

---

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

---

Anonymous Author(s)

Affiliation

Address

email

## Abstract

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

## 20 1 Introduction

21 Building upon advances in multimodal understanding from models integrating vision and language [1,  
22 2, 3, 4, 5], Vision-Language-Action (VLA) models enable transformative embodied intelligence.  
23 These systems, such as OpenVLA [6], CogACT [7],  $\pi_0$  [8] and RT-2 [9], directly translate multimodal  
24 inputs into executable actions, successfully tackling complex robotic manipulation and reasoning  
25 tasks using large-scale datasets [10, 11]. Many cutting-edge VLAs couple a Vision-Language  
26 Model (VLM) for scene and instruction parsing with a diffusion model to handle multi-modal action  
27 distribution [7, 12, 13, 14]. However, the significant computational and memory overheads of these  
28 Diffusion-based VLA architectures during the inference time pose critical barriers to their practical  
29 deployment, particularly for real-time interaction on resource-constrained robotic platforms.

30 Diffusion-based VLA architectures typically comprise a vision encoder to extract features, a large  
31 language model (LLM) [15, 16, 17, 18] core for multimodal reasoning, and a diffusion-based action  
32 decoder to predicts the final actions through multiple denoising steps. While this modular design  
33 underpins their powerful capabilities, it inherently results in substantial computational and memory  
34 overhead. Our findings (Table 1) indicate that the language module and the iterative diffusion head are  
35 primary contributors to overall latency and computational load. Furthermore, as illustrated in Figure 1

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%)

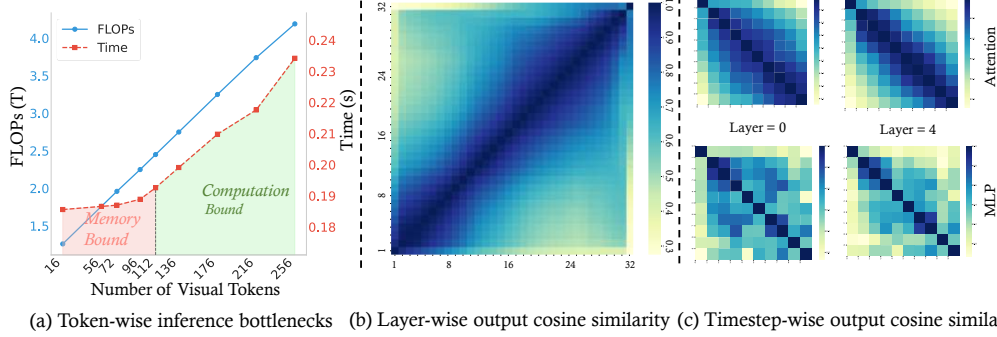


Figure 1: VLA inference bottleneck and redundancy analysis: **(a)** Visual token pruning impact on FLOPs and inference time, revealing computation-bound and memory-bound regimes. **(b)** High inter-layer cosine similarity of LLM hidden states, indicating depth-wise redundancy. **(c)** Temporal cosine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

36 (a), while visual token pruning initially reduces inference time in computation-bound scenarios, its  
 37 efficacy quickly diminishes as the system becomes memory-bound by the LLM.

38 Prior VLA acceleration efforts have largely focused on isolated tweaks, delivering minimal overall  
 39 gains. These fragmented approaches often fail because they ignore the integrated nature of VLA,  
 40 where optimizing one module in isolation merely shifts bottlenecks. Gains are limited by unaddressed  
 41 inefficiencies elsewhere, such as the memory demands of LLM or the computational intensity of  
 42 action head. For example, methods like TinyVLA [14] and DeeR-VLA [19] focus on specialized  
 43 model architectures rather than broadly applicable inference acceleration frameworks for pre-trained  
 44 VLAs. Other approaches, such as Mole-VLA [20], tackle LLM layer redundancy but require costly  
 45 retraining and overlook other pipeline stages. Similarly, VLA-Cache [21] caches static visual  
 46 tokens but provides limited speedup, constrained by the significant memory footprint of LLM and  
 47 computational demands of the action head. Consequently, these existing approaches fall short of  
 48 providing a truly holistic solution to navigate the complex landscape of VLA inefficiencies.

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

60 Motivated by this, we introduce EfficientVLA, a structured, training-free acceleration framework for  
 61 Diffusion-based VLAs that systematically targets these issues. Using a similarity-derived importance  
 62 metric to target the primary memory bottleneck of the language module and its observed depth-wise  
 63 redundancy (Figure 1 (b)), EfficientVLA employs a similarity-derived importance metric to prune  
 64 functionally inconsequential layers, thus reducing the depth of the model and the demands for memory  
 65 without retraining. To manage the initial computational load from visual inputs before the memory  
 66 of LLM limit is reached (Figure 1 (a)), our visual token pruning strategy tackles both task-relevant  
 67 and inherent image redundancies by first selecting critical task-aligned tokens, then augmenting this  
 68 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 [22], 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 Models.** Vision-Language-Action (VLA) models [6, 19, 23, 24] extend Vision-Language Models (VLMs) [3, 25, 26, 27, 28] 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 [23, 29] such as robotic manipulation and object retrieval. VLA models typically use pretrained VLMs [26] 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 [13], DiVLA [12],  $\pi_0$  [8], and TinyVLA [14]. 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 Vision-Language Models (VLMs) 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 [30], EfficientVLM [31], and DeeR-VLA [19], 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 [20] 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 [21], 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.

## 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

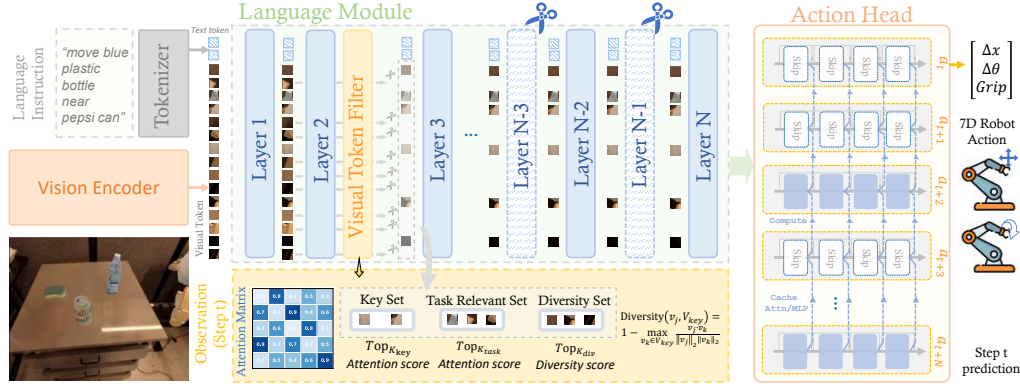


Figure 2: Overview of the **EfficientVLA** framework, our training-free, structured approach to accelerate Diffusion-based VLAs. It employs: (1) pruning of redundant language module layers; (2) VLA task-aware visual token selection balancing task relevance and informational diversity; and (3) temporal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

executable action sequences. The initial stage of our basic VLA model employs a Vision Module, comprising powerful pre-trained encoders DINOv2 [32] and SigLIP [33], 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.

#### 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  $\mathbf{h}_{T_{start}}^{\text{attn}}$  and  $\mathbf{h}_{T_{start}}^{\text{mlp}}$  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:

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

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

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
	EfficientVLA (L=22, T=56)	✓	93.3%	81.3%	68.2%	53.8%	74.2%	28.9%	1.93×	4.86
Variant Aggregation	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

And the outputs for this step are  $\mathbf{h}_t^{\text{attn}} = \mathcal{C}_{\text{attn}}$  and  $\mathbf{h}_t^{\text{mlp}} = \mathcal{C}_{\text{mlp}}$ . 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:

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

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

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 [22], 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* [34], which integrates powerful vision encoders (DINOv2 [32] and SigLIP [33]), a Llama2-7B [15] 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* [35], a notable approach focused on accelerating inference by pruning redundant visual tokens, and *VLA-Cache* [21], which leverages temporal analysis to cache static tokens across timesteps.

**Implementation Details.** For EfficientVLA, in addition to layer pruning, we further compressed the model parameters by adopting the PruneNet [36] configuration for LLM compression. Specifically, we applied a sparsity of 25% to the MLP layers of all Transformer blocks. For visual token pruning,



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

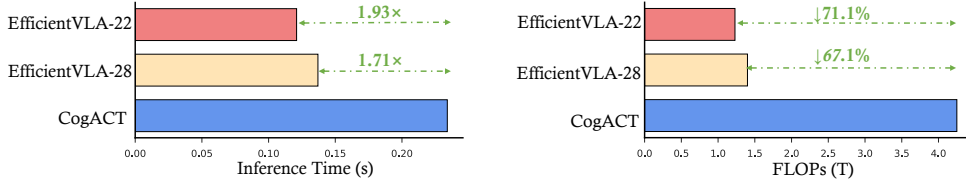


Figure 3: **Efficiency analysis** in simulation, comparing FLOPs and inference time of our EfficientVLA variants against the original model backbone. EfficientVLA-22 and EfficientVLA-28 denote configurations retaining 22 and 28 LLM layers, respectively.

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 [35] ( $T = 56$ ) 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 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.



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	72	96	112	256
	77.8%	71.8%	62.5%	56.2%	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>

**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 [22], 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] A. Awadalla, I. Gao, J. Gardner, J. Hessel, Y. Hanafy, W. Zhu, K. Marathe, Y. Bitton, S. Gadre, S. Sagawa, J. Jitsev, S. Kornblith, P. W. Koh, G. Ilharco, M. Wortsman, and L. Schmidt, “Openflamingo: An open-source framework for training large autoregressive vision-language models,” *arXiv preprint arXiv:2308.01390*, 2023.
- [2] J. Li, D. Li, C. Xiong, and S. Hoi, “Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation,” in *International conference on machine learning*. PMLR, 2022, pp. 12 888–12 900.
- [3] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, “Learning transferable visual models from natural language supervision,” in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.
- [4] R. An, S. Yang, M. Lu, K. Zeng, Y. Luo, Y. Chen, J. Cao, H. Liang, Q. She, S. Zhang *et al.*, “Mc-llava: Multi-concept personalized vision-language model,” *arXiv preprint arXiv:2411.11706*, 2024.
- [5] Y. Luo, R. An, B. Zou, Y. Tang, J. Liu, and S. Zhang, “Llm as dataset analyst: Subpopulation structure discovery with large language model,” in *European Conference on Computer Vision*. Springer, 2024, pp. 235–252.
- [6] M. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn, “Openvla: An open-source vision-language-action model,” *arXiv preprint arXiv:2406.09246*, 2024.
- [7] Q. Li, Y. Liang, Z. Wang, L. Luo, X. Chen, M. Liao, F. Wei, Y. Deng, S. Xu, Y. Zhang, X. Wang, B. Liu, J. Fu, J. Bao, D. Chen, Y. Shi, J. Yang, and B. Guo, “Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation,” 2024.
- [8] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter *et al.*, “ $\pi_0$ : A vision-language-action flow model for general robot control,” *arXiv preprint arXiv:2410.24164*, 2024.
- [9] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich, “Rt-2: Vision-language-action models transfer web knowledge to robotic control,” 2023. [Online]. Available: <https://arxiv.org/abs/2307.15818>
- [10] H.-S. Fang, H. Fang, Z. Tang, J. Liu, C. Wang, J. Wang, H. Zhu, and C. Lu, “Rh20t: A comprehensive robotic dataset for learning diverse skills in one-shot,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 653–660.
- [11] A. O’Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain *et al.*, “Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6892–6903.
- [12] J. Wen, M. Zhu, Y. Zhu, Z. Tang, J. Li, Z. Zhou, C. Li, X. Liu, Y. Peng, C. Shen *et al.*, “Diffusion-vla: Scaling robot foundation models via unified diffusion and autoregression,” *arXiv preprint arXiv:2412.03293*, 2024.
- [13] J. Wen, Y. Zhu, J. Li, Z. Tang, C. Shen, and F. Feng, “Dexvla: Vision-language model with plug-in diffusion expert for general robot control,” *arXiv preprint arXiv:2502.05855*, 2025.
- [14] J. Wen, Y. Zhu, J. Li, M. Zhu, K. Wu, Z. Xu, N. Liu, R. Cheng, C. Shen, Y. Peng *et al.*, “Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation,” *arXiv preprint arXiv:2409.12514*, 2024.
- [15] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, “Llama 2: Open foundation and fine-tuned chat models,” *arXiv preprint arXiv:2307.09288*, 2023.
- [16] L. Floridi and M. Chiriatti, “Gpt-3: Its nature, scope, limits, and consequences,” *Minds and Machines*, vol. 30, pp. 681–694, 2020.

- [17] OpenAI *et al.*, “Gpt-4 technical report,” 2024. [Online]. Available: <https://arxiv.org/abs/2303.08774>
- [18] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” 2019. [Online]. Available: <https://arxiv.org/abs/1907.11692>
- [19] Y. Yue, Y. Wang, B. Kang, Y. Han, S. Wang, S. Song, J. Feng, and G. Huang, “Deer-vla: Dynamic inference of multimodal large language models for efficient robot execution,” in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- [20] R. Zhang, M. Dong, Y. Zhang, L. Heng, X. Chi, G. Dai, L. Du, D. Wang, Y. Du, and S. Zhang, “Mole-vla: Dynamic layer-skipping vision language action model via mixture-of-layers for efficient robot manipulation,” *arXiv preprint arXiv:2503.20384*, 2025.
- [21] S. Xu, Y. Wang, C. Xia, D. Zhu, T. Huang, and C. Xu, “Vla-cache: Towards efficient vision-language-action model via adaptive token caching in robotic manipulation,” *arXiv preprint arXiv:2502.02175*, 2025.
- [22] X. Li, K. Hsu, J. Gu, K. Pertsch, O. Mees, H. R. Walke, C. Fu, I. Lunawat, I. Sieh, S. Kirmani *et al.*, “Evaluating real-world robot manipulation policies in simulation,” *arXiv preprint arXiv:2405.05941*, 2024.
- [23] X. Li, M. Liu, H. Zhang, C. Yu, J. Xu, H. Wu, C. Cheang, Y. Jing, W. Zhang, H. Liu *et al.*, “Vision-language foundation models as effective robot imitators,” *arXiv preprint arXiv:2311.01378*, 2023.
- [24] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid *et al.*, “Rt-2: Vision-language-action models transfer web knowledge to robotic control,” in *Conference on Robot Learning*. PMLR, 2023, pp. 2165–2183.
- [25] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, R. Ring, E. Rutherford, S. Cabi, T. Han, Z. Gong, S. Samangooei, M. Monteiro, J. Menick, S. Borgeaud, A. Brock, A. Nematzadeh, S. Sharifzadeh, M. Binkowski, R. Barreira, O. Vinyals, A. Zisserman, and K. Simonyan, “Flamingo: a visual language model for few-shot learning,” 2022. [Online]. Available: <https://arxiv.org/abs/2204.14198>
- [26] S. Karamcheti, S. Nair, A. Balakrishna, P. Liang, T. Kollar, and D. Sadigh, “Prismatic vlms: Investigating the design space of visually-conditioned language models,” 2024. [Online]. Available: <https://arxiv.org/abs/2402.07865>
- [27] J. Li, D. Li, C. Xiong, and S. Hoi, “Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation,” 2022. [Online]. Available: <https://arxiv.org/abs/2201.12086>
- [28] H. Liu, C. Li, Q. Wu, and Y. J. Lee, “Visual instruction tuning,” 2023. [Online]. Available: <https://arxiv.org/abs/2304.08485>
- [29] A. Awadalla, I. Gao, J. Gardner, J. Hessel, Y. Hanafy, W. Zhu, K. Marathe, Y. Bitton, S. Gadre, S. Sagawa, J. Jitsev, S. Kornblith, P. W. Koh, G. Ilharco, M. Wortsman, and L. Schmidt, “Openflamingo: An open-source framework for training large autoregressive vision-language models,” 2023. [Online]. Available: <https://arxiv.org/abs/2308.01390>
- [30] J. Liu, M. Liu, Z. Wang, P. An, X. Li, K. Zhou, S. Yang, R. Zhang, Y. Guo, and S. Zhang, “Robomamba: Efficient vision-language-action model for robotic reasoning and manipulation,” 2024. [Online]. Available: <https://arxiv.org/abs/2406.04339>
- [31] T. Wang, W. Zhou, Y. Zeng, and X. Zhang, “Efficientvlm: Fast and accurate vision-language models via knowledge distillation and modal-adaptive pruning,” 2022. [Online]. Available: <https://arxiv.org/abs/2210.07795>
- [32] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby *et al.*, “Dinov2: Learning robust visual features without supervision,” *arXiv preprint arXiv:2304.07193*, 2023.
- [33] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer, “Sigmoid loss for language image pre-training,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 11 975–11 986.
- [34] Q. Li, Y. Liang, Z. Wang, L. Luo, X. Chen, M. Liao, F. Wei, Y. Deng, S. Xu, Y. Zhang, X. Wang, B. Liu, J. Fu, J. Bao, D. Chen, Y. Shi, J. Yang, and B. Guo, “Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation,” 2024. [Online]. Available: <https://arxiv.org/abs/2411.19650>

- 434 [35] L. Chen, H. Zhao, T. Liu, S. Bai, J. Lin, C. Zhou, and B. Chang, “An image is worth 1/2 tokens after layer  
435 2: Plug-and-play inference acceleration for large vision-language models,” in *European Conference on*  
436 *Computer Vision*. Springer, 2025, pp. 19–35.
- 437 [36] A. Sengupta, S. Chaudhary, and T. Chakraborty, “You only prune once: Designing calibration-free model  
438 compression with policy learning,” *arXiv preprint arXiv:2501.15296*, 2025.

## NeurIPS Paper Checklist

### 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, hyperparameters, 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:

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

## 758 16. Declaration of LLM usage

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

763 Answer: [NA]

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

765 Guidelines:

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