ACTWEAVE: EFFICIENT ACTION GENERATION FOR VISION-LANGUAGE-ACTION MODELS VIA HIERARCHICAL AUTOREGRESSION AND WAVEFRONT DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive vision-language-action (VLA) models have demonstrated strong robotic manipulation capabilities. However, fine-grained action tokenization can produce long sequences, making token-wise decoding through large vision-language models (VLMs) expensive. We introduce **ACTWEAVE**, a hierarchical autoregressive framework that shifts VLM computation from tokens to structured action patches. A deterministic discrete cosine transform (DCT) tokenizer retains all quantized coefficients in an action-dimension–frequency grid without a learned codebook. The VLM-based Thinker models inter-patch dependencies, while a lightweight Weaver performs **wavefront decoding**, predicting each anti-diagonal in parallel conditioned on earlier wavefronts. Direct VLM fine-tuning without robot-data pretraining yields competitive LIBERO and SimplerEnv performance. Controlled LIBERO comparisons show a 17.9× full-chunk speedup over flat global autoregression. Wavefront decoding reduces latency by 36.0% relative to raster local decoding at comparable success rates. VLABench reconstruction-and-replay experiments show that the tokenizer preserves control-relevant information under the evaluated distribution shift. These results demonstrate efficient autoregressive action generation without shortening the action-token sequence.

1 INTRODUCTION

Vision–language–action (VLA) models connect pretrained visual and language understanding with robotic control (Zitkovich et al., 2023). Existing approaches include continuous action generation through diffusion or flow matching (Ghosh et al., 2024; Black et al., 2025b), and discrete action generation through autoregression (Zitkovich et al., 2023; Kim et al., 2025b; Pertsch et al., 2025). Autoregressive modeling is particularly appealing because it builds on the sequence-modeling capabilities of pretrained vision–language models (VLMs). Discrete autoregressive action supervision can further accelerate policy adaptation and help preserve pretrained knowledge under appropriate training strategies (Driess et al., 2025). These advantages motivate improving the inference efficiency of autoregressive action generation.

Existing autoregressive VLAs discretize continuous actions into tokens and learn to predict them from observations and instructions (Zitkovich et al., 2023; Kim et al., 2025b). A central bottleneck arises when these tokens are generated sequentially through a large VLM: each additional token incurs another expensive model evaluation. Fine-grained action representations can produce long sequences, further increasing this cost. Existing methods address this challenge through learned action compression (Wang et al., 2025; Liu et al., 2026a; Lian et al., 2026) or compact analytical representations based on the discrete cosine transform (DCT) and splines (Pertsch et al., 2025; Zhou et al., 2025). Action compression must preserve control-relevant detail (Dong et al., 2026), and its reconstruction fidelity must remain reliable under distribution shift. Other approaches accelerate generation through block-wise or parallel decoding and lightweight action modules (Liu et al., 2026c; Song et al., 2025). Fully parallel prediction without generated-token conditioning removes sequential dependencies, whereas token-wise autoregression serializes every prediction. We therefore investigate

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

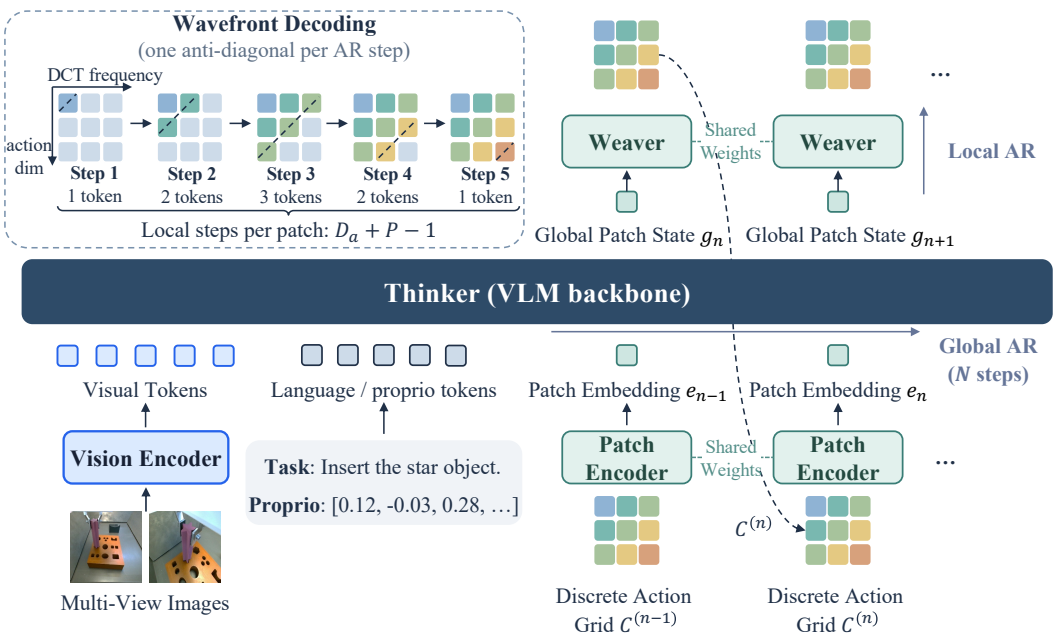


Figure 1: ACTWEAVE retains the full action grid while reducing sequential decoding depth. The Thinker advances once per patch; the Weaver generates all its tokens using wavefront decoding. A patch of length P has P retained DCT coefficients per action dimension. For D_a action dimensions, each grid contains $D_a P$ tokens, but local generation takes $D_a + P - 1$ sequential steps. AR denotes autoregression.

how the structure of a fine-grained action grid can guide dependency modeling and reduce expensive VLM evaluations without shortening the token sequence.

We introduce **ACTWEAVE**, a hierarchical autoregressive framework based on the insight that *the granularity of the action representation need not determine the granularity of VLM computation*. We partition an action chunk into short temporal patches and use a deterministic DCT tokenizer to represent each patch as an action-dimension–frequency grid, retaining all quantized coefficient positions without a learned codebook. Lower frequencies describe broad motion trends, while higher frequencies capture finer temporal variation. Our analysis shows that a compact predictor can exploit dependencies within these grids, supporting lightweight local action modeling. As illustrated in Figure 1, the VLM-based Thinker models dependencies across patches using multimodal observations and preceding patch embeddings. A lightweight Weaver generates each patch's action tokens conditioned on the resulting global state. Wavefront Decoding generates tokens along each anti-diagonal in parallel while conditioning on earlier wavefronts. The Patch Encoder then summarizes the completed grid for the next global prediction. By advancing the Thinker per patch and the Weaver per wavefront, ACTWEAVE reduces both expensive VLM evaluations and local sequential decoding steps while retaining the full quantized action grid.

Our main contributions are:

- **Hierarchical action generation.** We introduce ACTWEAVE, which separates global patch-level autoregression from lightweight local token generation, enabling efficient action generation without shortening the fine-grained token sequence.
- **Structure-aware Wavefront Decoding.** We develop Wavefront Decoding for action-dimension–frequency grids, reducing local sequential decoding by predicting each anti-diagonal in parallel while conditioning on earlier wavefronts.
- **Empirical validation.** ACTWEAVE achieves competitive performance on LIBERO (Liu et al., 2023) and SimplerEnv (Li et al., 2025) without robot-data pretraining. We also evaluate cross-dataset adaptation from Bridge to VLABench (Zhang et al., 2025). Controlled comparisons show a 17.9× full-chunk speedup over flat global autoregression. Wavefront decoding

further reduces latency by 36.0% relative to raster local decoding at comparable success rates. Reconstruction experiments demonstrate high action fidelity under the evaluated distribution shift.

2 ACTWEAVE

ACTWEAVE generates actions through hierarchical autoregression, separating dependencies across action patches from fine-grained structure within each patch. A pretrained VLM serves as the **Thinker** to model inter-patch dependencies, while a lightweight **Weaver** generates each patch using wavefront decoding.

2.1 PROBLEM FORMULATION

For a fixed observation time t, a VLA policy receives visual observations o_t, a language instruction ℓ, and proprioceptive state (proprio) s_t, and predicts an action chunk spanning the next H time steps, where H is the *prediction horizon*. Each action is a column vector $a_{t+i} \in \mathbb{R}^{D_a}$, where D_a is the action dimension and $i = 0, \ldots, H-1$. The action chunk stacks these vectors as rows:

$$A = [a_t, \ldots, a_{t+H-1}]^\top \in \mathbb{R}^{H \times D_a}. \tag{1}$$

We partition A into non-overlapping temporal patches of *patch length* P, yielding $N = \lceil H/P \rceil$ patches after padding the final patch if necessary. For $n \in \{1, \ldots, N\}$, the *continuous action patch* $A^{(n)} \in \mathbb{R}^{P \times D_a}$ is the n-th consecutive block of P rows of the padded chunk. Each patch is tokenized into a *discrete action grid*

$$C^{(n)} = [c_{d,f}^{(n)}] \in \{0, \ldots, K-1\}^{D_a \times P}. \tag{2}$$

Each cell $c_{d,f}^{(n)}$ is an action token. Here, K is the token vocabulary size, $d \in \{0, \ldots, D_a - 1\}$ indexes rows (action dimensions), and $f \in \{0, \ldots, P-1\}$ indexes columns (temporal DCT frequencies). Applying the DCT along the temporal axis yields P coefficients per action dimension. We transpose the quantized coefficient matrix to obtain this action-dimension–frequency layout. Retaining all quantized coefficient positions gives $D_a P$ tokens per patch.

For patch n, the Thinker computes a *global patch state* g_n conditioned on the multimodal context and preceding patch embeddings. The Weaver generates $C^{(n)}$ conditioned on g_n. The **Patch Encoder** summarizes the completed grid as a *patch embedding* e_n for the next global step. Figure 2 illustrates these representations and modules.

2.2 FIXED-GRID DCT ACTION TOKENIZATION

Our fixed-grid DCT tokenizer (FG-DCT) independently tokenizes each continuous action patch. Following the frequency-domain discretization in FAST (Pertsch et al., 2025), we normalize, transform, and quantize each patch, but omit byte-pair encoding and retain the full coefficient grid. We first apply benchmark-specific normalization:

$$\bar{A}^{(n)} = \mathrm{Normalize}\Big(A^{(n)}\Big) \in \mathbb{R}^{P \times D_a}. \tag{3}$$

Normalization statistics are computed from the training data and held fixed during policy training and inference. Appendix A.1 and Table 4 specify the normalization schemes and benchmark-specific settings.

Let $\bar{a}_{j,d}^{(n)}$ denote the entry of $\bar{A}^{(n)}$ at local time $j = 0, \ldots, P-1$ and action dimension $d = 0, \ldots, D_a - 1$. We apply an orthonormal DCT-II (Ahmed et al., 1974) independently along the temporal axis of each action dimension:

$$z_{f,d}^{(n)} = \alpha_f \sum_{j=0}^{P-1} \bar{a}_{j,d}^{(n)} \cos\left[\frac{\pi}{P}\left(j + \frac{1}{2}\right) f\right], \tag{4}$$

where $f = 0, \ldots, P-1$, with normalization factors $\alpha_0 = 1/\sqrt{P}$ and $\alpha_f = \sqrt{2/P}$ for $f > 0$. The coefficient matrix $Z^{(n)} = [z_{f,d}^{(n)}]$ has the same $P \times D_a$ shape as the normalized patch $\bar{A}^{(n)}$: the

transform changes the temporal basis, not the number of entries. We multiply these coefficients by a positive quantization scale γ and round to the nearest integers:

$$Q^{(n)} = \mathrm{round}\Big(\gamma Z^{(n)}\Big) \in \mathbb{Z}^{P \times D_a}. \tag{5}$$

Larger γ gives finer quantization. To form the discrete action grid, we transpose $Q^{(n)}$ and map each integer entry $q^{(n)}_{f,d}$ to a token ID, $c^{(n)}_{d,f} = \iota(q^{(n)}_{f,d})$. The fixed mapping ι clips coefficients to the supported integer range before applying an offset (Appendix A.1). All coefficient positions, including those quantized to zero, are retained, yielding a fixed-size action grid.

For each action dimension, the DCT provides a coarse-to-fine temporal organization: the zero-frequency coefficient is proportional to the patch mean, while progressively higher-frequency coefficients capture increasingly rapid temporal variations. This ordering allows wavefront decoding to condition higher-frequency tokens on previously generated lower-frequency tokens within each action dimension. Actions are reconstructed by decoding token IDs, dividing by γ, applying the inverse DCT, and denormalizing (Appendix A.1).

2.3 Global Patch Autoregression

To model dependencies across patches, the Thinker processes one embedding per completed patch together with the multimodal context. For each discrete action grid, the Patch Encoder adds learned action-dimension and frequency positional embeddings to frequency-specific token embeddings. A bidirectional transformer processes the flattened sequence with a learned pooling token; its output at that token is projected to the VLM hidden width and normalized with RMSNorm (Zhang & Sennrich, 2019):

$$e_n = E_{\mathrm{patch}}(C^{(n)}) \in \mathbb{R}^{d_{\mathrm{VLM}}}, \tag{6}$$

where E_{patch} denotes the Patch Encoder and d_{VLM} is the VLM hidden width. The patch embedding e_n summarizes the entire discrete action grid as a single input embedding to the Thinker.

Conditioned on the multimodal context $x_t = (o_t, \ell, s_t)$, the Thinker maps shifted patch embeddings to global patch states $g_n \in \mathbb{R}^{d_{\mathrm{VLM}}}$:

$$(g_1, \ldots, g_N) = \mathrm{Thinker}(x_t, [e_{\mathrm{bos}}, e_1, \ldots, e_{N-1}]), \tag{7}$$

where e_{bos} is a learned beginning-of-action embedding. The global patch state g_n is the VLM hidden state at the position of e_{bos} for the first patch and e_{n-1} for subsequent patches. Causal masking ensures that g_n depends only on x_t, e_{bos}, and preceding patch embeddings $e_1, \ldots, e_{n-1}$. During training, ground-truth patch embeddings are shifted by one position, allowing all global patch states to be computed in a single causally masked forward pass.

At inference, the Weaver generates the current patch conditioned on g_n. For $n < N$, the Patch Encoder embeds the generated grid, and the Thinker uses this embedding to compute g_{n+1}. Each chunk requires N sequential VLM evaluations: one context prefill and $N - 1$ cached global decoding steps.

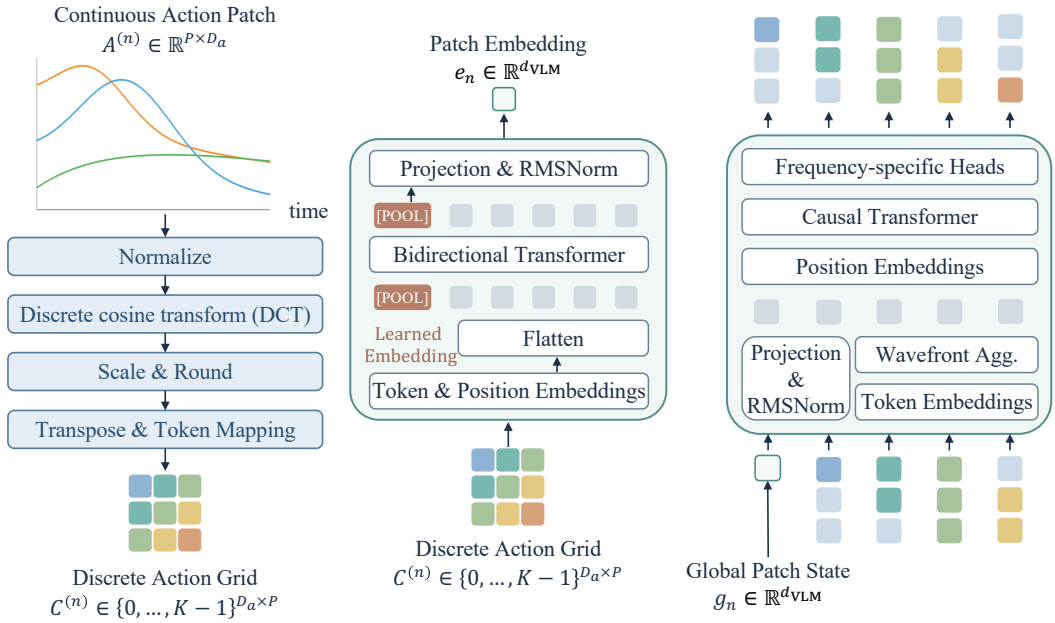
2.4 Local Wavefront Decoding

Patch-level prediction reduces sequential VLM decoding steps, but a token-wise local decoder would still require $D_a P$ sequential decoding steps per discrete action grid. Wavefront decoding predicts tokens on each anti-diagonal in parallel while retaining autoregressive dependencies across anti-diagonals. This schedule is related to delayed audio-token generation (Copet et al., 2023; Xiaomi LLM-Core Team, 2025). For a cell at action dimension d and frequency f, the wavefront index is $m = d + f$. The cells generated at step m form the *wavefront*

$$\Gamma_m = \{(d, f) \mid 0 \le d < D_a,\ 0 \le f < P,\ d + f = m\}, \qquad m = 0, \ldots, D_a + P - 2. \tag{8}$$

The Weaver predicts all cells in Γ_m in parallel, conditioned on the global patch state g_n and tokens from earlier wavefronts, denoted by $C^{(n)}_{<m}$. Its probability distribution factorizes as

$$p(C^{(n)} \mid g_n) = \prod_{m=0}^{D_a+P-2} \prod_{(d,f)\in\Gamma_m} p\Big(c^{(n)}_{d,f} \mid g_n, C^{(n)}_{<m}\Big). \tag{9}$$



(a) FG-DCT tokenization (b) Patch Encoder (c) Weaver

Figure 2: Action tokenization and patch-level modules. (a) FG-DCT maps a $P \times D_a$ continuous action patch to a $D_a \times P$ discrete action grid. (b) The Patch Encoder summarizes the grid as a patch embedding e_n. (c) The Weaver conditions on the global patch state g_n and preceding wavefronts. Wavefront aggregation sums frequency-specific token embeddings; sequence positional embeddings are added before the causal transformer, followed by frequency-specific output heads.

Cells on the same wavefront are conditionally independent under this factorization. Each token is conditioned on all earlier wavefronts, including lower-frequency coefficients of the same action dimension.

Wavefront aggregation and prediction. Let d_{loc} be the Weaver hidden width and $B_f : \{0, \ldots, K-1\} \to \mathbb{R}^{d_{\text{loc}}}$ its learned token embedding function for frequency f. Wavefront aggregation in Figure 2 sums the embeddings of tokens in each completed wavefront:

$$h_m^{(n)} = \sum_{(d,f) \in \Gamma_m} B_f(c_{d,f}^{(n)}), \qquad S_m^{(n)} = [\text{Proj}(g_n), h_0^{(n)}, \ldots, h_{m-1}^{(n)}]. \tag{10}$$

Here, $h_m^{(n)} \in \mathbb{R}^{d_{\text{loc}}}$ is a wavefront representation, and $S_m^{(n)}$ is the input sequence for predicting wavefront m. The representation $h_m^{(n)}$ is used only after wavefront m has been generated, when predicting subsequent wavefronts. The first input token, $\text{Proj}(g_n)$, is a linear projection followed by RMSNorm; for $m = 0$, it is the only input token. Learned sequence positional embeddings are added before the causal transformer.

Let $u_m^{(n)} \in \mathbb{R}^{d_{\text{loc}}}$ denote the Weaver's last-layer hidden state at the final input position of $S_m^{(n)}$. Frequency-specific linear token output heads produce

$$p(c_{d,f}^{(n)} = k \mid g_n, C_{<m}^{(n)}) = \left[\text{softmax}(W_f u_m^{(n)} + b_f)\right]_k, \qquad (d,f) \in \Gamma_m, \tag{11}$$

where $W_f \in \mathbb{R}^{K \times d_{\text{loc}}}$ and $b_f \in \mathbb{R}^K$ are learned parameters, and $k \in \{0, \ldots, K-1\}$ is a token ID. At wavefront m, each valid frequency f corresponds to a single action dimension $d = m - f$. Frequency-specific heads therefore predict all tokens in the wavefront in parallel from the shared hidden state $u_m^{(n)}$. During training, ground-truth wavefront representations are shifted by one position. Causal masking allows all wavefront predictions to be computed in parallel without access to their targets.

Local generation requires $D_a + P - 1$ sequential decoding steps per grid. A full chunk requires N global and $N(D_a + P - 1)$ local sequential decoding steps. These counts describe sequential depth; actual latency also depends on the cost of each model evaluation.

2.5 Learning and Inference

We jointly optimize the Thinker, Patch Encoder, and Weaver using teacher forcing, while keeping FG-DCT fixed. For each action chunk, we minimize a weighted negative log-likelihood:

$$\mathcal{L}_{\text{act}} = -\sum_{n=1}^{N} \lambda_n \sum_{m=0}^{D_a+P-2} \sum_{(d,f)\in\Gamma_m} \omega_f \log p\Big(c_{d,f}^{(n)} \mid g_n, C_{<m}^{(n)}\Big). \tag{12}$$

Here, $\lambda_n \geq 0$ and $\omega_f \geq 0$ are fixed loss weights for patch n and DCT frequency f, respectively. Setting both to 1 recovers the unweighted objective. Table 3 lists default weights; study-specific overrides are given in the corresponding experimental protocols.

At inference, the policy alternates between computing global patch states and generating action grids through local wavefront decoding, then reconstructs the continuous action chunk. We use receding-horizon control, executing actions for E control steps before replanning from updated observations, where E is the execution horizon and $1 \leq E \leq H$. Algorithms 1 and 2 in Appendix A.2 summarize training and closed-loop inference, respectively. Token-selection and action-postprocessing settings are specified in Appendix B.

3 Experiments

We evaluate whether ActWeave can reduce the cost of generating fine-grained robot actions while maintaining effective closed-loop control. Our experiments address 3 questions:
Q1: Policy performance. How well does ActWeave perform under direct VLM fine-tuning and cross-dataset adaptation?
Q2: Generation design. How do hierarchical autoregression, wavefront decoding, and patch granularity affect policy success and inference latency?
Q3: Action representation. Can a lightweight predictor exploit local dependencies in action grids, and does FG-DCT preserve reconstruction fidelity under distribution shift?

3.1 Experimental Setup

Benchmarks and baselines. We evaluate 4 LIBERO suites (Liu et al., 2023), 4 SimplerEnv WidowX tasks (Li et al., 2025), and 10 in-distribution VLABench primitive tasks (Zhang et al., 2025). Table 1 compares source-reported discrete-AR (OpenVLA, π_0-FAST), diffusion/flow-matching (Diffusion Policy, π_0), and continuous-regression (OpenVLA-OFT) policies. Table 2 reports controlled architectural comparisons. Appendix B.2 details protocols and sources.

Metrics. We report closed-loop success rate (SR) and p50/p95 latency per full chunk and amortized per executed action. Representation analysis uses action mean absolute error (MAE) and reconstructed-action replay success (Replay SR).

Implementation details. All parameters, including the vision encoder, are fine-tuned. LIBERO and SimplerEnv initialize from Qwen3-VL-4B-Instruct (Bai et al., 2025); VLABench adapts a Bridge-trained policy. With proprio inputs, the policy predicts the next 10 control steps as 2 length-5 patches of end-effector pose deltas and absolute gripper commands. Appendix B details training and deployment.

3.2 Policy Performance

Policy learning without robot-data pretraining. Direct VLM fine-tuning yields competitive SRs of 95.9% on LIBERO and 84.38% on SimplerEnv (Table 1), supporting effective patch-level control without robot-data pretraining. Appendix C.1 provides detailed results; Figure 5 visualizes representative rollouts.

Transfer to VLABench. Adapting a Bridge-trained policy for 10k steps yields 42.8% SR on 10 in-distribution VLABench primitive tasks without architectural changes. This complements direct VLM fine-tuning with cross-dataset policy adaptation. Appendix C.3 provides task-level results and evaluation details.

Table 1: Policy SR (%, ↑). Avg.: source-reported average; bold/underline: best/second-best distinct values among displayed rows (including ties); –: unavailable or unverified.

Model	LIBERO (SR ↑)					SimplerEnv (SR ↑)				
	Spatial	Object	Goal	Long	Avg.	Spoon	Carrot	Block	Eggplant	Avg.
Diffusion Policy (Chi et al., 2023)	78.3	92.5	68.3	50.5	72.4	–	–	–	–	–
Octo-Base (Ghosh et al., 2024)	78.9	85.7	84.6	51.1	75.1	12.5	8.3	0.0	43.1	16.0
SpatialVLA (Qu et al., 2025)	88.2	89.9	78.6	55.5	78.1	16.7	25.0	29.2	**100.0**	42.7
π_0 (Black et al., 2025b)	96.8	98.8	95.8	85.2	94.2	66.7	58.3	58.3	88.3	66.7
$\pi_{0.5}$ (Black et al., 2025a)	**98.8**	98.2	**98.0**	92.4	**96.8**	–	–	–	–	–
OpenVLA-OFT (Kim et al., 2025a)	97.7	98.0	96.1	**95.3**	**96.8**	12.5	4.2	8.3	0.0	6.3
UniVLA (Bu et al., 2025)	96.5	96.8	95.6	92.0	95.2	54.2	66.7	50.0	4.2	43.8
OpenVLA (Kim et al., 2025b)	84.7	88.4	79.2	53.7	76.5	32.0	30.0	18.0	38.0	29.5
PaliGemma–Naive (Liu et al., 2026c)	55.8	64.8	64.4	31.2	54.1	66.7	29.2	12.5	54.2	40.9
MiniVLA (Belkhale & Sadigh, 2024)	–	–	–	–	–	68.0	44.0	70.0	14.0	49.0
VQ-VLA (Wang et al., 2025)	–	–	75.2	60.0	–	12.5	8.0	6.0	0.0	6.3
π_0-FAST-R (Pertsch et al., 2025)	96.4	96.8	88.6	60.2	85.5	29.1	21.9	10.8	66.6	32.1
π_0-FAST-D (Pertsch et al., 2025)	96.6	97.2	96.0	86.8	94.2	77.5	**88.3**	68.3	71.7	76.5
ActionPiece (Lian et al., 2026)	96.2	99.0	93.4	90.6	94.8	**83.3**	54.2	58.3	91.7	71.9
MemoryVLA (Shi et al., 2026)	98.4	98.4	96.4	93.4	96.7	75.0	75.0	37.5	**100.0**	71.9
SP-VLA (Li et al., 2026)	84.4	85.6	75.4	54.2	74.9	70.83	54.17	29.17	75.00	57.29
CoWVLA (Yang et al., 2026)	97.2	97.8	94.6	92.8	95.6	79.2	66.7	62.5	95.8	76.0
PD-VLA (Song et al., 2025)	95.5	96.7	94.9	91.7	94.7	–	–	–	–	–
ACTWEAVE (ours)	97.4	**99.8**	95.6	90.8	95.9	79.2	70.8	**91.7**	95.8	**84.38**

Sources and evaluation protocols: Appendix B.2.

3.3 GENERATION EFFICIENCY AND ARCHITECTURAL DESIGN

Inference efficiency. ACTWEAVE achieves a 17.9× full-chunk speedup over Flat Global AR with higher observed SR (Table 2). Both generate 70 discrete action tokens per chunk, but ACTWEAVE uses only 2 global evaluations, delegating local prediction to the Weaver. This changes computational granularity without shortening the representation. Appendix D specifies the controlled protocol; Appendices C.1 and C.2 separately report deployment costs and schedule-only measurements.

Table 2: LIBERO architecture ablations at 10k training steps. Succ.: successes/trials; SR: success rate (%); ΔSR: row minus Ref. (pp); p50: full-chunk latency (ms); G/L: global/local steps. Bold/underline: best/second-best SR and p50.

ID	Variant	Ref.	Succ.	SR ↑	ΔSR ↑	p50 ↓	G/L ↓
A	**ACTWEAVE (full)**	–	149/160	93.13	–	121.50	2/22
	(a) Patch Encoder architecture						
B	w/ Projection Encoder	A	139/160	86.88	−6.25	120.92	2/22
	(b) Local decoding schedule						
C	w/ Raster Local AR	A	150/160	**93.75**	+0.63	189.84	2/70
D	w/ Parallel Local Decoding	A	144/160	90.00	−3.13	86.22	2/2
	(c) Weaver architecture						
E	w/ Linear Local Head	D	140/160	87.50	−2.50	**84.25**	2/2
	(d) Global token-wise reference						
F	Flat Global AR	A	140/160	87.50	−5.63	2175.66	70/0

Architectural ablations. We compare 6 variants with the same pretrained VLM initialization, 10k training steps, and 160 LIBERO episodes (Table 2). Wavefront decoding reduces latency by 36.0% relative to raster decoding at comparable SR: serializing every local token appears unnecessary under this budget. Fully parallel decoding is faster but yields lower observed SR, supporting conditioning on earlier wavefronts. Replacing the Patch Encoder with a projection lowers observed SR at similar latency, suggesting that richer patch representations benefit global prediction. Under parallel decoding, the Transformer-based Weaver outperforms linear heads in observed SR. Neither conditions on generated tokens, supporting local modeling beyond autoregressive conditioning. These comparisons support patch-level VLM computation with structured local generation; Appendix D reports uncertainty.

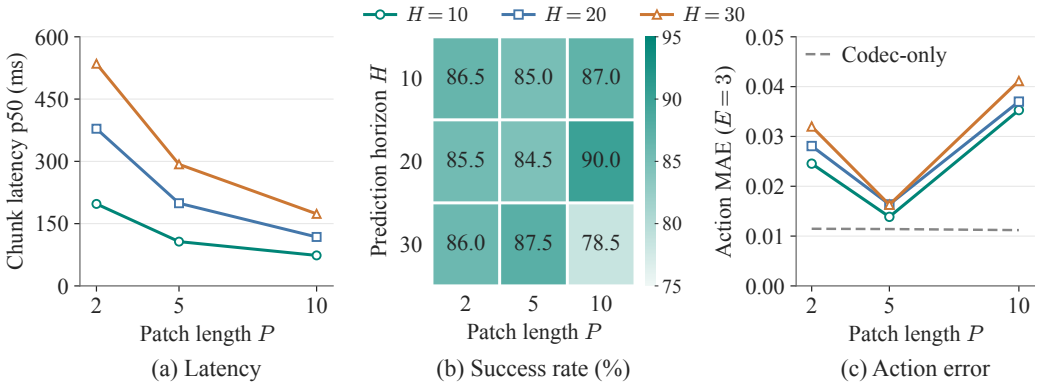


Figure 3: Effect of patch length P and prediction horizon H. (a) Full-chunk p50 latency decreases with patch length. (b) Closed-loop SR depends on both patch length and horizon. (c) Predicted-action and codec-only MAE over the first 3 actions. Results use 1k adaptation steps and execution horizon $E = 3$; Appendix E reports protocols and uncertainty.

Effect of patch length and prediction horizon. We vary $H \in \{10, 20, 30\}$ and $P \in \{2, 5, 10\}$ at $E = 3$, with 1k adaptation steps and 200 paired episodes per configuration (Figure 3). At fixed H, larger patches reduce sequential decoding steps and latency, but SR varies non-monotonically. Increasing H at fixed P raises cost without consistent SR gains; fixed E means only the prediction window grows. Prediction MAE is lowest at $P = 5$, while codec-only error changes little, implicating prediction rather than quantization. Lower MAE does not consistently improve SR, making closed-loop evaluation essential for choosing patch granularity. Appendix E reports full results and uncertainty.

3.4 Action Representation Analysis

Local dependencies in action tokens. Higher-frequency motion tokens generally have lower marginal entropy (Figure 4a). Conditioning a separately trained compact Transformer on earlier wavefronts reduces held-out cross-entropy by 12.9% over a position-wise prior without token context (Figure 4b). Earlier wavefronts thus provide information beyond position statistics, supporting lightweight modeling of local action-grid dependencies. Appendix F details the probes.

Action reconstruction under distribution shift. On 128 OOD VLABench trajectories, FG-DCT achieves the lowest normalized MAE and highest Replay SR (96.09%) among 8 tokenizer–adapter pipelines (Figure 4c), preserving control-relevant information under this shift. Combined with Section 3.3, this supports retaining action detail while reducing generation cost. Replay evaluates representation fidelity, not learned-policy generalization. Appendix C.4 defines the shift, subset selection, and adapters.

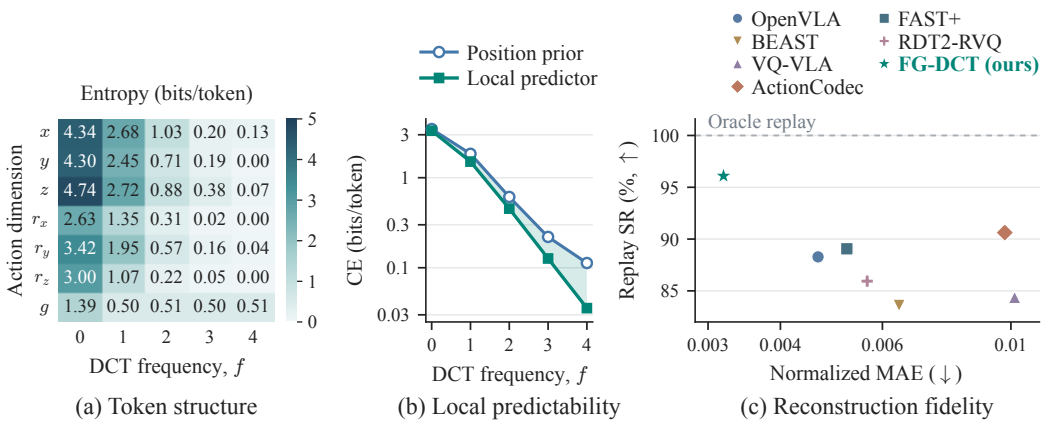


Figure 4: Action-token structure and reconstruction fidelity. (a) LIBERO token entropy. (b) Held-out LIBERO cross-entropy (CE; log y-axis); shading: CE gap, not a confidence interval. (c) VLABench reconstruction with action-space adapters (log x-axis); dashed line: oracle replay. OpenVLA denotes binning. X-Tokenizer (MAE 0.0983; SR 7.03%) lies outside the plot; Appendix C.4 reports all methods.

4 Related Work

Vision–language–action policies. VLA policies adapt pretrained representations through discrete autoregression (Zitkovich et al., 2023; Kim et al., 2025b) or continuous regression, diffusion, and flow matching (Kim et al., 2025a; Ghosh et al., 2024; Black et al., 2025b). Representation does not determine the decoding schedule: AR-VLA generates continuous actions autoregressively (Hu et al., 2026). ACT and Diffusion Policy predict action chunks (Zhao et al., 2023a; Chi et al., 2023); ActWeave partitions chunks into temporal patches for discrete autoregression.

Discrete action representations. Action tokenizers range from per-dimension binning (Zitkovich et al., 2023; Kim et al., 2025b) to learned quantized representations (Wang et al., 2025; Liu et al., 2026c). Learned tokenizers also incorporate physical constraints, ordering objectives (Lian et al., 2026; Liu et al., 2026a), and cross-modal alignment (Kang et al., 2026; Lyu et al., 2026). ActionCodec studies their effects on VLA optimization beyond reconstruction fidelity (Dong et al., 2026). Analytical alternatives include FAST's DCT quantization with learned BPE compression (Pertsch et al., 2025) and BEAST's quantized B-splines (Zhou et al., 2025). FG-DCT omits BPE and retains all coefficient positions in an action-dimension–frequency grid, exposing its structure to the generator rather than learning latent codes.

Efficient action generation. OpenVLA-OFT uses parallel continuous prediction (Kim et al., 2025a). Discrete generators use parallel fixed-point iterations (Song et al., 2025) or block-wise autoregression with lightweight action experts (Liu et al., 2026c). ActWeave assigns VLM computation to patches and lightweight wavefront decoding within patches, preserving inter-wavefront conditioning without shortening token grids.

5 Conclusion

ActWeave combines patch-level VLM autoregression with lightweight wavefront decoding to reduce sequential generation depth while retaining fine-grained action information. Direct VLM fine-tuning yields competitive LIBERO and SimplerEnv performance, complemented by Bridge-to-VLABench transfer. Architectural comparisons, local-predictability analysis, and reconstructed-action replay support this division of global and local modeling. These findings highlight computation allocation, rather than token compression alone, as an effective route to efficient generation of fine-grained robot actions.

AI USE STATEMENT

Generative AI tools were used to assist with language editing and polishing of the manuscript, and to retrieve and identify related literature. The authors take full responsibility for the final manuscript, including the accuracy of its content and references.

ETHICS STATEMENT

The experiments use existing benchmark data and simulated robot environments. We report aggregate task outcomes and do not use personally identifiable data. Applying these policies to physical robots would require appropriate safety measures, including workspace limits, emergency stops, collision monitoring, and human supervision.

REPRODUCIBILITY STATEMENT

Appendices A–B specify algorithms and training/evaluation settings; Appendix C.1 gives policy counts, deployment latency, and memory usage, and Appendix C.2 documents schedule timing. Appendices D–F provide ablation budgets, paired evaluation protocols, probe splits, and uncertainty estimates.

REFERENCES

N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, 1974. doi: 10.1109/T-C.1974.223784.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-VL technical report. arXiv preprint arXiv:2511.21631, 2025. URL https://arxiv.org/abs/2511.21631.

Suneel Belkhale and Dorsa Sadigh. MiniVLA: A better VLA with a smaller footprint, 2024. URL https://ai.stanford.edu/blog/minivla/. Stanford AI Lab blog and software release.

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 *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40, 2025a. URL https://proceedings.mlr.press/v305/black25a.html.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision–language–action flow model for general robot control. In *Robotics: Science and Systems*, 2025b. doi: 10.15607/RSS.2025.XXI.010. URL https://www.roboticsproceedings.org/rss21/p010.html.

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 *Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.014. URL https://arxiv.org/abs/2505.06111.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin C. M. Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://arxiv.org/abs/2303.04137.

Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild

robots. In *Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.045. URL https://www.roboticsproceedings.org/rss20/p045.html.

Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation. In *Advances in Neural Information Processing Systems*, volume 36, pp. 47704–47720, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/94b472a1842cd7c56dcb125fb2765fbd-Abstract-Conference.html.

Zibin Dong, Yicheng Liu, Shiduo Zhang, Baijun Ye, Yifu Yuan, Fei Ni, Jingjing Gong, Xipeng Qiu, Hang Zhao, Yinchuan Li, and Jianye Hao. ActionCodec: What makes for good action tokenizers. arXiv preprint arXiv:2602.15397, 2026. URL https://arxiv.org/abs/2602.15397.

Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z. Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, and Sergey Levine. Knowledge insulating vision–language–action models: Train fast, run fast, generalize better. In *Advances in Neural Information Processing Systems*, volume 38, pp. 102867–102888, 2025. doi: 10.52202/085713-3439. URL https://papers.nips.cc/paper_files/paper/2025/hash/94e936034d12bcd04834ec2773f02aff-Abstract-Conference.html.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R. Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.090. URL https://arxiv.org/abs/2405.12213.

Yutong Hu, Jan-Nico Zaech, Nikolay Nikolov, Yuanqi Yao, Sombit Dey, Giuliano Albanese, Renaud Detry, Luc Van Gool, and Danda Pani Paudel. AR-VLA: Autoregressive action expert for vision–language–action models. In *Robotics: Science and Systems*, 2026. doi: 10.15607/RSS.2026.XXII.085. URL https://www.roboticsproceedings.org/rss22/p085.html.

Miracle Kang, Lights Shi, Lucy Liang, Roy Gan, Dongxiu Liu, Pushi Zhang, Sylas Chen, Shawn Qin, Yinan Zheng, Jinliang Zheng, Hao Wang, Xianyuan Zhan, and Hang Su. X-Tokenizer: A multimodal action tokenizer for vision–language–action pretraining. arXiv preprint arXiv:2606.14752, 2026. URL https://arxiv.org/abs/2606.14752.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision–language–action models: Optimizing speed and success. In *Robotics: Science and Systems*, 2025a. doi: 10.15607/RSS.2025.XXI.017. URL https://arxiv.org/abs/2502.19645.

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 *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713, 2025b. URL https://proceedings.mlr.press/v270/kim25c.html.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Oier Mees, Karl Pertsch, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3705–3728, 2025. URL https://proceedings.mlr.press/v270/li25c.html.

Ye Li, Yuan Meng, Zewen Sun, Kangye Ji, Chen Tang, Jiajun Fan, Xinzhu Ma, Shu-Tao Xia, Zhi Wang, and Wenwu Zhu. SP-VLA: A joint model scheduling and token pruning approach for VLA model acceleration. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RwdGIIjPlC.

Shijie Lian, Bin Yu, Zhaolong Shen, Xiaopeng Lin, Yichao Du, Zhirui Zhang, Laurence T. Yang, and Kai Chen. ActionPiece: Rethinking action tokenization for autoregressive vision–language–action models. arXiv preprint arXiv:2609.18487, 2026. URL https://arxiv.org/abs/2609.18487.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems: Datasets and Benchmarks Track*, volume 36, pp. 44776–44791, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

Chaoqi Liu, Xiaoshen Han, Jiawei Gao, Yue Zhao, Haonan Chen, and Yilun Du. OAT: Ordered action tokenization. In *Robotics: Science and Systems*, 2026a. doi: 10.15607/RSS.2026.XXII.075. URL https://www.roboticsproceedings.org/rss22/p075.html.

Songming Liu, Bangguo Li, Kai Ma, Lingxuan Wu, Hengkai Tan, Xiao Ouyang, Hang Su, and Jun Zhu. RDT2: Exploring the scaling limit of UMI data towards zero-shot cross-embodiment generalization. In *International Conference on Machine Learning*, 2026b. URL https://icml.cc/virtual/2026/poster/65782.

Yicheng Liu, Shiduo Zhang, Zibin Dong, Baijun Ye, Tianyuan Yuan, Xiaopeng Yu, Linqi Yin, Chenhao Lu, Junhao Shi, Luca Jiang-Tao Yu, Liangtao Zheng, Jingjing Gong, Tao Jiang, Xipeng Qiu, and Hang Zhao. FASTer: Toward powerful and efficient autoregressive vision–language–action models with learnable action tokenizer and block-wise decoding. In *International Conference on Learning Representations*, 2026c. URL https://openreview.net/forum?id=k6nTUFoqeT.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Huaihai Lyu, Chaofan Chen, Yuheng Ji, Xiansheng Chen, Pengwei Wang, Shanghang Zhang, and Changsheng Xu. LAST: Bridging vision–language and action manifolds via Gromov–Wasserstein alignment. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2606.11221.

OpenVLA-OFT Authors. OpenVLA-OFT: LIBERO training and evaluation. Official repository documentation, 2025. URL https://github.com/moojink/openvla-oft/blob/213c5ff8d6e9673885d7e4dd6f2ce73b5d8a45db/LIBERO.md. Accessed September 22, 2026.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision–language–action models. In *Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.012. URL https://arxiv.org/abs/2501.09747.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Jiayuan Gu, Zhigang Wang, Yan Ding, Bin Zhao, Dong Wang, and Xuelong Li. SpatialVLA: Exploring spatial representations for visual–language–action models. In *Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.011. URL https://arxiv.org/abs/2501.15830.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. MemoryVLA: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2508.19236v2.

Wenxuan Song, Jiayi Chen, Pengxiang Ding, Han Zhao, Wei Zhao, Zhide Zhong, Zongyuan Ge, Zhijun Li, Donglin Wang, Jun Ma, Lujia Wang, and Haoang Li. PD-VLA: Accelerating vision-language-action model integrated with action chunking via parallel decoding. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2025. URL https://arxiv.org/abs/2503.02310v2. LIBERO results from arXiv version 2, February 2026.

Homer Rich Walke, Kevin Black, Tony Z. Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, Abraham Lee, Kuan Fang, Chelsea Finn, and Sergey Levine. BridgeData V2: A dataset for robot learning at scale. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 1723–1736, 2023. URL https://proceedings.mlr.press/v229/walke23a.html.

Yating Wang, Haoyi Zhu, Mingyu Liu, Jiange Yang, Hao-Shu Fang, and Tong He. VQ-VLA: Improving vision–language–action models via scaling vector-quantized action tokenizers. In *International Conference on Computer Vision*, pp. 11089–11099, 2025. URL https://arxiv.org/abs/2507.01016.

Xiaomi LLM-Core Team. MiMo-Audio: Audio language models are few-shot learners. arXiv preprint arXiv:2512.23808, 2025. URL https://arxiv.org/abs/2512.23808.

Fuxiang Yang, Donglin Di, Lulu Tang, Xuancheng Zhang, Lei Fan, Hao Li, Wei Chen, Tonghua Su, and Baorui Ma. Chain of world: World model thinking in latent motion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6675–6684, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Yang_Chain_of_World_World_Model_Thinking_in_Latent_Motion_CVPR_2026_paper.html.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/1e8a19426224ca89e83cef47f1e7f53b-Abstract.html.

Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, and Xipeng Qiu. VLABench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. In *International Conference on Computer Vision*, pp. 11142–11152, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Zhang_VLABench_A_Large-Scale_Benchmark_for_Language-Conditioned_Robotics_Manipulation_with_Long-Horizon_ICCV_2025_paper.html.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems*, 2023a. doi: 10.15607/RSS.2023.XIX.016. URL https://www.roboticsproceedings.org/rss19/p016.html.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. PyTorch FSDP: Experiences on scaling fully sharded data parallel. arXiv preprint arXiv:2304.11277, 2023b. URL https://arxiv.org/abs/2304.11277.

Hongyi Zhou, Weiran Liao, Xi Huang, Yucheng Tang, Fabian Otto, Xiaogang Jia, Xinkai Jiang, Simon Hilber, Ge Li, Qian Wang, Ömer Yağmurlu, Nils Blank, Moritz Reuss, and Rudolf Lioutikov. BEAST: Efficient tokenization of B-Splines encoded action sequences for imitation learning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 172934–172959, 2025. doi: 10.52202/085713-5755. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/fc75cfcec16170f2b54d00283c739928-Abstract-Conference.html.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. RT-2: Vision–language–action models transfer web knowledge to robotic control. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183, 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

A Representation and Decoding Details

A.1 Action Normalization, Quantization, and Reconstruction

Normalization. Bound-based normalization uses fixed lower and upper bounds $b_d^- < b_d^+$:

$$\bar{a}_{j,d}^{(n)} = \mathrm{clip}\left(2\frac{a_{j,d}^{(n)} - b_d^-}{b_d^+ - b_d^-} - 1, -1, 1\right). \tag{13}$$

Here, $\mathrm{clip}(u, l, r) = \min(\max(u, l), r)$. Bounds are training-set minima/maxima for LIBERO and the 1st/99th percentiles for Bridge. VLABench uses $\bar{a}_{j,d}^{(n)} = (a_{j,d}^{(n)} - \mu_d)/\sigma_d$, with training-set means μ_d and positive standard deviations σ_d for pose components, and $\mu_d = \sigma_d = 0.5$ for the gripper (Table 4).

Token mapping and reconstruction. The vocabulary covers K integers starting at $q_{\min}$. Encoding uses $\iota(q) = \mathrm{clip}(q - q_{\min}, 0, K-1)$. Generated tokens $\widehat{c}_{d,f}^{(n)}$ recover the coefficient matrix $\widehat{Q}^{(n)}$ and normalized action patch $\widehat{\overline{A}}^{(n)}$ as

$$\widehat{q}_{f,d}^{(n)} = \widehat{c}_{d,f}^{(n)} + q_{\min}, \qquad \widehat{\overline{A}}^{(n)} = \mathrm{IDCT}_{\mathrm{time}}\left(\frac{\widehat{Q}^{(n)}}{\gamma}\right), \tag{14}$$

where $\mathrm{IDCT}_{\mathrm{time}}$ is the inverse orthonormal temporal DCT and hats denote reconstructed quantities. The bound-based and mean/std inverses are, respectively,

$$\widehat{a}_{j,d}^{(n)} = \frac{\widehat{\bar{a}}_{j,d}^{(n)} + 1}{2}(b_d^+ - b_d^-) + b_d^-, \qquad \widehat{a}_{j,d}^{(n)} = \sigma_d \widehat{\bar{a}}_{j,d}^{(n)} + \mu_d. \tag{15}$$

For actions normalized to $[-1, 1]$, orthonormality bounds coefficient magnitudes by $\sqrt{P}$. Rounding and any normalization or vocabulary clipping are irreversible.

A.2 Training and Inference Algorithms

Algorithms 1–2 follow Section 2, with $N = \lceil H/P \rceil$ patches and $M = D_a + P - 1$ wavefronts per patch. $\mathcal{K}$ denotes the within-chunk Thinker key–value cache. Cells in the same wavefront are conditionally independent given the global state and earlier wavefronts (Equation 9); this modeling assumption enables parallel prediction.

Training. Teacher forcing batches global positions and local wavefronts. One-step-shifted inputs and causal masks prevent target leakage. $\mathrm{DropHist}$ independently zeros global and local history embeddings during training, without changing targets. $\mathrm{WeaverTF}$ follows Equation 10, including sequence positional embeddings; θ contains all trainable parameters.

Algorithm 1 ActWeave training with teacher forcing

Require: Dataset $\mathcal{D}$ of pairs of observations and action chunks; pretrained VLM; fixed FG-DCT tokenizer
Ensure: Trained parameters θ

```
Initialize VLM from pretrained weights; initialize Patch Encoder and Weaver
for each training step do
    Sample a minibatch B ⊂ D
    for all (x_t, A) ∈ B do                                   ▷ Batched in implementation
        Split A into N continuous action patches A^(1:N) of length P; pad if needed
        C^(1:N) ← FG-DCT(A^(1:N)) using fixed training-set statistics
        e_n ← DropHist(Patch Encoder(C^(n))), for n = 1, ..., N − 1
        (g_1, ..., g_N) ← Thinker(x_t, [e_bos, e_1, ..., e_{N−1}])
        π^(1:N) ← WeaverTF(g_{1:N}, C^(1:N); DropHist)
        Compute weighted token loss L_act(x_t, A) using Equation 12
    end for
    Average the weighted loss over B; backpropagate through all trainable modules
    Update θ with AdamW and the configured learning-rate schedule
end for
```

Inference. In Algorithm 2, $\mathrm{SelectToken}(\pi; \eta)$ selects a token from distribution π using decoding settings η (Table 4); history dropout is disabled. Token selection does not change the wavefront schedule.

Each chunk uses one context prefill, $N-1$ cached VLM steps, and $N(D_a+P-1)$ local evaluations.

Algorithm 2 ActWeave closed-loop inference with configurable token selection

Require: Trained policy; instruction ℓ; FG-DCT tokenizer; horizons H, E; patch length P; decoding configuration η
Ensure: Receding-horizon action execution

while the episode has not terminated **do**
  Observe images o_t and proprio s_t; set $x_t \leftarrow (o_t, \ell, s_t)$
  $(g_1, \mathcal{K}) \leftarrow \mathrm{ThinkerPrefill}(x_t, e_{\mathrm{bos}})$
  for $n = 1, \ldots, N$ **do**
    Initialize an empty discrete action grid $\widehat{C}^{(n)} \in \{0, \ldots, K-1\}^{D_a \times P}$
    for $m = 0, \ldots, M-1$ **do**
      $\{\pi_{d,f}\}_{(d,f)\in\Gamma_m} \leftarrow \mathrm{WeaverStep}(g_n, \widehat{C}^{(n)}_{<m})$
      $\widehat{c}^{(n)}_{d,f} \leftarrow \mathrm{SelectToken}(\pi_{d,f}; \eta)$, for all $(d,f) \in \Gamma_m$ **in parallel**
    end for
    if $n < N$ **then**
      $\widehat{e}_n \leftarrow \mathrm{Patch\,Encoder}(\widehat{C}^{(n)})$
      $(g_{n+1}, \mathcal{K}) \leftarrow \mathrm{ThinkerStep}(\widehat{e}_n, \mathcal{K})$
    end if
    Reconstruct $\widehat{A}^{(n)}$ from $\widehat{C}^{(n)}$ as in Appendix A.1
  end for
  $\widehat{A} \leftarrow \mathrm{Concat}_{\mathrm{time}}(\widehat{A}^{(1)}, \ldots, \widehat{A}^{(N)})[0:H]$ ▷ Discard padding
  Apply the configured action postprocessing to $\widehat{A}$, if any
  Execute the first E actions of $\widehat{A}$, stopping early if the episode terminates
  Advance t by the number of actions executed
end while

B Implementation and Evaluation Details

B.1 Training and Model Configurations

Tables 3 and 4 specify the main-policy settings; auxiliary studies report their overrides separately. LIBERO and Bridge start from Qwen3-VL-4B-Instruct (Bai et al., 2025); VLABench transfers the full Bridge-trained 10k policy. Training uses AdamW (Loshchilov & Hutter, 2019) and Fully Sharded Data Parallel (FSDP) (Zhao et al., 2023b). Action modules comprise the Patch Encoder, Weaver, and their embeddings and projections.

Table 3: Shared architecture and LIBERO/Bridge optimization defaults. VLABench overrides are given in Table 4 and the accompanying text.

LIBERO/Bridge optimization

Configuration	Value
Optimizer	AdamW
Adam betas	β_1=0.9, β_2=0.95
Weight decay	0.05
Global batch size	256
Precision	BF16
Parallelism	FSDP, 4 GPUs
Compute	H800 80 GB
LR schedule	5% warmup; cosine to 0.1×
Patch weights λ	[2, 1]
Frequency weights ω	[6, 4, 2, 1, 1]
Image augmentation	Enabled
Pretraining on robot data	None

Model and action modules

Configuration	Value
Base model	Qwen3-VL-4B-Instruct
Trainable modules	VLM (including vision encoder), action modules
Tokenizer	FG-DCT
Vocabulary size	2,048
Patch Encoder	2 transformer layers
Weaver	4 transformer layers
Hidden width / heads	512 / 8
FFN width	2,048
History-embedding dropout	0.1
Proprio input	7D → 256-bin text
Default H/P	10 / 5

B.2 Benchmark-Specific Training and Evaluation

Action representation. Following UMI (Chi et al., 2024), relative actions reference the observation-time proprio, whereas delta actions reference current proprio for the first action and the preceding

Table 4: Benchmark-specific training and evaluation configurations.

Configuration	LIBERO	SimplerEnv	VLABench
Evaluated scope	Spatial, Object, Goal, Long; 40 tasks	Visually matched WidowX; Spoon, Carrot, Block, Eggplant	In-distribution primitive subset; 10 tasks
Training data	1,693 converted episodes	BridgeData V2	`vlabench_primitive_ft_lerobot_video`
Initialization	Pretrained VLM	Pretrained VLM	Bridge-trained 10k policy
Target training steps	100k	30k	10k
Evaluated checkpoint	80k	30k + 500 + 250 adaptation steps	10k, EMA 0.99
Global batch size	256	256	32
Peak LR: action modules	2×10^{-4}	2×10^{-4}	10^{-4}
Peak LR: language	5×10^{-5}	5×10^{-5}	2×10^{-5}
Peak LR: vision	5×10^{-5}	5×10^{-5}	5×10^{-6}
Normalization	Min–max	Q01/Q99	Unified training-set mean/std
Quantization scale γ	10	10	10
Action representation	delta 6D pose + absolute 1D gripper	delta 6D pose + absolute 1D gripper	delta 6D pose + absolute 1D gripper
$H/P/E$; G/L	10/5/1; 2/22	10/5/2; 2/22	10/5/5; 2/22
Inference settings	Top-32 mean + temporal ensembling, k_{ens}=0.01	Confidence-aware top-32, T=0.7, τ=0.35	Posterior median, T=1; no temporal ensembling
Main evaluation	50 states/task; 2,000 total	120 episodes/task; 480 total	50 episodes/task; 500 total
Step budget / success	220/280/300/520	60/60/60/120 at 5 Hz; final-step flag	Official task limits; environment success
Reported metrics	SR	SR	SR

$H/P/E$: prediction horizon/patch length/execution horizon; G/L: global/local sequential steps; Q01/Q99: 1st/99th percentiles. Benchmarks: LIBERO (Liu et al., 2023), SimplerEnv (Li et al., 2025), and VLABench (Zhang et al., 2025). SimplerEnv uses BridgeData V2 (Walke et al., 2023); temporal ensembling follows ACT (Zhao et al., 2023a).

target thereafter; gripper commands are absolute. Normalization uses fixed training-set statistics. VLABench uses translation and wrapped Euler-angle deltas with shared per-dimension mean/std across chunk positions, gripper mean/std of 0.5, and no pose clipping. The separate OOD reconstruction study uses absolute targets and $\gamma = 50$ (Appendix C.4).

VLABench transfer recipe. The official 5,000-demonstration dataset reserves two demonstrations per instruction/task index for validation. Inputs comprise three full-field-of-view cameras (two external and one wrist), without augmentation; standardized Euler proprio is clipped at ± 3 standard deviations and quantized into 256 bins. Training uses four RTX PRO 6000 GPUs, FP32 master parameters and Adam states, and BF16 computation. It retains a 30k-step cosine schedule with 1k warmup and minimum LR ratio 0.1, but stops at 10k steps. Other settings follow Tables 3–4. Terminal padding repeats the last absolute target before delta conversion; normalization uses unpadded training anchors. Evaluation data are not added to training.

Token selection. Top-32 aggregation rounds the probability-weighted mean of the 32 most probable token IDs. SimplerEnv applies temperature T and selects argmax at frequencies $f = 0, 1$ when the maximum token probability reaches τ; otherwise it uses top-32 aggregation. LIBERO uses oldest-first temporal ensembling with coefficient $k_{\text{ens}} = 0.01$ and a 10-step window. VLABench selects the posterior median: the first ordered coefficient code whose cumulative probability reaches 0.5, uniformly across tasks.

Baseline provenance. Earlier baseline rows in Table 1 follow FASTer Table 1 (Liu et al., 2026c) in arXiv:2512.04952v2, including third-party SimplerEnv reproductions; OpenVLA-OFT instead

uses its official combined-policy LIBERO results (Kim et al., 2025a; OpenVLA-OFT Authors, 2025). PaliGemma–Naive uses action binning; FAST-R/FAST-D use relative/delta actions.

ActionPiece (Lian et al., 2026), MemoryVLA (Shi et al., 2026), SP-VLA (Li et al., 2026), CoWVLA (Yang et al., 2026), and PD-VLA (Song et al., 2025) use their own reported results. ActionPiece averages five 24-episode SimplerEnv runs per task; MemoryVLA excludes LIBERO-90; SP-VLA uses OpenVLA on LIBERO and CogACT on SimplerEnv; PD-VLA follows its February 2026 revision. SimplerEnv entries cover four WidowX tasks. Source protocols are retained; only ACTWEAVE is evaluated here.

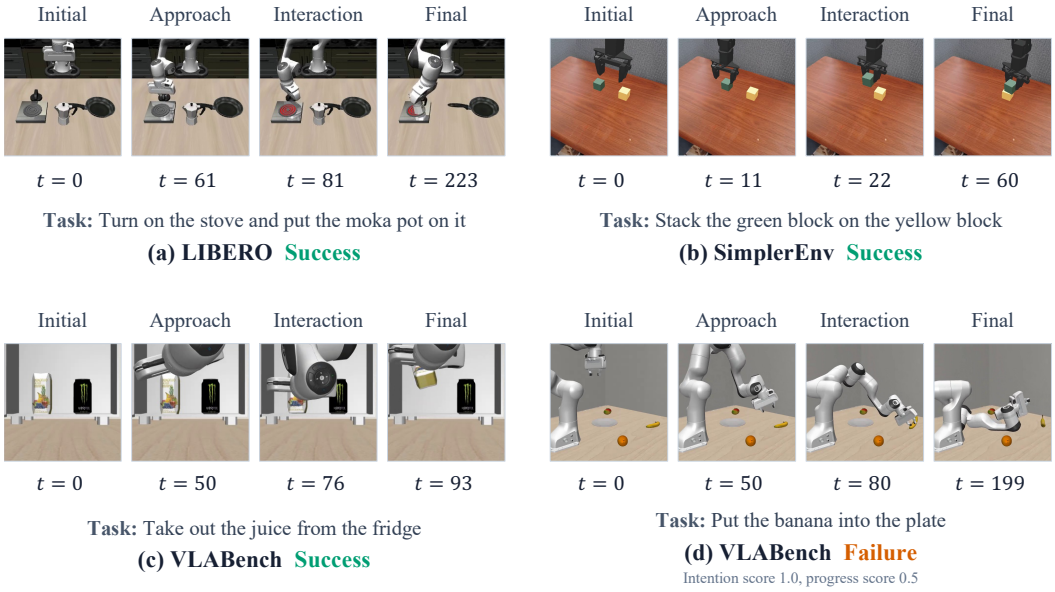


Figure 5: Representative ACTWEAVE simulation rollouts. Frames progress from the initial state through approach and interaction to the final observation. (a–c) Successful executions on LIBERO, SimplerEnv, and VLABench. (d) A VLABench partial failure that selects the correct target (intention 1.0) but does not complete placement (progress 0.5).

C ADDITIONAL RESULTS

C.1 COMPLETE ACTWEAVE BENCHMARK RESULTS

Table 5 gives the counts underlying the ACTWEAVE row in Table 1, using Table 4's settings. No evaluation errors are excluded. LIBERO uses environment seed 7+episode ID; SimplerEnv measures success at the final environment step.

Final-policy deployment latency and memory. Table 6 times the final checkpoints and deployment settings on an exclusive RTX PRO 6000 Blackwell GPU at batch size 1. Three independent launches replay identical observation sequences, each with 50 warm-up and 1,000 measured calls; percentiles pool 3,000 samples per benchmark.

Timing spans host observations to executable actions, including preprocessing, transfer, generation, decoding, aggregation, and ensembling; loading and simulation are excluded. Amortized latency divides by E. Device memory is sampled every 100 ms, including CUDA context and allocator cache. SR is measured separately in closed-loop rollouts; schedule-only timings follow Appendix C.2.

C.2 CONTROLLED ACTION-GENERATION LATENCY

Table 7 measures schedules implemented with a common backbone, not released baseline policies. Chunk-based schedules generate five 7-dimensional actions and execute two; single-action AR

Table 5: Complete closed-loop results for the ACTWEAVE row in Table 1. SR: success rate; episodes report successes over trials.

Benchmark	Split / task	SR (%) ↑	Episodes
LIBERO	Spatial	97.4	487/500
	Object	99.8	499/500
	Goal	95.6	478/500
	Long	90.8	454/500
	Overall	**95.9**	**1,918/2,000**
SimplerEnv	Spoon	79.2	95/120
	Carrot	70.8	85/120
	Block	91.7	110/120
	Eggplant	95.8	115/120
	Overall	**84.38**	**405/480**

Table 6: Final-policy deployment cost. $H/P/E$: prediction horizon/patch length/execution horizon; p50/p95: latency percentiles; Amort.: per executed action; Mem.: sampled peak GPU memory.

Benchmark	$H/P/E$	Chunk p50/p95 (ms, ↓)	Amort. p50/p95 (ms/action, ↓)	Mem. (GiB, ↓)
LIBERO	10/5/1	115.41 / 145.35	115.41 / 145.35	9.17
SimplerEnv	10/5/2	131.35 / 173.31	65.68 / 86.65	9.13

generates and executes one. Variable-length AR uses 13 tokens including EOS. These measurements isolate scheduling cost and are separate from the final-policy deployment results in Table 6.

Table 7: Controlled full-chunk latency (ms). p50/p95: latency percentiles; Amort.: p50 per executed action; G/L: global/local steps. Bold/underline: best/second-best latency.

Generation schedule	p50 / p95 ↓	Amort. ↓	G ↓	L ↓
Single-action global AR	168.62 / 185.25	168.62	7	0
Variable-length global AR	305.28 / 318.82	152.64	13	0
Flat global AR	789.67 / 802.52	394.84	35	0
Parallel continuous head	**38.88 / 39.49**	**19.44**	1	1
Flow-matching head (10 steps)	<u>53.76</u> / <u>55.63</u>	<u>26.88</u>	1	10
ACTWEAVE hierarchical AR	58.65 / 60.72	29.32	1	11

C.3 VLABENCH

Protocol. We evaluate 50 episodes for each of 10 in-distribution primitive tasks (Track 1), using the Bridge-initialized 10k EMA policy and median decoding in Table 4. Official task limits are retained, with `max_substeps`=1. All 500 episodes are included; two physics exceptions count as failures. This is policy evaluation, distinct from reconstruction replay in Appendix C.4.

Selection and uncertainty. The final run achieves 214/500 successes (42.8%; descriptive Wilson 95% CI: 38.53–47.18%). The 500 episodes include 50 scene IDs used in this round's pilot selection; evaluation scenes were also reused in earlier experiments. The interval does not account for selection, task dependence, or training-seed variability.

Decoding comparison. With the same checkpoint and scene IDs, median decoding improves SR from 36.0% to 42.8%, with task-dependent gains and regressions (Figure 6); Table 8 gives final counts. Continuing training to 13k with top-32 decoding instead yields 34.2%. The final gain therefore comes from token selection, not additional training or architectural changes, and covers only the evaluated in-distribution tasks.

Table 8: VLABench results with 10k EMA weights and median decoding. Succ.: successes/trials; SR: success rate (%); PE: physics exceptions (all failures).

Task	Succ.	SR ↑	PE ↓
Add condiment	25/50	50.0	0
Insert flower	11/50	22.0	0
Select book	36/50	72.0	0
Select chemistry tube	22/50	44.0	0
Select drink	13/50	26.0	0
Select fruit	19/50	38.0	0
Select mahjong	15/50	30.0	2
Select painting	15/50	30.0	0
Select poker	26/50	52.0	0
Select toy	32/50	64.0	0
Overall	**214/500**	**42.8**	2

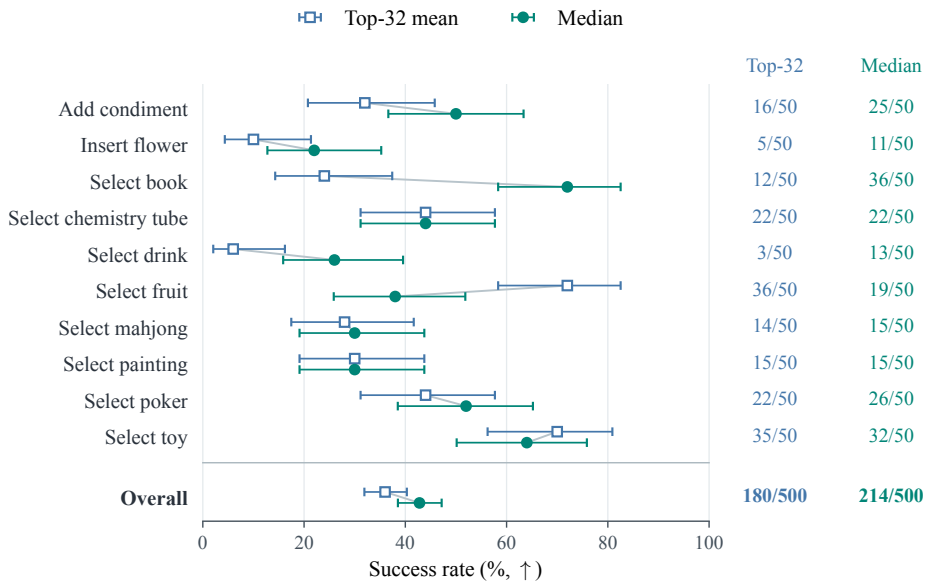


Figure 6: Effect of token selection on VLABench. Both decoders use the same 10k EMA checkpoint and execution horizon $E = 5$. Points show SR over 50 episodes per task, with a separate 500-episode overall row; labels give successes/trials. Whiskers are descriptive Wilson 95% intervals. Median decoding improves overall SR, with task-dependent gains and regressions.

C.4 Action Reconstruction under Distribution Shift

Protocol and metrics. We reconstruct 128 freshly generated expert trajectories from 2 VLABench tasks (64 each from `select_book` and `select_poker`; 9,788 timesteps) and replay them from their original environment initializations. Episodes are selected solely by successful original-action replay, independently of tokenizer outputs. OOD refers to transfer from the released tokenizers' documented training corpora to VLABench, not a held-out VLABench task or object split. Pretrained tokenizer weights and learned vocabularies remain fixed; normalization and adapter statistics are calibrated on VLABench training demonstrations. Replay SR measures reconstructed-trajectory completion on this subset, not policy success; Gap is 100% minus Replay SR.

Normalized action MAE averages 7 equally weighted components: 3 translation errors normalized by training-set Q01/Q99 statistics, the $\mathrm{SO}(3)$ geodesic rotation angle divided by π and counted as 3 components, and binary gripper disagreement. Errors include action-space conversion and resampling.

The baselines are OpenVLA binning (Kim et al., 2025b), BEAST (Zhou et al., 2025), VQ-VLA (Wang et al., 2025), ActionCodec (Dong et al., 2026), X-Tokenizer (Kang et al., 2026), FAST+ (Pertsch et al., 2025), and RDT2-RVQ (Liu et al., 2026b).

Table 9: Adapter-based OOD reconstruction on 128 VLABench episodes. Norm. MAE: normalized action mean absolute error; Replay SR: replay success rate of reconstructed actions; Gap: oracle replay SR minus Replay SR (pp).

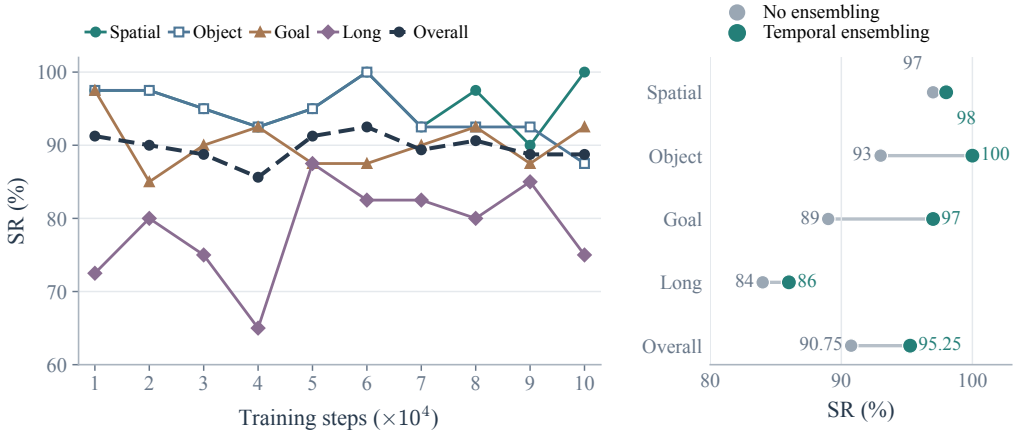
Pipeline	Norm. MAE ↓	Replay SR ↑	Gap ↓
OpenVLA binning	0.004645	113/128 (88.28%)	11.72 pp
BEAST	0.006411	107/128 (83.59%)	16.41 pp
VQ-VLA	0.010149	108/128 (84.38%)	15.63 pp
ActionCodec	0.009760	116/128 (90.63%)	9.38 pp
X-Tokenizer	0.098318	9/128 (7.03%)	92.97 pp
FG-DCT (ours)	**0.003190**	**123/128 (96.09%)**	**3.91 pp**
FAST+	0.005209	114/128 (89.06%)	10.94 pp
RDT2-RVQ	0.005647	110/128 (85.94%)	14.06 pp

Action-space adapters. VQ-VLA uses observation-relative poses and a binary gripper. ActionCodec uses Franka-DROID Euler deltas with symmetric Q01/Q99 scaling, resampled from 15 to 10 Hz. RDT2-RVQ uses observation-relative UMI poses, column-wise 6D rotations, and metric gripper aperture, resampled from 30 to 10 Hz. X-Tokenizer uses 26D absolute actions with ground-truth chunk-boundary states. FG-DCT uses normalized absolute targets, $P = 5$, and $\gamma = 50$. Thus, comparisons include each tokenizer's action-space adapter.

Results. FG-DCT leads both metrics (Table 9); differing baseline rankings confirm that reconstruction error and task completion measure complementary aspects of fidelity.

C.5 Training Dynamics and Decoding on LIBERO

Figure 7a evaluates 10 checkpoints with argmax and 160 episodes each; SR varies non-monotonically, particularly on LIBERO-Long. At 100k, ensembling with $E = 1$ improves SR over argmax with $E = 5$ (panel b, 400 episodes each), jointly changing aggregation and replanning frequency. Main-policy results instead use the 80k checkpoint in Table 4.



(a) Success across checkpoints

(b) Ensembling and replanning

Figure 7: LIBERO training and decoding. (a) Argmax success across 10 checkpoints, with 160 episodes each. (b) Decoding at 100k training steps on 400 episodes per configuration; temporal ensembling also changes the execution horizon from 5 to 1.

D Controlled Architectural Ablations

Shared protocol. All variants train directly from Qwen3-VL-4B-Instruct for 10k steps with seed 7, shared data order, and the LIBERO settings in Tables 3–4. The 100k-step LR schedule is retained (5k warmup), with all modules trainable. Four H800 or RTX PRO 6000 GPUs are used; Flat Global AR additionally uses activation checkpointing. Evaluation uses 160 episodes (4 initial states per task; trial IDs 0–3), environment seed 7 + trial ID, argmax, no ensembling, $H/P/E = 10/5/5$, and a 520-action limit for every suite. Reference A is the 10k full model, not the main 80k policy.

Latency measurement. All six models use the same observations on an exclusive RTX PRO 6000 Blackwell GPU at batch size 1. Three process launches each perform 50 warm-up and 200 measured calls; p50 pools 600 samples. Timing includes preprocessing, transfer, generation, and reconstruction, but excludes disk I/O and simulation.

Architectural controls. IDs follow Table 2. B projects flattened token and positional embeddings into the VLM space. C scans low-to-high frequencies within each action dimension; D predicts all patch tokens without generated-token conditioning, using the same local transformer as A. E replaces D's transformer with position-specific linear heads. F removes both patch encoding and the local module, predicting all tokens through the VLM. All retain the DCT grid; parameter counts can differ. G/L counts sequential evaluations per chunk, including one local evaluation per patch for D and E.

Reporting. Table 10 reports per-suite SR; aggregate differences appear in Table 2. Exact paired McNemar tests for B–A, C–A, and F–A yield $p = 0.064$, 1.000, and 0.064, respectively; none rejects equal success probabilities at the 0.05 level. Tests for D–A and E–D are not reported because matching episode-level outcomes for D are unavailable.

Table 10: Per-suite LIBERO ablation SR (%, ↑), with 40 episodes per suite. IDs follow Table 2.

ID	Variant	Spatial ↑	Object ↑	Goal ↑	Long ↑
A	**ActWeave (full)**	97.5	100.0	97.5	77.5
B	w/ Projection Encoder	97.5	90.0	90.0	70.0
C	w/ Raster Local AR	97.5	100.0	97.5	80.0
D	w/ Parallel Local Decoding	97.5	95.0	92.5	75.0
E	w/ Linear Local Head	97.5	97.5	85.0	70.0
F	Flat Global AR	95.0	95.0	90.0	70.0

Results. Hierarchy reduces VLM evaluations from 70 to 2 without shortening the token grid. Wavefront decoding further reduces local steps from 70 to 22 relative to raster while achieving comparable SR. Parallel variants are faster but have lower observed SR; contextual patch encoding and a local transformer also improve point estimates over their simpler replacements. These single-seed comparisons characterize quality–latency trade-offs at a fixed budget, not converged-model rankings or formal noninferiority. Small latency differences between A/B and D/E fall below between-process variation and do not establish component overhead.

E Patch Length and Prediction Horizon

Computational scaling. For $N = \lceil H/P \rceil$ patches, generation uses N global and $N(D_a + P - 1)$ local evaluations. When P divides H, these sequential step counts are

$$S_{\text{global}} = \frac{H}{P}, \qquad S_{\text{local}} = H + \frac{H(D_a - 1)}{P}. \tag{16}$$

Larger patches reduce both counts at fixed H. Actual latency also includes one VLM prefill, cached global decoding, and patch encoding.

Training protocol. We cross $H \in \{10, 20, 30\}$ with $P \in \{2, 5, 10\}$ and add $(10, 1)$. Each configuration restores the same 60k-step LIBERO backbone trained at $(10, 5)$, reinitializes action

modules, and adapts with seed 7. Optimization follows Table 3, except that action/language/vision learning rates are $2 \times 10^{-4}/10^{-6}/2 \times 10^{-7}$ and loss weights are uniform. Fixed 400- and 1k-step checkpoints are evaluated without rollout-based selection.

All configurations share 224,368 ordered anchors with valid 30-action targets, two-camera 230-pixel crops, and 256-bin proprio. Tokenization uses training-set min–max normalization, temporal DCT, $\gamma = 10$, and 2,048 codes.

Evaluation, timing, and statistics. Each checkpoint uses 200 paired episodes (trial IDs 43–47 across 40 tasks), argmax, no ensembling, $E = 3$, and a 520-action limit. Separate full-chunk timing uses batch size 1, 20 warm-up and 100 measured calls, including preprocessing, transfer, generation, and reconstruction but excluding loading and simulation; it does not measure real-time deadline failures.

CIs use 10,000 suite-stratified task-cluster bootstrap draws preserving paired trials. Exact McNemar tests apply Holm correction over nine comparisons against $(10, 5)$ per budget. Intervals exclude training-seed variability. Figure 3 shows the 1k-step grid; Table 11 includes both budgets and $(10, 1)$.

Table 11: Complete patch/horizon sweep. SR: success rate (%) with 95% task-cluster CI; p50/p95: full-chunk latency (ms); Pred./Codec: native action-space MAE over the first 3 actions; G/L: global/local steps. Each row uses 200 episodes.

Adapt. steps	(H, P)	SR [95% CI] ↑	p50 ↓	p95 ↓	Pred. ↓	Codec ↓	G/L ↓
400	$(10, 1)$	78.5 [72.5, 84.0]	370.8	463.9	0.03248	0.01172	10/70
400	$(10, 2)$	83.5 [77.5, 89.5]	201.3	252.2	0.02866	0.01147	5/40
400	$(10, 5)$	84.0 [77.5, 90.0]	104.2	105.1	0.01648	0.01141	2/22
400	$(10, 10)$	78.0 [70.0, 85.5]	71.9	72.6	0.04593	0.01120	1/16
400	$(20, 2)$	81.0 [75.0, 87.0]	358.0	399.0	0.03263	0.01147	10/80
400	$(20, 5)$	86.5 [81.0, 92.0]	189.4	193.8	0.01742	0.01141	4/44
400	$(20, 10)$	79.5 [74.0, 85.0]	116.1	118.0	0.04426	0.01120	2/32
400	$(30, 2)$	85.0 [79.5, 90.5]	517.3	653.5	0.03538	0.01147	15/120
400	$(30, 5)$	86.0 [80.5, 91.0]	290.2	347.1	0.02058	0.01141	6/66
400	$(30, 10)$	69.0 [60.0, 78.0]	169.3	225.0	0.05461	0.01120	3/48
1k	$(10, 1)$	86.0 [80.5, 91.5]	344.0	348.4	0.02722	0.01172	10/70
1k	$(10, 2)$	86.5 [80.0, 92.5]	197.5	198.9	0.02454	0.01147	5/40
1k	$(10, 5)$	85.0 [79.5, 90.0]	106.7	111.1	0.01386	0.01141	2/22
1k	$(10, 10)$	87.0 [81.0, 92.5]	73.5	76.1	0.03528	0.01120	1/16
1k	$(20, 2)$	85.5 [80.0, 91.0]	378.6	381.8	0.02807	0.01147	10/80
1k	$(20, 5)$	84.5 [80.0, 89.0]	199.2	201.2	0.01636	0.01141	4/44
1k	$(20, 10)$	90.0 [86.0, 93.5]	117.9	118.6	0.03699	0.01120	2/32
1k	$(30, 2)$	86.0 [80.0, 92.0]	535.2	537.4	0.03199	0.01147	15/120
1k	$(30, 5)$	87.5 [83.0, 91.5]	292.8	295.8	0.01630	0.01141	6/66
1k	$(30, 10)$	78.5 [72.5, 84.0]	173.6	220.0	0.04113	0.01120	3/48

Success–latency trade-off. The 1k-step point-estimate frontier contains $(10, 10)$ and $(20, 10)$, but no SR comparison against the default survives Holm correction (Table 12). At 400 steps, only $(30, 10)$ differs significantly (-15.0 pp; 95% CI $[-24.5, -5.5]$; adjusted $p = 4.69 \times 10^{-4}$), so the early degradation is not a general failure of large patches.

Prediction versus codec error. Diagnostics use 400 fixed training-distribution anchors (10 per task). MAE covers the first three denormalized actions and all seven dimensions, including continuous gripper outputs. Prediction MAE is lowest at $P = 5$ across horizons, whereas codec MAE stays near 0.011; lower prediction error does not consistently maximize closed-loop SR.

Scope. This is single-seed adaptation, not training from the original VLM. Action modules contain 24.700M/27.849M/37.296M/53.040M parameters for $P = 1/2/5/10$; diagnostics use training, not held-out, episodes.

Table 12: Paired SR differences relative to $(H, P) = (10, 5)$ at 1k adaptation steps. ΔSR and CIs are in pp; p is the exact McNemar value and p_{Holm} adjusts 9 comparisons.

(H, P)	ΔSR $\uparrow$	95% CI	p	p_{Holm}
$(10, 1)$	$+1.0$	$[-4.5, 6.5]$	0.868	1.000
$(10, 2)$	$+1.5$	$[-3.5, 6.5]$	0.711	1.000
$(10, 10)$	$+2.0$	$[-4.5, 8.5]$	0.627	1.000
$(20, 2)$	$+0.5$	$[-6.0, 7.0]$	1.000	1.000
$(20, 5)$	-0.5	$[-5.5, 4.5]$	1.000	1.000
$(20, 10)$	$+5.0$	$[0.0, 10.0]$	0.110	0.881
$(30, 2)$	$+1.0$	$[-4.5, 6.0]$	0.864	1.000
$(30, 5)$	$+2.5$	$[-1.5, 6.5]$	0.511	1.000
$(30, 10)$	-6.5	$[-12.5, -0.5]$	0.072	0.652

F Action-Token Structure and Local Predictability

F.1 Local Dependencies in Action Grids

Figure 4(a,b) analyzes LIBERO tokens, separately from VLABench reconstruction replay.

Setup. An episode-disjoint split across 40 tasks uses 1,393/150/150 training/validation/test episodes (seed 7), with up to three uniformly spaced, unpadded windows per episode (4,179/450/450 total). Each pairs five historical and five future actions. Historical actions form 7×5 grids using min–max normalization and scale-10 DCT rounding. Normalization uses full-data statistics, potentially including held-out distribution information; probe standardization uses training data only.

Probes use seeds 7/17/29, AdamW (10^{-3} LR, 10^{-4} weight decay), batch 512, and up to 200 epochs with validation selection and patience 12. Metrics weight tasks and within-task episodes equally; 2,000 paired task–episode bootstrap draws follow seed averaging. Labels $(x, y, z, r_x, r_y, r_z, g)$ denote translation, rotation, and gripper controls.

Frequency statistics and local prediction. The prior uses position-wise training counts with uniform pseudocount mass 1. A 2-layer, width-128, 4-head Transformer predicts tokens from earlier wavefronts only. Table 13 reports entropy and held-out cross-entropy (CE).

Table 13: Action-token statistics by DCT frequency. Entropy and CE (cross-entropy) are in bits/token; Zero is the zero-coefficient fraction.

Freq.	Entropy	Zero (%)	Prior CE $\downarrow$	Local CE $\downarrow$
0	3.4027	20.65	3.4710	3.3248
1	1.8168	60.79	1.8427	1.5107
2	0.6053	89.03	0.6115	0.4534
3	0.2143	96.87	0.2200	0.1267
4	0.1074	98.57	0.1129	0.0356

Local context reduces CE from 1.252 to 1.090 bits/token (12.9%; paired reduction 95% CI: 0.137–0.189), with gains at every frequency. This establishes predictability, not lossless compression or safe coefficient removal.

F.2 Probing Aggregated Patch Representations

Setup. Representations in Table 14, paired with proprio, feed separate 2-layer, width-256 MLP probes predicting five future actions. MSE covers six training-standardized pose dimensions; gripper uses an auxiliary loss. The structured encoder (2 width-128 Transformer layers, axis embeddings) and a parameter-matched MLP reconstruct identical historical tokens through a common decoder; their frozen 256D outputs are compared with the full token grid. This probes readout-accessible information, not the jointly trained policy.

Table 14: Future-action prediction from historical tokens. MSE: standardized error; brackets: 95% CIs.

Representation	MSE ↓ [95% CI]
None (state only)	0.487 [0.403, 0.585]
Single action token	0.503 [0.415, 0.607]
All 35 action tokens	0.141 [0.116, 0.173]
Structured patch representation	0.299 [0.241, 0.365]
MLP patch representation	0.148 [0.122, 0.181]

Results. Structured aggregation underperforms the full grid and MLP aggregation. Neither aggregate meets the prespecified 5% MSE noninferiority margin with an MLP probe; MLP aggregation passes with a linear readout, indicating probe dependence rather than lossless aggregation. Tasks are shared across splits, some probes reach the epoch limit, and intervals are unadjusted and exclude full training-seed variability. These results do not establish policy-level benefits of aggregation.