From Language to Action: Transferring the LLM Recipe to VLA Action Generation

Anonymous ACL submission

Abstract

The effectiveness of large language models rests on a systematic recipe: supervised fine-tuning (SFT) teaches instruction-following behavior, preference alignment (e.g., DPO) corrects distributional drift, and inference-time decoding strategies explore the output space for higher-quality solutions. This recipe has been refined extensively for text, but its transferability to other sequential generation domains remains an open question. The "Action-as-Language" paradigm in Vision-Language-Action (VLA) models offers a natural testbed: by tokenizing robotic actions into the LLM's vocabulary, action generation becomes formally identical to language generation. This equivalence invites a direct inquiry: *does the full LLM recipe transfer to action token generation, and does each stage yield the benefits observed in language?* We explore this by extending action generation to long-horizon token sequences and applying the recipe stage by stage—with adaptations tailored to the action domain: (1) **Long-horizon Behavior Cloning** mirrors SFT for extended action sequences; (2) **Step-wise Action Alignment** adapts DPO to correct compounding errors at their origin; and (3) **Inference-time decoding** applies standard LLM decoding strategies (e.g., top-k sampling and beam search) directly to action token streams. Our findings suggest the LLM recipe generalizes beyond language to sequential decision-making when the output space is properly tokenized.

1 Introduction

The success of large language models (LLMs) rests on a systematic recipe: supervised fine-tuning (SFT) on instruction–response pairs teaches desired output formats (Wei et al., 2021; Shengyu et al., 2023; Lou et al., 2024); preference alignment (e.g., DPO (Rafailov et al., 2023), RLHF (Ouyang et al., 2022)) corrects distributional drift; and inference-time decoding—beam search, top-k sampling, and variants (Fan et al., 2018; Wiher et al., 2022; Shi et al., 2024)—explores the output space at deployment. Each stage progressively improves generation quality, and together they form a pipeline validated across diverse language tasks (Lou et al., 2024; Shi et al., 2024). A natural question arises: *is this recipe specific to language, or does it reflect something more general about autoregressive sequence generation?*

Vision-Language-Action (VLA) models (Brohan et al., 2024; Kim et al., 2024) provide a natural testbed. The "Action-as-Language" paradigm discretizes robotic actions into LLM vocabulary tokens, making action generation *formally identical* to text generation—the same autoregressive next-token prediction, softmax output space, and cross-entropy objective (Zitkovich et al., 2023; Brohan et al., 2022; Kim et al., 2024). If the recipe's power stems from autoregressive generation rather than language-specific properties, it should transfer to action tokens with comparable effect. This equivalence offers a controlled setting in which each recipe stage can be isolated and measured in a non-language domain. We focus on autoregressive VLAs; diffusion- and flow-matching-based policies (Chi et al., 2023; Black et al., 2024) operate in continuous action spaces and lie outside this scope.

In this paper, we apply the recipe stage by stage to VLA action token generation, adapting each stage to the action domain (Section 3):

SFT → Long-horizon Behavior Cloning (L-BC). Just as instruction tuning teaches an LLM to produce coherent, extended responses (Wei et al., 2021; Shengyu et al., 2023), L-BC restructures expert demonstrations into observation-to-action-sequence pairs and fine-tunes the VLA to emit multiple action units from a single observation. The observation serves as the "instruction" and the concatenated future actions as the "response," extending the generation horizon beyond OpenVLA's con-

ventional single-step regime (Kim et al., 2024).

Preference Alignment → Step-wise Action Alignment (S-AA). In language modeling, preference alignment reduces mismatch between the model's output distribution and the desired target (Rafailov et al., 2023; Ouyang et al., 2022). In VLA action generation, a similar challenge appears as *compounding errors*: small deviations accumulate over sequential execution, causing the policy to drift from expert behavior (Bengio et al., 2015; Bachmann and Nagarajan, 2024). We adapt DPO (Rafailov et al., 2023) by rolling out the policy online, identifying the *first* step where the policy is overconfident relative to the expert—signaling deviation from expert behavior—and applying a localized DPO loss at that step. Unlike prior methods that correct long rollouts via backtracking (Cundy and Ermon, 2023) or guided test-time sampling (Liu et al., 2024b), S-AA targets drift at the earliest divergence.

Decoding → Inference-time Action Search. Greedy decoding is suboptimal for language generation (Holtzman et al., 2019); sampling and beam search substantially improve output quality (Shi et al., 2024; Vijayakumar et al., 2016). Because action generation uses the same next-token mechanism, top-k sampling and beam search transfer directly, enabling the policy to explore multiple candidate trajectories and select the most promising at deployment.

We present an empirical study on LIBERO (Liu et al., 2023) and real-world robotic tasks that instantiates each recipe stage in isolation and in combination. Across simulation and hardware, we observe stage-wise effects that qualitatively mirror language-model practice—extended horizons trade off against drift, alignment partially restores stability, and inference-time search can recover performance that longer rollouts otherwise lose. These findings support treating the LLM recipe as a general autoregressive toolkit for sequential decision-making, not a collection of language-specific tricks.

2 Preliminary

2.1 Vision-Language-Action (VLA) Models

Our work builds upon OpenVLA (Kim et al., 2024), a powerful open-source VLA model. We briefly introduce its core components below.

Architecture OpenVLA is built on a VLM framework, combining SigLIP (Zhai et al., 2023) and DINOV2 (Oquab et al., 2023) visual encoders with a Llama-2-7B LLM (Touvron et al., 2023) backbone for generation. At each time step, the policy takes an RGB image and language instruction as input. The image is encoded into embedding tokens while the instruction is tokenized via the Llama-2 tokenizer. These token sequences are concatenated and processed by the causal transformer decoder to generate outputs.

Action Tokenization and Vocabulary Remapping To enable an LLM to generate physical commands, OpenVLA bridges the continuous robot action space with the LLM's text generation capabilities through discretization. The 7-dimensional action vector (6-DoF end-effector displacement plus gripper state) is uniformly discretized into 256 bins per dimension following (Brohan et al., 2022, 2024), representing each action as seven integer indices. These indices are mapped to the 256 least-frequently used tokens in the Llama-2 vocabulary, transforming action prediction into a standard text generation task where the policy autoregressively predicts action tokens just as it would predict words in a sentence.

This autoregressive, discrete-token design is the key prerequisite for the recipe transfer studied in this paper: because action generation shares the same next-token prediction mechanism, softmax output space, and cross-entropy training objective as language generation, each component of the LLM recipe—SFT, preference alignment, and inference-time decoding—can in principle be applied directly (Kim et al., 2024; Rafailov et al., 2023; Shi et al., 2024). This stands in contrast to VLA models based on diffusion (Chi et al., 2023) or flow matching (Black et al., 2024; Intelligence et al., 2025), which generate actions as continuous vectors via iterative denoising and therefore operate outside the autoregressive framework that the LLM recipe assumes.

2.2 Direct Preference Optimization (DPO)

Direct Preference Optimization (DPO) (Rafailov et al., 2023) aligns LLMs with human preferences using static preference datasets, bypassing explicit reward models and online RL. Given preference triplets (x, y_w, y_l) and a frozen reference policy π_{ref}, DPO directly optimizes the policy by mini-

mizing:

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{\mathcal{D}}\Big[\log \sigma\big(\beta \log \tfrac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \tfrac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\big)\Big], \tag{1}$$

where β controls the strength of preference enforcement. A key insight of DPO is that the policy implicitly defines a reward function $R(y \mid x) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$, which we exploit in our action-domain adaptation (Section 3).

3 Transferring the LLM Recipe to Action Generation

3.1 The Action-as-Language Interface

The Action-as-Language design of OpenVLA establishes a formal equivalence between text generation and action generation that makes recipe transfer possible. We first define the notation, then spell out the mapping from LLM concepts to the action domain.

The state at time step t is $s_t = (v_t, l)$, where v_t is the current visual observation and l is the language instruction (fixed within an episode). A single action at time step t, termed an **action unit** $\mathbf{a}_t$, comprises K discrete tokens:

$$\mathbf{a}_t = (a_t^1, a_t^2, \ldots, a_t^K) \tag{2}$$

where K=7 (6-DoF end-effector displacement + 1 gripper state) and each a_t^k indexes a token in the LLM's vocabulary.

Under this interface, the standard LLM generation pipeline maps directly to action generation: the observation s_t plays the role of the *prompt*, action tokens play the role of *response tokens*, and generation proceeds via the same next-token prediction mechanism. This mapping is specific to autoregressive VLA models; diffusion- and flow-matching-based architectures operate in continuous action spaces and do not admit the same transfer (see Section 2). Current VLA practice, however, exercises this equivalence only minimally (Kim et al., 2024). The original OpenVLA policy π_{base} generates a single action unit per observation:

$$\pi_{\text{base}}(\mathbf{a}_t|s_t) = \prod_{k=1}^{K} P(a_t^k|s_t, a_t^{<k}) \tag{3}$$

This is analogous to an LLM that answers every prompt with a single, minimal token sequence—leaving the extended-response SFT, post-SFT alignment, and search-based decoding that are central to modern language model performance entirely untapped.

The remainder of this section instantiates each of these three recipe stages for action tokens and describes the adaptations required.

3.2 Stage 1: SFT → Long-Horizon Behavior Cloning

In standard LLM development, SFT trains the model to produce coherent, extended responses given an instruction (Wei et al., 2021; Shengyu et al., 2023). We test whether this stage transfers to action tokens by applying the same procedure: the observation serves as the instruction, and the supervised target is a contiguous token sequence formed from expert demonstrations. We term this instantiation **Long-horizon Behavior Cloning (L-BC)**.

Data Reformatting. Standard demonstrations provide single-step $(s_t, \mathbf{a}_t)$ pairs (Kim et al., 2024; Brohan et al., 2022). To mirror instruction tuning, we restructure them into $(s_t, A_{t,E}^H)$ pairs where the "response" is formed by concatenating H consecutive expert action units with a separator token `[;]` and flattening into a single token sequence:

$$A_{t,E}^H = \text{concat}\big(\mathbf{a}_t, \texttt{[;]}, \mathbf{a}_{t+1}, \texttt{[;]}, \ldots, \mathbf{a}_{t+H-1}, \texttt{[;]}\big) \tag{4}$$

This yields $\mathcal{D}_{\text{SFT}} = \{(s_t, A_{t,E}^H)\}$ from the original dataset $\mathcal{D} = \{(s_t, \mathbf{a}_t)\}$. The horizon H plays the role of maximum response length in language SFT, not an independently optimized planning horizon.

Training Objective. The state s_t is the "instruction" and $A_{t,E}^H$ is the ground-truth "response" (Shengyu et al., 2023). We fine-tune π_θ with the standard SFT cross-entropy loss:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{\mathcal{D}_{\text{SFT}}}\left[\sum_{j=1}^{L} \log P_\theta(a_j \mid s_t, a_{<j})\right] \tag{5}$$

where $L = |A_{t,E}^H|$ is the total number of tokens in the response. At inference, the resulting policy π_{SFT} autoregressively emits up to H action units before replanning. Generation is two-level autoregressive—tokens within each action unit, and action units within the response (Figure 1)—directly mirroring how an instruction-tuned LLM produces extended text.

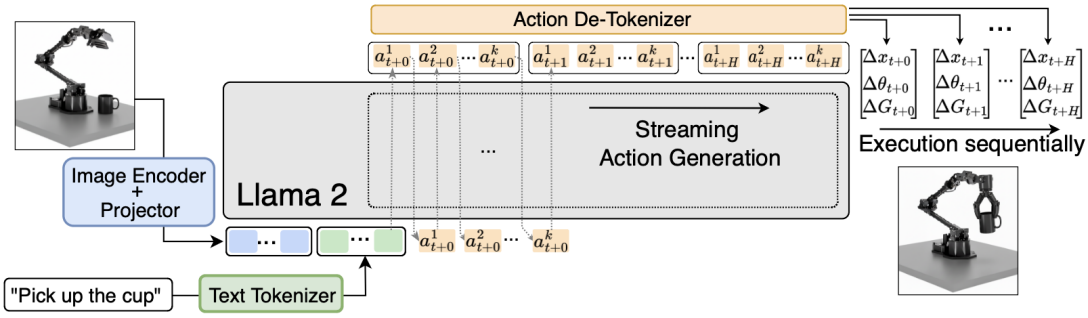

Figure 1: Action generation cast as autoregressive token generation, mirroring text generation. Under the Action-as-Language interface, the observation serves as the prompt and action tokens serve as the response, enabling each stage of the LLM recipe—SFT (L-BC), preference alignment (S-AA), and inference-time decoding—to be transferred directly to the action domain.

3.3 Stage 2: Preference Alignment → Step-wise Action Alignment

Motivation. Once SFT is applied to extended action responses, the policy must condition on its own outputs at deployment, and errors in early action units compound over the remainder of the response (Bengio et al., 2015; Bachmann and Nagarajan, 2024; Wulfmeier et al., 2024)—the same train–test mismatch that motivates preference alignment after SFT in language (Ziegler et al., 2019; Ouyang et al., 2022). We test whether the second recipe stage can mitigate this compounding error in the action domain.

Adaptation. We adapt Direct Preference Optimization (DPO) (Rafailov et al., 2023) to action sequences: the policy rolls out its own SFT-length responses online and is corrected at the earliest step of divergence from expert behavior.

Given a state s_t, the policy generates $\mathcal{A}^H_{t,\pi} = (\mathbf{a}_{t,\pi}, \ldots, \mathbf{a}_{t+H-1,\pi})$ and is compared against the expert target $\mathcal{A}^H_{t,E} = (\mathbf{a}_{t,E}, \ldots, \mathbf{a}_{t+H-1,E})$ constructed for SFT. We locate the first deviation using the implicit DPO reward:

$$R(\mathbf{a} \mid c) = \beta \log \frac{\pi_\theta(\mathbf{a} \mid c)}{\pi_{\text{ref}}(\mathbf{a} \mid c)} \tag{6}$$

Here, π_{ref} is the frozen post-SFT policy π_{SFT}. We flag deviation at step h^* when $R_\theta(\mathbf{a}_{\pi,h^*} \mid c_{h^*}) > R_\theta(\mathbf{a}_{E,h^*} \mid c_{h^*})$, indicating overconfidence in a suboptimal rollout relative to the expert. Unlike full-response DPO, we apply the loss only at h^*:

$$\mathcal{L}_{\text{S-AA}} = -\mathbb{E}_{\mathcal{D}}\Big[\mathbb{I}(h^* \text{ exists}) \cdot \log \sigma\big(R^*_E - R^*_\pi\big)\Big], \tag{7}$$

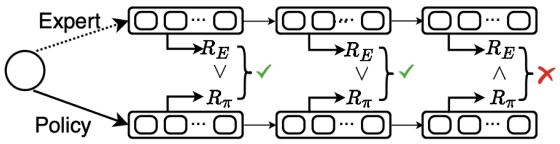

Figure 2: Illustration of S-AA: after SFT produces extended action responses, the policy rolls out its own sequence, compares implicit preference rewards step by step to find the first divergence from the expert, and applies a localized DPO loss at that step.

where $R^*_E = R_\theta(\mathbf{a}_{E,h^*} \mid c_{h^*})$, $R^*_\pi = R_\theta(\mathbf{a}_{\pi,h^*} \mid c_{h^*})$, and $\mathbb{I}(\cdot)$ activates the loss only when a first deviation exists.

This step-wise correction targets distributional drift at the first action unit where the policy's own SFT response diverges from expert behavior, analogous to how preference alignment in LLMs corrects unreliable continuations that supervised training alone cannot prevent (Ziegler et al., 2019; Rafailov et al., 2023). We term this adaptation **Step-wise Action Alignment (S-AA)**.

3.4 Stage 3: Decoding Strategies → Inference-time Action Search

The final recipe stage applies standard LLM inference to action tokens without architectural change (Shi et al., 2024). **Top-k sampling** draws each token from the k highest-probability choices autoregressively; with $H > 1$, the longer response supports diverse trajectory rollouts (Fan et al., 2018; Wiher et al., 2022). **Beam search** keeps B partial sequences at each action-unit boundary, expands them in parallel, and executes the rollout with highest cumulative log-probability (Vijayakumar et al., 2016); we branch beams at action-unit rather than token boundaries.

Table 1: Success rates across different generation horizons, four LIBERO task suites, and training stages. Notably, under both the L-BC and L-BC+S-AA training paradigms, the models maintain higher success rates even as the generation horizon doubles.

	Horizon	Spatial	Object	Goal	Long	Average
OpenVLA	1	84.4±0.9%	86.6±0.8%	77.2±1.0%	53.7±1.3%	75.5±1.0%
L-BC	×2	83.6±0.8%	85.6±0.9%	78.4±0.9%	53.4±1.2%	75.2±0.9%
	×4	79.4±1.0%	81.4±0.7%	75.4±0.9%	52.4±1.1%	72.2±0.9%
	×6	73.2±0.9%	76.2±0.9%	67.8±0.8%	48.0±1.0%	66.3±0.9%
	×8	64.6±1.0%	71.2±1.1%	58.2±0.7%	39.6±0.9%	58.4±0.9%
	×10	53.0±1.2%	62.8±1.4%	48.8±1.3%	27.2±1.2%	47.9±1.3%
L-BC+S-AA	×2	**85.4±0.8%**	**89.0±0.7%**	**80.0±0.9%**	**57.6±1.1%**	**78.0±0.9%**
	×4	82.0±0.7%	84.0±0.7%	76.8±0.9%	**54.2±1.0%**	74.2±0.8%
	×6	75.4±0.6%	78.0±0.6%	70.0±0.8%	47.6±0.9%	67.8±0.7%
	×8	65.8±0.5%	73.6±0.5%	60.6±0.7%	41.2±0.8%	60.3±0.6%
	×10	55.6±0.4%	66.0±0.4%	52.0±0.6%	29.6±0.7%	50.8±0.5%

4 Experiments

4.1 Setup

We evaluate on the LIBERO dataset (Liu et al., 2023), which consists of four task suites (each containing 10 tasks with 50 human-teleoperated demonstrations): LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long. To validate the findings beyond simulation, we also conduct experiments on a **Kinova Jaco2** 6-DoF robotic arm with a parallel-jaw gripper. Four table-top manipulation tasks are designed, covering pick-and-place and wiping motions with diverse objects. Details can be found in the Appendix. We use OpenVLA (Kim et al., 2024) as our backbone with LoRA (Hu et al., 2022) fine-tuning. For Stage 1 (L-BC), we follow the standard SFT paradigm on an A6000 GPU with batch size 4, context length 512, learning rate 5e-4, and train for 50,000 steps. For Stage 2 (S-AA), we use training trajectories as expert demonstrations with $\beta = 0.1$ and learning rate 0.0005, continuing to fine-tune the same LoRA parameters. Additional details are in the Appendix.

4.2 Stages 1 & 2: SFT and Preference Alignment Transfer

We test the first two stages of the recipe by asking two questions. First, *can action token generation maintain quality when the SFT response is extended*, as routinely done in language (Shengyu et al., 2023)? Second, *does preference alignment mitigate the compounding errors introduced by longer autoregressive generation*, mirroring its role in language modeling? We vary the horizon H and compare two settings: **L-BC only** (Stage 1 alone), which isolates the effect of extended-response SFT; and **L-BC + S-AA** (Stages 1+2), which adds preference alignment to answer the second question.

Results. Table 1 shows how horizon length H affects task success rates across training regimes, including a single-step OpenVLA baseline (Kim et al., 2024). For L-BC only models, performance shows minimal decline when H increases from 1 to 2 (0.3% difference) and modest reduction at $H = 4$ (3.1% decrease). However, performance drops sharply at longer horizons: $H = 6$ (9.8% decrease), $H = 8$ (17.1% decrease), and $H = 10$ (27.6% decrease). This answers the first question affirmatively but with a clear boundary: *the SFT stage transfers well at moderate horizons* ($H \leq 4$), where extending the response length preserves task performance, but the compounding errors inherent in teacher-forced training become dominant at longer horizons—paralleling the well-known quality degradation of SFT-only LLMs when generating very long responses (Holtzman et al., 2019; Liu et al., 2024a).

In contrast, adding S-AA (Stage 2) yields notable improvements. When H increases from 1 to 2, performance actually improves from 75.2% to 78.0%, indicating that preference alignment not only sustains extended generation but enhances overall task success. At every subsequent horizon (H=4 through H=10), S-AA consistently degrades more slowly than L-BC only—for example, at H=4 S-AA drops only 1.3% from H=2 versus L-BC's 3.2%, and the gap widens at longer horizons. These results answer the second ques-

Table 2: Pass@N results across four LIBERO task suites and different generation horizons. Notably, while improvements are modest at horizon 1, extending the horizon yields substantial gains in Pass@N, indicating more thorough exploration of the solution space.

		Horizon	Spatial	Object	Goal	Long	Average
OpenVLA	Pass@2	1	85.8±1.0%	89.6±0.8%	81.4±1.2%	54.8±1.1%	77.9±1.0%
	Pass@5	1	86.6±0.7%	90.4±1.1%	83.2±0.9%	57.4±1.0%	79.4±0.9%
L-BC+S-AA	Pass@2	×2	94.4±0.8%	92.0±0.9%	93.6±0.7%	80.6±1.2%	90.2±0.9%
		×4	92.8±1.0%	92.6±0.8%	89.2±1.1%	76.4±0.9%	87.8±1.0%
		×6	90.8±0.9%	86.0±1.2%	82.4±1.0%	69.2±1.1%	82.1±1.1%
		×8	82.2±1.1%	85.4±0.7%	72.0±1.2%	62.4±1.0%	75.5±1.0%
		×10	71.8±1.2%	72.0±1.1%	64.4±0.9%	46.6±1.3%	63.7±1.1%
	Pass@5	×2	98.8±0.6%	97.6±0.7%	95.2±0.8%	87.4±0.9%	94.8±0.8%
		×4	97.4±0.7%	96.4±0.8%	93.6±0.9%	82.8±1.1%	92.6±0.9%
		×6	96.6±0.8%	95.8±0.9%	88.2±1.0%	76.6±1.2%	89.3±1.0%
		×8	88.4±1.0%	91.2±0.9%	85.4±1.1%	69.4±1.2%	83.6±1.1%
		×10	82.2±1.1%	84.6±1.0%	83.2±0.9%	58.2±1.3%	77.1±1.1%

tion: *preference alignment transfers to action tokens with the same corrective effect observed in language* (Ziegler et al., 2019; Rafailov et al., 2023; Ouyang et al., 2022)—it mitigates the distributional drift that SFT alone cannot resolve, shifting the usable horizon boundary outward.

4.3 Stage 3: Inference-time Action Search

Having established that the training-time stages (SFT and alignment) transfer to action tokens, we now evaluate the final recipe stage: inference-time decoding. We apply the stochastic sampling and beam search strategies described in Section 3 and examine whether they yield the quality gains observed in language generation.

Stochastic Sampling (Pass@N). We use top-k sampling (k=3) and evaluate with the $Pass@N$ metric (Chen et al., 2021), which counts a success if at least one of N sampled trajectories completes the task. Table 2 presents results across task suites and horizon lengths. For $H = 2$, success rates increase by 8.6%, and notably, $Pass@2$ average performance at $H = 6$ (82.1%) surpasses the $H = 1$ baseline (77.9%). With $N = 5$, all metrics show substantial improvements, with gains reaching up to 15.4% for longer horizons. A particularly revealing contrast emerges with the single-step baseline (H=1), which shows minimal benefits from $Pass@N$. This is expected: a 7-token response offers very limited room for sampling variation, whereas extended-horizon responses contain $7 \times H$ tokens with compounding branching points, producing a substantially richer distribution over trajectories. This mirrors a well-known property of language generation: sampling benefits grow with response length, as longer sequences offer more opportunities for the model to explore diverse continuations (Shi et al., 2024).

Beam Search. The high $Pass@N$ results confirm that the extended-horizon policy produces multiple viable trajectories. Beam search (Section 3) systematically exploits this by maintaining B candidates and selecting the highest-scoring complete sequence.

Figure 3 shows the beam search results under different beam width B. Beam search significantly improves performance across all horizon lengths. With $B = 2$, performance at $H = 2$ already surpasses the single-step baseline across all task suites. At $B = 4$ and $B = 5$, even $H = 4$ exceeds the baseline—meaning that a four-fold increase in response length is fully compensated by search. Moreover, as beam width increases, the performance degradation curve with respect to H flattens considerably. This result is noteworthy: it demonstrates that inference-time compute—the final stage of the LLM recipe—transfers to action generation and can recover performance that would otherwise be lost to compounding errors, just as beam search in language generation compensates for the noise of sampling-based decoding (Vijayakumar et al., 2016; Shi et al., 2024).

4.4 Trajectory Smoothness: An Emergent Benefit

Figure 4 shows the robot end-effector movement trajectories under different horizon lengths. When the generation horizon is extended, trajectories be-

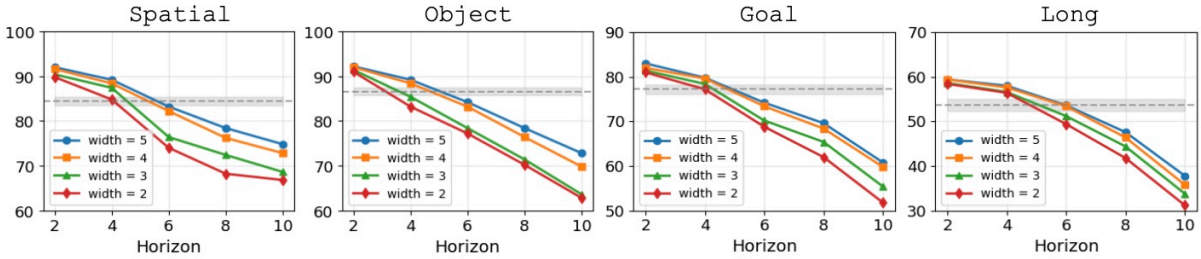

Figure 3: Beam search results under different beam widths across four LIBERO task suites. Gray bars indicate the single-step baseline (H=1). As beam width increases, even at H=4 the policy surpasses the baseline, demonstrating that inference-time search transfers effectively to action generation.

Table 3: Real robot task success rates (%) with different horizon lengths. With extended-horizon generation (H=2), performance improves over the single-step baseline.

	Horizon	PlaceBlock	PlaceCarrot	CleanTable	RelocateCup	Average
OpenVLA	1	85.0	90.0	80.0	75.0	82.5
L-BC + S-AA	×2	85.0	95.0	90.0	80.0	**87.5**
	×3	80.0	85.0	85.0	75.0	81.25
	×4	70.0	85.0	80.0	70.0	76.25
	×5	65.0	75.0	70.0	65.0	68.75

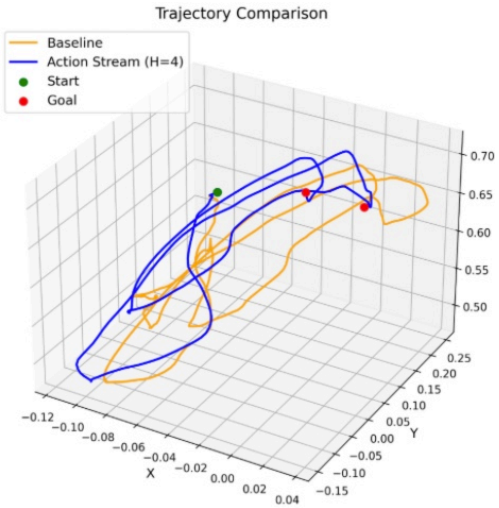

Figure 4: Visualization of robot end-effector trajectories under different generation horizons. Longer horizons lead to noticeably smoother and more coherent motion compared to conventional step-wise prediction.

come noticeably smoother, whereas single-step prediction (H=1) exhibits frequent spikes and discontinuities. The mechanism is analogous to a well-known property of autoregressive text generation: conditioning on a longer self-generated prefix promotes coherence within the response (Holtzman et al., 2019). Under single-step prediction, each action is generated independently given a new observation, leading to potential inconsistencies across timesteps (Liu et al., 2024b). Under extended-horizon generation, successive action units are conditioned on the preceding units within the same autoregressive response, enforcing temporal consistency—an emergent benefit of applying the SFT stage with longer responses, rather than an explicitly engineered objective. More analysis can be found in the Appendix.

4.5 Results on Real Robot

Using an Xbox Gamepad, we teleoperate the Jaco2 arm and collect demonstrations at a frequency of 10 Hz, recording RGB images and robot states throughout each trajectory. We design four distinct tasks for evaluation: PlaceBlock, PlaceCarrot, CleanTable, and RelocateCup. Figure 5 shows some sample visual observations for the RelocateCup and PlaceBlock tasks. We gather 50–200 demonstrations per task to form the training dataset and apply the two training stages (L-BC followed by S-AA). Table 3 shows the results. The pattern observed in simulation transfers to the real world: at H=2 the extended-horizon policy outperforms the single-step baseline across all four tasks, with an average improvement of 5.0%. Performance then degrades gradually as H increases further, consistent with the compounding-error trend seen in LIBERO. This cross-domain consistency strengthens the conclusion that the recipe transfer is not an artifact of the simulation environment but reflects a genuine property of autoregressive action gener-

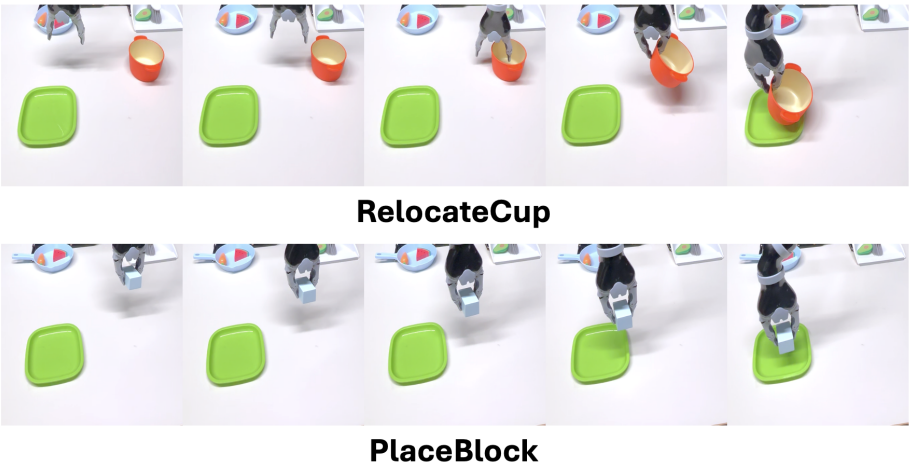

Figure 5: Visual observations for the RelocateCup and PlaceBlock tasks in the real robot environment.

ation (Kim et al., 2024). More task and training details can be found in the Appendix.

5 Related Work

5.1 Transforming VLM to VLA

Recent VLA models (Kim et al., 2024; Wang et al., 2025) adapt VLMs for robotic control via two approaches: output space unification, which maps all modalities into a shared token space, and module grafting, which attaches specialized action heads to VLMs.

The unification approach serializes all modalities into a shared discrete token space for autoregressive modeling (Reed et al., 2022), with the RT series (Brohan et al., 2022, 2024; Belkhale et al., 2024) and Open X-Embodiment (O'Neill et al., 2024) further advancing action discretization and cross-robot standardization. These methods preserve the original architecture, enabling end-to-end training. The grafting approach instead attaches specialized modules—e.g., variational decoders (Lynch and Sermanet, 2020), separate MLP heads (Jang et al., 2022), or hybrid autoregressive-diffusion policies (Liu et al., 2025)—to pre-trained VLMs, at the cost of added design complexity.

5.2 Post-training and Decoding for Action Generation

Individual components of the LLM training recipe have been explored for robotic action generation, though in isolation. On the SFT side, LongVLA (Fan et al., 2025) and MuST (Gao et al., 2025) extend the generation horizon by segmenting tasks or decomposing them into reusable skills, mirroring how instruction tuning teaches LLMs to produce extended responses. For decoding, diffusion- and flow-based VLAs (Chi et al., 2023; Kim et al., 2025; Black et al., 2024; Intelligence et al., 2025) employ action chunking to generate multi-step actions in parallel, while Liu et al. (2024b) improve autoregressive chunking via guided test-time sampling. On the alignment side, Cundy and Ermon (2023) mitigate compounding errors in autoregressive imitation through backtracking. However, none of these works systematically transfer the full LLM post-training recipe—SFT, preference alignment, and decoding—under a unified action-as-language interface.

6 Conclusion

We studied whether the LLM training and inference recipe—SFT, preference alignment, and inference-time decoding—transfers to action token generation under the Action-as-Language paradigm. On LIBERO and real-world robotic tasks, each stage yields additive gains: L-BC supports coherent multi-step generation at moderate horizons; S-AA mitigates compounding errors beyond SFT alone; and beam search recovers performance at longer horizons. These results suggest the recipe reflects properties of autoregressive generation rather than language-specific mechanisms, when the output space is properly tokenized. For robotics, this implies that advances in LLM post-training—new alignment objectives, more efficient search algorithms, and scaling strategies—can be directly leveraged to improve robot policy learning, reducing the need for domain-specific algorithmic design. We hope this inspires applying LLM techniques to other sequential decision tasks.

7 Limitations

Our study has several limitations. First, the recipe transfer is specific to autoregressive VLA models that adopt the Action-as-Language interface; architectures based on diffusion or flow matching (Chi et al., 2023; Black et al., 2024) operate in continuous action spaces and would require fundamentally different alignment and decoding strategies. Second, action discretization introduces quantization errors that accumulate over longer sequences, constraining the upper performance bound for tasks requiring fine-grained manipulation (Liu et al., 2025). Third, extended multi-step autoregressive generation incurs higher computational cost compared to single-step prediction, which may increase deployment latency in time-critical applications. Finally, our experiments are conducted on a single VLA backbone (OpenVLA) and evaluated primarily on the LIBERO benchmark; the generalizability of these findings to other VLA architectures, action tokenization schemes, and more diverse task domains remains to be validated.

References

Gregor Bachmann and Vaishnavh Nagarajan. 2024. The pitfalls of next-token prediction. *arXiv preprint arXiv:2403.06963*.

Suneel Belkhale, Tianli Ding, Ted Xiao, Pierre Sermanet, Quon Vuong, Jonathan Tompson, Yevgen Chebotar, Debidatta Dwibedi, and Dorsa Sadigh. 2024. Rt-h: Action hierarchies using language. *arXiv preprint arXiv:2403.01823*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, and 1 others. 2024. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, and 1 others. 2024. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023. *URL https://arxiv. org/abs/2307.15818*.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, and 1 others. 2022. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*.

Mark Chen, Jerry Tworek, Hewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. 2023. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668.

Chris Cundy and Stefano Ermon. 2023. Sequencematch: Imitation learning for autoregressive sequence modelling with backtracking. *arXiv preprint arXiv:2306.05426*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898.

Yiguo Fan, Pengxiang Ding, Shuanghao Bai, Xinyang Tong, Yuyang Zhu, Hongchao Lu, Fengqi Dai, Wei Zhao, Yang Liu, Siteng Huang, and 1 others. 2025. Long-vla: Unleashing long-horizon capability of vision language action model for robot manipulation. *arXiv preprint arXiv:2508.19958*.

Kai Gao, Fan Wang, Erica Aduh, Dylan Randle, and Jane Shi. 2025. Must: Multi-head skill transformer for long-horizon dexterous manipulation with skill progress. *arXiv preprint arXiv:2502.02753*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, and 1 others. 2025. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*.

Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. 2022. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR.

Moo Jin Kim, Chelsea Finn, and Percy Liang. 2025. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, and 1 others. 2024. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. 2023. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791.

Jiaming Liu, Hao Chen, Pengju An, Zhuoyang Liu, Renrui Zhang, Chenyang Gu, Xiaoqi Li, Ziyu Guo, Sixiang Chen, Mengzhen Liu, and 1 others. 2025. Hybridvla: Collaborative diffusion and autoregression in a unified vision-language-action model. *arXiv preprint arXiv:2503.10631*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173.

Yuejiang Liu, Jubayer Ibn Hamid, Annie Xie, Yoonho Lee, Maximilian Du, and Chelsea Finn. 2024b. Bidirectional decoding: Improving action chunking via guided test-time sampling. *arXiv preprint arXiv:2408.17355*.

Renze Lou, Kai Zhang, and Wenpeng Yin. 2024. Large language model instruction following: A survey of progresses and challenges. *Computational Linguistics*, 50(3):1053–1095.

Corey Lynch and Pierre Sermanet. 2020. Language conditioned imitation learning over unstructured data. *arXiv preprint arXiv:2005.07648*.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, and 1 others. 2023. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, and 1 others. 2024. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, and 1 others. 2022. A generalist agent. *arXiv preprint arXiv:2205.06175*.

Zhang Shengyu, Dong Linfeng, Li Xiaoya, Zhang Sen, Sun Xiaofei, Wang Shuhe, Li Jiwei, Runyi Hu, Zhang Tianwei, Fei Wu, and 1 others. 2023. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

Chufan Shi, Haoran Yang, Deng Cai, Zhisong Zhang, Yifan Wang, Yujiu Yang, and Wai Lam. 2024. A thorough examination of decoding methods in the era of llms. *arXiv preprint arXiv:2402.06925*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*.

Yuqi Wang, Xinghang Li, Wenxuan Wang, Junbo Zhang, Yingyan Li, Yuntao Chen, Xinlong Wang, and Zhaoxiang Zhang. 2025. Unified vision-language-action model. *arXiv preprint arXiv:2506.19850*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Gian Wiher, Clara Meister, and Ryan Cotterell. 2022. On decoding strategies for neural text generators. *Transactions of the Association for Computational Linguistics*, 10:997–1012.

Markus Wulfmeier, Michael Bloesch, Nino Vieillard, Arun Ahuja, Jorg Bornschein, Sandy Huang, Artem Sokolov, Matt Barnes, Guillaume Desjardins, Alex Bewley, and 1 others. 2024. Imitating language via scalable inverse reinforcement learning. *Advances in Neural Information Processing Systems*, 37:90714–90735.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. 2023. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11975–11986.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Long Ouyang, Ryan Lowe, Jan Leike, Paul Christiano, Geoffrey Irving, and Dario Amodei. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, and 1 others. 2023. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR.

A Appendix

A.1 Use of Large Language Models

We use LLMs only for text polishing and language refinement to improve the clarity and readability of our manuscript. All core ideas, experimental designs, methodological contributions, and technical innovations presented in this work are conceived and developed entirely by human researchers. The LLMs were not involved in any aspect of the research process, including problem formulation, algorithm design, experimental setup, data analysis, or result interpretation. Their usage was strictly limited to enhancing the linguistic quality of the written content.

A.2 Real robot experiments settings

Hardware and setup. We use a Kinova Jaco2 arm for real-world evaluation. Demonstrations are collected via Xbox Gamepad teleoperation at a control frequency of 10 Hz, recording synchronized RGB images and robot states at each step. We design four manipulation tasks: 1) **PlaceBlock**: Place the blue cube on the green plate. 2) **PlaceCarrot**: Place the carrot on the blue pan. 3) **CleanTable**: Clean the table. 4) **RelocateCup**: Pick up the orange pot and place it in the green plate. The visual observations for each task are shown in Figure 6.

Training data. For each task, we collect a set of human teleoperated demonstrations to construct the training dataset: 198 trajectories for PlaceBlock (average length 75 steps), 201 for PlaceCarrot (81 steps), 69 for CleanTable (58 steps), and 179 for RelocateCup (150 steps). We then apply the two-stage recipe—L-BC followed by S-AA—described in Section 3.

Evaluation protocol. For each trained policy, we perform 20 independent trials per task (80 trials in total). At the start of each trial, the robot is reset to a standardized home configuration, and the objects are placed at randomized but feasible positions on the workspace. The policy then executes until task completion or a timeout of 30 seconds.

Success criteria. A trial is counted as successful if the task-specific goal is satisfied: (i) the target object is fully placed inside the designated container (PlaceBlock, PlaceCarrot, RelocateCup), or (ii) the distractor object is completely removed from the designated region (CleanTable). The object must remain stable in the target region for at least one second to be considered a success. Success rates reported in Table 3 are computed as the fraction of successful trials over the total trials.

A.3 Analyzing gains from the LLM recipe on VLA

We summarize how each stage of the LLM recipe improves OpenVLA (Kim et al., 2024) beyond its default single-step ($H{=}1$) regime. All results use the same Action-as-Language backbone and LoRA fine-tuning; gains therefore isolate recipe transfer rather than architectural change.

Stage 1 (L-BC): extended-response SFT. Table 1 shows that L-BC preserves task success when the response is modestly extended: average success drops only 0.3% from $H{=}1$ to $H{=}2$ (75.5% $\rightarrow$ 75.2%) and 3.1% to $H{=}4$ (72.2%), confirming that the first recipe stage transfers to action tokens at moderate horizons. Beyond $H{=}4$, compounding errors dominate (e.g., 27.6% drop from $H{=}1$ to $H{=}10$), paralleling SFT-only LLMs on very long responses (Holtzman et al., 2019; Liu et al., 2024a). Alongside these task metrics, Stage 1 yields an *emergent* motion-quality gain: extended-horizon policies produce smoother end-effector trajectories (Section 4.4 and below), even though smoothness is not explicitly optimized—mirroring coherence improvements from extended SFT in language.

Stage 2 (S-AA): preference alignment. Adding S-AA shifts the usable horizon boundary outward. At $H{=}2$, average success *improves* from 75.2% (L-BC only) to 78.0% (+2.8%), outperforming the single-step OpenVLA baseline (75.5%). At longer horizons, S-AA degrades more slowly than L-BC alone: at $H{=}4$, success falls 1.3% from $H{=}2$ under L-BC+S-AA versus 3.2% under L-BC only;

Figure 6: Visual observations for the four real-robot evaluation tasks.

at $H{=}10$, L-BC+S-AA retains 50.8% average success versus 47.9% for L-BC only (+2.9%). These gains mirror preference alignment in LLMs: correcting distributional drift at the earliest divergence step mitigates compounding errors that extended SFT cannot fully prevent (Ziegler et al., 2019; Rafailov et al., 2023).

Stage 3: inference-time decoding. Training-time stages expand the *distribution* over action-token responses; decoding exploits it at deployment. With top-k sampling ($k{=}3$), Pass@2 at $H{=}2$ reaches 90.2% average success (+12.3% over the 77.9% single-step baseline; Table 2), and Pass@2 at $H{=}6$ (82.1%) already exceeds the $H{=}1$ baseline—showing that longer responses create a richer trajectory space for sampling to exploit, as in language (Shi et al., 2024). Pass@5 yields up to 15.4% additional gains at long horizons; the single-step baseline benefits minimally from Pass@N, since a 7-token response affords little sampling diversity. Beam search (Figure 3) further recovers performance: with $B{=}2$, $H{=}2$ already surpasses the single-step baseline across all LIBERO suites; at $B{\in}\{4, 5\}$, even $H{=}4$ exceeds the baseline, compensating for compounding errors via inference-time compute—the same role beam search plays in language generation (Vijayakumar et al., 2016; Shi et al., 2024).

Real-robot validation. Table 3 shows the same stage-wise pattern on hardware: L-BC+S-AA at $H{=}2$ improves average success from 82.5% (OpenVLA, $H{=}1$) to 87.5% (+5.0%) across four tasks, with gains on PlaceCarrot (+5.0%) and CleanTable (+10.0%). Performance then degrades gradually as H increases, consistent with LIBERO. Together, simulation and real-world results indicate that the LLM recipe provides *additive, stage-wise* benefits on VLA: L-BC enables extended autoregressive action generation, S-AA stabilizes longer horizons, and inference-time search unlocks performance that training alone cannot reach at deployment.

A.4 Extended-horizon trajectory analysis

This section supplements Section 4.4 and provides mechanistic context for the Stage 1 gains in Section A.3. We visualize end-effector trajectories from LIBERO and examine two consequences of extending the SFT response length in Stage 1 (L-BC): improved temporal coherence and compounding drift at longer horizons. We compare extended-horizon generation ($H>1$) against the OpenVLA single-step baseline ($H=1$).

Temporal coherence (Stage 1). Figure 7 illustrates end-effector trajectories for both settings during task execution. The single-step baseline exhibits noticeable **trajectory spikes and discontinuities (highlighted in red boxes)**, reflecting weak temporal coherence when each action unit is predicted independently under a fresh observation (Liu et al., 2024b). In contrast, L-BC generates **significantly smoother trajectories**: successive action units are conditioned on preceding units within the same autoregressive response, mirroring how extended SFT responses promote coherence in language generation (Holtzman et al., 2019). This smoothness is an emergent benefit of applying the first recipe stage with longer responses, not an explicitly optimized objective.

Trajectory drift and the momentum effect (motivation for Stage 2). Our analysis also reveals a distinctive momentum characteristic in extended-horizon generation. As shown in Figure 8, both settings exhibit similar initial trajectories, but the extended-horizon policy tends to persist on its pre-computed plan because the full H-step response is conditioned on the initial observation. Unlike single-step prediction, which replans at every timestep, extended-horizon generation operates open-loop within each response, creating executional momentum: the policy maintains its planned trajectory even as the environment changes, which can yield drift when the initial plan becomes suboptimal. This trade-off—coherence versus sensitivity to state change—parallels the train–test mismatch that motivates preference alignment after SFT in language (Ziegler et al., 2019; Rafailov et al., 2023) and motivates Stage 2 (S-AA) in our recipe.

A.5 Recipe Training Details

Stage 1: Long-horizon Behavior Cloning (L-BC) In Stage 1, we follow the standard SFT paradigm commonly used in LLMs. We randomly sample an expert demonstration, which includes image embeddings and text token embeddings as input, and concatenate the subsequent multi-step actions as supervised output targets. The training context length is set to 512 tokens. If the concatenated multi-step action sequence does not fill the entire context length, we pad it with placeholder tokens that do not participate in the loss function computation. Figure 9 shows the L-BC training curves. The model converges with steadily decreasing loss and accuracy approaching 100%, indicating that the first recipe stage successfully teaches extended action-token responses.

Stage 2: Step-wise Action Alignment (S-AA) Since S-AA requires online policy exploration and OpenVLA's setup does not support batch inference, we use a batch size of 1 during this phase. We sample a state from expert demonstrations and obtain the subsequent H-step action sequence, while also allowing the policy to explore for 10 steps (generating 10×8=80 tokens, consisting of 7 action dimensions plus 1 separator token per step). We then compute the implicit DPO rewards R_θ for each step to identify the first error. DPO loss is applied only to the first error step. For training stability, we maintain a 9:1 ratio between the S-AA (DPO) loss and the L-BC (SFT) loss. If no first error is identified, we fall back to L-BC fine-tuning on the expert demonstration.

During the training process, we monitor two key metrics: the expected action reward (representing expert demonstrations) and the policy's online exploration reward. Figure 10 illustrates the convergence curves for these two rewards, where "winner" represents the expected expert actions and "loser" represents the policy-generated actions. As training progresses, both rewards show an upward trend, indicating that the policy increasingly favors expert actions while simultaneously generating actions that closely align with expert behavior. This dual improvement demonstrates that Stage 2 (S-AA) effectively corrects distributional drift: the policy increasingly prefers expert demonstrations while its own rollouts become more expert-like, mirroring the effect of preference alignment in language models (Rafailov et al., 2023).

We select the optimal checkpoint for evaluation based on the convergence behavior of both expert and policy rewards. Specifically, we monitor the reward gap between expert demonstrations (winner) and policy-generated actions (loser) throughout training. The checkpoint is selected when: (1)

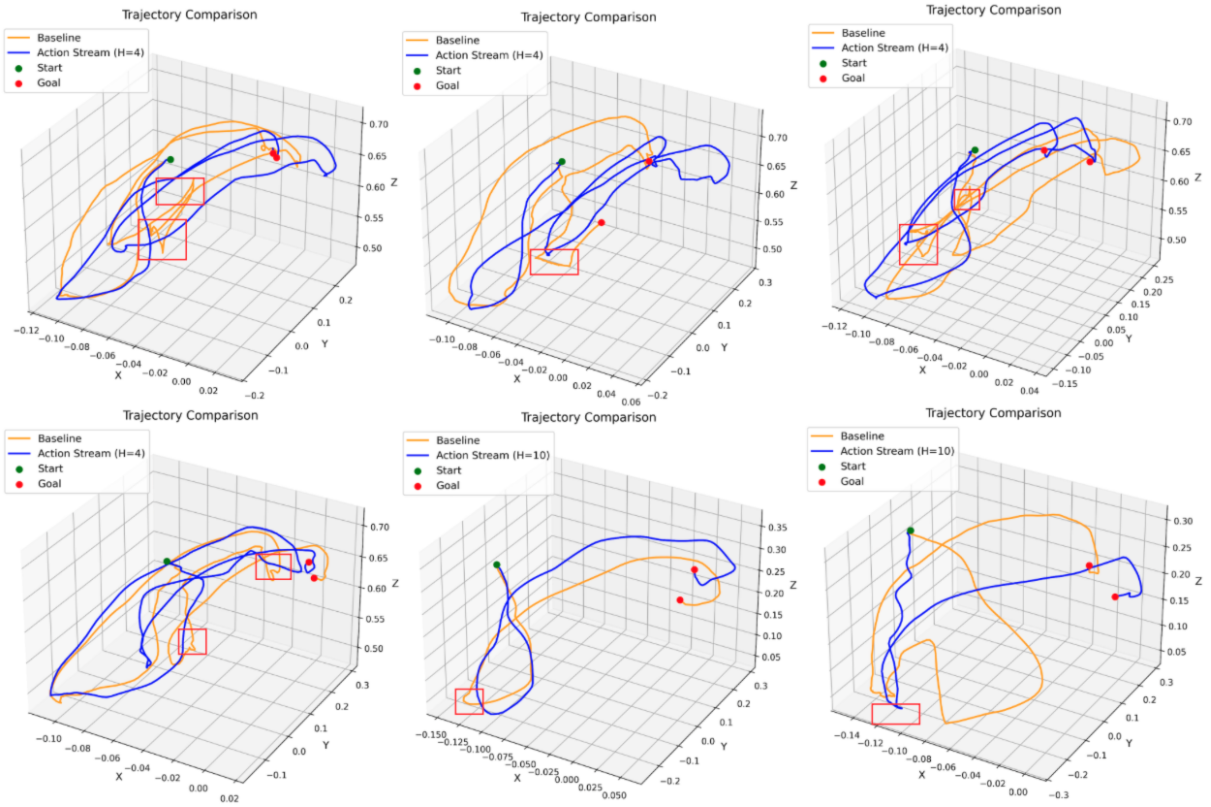

Figure 7: The action trajectory analysis results. The baseline step-wise action prediction exhibits trajectory spikes and abrupt directional changes (highlighted in red boxes), demonstrating weak temporal coherence in single-step prediction.

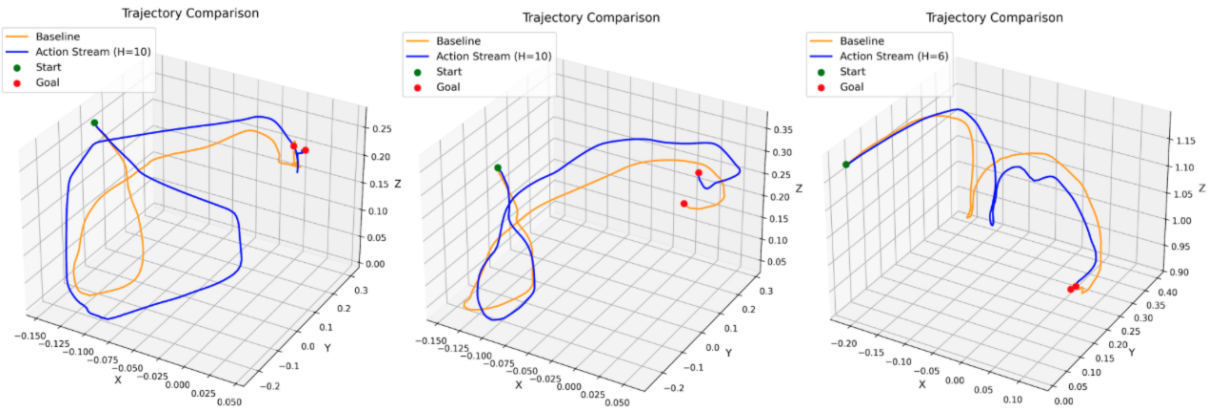

Figure 8: Trajectory drift under extended-horizon generation. Both settings exhibit overlapping initial trajectories, but L-BC ($H>1$) drifts over time due to open-loop execution within each autoregressive response, motivating Stage 2 (S-AA).

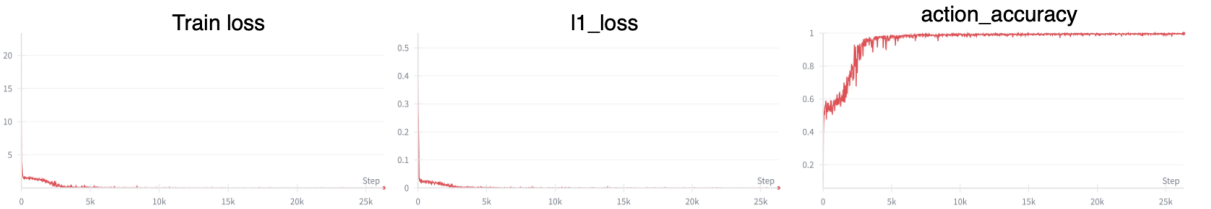

Figure 9: L-BC (Stage 1) training curves. Loss decreases steadily and accuracy approaches 100%, confirming successful transfer of extended-response SFT to action tokens.

the reward gap between expert and policy actions becomes minimal, indicating that policy-generated actions closely match expert quality, and (2) both reward curves exhibit stability without significant fluctuations, suggesting convergence. This selection strategy ensures that we evaluate the model at its optimal performance point where the policy has learned to generate expert-like actions consistently.

A.6 LIBERO Simulation Benchmark Settings

LIBERO (Liu et al., 2023) is a recent benchmark for embodied robot learning, designed to evaluate generalization across diverse manipulation scenarios. It consists of four task suites, each containing 10 subtasks with 50 human-teleoperated demonstrations per task. The suites differ in object diversity, spatial variation, and goal specification, thereby testing complementary aspects of policy robustness:

- **LIBERO-Spatial**: Same set of objects, but placed in different spatial layouts. This suite evaluates the ability to generalize across spatial configurations and adapt to changes in object positions and relative distances.
- **LIBERO-Object**: Same spatial layouts, but involving different objects. This suite tests object recognition and manipulation transfer when new but semantically similar objects are introduced.
- **LIBERO-Goal**: Same objects and layouts, but with varying goals or instructions. This suite emphasizes understanding task semantics and aligning actions with different high-level task objectives.
- **LIBERO-Long**: Also known as LIBERO-10, containing long-horizon tasks with multiple objects, diverse layouts, and compound goals. This suite is the most challenging, as it requires coherent multi-step planning and strong temporal consistency to succeed.

Together, these four suites provide a comprehensive evaluation: LIBERO-Spatial and LIBERO-Object measure generalization to environment or object shifts, LIBERO-Goal probes semantic grounding, and LIBERO-Long stresses long-horizon reasoning and error accumulation. Figure 11 shows example episodes from each suite, illustrating their diversity and complexity.

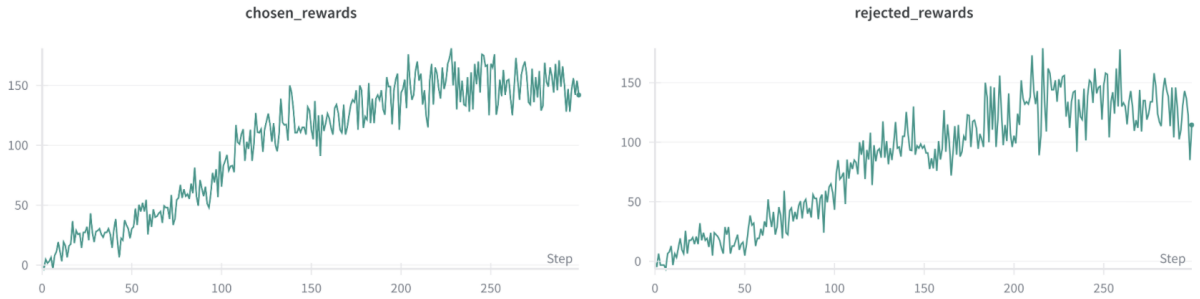

Figure 10: S-AA (Stage 2) training curves: winner (expert) vs. loser (policy rollout) implicit DPO rewards. The upward trend of both curves and narrowing gap indicate successful preference alignment on action tokens.

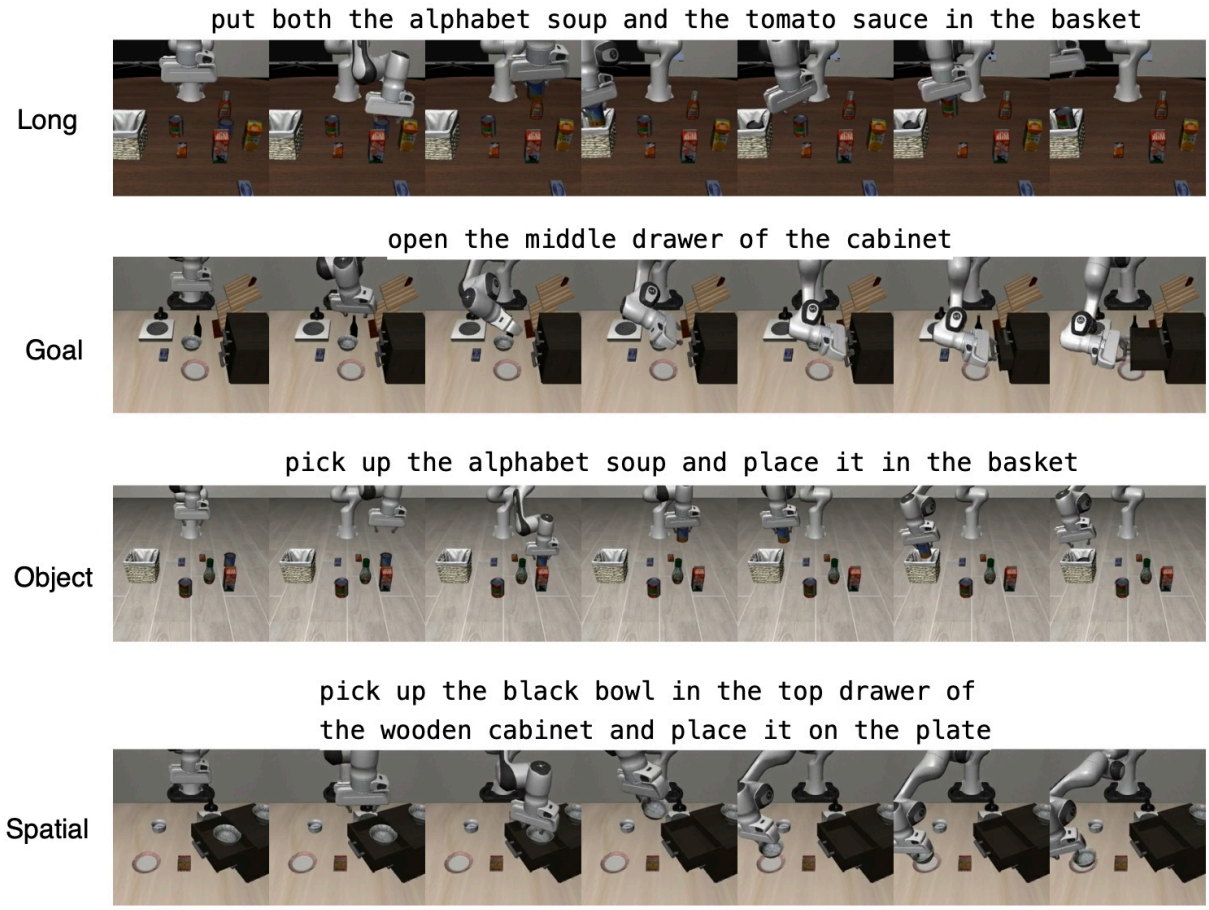

Figure 11: Visual observations for the four LIBERO evaluation task suites. We show episodes from 4 subtasks to demonstrate the diversity and complexity of the evaluation scenarios.