first in all thirteen columns, leads LIBERO by 5.3 and LIBERO-Plus by 6.4 points on average, and is the only method in the upper right of panel (c). It wins 77 more of the 1,200 variants than VLA-Adapter (exact McNemar test, $p = 1.5 \times 10^{-5}$).

Data and interaction. All methods share the 10% split, and NS-VLA additionally interacts online during HJPO. Without its online stage, NS-VLA uses exactly the data of the baselines and no interaction, yet averages 41.2 on the 500 LIBERO-Long test episodes over seeds 7, 17 and 27 against 29.5 for VLA-Adapter. Flat RL in the main text uses seed 7, whose start scores 0.0 and collects no success; with seeds 17 and 27 it collects 239 and 227 of 640 (HJPO: 307 and 192).

G.2 Ablation study (RQ2)

Setting. Table 7 adds milestones per episode, the share of episodes that reach four stages, and the SR change with a 95% interval from a paired bootstrap over the ten tasks (10,000 resamples, seed 0). Figure 4(c) shows the share of episodes reaching stage $k + 1$ among those reaching stage k.

Table 6: Hyperparameters of NS-VLA, of its two training stages and of the evaluation protocol.

Group	Hyperparameter	Value
Backbone	Language model / vision encoders	Qwen2.5-0.5B / DINOv2 + SigLIP
	Adaptation	LoRA rank 64
	Action head	L1 regression, 7-D actions
	Action chunk H	8 steps, executed open-loop
	Initialization	generic vision-language model
Plan and segments	Primitive vocabulary	pick, place, open, close, turn on, turn off, push
	Pick-and-place clause	split into pick + place
	Pick segment end	gripper close persisting ≥ 30 frames
	Place segment end	next gripper open
	Bridged input	"`<instruction>. now: <sub-instruction>`"
Supervised stage	Optimizer	AdamW, peak lr 2×10^{-4}
	Schedule	100-step warmup, cosine decay to 2×10^{-5}
	Batch size / precision	16 / BF16
	Steps	20k (Long), 10k (Spatial, Object, Goal)
	Sub-instruction probability	0.5
	Snapshot selection	every 2.5k steps, validation SR
Stage pointer	Classifier g_ϕ	stay/advance, class-balanced cross-entropy
	Training	AdamW, lr 3×10^{-4}, 30 epochs
	Advance threshold	$\sigma(g_\phi(z_t)) > 1/2$
	Minimum dwell $d_{\min}$	2 chunks
	Pick gate	closed, finger gap $> w_{\min} = 0.0025$
	Place gate	object held, then gripper opened within the stage
Online stage (HJPO)	Iterations K	8
	Rollouts	8 per task per iteration (80 per suite)
	Exploration noise	Gaussian, $\sigma_{\text{exp}} = 0.03$ on the 6 arm dimensions
	Updates per iteration	400, lr 5×10^{-5}
	Batch size	16 (8×2 accumulation)
	Demonstration ratio η	0.5
	Updated parameters	LoRA, action head, proprioception projection
Selection	Validation states	5 held-out initial states per task
	Candidates	iterations $\{0, 2, 4, 6, 8\}$, best validation SR
Evaluation	Initial states	official LIBERO test states
	Max episode steps	220 / 280 / 300 / 520 (S / O / G / L)
	Seed	7 (also 17 and 27 for Figures 6 and 7(d) and VLA-Adapter on Long)
	Rendering	EGL, the same machines for all methods

Table 7: Extended ablations of every component on the 500 LIBERO-Long test episodes (seed 7).

Variant	SR	PSCR	Milestones	4 stages (%)	ΔSR [95% CI]
NS-VLA	**40.6**	**54.4**	**1.79**	**14.2**	–
w/o gate	39.0	53.9	1.77	13.0	-1.6 $[-3.6, 0.0]$
w/o Encoder	38.0	53.1	1.75	12.8	-2.6 $[-8.0, +1.2]$
w/o HJPO	28.2	48.9	1.62	8.4	-12.4 $[-19.8, -5.0]$
w/o Solver	0.0	7.3	0.25	0.0	-40.6 $[-57.0, -24.4]$

Results. The full model leads every variant on all four measures. Its continuation rates are 77.0, 63.0 and 40.1% up to the fourth stage, against 67.0, 51.2 and 32.1% without HJPO refinement.

G.3 Neuro-Symbolic Encoder (RQ3)

Setting. For every pointer advance, the lag is the number of chunks from the advance to the milestone of the stage it leaves, negative when the milestone came first. An advance is grounded when this

milestone arrives at most three chunks after it, and an episode is aligned when its final pointer lies within one stage of its completed stages. Panels (a, b) of Figure 5 use the 500 test episodes, (c) the 200 episodes of Table 2 (VLA-Adapter: 3 seeds), and (d) all 1,537 language variants by level.

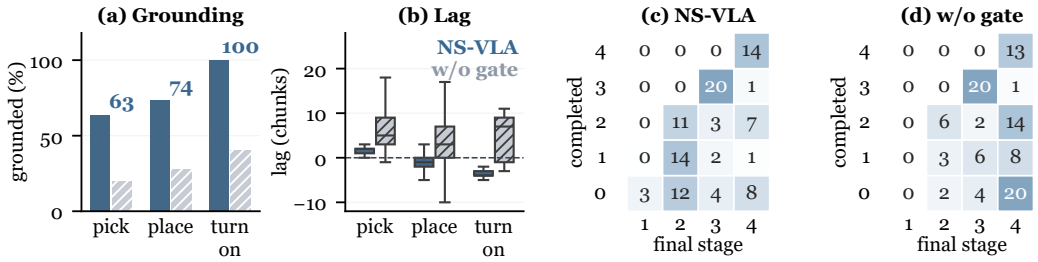


Figure 13: Pointer on the 500 LIBERO-Long test episodes: (a) grounded advances and (b) lag per primitive left, with and without the gate; (c, d) final pointer stage against completed stages.

Results. NS-VLA grounds 68.2% of its 940 advances against 23.2% of 1,220 without the gate, aligns 87.4% of its episodes against 68.0%, and lowers the median lag from five chunks to one. The gain holds for every primitive the pointer leaves, 63 against 20% for `pick`, 74 against 28% for `place` and 100 against 40% for `turn on` (Figure 13), and the gate removes most runaway episodes that reach the last stage without completing any. NS-VLA completes 55.1% of plan stages on the 200 episodes against 51.6% for BAKU, and on paraphrases reaches 68.3, 69.3, 68.3, 66.2 and 32.2% from difficulty level 1 to 5, against 64.3, 61.8, 48.6, 46.0 and 30.1% for its VLA-Adapter backbone.

G.4 Neuro-Symbolic Solver (RQ4)

Setting. Figure 6 averages over training seeds 7, 17 and 27: (a) validation SR of every snapshot and (b) the L1 loss smoothed over 25 logged points. Panel (c) takes the 200 LIBERO-Long episodes of Table 2 for all methods (VLA-Adapter: 3 seeds) and (d) the 500 test episodes of each seed.

Results. At 20k steps NS-VLA reaches 50.7% mean validation SR against 34.0%, at loss 0.058 against 0.078. NS-VLA completes the task in 57.9% of its 145 episodes that finish the first primitive, against 47.0% of 117 for SmolVLA, and averages 45.3 ± 9.6 over seeds against 29.5 ± 25.6.

G.5 Hierarchical Joint Policy Optimization (RQ5)

Setting. Each iteration collects 80 rollouts (8 per task) from training states; the flat baseline runs the same stage on VLA-Adapter without plan or bridged input. Panel (c) of Figure 7 pairs the 500 test episodes of θ_0 and the selected model, and (d) covers all 19 online runs: 3 flat on Long and 16 of NS-VLA, with the one-slot runs of Table 2 and exploratory multi-slot and repeat runs only here.

Table 8: Online stage on LIBERO-Long (seed 7): rollout SR (%) and milestones per rollout.

		1	2	3	4	5	6	7	8
Rollout SR	flat RL	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	NS-VLA	**32.5**	**22.5**	**38.8**	**28.8**	**40.0**	**36.3**	**18.8**	**26.3**
Milestones	flat RL	0.25	0.19	0.16	0.23	0.18	0.24	0.31	0.25
	NS-VLA	**1.76**	**1.53**	**1.88**	**1.64**	**2.01**	**1.68**	**1.39**	**1.49**

Results. NS-VLA collects successes in every iteration (Table 8) and reaches 1.67 milestones per rollout, 7.4× the 0.225 of the flat baseline, whose replay set stays empty. On the test episodes the online stage turns 105 failures into successes, a net gain of 62. Selection improves on the supervised start in 9 of the 19 runs, by 4 to 8 validation points, and matches it in the other 10.

G.6 Case study (RQ6)

Setting. Rollouts use the reported checkpoints from official initial state 0: LIBERO-Long task 3 and its paraphrase (language variant 1225, level 4) in Figure 8, and task 2 in Figure 14.

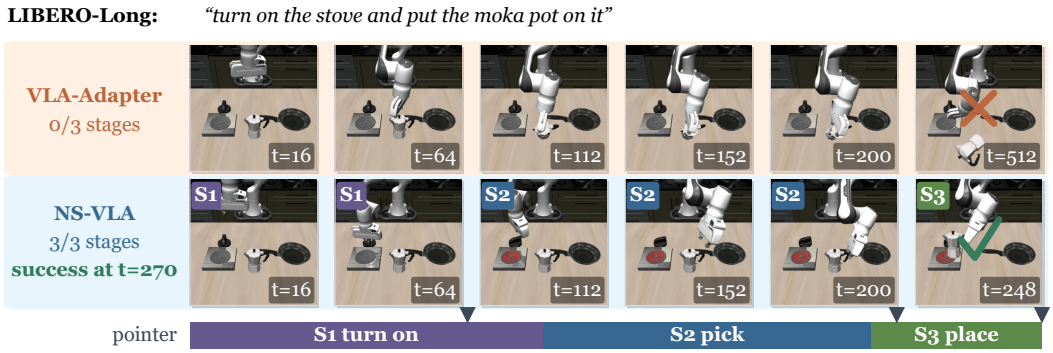


Figure 14: A second case, whose plan begins with the grasp–release primitive `turn on`.

Results. NS-VLA turns on the stove, picks up the moka pot and places it on the stove, with milestones at steps 88, 224 and 270, and succeeds at step 270, while VLA-Adapter reaches no stage. The pointer leaves every stage, `turn on` included, only once the gate sees the required gripper evidence.

G.7 Real-world tasks (RQ7)

Setting. The myCobot 320pi arm observes the scene with a wrist-mounted and a third-person RealSense D435i camera (Figure 9). Three Pick, three Move and four Place tasks are each repeated ten times under randomized object placements for both policies (30, 30 and 40 trials).

Results. NS-VLA succeeds in 22 of 30 Pick, 18 of 30 Move and 20 of 40 Place, against 16, 12 and 12 for VLA-Adapter, leading each category by 20 points and reaching 60.0 against 40.0% overall.

H Extended related work

Reinforcement learning for robot policies. Trust-region and clipped policy gradients with advantage estimation [66–68] are the standard objectives of online RL, while advantage-weighted and offline methods keep the policy close to the data [22, 23, 53, 59, 82]. Preference and group-relative objectives from language modeling [56, 61, 69] were carried to VLAs and embodied reasoners [3, 18, 39, 40, 45, 72, 85, 86], and reward shaping densifies sparse returns [54, 80]. HJPO keeps the binary reward unshaped and makes it reachable through structure, turning every success into a parameter-free regression target for the action policy, with no critic and no learned reward model.

Structure in language-conditioned manipulation. Language models decompose instructions into skills, code or value maps [28, 29, 31, 42], and VLAs reason through language hierarchies, chains of thought, visual traces or predicted futures [4, 76, 87, 88, 91]. Options, feudal managers and goal-conditioned hierarchies organize control in time [2, 15, 36, 38, 52, 73, 77], and neuro-symbolic planners execute parsed programs [1, 13]. NS-VLA joins these lines within one chunked VLA policy.

Hierarchical VLAs and their interfaces. Two-level control is the setting of NS-VLA, not its claim; hierarchical VLAs differ in what crosses the interface (Table 9). LCB [70] lets a language model emit a latent code that steers a pretrained policy, VAMOS [10] separates a general planner from an embodiment-specific capability model, and VINE [58] derives its hierarchy from success and failure demonstrations. NS-VLA passes a typed primitive, its free-text arguments and an observable pointer state, so the active primitive and every transition can be read off, certified and measured. A capability model decides which plans are admissible, this interface how a plan is executed and verified.

I Scaling the symbolic layer

Division of labor. The symbolic layer carries only the plan structure and the execution gate; perception, grounding and control stay neural. Operations form a small vocabulary, while objects, supports and relations stay free-text arguments, so the unbounded side of the world needs no symbol. The pointer reads its features from the fine-tuned backbone, so no frozen encoder bounds the visual domain. Unlike classical expert systems, whose rules carried perception and control and failed hard at

Table 9: What crosses the interface between the two levels of recent hierarchical VLAs.

Method	Interface between the levels	Source of the hierarchy
LCB [70]	continuous latent code	language model fine-tuned to emit the code
VAMOS [10]	navigation plan admitted by a capability model	general planner and embodiment-specific model
VINE [58]	plan scored by a success estimator	success and failure demonstrations
NS-VLA	typed primitive, free-text arguments and pointer state	compositional structure of the instruction

every gap, a gap in the grammar costs only the decomposition: an instruction that is not decomposed runs with the one-slot plan, on which NS-VLA is exactly its backbone (Proposition 4.2).

What a new task costs. Only a new operation requires retraining (Table 10), and it costs a grammar rule and one pass of the automatic pipeline rather than a manual annotation campaign.

Table 10: What NS-VLA retrains, and what it keeps, when a new task brings something new.

What is new	What is retrained
an object, support or relation	nothing: it enters as a free-text argument
a phrasing that the grammar parses	nothing: the same plan is produced
a phrasing that the grammar does not parse	nothing: the one-slot plan runs the backbone exactly
an operation	a grammar rule with its gripper class, automatic segmentation of its demonstrations, and the supervised stage on them

Adaptation within an episode. The plan depends on the instruction alone, so recompiling it mid-episode returns the same plan; the pointer decides from observations when a primitive is done, and the Solver recovers inside a primitive, e.g. by regrasping while the gate holds the pointer on `pick`.

J Limitations and future work

Limitations. The operations come from a grammar: a new operation needs a rule and its gripper class, and an instruction the grammar does not decompose runs as the plain backbone, without plan or pointer. The plan is fixed per episode and the pointer only stays or advances, so a stage left too early is not revisited and the plan content is not revised when the scene contradicts it. The gate certifies grasp and release primitives from proprioception, whereas gripper-free primitives rely on the classifier and the dwell. The evaluation covers the four LIBERO suites with five demonstrations per task, LIBERO-Plus and one real arm with 100 trials per policy, with VLA-Adapter as the only backbone and seed 7 for the main table; online training needs simulated rollouts with binary success.

Future work. Four directions build on the interface: a pointer that may also retreat to an earlier stage; observation-triggered revision of the plan, such as inserting "open the drawer" when the drawer is found shut; label-free discovery of operations by clustering contact and gripper events across demonstrations; and object-centric arguments, where single-image 3D reconstruction such as SAM 3D [64] supplies per-object shape and pose, so that a novel object needs no new symbol and grasping and placing become directly observable. Since the interface needs only an instruction-conditioned chunked policy, transferring it to further VLA backbones and robot platforms is a direct next step.

K Broader impact

Learning multi-step manipulation from few demonstrations lowers the cost of teaching robots household and assistive tasks, and the explicit plan makes every decision readable to an operator. As NS-VLA reduces exactly to its backbone on single-primitive instructions and selection never falls below the supervised start on validation, adopting it preserves the behavior of an existing policy wherever structure is absent. A misparsed instruction or a failed grasp can still move objects unexpectedly, so deployment should follow standard robot-safety practice, with the observable stage pointer as a natural point for monitoring and human intervention at every plan stage.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and §1 state the three components of NS-VLA (the Neuro-Symbolic Encoder, the Neuro-Symbolic Solver and Hierarchical Joint Policy Optimization) and the setting: few demonstrations (five per task, 10% of LIBERO), held-out LIBERO-Plus evaluation and a real robot arm. Each component is backed by a proposition (Propositions 4.1, 4.2, 4.3) and by the experiments of §5: the comparison with ten baselines and VLA-Adapter (Table 2), the ablation (Table 3), the component analyses (RQ3–RQ5), the case study (RQ6) and the real-robot study (RQ7).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix J (Limitations and future work) discusses the limitations: the grammar-defined operations and the one-slot fallback, the fixed plan and the monotone pointer, the gripper-based gate, and the evaluation scope (four LIBERO suites with five demonstrations per task, LIBERO-Plus, one real arm, VLA-Adapter as the backbone). The assumptions of each proposition are stated with it, and the evaluation protocol is specified in §5.1 and Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper contains three propositions, each stated with its assumptions in §4: the stage pointer splits every episode into at most M ordered contiguous segments, each non-final one lasting at least $d_{\min}$ decisions and left only once the gate certifies it (Proposition 4.1); with a single-primitive plan NS-VLA reduces exactly to its backbone (Proposition 4.2); and filtered advantage-weighted regression is the $\beta \to 0$ limit of advantage-weighted regression under binary task reward, with validation-based selection never below the supervised start (Proposition 4.3). Complete proofs are given in Appendix B.1, Appendix B.2 and Appendix B.3, respectively.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The architecture, the deterministic plan parser and segmentation rule, the pointer and gate thresholds, the supervised and online training schedules, the selection rule, the 10% demonstration split, the LIBERO-Plus sampling (seed 2026 and the pre-registered extension with seed 2027) and the evaluation protocol (seed 7, official initial states, episode limits) are documented in §5.1 and Appendices C, D, E and F (Table 6); the upstream code version of every baseline is given in Appendix D and its training configuration in Table 5. Public code, model checkpoints and dataset resources accompany the paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed

instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The public code repository provides the training, online-stage and evaluation scripts with their configuration files; the model checkpoints and released dataset are publicly available; LIBERO and LIBERO-Plus are accessed through their official releases, and the baseline codebases at the commits and versions listed in Appendix D.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The training split (first five demonstrations per task), the held-out LIBERO-Plus evaluation, the optimizer, learning rates, batch sizes, step counts, LoRA rank, chunk

length, sub-instruction probability, pointer thresholds, online-stage schedule (iterations, rollouts, exploration noise, demonstration ratio) and the validation-based checkpoint and iterate selection are specified in §5.1 and Appendix F; the baseline configurations are given in Appendix D.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Table 2 reports, for every suite and dimension, the binomial standard error over evaluation episodes, $\sqrt{p(1-p)/n}$, and for the averages the standard error propagated from these; a cell with no success shows the one-sided 95% Clopper–Pearson interval $[0, u]$ as $u/2 \pm u/2$, so no bar leaves $[0, 100]$, and the LIBERO-Long cell of VLA-Adapter is the mean over three training seeds with the standard error across seeds. The LIBERO-Plus comparison with VLA-Adapter is tested by an exact McNemar test over the 1,200 paired variants. The ablation reports the SR change of every variant with a 95% interval from a task-clustered paired bootstrap over the ten tasks (10,000 resamples), which keeps episodes paired across variants and makes no normality assumption (Table 7). The Solver analysis (Figure 6) reports the mean and sample standard deviation of test SR over three training seeds and the range over seeds of the validation curve. All comparisons use identical episodes for every method (§5.1, Appendix E).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: NS-VLA and VLA-Adapter are trained on a single NVIDIA H100 GPU per run in BF16. For NS-VLA, the supervised stage at batch 16 takes 2.6–4.6 h for 10k steps (Spatial, Object, Goal) and 5.2–5.4 h for 20k steps (Long) with a peak memory below 39 GB, and the online stage (8 iterations of 80 rollouts and 400 updates) takes 1.5–3.1 h. The ten baselines are trained on NVIDIA V100 (32 GB) GPUs with the per-suite configurations of Table 5, and all methods are evaluated on the same machines with EGL rendering (Appendices D and F). Preliminary and diagnostic runs that are not reported in the paper used additional compute of the same type.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The research uses publicly available simulation benchmarks and a tabletop robot arm in a laboratory setting, involves no human subjects or personal data, and conforms to the NeurIPS Code of Ethics in all respects.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Appendix K discusses the societal impact. Learning multi-step manipulation from five demonstrations per task lowers the cost of teaching robots household and assistive tasks, and the explicit plan makes every decision readable to an operator. As a negative impact, a misparsed instruction or a failed grasp can move objects unexpectedly; deployment should therefore follow standard robot-safety practice, with the observable stage pointer as a point for monitoring and human intervention, and NS-VLA reduces exactly to its backbone where an instruction has no multi-step structure.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate

to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The released artifact is a robotic manipulation policy fine-tuned on publicly available simulation benchmarks; it does not constitute a generative model, a scraped dataset, or any artifact that poses a high risk for misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All assets are publicly released by their original authors, cited in the paper and used under their terms: the LIBERO benchmark and demonstrations [43] (MIT), the LIBERO-Plus benchmark [19] (official release; no license file is declared in its repository), the VLA-Adapter backbone [79] (MIT) with its Qwen2.5-0.5B (Apache-2.0), DINOv2 (Apache-2.0) and SigLIP (Apache-2.0) components, and the baseline codebases and pre-trained weights: LIBERO (MIT), QueST (MIT), BAKU (MIT), Octo and octo-base-1.5 (MIT), and LeRobot with smolvla_base (Apache-2.0), with the pretrained BERT, T5 and MiniLM (Apache-2.0), CLIP (MIT) and SmolVLM2 (Apache-2.0) encoders they use, whose exact versions are listed in Appendix D.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The public code repository is released under the Apache-2.0 license and includes a README with installation instructions, training and evaluation commands, and configuration files. The model checkpoints and released dataset are documented on their respective artifact pages.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects; all experiments are conducted in simulation or with a robot arm operated by the authors.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects; no IRB approval or equivalent is required.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: NS-VLA builds on VLA-Adapter, whose Qwen2.5-0.5B language model, adapted with LoRA rank 64, is a core component of the policy and is described in §5.1 and Appendix F. The primitive plan is produced by the deterministic grammar parser of §4.1, so no language-model prompting is involved in plan extraction.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.