

---

# EfficientVLA: Training-Free Acceleration and Compression for Vision-Language-Action Models

---

Yantai Yang<sup>1,2</sup> Yuhao Wang<sup>1,3</sup> Zichen Wen<sup>1</sup> Luo Zhongwei<sup>1</sup> Chang Zou<sup>1,4</sup> Zhipeng Zhang<sup>1,5</sup>  
Chuan Wen<sup>1\*</sup> Linfeng Zhang<sup>1\*</sup>

<sup>1</sup>School of Artificial Intelligence, Shanghai Jiao Tong University

<sup>2</sup>Harbin Institute of Technology

<sup>3</sup>Xi'an Jiaotong University

<sup>4</sup>University of Electronic Science and Technology of China

<sup>5</sup>Anyverse Dynamics

🔗 Code: <https://github.com/YantaiYang-05/EfficientVLA>

## Abstract

Vision-Language-Action (VLA) models, particularly diffusion-based architectures, demonstrate transformative potential for embodied intelligence but are severely hampered by high computational and memory demands stemming from extensive inherent and inference-time redundancies. While existing acceleration efforts often target isolated inefficiencies, such piecemeal solutions typically fail to holistically address the varied computational and memory bottlenecks across the entire VLA pipeline, thereby limiting practical deployability. We introduce **EfficientVLA**, a structured and training-free inference acceleration framework that systematically eliminates these barriers by cohesively exploiting multifaceted redundancies. EfficientVLA synergistically integrates three targeted strategies: (1) pruning of functionally inconsequential layers from the language module, guided by an analysis of inter-layer redundancies; (2) optimizing the visual processing pathway through a task-aware strategy that selects a compact, diverse set of visual tokens, balancing task-criticality with informational coverage; and (3) alleviating temporal computational redundancy within the iterative diffusion-based action head by strategically caching and reusing key intermediate features. We apply our method to a standard VLA model CogACT, yielding a  $1.93\times$  inference speedup and reduces FLOPs to 28.9%, with only a 0.6% success rate drop in the SIMPLER benchmark.

## 1 Introduction

Building upon advances in multimodal understanding from models integrating vision and language [1, 2, 3, 4, 5], Vision-Language-Action (VLA) models enable transformative embodied intelligence. These systems, such as OpenVLA [6], CogACT [7],  $\pi_0$  [8] and RT-2 [?], directly translate multimodal inputs into executable actions, successfully tackling complex robotic manipulation and reasoning tasks using large-scale datasets [9, 10]. Many cutting-edge VLAs couple a Vision-Language Model (VLM) for scene and instruction parsing with a diffusion model to handle multi-modal action distribution [7, 11, 12, 13]. However, the significant computational and memory overheads of these Diffusion-based VLA architectures during the inference time pose critical barriers to their practical deployment, particularly for real-time interaction on resource-constrained robotic platforms.

Diffusion-based VLA architectures typically comprise a vision encoder to extract features, a large language model (LLM) [14, 15, 16, 17] core for multimodal reasoning, and a diffusion-based action

---

\*Corresponding authors.

Table 1: Module-wise inference characteristics of a baseline VLA model (CogACT, Left) compared to our proposed EfficientVLA (Right). EfficientVLA demonstrates significant improvements in overall inference speed and computational efficiency (FLOPs).

|                     | Vision Module | Language Module | Action Module |                     | Vision Module | Language Module      | Action Module       |
|---------------------|---------------|-----------------|---------------|---------------------|---------------|----------------------|---------------------|
| #Param (M)          | 802.3         | 6738.9          | 89.0          | #Param (M)          | 802.3         | <b>3971.1</b> (↓41%) | 89.0                |
| Vision Token        | 256           | 256             | -             | Vision Token        | 256           | <b>56</b> (↓78%)     | -                   |
| Denoising Steps     | -             | -               | 10            | Denoising Steps     | -             | -                    | <b>2</b> (↓80%)     |
| Inference Time (ms) | 24.9          | 134.5           | 51.5          | Inference Time (ms) | 24.9          | <b>58.9</b> (↓56%)   | <b>26.2</b> (↓49%)  |
| FLOPs (G)           | 405.50        | 3726.55         | 57.96         | FLOPs (G)           | 405.50        | <b>792.58</b> (↓78%) | <b>11.72</b> (↓80%) |

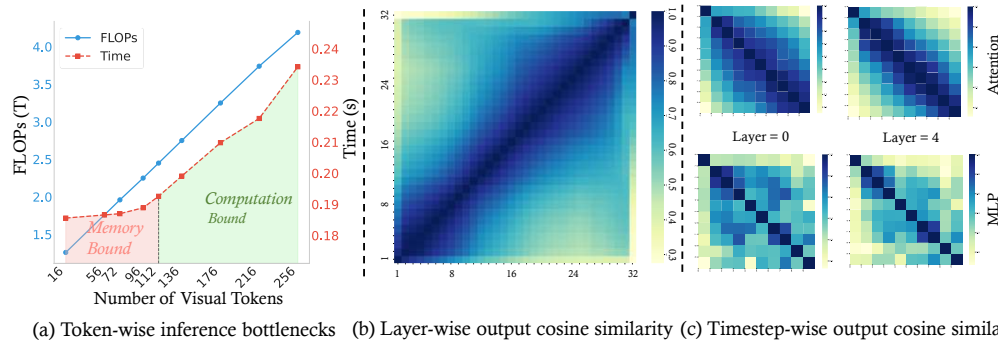


Figure 1: VLA inference bottleneck and redundancy analysis: **(a)** Visual token pruning impact on FLOPs and inference time, revealing computation-bound and memory-bound regimes. **(b)** High inter-layer cosine similarity of LLM hidden states, indicating depth-wise redundancy. **(c)** Temporal cosine similarity of MLP/attention features in diffusion steps, showing computational redundancy.

decoder to predicts the final actions through multiple denoising steps. While this modular design underpins their powerful capabilities, it inherently results in substantial computational and memory overhead. Our findings (Table 1) indicate that the language module and the iterative diffusion head are primary contributors to overall latency and computational load. Furthermore, as illustrated in Figure 1 (a), while visual token pruning initially reduces inference time in computation-bound scenarios, its efficacy quickly diminishes as the system becomes memory-bound by the LLM.

Prior VLA acceleration efforts have largely focused on isolated tweaks, delivering minimal overall gains. These fragmented approaches often fail because they ignore the integrated nature of VLA, where optimizing one module in isolation merely shifts bottlenecks. Gains are limited by unaddressed inefficiencies elsewhere, such as the memory demands of LLM or the computational intensity of action head. For example, methods like TinyVLA [13] and DeeR-VLA [18] focus on specialized model architectures rather than broadly applicable inference acceleration frameworks for pre-trained VLAs. Other approaches, such as Mole-VLA [19], tackle LLM layer redundancy but require costly retraining and overlook other pipeline stages. Similarly, VLA-Cache [20] caches static visual tokens but provides limited speedup, constrained by the significant memory footprint of LLM and computational demands of the action head. Consequently, these existing approaches fall short of providing a truly holistic solution to navigate the complex landscape of VLA inefficiencies.

To develop a more effective acceleration strategy, we systematically analyze the inference characteristics and multifaceted redundancies within each VLA module. In many Diffusion-based VLAs, the diffusion action head operates as a separate module, guided by features extracted from the VLM. This separation may underutilize the full reasoning capacity of VLM for action generation, questioning the necessity of its entire scale. As illustrated in Figure 1 (b), the language module demonstrates shows considerable depth-wise representational redundancy with high inter-layer hidden state similarity. The visual processing pathway exacerbates this issue by processing superfluous tokens, characterized by low task-relevance or high informational overlap due to visual similarity, which strains computational resources and intensifies the memory-bound condition of LLM. As shown in Figure 1 (c), the iterative diffusion action head displays significant temporal redundancy. The high similarity of its intermediate features across adjacent denoising steps implies extensive and near-static recomputations.

Motivated by this, we introduce EfficientVLA, a structured, training-free acceleration framework for Diffusion-based VLAs that systematically targets these issues. Using a similarity-derived importance metric to target the primary memory bottleneck of the language module and its observed depth-wise

redundancy (Figure 1 (b)), EfficientVLA employs a similarity-derived importance metric to prune functionally inconsequential layers, thus reducing the depth of the model and the demands for memory without retraining. To manage the initial computational load from visual inputs before the memory of LLM limit is reached (Figure 1 (a)), our visual token pruning strategy tackles both task-relevant and inherent image redundancies by first selecting critical task-aligned tokens, then augmenting this set to ensure representational diversity while maintaining high task relevance. Lastly, EfficientVLA addresses temporal redundancy in the compute-intensive action generator (highlighted by high feature similarity across timesteps, Figure 1 (c)) by caching and reusing intermediate attention and MLP outputs, thus curtailing redundant computations. This synergistic, structured approach provides a more holistic alleviation of GPU compute and memory bottlenecks than isolated optimizations.

The main contributions of this work are summarized as follows:

1. We present a systematic analysis identifying critical computational and memory-bound bottlenecks, alongside multifaceted redundancies within contemporary Diffusion-based Vision-Language-Action (VLA) architectures, thereby motivating the need for structured acceleration.
2. We propose *EfficientVLA*, a novel training-free, structured inference acceleration framework that synergistically prunes redundant layers from the language module based on their informational impact and strategically selects a compact, task-focused subset of visual tokens by considering both VLA task relevance and inherent image feature diversity.
3. Our framework further enhances efficiency by exploiting temporal redundancies in the diffusion-based action head, introducing a caching mechanism for intermediate attention and MLP computations during the iterative denoising process.
4. We demonstrate the efficacy of EfficientVLA through extensive experiments on the CogACT in the SIMPLER environment [21], achieving a  $1.93\times$  inference speedup and reducing FLOPs to 28.9%, all while incurring a minimal accuracy degradation of only 0.6%. This will facilitate the application of large-scale VLAs on the resource-constrained robotics platforms in the real world.

## 2 Related Work

**Vision-Language-Action (VLA) Models.** Vision-Language-Action models [6, 18, 22, 23, 24, 25, 26, 27] extend Vision-Language Models (VLMs) [3, 28, 29, 30, 31] by incorporating action generation, bridging the gap between perception and action. These models enable machines to understand visual and textual inputs and generate corresponding actions for tasks [22, 32] such as robotic manipulation and object retrieval. VLA models typically use pretrained VLMs [29] to encode visual and linguistic data into a shared representation, from which actions are generated either as discrete tokens or continuous values. A prominent recent trend within VLA is the adoption of diffusion models for generating coherent continuous action sequences. This paradigm is exemplified by models such as CogACT [7], DexVLA [12], DiVLA [11],  $\pi_0$  [8], and TinyVLA [13]. Many of these diffusion-based VLAs employ a componentized design: the foundational VLM processes visual and linguistic inputs to produce a condensed feature representation, which then conditions a distinct diffusion-based action module responsible for the iterative generation of precise action trajectories. This often involves the VLM output steering the denoising process within the specialized action decoder.

**Efficient Vision-Language-Action Models.** The computational complexity of VLMs [33, 34, 35, 36] poses significant challenges for their real-time deployment, particularly in applications such as robotic control that require rapid decision-making. To address this issue, recent efforts to accelerate VLA models have been primarily categorized into training-aware and training-free methods. Training-aware approaches, such as RoboMamba [37], EfficientVLM [38], and DeeR-VLA [18], focus on optimizing model architectures or applying compression techniques followed by retraining, achieving significant speedups while maintaining performance. For instance, DeeR-VLA reduces computational costs by leveraging dynamic reparameterization and efficient pruning strategies, which enable more flexible and scalable model deployment. Similarly, For example, Mole-VLA [19] reduces computational costs by dynamically activating only a subset of model layers based on task-specific needs. In contrast, training-free methods, such as VLA-Cache [20], enhance efficiency by reusing previously computed results for unchanged tokens between consecutive frames, which is particularly beneficial in scenarios with minimal variation in visual input.

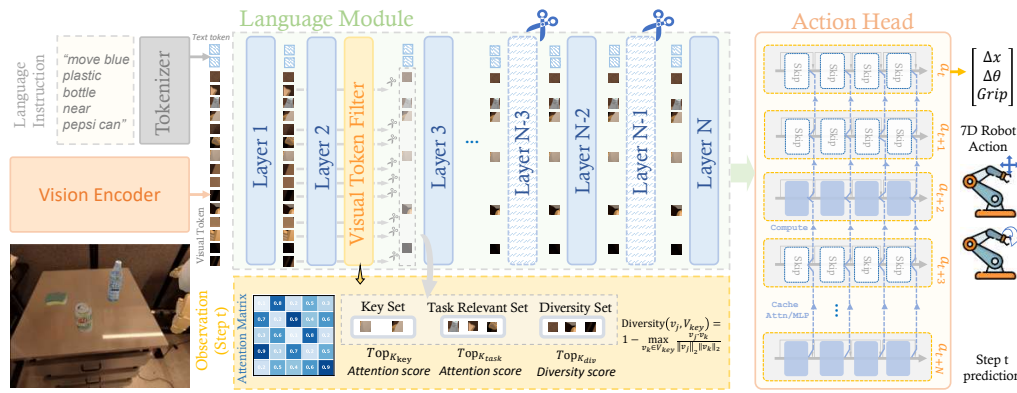


Figure 2: Overview of the **EfficientVLA** framework, our training-free, structured approach to accelerate Diffusion-based VLAs. It employs: (1) pruning of redundant language module layers; (2) VLA task-aware visual token selection balancing task relevance and informational diversity; and (3) temporal caching of intermediate features in the diffusion action head.

## 3 Method

### 3.1 Preliminaries: Vision-Language-Action Models

Vision-Language-Action (VLA) models represent a class of multimodal systems designed to bridge perception, language understanding, and robotic action. These models typically process image observations and natural language instructions through a sequence of specialized modules to generate executable action sequences. The initial stage of our basic VLA model employs a Vision Module, comprising powerful pre-trained encoders DINOv2 [39] and SigLIP [40], to transform the raw visual input  $O_{img}$  into a set of rich feature embeddings  $F_V$ . These visual features  $F_V$ , along with tokenized language instructions, are then ingested by a language model backbone. This LLM performs multimodal fusion and contextual reasoning to derive a task-oriented representation or conditioning signal,  $F_{VL}$ , which encapsulates the understanding of the scene and the instructed goal. Finally, a Diffusion-based Action Head takes the cognition feature extracted from the output feature  $F_{VL}$  as input and predicts the final action space of a gripper with 7 degrees of freedom (DoF).

### 3.2 Vision-Language Model Pruning

#### 3.2.1 Layer Redundancy Analysis

The language module within VLA models, typically a multi-layer Transformer decoder, is critical for multimodal reasoning but often introduces substantial computational overhead. Each layer  $\ell$  in such a transformer updates its input hidden state  $\mathbf{x}^{(\ell)} \in \mathbb{R}^{d \times S}$  via a residual transformation:  $\mathbf{x}^{(\ell+1)} = \mathbf{x}^{(\ell)} + f(\mathbf{x}^{(\ell)}, \theta^{(\ell)})$ , where  $f(\cdot)$  is the layer-specific function with parameters  $\theta^{(\ell)}$ ,  $d$  is the hidden dimension, and  $S$  is the sequence length. Our empirical analysis, illustrated in Figure 1 (b), reveals significant depth-wise representational redundancy within this language module component. Specifically, we observe high cosine similarity between the input  $\mathbf{x}^{(\ell)}$  and output  $\mathbf{x}^{(\ell+1)}$  states for numerous, particularly deeper layers. This indicates that the effective transformation  $f(\mathbf{x}^{(\ell)}, \theta^{(\ell)})$  imparted by these layers is minimal, rendering them functionally less critical and prime candidates for pruning to enhance inference efficiency with negligible impact on task performance.

#### 3.2.2 Importance-Driven Non-Contiguous Layer Pruning

To address the identified depth-wise redundancy within the language module of VLA models, we first rigorously quantify the functional importance of each layer. Our approach aims to identify layers that contribute minimally to the transformation of hidden state representations, rendering them candidates for pruning. We define the importance score  $I^{(\ell)}$  for a given layer  $\ell$  based on the principle that a layer effecting substantial change to its input is more critical than one whose output closely mirrors its input. Specifically,  $I^{(\ell)}$  is quantified as one minus the average cosine similarity between its input and output hidden states across a representative dataset  $\mathcal{D}$  of VLA training samples and all  $L$  token

positions within each sample:

$$I^{(\ell)} = 1 - \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \left( \frac{1}{L} \sum_{j=1}^L \frac{\mathbf{x}_{i,j}^{(\ell)} \cdot \mathbf{x}_{i,j}^{(\ell+1)}}{\|\mathbf{x}_{i,j}^{(\ell)}\|_2 \|\mathbf{x}_{i,j}^{(\ell+1)}\|_2} \right) \quad (1)$$

where  $\mathbf{x}_{i,j}^{(\ell)}, \mathbf{x}_{i,j}^{(\ell+1)} \in \mathbb{R}^d$  denote the input and output hidden state vectors, respectively, at position  $j$  of sample  $i$  for layer  $\ell$ . A high cosine similarity signifies a minimal transformative effect by the layer function  $f(\mathbf{x}^{(\ell)}, \theta^{(\ell)})$ , resulting in a low importance score  $I^{(\ell)}$  and indicating functional redundancy.

Based on these importance scores, we employ a non-contiguous pruning strategy. For an LLM comprising  $N$  layers, the importance score  $I^{(\ell)}$  is computed for every layer  $\ell \in \{1, \dots, N\}$ . These scores are then sorted in ascending order, yielding an ordered list of layer indices  $\mathcal{L}_{ranked} = [\ell_{(1)}, \ell_{(2)}, \dots, \ell_{(N)}]$  such that  $I^{(\ell_{(1)})} \leq I^{(\ell_{(2)})} \leq \dots \leq I^{(\ell_{(N)})}$ . Subsequently, the first  $n$  layers from this list,  $\{\ell_{(1)}, \ell_{(2)}, \dots, \ell_{(n)}\}$ , are selected for removal from the model.

### 3.3 Task-Relevance and Diversity-Driven Visual Token Pruning

Visual token streams processed by VLA models, despite their rich informational content, frequently exhibit significant redundancy, imposing substantial computational and memory overhead. This redundancy typically manifests in two primary forms: (i) tokens possessing low relevance to the specific VLA task objectives and (ii) tokens that are informationally duplicative due to inherent visual similarities within the input. To counteract these distinct forms of superfluity, we introduce a novel, training-free, VLA task-aware visual token pruning methodology. Our approach strategically distills a compact yet maximally informative subset of visual tokens,  $V_{pruned} \subset V$  of a predetermined size  $K_{final}$  (from an initial set of  $N_{total}$  token embeddings  $V = \{v_1, v_2, \dots, v_{N_{total}}\}$  derived from the input image). This is achieved by first anchoring the selection with task-critical tokens identified via attention analysis, and subsequently augmenting this core set by judiciously balancing continued task relevance with the explicit promotion of feature diversity through similarity measures. The retained visual tokens for inference can be found in the supplementary material.

#### 3.3.1 Quantifying Task Relevance

To guide visual token pruning, we quantify the task relevance for each initial visual token  $v_i$  (from a set of  $N_{total}$ ) by leveraging cross-attention scores from selected VLM layers. These scores capture the attention from  $v_i$  towards  $L_{ctx}$  task-defining contextual embeddings (e.g., language instructions). Let  $A_{i,j}^{(h)}$  denote the attention from visual token  $v_i$  to the  $j^{th}$  contextual token in the  $h^{th}$  attention head (of  $H$  total heads). The raw task relevance score  $r_i$  for  $v_i$  is computed by first averaging attention contributions across all  $H$  heads for each visual-contextual pair  $(i, j)$ , and then summing these averaged attentions over all  $L_{ctx}$  contextual elements:

$$r_i = \sum_{j=1}^{L_{ctx}} \left( \frac{1}{H} \sum_{h=1}^H A_{i,j}^{(h)} \right) \quad (2)$$

These raw scores  $r_i$ , signifying each token's overall engagement with the task context, are subsequently normalized (e.g., via min-max scaling) to standardized scores  $s_i \in [0, 1]$  for robust comparison and subsequent token selection.

#### 3.3.2 Selection of Key Task-Relevant Tokens

Armed with the normalized task relevance scores  $\{s_i\}$ , the first phase of pruning identifies an initial set of  $K_{key}$  visual tokens (e.g.,  $K_{key}$  empirically set between 4 and 8) that demonstrate the highest relevance to the VLA task. These tokens constitute the core and indispensable visual token set,  $V_{key}$ :

$$V_{key} = \{v_i \in V \mid s_i \text{ is among the top } K_{key} \text{ scores in } \{s_k\}_{k=1}^{N_{total}}\} \quad (3)$$

The tokens in  $V_{key}$  are unconditionally retained in  $V_{pruned}$ , forming a foundational scaffold of visual cues deemed essential for task comprehension and successful execution. The set of remaining candidate tokens for further consideration is denoted as  $V_{rem} = V \setminus V_{key}$ .

### 3.3.3 Augmentative Selection Balancing Relevance and Diversity

To supplement the core set  $V_{key}$  and achieve the target final token count  $K_{final}$ , an additional  $K_{aug} = K_{final} - K_{key}$  tokens are meticulously selected from  $V_{rem}$ . This crucial augmentation phase is guided by a ratio  $\alpha \in [0, 1]$ , which orchestrates a hybrid selection strategy that concurrently promotes continued emphasis on task relevance and the introduction of informational diversity.

**Task-Driven Augmentation.** A fraction of the augmentation quota, specifically  $K_{task} = \lfloor \alpha \cdot K_{aug} \rfloor$  tokens, is selected from  $V_{rem}$  by further prioritizing tokens based on their high task relevance scores  $s_i$ .  $V_{task}$  reinforces the task-centric nature of the pruned representation by incorporating additional tokens that, while not part of the initial  $K_{key}$  elite, still exhibit strong relevance signals. These tokens are added to the selection, and the pool of remaining candidates is updated:  $V_{rem} \leftarrow V_{rem} \setminus V_{task}$ .

**Diversity-Driven Augmentation.** The remaining  $K_{div} = K_{aug} - K_{task}$  tokens are selected from the updated  $V_{rem}$  with the explicit objective of maximizing feature diversity relative to the key selected tokens. This step is vital for capturing a broader spectrum of visual information and mitigating inherent redundancies not addressed by task relevance alone. For each candidate token  $v_j \in V_{rem}$ , its dissimilarity to the set  $V_{key}$  is computed. A common measure is the cosine distance, ensuring that selected tokens are distinct in the embedding space:

$$\text{Diversity}(v_j, V_{key}) = 1 - \max_{v_k \in V_{key}} \frac{v_j \cdot v_k}{\|v_j\|_2 \|v_k\|_2} \quad (4)$$

The  $K_{div}$  tokens from  $V_{rem}$  exhibiting the highest dissimilarity scores (i.e., those maximally different from already selected tokens) are chosen to form the set  $V_{div}$ . This targeted inclusion of diverse tokens ensures the final selection is not overly specialized and retains a richer contextual understanding.

**Final Pruned Visual Token Set.** The comprehensive set of visual tokens retained after pruning is the union of these strategically selected components:

$$V_{pruned} = V_{key} \cup V_{task} \cup V_{div} \quad (5)$$

This final set  $V_{pruned}$ , of cardinality  $K_{final}$ , is subsequently utilized for all downstream processing within the VLA model. This systematic reduction in visual sequence length significantly alleviates computational demands while preserving critical task-specific and diverse visual information.

## 3.4 Caching Intermediate Features in Action Prediction

Generating high-fidelity action sequences with Diffusion-based VLA models involves an iterative denoising process that demands significant computation due to repeated self-attention and MLP computations over  $T$  timesteps. We observe strong temporal coherence in the intermediate features produced during action generation (Figure 1 (c)), indicating substantial redundancy across timesteps. To address this inefficiency and accelerate the action generation phase, we propose a static caching mechanism. This strategy periodically recomputes and caches critical intermediate attention and MLP output at a fixed interval  $N$ , reusing these cached values for the intervening time steps in the generation of action sequences. This selective computation aims to significantly reduce the computational cost associated with generating the action sequence while preserving its quality.

### 3.4.1 Feature Generation and Temporal Coherence in DiT Blocks

Let  $t$  denote the current denoising timestep, typically iterating from an initial  $T_{start}$  down to 1. Within each DiT block at timestep  $t$ , the input features  $\mathbf{z}_t$  (which may incorporate cognitive features  $\mathbf{f}_t$  from upstream VLM modules and the current noise estimate) are processed sequentially by a self-attention module and an MLP module to produce intermediate hidden states:

$$\mathbf{h}_t^{\text{attn}} = \text{Self-Attn}(\mathbf{z}_t) \quad (6)$$

$$\mathbf{h}_t^{\text{mlp}} = \text{MLP}(\mathbf{h}_t^{\text{attn}} + \mathbf{z}_t) \quad (7)$$

These features,  $\mathbf{h}_t^{\text{attn}}$  and  $\mathbf{h}_t^{\text{mlp}}$ , are fundamental to the denoising capacity of the diffusion model. Our observation of their high temporal coherence—meaning  $\mathbf{h}_t^{\text{module}} \approx \mathbf{h}_{t-1}^{\text{module}}$  for many  $t$  and module types—motivates their periodic caching and reuse.

Table 2: Performance of EfficientVLA on the CogACT versus the other baselines in the SIMPLER environment. Settings vary by retained LLM layers (L) and visual tokens (T). *Random Dropping* denotes a method involving the random retention of 112 visual tokens.

| SIMPLER             | Method                     | Training-free | PickCan | MoveNear | Drawer | DrawerApple | Average | FLOPs↓ | Speedup↑ | Params (B) |
|---------------------|----------------------------|---------------|---------|----------|--------|-------------|---------|--------|----------|------------|
| Visual Matching     | CogACT                     | -             | 91.3%   | 85.0%    | 71.8%  | 50.9%       | 74.8%   | 100.0% | 1.00×    | 7.63       |
|                     | Random Dropping            | ✓             | 9.7%    | 20.4%    | 53.5%  | 0.0%        | 20.9%   | 58.5%  | 1.20×    | 7.63       |
|                     | FastV                      | ✓             | 92.6%   | 81.4%    | 69.8%  | 52.4%       | 74.1%   | 42.0%  | 1.21×    | 7.63       |
|                     | VLA-Cache                  | ✓             | 92.0%   | 83.3%    | 70.5%  | 51.6%       | 74.4%   | 80.1%  | 1.38×    | 7.63       |
|                     | EfficientVLA (L=28, T=112) | ✓             | 95.3%   | 83.3%    | 70.3%  | 56.5%       | 76.4%   | 45.1%  | 1.59×    | 5.87       |
|                     | EfficientVLA (L=28, T=56)  | ✓             | 94.7%   | 82.4%    | 69.8%  | 55.4%       | 75.5%   | 32.9%  | 1.71×    | 5.87       |
|                     | EfficientVLA (L=22, T=112) | ✓             | 94.0%   | 82.1%    | 69.2%  | 54.6%       | 75.0%   | 38.2%  | 1.78×    | 4.86       |
| Variant Aggregation | EfficientVLA (L=22, T=56)  | ✓             | 93.3%   | 81.3%    | 68.2%  | 53.8%       | 74.2%   | 28.9%  | 1.93×    | 4.86       |
|                     | CogACT                     | -             | 89.6%   | 80.8%    | 28.3%  | 46.6%       | 61.3%   | 100.0% | 1.00×    | 7.63       |
|                     | Random Dropping            | ✓             | 4.0%    | 16.1%    | 15.6%  | 0.0%        | 8.9%    | 58.5%  | 1.20×    | 7.63       |
|                     | FastV                      | ✓             | 91.4%   | 78.6%    | 27.6%  | 50.6%       | 62.1%   | 42.0%  | 1.19×    | 7.63       |
|                     | VLA-Cache                  | ✓             | 91.7%   | 79.3%    | 32.5%  | 45.8%       | 62.3%   | 82.6%  | 1.37×    | 7.63       |
|                     | EfficientVLA(L=28, T=112)  | ✓             | 94.8%   | 77.6%    | 28.4%  | 51.9%       | 63.2%   | 45.1%  | 1.57×    | 5.87       |
|                     | EfficientVLA (L=28, T=56)  | ✓             | 94.4%   | 77.2%    | 27.6%  | 51.3%       | 62.6%   | 32.9%  | 1.69×    | 5.87       |
|                     | EfficientVLA (L=22, T=112) | ✓             | 93.9%   | 76.4%    | 27.3%  | 50.6%       | 62.1%   | 38.2%  | 1.76×    | 4.86       |
|                     | EfficientVLA (L=22, T=56)  | ✓             | 93.2%   | 75.8%    | 26.9%  | 49.2%       | 61.2%   | 28.9%  | 1.91×    | 4.86       |

### 3.4.2 Static N-Step Caching Implementation

We define a cache interval  $N$  ( $1 \leq N < T_{start}$ ). At the initial timestep  $t = T_{start}$ , the features  $\mathbf{h}_{T_{start}}^{attn}$  and  $\mathbf{h}_{T_{start}}^{mlp}$  are computed via Equations 6 and 7 and stored in a persistent cache, denoted  $\mathcal{C}_{attn}$  and  $\mathcal{C}_{mlp}$ . For any subsequent timestep  $t < T_{start}$ , these features are recomputed and the caches are updated if and only if  $t \pmod N = 0$  (assuming  $t > 0$  and  $t$  aligns with desired multiples for caching, e.g.,  $T_{start}, T_{start} - N, T_{start} - 2N, \dots$ ). Thus, for such recomputation timesteps:

$$\mathcal{C}_{attn} \leftarrow \text{Self-Attn}(\mathbf{z}_t) \quad (8)$$

$$\mathcal{C}_{mlp} \leftarrow \text{MLP}(\mathcal{C}_{attn} + \mathbf{z}_t) \quad (9)$$

And the outputs for this step are  $\mathbf{h}_t^{attn} = \mathcal{C}_{attn}$  and  $\mathbf{h}_t^{mlp} = \mathcal{C}_{mlp}$ . In all other timesteps, where  $t \pmod N \neq 0$ , the computationally intensive Self-Attn and MLP operations are entirely bypassed. Instead, the required features are directly retrieved from the most recently populated cache:

$$\mathbf{h}_t^{attn} \leftarrow \mathcal{C}_{attn} \quad (\text{when } t \pmod N \neq 0) \quad (10)$$

$$\mathbf{h}_t^{mlp} \leftarrow \mathcal{C}_{mlp} \quad (\text{when } t \pmod N \neq 0) \quad (11)$$

This static caching schedule effectively prunes the execution of these core modules for  $N - 1$  out of every  $N$  timesteps post-initialization, leading to a substantial reduction in floating-point operations and latency for the action generation component of the VLA. The choice of  $N$  allows for a tunable trade-off between acceleration and the fidelity of the generated actions, as reusing features for longer intervals might introduce slight deviations if underlying representations were to change rapidly.

## 4 Experiment

### 4.1 Experimental Settings

**Simulation Implementation Details.** To assess our VLA model, we utilize the SIMPLER environment [21], a simulation-based benchmark for table-top manipulation. SIMPLER is designed to closely emulate real-world dynamics for robots such as the Google Robot and WidowX, demonstrating robust alignment between simulation and real-world performance. The VLA model in this setup takes  $224 \times 224$  RGB image observations and natural language task instructions (e.g., "Pick coke can") as input and outputs a sequence of actions in 7-DoF Cartesian space. The SIMPLER supports two evaluation configurations: *Visual Matching*, which prioritizes fidelity to real-world appearances, and *Variant Aggregations*, which incorporates diverse conditions such as altered lighting, backgrounds, and surface textures. For the Google robot, SIMPLER provides both two evaluation settings, each featuring the same four tasks: 1) Pick coke can; 2) Move near; 3) Open/close drawer; and 4) Open top drawer and place apple. Success rate is used as the evaluation metric.

**Baselines.** Our primary experimental validation of EfficientVLA is performed on the CogACT [41], which integrates powerful vision encoders (DINOv2 [39] and SigLIP [40]), a Llama2-7B [14]

Table 3: **Scalability Analysis:** We evaluate mean success rate and inference time in a simulation environment of visual matching across various model sizes. Our EfficientVLA configuration maintains  $L = 22$  and  $T = 56$ .

| SIMPLER  | Model        | Action-Params | Methods      | Average | Inference time (s) | Total-Params (B) |
|----------|--------------|---------------|--------------|---------|--------------------|------------------|
| Visual   | CogACT-Small | 13M           | CogACT       | 73.3%   | 0.2156             | 7.55             |
|          |              |               | EfficientVLA | 72.6%   | 0.1173             | 4.78             |
| Matching | CogACT-Base  | 89M           | CogACT       | 74.8%   | 0.2342             | 7.63             |
|          |              |               | EfficientVLA | 74.2%   | 0.1213             | 4.86             |
|          | CogACT-Large | 308M          | CogACT       | 76.7%   | 0.2628             | 7.85             |
|          |              |               | EfficientVLA | 76.1%   | 0.1312             | 5.08             |

language module for multimodal reasoning, and a Diffusion Transformer (DiT) for generating action trajectories. We benchmark against relevant baseline methodologies. These include a *Random Dropping* approach, where 112 visual tokens are retained uniformly at random, to evaluate the benefits of our guided vision token pruning. We further compare with *FastV* [42], a notable approach focused on accelerating inference by pruning redundant visual tokens, and *VLA-Cache* [20], which leverages temporal analysis to cache static tokens across timesteps. The experimental results and model details for  $\pi_0$  on the LIBERO [43] can be found in the Appendix.

**Implementation Details.** For EfficientVLA, in addition to layer pruning, we further compressed the model parameters by adopting the PruneNet [44] configuration for LLM compression. Specifically, we applied a sparsity of 25% to the MLP layers of all Transformer blocks. For visual token pruning, we started from the 2nd Transformer layer with a ratio  $\alpha = 50\%$  and  $K_{key} = 4$  for key task-critical tokens. Furthermore, the cache interval was set to 5. All experiments were conducted on NVIDIA A40 GPUs, and the inference time was measured as the average single-step inference duration. More details can be found in the supplementary material.

## 4.2 Results on Simulation Environment

**Main Results on SIMPLER.** Table 2 details the performance of our structured, entirely training-free pruning method in the SIMPLER environment. Our approach consistently excels across configurations retaining 22/28 layers and 56/112 visual tokens. For instance, pruning 10 layers with 112 tokens surpasses both CogACT and VLA-cache in success rate and inference speed. Remarkably, on the *pick coke can* task, pruning 36% of parameters paradoxically improved the success rate from 91.3% to 94.0%, highlighting significant parameter redundancy in the VLA model. Conversely, random token dropping to 112 tokens drastically reduces the average success rate to 20.9%, affirming the superiority of our guided selection strategy. Furthermore, a 22-layer, 56-token setup achieved a 71.1% reduction in FLOPs with merely a 0.6% drop in average success rate, demonstrating exceptional efficiency. In comparison, approaches like FastV [42] ( $T = 56$ ) show that solely optimizing visual tokens yields only a  $1.21\times$  speedup due to unaddressed memory bottlenecks, despite acceptable task performance.

**Efficiency Analysis.** As demonstrated in Table 2 and Figure 3, our proposed method significantly outperforms previous baselines, achieving a 71.1% reduction in FLOPs and a  $1.93\times$  speedup in inference time. In stark contrast, VLA-cache, when applied to the CogACT model, only reduces FLOPs by 19.9% and achieves a mere  $1.38\times$  speedup. This disparity substantiates our prior analysis: VLA-cache, functioning solely as a cache for visual tokens between adjacent time steps, is inherently constrained by memory bounds, thereby limiting the efficacy of token-only acceleration. Consequently, the structured framework of our system offers distinct advantages, highlighting our method’s superior capability in balancing computational efficiency with robust performance.

**Scalability Evaluation.** Table 3 illustrates the scalability of our proposed method across different sizes of the CogACT model. With the primary difference among the three models being the parameter size of their action modules, the results reveal that our method’s effectiveness becomes more pronounced with larger models. Specifically, on the CogACT-Large model, our approach achieves a  $2.0\times$  inference speedup, while performance only marginally decreases from 76.7% to 76.1%. This increased impact is because action modules in larger models, having more parameters, inherently

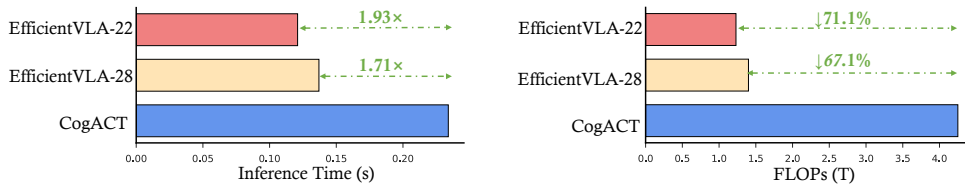


Figure 3: **Efficiency analysis** in simulation, comparing FLOPs and inference time of our EfficientVLA variants against the original model backbone. EfficientVLA-22 and EfficientVLA-28 denote configurations retaining 22 and 28 LLM layers, respectively.

Table 4: Performance impact of varying visual token reduction ratios (Left) and diffusion action head cache intervals (Right) as applied within our EfficientVLA framework.

| Token Ratio        | 56<br>77.8% | 72<br>71.8% | 96<br>62.5% | 112<br>56.2% | 256<br>100.0% |
|--------------------|-------------|-------------|-------------|--------------|---------------|
| Accuracy           | 95.0%       | 95.3%       | 95.0%       | 96.0%        | 91.3%         |
| Inference time (s) | 0.1866      | 0.1870      | 0.1889      | 0.1956       | 0.2342        |
| FLOPs (T)          | 1.76        | 1.96        | 2.25        | 2.45         | 4.19          |

| Cache Interval     | 1      | 2      | 3      | 4      | 5      |
|--------------------|--------|--------|--------|--------|--------|
| Accuracy           | 91.3%  | 94.0%  | 93.7%  | 90.3%  | 93.7%  |
| Inference time (s) | 0.2342 | 0.2031 | 0.1987 | 0.1953 | 0.1909 |
| FLOPs (T)          | 4.190  | 4.161  | 4.155  | 4.150  | 4.144  |

Table 5: **Ablation study** on our EfficientVLA, where 'Layer' denotes applying only the LLM layer pruning component, and 'MLP' refers to a distinct strategy of compressing 25% of MLP weights within each layer.

|     | Model Compression |     | Visual Token | Action | Success      | Inference     | Speedup $\uparrow$             |
|-----|-------------------|-----|--------------|--------|--------------|---------------|--------------------------------|
|     | Layer             | MLP | Pruning      | Cache  | Rate         | Time (s)      |                                |
| Ex0 | ✗                 | ✗   | ✗            | ✗      | 91.3%        | 0.2342        | 1.00 $\times$                  |
| Ex1 | ✗                 | ✗   | ✓            | ✗      | 95.6%        | 0.1866        | 1.25 $\times$                  |
| Ex2 | ✗                 | ✗   | ✗            | ✓      | 93.7%        | 0.1909        | 1.23 $\times$                  |
| Ex3 | ✓                 | ✗   | ✓            | ✗      | 85.7%        | 0.1604        | 1.46 $\times$                  |
| Ex4 | ✓                 | ✓   | ✗            | ✗      | 92.3%        | 0.1638        | 1.43 $\times$                  |
| Ex5 | ✓                 | ✓   | ✗            | ✓      | 93.3%        | 0.1387        | 1.69 $\times$                  |
| Ex6 | ✗                 | ✗   | ✓            | ✓      | <b>95.3%</b> | 0.1592        | 1.47 $\times$                  |
| Ex7 | ✓                 | ✓   | ✓            | ✓      | 93.3%        | <b>0.1213</b> | <b>1.93<math>\times</math></b> |

exhibit longer inference times, thus allowing our method to yield more significant accelerations. These findings also underscore the robustness of our method across models of various scales.

**Impact of Token Reduction Ratio and Cache Interval.** Table 4 details our experiments on the *pick coke can* task reveal distinct impacts of component-specific optimizations. Aggressively pruning visual tokens, down to retaining only 22%, effectively lessens computational load for substantial, near-lossless acceleration. However, inference speed gains largely saturate beyond this point, with further token reduction yielding only marginal improvements, thereby revealing dominant system-level performance bottlenecks. Separately, for the diffusion-based action generator, we observed that increasing the cache reuse interval  $N$  for intermediate attention and MLP features progressively and significantly accelerates action trajectory generation.

### 4.3 Ablation Study

We conducted an ablation study on the components of our proposed framework, using the *pick coke can* task as an illustrative example. As suggested by prior analysis (e.g., Figure 1 (a), solely optimizing visual tokens for VLA inference tasks yields limited acceleration; retaining just 56 tokens resulted in a mere 1.23 $\times$  speedup, although the success rate paradoxically rose from 91.3% to 95.6%. This underscores the inherent limitations of token-centric optimization methods, such as VLA-cache, and affirms that achieving substantial VLA inference acceleration viable for hardware deployment necessitates a more model-centric strategy. In contrast, our model compression approach—concurrently pruning layers and compressing MLP in the remaining layers—achieved a 1.43 $\times$  speedup. Critically, when all components were integrated, a 1.93 $\times$  speedup was realized, and the overall task success rate still saw an improvement of 2% points. These collective results highlight the imperative and significance of adopting a structured framework for effective VLA inference acceleration.

## 5 Conclusion

In this paper, we addressed the critical challenge of high computational and memory overheads that impede the practical deployment of powerful Diffusion-based Vision-Language-Action (VLA) models. We proposed EfficientVLA, a novel training-free, structured framework to accelerate VLA models. Our framework enhances efficiency by synergistically pruning redundant layers of language module identified by their minimal impact on transforming hidden states and by strategically selecting a compact set of visual tokens that balances VLA task relevance with inherent feature diversity. Furthermore, it optimizes the action module by caching critical intermediate computations across its iterative denoising steps. We demonstrate the efficacy of EfficientVLA through extensive experiments on the CogACT in the SIMPLER environment [21], achieving a  $1.93\times$  inference speedup and reducing FLOPs to 28.9%, all while incurring a minimal accuracy degradation of only 0.6%.

## References

- [1] Anas Awadalla, Irena Gao, Josh Gardner, Jack Hessel, Yusuf Hanafy, Wanrong Zhu, Kalyani Marathe, Yonatan Bitton, Samir Gadre, Shiori Sagawa, Jenia Jitsev, Simon Kornblith, Pang Wei Koh, Gabriel Ilharco, Mitchell Wortsman, and Ludwig Schmidt. Openflamingo: An open-source framework for training large autoregressive vision-language models. *arXiv preprint arXiv:2308.01390*, 2023.
- [2] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR, 2022.
- [3] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
- [4] Ruichuan An, Sihan Yang, Ming Lu, Kai Zeng, Yulin Luo, Ying Chen, Jiajun Cao, Hao Liang, Qi She, Shanghang Zhang, et al. Mc-llava: Multi-concept personalized vision-language model. *arXiv preprint arXiv:2411.11706*, 2024.
- [5] Yulin Luo, Ruichuan An, Bocheng Zou, Yiming Tang, Jiaming Liu, and Shanghang Zhang. Llm as dataset analyst: Subpopulation structure discovery with large language model. In *European Conference on Computer Vision*, pages 235–252. Springer, 2024.
- [6] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.
- [7] Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, Xiaofan Wang, Bei Liu, Jianlong Fu, Jianmin Bao, Dong Chen, Yuanchun Shi, Jiaolong Yang, and Baining Guo. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation, 2024.
- [8] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al.  $\pi_0$ : A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.
- [9] Hao-Shu Fang, Hongjie Fang, Zhenyu Tang, Jirong Liu, Chenxi Wang, Junbo Wang, Haoyi Zhu, and Cewu Lu. Rh20t: A comprehensive robotic dataset for learning diverse skills in one-shot. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 653–660. IEEE, 2024.
- [10] Abby O’Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya 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)*, pages 6892–6903. IEEE, 2024.

- [11] Junjie Wen, Minjie Zhu, Yichen Zhu, Zhibin Tang, Jinming Li, Zhongyi Zhou, Chengmeng Li, Xiaoyu Liu, Yaxin Peng, Chaomin Shen, et al. Diffusion-vla: Scaling robot foundation models via unified diffusion and autoregression. *arXiv preprint arXiv:2412.03293*, 2024.
- [12] Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control. *arXiv preprint arXiv:2502.05855*, 2025.
- [13] Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2409.12514*, 2024.
- [14] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- [15] Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694, 2020.
- [16] OpenAI et al. Gpt-4 technical report, 2024.
- [17] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.
- [18] Yang Yue, Yulin Wang, Bingyi Kang, Yizeng Han, Shenzhi Wang, Shiji Song, Jiashi Feng, and Gao 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.
- [19] Rongyu Zhang, Menghang Dong, Yuan Zhang, Liang Heng, Xiaowei Chi, Gaole Dai, Li Du, Dan Wang, Yuan Du, and Shanghang Zhang. Mole-vla: Dynamic layer-skipping vision language action model via mixture-of-layers for efficient robot manipulation. *arXiv preprint arXiv:2503.20384*, 2025.
- [20] Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. Vla-cache: Towards efficient vision-language-action model via adaptive token caching in robotic manipulation. *arXiv preprint arXiv:2502.02175*, 2025.
- [21] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, et al. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.
- [22] Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, et al. Vision-language foundation models as effective robot imitators. *arXiv preprint arXiv:2311.01378*, 2023.
- [23] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.
- [24] Wenxuan Song, Ziyang Zhou, Han Zhao, Jiayi Chen, Pengxiang Ding, Haodong Yan, Yuxin Huang, Feilong Tang, Donglin Wang, and Haoang Li. Reconvla: Reconstructive vision-language-action model as effective robot perceiver. *arXiv preprint arXiv:2508.10333*, 2025.
- [25] Zhide Zhong, Haodong Yan, Junfeng Li, Xiangchen Liu, Xin Gong, Wenxuan Song, Jiayi Chen, and Haoang Li. Flowvla: Thinking in motion with a visual chain of thought. *arXiv preprint arXiv:2508.18296*, 2025.
- [26] Can Cui, Pengxiang Ding, Wenxuan Song, Shuanghao Bai, Xinyang Tong, Zirui Ge, Runze Suo, Wanqi Zhou, Yang Liu, Bofang Jia, et al. Openhelix: A short survey, empirical analysis, and open-source dual-system vla model for robotic manipulation. *arXiv preprint arXiv:2505.03912*, 2025.

- [27] Wenxuan Song, Jiayi Chen, Wenxue Li, Xu He, Han Zhao, Can Cui, Pengxiang Ding Shiyang Su, Feilong Tang, Xuelian Cheng, Donglin Wang, et al. Rationalvla: A rational vision-language-action model with dual system. *arXiv preprint arXiv:2506.10826*, 2025.
- [28] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning, 2022.
- [29] Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models, 2024.
- [30] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation, 2022.
- [31] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023.
- [32] Anas Awadalla, Irena Gao, Josh Gardner, Jack Hessel, Yusuf Hanafy, Wanrong Zhu, Kalyani Marathe, Yonatan Bitton, Samir Gadre, Shiori Sagawa, Jenia Jitsev, Simon Kornblith, Pang Wei Koh, Gabriel Ilharco, Mitchell Wortsman, and Ludwig Schmidt. Openflamingo: An open-source framework for training large autoregressive vision-language models, 2023.
- [33] Xuyang Liu, Zichen Wen, Shaobo Wang, Junjie Chen, Zhishan Tao, Yubo Wang, Xiangqi Jin, Chang Zou, Yiyu Wang, Chenfei Liao, et al. Shifting ai efficiency from model-centric to data-centric compression. *arXiv preprint arXiv:2505.19147*, 2025.
- [34] Zichen Wen, Yifeng Gao, Shaobo Wang, Junyuan Zhang, Qintong Zhang, Weijia Li, Conghui He, and Linfeng Zhang. Stop looking for important tokens in multimodal language models: Duplication matters more. *arXiv preprint arXiv:2502.11494*, 2025.
- [35] Wenxuan Song, Jiayi Chen, Pengxiang Ding, Yuxin Huang, Han Zhao, Donglin Wang, and Haoang Li. Ceed-vla: Consistency vision-language-action model with early-exit decoding. *arXiv preprint arXiv:2506.13725*, 2025.
- [36] Wenxuan Song, Jiayi Chen, Pengxiang Ding, Han Zhao, Wei Zhao, Zhide Zhong, Zongyuan Ge, Jun Ma, and Haoang Li. Accelerating vision-language-action model integrated with action chunking via parallel decoding. *arXiv preprint arXiv:2503.02310*, 2025.
- [37] Jiaming Liu, Mengzhen Liu, Zhenyu Wang, Pengju An, Xiaoqi Li, Kaichen Zhou, Senqiao Yang, Renrui Zhang, Yandong Guo, and Shanghang Zhang. Robomamba: Efficient vision-language-action model for robotic reasoning and manipulation, 2024.
- [38] Tiannan Wang, Wangchunshu Zhou, Yan Zeng, and Xinsong Zhang. Efficientvlm: Fast and accurate vision-language models via knowledge distillation and modal-adaptive pruning, 2022.
- [39] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.
- [40] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11975–11986, 2023.
- [41] Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, Xiaofan Wang, Bei Liu, Jianlong Fu, Jianmin Bao, Dong Chen, Yuanchun Shi, Jiaolong Yang, and Baining Guo. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation, 2024.
- [42] Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao 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*, pages 19–35. Springer, 2025.

- [43] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36, 2024.
- [44] Ayan Sengupta, Siddhant Chaudhary, and Tanmoy Chakraborty. You only prune once: Designing calibration-free model compression with policy learning. *arXiv preprint arXiv:2501.15296*, 2025.
- [45] Shuzhou Yuan, Ercong Nie, Bolei Ma, and Michael Färber. Why lift so heavy? slimming large language models by cutting off the layers. *arXiv preprint arXiv:2402.11700*, 2024.
- [46] Gui Ling, Ziyang Wang, and Qingwen Liu. Slimgpt: Layer-wise structured pruning for large language models. *Advances in Neural Information Processing Systems*, 37:107112–107137, 2024.
- [47] Lanxiang Hu, Tajana Rosing, and Hao Zhang. Trimllm: Progressive layer dropping for domain-specific llms. *arXiv preprint arXiv:2412.11242*, 2024.
- [48] Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, et al. Sparsevlm: Visual token sparsification for efficient vision-language model inference. *arXiv preprint arXiv:2410.04417*, 2024.
- [49] Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. *arXiv preprint arXiv:2403.15388*, 2024.
- [50] Ting Liu, Liangtao Shi, Richang Hong, Yue Hu, Qunjun Yin, and Linfeng Zhang. Multi-stage vision token dropping: Towards efficient multimodal large language model. *arXiv preprint arXiv:2411.10803*, 2024.
- [51] Pratheba Selvaraju, Tianyu Ding, Tianyi Chen, Ilya Zharkov, and Luming Liang. Fora: Fast-forward caching in diffusion transformer acceleration. *arXiv preprint arXiv:2407.01425*, 2024.
- [52] Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. *arXiv preprint arXiv:2410.05317*, 2024.

## NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and precede the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [NA].
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

**The checklist answers are an integral part of your paper submission.** They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes]" is generally preferable to "[No]", it is perfectly acceptable to answer "[No]" provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No]" or "[NA]" is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist",**
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers.**

### 1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: We believe that our abstract and introduction accurately reflect and highlight the contributions of our paper.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

### 2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: We have discussed the limitations and future work in supplementary material.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

### 3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: This paper focuses on experimental approaches and acceleration effect, and it does not include theoretical proofs.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

### 4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provided the detailed experimental settings in the Section 4.1 and also in the supplementary material to ensure reproducibility of our method.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

## 5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [NA]

Justification: This paper uses public datasets and models, and the specific links are given in the additional materials.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

## 6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [\[Yes\]](#)

Justification: We provided the detailed experimental settings in the Section 4.1 and also in the supplementary material to ensure reproducibility of our method.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

## 7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [\[NA\]](#)

Justification: We follow the settings of previous methods where the result of a single run is provided.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

## 8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [\[Yes\]](#)

Justification: We provided the detailed experiments compute resources in the Section 4.1. All of our experiments were conducted on a single NVIDIA A40 GPUs.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

#### 9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines?>

Answer: [Yes]

Justification: We followed the NeurIPS Code of Ethics in preparing our manuscript.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

#### 10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Provided in the supplementary material.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

#### 11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: This article is a training-free acceleration method, using open source and secure models.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

## 12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We properly cite the relevant papers and acknowledge the checkpoints we use in the supplementary material.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, [paperswithcode.com/datasets](https://paperswithcode.com/datasets) has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

## 13. New Assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: This paper proposes a training-free inference acceleration method. The specific implementation of the method has been detailed in the paper, and no new datasets or software models are introduced that are intended to be released as independent assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.

- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

#### 14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The relevant aspect of our research is solely focused on accelerating a pre-trained model.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

#### 15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The relevant aspect of our research is solely focused on accelerating a pre-trained model.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

#### 16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [NA]

Justification: Our research does not involve LLMs as any important components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

## A Experimental Settings

### A.1 SIMPLER Environment

The SIMPLER simulation environment serves as our primary benchmark for evaluating CogACT models. It is specifically designed to closely mirror real-world robotic setups, thereby facilitating more realistic evaluations and bridging the real-to-sim control and visual gap.

SIMPLER offers two distinct evaluation settings:

- **Visual Matching (VM):** This setting closely replicates real-world tasks by minimizing discrepancies between the simulated and real environments, prioritizing fidelity to real-world appearances.
- **Variant Aggregation (VA):** This setting builds upon Visual Matching by introducing variations to elements such as background, lighting, distractors, and table texture, challenging models to generalize across diverse conditions.

For the Google robot setup, SIMPLER provides both evaluation settings, each featuring the same four tasks: 1) Pick coke can, 2) Move near, 3) Open/close drawer, and 4) Open top drawer and place apple. These four tasks for the Google robot are also illustrated in Figure 4.

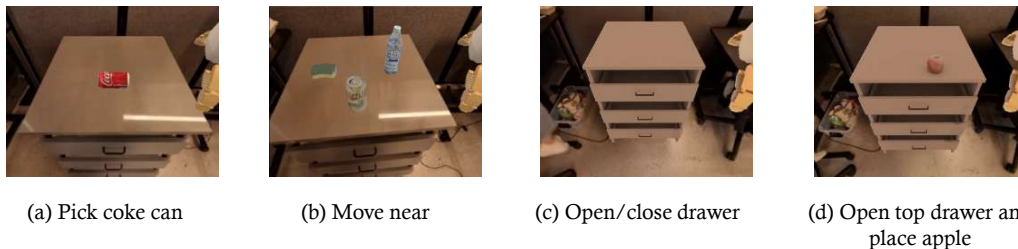


Figure 4: Representative robotic manipulation tasks for the Google robot in the SIMPLER environment: (a) Pick coke can, (b) Move near, (c) Open/close drawer, and (d) Open top drawer and place apple.

### A.2 Baselines

We benchmark our EfficientVLA against the following relevant baseline and backbone methodologies:

- **CogACT:** This model serves as our primary experimental validation platform. CogACT integrates powerful vision encoders (DINOv2 and SigLIP) to process raw visual input, a Llama2-7B language module for multimodal reasoning, and a Diffusion Transformer (DiT) as a specialized action module for generating precise action trajectories. It aims to synergize "cognition" (VLM output) with "action" capabilities by conditioning the diffusion-based action module on the VLM's extracted features, addressing the continuous, multimodal, and temporally correlated nature of robot actions.
- $\pi_0$ : This model is a prototype generalist robot policy developed by prior research to address the versatility gap between human and machine intelligence. Built upon a pre-trained vision-language model to harness Internet-scale image-text corpora for semantic reasoning and problem-solving,  $\pi_0$  is extended into a VLA model through cross-embodiment training, integrating diverse data from various robot types to enhance generalization and mitigate data scarcity. Featuring an action chunking architecture with flow matching, it enables high-frequency control up to 50 Hz for dexterous tasks such as laundry folding, and is pre-trained on over 10,000 hours of robot data before fine-tuning on curated datasets to optimize dexterity, efficiency, and robustness, serving as a robust reference for evaluating our proposed EfficientVLA framework.
- **VLA-Cache:** This is a training-free acceleration method designed to improve VLA model efficiency in robotic manipulation. VLA-Cache operates on the principle that visual inputs in

sequential robotic tasks often exhibit minimal variation between successive steps, particularly in background regions. It incorporates a token-selection mechanism that identifies visually static tokens with minimal changes from the previous step and reuses their computational results via KV-cache. Additionally, it includes a fine-grained selection scheme to filter out critical task-relevant tokens, ensuring they undergo full computation to preserve accuracy, and employs a layer-adaptive strategy to adjust reuse ratios based on attention concentration.

- **FastV:** This approach focuses on accelerating inference in Large Vision-Language Models (LVLMs) by pruning redundant visual tokens. FastV’s core insight is the identification of inefficient attention phenomena in deeper layers of popular LVLMs, where image tokens receive significantly less attention despite accounting for a large portion of input tokens. It proposes a plug-and-play method that dynamically prunes a percentage of these less impactful visual tokens after a specific layer, guided by attention scores, to reduce computational costs (FLOPs) without sacrificing performance.

## B More Experimental Results

### B.1 Results on Vision-Language Tasks

Table 6 illustrates the performance of our EfficientVLA token pruning strategy in comparison to the vanilla LLaVA-1.5-7B baseline and the conventional FastV method across a suite of multimodal vision-language benchmarks. Our approach, which employs an advanced token pruning mechanism tailored to preserve semantically critical visual tokens while aggressively reducing redundancy, consistently outperforms FastV at equivalent token retention levels. Specifically, at 192 tokens, EfficientVLA achieves an average performance of 98.0%, retaining nearly all baseline efficacy, whereas FastV drops to 91.2%; similarly, at 128 tokens, our method yields 96.6% average accuracy, markedly superior to FastV’s 86.4%. This superior retention of multimodal understanding capabilities underscores EfficientVLA’s efficacy in mitigating information loss during pruning, enabling substantial inference acceleration in vision-language tasks without compromising downstream task performance.

Table 6: Performance evaluation of the EfficientVLA token pruning strategy compared to the vanilla LLaVA-1.5-7B baseline and the traditional FastV method across diverse multimodal vision-language benchmarks.

| Method                             | Benchmarks |      |        |      |      |      |        |          |        |          | Avg    |
|------------------------------------|------------|------|--------|------|------|------|--------|----------|--------|----------|--------|
|                                    | GQA        | MMB  | MMB-CN | MME  | POPE | SQA  | VQA-v2 | VQA-Text | VizWiz | OCRBench |        |
| LLaVA-1.5-7B (Vanilla, 576 tokens) | 61.9       | 64.7 | 58.1   | 1862 | 85.9 | 69.5 | 78.5   | 58.2     | 50.0   | 297      | 100.0% |
| +FastV (192 tokens)                | 52.7       | 61.2 | 57.0   | 1612 | 64.8 | 67.3 | 67.1   | 52.5     | 50.8   | 291      | 91.2%  |
| +Ours (192 tokens)                 | 58.5       | 62.3 | 57.0   | 1823 | 82.8 | 69.6 | 76.7   | 57.6     | 51.0   | 292      | 98.0%  |
| +FastV (128 tokens)                | 49.4       | 56.1 | 56.4   | 1490 | 56.0 | 60.2 | 61.8   | 50.6     | 51.3   | 285      | 86.4%  |
| +Ours (128 tokens)                 | 58.2       | 61.5 | 56.6   | 1763 | 79.8 | 68.5 | 75.7   | 56.1     | 51.4   | 289      | 96.6%  |

### B.2 More Generalizability Experiments

To validate the generalizability of our proposed method, we conducted additional experiments on the model using the LIBERO [43] benchmark, fine-tuning it for 30k steps as a baseline and performing inference on a NVIDIA 4090. The LIBERO benchmark evaluates lifelong robotic manipulation in four challenging task suites, LIBERO-Spatial, LIBERO-Object, LIBERO-Goal and LIBERO-10, introducing variations in spatial layouts, object selection, task goals, and long-horizon planning, which test a model’s ability to generalize across diverse manipulation scenarios. As shown in Table 7, our training-free approach was evaluated on these four tasks. Random Dropping, which randomly prunes 112 visual tokens, achieved a success rate of only 18.5% with limited acceleration ( $1.14\times$ ). FastV provided a modest  $1.16\times$  speedup but incurred performance loss. In contrast, EfficientVLA without pruning already achieved a  $1.46\times$  speedup with only a 1.8% performance drop. With pruning, EfficientVLA delivered a  $1.71\times$  speedup, increasing inference frequency from 11.65 Hz to 19.96 Hz, while outperforming FastV in success rate. These results demonstrate the method’s robustness and scalability across diverse VLA models and datasets.

Table 7: Quantitative assessment of performance and efficiency metrics for the  $\pi_0$  baseline model and comparative methods on the LIBERO benchmark, evaluating task-specific success rates (Spatial, Object, Goal, 10), average accuracy, latency, speedup, and operational task frequency.

| Method                        | Spatial | Object | Goal | 10   | Avg. | Latency/s | Speedup | Freq. (Hz) |
|-------------------------------|---------|--------|------|------|------|-----------|---------|------------|
| $\pi_0$                       | 98.2    | 98.2   | 94   | 83.2 | 93.4 | 0.0858    | -       | 11.65      |
| Random Dropping               | 36.8    | 21.4   | 15.8 | 0    | 18.5 | 0.0753    | 1.14×   | 13.28      |
| FastV                         | 95.2    | 96.6   | 90.2 | 79.2 | 90.3 | 0.0739    | 1.16×   | 13.53      |
| <b>EfficientVLA w/o prune</b> | 95.8    | 97.8   | 91.6 | 81.2 | 91.6 | 0.0588    | 1.46×   | 17.01      |
| <b>EfficientVLA</b>           | 95.2    | 97.2   | 90.6 | 78.6 | 90.4 | 0.0501    | 1.71×   | 19.96      |

## C Related Work on Inference Acceleration Techniques

A variety of training-free methods have emerged to expedite inference in large-scale models by leveraging layer pruning, token pruning, and temporal caching mechanisms, thereby reducing computational overhead without necessitating model retraining. In the context of layer pruning for large language models (LLMs), Yuan et al. [45] mitigate redundancy by collapsing multiple redundant layers into a single effective layer, preserving essential functionalities; SlimGPT [46] applies batched greedy pruning coupled with incremental pruning ratios to systematically minimize error accumulation during compression; and TrimLLM [47] enables progressive and targeted layer dropping by exploiting layer-wise specialization, allowing for adaptive model thinning tailored to specific tasks. For visual token pruning in large vision-language models (LVLMs), FastV [42] executes dynamic token elimination guided by real-time attention scores to streamline processing; SparseVLM [48] introduces a text-guided paradigm for rank-based pruning, further enhanced by token recycling to retain informative elements; PruMerge [49] adaptively merges redundant tokens by capitalizing on inherent attention sparsity patterns; and MustDrop [50] refines token selection and retention strategies across distinct phases including encoding, prefilling, and decoding, ensuring minimal performance degradation. Regarding temporal caching in diffusion transformers, FORA [51] supports seamless direct reuse of cached intermediate features in a training-free manner, accelerating iterative generation processes; while ToCa [52] employs token-wise caching with dynamically adaptive retention ratios, optimizing memory usage and inference speed across varying sequence lengths.

## D Impact Statement

This paper introduces EfficientVLA, a crucial contribution to Vision-Language-Action (VLA) models by directly addressing their significant computational and memory demands through a novel training-free framework. EfficientVLA synergistically prunes redundant language layers, optimizes visual token selection for task-relevance and diversity, and caches intermediate features in the diffusion-based action head, thereby significantly boosting VLA model efficiency and speed. This enables the practical deployment of powerful VLA models on resource-constrained robotic platforms for real-time interaction, accelerating progress in robotic manipulation and reasoning tasks, and contributing to more development by reducing computational load. We intend this work for ethical academic research and authorized commercial applications and strictly prohibit its use for harmful, unethical, or unlawful robotic actions, underscoring our responsibility to ensure societal benefit aligned with fundamental ethical principles.

## E Limitations

While EfficientVLA significantly advances VLA model acceleration, certain limitations warrant discussion. Our training-free approach, may not achieve the maximal compression or speedup attainable by training-aware methods. The fixed cache interval  $N$  in the action head introduces a trade-off between acceleration and action fidelity and future work could explore adaptive caching. Furthermore, due to the limited availability of open-source diffusion-based VLA models, our current demonstrations are primarily on CogACT. In the future, we aim to validate scalability and effective-

ness of EfficientVLA across a wider range of models and tasks as more such architectures become available.