VLA Knows Its Limits: Adaptive Execution Horizons for Robot Policies

Haoxuan Wang[1,2], Gengyu Zhang[1,2], Yan Yan[1], Ramana Rao Kompella[2], and Gaowen Liu[2]

[1] University of Illinois Chicago, USA
[2] Cisco Research, USA

Abstract. Action chunking has recently emerged as a standard practice in flow-based Vision-Language-Action (VLA) models. However, the effect and choice of the execution horizon—the number of actions to be executed from each predicted chunk—remains underexplored. In this work, we first show that varying the execution horizon leads to substantial performance deviations, with performance initially improving and then declining as the horizon increases. To uncover the reasons, we analyze the cross- and self-attention weights in flow-based VLAs and reveal two key phenomena: (i) intra-chunk actions attend invariantly to vision–language tokens, limiting adaptability to environmental changes; and (ii) the initial and terminal action tokens serve as stable anchors, forming latent centers around which intermediate actions are organized. Motivated by these insights, we interpret action self-attention weights as a proxy for the model's predictive limit and propose **AutoHorizon**, the first test-time method that dynamically estimates the execution horizon for each predicted action chunk to adapt to changing perceptual conditions. Across simulated and real-world robotic manipulation tasks, AutoHorizon is performant, incurs negligible computational overhead, and generalizes across diverse tasks and flow-based models. Demonstration videos are available at this project page.

Keywords: Action chunking · Flow-matching · VLA

1 Introduction

As a key milestone toward artificial general intelligence, embodied AI seeks to endow agents with the ability to perceive, reason, and act within the physical world [14,22]. Recent advances in machine-learning–based control have enabled these agents to directly interact with real-world environments through learned representations [8,35,39]. Within this paradigm, Vision-Language-Action (VLA) models [3,12,16,19,26,27,40] have emerged as a promising direction for their ability to ground visual perception and linguistic instruction into executable actions, demonstrating strong performance across diverse language-conditioned robot manipulation tasks.

To better capture multimodal action distributions and enforce temporal consistency, *action chunking* [8,17,39] has become a standard practice for VLAs

trained via imitation learning [1,34]. Instead of predicting a single action at each step, the policy outputs a sequence of actions—an action chunk. During policy rollout, the robot executes the initial portion (and only rarely the entirety) of the predicted chunk before re-planning, discarding the remaining actions [3,8]. The total chunk length is termed the *prediction horizon*, and the executed prefix is the *execution horizon* [4]. This formulation establishes a closed-loop prediction mechanism, sacrificing long-term consistency for reactivity [21].

Empirically, we observed that the policy's behavior is inherently tied to the length of its executed prefix. As illustrated in Fig. 1, varying the execution horizon leads to substantial performance fluctuations—ranging from consistent successes to frequent failures. Interestingly, the performance also exhibits a characteristic trend: it initially improves and then declines as the execution horizon increases. These findings underscore the importance of selecting an appropriate execution horizon and motivate a fundamental yet underexplored question in action chunking: **How should the execution horizon be determined?**

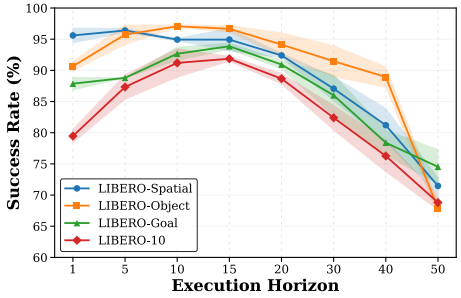

Fig. 1: The average success rates on LIBERO under $\pi_{0.5}$. Varying the execution horizon consistently leads to substantial success rate fluctuations. The policy performance exhibits a peaked pattern, initially improving and then declining as the execution horizon increases.

Prior works [3,8,12,24,39] typically set a *fixed* execution horizon, chosen either by human heuristics or through exhaustive evaluation across multiple configurations. Such brute-force tuning quickly becomes time- and compute-intensive as the prediction horizon and task complexity increases. Furthermore, we argue that a fixed execution horizon is inherently suboptimal. [21] showed that short horizons improve reactivity but induce instability due to frequent cross-chunk transitions, whereas long horizons enhance temporal smoothness at the expense of responsiveness. Since the balance between consistency and reactivity naturally varies across different phases of policy rollout, the optimal execution horizon should likewise adapt over time. For instance, reaching toward a coffee carafe favors a longer horizon for smooth motion, whereas pouring into a cup requires shorter horizons for heightened responsiveness. These insights emphasize the need of determining the execution horizon adaptively on a per-chunk basis.

In this paper, we aim to automatically and efficiently determine the execution horizon for each action chunk within flow-based VLAs [12,24]. To explain why the policy performance exhibits the characteristic trend shown in Fig. 1, we analyze how action generation integrates linguistic and visual information through the attention mechanism [30]. Our analysis reveals two key phenomena: ❶ Intra-chunk actions consistently attend to the same vision–language tokens, indicating that they rely on fixed perceptual contexts. These contexts provide useful guidance for early actions but become increasingly outdated for later ones

as the environment changes, revealing the limited adaptability of the predicted chunks. ❷ Predicted actions exhibit strong attention to the initial and terminal action tokens, with correspondence strength remaining high before sharply decaying as temporal distance increases. We refer to these boundary tokens as *radial action sinks*, which serve as stable anchors around which intermediate actions are organized.

Building on the observations of the limited adaptability of intra-chunk actions and their reliance on radial action sinks, we formulate execution horizon determination as **estimating the predictive limit of VLAs**. We interpret the action self-attention weights as implicit indicators of the model's prediction confidence and propose **AutoHorizon**, a dynamic execution horizon determination strategy for flow-based VLAs. Our approach leverages the intrinsic structure of attention weights to infer the temporal limit of the model's reliable forecasting capability. Specifically, we introduce a bidirectional soft-pointer mechanism that locates the first turning points where the attention mass ceases to advance and begins to plateau. These turning points, identified for both initial and terminal radial action sinks, define the estimated execution horizon.

Our contributions are threefold. **(1)** We focus on flow-based VLAs trained with action chunking, and provide both theoretical and empirical analyses of their performance pattern with respect to the execution horizon. We further uncover its underlying causes through two key observations linking attention correspondences to the model's predictive limit. **(2)** Building on these insights, we propose AutoHorizon, a novel attention-guided strategy that dynamically estimates the execution horizon for each action chunk, allowing the policy to adapt to varying perceptual conditions. **(3)** Extensive experiments on simulated and real-world robot manipulation tasks demonstrate that our method generalizes across different flow-based policies, incurs negligible computational overhead, and outperforms strong baselines that rely on exhaustive fixed-horizon tuning.

2 Related Work

2.1 Vision-Language-Action Models

The remarkable progress of large language models (LLMs) [25,29,33] and vision-language models (VLMs) [2, 6, 41] has sparked growing interest in AI systems capable of acting within the physical world. Among these advances, *Vision-Language-Action (VLA) models* [3, 5, 11, 12, 16, 18, 19, 24, 31, 36, 38, 40, 42] have emerged as a unifying paradigm that integrates perception, reasoning, and control. By jointly training on large-scale datasets of robotic and human demonstrations, VLAs enable end-to-end learning from visual and linguistic inputs to motor actions, demonstrating strong generalization across diverse language-conditioned manipulation tasks. Within this paradigm, diffusion- and flow-based VLAs [3, 12, 24, 27] have received particular attention for their sample-efficient generation process and high-quality action synthesis. Conditioned on the visual and linguistic embeddings extracted from the backbone VLMs [2, 6], these models iteratively sample from an action distribution to produce low-level motor

commands for robotic control, achieving fine-grained temporal consistency and robust performance across complex environments.

2.2 Action Chunking

Action chunking policies [3,8,12,35,39] generate short sequences of consecutive actions for a given state. ACT [39] first introduced the idea of action chunking and demonstrated that modeling temporally consistent action sequences enables the learning of fine-grained behaviors, outperforming policies that predict single actions. Subsequent works have extended this concept in several directions. ACT further applied weighted averaging to overlapping actions between chunks to enhance policy smoothness. BID [21] provided a theoretical analysis from a policy learning perspective, showing that action chunking promotes long-term temporal consistency but sacrifices short-term reactivity. They also proposed a rejection sampling strategy to select the best-performing chunk to address this trade-off. RTC [4] explored action chunking under asynchronous execution [27], formulating chunk prediction as an image inpainting problem [28].

Despite these advances, the *choice* of the execution horizon remains largely unexplored. Existing works [3,8,39] typically set this value using human heuristics, determined by the robot's control frequency and the desired duration of each motion segment. This practice has left the influence of the execution horizon on VLA performance largely unexplored, prompting a fundamental question: *are there better ways to determine the execution horizon?*

2.3 Attention Weights

Attention weights [9, 13, 32, 37] serve as interpretable indicators of information flow between tokens in Transformer-based models [30]. By examining self-attention patterns in large language models, StreamingLLM [32] observed that LLMs tend to focus disproportionately on initial tokens—a phenomenon termed *attention sink*. Preserving the corresponding key–value pairs was shown to enable efficient, infinite-length text processing without additional fine-tuning. Extending this concept to multimodal models, [13] found that large vision-language models often assign high attention weights to visually salient but semantically irrelevant tokens. To mitigate this, they proposed redistributing attention away from such tokens to improve cross-modal alignment and visual grounding.

In this work, we analyze the attention weights within flow-based VLAs [12, 24], uncovering key insights into how actions attend to other modalities and therefore propose to interpret action self-attention weights as informative cues for estimating the execution horizon.

3 Methodology

3.1 Preliminary

Denote the pretrained diffusion-/flow-based Vision-Language-Action (VLA) model as $\pi(\mathbf{A}_t|\mathbf{o}_t, \mathbf{c})$, where $\mathbf{o}_t$ represents the input visual observations at time step t,

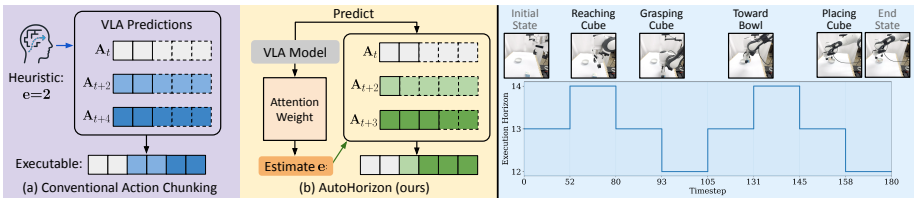

Fig. 2: Left: (a) In conventional action chunking, the execution horizon e is heuristically chosen by humans and remains fixed across chunks. (b) In contrast, AutoHorizon (our method) dynamically estimates the execution horizon for each predicted chunk based on the attention weights from the VLA model. **Right:** Real-world demonstration of showing how the estimated execution horizons evolve during policy rollout. When the environment is stable and reactivity is less critical (*e.g.*, reaching the cube or moving toward the bowl), the estimated horizon increases to promote smooth, stable motion. Conversely, during physical interaction (*e.g.*, grasping or placing the cube), the execution horizon shortens to enhance reactivity and adaptability.

and $\mathbf{c}$ denotes the corresponding language command. The policy is trained to predict a sequence of p continuous actions, $\mathbf{A}_t = [\,\mathbf{a}_t,\, \mathbf{a}_{t+1},\, \ldots,\, \mathbf{a}_{t+p-1}\,]$, referred to as an *action chunk*. Here, the parameter $p \in \mathbb{N}$ specifies the **prediction horizon**, *i.e.*, the temporal window over which the model forecasts future actions conditioned on the current perceptual-linguistic context $(\mathbf{o}_t, \mathbf{c})$. During execution, the agent typically performs the first e actions from the predicted chunk before re-sampling new input observations and generating the next action chunk, where $e \in \mathbb{N}$ defines the **execution horizon**.

Attention weights are widely employed to quantify the correspondence between different tokens. Let T_v, T_l, and T_a denote the number of encoded vision, language, and action tokens within the VLA respectively. Within a standard transformer-based attention mechanism, the attention weight matrix $\mathbf{S}$ is defined as the post-softmax similarity between the query and key embeddings:

$$\mathbf{S} = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right), \tag{1}$$

where $\mathbf{Q} \in \mathbb{R}^{T_a \times d}$ represents the queries derived from action tokens, $\mathbf{K} \in \mathbb{R}^{(T_v+T_l+T_a)\times d}$ denotes the concatenated keys from vision, language, and action tokens, and d is the shared token feature dimension. Each entry $\mathbf{S}_{ij}$ captures the relative attention weight between the i-th action query and the j-th key token, thus encoding how strongly the policy attends to specific visual regions, linguistic cues, or consecutive actions when generating its action sequence.

In the following, we first show that flow-matching policy performance exhibits a peaked trend with respect to the execution horizon, underscoring the need and feasibility to identify an optimal value. To explain this phenomenon, we analyze the attention weights and reveal two key phenomena that shape predicted chunk behavior. Building on these insights, we introduce an efficient strategy for exe-

cution horizon estimation. Unless otherwise specified, all analyses are conducted using the state-of-the-art $\pi_{0.5}$ [12] model.

3.2 Existence of Optimal Execution Horizon

Consider an action chunking policy that employs a fixed execution horizon of length e throughout its rollout. Assume the policy executes a total of L low-level actions before task termination. Let δ^c denote the loss in final task reward incurred at each chunk transition, assumed to be independent of e. Let $\delta_j^d(e)$ represent the total divergence loss between the j-th executed action chunk and its corresponding expert trajectory segment. With $m = \lceil L/e \rceil$ executed chunks in total, the expected error accumulated over the rollout can be expressed as:

$$\mathcal{L}(e) = \sum_{i=0}^{m-1} \delta^c \; + \; \sum_{j=0}^{m} \delta_j^d(e). \tag{2}$$

Proposition 1 (Unique Error Minimizer). *Suppose $\delta_j^d(e)$ is a monotonically increasing function with respect to e and can be modeled as $\delta_j^d(e) = ke \log e$, where $k > 0$ is a scaling factor. Denote p as the prediction horizon. Assuming L is divisible by e, there exists a unique minimizer of $\mathcal{L}(e)$:*

$$e^* \;=\; \text{clamp}\Big(\Big\lceil \tfrac{\delta^c}{k} \Big\rceil \;\; or \;\; \Big\lceil \tfrac{\delta^c}{k} \Big\rceil - 1, 1, \, p\Big)\,, \tag{3}$$

such that $\mathcal{L}(e)$ is strictly decreasing on $(0, e^*)$ and strictly increasing on (e^*, ∞).

Proof is provided in the supplementary material. Eq. (3) indicates that when the policy is trained on a diverse set of the underlying action distributions and the chunk transition loss δ^c is large, a longer execution horizon is preferred to promote temporally consistent action sequences. Conversely, when the policy struggles to accurately model the implicit environment dynamics and the intra-chunk divergence loss $\delta^d(e)$ dominates, a shorter execution horizon becomes more favorable, enhancing reactivity to environmental changes. Our analysis can be viewed as an extension of [21]. By modeling the total policy rollout error, we demonstrate that an optimal execution horizon exists and that performance monotonically decreases as the horizon deviates from this optimum, thereby establishing an *explicit* trade-off between long-term consistency and short-term reactivity. Note that Eq. (3) serves only as a theoretical proof of existence and does not directly guide method design.

Empirically, we also observe the characteristic relationship between policy performance and execution horizon. As shown in Fig. 1, varying the execution horizon leads to substantial performance deviations, underscoring the importance of selecting an appropriate value of e. Across all evaluated tasks, the behavior of the policy performance aligns with our theoretical findings: the success rate initially increases and then declines as e grows, peaking at an intermediate value. This indicates that optimal performance arises from balancing consistency with reactivity. In the supplementary material, we further illustrate the

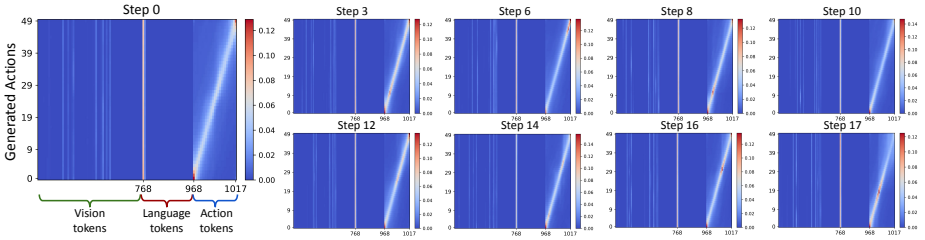

Fig. 3: Visualization of average attention weights in $\pi_{0.5}$ across different stages of task execution. Intra-chunk actions consistently attend to the same vision and language tokens across predicted chunks throughout the rollout. This invariance is consistently observed across different sampling steps, task rollouts, and pretrained models. The x-axis is rescaled for clarity of visualization.

case with a smaller prediction horizon ($p = 10$). In this constrained setting, the best performance typically occurs at the horizon boundary, i.e., $e^* = p$. This behavior arises because the policy is well-trained under a small p, allowing it to accurately capture the implicit environment dynamics. As a result, the predicted trajectories closely match the expert demonstrations, leading to $k \to 0$. However, once the execution horizon exceeds the prediction horizon ($e > p$), a pronounced performance drop emerges. We attribute this to a train–test mismatch, since the model has not been trained on trajectories longer than p.

3.3 VLA Knows Its Limits

To uncover the underlying causes of the observed performance pattern, we analyze the model's attention weights to examine how the VLA allocates focus across vision, language, and action tokens during action generation. By interpreting these attention distributions, we identify two key phenomena that jointly explain the empirical patterns reported in Sec. 3.2 and shed light on how attention dynamics reflect the VLA's intrinsic predictive limits.

❶ **Intra-chunk actions attend invariantly to vision–language tokens.** We visualize the cross-attention maps of the last sampling step from the $\pi_{0.5}$ model ($p = 50$) in Fig. 3. Each row in the heatmap represents the attention distribution between action queries and the concatenated sequence of vision, language, and action keys. The first 768 tokens correspond to visual input, followed by 200 language tokens, and the remaining correspond to action tokens. Column widths are rescaled for improved clarity.

From the figure, we observe a striking invariance: actions within the same chunk consistently attend to the same vision–language tokens with nearly identical importance. This suggests that, although the model predicts a temporally extended sequence of future actions, later actions fail to adaptively adjust to environmental changes. Instead, they repeatedly rely on static visual–linguistic features that are informative for early actions but increasingly outdated or misleading for later ones. Consequently, executing these later actions may become

redundant or even detrimental, contributing little new contextual information and corrective flexibility. This property manifests as overconfident policy rollouts with reduced reactivity and adaptability.

We also observe an unusually high concentration of attention on the *first language token*, a phenomenon that persists across all transformer blocks and sampling steps. This pattern resembles the attention sink effect reported in large language models [32]. However, unlike in LLMs, where sink tokens often encode structural or positional information, the language attention sink in VLAs appears largely redundant and carries minimal semantic value. Empirically, masking all language tokens (by setting their attention weights to zero) may results in only a marginal drop in task success rate. We infer that, due to the strong vision–language pretraining of the backbone model, most linguistic semantics are already embedded within the visual representations during action generation, making explicit linguistic attention largely superfluous.

❷ Radial action sinks emerge at the sequence boundaries. In Fig. 4, we visualize the normalized self-attention maps among action tokens under varying prediction horizons. A consistent pattern emerges: strong attention is concentrated on both the initial and terminal action tokens. The correspondence strength remains high for a short span, then rapidly decays over a few actions before stabilizing into a low-attention plateau. We refer to these highly attended boundary tokens as **radial action sinks**.

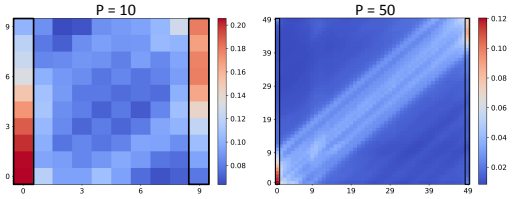

Fig. 4: Visualization of normalized action self-attention weights. The predicted actions exhibit strong attention toward the initial and terminal action tokens, with the correspondence strength remaining high before sharply decaying as the temporal distance increases. These boundary tokens (encircled in black) are referred to as *radial action sinks*.

Why do flow-based VLAs consistently concentrate attention at the beginning and end of each predicted chunk? We infer two underlying reasons. First, the initial action has the lowest cumulative error and serves as a stable anchor for subsequent actions. Later actions follow this anchor while increasingly attending to adjacent tokens, yielding smooth and temporally consistent trajectories. Second, because the policies are trained on expert demonstrations with randomly sampled starting timestamps, both the initial and terminal actions play a role in preserving inter-chunk continuity, reflecting the model's implicit objective to ensure smooth transitions across chunk boundaries. Together, these radial action sinks establish an attention structure in which the initial and terminal tokens define latent centers around which intermediate actions are organized, revealing an intrinsic bias in the model's temporal reasoning process.

Action self-attention as an indicator of predictive limit. Building on these two observations, we conclude that intra-chunk actions exhibit limited adaptability and attend strongly to radial action sinks. Therefore, we *pose to interpret the action self-attention weights as implicit indicators of the model's predictive*

Algorithm 1 AutoHorizon

Input: Attention matrix $\mathbf{S}_t$ and its reverse $\tilde{\mathbf{S}}_t$ at time t, quantile q, threshold τ

Obtain row entropy H_t with Eq. (5)
Retain low-entropy rows with Eq. (4)
// **Forward traversal**
Compute μ_t with Eq. (6)
$\Delta\mu_t \leftarrow \texttt{diff}(\mu_t)$
$P_t \leftarrow \{\, i \mid \Delta\mu_t[i] < \tau \,\}$
$N_f \leftarrow \lfloor \mu_t[\min(R_t \cap P_t)] \rfloor + 1$
// **Backward traversal**
Apply step 3-6 to $\tilde{\mathbf{S}}_t$
Obtain backward horizon N_b
// **Horizon determination**
if $N_f + N_b \geq p$ **then** $N \leftarrow p$ **else** $N \leftarrow N_f$

Output: N

limit. Specifically, when the self-attention weights associated with the radial action sinks remain high, the model is confident that its predicted actions are still aligned with the guiding anchors and thus valid under the current observation. In contrast, as these attention weights decline, the model increasingly conditions on its own previously generated actions rather than grounded sensory inputs. This shift toward self-referential dependence amplifies compounding errors and ultimately degrades reactivity and performance during extended rollouts.

3.4 AutoHorizon

Motivated by the above analysis, we propose leveraging attention weights as a proxy to estimate the execution horizon for *each* action chunk. Adopting a per-chunk horizon is intuitive: as task difficulty and environmental dynamics fluctuate throughout the policy rollout, the optimal execution horizon should adapt correspondingly to maintain an effective balance between temporal consistency and reactivity.

To this end, we introduce **AutoHorizon**—a data-adaptive approach that estimates execution horizons from the model's intrinsic attention dynamics. Given the attention weights obtained at each sampling step t, we first extract the action self-attention maps and average them across all transformer blocks and attention heads, followed by normalization to ensure that each query–key distribution sums to one. The resulting normalized attention matrix is denoted as $\mathbf{S}_t \in \mathbb{R}^{p\times p}$, where p is the prediction horizon, and notations are reused from Eq. (1) for simplicity. Intuitively, $\mathbf{S}_t[i, j]$ quantifies how strongly the i-th query action attends to the j-th key action, revealing how far the model effectively "looks ahead." Our objective is to identify the first turning point in the attention trajectory—where the attention mass stops advancing and begins to plateau—which marks the natural boundary of the VLA's predictive limit, ensuring that all executed actions remain reliable while maximizing temporal consistency.

To estimate this turning point, we first retain only the rows of $\mathbf{S}_t$ that exhibit low entropy, defined as

$$R_t = \{\, i \mid H_t[i] \le Q_q(H_t) \,\}, \tag{4}$$

where

$$H_t[i] = -\frac{1}{\log p} \sum_j \mathbf{S}_t[i,j] \log \mathbf{S}_t[i,j], \tag{5}$$

and Q_q denotes the q-quantile of the entropy distribution across rows. This filtering step removes actions with uniformly diffused attention, preserving those with sharper, more confident patterns that provide reliable structural cues.

Next, we employ a bidirectional soft-pointer mechanism to locate the plateau. Two pointers, $q_s = 0$ and $q_e = p - 1$, are initialized at the start and end of the chunk, respectively. For the forward pointer q_s, the expected predictive horizon for each row i is computed as

$$\mu_t[i] = \max\left(\sum_{j=0}^{p-1} j\, \mathbf{S}_t[i,j],\ \max_{k \le i} \mu_t[k] \right), \tag{6}$$

where the non-decreasing constraint enforces monotonic progression and prevents backward jumps. We then compute the incremental change $\Delta\mu_t[i] = \mu_t[i] - \mu_t[i-1]$, which tracks the evolution of the attention trajectory. A large $\Delta\mu_t[i]$ indicates a sudden shift in attention focus, signaling the onset of a plateau. The set of actions preceding this plateau is defined as

$$P_t = \{\, i \mid \Delta\mu_t[i] < \tau \,\}, \tag{7}$$

where τ is a fixed threshold. At sampling step t, the forward execution horizon is then determined as

$$N_f = \lfloor \mu_t[\min(R_t \cap P_t)] \rfloor + 1. \tag{8}$$

The same procedure is applied to the reversed attention matrix $\tilde{\mathbf{S}}_t$ to obtain the backward horizon N_b. If the combined coverage satisfies $N_f + N_b \ge p$, a full-range coverage is adopted ($N = p$); otherwise, only the forward prefix length is used as the effective execution horizon ($N = N_f$). Empirically, we find that the former case typically occurs when the prediction horizon p is small, whereas the latter dominates for larger p. Algorithm 1 summarizes the procedure and a concrete example is included in the supplementary material.

4 Experiments

4.1 Experimental Settings

Models. We evaluate our method on two representative flow-based VLA models: $\pi_{0.5}$ [12] and GR00T N1.5 [24]. For $\pi_{0.5}$, we conduct experiments with two

Table 1: Performance comparison of $\pi_{0.5}$ on LIBERO benchmark under different prediction horizons. Best results are in **bold**.

Setting		$p = 10$				$p = 50$			
Task Suite		LIB-Spatial	LIB-Object	LIB-Goal	LIB-10	LIB-Spatial	LIB-Object	LIB-Goal	LIB-10
Static Oracle	$e = 0.2p$	98.5 ± 0.5	94.1 ± 1.0	90.4 ± 0.0	76.0 ± 1.2	94.9 ± 0.2	97.1 ± 0.2	92.7 ± 1.0	91.2 ± 2.3
	$e = 0.4p$	98.9 ± 0.4	98.1 ± 0.5	94.8 ± 0.9	82.8 ± 1.5	92.4 ± 0.3	94.1 ± 1.9	90.9 ± 0.7	88.7 ± 0.8
	$e = 0.6p$	98.8 ± 0.3	98.7 ± 0.7	95.2 ± 0.7	85.9 ± 0.8	87.1 ± 1.9	91.5 ± 2.5	86.0 ± 3.3	82.4 ± 2.0
	$e = 0.8p$	98.9 ± 0.4	99.1 ± 0.5	95.1 ± 1.1	88.5 ± 1.5	81.2 ± 2.7	88.9 ± 1.6	78.4 ± 2.0	76.3 ± 2.5
	$e = 1.0p$	99.1 ± 0.5	98.8 ± 0.9	97.2 ± 1.0	90.4 ± 0.6	71.5 ± 1.3	67.9 ± 0.9	74.5 ± 2.7	68.6 ± 1.2
Static Oracle+		99.1 ± 0.5	99.1 ± 0.5	97.2 ± 1.0	90.4 ± 0.6	96.4 ± 0.3	97.6 ± 0.6	93.9 ± 0.5	91.9 ± 0.4
Random		98.4 ± 1.2	98.4 ± 0.7	96.3 ± 0.7	86.3 ± 1.5	82.8 ± 1.5	90.3 ± 1.6	85.3 ± 2.3	83.3 ± 0.7
AutoHorizon		**99.1 ± 0.2**	**99.2 ± 0.3**	**97.5 ± 0.2**	**91.6 ± 0.7**	**96.5 ± 0.9**	**98.0 ± 0.6**	**94.4 ± 1.0**	**92.1 ± 1.0**

variants using prediction horizons of $p = 10$ and $p = 50$ to examine horizon-dependent behavior. For GR00T N1.5, we adopt the publicly released pretrained checkpoints with the default prediction horizon of $p = 16$. AutoHorizon operates on the first or third sampling step and typically uses fixed hyperparameters of $q = 0.9$ and $\tau = 0.3$, requiring no additional parameter tuning.

Baselines. We compare against the following baselines:

- **Static Oracle.** This corresponds to the conventional setting in action chunking policies, where a fixed execution horizon is maintained throughout rollout. Execution horizons are uniformly sampled for full range coverage.
- **Static Oracle+.** An enhanced version of Static Oracle that performs brute-force search over the prediction horizon, thereby achieving optimal performance under fixed horizon settings. It serves as a strong yet costly baseline, as it requires p rollouts per task.
- **Random:** To assess the effect of adaptive horizon selection, we include a stochastic baseline where the execution horizon is randomly sampled at each rollout step, following $e \sim \mathcal{U}(1, p)$. This baseline isolates the contribution of structured, attention-guided adaptation from performance variations arising purely from random horizon changes.

For all experiments, we report both the mean and standard deviation to ensure fair comparison and robust evaluation. We also compare against two adaptive re-planning baselines in the supplementary material. Although these baselines are not central to the focus of this paper, we include them to further demonstrate the effectiveness of the adaptive strategy within AutoHorizon.

4.2 Simulation Results

We evaluate AutoHorizon in simulated robotic manipulation environments that require both short- and long-horizon decision making. Our experiments leverage two benchmark datasets: the LIBERO dataset [20], which offers a diverse suite of single-arm manipulation tasks, and the RoboTwin dataset [7,23], which focuses on bimanual coordination tasks. For LIBERO, we include four subsets—*LIBERO-Spatial*, *LIBERO-Goal*, *LIBERO-Object*, and *LIBERO-10*—and perform 25 independent rollouts per task. For RoboTwin, we evaluate on seven manipulation

Table 2: Performance comparison using GR00T N1.5 on the LIBERO benchmark. Best results are highlighted in **bold**.

Task Suite		LIB-Spatial	LIB-Object	LIB-Goal	LIB-10
Static Oracle	$e = 1$	92.7 ± 0.9	94.7 ± 3.4	82.7 ± 0.9	74.7 ± 3.4
	$e = 2$	96.0 ± 1.6	95.3 ± 3.8	82.0 ± 1.6	83.3 ± 0.9
	$e = 4$	94.7 ± 3.4	95.3 ± 2.5	90.7 ± 0.9	88.7 ± 1.9
	$e = 8$	95.3 ± 0.9	97.3 ± 0.9	94.7 ± 3.4	86.0 ± 5.9
	$e = 12$	91.3 ± 2.5	96.0 ± 0.0	90.0 ± 0.0	86.0 ± 4.3
	$e = 16$	94.0 ± 3.3	94.7 ± 0.9	90.7 ± 0.9	90.0 ± 1.6
Static Oracle+		96.0 ± 1.6	97.3 ± 0.9	94.7 ± 3.4	90.0 ± 1.6
Random		93.3 ± 0.9	96.0 ± 2.8	92.0 ± 1.6	88.7 ± 1.9
AutoHorizon		**96.7 ± 0.9**	**98.7 ± 1.8**	**96.0 ± 1.6**	**92.7 ± 2.5**

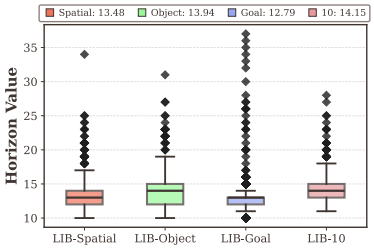

Fig. 5: Estimated execution horizon distribution. The legend displays the mean values of the horizon distributions.

tasks: *adjust bottle position*, *pick dual bottles*, *place container onto plate*, *stack two bowls*, *place empty cup on coaster*, *open laptop*, and *press stapler*. Each task is executed for 100 trials. All experiments are repeated under different random seeds to account for stochasticity in action sampling and environment initialization, ensuring statistical robustness.

LIBERO Benchmark. Tab. 1 reports the results of $\pi_{0.5}$ on the LIBERO benchmark. Under a small prediction horizon ($p = 10$), the optimal execution horizon for the *Static Oracle* baseline typically appears at the upper bound of valid values. The *Random* baseline also performs relatively well, suggesting that models trained with short prediction horizons tend to overfit and accurately capture short trajectory segments. When the prediction horizon increases to $p = 50$, we observe a significantly different behavior. The performance of *Static Oracle* first rises and then declines as the execution horizon extends, while the *Random* baseline suffers a pronounced drop in performance. In many cases, suboptimal horizon choices even cause *Static Oracle* to underperform the *Random* baseline, underscoring the importance of selecting an appropriate execution horizon. The enhanced *Static Oracle+* consistently achieves strong results, and the specific horizon values used for this baseline are listed in the supplementary material. Across both horizon configurations, AutoHorizon consistently outperforms all baselines. We attribute this improvement to its ability to dynamically adapt execution horizons during rollout, effectively balancing long-term consistency with short-term reactivity.

We further evaluate AutoHorizon on the LIBERO benchmark using GR00T N1.5 [24], which differs from $\pi_{0.5}$ in both architectural design and training pipeline. As shown in Tab. 2, although the precise performance trends vary, *Static Oracle* again exhibits a characteristic peak followed by degradation as the execution horizon increases. Our method consistently achieves superior results, demonstrating robustness and generalization across different architectures and training regimes. In the supplementary material we also visualize the attention weight distributions for GR00T N1.5, which yield conclusions consistent with our earlier analysis.

Table 3: Performance comparison using $\pi_{0.5}$ on the RoboTwin tasks. Best results are highlighted in **bold**.

Task Suite		Adjust Bottle	Pick Bottles	Place Container	Stack Bowls	Place Cup	Open Laptop	Press Stapler
Static Oracle	$e = 0.2p$	79.0 ± 1.4	40.7 ± 0.5	84.0 ± 2.8	90.7 ± 1.2	83.7 ± 0.9	68.3 ± 1.9	44.0 ± 0.0
	$e = 0.4p$	89.0 ± 0.0	67.0 ± 0.0	83.0 ± 0.0	87.3 ± 1.7	77.7 ± 2.5	78.3 ± 0.5	48.0 ± 0.0
	$e = 0.6p$	89.3 ± 0.9	65.0 ± 0.0	82.0 ± 0.0	86.7 ± 2.5	68.3 ± 0.5	71.7 ± 2.1	67.0 ± 0.0
	$e = 0.8p$	76.7 ± 1.2	58.0 ± 1.4	81.0 ± 0.0	90.0 ± 2.8	66.0 ± 1.6	78.7 ± 0.5	70.0 ± 0.0
	$e = 1.0p$	58.7 ± 1.2	30.0 ± 0.0	76.7 ± 0.5	87.7 ± 1.2	56.3 ± 3.8	84.0 ± 0.0	71.0 ± 0.0
Static Oracle+		98.7 ± 0.5	67.0 ± 0.0	91.0 ± 0.8	90.7 ± 1.2	83.7 ± 0.9	84.0 ± 0.0	72.0 ± 0.0
Random		85.3 ± 0.5	60.0 ± 0.0	86.0 ± 0.0	88.7 ± 3.4	70.7 ± 2.5	82.0 ± 0.0	69.0 ± 0.0
AutoHorizon		**100.0 ± 0.0**	**68.0 ± 0.0**	**91.0 ± 0.8**	**92.0 ± 0.8**	**85.3 ± 2.1**	**84.7 ± 0.9**	**75.0 ± 0.0**

RoboTwin Benchmark. Tab. 3 presents the results on the RoboTwin benchmark across tasks with varying difficulty. For both tasks that are highly sensitive (*pick bottles*) and those that are relatively insensitive (*stack bowls*) to the choice of execution horizon, our method achieves comparable or superior performance to the baselines, demonstrating its adaptability across diverse task dynamics.

Ablation. Fig. 5 visualizes the value distribution of execution horizons estimated by AutoHorizon across the four LIBERO task suites. AutoHorizon yields a broad range of horizon lengths during rollout, demonstrating adaptability to diverse input conditions and capturing the frequent shifts in VLA's prediction dynamics. Most estimated horizons fall within moderately low values—favoring reactivity—while occasional larger horizons facilitate faster task completion when long-term consistency is beneficial. We further compare with a variation of *Static Oracle* using fixed horizons closest to AutoHorizon's mean estimated values (*e.g.*, $e = 14$ and $e = 15$ for LIBERO-10) in the supplementary material, and find that AutoHorizon consistently achieves higher success rates. These results confirm the effectiveness of dynamic horizon adjustment over fixed-horizon strategies.

We also examine the effect of hyperparameters in the supplementary material. The results show that AutoHorizon's performance remains stable across different combinations of parameter settings. Compared with the strong *Static Oracle+* baseline, it always achieves comparable or even superior results, demonstrating robustness to hyperparameter choices.

4.3 Real-World Results

Setup. We further evaluate AutoHorizon in real-world robotic manipulation scenarios. Experiments are conducted on a Franka Research 3 robot (7-DoF arm) [10] following the DROID experimental setup [15]. The backbone VLA is $\pi_{0.5}$, configured with a prediction horizon of $p = 50$.

We assess all methods on three single-arm pick-and-place tasks of increasing difficulty: *put cucumber on plate*, *put Rubik's cube on plate*, and *put Rubik's cube into bowl*. A total of 150 trajectories are collected for fine-tuning. For evaluation, we adopt a stage-based solve rate that measures progress across four phases: (1) reaching the object, (2) grasping and lifting it, (3) moving it toward the target location, and (4) successfully placing it in the container. Each task is evaluated

over ten trials per setting, with each trial capped at 300 control steps, amounting to approximately three hours of total robot execution time. Object positions and orientations are randomized across trials to ensure robustness and generalization.

Results. Tab. 4 summarizes the performance across the three tasks. The occasionally large standard deviations arise from the binary nature of the outcomes, where some rollouts successfully complete the task, while others fail entirely. Several key observations emerge from the execution process. When the execution horizon is too short (*i.e.*, $e \in [1, 5]$), the robot frequently hesitates or stalls during motion. This behavior stems from the policy's tendency to predict subtle, low-amplitude movements for the initial few actions within a chunk, resulting in insufficient overall progression. At moderate horizons ($e \in [20, 40]$), the robot often overreaches or collides with the workspace, reflecting a loss of reactivity. When the execution horizon becomes excessively long ($e > 40$), the robot struggles to maintain accurate object localization, leading to frequent object drops and a diminished ability to correct errors during execution.

Table 4: Real-world task performance comparison. Best results in **bold**.

Task Suite		Cucumber→Plate	Cube→Plate	Cube→Bowl
Static Oracle	$e = 5$	91.5 ± 12.7	0.0 ± 0.0	76.0 ± 40.7
	$e = 10$	94.0 ± 11.5	81.5 ± 35.4	97.5 ± 4.2
	$e = 20$	88.0 ± 20.3	54.0 ± 43.9	89.5 ± 17.6
	$e = 30$	89.0 ± 19.0	74.5 ± 36.3	87.5 ± 17.0
	$e = 40$	79.0 ± 24.5	81.5 ± 28.7	56.5 ± 31.3
	$e = 50$	50.0 ± 40.8	51.5 ± 38.6	58.5 ± 34.0
Static Oracle+		97.0 ± 7.9	81.5 ± 35.4	97.5 ± 4.2
Random		88.5 ± 14.7	60.5 ± 44.1	77.0 ± 27.3
AutoHorizon		**98.0 ± 4.8**	**92.0 ± 15.7**	**99.0 ± 2.1**

In contrast, AutoHorizon dynamically adjusts the execution horizon throughout the rollout. As shown in Fig. 2, under stable conditions—such as reaching for or transporting the object—the estimated horizons increase, accelerating execution progress. When the robot begins to physically interact with the environment (*e.g.*, grasping or placing the cube), the estimated horizons adaptively shorten, enhancing reactivity to environmental changes. More video demonstrations of the experiments are provided in the supplementary materials.

5 Conclusion

Determining the execution horizon in action chunking flow policies remains an underexplored yet crucial challenge. In this work, we analyze the predictive behaviors of flow-based VLAs through attention weight inspection. Our analysis reveals that predicted action chunks exhibit limited temporal adaptability and consistently rely on radial action sinks for structural guidance. Building on these insights, we interpret action self-attention weights as implicit indicators of the model's predictive confidence and propose an autonomous, attention-guided execution horizon estimation algorithm that dynamically assigns chunk-specific horizons. Extensive evaluations in both simulated and real-world robotic manipulation tasks demonstrate that our method consistently outperforms other baselines, highlighting its effectiveness and generalizability.

Acknowledgements

This research is supported by NSF IIS-2525840, CNS-2432534, ECCS-2514574 and Cisco Research unrestricted gift. This article solely reflects opinions and conclusions of authors and not funding agencies.

References

1. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robotics and Autonomous Systems **57**(5), 469–483 (2009). https://doi.org/https://doi.org/10.1016/j.robot.2008.10.024, https://www.sciencedirect.com/science/article/pii/S0921889008001772
2. Beyer, L., Steiner, A., Pinto, A.S., Kolesnikov, A., Wang, X., Salz, D., Neumann, M., Alabdulmohsin, I., Tschannen, M., Bugliarello, E., Unterthiner, T., Keysers, D., Koppula, S., Liu, F., Grycner, A., Gritsenko, A., Houlsby, N., Kumar, M., Rong, K., Eisenschlos, J., Kabra, R., Bauer, M., Bošnjak, M., Chen, X., Minderer, M., Voigtlaender, P., Bica, I., Balazevic, I., Puigcerver, J., Papalampidi, P., Henaff, O., Xiong, X., Soricut, R., Harmsen, J., Zhai, X.: Paligemma: A versatile 3b vlm for transfer (2024), https://arxiv.org/abs/2407.07726
3. Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., Jakubczak, S., Jones, T., Ke, L., Levine, S., Li-Bell, A., Mothukuri, M., Nair, S., Pertsch, K., Shi, L.X., Tanner, J., Vuong, Q., Walling, A., Wang, H., Zhilinsky, U.: π_0: A vision-language-action flow model for general robot control (2024), https://arxiv.org/abs/2410.24164
4. Black, K., Galliker, M.Y., Levine, S.: Real-time execution of action chunking flow policies (2025), https://arxiv.org/abs/2506.07339
5. Cheang, C.L., Chen, G., Jing, Y., Kong, T., Li, H., Li, Y., Liu, Y., Wu, H., Xu, J., Yang, Y., Zhang, H., Zhu, M.: Gr-2: A generative video-language-action model with web-scale knowledge for robot manipulation (2024), https://arxiv.org/abs/2410.06158
6. Chen, G., Li, Z., Wang, S., Jiang, J., Liu, Y., Lu, L., Huang, D.A., Byeon, W., Le, M., Rintamaki, T., Poon, T., Ehrlich, M., Rintamaki, T., Poon, T., Lu, T., Wang, L., Catanzaro, B., Kautz, J., Tao, A., Yu, Z., Liu, G.: Eagle 2.5: Boosting long-context post-training for frontier vision-language models (2025), https://arxiv.org/abs/2504.15271
7. Chen, T., Chen, Z., Chen, B., Cai, Z., Liu, Y., Li, Z., Liang, Q., Lin, X., Ge, Y., Gu, Z., et al.: Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. arXiv preprint arXiv:2506.18088 (2025)
8. Chi, C., Xu, Z., Feng, S., Cousineau, E., Du, Y., Burchfiel, B., Tedrake, R., Song, S.: Diffusion policy: Visuomotor policy learning via action diffusion (2024), https://arxiv.org/abs/2303.04137
9. Gu, X., Pang, T., Du, C., Liu, Q., Zhang, F., Du, C., Wang, Y., Lin, M.: When attention sink emerges in language models: An empirical view. arXiv preprint arXiv:2410.10781 (2024)
10. Haddadin, S.: The franka emika robot: A standard platform in robotics research. IEEE Robotics & Automation Magazine **31**(4), 136–148 (2024). https://doi.org/10.1109/MRA.2024.3451788

11. Hu, Y., Guo, Y., Wang, P., Chen, X., Wang, Y.J., Zhang, J., Sreenath, K., Lu, C., Chen, J.: Video prediction policy: A generalist robot policy with predictive visual representations (2025), https://arxiv.org/abs/2412.14803
12. Intelligence, P., Black, K., Brown, N., Darpinian, J., Dhabalia, K., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Galliker, M.Y., Ghosh, D., Groom, L., Hausman, K., Ichter, B., Jakubczak, S., Jones, T., Ke, L., LeBlanc, D., Levine, S., Li-Bell, A., Mothukuri, M., Nair, S., Pertsch, K., Ren, A.Z., Shi, L.X., Smith, L., Springenberg, J.T., Stachowicz, K., Tanner, J., Vuong, Q., Walke, H., Walling, A., Wang, H., Yu, L., Zhilinsky, U.: $\pi_{0.5}$: a vision-language-action model with open-world generalization (2025), https://arxiv.org/abs/2504.16054
13. Kang, S., Kim, J., Kim, J., Hwang, S.J.: See what you are told: Visual attention sink in large multimodal models (2025), https://arxiv.org/abs/2503.03321
14. Kawaharazuka, K., Oh, J., Yamada, J., Posner, I., Zhu, Y.: Vision-language-action models for robotics: A review towards real-world applications. IEEE Access **13**, 162467–162504 (2025). https://doi.org/10.1109/ACCESS.2025.3609980
15. Khazatsky, A., Pertsch, K., Nair, S., Balakrishna, A., Dasari, S., Karamcheti, S., Nasiriany, S., Srirama, M.K., Chen, L.Y., Ellis, K., Fagan, P.D., Hejna, J., Itkina, M., Lepert, M., Ma, Y.J., Miller, P.T., Wu, J., Belkhale, S., Dass, S., Ha, H., Jain, A., Lee, A., Lee, Y., Memmel, M., Park, S., Radosavovic, I., Wang, K., Zhan, A., Black, K., Chi, C., Hatch, K.B., Lin, S., Lu, J., Mercat, J., Rehman, A., Sanketi, P.R., Sharma, A., Simpson, C., Vuong, Q., Walke, H.R., Wulfe, B., Xiao, T., Yang, J.H., Yavary, A., Zhao, T.Z., Agia, C., Baijal, R., Castro, M.G., Chen, D., Chen, Q., Chung, T., Drake, J., Foster, E.P., Gao, J., Guizilini, V., Herrera, D.A., Heo, M., Hsu, K., Hu, J., Irshad, M.Z., Jackson, D., Le, C., Li, Y., Lin, K., Lin, R., Ma, Z., Maddukuri, A., Mirchandani, S., Morton, D., Nguyen, T., O'Neill, A., Scalise, R., Seale, D., Son, V., Tian, S., Tran, E., Wang, A.E., Wu, Y., Xie, A., Yang, J., Yin, P., Zhang, Y., Bastani, O., Berseth, G., Bohg, J., Goldberg, K., Gupta, A., Gupta, A., Jayaraman, D., Lim, J.J., Malik, J., Martín-Martín, R., Ramamoorthy, S., Sadigh, D., Song, S., Wu, J., Yip, M.C., Zhu, Y., Kollar, T., Levine, S., Finn, C.: Droid: A large-scale in-the-wild robot manipulation dataset (2025), https://arxiv.org/abs/2403.12945
16. Kim, M.J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E., Lam, G., Sanketi, P., Vuong, Q., Kollar, T., Burchfiel, B., Tedrake, R., Sadigh, D., Levine, S., Liang, P., Finn, C.: Openvla: An open-source vision-language-action model (2024), https://arxiv.org/abs/2406.09246
17. Lai, L., Huang, A.Z., Gershman, S.J.: Action chunking as policy compression. PsyArXiv (2022)
18. Li, C., Wen, J., Peng, Y., Peng, Y., Feng, F., Zhu, Y.: Pointvla: Injecting the 3d world into vision-language-action models. arXiv preprint arXiv:2503.07511 (2025)
19. Li, S., Gao, Y., Sadigh, D., Song, S.: Unified video action model (2025), https://arxiv.org/abs/2503.00200
20. Liu, B., Zhu, Y., Gao, C., Feng, Y., Liu, Q., Zhu, Y., Stone, P.: Libero: Benchmarking knowledge transfer for lifelong robot learning (2023), https://arxiv.org/abs/2306.03310
21. Liu, Y., Hamid, J.I., Xie, A., Lee, Y., Du, M., Finn, C.: Bidirectional decoding: Improving action chunking via closed-loop resampling. International Conference on Learning Representations (ICLR) (2025), https://arxiv.org/abs/2408.17355
22. Ma, Y., Song, Z., Zhuang, Y., Hao, J., King, I.: A survey on vision-language-action models for embodied ai (2025), https://arxiv.org/abs/2405.14093

23. Mu, Y., Chen, T., Chen, Z., Peng, S., Lan, Z., Gao, Z., Liang, Z., Yu, Q., Zou, Y., Xu, M., Lin, L., Xie, Z., Ding, M., Luo, P.: Robotwin: Dual-arm robot benchmark with generative digital twins. In: Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR). pp. 27649–27660 (June 2025)
24. NVIDIA, :, Bjorck, J., Castañeda, F., Cherniadev, N., Da, X., Ding, R., Fan, L.J., Fang, Y., Fox, D., Hu, F., Huang, S., Jang, J., Jiang, Z., Kautz, J., Kundalia, K., Lao, L., Li, Z., Lin, Z., Lin, K., Liu, G., Llontop, E., Magne, L., Mandlekar, A., Narayan, A., Nasiriany, S., Reed, S., Tan, Y.L., Wang, G., Wang, Z., Wang, J., Wang, Q., Xiang, J., Xie, Y., Xu, Y., Xu, Z., Ye, S., Yu, Z., Zhang, A., Zhang, H., Zhao, Y., Zheng, R., Zhu, Y.: Gr00t n1: An open foundation model for generalist humanoid robots (2025), https://arxiv.org/abs/2503.14734
25. OpenAI: Gpt-4 technical report (2024), https://arxiv.org/abs/2303.08774
26. Shi, L.X., Ichter, B., Equi, M., Ke, L., Pertsch, K., Vuong, Q., Tanner, J., Walling, A., Wang, H., Fusai, N., Li-Bell, A., Driess, D., Groom, L., Levine, S., Finn, C.: Hi robot: Open-ended instruction following with hierarchical vision-language-action models (2025), https://arxiv.org/abs/2502.19417
27. Shukor, M., Aubakirova, D., Capuano, F., Kooijmans, P., Palma, S., Zouitine, A., Aractingi, M., Pascal, C., Russi, M., Marafioti, A., Alibert, S., Cord, M., Wolf, T., Cadene, R.: Smolvla: A vision-language-action model for affordable and efficient robotics (2025), https://arxiv.org/abs/2506.01844
28. Song, J., Vahdat, A., Mardani, M., Kautz, J.: Pseudoinverse-guided diffusion models for inverse problems. In: International Conference on Learning Representations (2023), https://api.semanticscholar.org/CorpusID:259298715
29. Team, G.: Gemini: A family of highly capable multimodal models (2025), https://arxiv.org/abs/2312.11805
30. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need (2023), https://arxiv.org/abs/1706.03762
31. Wen, J., Zhu, Y., Li, J., Zhu, M., Tang, Z., Wu, K., Xu, Z., Liu, N., Cheng, R., Shen, C., et al.: Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. IEEE Robotics and Automation Letters (2025)
32. Xiao, G., Tian, Y., Chen, B., Han, S., Lewis, M.: Efficient streaming language models with attention sinks (2024), https://arxiv.org/abs/2309.17453
33. Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., Qiu, Z.: Qwen3 technical report (2025), https://arxiv.org/abs/2505.09388
34. Zare, M., Kebria, P.M., Khosravi, A., Nahavandi, S.: A survey of imitation learning: Algorithms, recent developments, and challenges (2023), https://arxiv.org/abs/2309.02473
35. Ze, Y., Zhang, G., Zhang, K., Hu, C., Wang, M., Xu, H.: 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations (2024), https://arxiv.org/abs/2403.03954
36. Zhang, W., Liu, H., Qi, Z., Wang, Y., Yu, X., Zhang, J., Dong, R., He, J., Lu, F., Wang, H., et al.: Dreamvla: a vision-language-action model dreamed with comprehensive world knowledge. arXiv preprint arXiv:2507.04447 (2025)

37. Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Ré, C., Barrett, C., Wang, Z., Chen, B.: H_2o: Heavy-hitter oracle for efficient generative inference of large language models (2023), https://arxiv.org/abs/2306.14048
38. Zhao, Q., Lu, Y., Kim, M.J., Fu, Z., Zhang, Z., Wu, Y., Li, Z., Ma, Q., Han, S., Finn, C., et al.: Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 1702–1713 (2025)
39. Zhao, T.Z., Kumar, V., Levine, S., Finn, C.: Learning fine-grained bimanual manipulation with low-cost hardware (2023), https://arxiv.org/abs/2304.13705
40. Zhen, H., Qiu, X., Chen, P., Yang, J., Yan, X., Du, Y., Hong, Y., Gan, C.: 3d-vla: A 3d vision-language-action generative world model. arXiv preprint arXiv:2403.09631 (2024)
41. Zhu, J., Wang, W., Chen, Z., Liu, Z., Ye, S., Gu, L., Tian, H., Duan, Y., Su, W., Shao, J., Gao, Z., Cui, E., Wang, X., Cao, Y., Liu, Y., Wei, X., Zhang, H., Wang, H., Xu, W., Li, H., Wang, J., Deng, N., Li, S., He, Y., Jiang, T., Luo, J., Wang, Y., He, C., Shi, B., Zhang, X., Shao, W., He, J., Xiong, Y., Qu, W., Sun, P., Jiao, P., Lv, H., Wu, L., Zhang, K., Deng, H., Ge, J., Chen, K., Wang, L., Dou, M., Lu, L., Zhu, X., Lu, T., Lin, D., Qiao, Y., Dai, J., Wang, W.: Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models (2025), https://arxiv.org/abs/2504.10479
42. Zitkovich, B., Yu, T., Xu, S., Xu, P., Xiao, T., Xia, F., Wu, J., Wohlhart, P., Welker, S., Wahid, A., Vuong, Q., Vanhoucke, V., Tran, H., Soricut, R., Singh, A., Singh, J., Sermanet, P., Sanketi, P.R., Salazar, G., Ryoo, M.S., Reymann, K., Rao, K., Pertsch, K., Mordatch, I., Michalewski, H., Lu, Y., Levine, S., Lee, L., Lee, T.W.E., Leal, I., Kuang, Y., Kalashnikov, D., Julian, R., Joshi, N.J., Irpan, A., Ichter, B., Hsu, J., Herzog, A., Hausman, K., Gopalakrishnan, K., Fu, C., Florence, P., Finn, C., Dubey, K.A., Driess, D., Ding, T., Choromanski, K.M., Chen, X., Chebotar, Y., Carbajal, J., Brown, N., Brohan, A., Arenas, M.G., Han, K.: Rt-2: Vision-language-action models transfer web knowledge to robotic control. In: Tan, J., Toussaint, M., Darvish, K. (eds.) Proceedings of The 7th Conference on Robot Learning. Proceedings of Machine Learning Research, vol. 229, pp. 2165–2183. PMLR (06–09 Nov 2023), https://proceedings.mlr.press/v229/zitkovich23a.html