WAYPOINTVLA: TRAINING-EFFICIENT VISION-LANGUAGE-ACTION LEARNING VIA SPARSE ACTION WAYPOINTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) policies typically predict dense action chunks over a long, fixed future horizon, requiring supervision for every intermediate action. However, under closed-loop control, only a small portion of each predicted chunk is executed before replanning, creating a prediction–execution mismatch. Our analysis of a $\pi_{0.5}$ policy on LIBERO further shows that distant future actions become increasingly unreliable, questioning the necessity of dense long-horizon supervision. Meanwhile, within the practically useful horizon, action trajectories can be effectively reconstructed from only a sparse set of key actions using simple linear interpolation. Motivated by these, we introduce **WaypointVLA**, which predicts only two key actions—a start-point action and an endpoint action—while lightweight interpolation reconstructs the intermediate low-level controls. To support training, we derive sparse waypoint supervision directly from existing demonstrations and use it to fine-tune the original VLA model without modifying the base VLA backbone. By concentrating supervision on temporally informative actions rather than redundant intermediate controls, WaypointVLA achieves faster convergence and improved robustness to visual and environmental perturbations. We evaluate WaypointVLA on LIBERO and LIBERO-Plus using $\pi_{0.5}$ as the base model. For clarity, we refer to the original $\pi_{0.5}$ baseline as the *dense policy*. Remarkably, using only 200 demonstrations (13.9% of the full-data setting), WaypointVLA surpasses the full-data dense policy on both benchmarks while achieving a $15.2\times$ training speedup. These results suggest that sparse waypoint training provides a more efficient alternative to dense training.

1 INTRODUCTION

Vision-language-action models have recently emerged as a general framework for language-conditioned robotic manipulation. A common design choice in modern VLA policies is *action chunking*, which improves temporal consistency and alleviates the limitations of purely step-wise control (Black et al., 2026), and has therefore been widely adopted in continuous-action robot policies. However, this fixed-horizon formulation raises a less explored question: *is it necessary to densely predict and supervise every action over a long fixed horizon?*

In practice, VLA policies are typically executed in a closed-loop manner: they predict a chunk of H future actions but execute only the first few before replanning. For example, $\pi_{0.5}$ on LIBERO predicts $H = 50$ actions but executes only the first 10, leaving 80% of the predicted actions unused. As shown in Figure 1 (a), evaluation on 13 held-out episodes further shows that arm-action prediction error increases beyond the first 10 steps, making distant predictions progressively less reliable. This reveals a clear prediction–execution mismatch: many supervised future actions are neither executed nor accurately predicted. We further examine whether dense intermediate actions are necessary for trajectory representation. As shown in Figure 1 (b,c), a small number of action anchors can reconstruct the dense trajectory with low error through linear interpolation, with four anchors already substantially reducing interpolation error. Together, these results indicate that dense long-horizon supervision contains substantial temporal redundancy and may incur unnecessary training cost without providing corresponding benefits for closed-loop execution.

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.

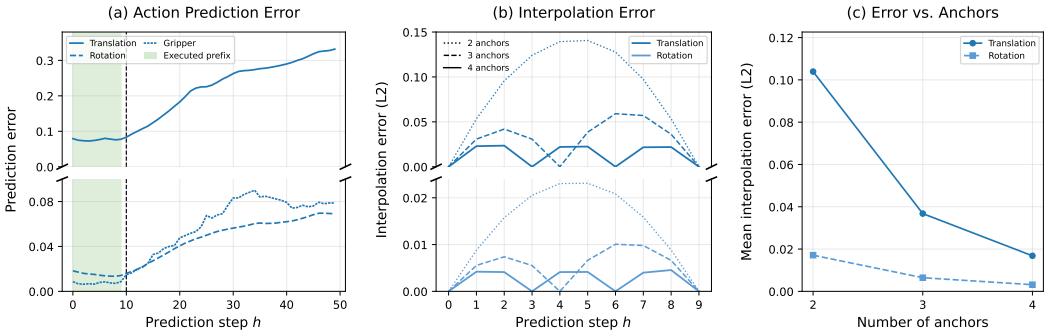


Figure 1: Prediction error and action interpolation analysis of $\pi_{0.5}$ policy on LIBERO. (a) Prediction errors increase beyond the executed prefix (the first 10 steps). (b,c) Dense action trajectories can be reconstructed from a small number of anchors with low interpolation error.

Motivated by this finding, we introduce WaypointVLA, a sparse future action prediction framework for VLA models. Instead of predicting a dense chunk of future actions, WaypointVLA predicts only two key actions—a start-point action and an endpoint action—while the intermediate low-level controls are reconstructed through lightweight linear interpolation. To support training, we construct waypoint supervision directly from existing demonstrations using an action-based waypoint selection procedure. The resulting training data can be derived directly from standard dense demonstrations without additional data collection. This sparse formulation substantially reduces the number of supervised action targets, thereby lowering training cost and accelerating optimization. Meanwhile, the underlying VLA architecture requires no modification, allowing WaypointVLA to be readily transferred across different VLA backbones.

We evaluate WaypointVLA on LIBERO and LIBERO-Plus using $\pi_{0.5}$ as the base VLA. Under matched training settings, WaypointVLA consistently outperforms dense policy across different data scales while requiring only about 40% of the training time. Notably, using only 13.9% of the training demonstrations required by the full-data $\pi_{0.5}$ dense policy, WaypointVLA already surpasses its performance on both LIBERO and LIBERO-Plus. On the challenging LIBERO-Plus benchmark, it further achieves a 13.8% relative improvement over the dense policy, demonstrating improved robustness under visual and environmental perturbations.

Our contributions are summarized as follows:

- We identify a *prediction–execution mismatch* in dense VLA policies and further show that dense action trajectories can be accurately approximated from only a few sparse anchors. Motivated by these observations, we introduce **WaypointVLA**, which replaces dense fixed-horizon action supervision with sparse waypoint supervision.
- We propose a simple waypoint data construction strategy that derives sparse waypoint supervision directly from existing demonstrations, without requiring additional data collection or changes to the underlying VLA architecture.
- Experiments on LIBERO and LIBERO-Plus show that WaypointVLA achieves higher performance with substantially less training data and computation, including a 13.8% relative improvement over the dense policy on the challenging LIBERO-Plus benchmark.

2 Related Work

Vision-language-action models and action chunking. Large-scale VLA models combine pre-trained vision-language representations with robot action generation, enabling generalizable language-conditioned manipulation across diverse tasks and embodiments (Kim et al., 2024; Black et al., 2024; Intelligence et al., 2025). Most modern VLA policies generate actions as dense chunks over a fixed future horizon. Action chunking can reduce the effective decision horizon and mitigate compounding errors, while improving temporal consistency in high-frequency control (Zhao et al., 2023; Black et al., 2026). These benefits have made dense action chunks a common action representation in modern VLA policies. However, existing approaches supervise every action within the prediction horizon. In contrast, we question whether such dense supervision is necessary.

Temporal abstraction and waypoint policies. Temporal abstraction has also been explored in imitation learning via waypoint-based action representations. HYDRA (Belkhale et al., 2023) introduces a hybrid action space combining sparse high-level waypoints for coarse motion with dense low-level actions for precise manipulation, dynamically switching between the two modes. SPHINX (Sundaresan et al., 2025) similarly pairs sparse waypoint motion with dense end-effector control, using task-relevant salient points for long-range movement and switching to dense control near regions requiring fine interaction. These methods demonstrate the value of sparse temporal abstractions, but rely on explicit policy switching between waypoint and dense-control modes. Instead, WaypointVLA retains a unified VLA action interface and directly replaces dense action-chunk supervision with sparse action-waypoint supervision, reconstructing intermediate controls via lightweight interpolation. Our focus is thus not on hybrid switching, but on reducing the temporal density of supervision in modern continuous VLA policies.

3 WAYPOINTVLA

3.1 PROBLEM FORMULATION

VLA policies typically take an image observation I, a language instruction L, and the robot state $\mathbf{s}_t$ as input to predict an action chunk $\mathbf{a}_{1:H}$ over the next H timesteps:

$$\mathbf{a}_{1:H} = \pi_\theta(I, L, \mathbf{s}_t), \tag{1}$$

where H denotes a fixed prediction horizon. For example, $\pi_{0.5}$ on LIBERO uses $H = 50$ and generates 50 future actions at each inference step. However, our experiments show that these actions are not entirely reliable—up to 80% of the predicted actions are never directly executed because the policy is queried again before the full chunk is consumed. This motivates rethinking the fixed action-chunk paradigm: dense supervision may produce smoother predictions, but it also increases training cost and does not necessarily yield useful actions for closed-loop execution.

We therefore explore a *waypoint* formulation. A fixed prediction horizon can be suboptimal because different situations require different temporal granularities. Smooth free-space motions may remain predictable over longer horizons, whereas contact-rich behaviors such as grasping and releasing contain discontinuities that benefit from finer-grained control. Instead of predicting an entire dense action chunk, we predict only two key actions: a *start point* and an *end point*:

$$\mathbf{a}_{\text{start}}, \mathbf{a}_{\text{end}} = \pi_\theta(I, L, \mathbf{s}_t). \tag{2}$$

A dense action sequence can then be recovered through linear interpolation:

$$\hat{\mathbf{a}}_h = \mathbf{a}_{\text{start}} + \frac{h-1}{H-1}\left(\mathbf{a}_{\text{end}} - \mathbf{a}_{\text{start}}\right), \quad h = 1, \ldots, H, \tag{3}$$

where H denotes the number of low-level actions reconstructed between the two predicted waypoints. This minimal formulation allows us to directly evaluate the feasibility of sparse waypoint prediction without modifying the underlying VLA architecture.

3.2 WAYPOINT DATA CONSTRUCTION

To transform dense action trajectories into sparse waypoint supervision without modifying the underlying VLA architecture, we construct waypoint training data entirely from existing demonstrations. We illustrate this process on LIBERO, which provides Franka Emika Panda arm data (7-DoF) containing 1,693 episodes across 40 tasks. The construction proceeds in two stages. First, we use ground-truth trajectory replay to determine whether sparse waypoint reconstruction is feasible and to select an appropriate anchor construction strategy. Second, we convert the selected waypoint segments into boundary and recovery samples for policy learning.

Sparse waypoint feasibility. We represent an expert demonstration as $\tau = \{(o_t, \mathbf{a}_t)\}_{t=0}^{T-1}$, where o_t denotes the observation before executing action $\mathbf{a}_t$, and $\mathbf{a}_t$ consists of a 6-DoF continuous arm action and a 1-DoF discrete gripper action. Our key hypothesis is that the arm-action trajectory is locally smooth and therefore contains substantial temporal redundancy: most intermediate arm actions can be reconstructed from a small number of action anchors. To validate this hypothesis and determine a suitable waypoint selection strategy, we reconstruct dense trajectories using the linear interpolation

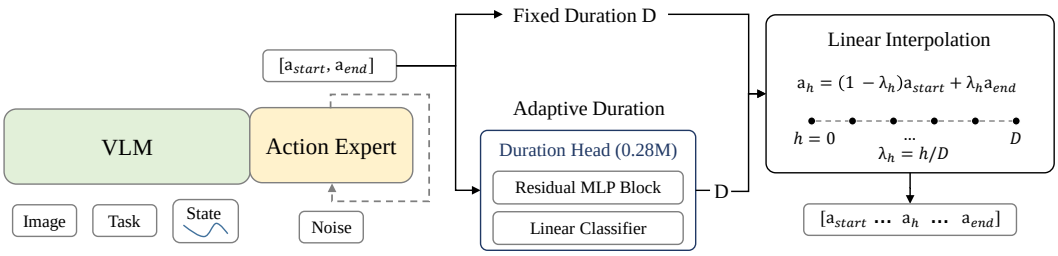


Figure 2: WaypointVLA predicts only start and end actions for each segment while keeping the base VLA architecture unchanged. Intermediate arm actions are reconstructed by linear interpolation, and the segment duration is either fixed or predicted by a lightweight duration head.

scheme in Eq. 3 and evaluate execution success on 19 diagnostic episodes with ground-truth actions. The selected strategy is then extended to the full dataset.

Oracle validation. We first replay the original dense trajectories from their recorded simulator states, verifying that the stored actions can successfully complete the task on the simulated platform. This oracle replay achieves 19/19 success on the diagnostic episodes. We then apply the same replay to all available demonstrations and retain only episodes that succeed in all three trials, filtering the original 1,693 episodes down to 1,439 valid trajectories.

Uniform sampling baseline. As a first test of temporal redundancy, we uniformly select K anchors along each trajectory and retain only their actions. The arm actions between adjacent anchors are reconstructed via linear interpolation (Eq. 3), while the discrete gripper action is handled by mid-point switching: the first half follows the start-anchor action, and the second half follows the end-anchor action. At $K = 32$, uniform sampling succeeds on $10/19$ diagnostic episodes. Given an average trajectory length of 165.7 steps, this corresponds to a $5.18\times$ compression ratio, showing that even a trajectory-agnostic selector can remove most dense action targets while preserving successful execution on a considerable subset of episodes. This provides initial evidence of substantial temporal redundancy in dense action trajectories. However, uniform sampling allocates anchors solely by elapsed time, without accounting for action complexity, and may therefore oversample smooth free-space motion while missing critical transitions. This motivates an action-aware selection strategy that preserves critical transitions while maintaining sufficient temporal coverage.

Two-stage anchor selection. A useful waypoint set must satisfy two complementary requirements. First, it should preserve action-critical points. Second, it should prevent individual segments from becoming excessively long, since interpolated actions are executed open-loop within each segment and small errors may accumulate into state drift. We therefore separate anchor construction into an action-aware stage and a temporal-coverage stage. The first stage selects a compact set of action-critical anchors, while the second supplements this set to a final fixed budget.

1. **Stage 1: Action-Greedy Selection ($K = 32$).** We first preserve the trajectory endpoints and all gripper transitions as mandatory anchors. For each gripper event at timestep t_g, we retain $\{t_g - 1, t_g, t_g + 1\}$ to preserve the local transition around grasping or releasing. Starting from these mandatory anchors, we reconstruct the 6-DoF arm-action trajectory by piecewise linear interpolation. For every unselected timestep, we compute the normalized reconstruction error

$$E_t = \frac{1}{6} \sum_{d=1}^{6} \left(\frac{a_{t,d} - \hat{a}_{t,d}}{\sigma_d + \epsilon} \right)^2, \tag{4}$$

where σ_d denotes the dataset-level standard deviation of action dimension d. At each iteration, the timestep with the largest reconstruction error is added to the anchor set $\mathcal{K}$. The trajectory is then re-interpolated and the reconstruction errors are recomputed. This process continues until $|\mathcal{K}| = K$. The resulting selector prioritizes high-curvature motion, abrupt velocity changes, nonlinear action transitions, and pre-contact deceleration. To determine a suitable Action-Greedy budget, we evaluate the single-segment recovery rate, defined as the fraction of waypoint segments that can be successfully executed independently using only their endpoint anchors and interpolation. Sweeping $K \in 16, 24, 32, 48, 64$, this rate

increases from 82.5% at $K = 16$ to 92.9% at $K = 32$, while further increasing K to 64 yields only a modest gain to 96.8% and reduces the compression ratio from $5.18\times$ to $2.59\times$. We therefore choose $K = 32$ as the compact base budget. Despite the high 92.9% single-segment recovery rate at $K = 32$, full-trajectory success is only $12/19$. One possible reason is that Action-Greedy may allocate multiple anchors to short, high-curvature regions while leaving long, smooth intervals sparsely covered, where prolonged open-loop interpolation can accumulate error.

2. **Stage 2: Longest-Segment Midpoint Supplementation ($K = 48$).** To improve temporal coverage without discarding action-critical anchors, we introduce Longest-Segment Midpoint Supplementation. Specifically, we retain the Action-Greedy anchor set and repeatedly identify the adjacent anchor pair (t_i, t_{i+1}) with the longest temporal interval and insert its midpoint until $|\mathcal{K}| = 48$. Compared with directly increasing pure Action-Greedy selection to $K = 48$, this strategy reduces the average maximum segment interval from 11.47 to 5.79 steps and the global maximum from 23 to 13 steps. Under the same $K = 48$ budget, it also improves full-trajectory success from $14/19$ to $16/19$, showing that temporal coverage provides a useful complement to reconstruction-error-based anchor selection. We also evaluate a larger waypoint budget of $K = 64$, which further improves success to $17/19$ but reduces the compression ratio from approximately $3.45\times$ at $K = 48$ to $2.59\times$. We therefore adopt $K = 48$ as our final operating point.

Final training data. We apply the proposed two-stage anchor selection to the $1{,}439$ episodes that pass oracle validation. Across this full candidate set, linear interpolation between the selected anchors achieves an 88.50% success rate, indicating that the sparse waypoint representation remains valid at scale. Rather than discarding the 11.5% of episodes that fail under waypoint replay, we retain all $1{,}439$ episodes. Excluding harder episodes would bias the training distribution toward easy, easily interpolatable trajectories. Keeping them preserves task diversity and encourages the policy to correct deviations beyond simple interpolation.

Boundary and recovery samples. Given $K = 48$ anchors, each trajectory is divided into 47 waypoint segments, from which we construct two types of supervision with the same two-anchor prediction format. For adjacent anchors (t_i, t_{i+1}), a boundary sample uses the observation at t_i and is supervised by $\left[\mathbf{a}_{t_i}, \mathbf{a}_{t_{i+1}}\right]$, corresponding to the current and segment-end actions. This yields $67{,}633$ boundary samples from $1{,}439$ demonstrations. For any intermediate timestep t satisfying $t_i < t < t_{i+1}$, a recovery sample pairs the current observation with $\left[\mathbf{a}_t, \mathbf{a}_{t_{i+1}}\right]$, yielding $154{,}318$ recovery samples in total. Boundary and recovery samples thus differ only in whether the observation lies at a segment boundary or inside the segment. In our main setting, we retain all boundary samples and subsample recovery samples at a 3:1 boundary-to-recovery ratio, resulting in $67{,}633$ boundary and $22{,}544$ recovery samples. This substantially reduces supervision density while preserving the original demonstrations and task coverage.

3.3 WaypointVLA Training and Execution

WaypointVLA does not require modifying the VLA backbone. For $\pi_{0.5}$, we preserve the original visual, proprioceptive, and language inputs together with its flow-matching action head. We only change the semantic meaning of the action output from a dense chunk to the waypoint pair in Eq. 2.

Training. We train the policy on a mixture of boundary and recovery samples. Given an observation o_t, the policy predicts a waypoint pair $\mathbf{W}_t = [\mathbf{a}_s, \mathbf{a}_e]$, corresponding to the start and end actions of the current segment. For a boundary sample, $\mathbf{a}_s$ is the action at the current segment boundary. For a recovery sample, $\mathbf{a}_s$ is the action associated with the current intermediate observation, while $\mathbf{a}_e$ remains the endpoint of the original segment.

Following the flow-matching objective used in $\pi_{0.5}$, we sample Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$ and construct the noisy waypoint target as

$$\mathbf{W}_t^\tau = \tau\boldsymbol{\epsilon} + (1 - \tau)\mathbf{W}_t, \tag{5}$$

where $\tau \in [0, 1]$ denotes the flow-matching timestep. The corresponding velocity target is $\boldsymbol{\epsilon} - \mathbf{W}_t$, and we optimize

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}\left[\left\|\mathbf{v}_\theta(\mathbf{W}_t^\tau, o_t, \tau) - (\boldsymbol{\epsilon} - \mathbf{W}_t)\right\|_2^2\right]. \tag{6}$$

Inference. At test time, the policy predicts a start waypoint $\hat{\mathbf{a}}_s$ and an endpoint $\hat{\mathbf{a}}_e$. Given a temporal span D, the intermediate arm actions are reconstructed by linear interpolation:

$$\alpha_h = \frac{h}{D}, \qquad \hat{\mathbf{a}}_h^{\text{arm}} = (1-\alpha_h)\hat{\mathbf{a}}_s^{\text{arm}} + \alpha_h \hat{\mathbf{a}}_e^{\text{arm}}, \quad h = 0, \ldots, D. \tag{7}$$

Here, D denotes the temporal span between the two waypoint actions, so the interpolation produces $D+1$ actions. For the discrete gripper action, we switch from the start command to the endpoint command at the segment midpoint. The policy is queried again after the interpolated segment is executed, allowing the next waypoint pair to be predicted from the latest observation.

Duration prediction. We first validate the feasibility of our waypoint framework using a *fixed-duration* baseline. The duration is set according to the training data distribution; for LIBERO, we sweep over candidate values $\{3, 6, 8, 10\}$ and select $D = 6$ based on empirical performance. However, a fixed duration is suboptimal because different segments naturally differ in temporal span, and it contradicts the core idea of adaptive control horizon. Therefore, as illustrated in Figure 2, we train a lightweight *duration head* g_ϕ that takes the two predicted arm waypoints as input and outputs a discrete duration:

$$\hat{D} = g_\phi\left(\hat{\mathbf{a}}_s^{\text{arm}}, \hat{\mathbf{a}}_e^{\text{arm}}\right). \tag{8}$$

The head is implemented as a stack of MLP blocks ($\approx$ 0.28M parameters) and is trained as a multi-class classifier over candidate durations. It is trained on the full set of boundary and recovery samples (222K samples) and converges in approximately 2 minutes on a single 4090-24GB GPU.

4 Experiments

4.1 Experiment Setup

We evaluate WaypointVLA on two simulation benchmarks, LIBERO and LIBERO-Plus, and further validate it through real-world robot experiments. Unless otherwise specified, we report results using the adaptive duration strategy. Implementation details are provided in Appendix A.

LIBERO. LIBERO (Liu et al., 2023) is a single-arm simulation benchmark featuring a 7-DoF Franka Emika Panda robot. It consists of four suites—LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long—each containing 10 tasks. To study data efficiency, we construct three training budgets by sampling 5, 10, and 15 demonstrations per task, resulting in 200, 400, and 600 training episodes, respectively. We additionally evaluate the full-data setting with 1,439 episodes. Dense policy and WaypointVLA use exactly the same demonstrations under each training budget. For each evaluated checkpoint, we repeat the evaluation three times, with 10 rollouts per task in each run, and report the mean success rate.

LIBERO-Plus. LIBERO-Plus (Fei et al., 2025) extends LIBERO with seven categories of visual and environmental perturbations. Without any additional training or adaptation, we directly evaluate the robustness of policies trained exclusively on the original LIBERO demonstrations. We use all 10,030 evaluation episodes provided by LIBERO-Plus.

Real-world experiments. We further evaluate WaypointVLA on an ALOHA pick-and-place task: "pick up a yellow carrot and place it into a green basket." Both the dense policy and WaypointVLA are initialized from the same $\pi_{0.5}$ checkpoint adapted on 98 ALOHA demonstrations. We then collect 50 task-specific demonstrations under the in-distribution setup shown in Figure 4 (a), and construct waypoint supervision with $K = 32$ anchors, including 20 Action-Greedy anchors and 12 Longest-Segment Midpoint anchors. We evaluate both policies under the in-distribution setting and unseen layout and lighting perturbations.

4.2 Main Results on LIBERO and LIBERO-Plus

Effectiveness and training efficiency. We first evaluate whether sparse waypoint supervision can improve policy learning while reducing training cost. Table 1 compares $\pi_{0.5}$ dense policy with WaypointVLA under matched demonstration budgets. Across all data scales, WaypointVLA consistently achieves higher success rates while using only 37–40% as many supervised training samples as dense policy training. With 200, 400, 600, and 1,439 demonstrations, WaypointVLA improves the

Table 1: Training efficiency under matched demonstration budgets. Dense and Waypt. denote $\pi_{0.5}$ dense policy and WaypointVLA ($\pi_{0.5}$), respectively. Waypoint supervision uses only 37–40% as many training samples as dense supervision. Training time is measured on 8×H20 (96 GB) GPUs. Improvements over dense policy are shown in cyan.

Episodes	Method	Samples	Spatial	Object	Goal	Long	**Overall**	Train Time (h)	Speedup
200	Dense	33.7K	97.0	97.0	77.0	82.0	88.3	3.32	2.46×
	Waypt.	12.5K	100.0	99.0	88.0	79.0	**91.5** (+3.2)	**1.35**	
400	Dense	66.4K	99.0	98.0	82.0	77.0	89.0	6.25	2.60×
	Waypt.	25.1K	100.0	99.0	93.0	84.0	**94.0** (+5.0)	**2.40**	
600	Dense	99.6K	98.0	100.0	82.0	77.0	89.3	9.23	2.49×
	Waypt.	37.6K	100.0	99.0	87.0	90.0	**94.0** (+4.7)	**3.70**	
1.4K	Dense	223.4K	99.0	100.0	85.0	77.0	90.3	20.50	2.41×
	Waypt.	90.2K	99.0	99.0	92.0	89.0	**94.8** (+4.5)	**8.52**	

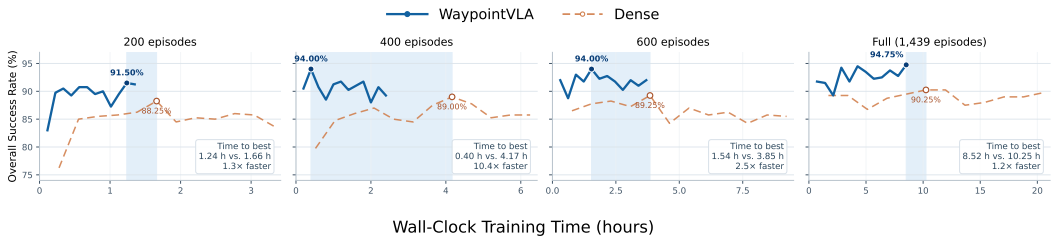


Figure 3: Wall-clock convergence of $\pi_{0.5}$ dense policy and WaypointVLA on LIBERO across different demonstration budgets. Each point represents an epoch checkpoint, with wall-clock time measured on one 8×H20 (96GB) node. Annotated markers denote the best-performing checkpoints, and the shaded regions indicate the differences in time to best. WaypointVLA reaches its best performance earlier and achieves higher success rates across the evaluated checkpoints.

overall LIBERO success rate by 3.2, 5.0, 4.7, and 4.5 percentage points, respectively. The improvement is particularly evident on the more challenging Goal and Long suites. These results indicate that sparse waypoint supervision does not sacrifice task performance despite substantially reducing the temporal density of action targets. The reduction in supervision also translates directly into training efficiency. Under the same demonstration budgets, WaypointVLA requires only 1.35, 2.40, 3.70, and 8.52 hours of training, compared with 3.32, 6.25, 9.23, and 20.50 hours for dense training. This corresponds to a consistent 2.4–2.6× wall-clock speedup on the same 8×H20 node. Notably, WaypointVLA trained with only 200 demonstrations already reaches 91.5% success, surpassing the dense policy trained with all 1,439 demonstrations (90.3%).

Faster convergence. Figure 3 compares the wall-clock convergence of the $\pi_{0.5}$ dense policy and WaypointVLA under different demonstration budgets. WaypointVLA consistently reaches a higher best success rate with a shorter time-to-best across all data scales, with especially pronounced gains at the 400- and 600-episode settings. These results show that sparse waypoint supervision enables faster and more effective policy learning.

Robustness across perturbations. Standard LIBERO performance is relatively saturated for modern VLA policies, so we further evaluate robustness on LIBERO-Plus under diverse perturbations. Following the official protocol, we compare against representative VLA methods, including π_0 (Black et al., 2024), π_0-FAST (Pertsch et al., 2025), UniVLA (Wang et al., 2026), OpenVLA-OFT (Kim et al., 2025), and ABot-M0 (Yang et al., 2026). As shown in Table 2, WaypointVLA consistently outperforms the matched dense baseline across all data budgets. With 1,439 episodes, it improves the overall success rate from 73.2% to 83.3% (+10.1 points). More importantly, it also surpassing the strong recent VLA foundation model ABot-M0 (80.5%) despite using fewer training demonstrations. Even with only 200 demonstrations, WaypointVLA reaches 78.9%, outperforming

Table 2: Success rate (%) on LIBERO-Plus under seven perturbations. Dense and Waypt. denote $\pi_{0.5}$ dense policy and WaypointVLA ($\pi_{0.5}$), respectively. The top block reports representative methods trained with 1,693 episodes, while the bottom block shows our controlled comparisons. Waypt. achieves the best overall performance with fewer training episodes; improvements over dense policy are shown in cyan. Total is computed over all 10,030 evaluation instances rather than as an unweighted average of the seven categories.

Method	Episodes	Camera	Robot	Language	Light	Background	Noise	Layout	Total
π_0	1,693	13.8	6.0	58.8	85.0	81.4	79.0	68.9	53.6
π_0-FAST		65.1	21.6	61.0	73.2	73.2	74.4	68.8	61.6
UniVLA		1.8	46.2	69.6	69.0	81.0	21.2	31.9	42.9
OpenVLA-OFT		56.4	31.9	79.5	88.7	93.3	75.8	74.2	69.6
ABot-M0		60.4	67.9	**86.4**	**96.2**	91.6	**86.4**	82.6	80.5
Dense	200	53.1	63.5	71.3	80.0	80.4	68.0	73.3	68.9
Waypt.		67.7	72.1	78.7	91.6	88.0	79.8	81.3	78.9 (+10.0)
Dense	400	55.7	66.7	71.7	81.0	78.9	71.4	73.8	70.5
Waypt.		68.1	73.0	79.4	89.6	86.7	80.5	80.3	78.9 (+8.4)
Dense	600	58.2	66.6	73.9	84.3	81.4	74.2	73.9	72.3
Waypt.		69.5	74.2	79.1	91.2	88.8	79.6	83.9	80.0 (+7.7)
Dense	1439	56.3	67.6	77.3	83.4	83.6	74.8	75.5	73.2
Waypt.		**73.3**	**75.3**	82.8	95.3	**93.7**	85.4	**84.0**	**83.3** (+10.1)

Table 3: Ablation studies of WaypointVLA. (a) Controlling for the output horizon. (b) Effect of boundary–recovery ratio on Full1439. (c) Effect of the adaptive duration on LIBERO-Plus.

(a) Output horizon

Episodes	Dense-H50	Dense-H2	Waypt.
200	88.3	88.3	**91.5**
400	89.0	85.3	**94.0**
600	89.3	87.5	**94.0**
1,439	90.3	87.3	**94.8**

(b) B:R ratio

B:R	Success (%)
7:1	90.5
3:1	**94.8**
1:1	88.8

(c) Duration prediction

Episodes	Fixed $D = 6$	Adaptive D
200	75.8	**78.9** (+3.1)
400	74.3	**78.9** (+4.6)
600	74.5	**80.0** (+5.5)
1,439	78.5	**83.3** (+4.8)

the dense policy trained on 1,439 demonstrations. These results show that waypoint supervision improves both data efficiency and robustness under distribution shifts.

Generalization across VLA backbones. To evaluate the generality of sparse waypoint supervision, we further apply the same LIBERO training formulation to SmolVLA (Shukor et al., 2025), a smaller and architecturally distinct VLA backbone. As shown in Table 4 (a), the benefits of waypoint supervision transfer consistently across both training-data budgets. With 400 demonstrations, waypoint supervision improves success from 30.8% to 35.8% while substantially reducing training time, and the same trend persists with the full training set.

4.3 Ablation Studies

Is the gain simply caused by a shorter output horizon? To test whether the gain comes merely from predicting fewer actions, we compare WaypointVLA with Dense-H2, which predicts two adjacent 7-D actions and thus has the same output size. In contrast, WaypointVLA predicts two non-adjacent actions at the boundaries of a waypoint segment. As shown in Table 3 (a), Dense-H2 consistently underperforms both the original dense policy and WaypointVLA, indicating that the gain comes from waypoint supervision rather than a shorter output horizon alone.

Effect of Recovery supervision. WaypointVLA combines boundary samples at segment transitions with recovery samples from intermediate states to support closed-loop replanning. Table 3 (b) shows that a moderate amount of recovery supervision is most effective: the 3:1 boundary-to-recovery ratio achieves the best success rate of 94.8%, outperforming both sparser and denser recovery sampling. This suggests that intermediate-state supervision is beneficial, while over-sampling recovery states may dilute the segment-boundary supervision that defines the primary waypoint transitions. We therefore adopt the 3:1 ratio in our main experiments.

Table 4: Training efficiency and transfer results. (a) SmolVLA transfer on 8×RTX 4090 GPUs. (b) Real-world $\pi_{0.5}$ training on 4×8 H20 (96 GB) GPUs, with inference on a single RTX A6000. Dense and Waypt. denote dense policy and WaypointVLA, respectively.

(a) Transfer to SmolVLA

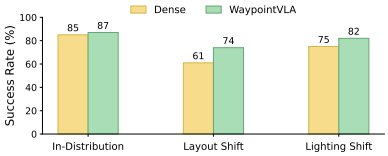
Episodes	Method	Best SR	Total Training Time (h)	Speedup
400	Dense	30.8	2.49	**2.65**×
	Waypt.	**35.8** (+5.0)	**0.94**	
1,439	Dense	34.0	8.30	**2.48**×
	Waypt.	**36.8** (+2.8)	**3.35**	

(b) Real-world efficiency

Method	Train Time (h)	Speedup	Inference Latency (ms)
Dense	4.2	**5.3**×	205.8
Waypt.	**0.8**		213.8



(a) Real-world perturbation settings.



(b) Success rates under perturbation.

Figure 4: Real-world robustness evaluation. Both policies are trained only in the in-distribution setting and directly evaluated under unseen layout and lighting perturbations. WaypointVLA consistently exhibits stronger robustness than $\pi_{0.5}$ dense policy across these perturbations.

Adaptive execution duration. The waypoint representation determines *what* actions define a local segment, while the execution duration determines *how long* that segment should be executed. We therefore replace the fixed $D = 6$ strategy with a learned Duration Head. As shown in Table 3 (c), adaptive duration consistently improves LIBERO-Plus performance across all training scales, with gains of roughly 3–6 percentage points. On the full-data setting, the success rate increases from 78.5% to 83.3%. These results suggest that while a fixed duration can provide a strong baseline, adapting the execution horizon is particularly beneficial under the perturbations.

4.4 Real-World Robot Experiments

We further evaluate WaypointVLA on real-world robot manipulation to examine whether its efficiency gains transfer beyond simulation. Both policies are initialized from the same $\pi_{0.5}$ checkpoint adapted on 98 ALOHA demonstrations, and are then fine-tuned using the same task-specific demonstrations under identical robot, camera, and control settings. As shown in Figure 4, both policies are trained exclusively in the in-distribution setting and directly evaluated under unseen layout and lighting perturbations. For each evaluation setting, we conduct 100 trials and report the corresponding task success rate. To ensure a fair comparison, the dense and waypoint policies are evaluated in alternating order under matched scene configurations, thereby minimizing potential bias caused by changes in object placement, illumination, or other environmental conditions. WaypointVLA maintains comparable in-distribution performance while exhibiting substantially stronger robustness under both perturbations. Meanwhile, Table 4 (b) shows that waypoint supervision substantially reduces training time while introducing only a small interpolation overhead at inference. Together, these results provide complementary real-world evidence that sparse waypoint supervision improves training efficiency and robustness with minimal additional deployment cost, and that its benefits are not limited to simulator-specific dynamics or action trajectories.

5 Conclusion

Motivated by the observations that VLA policies are trained to predict long, fixed-length action chunks but execute only a small portion before replanning, and that intermediate actions can be approximated through interpolation, we present WaypointVLA, which replaces dense action prediction with two action waypoints and reconstructs intermediate controls through linear interpolation. On LIBERO and LIBERO-Plus, WaypointVLA outperforms the dense baseline using only 13.9% of its demonstrations and achieves a 2.4–2.6× training speedup under matched demonstration budgets. Experiments with SmolVLA and on a real-world robot further demonstrate improved training efficiency and robustness. These results highlight sparse waypoint supervision as a simple and effective approach to efficient VLA learning.

AI USE STATEMENT

We used GPT-5.6 Sol to improve the fluency and presentation of the manuscript. All technical content, experimental design, results, and conclusions were developed and verified by the authors.

REPRODUCIBILITY STATEMENT

Detailed implementation and training configurations are provided in Appendix A. Anonymous training and evaluation code are included in the supplementary material. We will publicly release the full training code, evaluation code, and model checkpoints upon publication.

REFERENCES

Suneel Belkhale, Yuchen Cui, and Dorsa Sadigh. Hydra: Hybrid robot actions for imitation learning. In *Conference on Robot Learning*, pp. 2113–2133. PMLR, 2023.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael 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. *arXiv preprint arXiv:2410.24164*, 2024.

Kevin Black, Manuel Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. *Advances in Neural Information Processing Systems*, 38:33383–33407, 2026.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

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

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

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

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

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. *arXiv preprint arXiv:2501.09747*, 2025.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Priya Sundaresan, Hengyuan Hu, Quan Vuong, Jeannette Bohg, and Dorsa Sadigh. What's the move? hybrid imitation learning via salient points. In *International Conference on Learning Representations*, volume 2025, pp. 51806–51821, 2025.

Yuqi Wang, Xinghang Li, Wenxuan Wang, Junbo Zhang, Yingyan Li, Yuntao Chen, Xinlong Wang, and Zhaoxiang Zhang. Unified vision-language-action model. In *International Conference on Learning Representations*, volume 2026, pp. 80929–80944, 2026.

Yandan Yang, Shuang Zeng, Tong Lin, Xinyuan Chang, Dekang Qi, Junjin Xiao, Haoyun Liu, Ronghan Chen, Yuzhi Chen, Dongjie Huo, et al. Abot-m0: Vla foundation model for robotic manipulation with action manifold learning. *arXiv preprint arXiv:2602.11236*, 2026.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

A Implementation Details

Common training setup. We fine-tune $\pi_{0.5}$ end-to-end using bfloat16 precision and gradient checkpointing. The model contains approximately 4.14B parameters. State and action vectors are internally padded to 32 dimensions, while losses are computed only on the task-specific action dimensions. Unless otherwise specified, all experiments use AdamW with a peak learning rate of 2.5×10^{-5}, $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, weight decay 0.01, and gradient clipping at 1.0. We use cosine learning-rate decay to 2.5×10^{-6} and fix the random seed to 1000. Flow-matching inference uses 10 denoising steps.

LIBERO training. All LIBERO models are trained for 12 epochs with a per-GPU batch size of 16 and a global batch size of 128. The first four epochs are checkpointed every 0.5 epoch, followed by one checkpoint per epoch. Each process uses four DataLoader workers with a prefetch factor of 2 and persistent workers. Detailed training configurations, sample counts, and optimizer updates are summarized in Table 6.

Dense uses a 50-step action chunk and replans after executing the first 10 actions, whereas Waypoint uses `chunk_size=2` and `n_action_steps=2`. For the Dense-H2 control experiment, the dense training data are unchanged while both values are set to 2. Dense and Waypoint models use identical episode subsets and initialization.

LIBERO Duration Head. The Duration Head is trained independently from the VLA policy using all 67,633 Boundary and 154,318 Recovery pairs from Full1439, for a total of 221,951 samples. Its training configuration is summarized in Table 5(a).

The validation set is split at the episode level. After selecting epoch 13, the Duration Head is retrained on the full dataset for 13 epochs.

Real-world training. For the real-world experiments, both policies are initialized from the same $\pi_{0.5}$ checkpoint adapted on 98 ALOHA demonstrations. This initialization is trained for 10,000 optimizer steps with a global batch size of 256 and a dense action horizon of 50.

The task-specific dataset contains 50 demonstrations and 12,680 frames. Both Dense and Waypoint are trained for 120 epochs with a per-GPU batch size of 8, a global batch size of 256, and seed 1000. Dense fine-tuning uses all 12,680 training samples with a 50-step action chunk and executes 10 actions per policy query, resulting in 6,000 optimizer steps. Waypoint fine-tuning uses 2,066 training samples with `chunk_size=2` and `n_action_steps=2`, resulting in 1,080 optimizer steps. We use 50 warmup steps for Waypoint training, and assign equal loss weights of 0.5 to the start and endpoint anchors. The detailed configuration is reported in Table 5(b).

Real-world Duration Head. The real-world Duration Head uses a 12-D input formed by concatenating the two 6-D arm anchors and a residual MLP with hidden dimension 256. It is trained as a 50-way duration classifier using cross-entropy loss and Gaussian action noise with a standard deviation of 0.02. We use an episode-level 90/10 training/validation split with seed 1000. The best validation epoch is 172, after which the model is retrained on the full dataset.

Table 5: Training configurations for the Duration Head and real-world experiments.

(a) Duration Head

Configuration	Value
Input dimension	12
Hidden dimension	256
Residual blocks	2
Duration classes	50
Target range	1–50
Loss	Cross entropy
Anchor noise std.	0.02
Optimizer	AdamW
Learning rate	3×10^{-4}
Weight decay	1×10^{-4}
Validation split	Episode-disjoint
Selected epoch	13

(b) Real-world training

Configuration	Dense	Waypt.
Demonstrations	50	
Training samples	12,680	2,066
Batch / GPU	8	
Global batch	256	
Epochs	120	
Action steps	10	2
Total steps	6,000	1,080
Chunk size	50	2
Warmup steps	–	50
Anchor loss weights	–	$0.5/0.5$
Seed	1000	

Table 6: Training configuration, sample counts, and optimizer updates for the LIBERO experiments.

(a) Training configuration

Configuration	Dense	Waypt.
Batch / GPU	16	
Global batch	128	
Epochs	12	
Seed	1000	
Precision	bfloat16	
Chunk size	50	2
Action steps	10	2
Anchor loss weights	–	$0.5/0.5$

(b) Training samples and optimizer updates

	Samples		Optimizer Steps	
Episodes	Dense	Waypt.	Dense	Waypt.
200	33,742	12,533	3,168	1,176
400	66,386	25,066	6,228	2,352
600	99,618	37,600	9,348	3,528
1,439	223,390	90,177	20,952	8,460