

# CEED-VLA : Consistency Vision-Language-Action Model with Early-Exit Decoding

Wenxuan Song<sup>1\*</sup>, Jiayi Chen<sup>1\*</sup>, Pengxiang Ding<sup>2,3†</sup>,  
Yuxin Huang<sup>1</sup>, Han Zhao<sup>2,3</sup>, Donglin Wang<sup>2</sup>, Haoang Li<sup>1‡</sup>  
<sup>1</sup>HKUST(GZ) <sup>2</sup>Westlake University <sup>3</sup>Zhejiang University

## Abstract

In recent years, Vision-Language-Action (VLA) models have become a vital research direction in robotics due to their impressive multimodal understanding and generalization capabilities. Despite the progress, their practical deployment is severely constrained by inference speed bottlenecks, particularly in high-frequency and dexterous manipulation tasks. While recent studies have explored Jacobi decoding as a more efficient alternative to traditional autoregressive decoding, its practical benefits are marginal due to the lengthy iterations. To address it, we introduce consistency distillation training to predict multiple correct action tokens in each iteration, thereby achieving acceleration. Besides, we design mixed-label supervision to mitigate the error accumulation during distillation. Although distillation brings acceptable speedup, we identify that certain inefficient iterations remain a critical bottleneck. To tackle this, we propose an early-exit decoding strategy that moderately relaxes convergence conditions, which further improves average inference efficiency. Experimental results show that the proposed method achieves more than 4× inference acceleration across different baselines while maintaining high task success rates in both simulated and real-world robot tasks. These experiments validate that our approach provides an efficient and general paradigm for accelerating multimodal decision-making in robotics. Our project page is available at <https://irpn-eai.github.io/CEED-VLA/>.

## 1 Introduction

Recent advancements in Vision-Language Models (VLMs) [1, 2] have showcased impressive multimodal understanding capabilities, inspiring the development of Vision-Language-Action (VLA) models [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]. These end-to-end architectures, which are trained on large-scale robotic datasets [15, 16, 17, 18], integrate visual perception and language understanding to directly generate executable actions. Although VLA models generalize well across diverse tasks, their practical deployment is severely limited by inference speed bottlenecks, which hinder efficient execution and the handling of high-frequency, dexterous tasks. Therefore, our goal is to **significantly improve inference efficiency of VLAs while retaining the manipulation performance**.

To achieve this goal, recent works [13, 19] innovatively reframe autoregressive (AR) decoding as a system of nonlinear equations solved through the **Jacobi fixed-point iteration method** [20]. Specifically, the Jacobi decoding method begins by randomly initializing the  $n$  action tokens in a sequence. Then, the action sequence along with the prompt is then iteratively processed by the VLA to refine its predictions. Through  $k$  successive updates, the  $n$ -token sequence converges to the fixed point, which is the same as the output produced by AR decoding under a greedy strategy. Because Jacobi decoding is parallel and allows the model to predict several correct tokens in each iteration,

\* Equal contribution † Project Leader ‡ Corresponding author: haoangli@hkust-gz.edu.cn

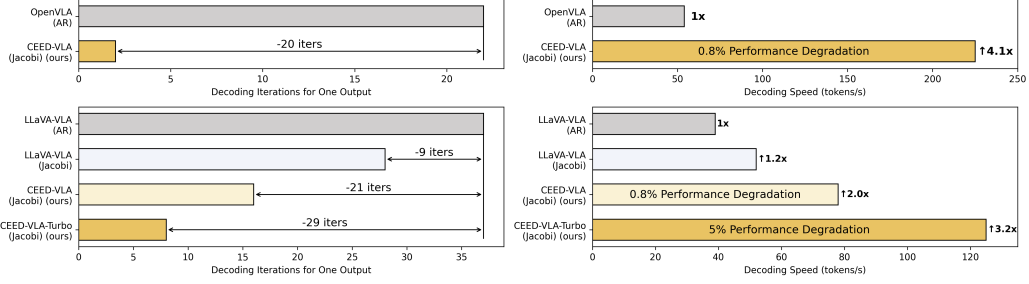


Figure 1: **Acceleration effect of CEED-VLA on OpenVLA and LLaVA-VLA.** **Left:** Comparison of the number of iterations required for a complete output. CEED-VLA largely reduces the iterations, thus allowing faster decoding. **Right:** Comparison of the decoding speed. The speedup of directly running Jacobi decoding in a vanilla VLA is marginal. Our CEED-VLA separately realizes  $3.6\times$  and  $2.0\times$  speedup with negligible performance degradation on OpenVLA and LLaVA-VLA. In scenarios targeting more aggressive acceleration, CEED-VLA-Turbo delivers even fewer iterations and much more speedup while incurring only a slight degradation in performance.

the number of iterations in Jacobi decoding can be fewer than the forward passes in AR decoding, i.e.,  $k \leq n$ . This indicates that VLAs with Jacobi decoding can be faster than AR ones theoretically.

However, in practice, Jacobi decoding on standard VLAs [13] delivers only limited acceleration over AR decoding, with recent studies reporting a modest  $1.28\times$  speedup. This limitation arises because VLAs are trained in a strictly autoregressive manner, where models are only exposed to ground-truth prefixes during training. As a result, when preceding tokens are incorrect, which is a natural condition in Jacobi decoding, the model struggles to generate accurate predictions. Due to the lack of this capability, the model usually only predicts the first token correctly in each parallel iteration, resulting in slower convergence of the full token sequence, as shown in Figure 8.

To address this issue, we propose a **consistency distillation** process in Figure 2, which enables the student model to predict several correct tokens in one iteration through the training objective of mapping arbitrary points along the Jacobi trajectory of the teacher model directly to the fixed point. Nonetheless, two key challenges must be addressed: 1. Relying solely on consistency objectives may undermine the model’s inherent autoregressive generation capability. 2. Error accumulated in the distillation process can result in unreliable supervision signals. To mitigate these challenges, we propose a dual-supervision framework. First, we introduce an auxiliary AR loss to preserve the student model’s native manipulation skills by aligning next-token distributions with those of the teacher model, thereby regularizing the distillation process. Second, we devise a mixed-label supervision mechanism that adaptively combines two sources of supervision. When the teacher model’s action accuracy exceeds a learned threshold, its output is used as the supervisory signal. Otherwise, the target is replaced with the ground-truth value. This adaptive strategy ensures robust student learning by selectively leveraging reliable teacher signals while mitigating the impact of performance degradation during distillation.

After consistency distillation, the student model demonstrates acceptable acceleration effect, while the ideal acceleration is still bottlenecked by certain **inefficient iterations**. Due to the strict convergence conditions of Jacobi decoding, these inefficient iterations often require a large number of iterations to reach convergence, significantly reducing the average decoding speed. To address this issue, we propose an **early-exit decoding** strategy, which relaxes the strict convergence conditions of Jacobi decoding and thereby circumvents the impact of inefficient iterations. Moreover, our analysis of the structural properties of the task and empirical observation shows that early-exit decoding has minimal effect on success rates, thus serving as a key accelerator. Our training approach closely resembles consistency models [21], as both aim to accelerate inference by directly mapping intermediate equation states to the final solution. Thus, we term our model as **Consistency Vision-Language-Action model with Early-Exit Decoding (CEED-VLA)**. A variant with a smaller exit point is termed as CEED-VLA-Turbo, featuring even faster inference.

Experiments in two simulated environments show that CEED-VLA realizes **2-4.1** $\times$  acceleration with comparable success rates across different baselines (Table 2). Real-world experiments show that CEED-VLA realize **4** $\times$  frequency in the real robotic arm deployment with improved success rates on high-frequency dexterous tasks. We provide a straightforward visualization and detailed analyses,

further revealing a key acceleration phenomenon. We also conducted extensive ablation studies to demonstrate the effectiveness of our key designs and to analyze the impact of training data size. To summarize, our key technical contributions are as follows:

- We propose CEED-VLA, a universal acceleration method for significant inference speedup while maintaining manipulation performance.
- We conduct a consistency distillation process to unlock the model’s capabilities of fast inference, and we further propose a mixed-label supervision in the autoregressive loss to preserve the model’s manipulation performance.
- We identify the inefficient iteration as the bottleneck of Jacobi decoding’s speedup and propose the early-exit decoding to solve it, resulting in  $4.1\times$  speedup, and more than  $4.3\times$  frequency.

## 2 Related Work

**Acceleration for Vision-Language-Action Models.** Various acceleration strategies, including quantization [22] and token pruning [23], have been effectively applied to LLMs, yet they often fail to meet the stringent real-time requirements of action generation. Efforts to enhance efficiency have led to architectural modifications in VLA models, such as DeeR-VLA [24], which dynamically adjusts inference depth, and QAIL [25], which integrates quantization-aware training. Further innovations, like RoboMamba [26] and TinyVLA [27], replace traditional attention mechanisms or focus on developing lightweight models from the ground up, frequently necessitating model re-training and additional data collection. Meanwhile, VLA-Cache [28] selectively caches static tokens and recomputes only dynamic or task-relevant ones. FAST [29] proposes a compression-based tokenization scheme based on the discrete cosine transform. MoLe-VLA [30] achieves this by selectively skipping transformer layers based on task-relevant cues. PD-VLA [13] framework reformulates autoregressive decoding as a nonlinear system and solves it using a parallel fixed-point iteration method, significantly improving decoding speed while maintaining model performance. OpenVLA-OFT [19] employs a similar parallel decoding method to speed up. In contrast, our CEED-VLA unlocks the acceleration potential of VLAs by fine-tuning them with a consistency distillation process. It also optimizes the decoding mechanisms, leading to more significant acceleration.

**Consistency Models for Manipulation.** In manipulation tasks, consistency policy [31] serves as a key acceleration technique for diffusion policies, which employs consistency models [21] in image generation fields to robotics. It enables single-step inference in a student model while analyzing the impact of design choices like objectives, variance, and chain steps on consistency distillation. ManiCM [32] applies consistency constraints to the diffusion process to enable fast inference without compromising action quality. FlowPolicy [33] enables single-step generation by normalizing velocity field consistency, refining flow dynamics for efficient inference. SDM policy [34] integrates score and distribution matching through a dual teacher framework to improve the speed of inference and the quality of action. While these consistency policies directly map intermediate states of ODEs to their final solution, we train VLAs to map the intermediate points in the Jacobi trajectory to the fixed point. This paper takes the first step to explore consistency training techniques for efficient VLAs.

## 3 Preliminary: Jacobi Decoding

Given a prompt  $x$  and a pre-trained LLM  $p(\cdot|x)$ , we typically predict tokens using the standard AR decoding method with greedy strategies:

$$y_i = \arg \max_y p(y|\mathcal{Y}_{i-1}, x) \text{ for } i = 1, \dots, n, \quad (1)$$

where  $\mathcal{Y}_{i-1}$  denotes  $\{y_1, \dots, y_{i-1}\}$ ,  $n$  represents the number of tokens to predict. Thus, LLM executes  $n$  forward passes to obtain  $n$  tokens  $\mathcal{Y}_n$ , which makes it hard to efficiently output a lengthy token sequence.

Unlike AR decoding, Jacobi decoding [35, 36] can accelerate the inference by predicting several tokens in one forward. To directly predict several tokens in each forward, Equation (1) is reformulated

as a system of nonlinear equations with respect to  $y_i$ :

$$\begin{cases} y_1^{(j+1)} &= \arg \max_y p(y|\mathbf{x}) \\ y_2^{(j+1)} &= \arg \max_y p(y|\mathcal{Y}_1^{(j)}, \mathbf{x}) \\ &\vdots \\ y_n^{(j+1)} &= \arg \max_y p(y|\mathcal{Y}_{n-1}^{(j)}, \mathbf{x}), \end{cases} \quad (2)$$

where  $j$  denotes the  $j$ -iteration of the Jacobi trajectory. This enables updates of every token in each forward iteration until convergence. The nonlinear equation system can be solved in the Jacobi fix-point iteration method [20]. Concretely, Jacobi decoding first initializes a random token sequence  $\mathcal{Y}^{(0)} = \{y_1^{(0)}, \dots, y_n^{(0)}\}$ . Then, both the prompt  $x$  and the token sequence are fed into the LLM simultaneously. Then, the variables  $y_i$  in  $\mathcal{Y}$  are iteratively updated through Equation (2) until convergence. The convergence condition is  $\mathcal{Y}^{(k)} = \mathcal{Y}^{(k-1)}$  at the step  $k$ . The  $\mathcal{Y}^* := \mathcal{Y}^{(k)}$  is defined as the **fixed point**. Because Jacobi decoding allows to predict multiple correct tokens in the  $n$ -token sequence in parallel, it requires fewer iterations than AR decoding, thus improving the inference speed.

## 4 Method

### 4.1 Teacher Model

Vanilla VLAs  $P_\theta$  learn to predict actions  $\hat{a}_{t+1}$  directly from the current observation  $s_t$  and language instruction  $l$  (Figure 2):

$$\hat{A}_t \sim P_\theta(a_t | s_t, l) \quad (3)$$

To further improve the planning abilities and stability of actions, the VLAs are combined with action chunking [37]. Specifically, at the current time step  $t$ , given chunk size  $m$ , the predicted actions will be extended into an action sequences  $A_t = [a_{t+1}, a_{t+2}, \dots, a_{t+m}]$ . Each  $a_t$  consists of 7 dimensions, which is formulated as:  $a_t = [X, Y, Z, \phi, \theta, \psi, G]$ , where  $X, Y, Z$  represent the Cartesian coordinates of the end effector’s position,  $\phi, \theta, \psi$  denote the rotation angles of the end effector along each axis, and  $G$  is the gripper state. In this paper, we specify 2 vanilla VLA models, the LLaVA-VLA used in [13, 19] and OpenVLA [8], as our teacher models.

Directly replacing the AR decoding in these VLAs with Jacobi decoding really accelerates the inference, while the speedup is **limited** ( $1.28\times$  in [13]). The reason is that VLAs, having been trained autoregressively, struggle to produce multiple correct tokens within a single Jacobi iteration: each newly generated token is conditioned on the previously generated ones, so when any preceding token is incorrect, the model is unlikely to generate the correct subsequent token.

### 4.2 Student Model

To address the aforementioned issue and fully unlock the acceleration potential, we finetune VLAs to empower them to output multiple correct actions with wrong preceding tokens. This section first illustrates the data collection for tuning CEED-VLA and then details the training procedure. Finally, we propose an efficient decoding strategy to further accelerate inference.

#### 4.2.1 Jacobi Trajectory Collection

For the target VLA  $P$ , we let  $Q_\theta(\cdot|\mathbf{x})$  denotes the CEED-VLA with parameters  $\theta$  initialized with those of  $P$ . To capture the inherent consistency within Jacobi trajectories, we first collect them by prompting  $P$  to predict actions with Jacobi decoding on the robot dataset  $\mathcal{C}$ . We record the Jacobi trajectories during decoding to build the consistency distillation dataset  $\mathcal{D}$ . The dataset construction process is detailed in Algorithm 2.

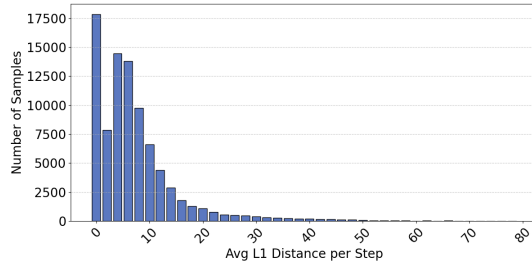


Figure 3: L1 distance between the generated Jacobi trajectory dataset and the ground-truth data.

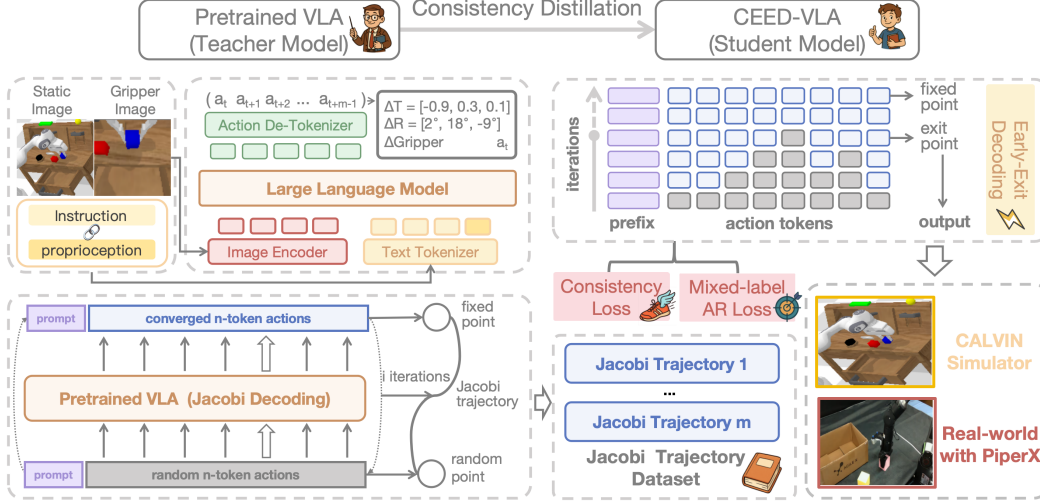


Figure 2: Overview of our proposed **CEED-VLA**. Our proposed framework first runs the pretrained VLA (e.g., LLaVA-VLA) with Jacobi decoding to generate the training dataset. Then we design an effective consistency distillation process with novel mixed-label supervision to get the student model. Finally, we propose early-exit decoding to further unlock inference speed. Experiments in simulators and the real world show significant acceleration with comparative success rates.

#### 4.2.2 Consistency Training

The consistency training procedure optimizes two objectives: (i) guiding the model to predict multiple correct tokens simultaneously, and (ii) constraining CEED-VLA from drifting away from the target VLA distribution to preserve manipulation skills.

**Consistency Loss.** The ideal finetuned model consistently maps any point  $\mathcal{Y}$  on the Jacobi trajectory  $\mathcal{J}$  to the fixed point  $\mathcal{Y}^*$ . We design the consistency loss based on this notion. Given a prefix  $x$  consisting of both instructions and visual inputs, and its Jacobi trajectory  $\mathcal{J}$ , let  $\mathcal{Y}$  be a randomly sampled intermediate state and  $\mathcal{Y}^*$  the corresponding fixed point. We train CEED-VLA to directly predict  $\mathcal{Y}^*$  from  $\mathcal{Y}$  by minimizing the following loss:

$$\mathcal{L}_C = \mathbb{E}_{(x, \mathcal{J}) \sim \mathcal{D}, \mathcal{Y} \sim \mathcal{J}} \left[ \sum_{i=1}^n \text{KL} (Q_{\theta^-}(\cdot | \mathcal{Y}_i^*, x) || Q_{\theta}(\cdot | \mathcal{Y}_i, x)) \right] \quad (4)$$

where  $\theta^- = \text{stopgrad}(\theta)$ .  $\text{KL}(\cdot || \cdot)$  denotes the forward KL divergence between two distributions.

**Mixed-label AR Supervision.** To avoid deviating from the distribution of the teacher model, we inherit the AR loss function  $\mathcal{L}_{AR}$  used for training the teacher model:

$$\mathcal{L}_{AR} = \mathbb{E}_{(x, \mathcal{Y}^*) \sim \mathcal{D}} \left[ - \sum_{i=1}^N \log Q_{\theta}(\mathcal{Y}_i^* | \mathcal{Y}_{<i}^*, x) \right]. \quad (5)$$

To address potential performance degradation stemming from teacher model inaccuracies in action prediction, we implement a mixed-label supervision strategy. Specifically, we compute the L1 distance to quantify the distributional divergence between the generated Jacobi trajectory dataset and the corresponding ground-truth action data. For high-deviation samples, we replace the AR loss labels of those samples with the corresponding ground-truth values. We define a correctness threshold  $\delta_{\max}$ , and any data in the dataset  $\mathcal{D}$  with an L1 distance exceeding this threshold is considered an outlier. This strategy effectively mitigates error propagation

#### Algorithm 1 Mixed-label Training

- 1: **Input:** Original robot dataset  $\mathcal{C}$ , Jacobi trajectory dataset  $\mathcal{D}$ , weight  $\omega$ , CEED-VLA  $Q_{\theta}(\cdot | x)$ , correctness threshold  $\delta_{\max}$
- 2: **repeat**
- 3:   Sample prefix  $x$ , Jacobi trajectory  $\mathcal{J}$  from  $\mathcal{D}$ , teacher model's output  $\mathcal{Y}^*$  from  $\mathcal{J}$ , ground-truth label GT from  $\mathcal{C}$
- 4:   **if**  $L1(\mathcal{Y}^*, \text{GT}) < \delta_{\max}$  **then**
- 5:     Compute  $\mathcal{L}_{AR}$  using eq. (5)
- 6:   **else**
- 7:     Reformulate eq. (5) as  $\mathcal{L}_{AR} = \mathbb{E}_{(x, \text{GT})}$
- 8:   **end if**
- 9:   Randomly sample  $\mathcal{Y}$  from  $\mathcal{J}$
- 10:   Compute  $\mathcal{L}_C$  using eq. (4)
- 11:   update parameters  $\theta$  using  $\mathcal{L}$  (eq. (6))
- 12: **until** convergence

during consistency distillation, maintaining generation quality substantially. Consequently, the total loss for training a CEED-VLA is:

$$\mathcal{L}(\theta) = \mathcal{L}_C + \omega \mathcal{L}_{AR} \quad (6)$$

where  $\omega$  represents a weighting coefficient. The training procedure is detailed in Algorithm 1.

### 4.2.3 Inference

**Inefficient iterations.** The student model CEED-VLA performs inference via Jacobi decoding, as illustrated in Section 3. However, our empirical observations reveal that the acceleration performance of CEED-VLA is bottlenecked by a few iterations that are even slower than the minimum speed of AR decoding (see Table 1). This phenomenon arises from the **strict convergence conditions** of the fixed point. It requires exact convergence between successive iterations, *i.e.*,  $\mathcal{Y}^{(k)} = \mathcal{Y}^{(k-1)}$ , which takes numerous iterations (*over 30 iterations*) to reach. It is further observed that token updates in the final steps of inefficient iterations are marginal, contributing little to the final action decisions. These inefficient iterations significantly hinder inference speedup and may introduce delays or discontinuities in high-frequency dexterous tasks, ultimately compromising task performance.

**Early-exit Decoding.** To address this limitation, we investigate whether relaxed convergence conditions or fewer iterations are feasible, particularly in terms of their impact on overall performance. We base our investigation on two aspects: 1. **Theoretical structural properties.** Unlike popular VQA tasks (*e.g.* multiple-choice), where each inference has a direct correctness, the success of a robotic manipulation task is decided on a sequence of actions. Within it, an **overlooked** yet critical insight is that task success is primarily determined by actions executed at a small number of key states, *e.g.*, releasing the gripper at the target location. In contrast, the majority of states along a trajectory do not demand optimal actions, or the set of “good-enough” actions is relatively large [38, 39]. Therefore, slight degradation in the quality of certain actions does not adversely affect task outcomes. 2. **Empirical observation.** We further observe that token updates in the final steps of inefficient iterations are marginal and contribute little to the final action decisions, further highlighting their inefficiency. This suggests that relaxing convergence and reducing iterations introduce only minor degradation in action quality. Based on the above analysis and empirical observations, we conclude that relaxing convergence conditions and reducing iterations are feasible without compromising performance.

To this end, we propose **early-exit decoding** to relax the condition and significantly reduce the number of iterations. Specifically, we introduce the exit point  $\sigma$ , at which the model prematurely halts the iterative process and directly emits the intermediate output at the  $\sigma$ -th iteration, bypassing further decoding steps. This approach mitigates inefficient iterations, significantly enhancing both the minimum and average decoding speeds while maintaining performance. We further conduct ablation studies to investigate the optimal value of the exit point, see Section 5.2. When the exit point is set to a small value, the model achieves even higher speedup with an acceptable performance drop, termed as CEED-VLA-Turbo.

Table 1: Comparison of inference speeds among AR decoding, Jacobi decoding, and early-exit decoding on CEED-VLA.

Decoding Method	Avg. Speed	Min. Speed	Max. Speed
AR	39.64	30.87	49.10
Jacobi	57.57 $\uparrow$	19.50 $\downarrow$	92.64 $\uparrow$
Early-exit	79.24 $\uparrow$	62.07 $\uparrow$	93.53 $\sim$



Figure 4: **An instance of Jacobi trajectory with early-exit decoding.** Gray numbers indicate incorrect tokens, while blue numbers denote correct ones. Blue numbers with underlines represent fixed tokens. The three rows from bottom to top illustrate the Jacobi trajectory, starting from the initialized point and ending at the exit point. The topmost row represents the Jacobi fixed point.

## 5 Experiments

In this section, we evaluate the proposed CEED-VLA in terms of its ability to accelerate performance while maintaining manipulation performance. We structure the experiments to answer the following questions:

Table 2: Comparison with various manipulation baselines in terms of inference speed, success rates or average length, and execution frequency. All experiments based on LLaVA-VLA are tested on a NVIDIA 4090 GPU and experiments based on OpenVLA are tested on a NVIDIA H100 GPU.

Acceleration Techniques	Action Chunking	Decoding Method	Splits	Speed (Tokens/s)	Avg. Len. SR (%)	Frequency (Hz)
Base model: <b>LLaVA-VLA</b> Benchmark: <b>CALVIN</b>						
N/A	-	AR	ABC→D	39.6 ( $\times 1$ )	2.01	2.23 ( $\times 1$ )
N/A	5	AR	ABC→D	39.6 ( $\times 1$ )	<b>3.70</b>	4.33 ( $\times 1.9$ )
FastV	5	AR	ABC→D	28.7 ( $\times 0.7$ )	2.54	1.87 ( $\times 0.8$ )
SparseVLM	5	AR	ABC→D	32.4 ( $\times 0.8$ )	2.83	2.01 ( $\times 0.9$ )
PD-VLA	5	Jacobi	ABC→D	52.8 ( $\times 1.3$ )	3.69	5.43 ( $\times 2.4$ )
CEED-VLA (ours)	5	Jacobi	ABC→D	<b>79.2</b> ( $\times 2.0$ )	3.67	<b>7.27</b> ( $\times 3.3$ )
Base model: <b>OpenVLA</b> Benchmark: <b>LIBERO</b>						
N/A	-	AR	Long	54.4 ( $\times 1$ )	53.2	5.95 ( $\times 1$ )
N/A	3	AR	Long	54.4 ( $\times 1$ )	60.4	7.08 ( $\times 1$ )
PD-VLA	3	Jacobi	Long	85.0 ( $\times 1.6$ )	<b>62.4</b>	10.79 ( $\times 2.4$ )
CEED-VLA (ours)	3	Jacobi	Long	<b>225.0</b> ( $\times 4.1$ )	62.2	<b>25.60</b> ( $\times 4.3$ )

- Can CEED-VLA serve as a general framework that achieves significant acceleration across different models, tasks, and deployment environments, while maintaining high performance? (see Section 5.1)
- What **underlying phenomena** contribute to the observed acceleration effects introduced by CEED-VLA? (see Section 5.1)
- How do the **key design choices** and **data amounts** in CEED-VLA influence the overall model performance and stability? (see Section 5.2)
- Can CEED-VLA effectively increase the execution frequency on **real-world** robotic platforms, thereby improving the success rate of dexterous manipulation tasks? (see Section 5.3)

## 5.1 Simulation Experiments

**Benchmarks and Metrics.** To evaluate the success rates and inference speedup of our CEED-VLA, we carefully selected two widely used simulation benchmarks in the robot learning field and VLA tasks for comprehensive experiments. The **CALVIN** benchmark [40] includes 34 tasks across four environments (A, B, C, and D). Following the classic CALVIN ABC→D setup, we run 500 rollouts per model and report success rates and average sequential completions. **LIBERO-Long** [41] consists of 10 long-horizon and multi-step manipulation tasks with diverse objects and skills, emphasizing temporal reasoning, goal consistency and delayed rewards. We evaluate each method across 50 rollouts with varying initial states per task and report both per-task and average success rates.

**Baselines.** In this section, we consider fine-tuning diverse baseline models for a comprehensive validation of our CEED-VLA. **LLaVA-VLA** is a vanilla VLA model, based on LLaVA [42], detailed in Appendix C. We also include **OpenVLA** [8], the most popular open-source VLA model, for general evaluation. We further reproduce an OpenVLA with action chunking, which enables more stable actions.

**Acceleration Results in CALVIN.** We compare our method with naïve baselines, advanced acceleration methods for VLMs (FastV [23], SparseVLM [43]), and VLAs in parallel decoding (PD-VLA). As shown in Table 2, the key findings on CALVIN are as follows: 1. Only Jacobi decoding without consistency training leads to a modest improvement in VLA decoding speed. 2. With mixed-label consistency training and early-exit decoding, CEED-VLA is able to predict more fixed points, resulting in a  $3.2\times$  speedup,  $4.4\times$  frequencies while the average length only reduces 0.03.

**Acceleration Results in LIBERO.** Experiments on the LIBERO-Long benchmark further validate our conclusions. Our proposed CEED-VLA realizes  $3.6\times$  speed up and  $4.1\times$  frequency, which is more evident than CALVIN. This is because the tasks in LIBERO are relatively easy and short-



Table 3: Ablation of different techniques used in our CEED-VLA on the base model OpenVLA(action chunk=3).

Model	Modifications				Speedup	Accuracy(%)
	Jacobi Decoding	Consistency Training	Mixed-Label Supervision	Early-exit Decoding		
CEED-VLA	✓	✓	✓	✓	×4.1	62.2
-	✓	✓	✓	×	×2.5	61.2
-	✓	✓	×	×	×2.3	54.2
PD-VLA	✓	×	×	×	×1.6	62.3
OpenVLA	×	×	×	×	×1.0	60.4

horizon, thus requiring fewer iterations in Jacobi decoding. Meanwhile, the integration with action chunking improves 15.8% success rates because of stronger planning abilities and smoother actions.

**Discuss: Why does OpenVLA achieve more significant acceleration than LLaVA-VLA?** OpenVLA achieves more than 4× acceleration, in contrast to LLaVA-VLA’s 2× speedup. We hypothesize that this difference stems from the nature of the evaluation tasks: CALVIN features more complex, long-horizon scenarios composed of five subtasks, which demand more careful planning during execution, thereby increasing the number of required iterative steps.

#### Visualization of Jacobi trajectory with early-exit decoding.

We randomly visualize a Jacobi trajectory during testing in Figure 4. The trajectory starts from a randomly initialized point and iteratively updates until reaching the predefined exit point, where the final output is generated. Notably, the value at the exit point does not exactly match the conventional Jacobi fixed point, indicating that the model terminates earlier before full convergence. The early-exit mechanism further reduces the number of iterations and contributes to the observed acceleration.

Table 4: Profiling results for the number of fixed tokens and accelerations.

Model	Num. of <i>fixed token</i>	Speedup
LLaVA-VLA	0	1×
PD-VLA	8.75	1.33×
CEED-VLA	13.5	<b>2.00×</b>

**Underlying Acceleration Phenomena.** In Figure 4, we observe that our CEED-VLA is capable of correctly predicting certain action tokens (underlined blue numbers) in one iteration, even if preceding tokens are incorrect. We refer to such tokens as *fixed tokens*. The presence of fixed tokens significantly enhances the model’s ability to predict multiple tokens within each iteration. Table 4 presents the correlation between the number of fixed tokens and the resulting acceleration gains.

## 5.2 Ablation Experiments

In this section, we first evaluate the significance of several key designs in our method on the base of OpenVLA. As illustrated in Table 3, consistency training and early-exit decoding substantially accelerate the model, whereas mixed-label supervision plays a crucial role in maintaining action accuracy and success rate.

Then we investigate the choice of different exit points, data amounts, supervising labels, and losses in turn. The main model is fine-tuned on LLaVA-VLA with an exit point of 16, 960k training trajectories, mixed-label supervision, and a loss ratio of 1.

**Early-exit Decoding.** Early-exit decoding serves as a key accelerator, balancing inference speed and success rates. In Figure 5, we conduct inference on different values  $\sigma$  of exit points. AR decoding requires 37 iterations to complete a single action. We observe that introducing early-exit decoding and setting  $\sigma$  to 16, *i.e.*, a maximum of 16 iterations, leads to a noticeable speedup with negligible performance drop. This suggests that enabling early exit at a few steps within a task does not affect overall success. Reducing the  $\sigma$  further to 8 results in a slight performance degradation but brings even greater

Table 5: Comparison of performance between different supervising labels for AR loss.

Label	Speedup	Avg. Len.
Mixed (ours)	<b>2.00×</b>	<b>3.67</b>
Teacher Model	1.83×	3.48
GT	1.67×	3.20



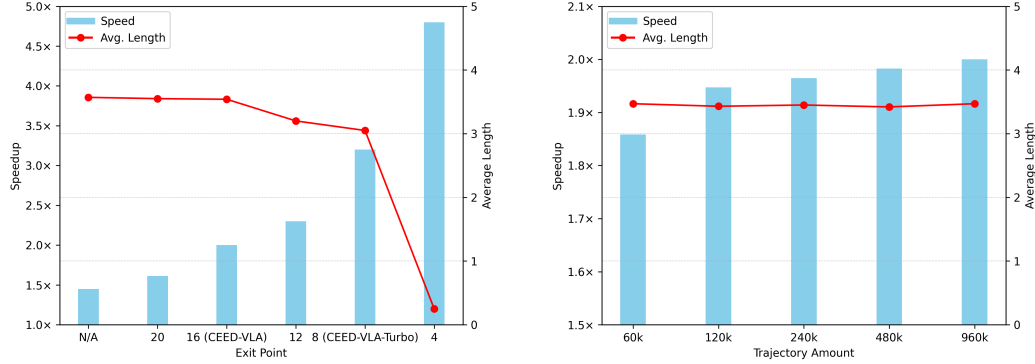


Figure 5: Speedup and average length of CEED-VLA decoding with different values of exit point (left) and trained with different data amounts (right). On the left, our CEED-VLA employs an exit point of 16, and the extremely accelerated version CEED-VLA-Turbo exits at 8.

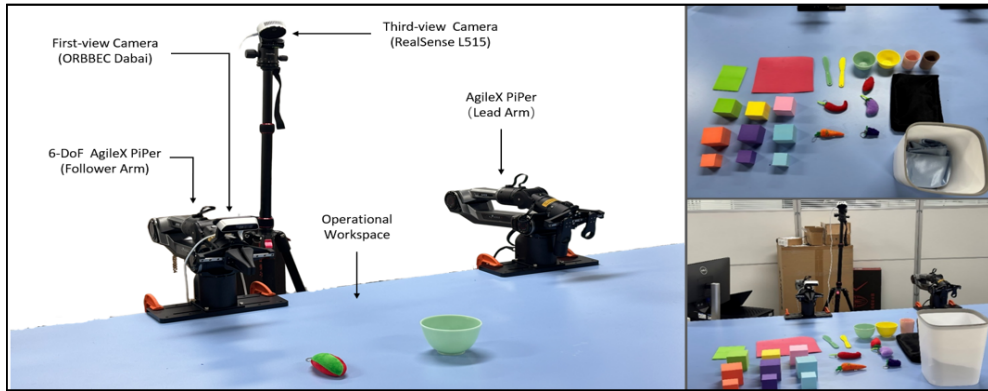


Figure 6: **The real-world robotic system for experiments.** The system consists of two AgileX PiPer 6-DoF robotic arms, an ORBBEC Dabai depth camera, and a RealSense L515 depth camera.

acceleration. This version is termed as CEED-VLA-Turbo. However, lowering the exit point to 4 yields extreme speedup at the cost of substantial action, making task success difficult.

**Mixed-label Supervision.** We investigate the impact of label choices for supervising the AR loss. As shown in Table 5, our mixed-label supervision best balances between speed-up and performance. This is because it effectively anchors the student model between the teacher output and the ground truth, preventing drift from either side. Using only the teacher output as the label causes error accumulation, resulting in shorter average episode lengths. Conversely, using only the ground truth creates conflicting supervision between the AR and consistency losses, yielding the worst performance.

**Loss Design.** The consistency loss guides the model to learn the convergence behavior of the Jacobian trajectory toward a fixed point, thereby reducing the iterations and accelerating the overall decoding process. The autoregressive (AR) loss encourages the model to learn correct actions, preventing it from deviating from the distribution of teacher model and ensuring performance. The trade-off between speed and accuracy is influenced by the relative weighting of the two losses. We experimented with weight ratios of 1 and 10, respectively. As shown in Table 6, increasing the emphasis on the AR loss does indeed improve accuracy, albeit at the cost of speedup.

Table 6: Comparison of inference speed and average length between different ratios of losses to evaluate the trade-off.

Loss	Speedup	Avg. Len.
$\mathcal{L}_C + \mathcal{L}_{AR}(\text{ours})$	<b>2.00×</b>	3.67
$\mathcal{L}_C + 10 * \mathcal{L}_{AR}$	1.79×	<b>3.74</b>

**Data Size.** CEED-VLA learns to predict multiple tokens in each step by leveraging pre-collected Jacobian trajectories. As shown in Figure 5, 60k trajectories are sufficient to achieve significant acceleration, demonstrating strong data efficiency because training for one epoch on this dataset takes only 10 hours. When the number of trajectories exceeds 120k, the performance gain plateaus and becomes marginal. Another observation is that increasing the data volume has a limited impact on the average episode length.

Table 7: Real-world success rate comparison with LLaVA-VLA. We evaluate all the methods with 4 (tasks)  $\times$  20 (variants) rollouts. Our method achieves better performance among all tasks than baselines.

Method	Push button (basic)	Lift block (basic)	Pour water (dexterous)	Fold towel (dexterous)	Avg. success rate (%)	Frequency (Hz)
LLaVA-VLA	65	40	15	5	33.25	3.3
PD-VLA	85	65	45	35	57.50	6.0 (1.8 $\times$ )
CEED-VLA (ours)	<b>85</b>	<b>70</b>	<b>80</b>	<b>75</b>	<b>77.50</b>	<b>13.0</b> (3.9 $\times$ )

### 5.3 Real-world Experiments

**Real-world Setup.** Our real-world experiments were performed on a dual-arm AgileX PiPer robotic system. During the data collection phase, one arm was teleoperated by a human to provide demonstrations (designated as the lead arm), while the other arm (designated as the follower arm) remained passive. During evaluation, only the follower arm was controlled by VLAs to perform the tasks. To provide visual observations, we employ a RealSense L515 depth camera as Third-view and an ORBBEC Dabai depth camera as First-view input.

**Tasks and Datasets.** Our dataset covers three levels of manipulation difficulty, categorized into basic Tasks and dexterous Tasks, as described below. Basic Tasks are short-horizon, atomic interactions such as button pushing and block lifting. These tasks focus on simple object interactions under basic planning and are collected at a frequency of 10 Hz. Dexterous Tasks demand high-frequency, continuous control, and fine-grained manipulation skills. These tasks include towel folding and water pouring, which are collected at 30Hz.

**Results.** Table 7 presents the real-world results. We observe that LLaVA-VLA achieves decent success rates on basic tasks, but struggles to learn effective policies with high-frequency robot data. Its discontinuous actions often lead to task failures. PD-VLA improves action continuity and execution frequency by integrating parallel decoding with action chunking, leading to higher success rates on both basic and dexterous tasks. Our CEED-VLA significantly boosts inference speed and control frequency, enabling the model to learn and execute high-frequency actions. As a result, it substantially improves success rates on dexterous tasks, exceeding 70%.

## 6 Conclusions

In this paper, we propose CEED-VLA, a universal acceleration method for significant inference speedup while maintaining manipulation performance. We propose a mixed-label supervision during consistency training and early-exit decoding during inference. Extensive experiments in several simulation and real-world environments demonstrate that our CEED-VLA significantly improves the inference speed and execution frequency, while maintaining comparative manipulation performance and better managing high-frequency dexterous tasks.

## 7 Future work

Recent works investigate more efficient tokenization schemes for robot actions in autoregressive VLAs [29, 44]. They allow representing  $x_2$  action components with  $x_1$  tokens where  $x_1 < x_2$ , thereby improving the efficiency of model training and inference, as well as the capture of high-frequency actions. Combining our method with these works to further achieve real-time inference is a promising direction. Our method also demonstrates potential to drive progress across embodied chain-of-thought [45] that produce extensive intermediate reasoning tokens and knowledge insulating [46] that require simultaneous generation of both linguistic responses and action tokens.

## References

- [1] A. Awadalla, I. Gao, J. Gardner, J. Hessel, Y. Hanafy, W. Zhu, K. Marathe, Y. Bitton, S. Gadre, S. Sagawa, J. Jitsev, S. Kornblith, P. W. Koh, G. Ilharco, M. Wortsman, and L. Schmidt, “Openflamingo: An open-source framework for training large autoregressive vision-language models,” *arXiv preprint arXiv:2308.01390*, 2023.
- [2] H. Liu, C. Li, Q. Wu, and Y. J. Lee, “Visual instruction tuning,” *Advances in neural information processing systems*, vol. 36, 2024.
- [3] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, *et al.*, “Rt-1: Robotics transformer for real-world control at scale,” *Proceedings of Robotics: Science and Systems*, 2023.
- [4] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid *et al.*, “Rt-2: Vision-language-action models transfer web knowledge to robotic control,” in *Conference on Robot Learning*. PMLR, 2023, pp. 2165–2183.
- [5] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine, “Octo: An open-source generalist robot policy,” in *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.
- [6] D. Niu, Y. Sharma, G. Biamby, J. Quenum, Y. Bai, B. Shi, T. Darrell, and R. Herzig, “Llarva: Vision-action instruction tuning enhances robot learning,” *arXiv preprint arXiv:2406.11815*, 2024.
- [7] W. Song, H. Zhao, P. Ding, C. Cui, S. Lyu, Y. Fan, and D. Wang, “Germ: A generalist robotic model with mixture-of-experts for quadruped robot,” in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 11 879–11 886.
- [8] M. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn, “Openvla: An open-source vision-language-action model,” *arXiv preprint arXiv:2406.09246*, 2024.
- [9] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang *et al.*, “Palm-e: An embodied multimodal language model,” 2023.
- [10] H. Zhen, X. Qiu, P. Chen, J. Yang, X. Yan, Y. Du, Y. Hong, and C. Gan, “3d-vla: A 3d vision-language-action generative world model,” *arXiv preprint arXiv:2403.09631*, 2024.
- [11] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang *et al.*, “Gr00t n1: An open foundation model for generalist humanoid robots,” *arXiv preprint arXiv:2503.14734*, 2025.
- [12] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter *et al.*, “ $\pi_0$ : A vision-language-action flow model for general robot control,” *arXiv preprint arXiv:2410.24164*, 2024.
- [13] W. Song, J. Chen, P. Ding, H. Zhao, W. Zhao, Z. Zhong, Z. Ge, J. Ma, and H. Li, “Accelerating vision-language-action model integrated with action chunking via parallel decoding,” *arXiv preprint arXiv:2503.02310*, 2025.
- [14] W. Song, J. Chen, W. Li, X. He, H. Zhao, P. D. S. Su, F. Tang, X. Cheng, D. Wang, Z. Ge *et al.*, “Rationalvla: A rational vision-language-action model with dual system,” *arXiv preprint arXiv:2506.10826*, 2025.
- [15] A. O’Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain *et al.*, “Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6892–6903.
- [16] H.-S. Fang, H. Fang, Z. Tang, J. Liu, C. Wang, J. Wang, H. Zhu, and C. Lu, “Rh20t: A comprehensive robotic dataset for learning diverse skills in one-shot,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 653–660.
- [17] N. M. M. Shafiullah, A. Rai, H. Etukuru, Y. Liu, I. Misra, S. Chintala, and L. Pinto, “On bringing robots home,” *arXiv preprint arXiv:2311.16098*, 2023.

- [18] H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, A. W. He, V. Myers, M. J. Kim, M. Du *et al.*, “Bridgedata v2: A dataset for robot learning at scale,” in *Conference on Robot Learning*. PMLR, 2023, pp. 1723–1736.
- [19] M. J. Kim, C. Finn, and P. Liang, “Fine-tuning vision-language-action models: Optimizing speed and success,” 2025. [Online]. Available: <https://arxiv.org/abs/2502.19645>
- [20] J. M. Ortega and W. C. Rheinboldt, *Iterative solution of nonlinear equations in several variables*. SIAM, 2000.
- [21] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever, “Consistency models,” 2023.
- [22] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han, “Awq: Activation-aware weight quantization for on-device llm compression and acceleration,” *Proceedings of Machine Learning and Systems*, vol. 6, pp. 87–100, 2024.
- [23] L. Chen, H. Zhao, T. Liu, S. Bai, J. Lin, C. Zhou, and B. Chang, “An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models,” in *European Conference on Computer Vision*. Springer, 2024, pp. 19–35.
- [24] Y. Yue, Y. Wang, B. Kang, Y. Han, S. Wang, S. Song, J. Feng, and G. Huang, “Deer-vla: Dynamic inference of multimodal large language models for efficient robot execution,” in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- [25] S. Park, H. Kim, W. Jeon, J. Yang, B. Jeon, Y. Oh, and J. Choi, “Quantization-aware imitation-learning for resource-efficient robotic control,” *arXiv preprint arXiv:2412.01034*, 2024.
- [26] J. Liu, M. Liu, Z. Wang, L. Lee, K. Zhou, P. An, S. Yang, R. Zhang, Y. Guo, and S. Zhang, “Robomamba: Multimodal state space model for efficient robot reasoning and manipulation,” *arXiv preprint arXiv:2406.04339*, 2024.
- [27] J. Wen, Y. Zhu, J. Li, M. Zhu, K. Wu, Z. Xu, N. Liu, R. Cheng, C. Shen, Y. Peng *et al.*, “Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation,” *arXiv preprint arXiv:2409.12514*, 2024.
- [28] S. Xu, Y. Wang, C. Xia, D. Zhu, T. Huang, and C. Xu, “Vla-cache: Towards efficient vision-language-action model via adaptive token caching in robotic manipulation,” 2025. [Online]. Available: <https://arxiv.org/abs/2502.02175>
- [29] K. Pertsch, K. Stachowicz, B. Ichter, D. Driess, S. Nair, Q. Vuong, O. Mees, C. Finn, and S. Levine, “Fast: Efficient action tokenization for vision-language-action models,” *arXiv preprint arXiv:2501.09747*, 2025.
- [30] R. Zhang, M. Dong, Y. Zhang, L. Heng, X. Chi, G. Dai, L. Du, D. Wang, Y. Du, and S. Zhang, “Mole-vla: Dynamic layer-skipping vision language action model via mixture-of-layers for efficient robot manipulation,” *arXiv preprint arXiv:2503.20384*, 2025.
- [31] A. Prasad, K. Lin, J. Wu, L. Zhou, and J. Bohg, “Consistency policy: Accelerated visuomotor policies via consistency distillation,” *Robotics: Science and Systems*, 2024.
- [32] G. Lu, Z. Gao, T. Chen, W. Dai, Z. Wang, and Y. Tang, “Manicm: Real-time 3d diffusion policy via consistency model for robotic manipulation,” *arXiv preprint arXiv:2406.01586*, 2024.
- [33] Q. Zhang, Z. Liu, H. Fan, G. Liu, B. Zeng, and S. Liu, “Flowpolicy: Enabling fast and robust 3d flow-based policy via consistency flow matching for robot manipulation,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 14, 2025, pp. 14 754–14 762.
- [34] B. Jia, P. Ding, C. Cui, M. Sun, P. Qian, S. Huang, Z. Fan, and D. Wang, “Score and distribution matching policy: Advanced accelerated visuomotor policies via matched distillation,” *arXiv preprint arXiv:2412.09265*, 2024.
- [35] A. Santilli, S. Severino, E. Postolache, V. Maiorca, M. Mancusi, R. Marin, and E. Rodola, “Accelerating transformer inference for translation via parallel decoding,” in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023, pp. 12 336–12 355.
- [36] S. Kou, L. Hu, Z. He, Z. Deng, and H. Zhang, “Cllms: Consistency large language models,” in *Forty-first International Conference on Machine Learning*, 2024.
- [37] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, “Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware,” in *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023.

- [38] A. Kumar, J. Hong, A. Singh, and S. Levine, “When should we prefer offline reinforcement learning over behavioral cloning?” *arXiv preprint arXiv:2204.05618*, 2022.
- [39] H. Zhao, W. Song, D. Wang, X. Tong, P. Ding, X. Cheng, and Z. Ge, “More: Unlocking scalability in reinforcement learning for quadruped vision-language-action models,” *arXiv preprint arXiv:2503.08007*, 2025.
- [40] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard, “Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks,” *IEEE Robotics and Automation Letters*, 2021.
- [41] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone, “Libero: Benchmarking knowledge transfer for lifelong robot learning,” *Advances in Neural Information Processing Systems*, vol. 36, 2024.
- [42] H. Liu, C. Li, Q. Wu *et al.*, “Visual instruction tuning,” 2023. [Online]. Available: <https://arxiv.org/abs/2304.08485>
- [43] Y. Zhang, C.-K. Fan, J. Ma, W. Zheng, T. Huang, K. Cheng, D. Gudovskiy, T. Okuno, Y. Nakata, K. Keutzer *et al.*, “Sparsevlm: Visual token sparsification for efficient vision-language model inference,” *arXiv preprint arXiv:2410.04417*, 2024.
- [44] S. Belkhale and D. Sadigh, “Minivla: A better vla with a smaller footprint,” 2024. [Online]. Available: <https://github.com/Stanford-ILIAD/openvla-mini>
- [45] M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine, “Robotic control via embodied chain-of-thought reasoning,” *arXiv preprint arXiv:2407.08693*, 2024.
- [46] D. Driess, J. T. Springenberg, B. Ichter, L. Yu, A. Li-Bell, K. Pertsch, A. Z. Ren, H. Walke, Q. Vuong, L. X. Shi *et al.*, “Knowledge insulating vision-language-action models: Train fast, run fast, generalize better,” *arXiv preprint arXiv:2505.23705*, 2025.
- [47] E. Coumans and Y. Bai, “Pybullet, a python module for physics simulation for games, robotics and machine learning,” <http://pybullet.org>, 2016–2019.
- [48] W. Zhao, P. Ding, M. Zhang, Z. Gong, S. Bai, H. Zhao, and D. Wang, “Vlas: Vision-language-action model with speech instructions for customized robot manipulation,” *International Conference on Learning Representations (ICLR)*, 2025.

## A Limitations

It easily applies to models with discrete actions, but its **limitation** is less suitable for continuous action spaces where convergence conditions are harder to define. We will explore more effective and unified convergence conditions in future work.

## B Generating Jacobi Trajectory Datasets

---

### Algorithm 2 Jacobi trajectory generation

---

```

1: Input: Dataset  $\mathcal{C} = \{(I, S, O)\}$  (image, instruction, proprioception), teacher model  $P$ .
2: repeat
3:   Sample  $(I, S, O)$  from dataset  $\mathcal{C}$ 
4:    $\mathbf{x}_{\text{text}} = [S, O]$ 
5:    $\mathbf{E}_{\text{img}} = \text{VisionTower}(\mathbf{I})$ 
6:    $\mathbf{E}_{\text{text}} = \text{Embed}(\text{Tokenizer}(\mathbf{x}_{\text{text}}))$ 
7:    $\mathbf{E}_{\text{all}} = [\mathbf{E}_{\text{img}}, \mathbf{E}_{\text{text}}]$ 
8:    $\mathcal{J} = \{\mathcal{Y}^0, \dots, \mathcal{Y}^*\} \leftarrow \text{Jacobi Decoding}(P, \mathbf{E}_{\text{all}})$ 
9:    $\mathbf{x} = (I, \mathbf{x}_{\text{text}})$ 
10:  Append  $(\mathbf{x}, \mathcal{J})$  to training set  $\mathcal{D}$ 
11: until all samples in  $\mathcal{C}$  are processed

```

---

## C Benchmark

We carefully selected two simulation benchmarks that are widely evaluated in robot learning fields and VLA tasks. The **CALVIN** benchmark [40] is built on top of the PyBullet [47] simulator and involves a Franka Panda Robot arm that manipulates the scene. CALVIN consists of 34 tasks and 4 different environments (A, B, C and D). We evaluate all methods on the classic **CALVIN ABCD**→**D** setup [40]. Each model is tested for 500 rollouts, and we report success rates and the average number of completed sequential tasks. **LIBERO-Long** [41] consists of long-horizon tasks, encompassing diverse object interactions and versatile motor skills. To evaluate a model’s ability in temporal reasoning and long-horizon planning, we introduce the **LIBERO-Long** benchmark. This suite consists of multi-step manipulation tasks that require agents to complete a sequence of subgoals in a coherent and temporally aligned manner. Unlike atomic tasks that focus on single-stage actions, LIBERO-Long challenges the model with increased task complexity, delayed rewards, and the need for consistent goal tracking. Each method is evaluated across 100 rollouts with varying initial states for each task. We report both per-task and average success rates.

## D Baseline

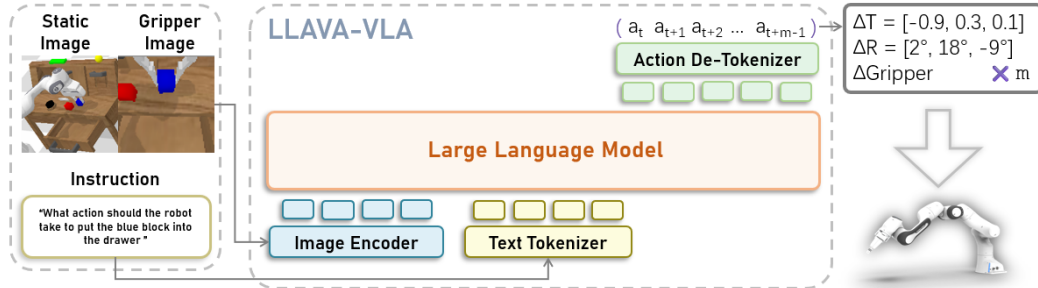


Figure 7: **LLaVA-VLA model architecture.** Given a sequence of images and language instructions, the input is first tokenized and fed into the language model. The model then generates action tokens, which are subsequently detokenized into executable action values and deployed on the robotic arm.

In this section, we consider fine-tuning diverse baseline models for a comprehensive validation of our CEED-VLA. **LLaVA-VLA** is a vanilla Vision-Language-Action (VLA) model with action

chunking at the output stage, derived by fine-tuning the widely adopted vision-language model LLaVA [42]. An overview of the model architecture is provided in Figure 7. It incorporates action chunking at the output stage to promote temporally coherent action generation. LLaVA-VLA exhibits stable performance across both simulated and real-world environments. It has also been adopted as a baseline in several prior works [48, 13]. We conduct most experiments based on LLaVA-VLA on the CALVIN ABC-D task to investigate the effectiveness of CEED-VLA.

**OpenVLA** [8], on the other hand, is the most widely used open-source VLA model. The model architecture builds upon an LLaMA 2 language model, augmented with a visual encoder that integrates pretrained features from both DINOv2 and SigLIP, enabling robust multimodal grounding in complex manipulation tasks. We further validate the general applicability of our approach by using OpenVLA as an additional baseline.

## E Adaptability of Our Proposed Method

Our method introduces a lightweight and practical solution that integrates seamlessly with existing vision-language models. It offers three distinct advantages. First, it is model redesign-free, meaning that no architectural changes to the backbone model are required, thereby preserving the integrity and performance characteristics of the original network. Second, it is modification-free as it does not require modifications or adding auxiliary components to pre-trained VLA models. Third, the method incurs no additional GPU memory overhead, making it highly resource-efficient and suitable for deployment in environments with constrained computational budgets. Together, these features make our approach both flexible and scalable, facilitating broad applicability without sacrificing efficiency or accuracy.

## F Visualization of the Decoding Process

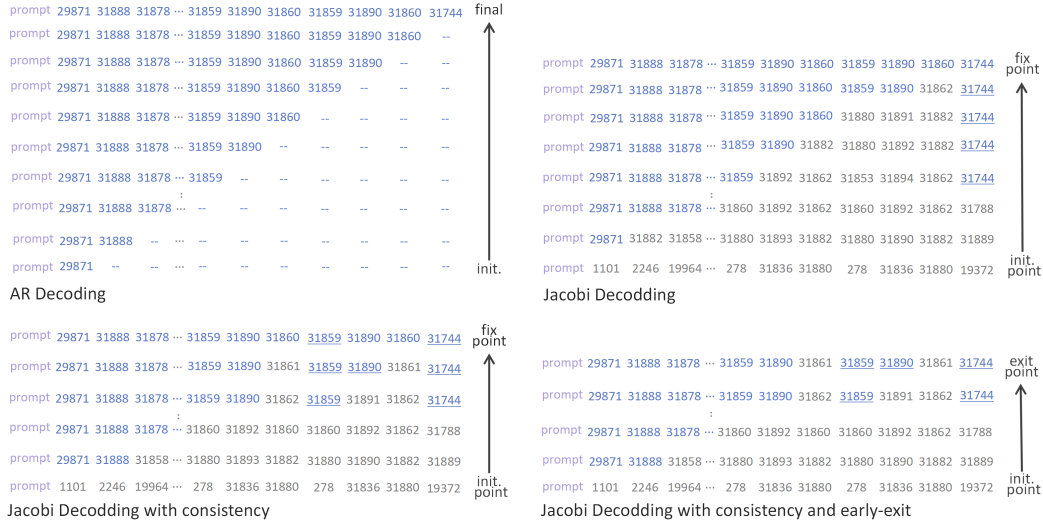


Figure 8: Comparison of four decoding methods

**AR Decoding.** As shown in the top-left corner of Figure 8, AR decoding is the conventional and widely adopted approach in sequence generation tasks. It generates tokens sequentially in a left-to-right manner, where each token depends strictly on all previously generated tokens. This method ensures stable performance and high output accuracy. However, its inherently sequential nature prevents parallelization, leading to slow inference and becoming a major bottleneck in real-time applications.

**Jacobi Decoding.** As shown in the top-right corner of Figure 8, Jacobi decoding draws inspiration from the classical Jacobi iterative method in numerical computation. It enables simultaneous updates of all tokens by iteratively refining predictions based on the outputs from the previous step. While this method successfully breaks the sequential dependency of autoregressive decoding and supports



Table 8: Evaluation of LLaVA-VLA on the CALVIN ABC→D benchmark, focusing on early-exit behavior. Metrics include speed and accuracy. Experiments are conducted on an NVIDIA RTX 4090 GPU.

Exit point	Decoding Method	Avg. Speed (Tokens/s)	Avg. Accuracy
<b>w/o Consistency Training</b>			
–	AR	39.6	3.50
no (pd-vla)	Jacobi	52.8	3.49
16	Jacobi	56.2	3.08
12	Jacobi	74.2	1.79
8	Jacobi	87.3	1.24
<b>with Consistency Training</b>			
no	Jacobi	57.42	3.49
16 (ceed-vla)	Jacobi	79.2	3.47
12	Jacobi	89.1	3.20
8 (ceed-vla-turbo)	Jacobi	126.72	3.04

parallelism, its convergence speed remains relatively slow. Without specialized training strategies, its acceleration potential is limited in practice.

**Jacobi Decoding with consistency training** . As shown in the Bottom-left corner of Figure 8, Consistency training is introduced to enhance the model’s ability to converge from arbitrary initial predictions toward a fixed point. By encouraging the prediction of more accurate tokens across iterations, it significantly reduces the number of steps required for convergence, thereby improving decoding efficiency. Despite this advancement, inference still suffers from inefficiency points, where certain tokens require more iterations to stabilize. These outlier tokens constrain the minimum achievable decoding latency, leaving room for further optimization.

**Early-exit Decoding with consistency training** . As shown in the Bottom-right corner of Figure 8, building on consistency training, the early-exit decoding with consistency training strategy allows token-wise termination once predictions converge or exhibit negligible change. Alternatively, a global iteration cap can be imposed to force early termination across all tokens. The empirical results shown in Figure 5, demonstrate that models trained with consistency loss maintain robust performance even under strict iteration limits. Notably, limiting the number of iterations at certain decoding steps does not significantly affect the overall task completion rate.

By striking an elegant balance between speed and performance, early-exit Jacobi decoding represents one of the most practical and efficient solutions for parallel inference.

## G Ablation experiment

Consistency training preserves the performance of early-exit decoding. As shown in Table. 8, we compare the average and average length of models with and without consistency training under different exit points. Consistency models exhibit more pronounced acceleration effects while maintaining high success rates even with early exit points. This stems from their ability to converge most action tokens with minimal iterations, whereas vanilla models fail to converge adequately under limited iterations, suffering significant performance degradation.