Towards More Efficient Decoding for Autoregressive Vision-language-action Models

Wenxuan Song[1*], Jiayi Chen[1*], Pengxiang Ding[2,3†], Yuxin Huang[1], Han Zhao[2,3], Yinchuan Li, Yingcong Chen[1], Donglin Wang[2], and Haoang Li[1(✉)]

[1] The Hong Kong University of Science and Technology (Guangzhou)
[2] Westlake University
[3] Zhejiang University
haoangli@hkust-gz.edu.cn

Abstract. The practical deployment of autoregressive (AR) Vision–Language–Action (VLA) models 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 this problem, we introduce consistency distillation to teach the model to predict multiple correct action tokens in each iteration, thereby reducing the total iterations. While the distillation brings moderate speedup, we identify that certain redundancy iterations remain a critical limitation. To tackle this, we propose an adaptive 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 base models 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 code and videos could be found in the supplementary files.

Keywords: Vision-language-action Model · Parallel Decoding · Model Acceleration

1 Introduction

Recent advancements in Vision-Language Models (VLMs) [1,2,15,20,32,34] have showcased impressive multimodal understanding capabilities, inspiring the development of Vision-Language-Action (VLA) models [4,5,11,13,16,31,48]. These end-to-end architectures are trained on large-scale robotic datasets, integrating

[*] Equal contribution. [†] Project leader. [✉] Corresponding author.

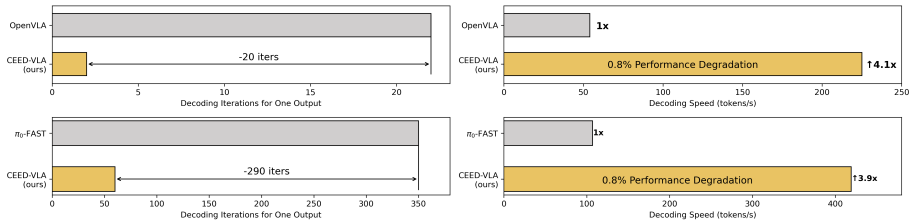

Fig. 1: Acceleration effect of our CEED-VLA on OpenVLA [16] (the first row) and π_0-FAST [28] (the second row). **Left**: Comparison of the number of iterations required for a complete output. CEED-VLA largely reduces the number of iterations, thus allowing faster decoding. **Right**: Comparison of the decoding speed. Our CEED-VLA separately realizes 3.6× and 3.9× speedup with negligible performance degradation on OpenVLA and π_0-FAST.

visual perception and language understanding to directly generate executable actions. While many recent models [3, 4, 14] adopt continuous flow-matching heads to enable fine-grained control, several mainstream approaches continue to rely on autoregressive (AR) architectures due to their superior training efficiency [28], stronger instruction-following capabilities [12, 28], and better unified understanding and generation [7, 36].

Although AR VLAs offer several advantages, 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 AR VLAs while retaining the manipulation performance**.

Jacobi decoding method [25] is promising to achieve this goal by reframing autoregressive (AR) decoding as a system of nonlinear equations solved to predict several correct tokens in parallel in each iteration. This brings fewer total iterations and thus realizes acceleration. However, in practice, Jacobi decoding on standard VLAs delivers only limited acceleration over AR decoding, with recent studies reporting a modest 1.28× speedup [30]. This limitation arises because VLAs are trained in a strictly AR 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 correctly predicts one token in each iteration in practice, resulting in slower convergence of the full token sequence.

To address this issue, we propose **C**onsistency **V**ision-**L**anguage-**A**ction model (**CEED-VLA**) with early-exit decoding. Specifically, the distillation process (See Figure 2) aims to teach the student model to predict several correct tokens in one iteration and shorten the decoding trajectory by learning from the decoding trajectory of the teacher model. The student model is designed to be self-consistent, directly mapping any intermediate decoding state to the final output and accordingly, we formulate a consistency objective. Thus, we term the process as **consistency distillation** process. While targeting the proposed

consistency objective, we still need to preserve the action precision with an auxiliary AR loss.

After consistency distillation, the student model demonstrates a moderate acceleration effect, while the ideal acceleration is still bottlenecked by certain **redundant iterations**. Due to the strict convergence conditions of Jacobi decoding, these redundant iterations often require a large number of iterations to reach convergence, significantly reducing the average decoding speed (shown in Table 1). To address this issue, we propose an **adaptive early-exit decoding** strategy, which relaxes the strict convergence conditions of Jacobi decoding by incorporating a confidence threshold. Specifically, if the minimum token confidence at an iteration exceeds the threshold, the iteration is terminated, and the intermediate results are directly used as the model's output. Our empirical observation shows that early-exit decoding has minimal effect on success rates, thus serving as a key accelerator.

We conduct experiments in two simulated benchmarks as well as real-world robotic arms. Figure 1 shows that CEED-VLA realizes up to **4.1**× acceleration with comparable success rates across different base models. Real-world experiments show that CEED-VLA realizes **4**× frequency in the practical robotic arm deployment with improved success rates on high-frequency dexterous tasks. We provide detailed analyses and extensive ablation studies to further reveal a key acceleration phenomenon and demonstrate the effectiveness of our key designs.

To summarize, our key technical contributions are as follows: 1) We propose CEED-VLA, a consistency distillation method for AR VLAs to unlock the acceleration advantages of Jacobi decoding. 2) We identify the redundant iterations as the bottleneck of CEED-VLA's speedup and propose the adaptive early-exit decoding based on the confidence score to solve it. 3) Extensive experiments on different baselines and tasks demonstrate that our approach achieves a substantial speed-up (up to 4×) and very slight performance drop.

2 Related Work

2.1 Vision-Language-Action Models

Vision-language-action (VLA) [5, 6, 9, 18, 22, 24, 42, 45, 48] models have demonstrated superior performance on perception and action prediction. Depending on the formulation of the output space, VLA models can be broadly categorized into two mainstream paradigms: those that rely on diffusion-based action experts to generate continuous outputs, and those that adopt action tokenization to produce discrete outputs.

Continuous outputs. To enable fine-grained continuous action control, $\pi 0$ [4] combines flow matching diffusion modeling with action chunking, replacing discrete tokenization and enabling the efficient generation of high-frequency, smooth action trajectories. $\pi 0.5$ [10, 14] further expanded this paradigm on mobile manipulation. More recently, GR00T N1 [3] advances general-purpose humanoid control by combining a vision-language module with a diffusion-based action

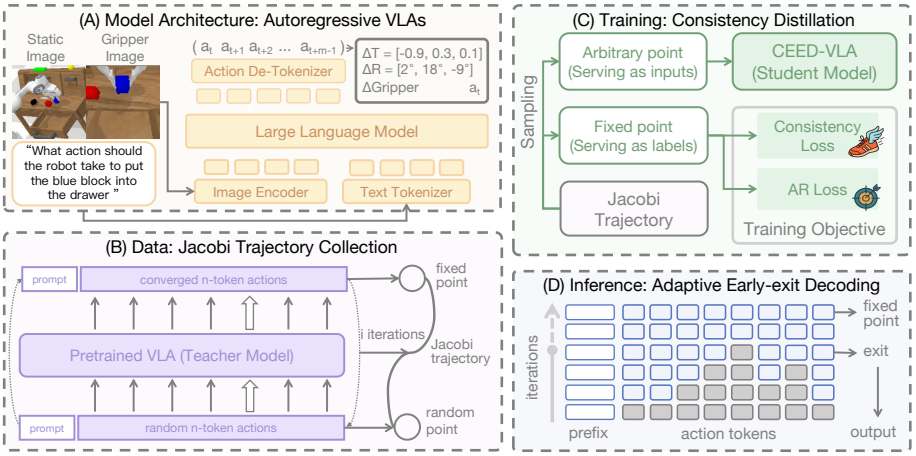

Fig. 2: Overview of our proposed **CEED-VLA**. Given an autoregressive VLA model (*e.g.*, OpenVLA [16]) that is pretrained on an in-domain task (**A**), we first prompt it to perform Jacobi decoding to collect Jacobi trajectory dataset (**B**). Then we design an effective consistency training process to train a student model (**C**). Finally, we propose adaptive early-exit decoding to further unlock inference speed (**D**).

generator in a dual-system architecture, achieving strong performance across diverse tasks and embodiments.

Discrete outputs. OpenVLA [16] introduced the first open-source 7B-parameter VLA model, trained entirely on public datasets and capable of generating discrete action tokens. π0-FAST [28] proposed a fully autoregressive decoding framework with a novel action tokenizer, which demonstrates that AR-based VLAs do better in learning efficiency and instruction following than diffusion-based VLAs. UniVLA [36] and WorldVLA [7] unify visual, textual, and action tokens within a shared discrete space, thereby endowing VLA models with world-modeling capabilities. VLM2VLA [12] directly represents actions in natural language, which better preserves the pretrained capabilities of the VLA backbone and enhances its linguistic generalization ability. However, the AR-based VLAs are still limited by slow inference speed. Under the premise of leveraging their advantages, we propose CEED-VLA to overcome the limitations of the autoregressive nature, thereby significantly improving inference speed and making the application of AR-based VLAs practically feasible.

2.2 Acceleration for VLA Models

Acceleration is an important research topic for VLAs to complete complex, high-frequency tasks. Efforts to enhance efficiency have led to layer-wise computational reduction in VLA models, such as DeeR-VLA [41], which dynamically adjusts inference depth, and MoLe-VLA [43], which achieves this by selectively skipping transformer layers based on task-relevant cues. BitVLA [33] and QVLA [39] integrate quantization-aware training. Further architecture changes, like RoboMamba [21] and TinyVLA [37], replace traditional attention mecha-

nisms or focus on developing lightweight models from the ground up, frequently necessitating model re-training and additional data collection. For token caching and pruning, VLA-Cache [38] selectively caches static tokens and recomputes only dynamic or task-relevant ones, while ADP [27] adaptively adjusts token retention ratios in accordance with dynamics. Furthermore, EfficientVLA [40] first designs a systematic method to eliminate multifaceted redundancies across the visual encoder, language model, and action head. FAST [28] introduces a frequency-domain action tokenization scheme that improves the efficiency and scalability of autoregressive VLAs. PD-VLA [30] 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. By contrast, our CEED-VLA targets decoder-side acceleration by reshaping decoding dynamics through consistency distillation, without modifying model architecture or reducing parameters, and can be seamlessly combined with the above complementary techniques, such as architectural compression, pruning, and efficient tokenization.

3 CEED-VLA

3.1 Preliminary: Jacobi Decoding

Given a prompt $\boldsymbol{x}$, an autoregressive LLM p typically generates responses in a next-token-prediction manner:

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

where $\mathcal{Y}_{i-1}$ denotes $\{y_1, \ldots, y_{i-1}\}$, 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 [17, 29] 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:

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

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 [25]. Concretely, Jacobi decoding first initializes a random token sequence $\mathcal{Y}^{(0)} = \{y_1^{(0)}, \ldots, y_n^{(0)}\}$. 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 met at iteration k when $\mathcal{Y}^{(k)} = \mathcal{Y}^{(k-1)}$, i.e., the iteration no longer change the sequence. At this point, we define the fixed point as $\mathcal{Y}^* := \mathcal{Y}^{(k)}$. 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.

3.2 Teacher Model

Vanilla VLAs P_γ learn to predict actions $\hat{a}_{t+1}$ directly from the current observation s_t and language instruction l (See Figure 2). Some research [30] replaces the AR decoding in these VLAs with Jacobi decoding to accelerate the inference, while the speedup is **limited** (1.28×). The reason is that VLAs, which have 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. Therefore, when any preceding token is incorrect, the model is unlikely to generate the correct subsequent token.

3.3 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.

For the target VLA P_γ, we let $Q_\theta(\cdot|s_t, l)$ denote the CEED-VLA with parameters θ initialized with those of P_γ. To capture the inherent consistency within Jacobi trajectories, we first collect them by prompting P_γ 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 1.

Algorithm 1 Jacobi trajectory collection

Input: Dataset $\mathcal{C}$, teacher model P_γ
repeat
  Sample observation s_t and instruction l from $\mathcal{C}$
  Initialize $\mathcal{Y}^{\{0\}}$ randomly
  repeat
    $\mathcal{Y}^{\{j\}} \leftarrow P_\gamma(\mathcal{Y}^{\{j-1\}}, s_t, l)$
    Add $\mathcal{Y}^{\{j\}}$ to $\mathcal{J}$
  until $\mathcal{Y}^{\{j\}} = \mathcal{Y}^{\{j-1\}}$
  Get complete $\mathcal{J} = \{\mathcal{Y}^{\{0\}}, \ldots, \mathcal{Y}^*\}$
  Append $(s_t, l, \mathcal{J})$ to $\mathcal{D}$
until all samples in $\mathcal{C}$ are processed

Algorithm 2 Consistency Training

Input: Original robot dataset $\mathcal{C}$, Jacobi trajectory dataset $\mathcal{D}$, weight ω, CEED-VLA $Q_\theta(\cdot \mid s_t, l)$, correctness threshold $\delta_{\max}$
repeat
  Sample observation s_t and language instruction l, Jacobi trajectory $\mathcal{J}$ from $\mathcal{D}$, teacher model's output $\mathcal{Y}^*$ from $\mathcal{J}$, ground-truth label GT from $\mathcal{C}$
  Compute $\mathcal{L}_{\text{AR}}$ using Equation (5) with Y^* or GT as the target
  Randomly sample $\mathcal{Y}$ from $\mathcal{J}$
  Compute $\mathcal{L}_{\text{C}}$ using Equation (3)
  update parameters θ using $\mathcal{L}$ (Equation (6))
until convergence

Consistency Training. The consistency training procedure optimizes two objectives: *(i)* guiding the model to predict multiple correct action tokens simultaneously, and *(ii)* constraining CEED-VLA from drifting away from the target VLA distribution to preserve manipulation skills. To achieve this goal, we meticulously designed two losses.

We first introduce our consistency loss. The ideal finetuned student 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 $\boldsymbol{x}$ consisting of both instructions l and visual inputs s_t, and Jacobi trajectory $\mathcal{J}$, let $\mathcal{Y}$ be a randomly sampled intermediate iteration and $\mathcal{Y}^*$ be the corresponding fixed point from $\mathcal{J}$. We train CEED-VLA to directly predict $\mathcal{Y}^*$ from $\mathcal{Y}$ by minimizing the following loss:

$$\mathcal{L}_{\text{C}} = \mathbb{E}_{(s_t,l,\mathcal{J})\sim\mathcal{D},\ \mathcal{Y}\sim\mathcal{J}} \left[\sum_{i=1}^{n} \text{KL}(\text{T} \,\|\, \text{S}) \right], \tag{3}$$

where T and S denote the teacher and student model distributions, respectively. Here, $\text{KL}(\cdot \,\|\, \cdot)$ denotes the forward Kullback–Leibler (KL) divergence between two distributions. The detailed formulation of the KL divergence used in our method is as follows:

$$\text{KL}(\text{T} \,\|\, \text{S}) = Q_{\theta^-}(\cdot|\mathcal{Y}_i^*, s_t, l) \log \frac{Q_{\theta^-}(\cdot|\mathcal{Y}_i^*, s_t, l)}{Q_\theta(\cdot|\mathcal{Y}_i, s_t, l)}, \tag{4}$$

where $\theta^- = \text{stopgrad}(\theta)$.

Then, we introduce another AR supervision. To avoid deviating from the distribution of the teacher model, we inherit the AR loss function $\mathcal{L}_{\text{AR}}$ used for training the teacher model:

$$\mathcal{L}_{\text{AR}} = \mathbb{E}_{(s_t,l,\mathcal{Y}^*)\sim\mathcal{D}} \Big[-\sum_{i=1}^{N} \log Q_\theta(\mathcal{Y}_i^*|\mathcal{Y}_{<i}^*, s_t, l) \Big]. \tag{5}$$

Consequently, the total loss for training our CEED-VLA is:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{C}} + \omega \mathcal{L}_{\text{AR}}, \tag{6}$$

Table 1: Comparison of inference speeds among AR decoding, Jacobi decoding, and adaptive early-exit decoding on our CEED-VLA. Here, the minimum speed 19.50 of Jacobi decoding, which is even lower than AR's, corresponds to the **redundant iterations**.

Decoding Method	Average Speed	Minimum Speed	Maximum Speed
AR	39.64	30.87	49.10
Jacobi	57.57 ↑	19.50 ↓	82.64 ↑
Adaptive Early-exit	79.24 ↑	62.07 ↑	93.53 ↑

where ω is a weight factor. The training procedure is detailed in Algorithm 2.

Inference. We first introduce the existence of redundant iterations. Although the student model CEED-VLA performs inference via Jacobi decoding after consistency distillation, our empirical observations reveal that the acceleration performance of CEED-VLA is bottlenecked by a few iterations that are even slower than the speed of AR decoding (see Table 1). This phenomenon arises from the **strict convergence conditions** about the fixed point of Jacobi decoding. It requires exact convergence between successive iterations, *i.e.*, $\mathcal{Y}^{(k)} = \mathcal{Y}^{(k-1)}$, which takes numerous iterations to reach. Empirically, we observed that token updates in the final steps of redundant iterations are limited, contributing little to the final action decisions. These redundant iterations significantly hinder the speedup of inference and may introduce delays or discontinuities in high-frequency dexterous tasks, ultimately compromising task performance. This suggests that relaxing convergence and reducing iterations can further improve the decoding process.

To this end, we propose **Adaptive Early-exit Decoding** to relax the termination condition and significantly reduce the number of iterations. The adaptive early-exit decoding is based on the confidence score for token i at iteration j:

$$c_{j,i} = \max_r Q_\theta\big(r \mid \mathcal{Y}^{(j)}_{<i}, s_t, l\big), \tag{7}$$

where r denotes the candidate token probability in the vocabulary distribution. We further define the minimum token confidence at iteration j as $c_j^{\min} = \min\{\, c_{j,i} \mid 0 \le i \le n \,\}$. and introduce a confidence threshold σ. Once $c_j^{\min} \ge \sigma$, the model halts the iterative process early and directly outputs the intermediate sequence $\mathcal{Y}^{(j)}$, skipping all subsequent decoding steps. This approach mitigates redundant iterations and significantly improves both the minimum and average decoding speeds while maintaining performance.

In addition, since the prefix tokens (*e.g.,* the task instruction) remain unchanged across iterations, we cache and reuse their attention key–value states, further reducing redundant computation.

Table 2: Comparison with various base models in terms of inference speed, success rate or average length, and execution frequency. Experiments based on LLaVA-VLA [31] are conducted on CALVIN [23] ABC→D, while those based on OpenVLA [16] are evaluated on LIBERO [19] Long.

Acceleration Techniques	Chunk Size	Decoding Method	Splits	Speed (Tokens/s)	Average Length	SR (%)	Freq. (Hz)
Base model: LLaVA-VLA Benchmark: CALVIN							
N/A	-	AR	ABC→D	39.6 (×1)	2.01	–	2.23 (×1)
N/A	5	AR	ABC→D	39.6 (×1)	**3.70**	–	4.33 (×1.9)
FastV [8]	5	AR	ABC→D	28.7 (×0.7)	2.54	–	1.87 (×0.8)
SparseVLM [44]	5	AR	ABC→D	32.4 (×0.8)	2.83	–	2.01 (×0.9)
PD-VLA [30]	5	Jacobi	ABC→D	52.8 (×1.3)	3.69	–	5.43 (×2.4)
CEED-VLA (Ours)	5	Adaptive Early-exit	ABC→D	**79.0 (×2.0)**	3.68	–	**7.26 (×3.3)**
Base model: OpenVLA Benchmark: LIBERO							
N/A	-	AR	Long	54.4 (×1)	–	53.2	5.95 (×1)
N/A	3	AR	Long	54.4 (×1)	–	60.4	7.08 (×1.2)
PD-VLA [30]	3	Jacobi	Long	85.0 (×1.6)	–	**62.4**	10.79 (×2.4)
CEED-VLA (Ours)	3	Adaptive Early-exit	Long	**225.2 (×4.1)**	–	62.3	**25.62 (×4.3)**

Table 3: Comparison between base models (π_0-FAST, UniVLA) and our acceleration method on LIBERO, reporting Success Rate (SR, %) and Speed (Tokens/s) across task categories.

Method	Spatial		Object		Goal		Long		Avg.	
	SR↑	Speed↑	SR↑	Speed↑	SR↑	Speed↑	SR↑	Speed↑	SR↑	Speed↑
π_0-FAST [28]	96.4	105.3	96.8	108.1	88.6	109.2	60.2	107.5	85.5	107.5 (×1.0)
CEED-VLA (Ours)	96.6	**419.4**	96.2	**422.1**	89.4	**421.6**	59.9	**419.3**	85.5	**420.6 (×3.9)**
UniVLA [36]	97.0	50.2	99.0	50.6	92.6	50.4	90.8	51.1	94.8	50.6 (×1.0)
CEED-VLA (Ours)	96.8	**200.4**	98.2	**199.6**	92.0	**198.7**	90.6	**202.4**	94.4	**200.3 (×4.0)**

4 Experiments

4.1 Simulated 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 [23] includes 34 tasks across four environments (A, B, C, and D). Following the classic ABC→D setup, we run 500 rollouts per model, where each rollout involves a sequence of 5 consecutive sub-tasks. We report the average length of successful sub-task completions of all rollouts, denoted as *avg. len.*, with a maximum value of 5. LIBERO-Long [19] consists of 10 long-horizon and multi-step manipulation tasks with diverse objects and skills, emphasizing temporal reasoning, goal consistency and delayed rewards. Each method is eval-

uated over 50 rollouts per task, for a total of 500 rollouts, and we report the overall average success rate.

Base Models. In this section, we consider fine-tuning diverse base models for a comprehensive validation of our CEED-VLA. LLaVA-VLA is a vanilla VLA model based on LLaVA [20], which has been wildly used as baselines [30, 46, 47] with further details provided in our supplementary material. We also include OpenVLA [16], the most popular open-source VLA model, and π0-FAST, the SOTA-level AR-based VLA model, for general evaluation.

4.2 Acceleration Results on Traditional Autoregressive VLA .

Acceleration Results of LLaVA-VLA on CALVIN. We compare our method with naïve base models, advanced acceleration methods for VLMs (FastV [8], SparseVLM [44]), and VLAs in parallel decoding (PD-VLA [30]). 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 consistency training and adaptive early-exit decoding, CEED-VLA is able to predict more accurate action tokens, resulting in a 2.0× speedup, 3.3× frequencies while the average length only reduces 0.02.

Acceleration Results of OpenVLA on LIBERO. Compared to LLaVA-VLA, OpenVLA benefits from pre-training on the large-scale OXE [26] robotic dataset under an autoregressive objective. Experiments of OpenVLA with our consistency training on the LIBERO-Long benchmark further validate our conclusions. Our proposed CEED-VLA realizes 4.1× speed and 4.3× frequency based on OpenVLA. The improvement is more evident than CALVIN, because the tasks on LIBERO are relatively easy and short-horizon, thus requiring fewer iterations in Jacobi decoding. Meanwhile, the integration with action chunking improves 9.1% success rates because of stronger planning abilities and smoother actions.

Adaptability to Different Action Tokenization Schemes. π_0-FAST features fewer parameters with an efficient action tokenization method. Building on this, as shown in Table 3, CEED-VLA achieves a 3.9× speedup with comparable success rates, demonstrating that: (1) CEED-VLA generalizes effectively across different network architectures and model scales, and (2) as a decoder-side acceleration technique, CEED-VLA can be seamlessly integrated with various tokenization schemes (e.g., FAST [28] and VQ-based tokenization [35]) to further boost performance.

Extensibility to Unified VLA Architecture. UniVLA discretizes images into tokens through VQ-VAE, and jointly trains future images generation and actions prediction in a unified framework. Table 3 shows that our CEED-VLA is also applicable to the this paradigm, achieving a 4.0×

Table 4: Profiling results for fixed tokens and speedups with LLaVA-VLA [31], PD-VLA [30], and our method.

Model	Fixed Tokens	Speedup
LLaVA-VLA	0	1×
PD-VLA	8.75	1.33×
CEED-VLA	**13.5**	**2.04**×

Table 5: Ablation of different techniques used in our CEED-VLA applied to the base model OpenVLA of [16] on LIBERO LONG [19] tasks.

Jacobi Decoding	Consistency Training	Adaptive Early-exit	Speedup	Success Rate (%)
✓	✓	✓	**×4.1**	62.3
✓	✓	×	×2.5	61.2
✓	✓	×	×2.3	54.2
✓	×	×	×1.6	**62.4**
×	×	×	×1.0	60.4

acceleration for both action and image generation, offering a feasible solution for world-modeling architecture.

Underlying Acceleration Phenomena. We conduct a visualization of Jacobi trajectory in Figure 1 of the supplementary documents and 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.

4.3 Ablation Study

In this section, we first overview the significance of our key designs on the base model OpenVLA on LIBERO. As illustrated in Table 5, consistency training and early-exit decoding substantially accelerate the model. Then, on LLaVA-VLA, we investigate the choice of data amounts in Figure 3, and losses in Table 6.

Adaptive Early-exit Decoding. Adaptive early-exit decoding serves as a key accelerator. While maintaining the success rate, it brings a substantial acceleration, improving speed by 2.5× to 4.1×. This is because confidence-guided decoding ensures output reliability while effectively reducing the number of Jacobi iterations.

Consistency Training. To better understand its necessity, we further conducted a comparison on the model without consistency training. Specifically, we assume that intermediate states at the j-step iteration, which have not yet reached the fixed point, can be treated as outputs. This allows us to assess whether consistency training truly improves the quality of intermediate predictions during Jacobi decoding and accelerates overall convergence. Figure 3 shows that for the model without consistency training, the average length decreases significantly as j becomes smaller. Specifically, when j is 12 and 8, the average lengths drop to 1.99 and 1.44, respectively, whereas our model with consistency training maintains an average length above 3.34. This indicates that consistency training plays a central role in enabling parallel decoding within CEED-VLA, allowing the model to produce accurate actions with minimal forward passes.

Loss Design. The consistency loss guides the model to learn the convergence behavior of the Jacobi trajectory toward a fixed point, thereby reducing the

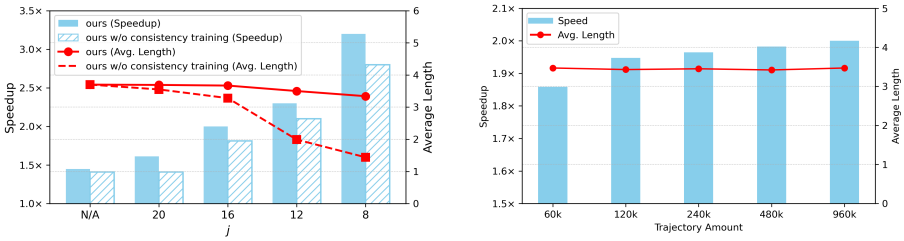

Fig. 3: In-depth analysis of consistency training by directly mapping any intermediate point j as output (left), and on different amounts of training data (right). This figure reports speedup and average length using LLaVA-VLA [31] as the base model on the CALVIN [23] benchmark.

Table 7: Ablation of confidence threshold σ. We evaluate LLaVA-VLA [31] on the CALVIN and Real-world, and π_0-FAST [28] on LIBERO to analyze the impact of different confidence threshold settings.

σ	Calvin ABC→D		Libero 10		Real-world	
	Speed↑	Avg. Len.↑	Speed↑	Success↑ (%)	Speed↑	Success↑ (%)
0.7	1.3×	3.72	1.5×	61.3	1.8×	81.0
0.6	1.5×	3.68	2.4×	60.5	2.5×	79.0
0.5 (ours)	2.0×	3.67	3.9×	59.9	3.9×	77.5
0.4	3.2×	3.63	4.6×	58.3	4.4×	72.5

iterations and accelerating the overall decoding process. The 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.

Data Size. CEED-VLA learns to predict multiple tokens in each step by leveraging pre-collected Jacobian trajectories. As shown in Figure 3, 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 6: Comparison of inference speed and average length between different ratios of losses to evaluate the trade-off.

Loss	Speedup	Avg. Len.
$\mathcal{L}_{\mathrm{C}} + \mathcal{L}_{\mathrm{AR}}$ (ours)	**2.00**×	3.67
$\mathcal{L}_{\mathrm{C}} + 10 * \mathcal{L}_{\mathrm{AR}}$	1.79×	**3.74**

Ablation of Confidence Threshold σ. We conduct a comprehensive analysis of the confidence threshold σ by varying it from 0.4 to 0.7 and evaluating across multiple benchmarks. As shown in Table 7, $\sigma = 0.5$ achieves the best trade-off between accuracy and efficiency, delivering substantial speedup with

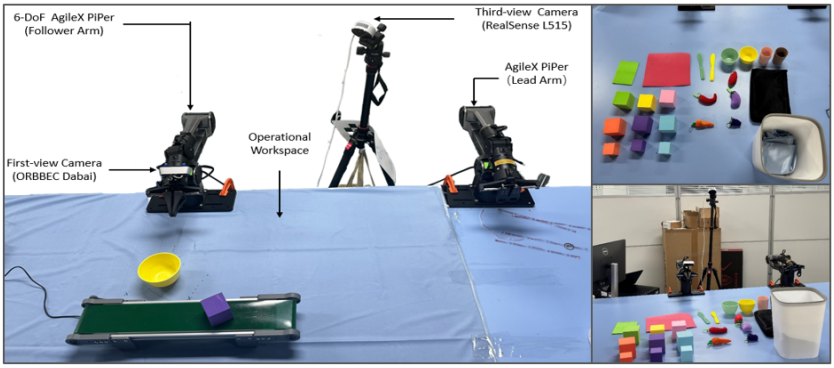

Fig. 4: The real-world robotic system for experiments. The system consists of two AgileX PiPer 6-DoF robotic arms, an ORBBEC Dabai depth camera, and a RealSense L515 depth camera.

Table 8: Computational Overhead of Our Pipeline. All reported results are obtained using 8 NVIDIA A100 GPUs.

Stage	Time (hours)
Fine-tuning	48
Jacobi Trajectory Generation	2
Consistency Distillation	12

only marginal performance degradation. Overall, the performance remains stable within different hyperparameters, indicating that our method is robust to the choice of σ.

Additional Computational Overhead of Our Pipeline. We quantify the computational overhead introduced by CEED-VLA when applied to LLaVA-VLA. As shown in Table 8, fine-tuning LLaVA to obtain LLaVA-VLA requires 48 hours on 8 NVIDIA A100 GPUs. In comparison, Jacobi trajectory generation takes only 2 hours, and consistency distillation requires 12 hours, resulting in only modest additional overhead relative to standard VLA fine-tuning. Moreover, the generated Jacobi trajectories can be reused across multiple distillation runs, further improving overall efficiency.

5 Real-world Setup

5.1 Real-world Experiments

Real-world Setup. As shown in Figure 4, our real-world experiments were performed on a dual-arm AgileX PiPer robotic system (See supplementary documents). During the data collection phase, one arm is teleoperated by a human to provide demonstrations (designated as the lead arm), while the other arm

Table 9: Success rates in real-world experiments of our CEED-VLA, compared with the base model LLaVA-VLA [31] and baseline method PD-VLA [30]. We evaluate all methods on 4 tasks, with 20 rollouts per task (*i.e.*, 80 rollouts in total) and report success rates (%).

Method	Push button (basic)	Lift block (basic)	Pour water (dexterous)	Fold towel (dexterous)	Average	Frequency (Hz)
LLaVA-VLA	65	40	15	5	33.25	3.3
PD-VLA	85	65	45	35	57.50	6.0 (*1.8×*)
CEED-VLA (ours)	**85**	**70**	**80**	**75**	**77.50**	**13.0** (*3.9×*)

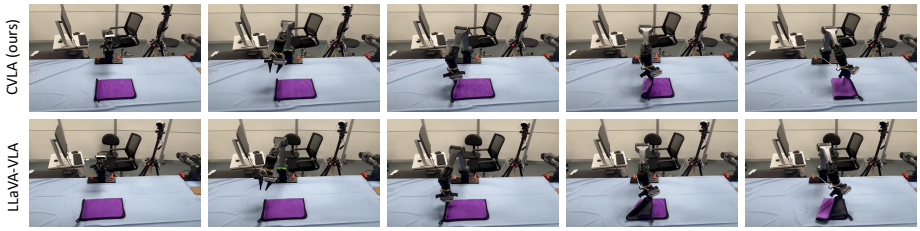

Fig. 5: Visualization of the dexterous task in the real-world experiment. We take the towel folding for example. The first row illustrates a successful case of our CEED-VLA, while the second row presents a failure case of LLaVA-VLA under the same instruction and scene. Besides, the duration of the bottom figure is approximately 4× that of the top figure.

(designated as the follower arm) remains passive. During evaluation, only the follower arm is controlled by VLAs to perform the tasks. To provide visual observations, we employ a RealSense L515 depth camera as third-view input and an ORBBEC Dabai depth camera as first-view input.

Tasks and Datasets. Our dataset covers two levels of manipulation difficulty, categorized into basic tasks and dexterous tasks. 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 30 Hz.

Results. Table 9 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%.

Visualization. To intuitively analyze the performance gain of CEED-VLA on dexterous tasks brought by accelerated inference, we visualize two representative

cases in Figure 5. In the *fold towel* task, CEED-VLA successfully grasps the towel and completes the folding process with smooth actions. In contrast, although LLaVA-VLA is also capable of picking up the towel, its unstable motion causes one side of the towel to slip during the folding process, ultimately resulting in failure. The high-frequency inference of CEED-VLA not only shortened the task completion time, but also mitigated action latency, which induced pauses and jitters in lightweight robotic arms (*e.g.*, the PiPer arm shown in the figure). This issue is usually overlooked on industrial robotic arms.

6 Conclusions

In this paper, we propose CEED-VLA, a universal acceleration method for significant inference speedup while maintaining manipulation performance. Extensive experiments in several simulation and real-world environments demonstrate that our CEED-VLA better manages high-frequency dexterous tasks. We provide more discussion about future work in the supplementary document.

Acknowledgements

This work was supported in part by the Natural Science Foundation of China under Grant 62403401, in part by the National Science and Technology Innovation 2030 - Major Project under Grant 2022ZD0208800, in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2024A1515011992 and Grant 2026A1515012323, and in part by the Guangdong Provincial Project under Grant 2024QN11X127.

References

1. An, X., Xie, Y., Yang, K., Zhang, W., Zhao, X., Cheng, Z., Wang, Y., Xu, S., Chen, C., Wu, C., et al.: Llava-onevision-1.5: Fully open framework for democratized multimodal training. arXiv preprint arXiv:2509.23661 (2025)
2. Awadalla, A., Gao, I., Gardner, J., Hessel, J., Hanafy, Y., Zhu, W., Marathe, K., Bitton, Y., Gadre, S., Sagawa, S., Jitsev, J., Kornblith, S., Koh, P.W., Ilharco, G., Wortsman, M., Schmidt, L.: Openflamingo: An open-source framework for training large autoregressive vision-language models. arXiv preprint arXiv:2308.01390 (2023)
3. Bjorck, J., Castañeda, F., Cherniadev, N., Da, X., Ding, R., Fan, L., Fang, Y., Fox, D., Hu, F., Huang, S., et al.: Gr00t n1: An open foundation model for generalist humanoid robots. arXiv preprint arXiv:2503.14734 (2025)
4. Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., et al.: π_0: A vision-language-action flow model for general robot control. arXiv preprint arXiv:2410.24164 (2024)
5. Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., , et al.: Rt-1: Robotics transformer for real-world control at scale. Proceedings of Robotics: Science and Systems (2023)

6. Bu, Q., Yang, Y., Cai, J., Gao, S., Ren, G., Yao, M., Luo, P., Li, H.: Univla: Learning to act anywhere with task-centric latent actions. arXiv preprint arXiv:2505.06111 (2025)
7. Cen, J., Yu, C., Yuan, H., Jiang, Y., Huang, S., Guo, J., Li, X., Song, Y., Luo, H., Wang, F., et al.: Worldvla: Towards autoregressive action world model. arXiv preprint arXiv:2506.21539 (2025)
8. Chen, L., Zhao, H., Liu, T., Bai, S., Lin, J., Zhou, C., Chang, B.: 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. pp. 19–35. Springer (2024)
9. Deng, T., Pan, Y., Yuan, S., Li, D., Wang, C., Li, M., Chen, L., Xie, L., Wang, D., Wang, J., Civera, J., Wang, H., Chen, W.: What is the best 3d scene representation for robotics? from geometric to foundation models. arXiv preprint arXiv:2512.03422 (2025)
10. Driess, D., Springenberg, J.T., Ichter, B., Yu, L., Li-Bell, A., Pertsch, K., Ren, A.Z., Walke, H., Vuong, Q., Shi, L.X., et al.: Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. arXiv preprint arXiv:2505.23705 (2025)
11. Driess, D., Xia, F., Sajjadi, M.S., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., et al.: Palm-e: an embodied multimodal language model. In: Proceedings of the 40th International Conference on Machine Learning. pp. 8469–8488 (2023)
12. Hancock, A.J., Wu, X., Zha, L., Russakovsky, O., Majumdar, A.: Actions as language: Fine-tuning vlms into vlas without catastrophic forgetting. arXiv preprint arXiv:2509.22195 (2025)
13. 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. arXiv preprint arXiv:2412.14803 (2024)
14. Intelligence, P., Black, K., Brown, N., Darpinian, J., Dhabalia, K., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., et al.: $\pi_0.5$: a vision-language-action model with open-world generalization. arXiv preprint arXiv:2504.16054 (2025)
15. Karamcheti, S., Nair, S., Balakrishna, A., Liang, P., Kollar, T., Sadigh, D.: Prismatic vlms: Investigating the design space of visually-conditioned language models. In: Forty-first International Conference on Machine Learning (2024)
16. Kim, M.J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E.P., Sanketi, P.R., Vuong, Q., et al.: Openvla: An open-source vision-language-action model. In: 8th Annual Conference on Robot Learning (2024)
17. Kou, S., Hu, L., He, Z., Deng, Z., Zhang, H.: Cllms: Consistency large language models. In: Forty-first International Conference on Machine Learning (2024)
18. Li, X., Liu, M., Zhang, H., Yu, C., Xu, J., Wu, H., Cheang, C., Jing, Y., Zhang, W., Liu, H., et al.: Vision-language foundation models as effective robot imitators. In: The Twelfth International Conference on Learning Representations (2024)
19. Liu, B., Zhu, Y., Gao, C., Feng, Y., Liu, Q., Zhu, Y., Stone, P.: Libero: Benchmarking knowledge transfer for lifelong robot learning. Advances in Neural Information Processing Systems **36** (2024)
20. Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. Advances in neural information processing systems **36** (2024)
21. Liu, J., Liu, M., Wang, Z., An, P., Li, X., Zhou, K., Yang, S., Zhang, R., Guo, Y., Zhang, S.: Robomamba: Efficient vision-language-action model for robotic reasoning and manipulation. Advances in Neural Information Processing Systems **37**, 40085–40110 (2024)

22. Liu, S., Wu, L., Li, B., Tan, H., Chen, H., Wang, Z., Xu, K., Su, H., Zhu, J.: Rdt-1b: a diffusion foundation model for bimanual manipulation. In: The Thirteenth International Conference on Learning Representations (2025)
23. Mees, O., Hermann, L., Rosete-Beas, E., Burgard, W.: Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. IEEE Robotics and Automation Letters (2021)
24. Octo Model Team, Ghosh, D., Walke, H., Pertsch, K., Black, K., Mees, O., Dasari, S., Hejna, J., Xu, C., Luo, J., Kreiman, T., Tan, Y., Chen, L.Y., Sanketi, P., Vuong, Q., Xiao, T., Sadigh, D., Finn, C., Levine, S.: Octo: An open-source generalist robot policy. In: Proceedings of Robotics: Science and Systems. Delft, Netherlands (2024)
25. Ortega, J.M., Rheinboldt, W.C.: Iterative solution of nonlinear equations in several variables. SIAM (2000)
26. O'Neill, A., Rehman, A., Maddukuri, A., Gupta, A., Padalkar, A., Lee, A., Pooley, A., Gupta, A., Mandlekar, A., Jain, A., 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). pp. 6892–6903. IEEE (2024)
27. Pei, X., Chen, Y., Xu, S., Wang, Y., Shi, Y., Xu, C.: Action-aware dynamic pruning for efficient vision-language-action manipulation. arXiv preprint arXiv:2509.22093 (2025)
28. Pertsch, K., Stachowicz, K., Ichter, B., Driess, D., Nair, S., Vuong, Q., Mees, O., Finn, C., Levine, S.: Fast: Efficient action tokenization for vision-language-action models. arXiv preprint arXiv:2501.09747 (2025)
29. Santilli, A., Severino, S., Postolache, E., Maiorca, V., Mancusi, M., Marin, R., Rodola, E.: Accelerating transformer inference for translation via parallel decoding. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL). pp. 12336–12355 (2023)
30. Song, W., Chen, J., Ding, P., Zhao, H., Zhao, W., Zhong, Z., Ge, Z., Li, Z., Wang, D., Wang, L., Ma, J., Li, H.: Pd-vla: Accelerating vision-language-action model integrated with action chunking via parallel decoding. In: 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 13162–13169 (2025). https://doi.org/10.1109/IROS60139.2025.11247519
31. Song, W., Chen, J., Sun, X., Lei, H., Qin, Y., Zhao, W., Ding, P., Zhao, H., Wang, T., Hou, P., et al.: Rethinking the practicality of vision-language-action model: A comprehensive benchmark and an improved baseline. arXiv preprint arXiv:2602.22663 (2026)
32. Team, G., Kamath, A., Ferret, J., Pathak, S., Vieillard, N., Merhej, R., Perrin, S., Matejovicova, T., Ramé, A., Rivière, M., et al.: Gemma 3 technical report. arXiv preprint arXiv:2503.19786 (2025)
33. Wang, H., Xiong, C., Wang, R., Chen, X.: Bitvla: 1-bit vision-language-action models for robotics manipulation. arXiv preprint arXiv:2506.07530 (2025)
34. Wang, P., Bai, S., Tan, S., Wang, S., Fan, Z., Bai, J., Chen, K., Liu, X., Wang, J., Ge, W., et al.: Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. arXiv preprint arXiv:2409.12191 (2024)
35. Wang, Y., Zhu, H., Liu, M., Yang, J., Fang, H.S., He, T.: Vq-vla: Improving vision-language-action models via scaling vector-quantized action tokenizers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 11089–11099 (2025)
36. Wang, Y., Li, X., Wang, W., Zhang, J., Li, Y., Chen, Y., Wang, X., Zhang, Z.: Unified vision-language-action model. arXiv preprint arXiv:2506.19850 (2025)

37. Wen, J., Zhu, Y., Li, J., Zhu, M., Tang, Z., Wu, K., Xu, Z., Liu, N., Cheng, R., Shen, C., Peng, Y., Feng, F., Tang, J.: Tinyvla: Toward fast, data-efficient vision-language-action models for robotic manipulation. IEEE Robotics and Automation Letters **10**(4), 3988–3995 (2025). https://doi.org/10.1109/LRA.2025.3544909
38. Xu, S., Wang, Y., Xia, C., Zhu, D., Huang, T., Xu, C.: Vla-cache: Towards efficient vision-language-action model via adaptive token caching in robotic manipulation (2025)
39. Xu, Y., Yang, Y., Fan, Z., Liu, Y., Li, Y., Li, B., Zhang, Z.: Qvla: Not all channels are equal in vision-language-action model's quantization. arXiv preprint arXiv:2602.03782 (2026)
40. Yang, Y., Wang, Y., Wen, Z., Zhongwei, L., Zou, C., Zhang, Z., Wen, C., Zhang, L.: Efficientvla: Training-free acceleration and compression for vision-language-action models. arXiv preprint arXiv:2506.10100 (2025)
41. Yue, Y., Wang, Y., Kang, B., Han, Y., Wang, S., Song, S., Feng, J., Huang, G.: 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)
42. Zhang, J., Guo, Y., Hu, Y., Chen, X., Zhu, X., Chen, J.: Up-vla: A unified understanding and prediction model for embodied agent. arXiv preprint arXiv:2501.18867 (2025)
43. Zhang, R., Dong, M., Zhang, Y., Heng, L., Chi, X., Dai, G., Du, L., Du, Y., Zhang, S.: Mole-vla: Dynamic layer-skipping vision language action model via mixture-of-layers for efficient robot manipulation. arXiv preprint arXiv:2503.20384 (2025)
44. Zhang, Y., Fan, C.K., Ma, J., Zheng, W., Huang, T., Cheng, K., Gudovskiy, D.A., Okuno, T., Nakata, Y., Keutzer, K., Zhang, S.: SparseVLM: Visual token sparsification for efficient vision-language model inference. In: Forty-second International Conference on Machine Learning (2025), https://openreview.net/forum?id=80faIPZ67S
45. Zhang, Z., Zheng, H., Wang, Y., Xu, L., Deng, T., Chen, X., Chen, Q., Zhang, B., Huang, W.: Omnidrive-r1: Reinforcement-driven interleaved multi-modal chain-of-thought for trustworthy vision-language autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings. pp. 1106–1116 (June 2026)
46. Zhao, W., Ding, P., Zhang, M., Gong, Z., Bai, S., Zhao, H., Wang, D.: Vlas: Vision-language-action model with speech instructions for customized robot manipulation. International Conference on Learning Representations (ICLR) (2025)
47. Zhao, W., Li, G., Gong, Z., Ding, P., Zhao, H., Wang, D.: Unveiling the potential of vision-language-action models with open-ended multimodal instructions. arXiv preprint arXiv:2505.11214 (2025)
48. Zitkovich, B., Yu, T., Xu, S., Xu, P., Xiao, T., Xia, F., Wu, J., Wohlhart, P., Welker, S., Wahid, A., et al.: Rt-2: Vision-language-action models transfer web knowledge to robotic control. In: Conference on Robot Learning. pp. 2165–2183. PMLR (2023)