AIA-VLA: TRAINING-FREE ADAPTIVE INFERENCE ACCELERATION FOR VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action models (VLAs) combine the semantic understanding of pretrained vision-language models with diffusion-based action generation, demonstrating strong capabilities in robotic manipulation. However, their computational cost and inference latency hinder deployment on resource-constrained robotic platforms. Acceleration strategies with fixed computational budgets cannot readily accommodate varying computational demands across observations and inference stages, potentially compromising task performance. We present AIA-VLA, a training-free adaptive inference acceleration framework that uses intermediate inference signals to jointly regulate visual token budgets, language model execution depth, and feature reuse in diffusion-based action models. Specifically, AIA-VLA determines visual token budgets through cumulative task-relevant attention, uses recent inter-layer representation stability to trigger subsequent layer skipping, and uses feature drift between consecutive fully computed diffusion steps to decide whether to reuse attention and MLP branch outputs in the next step. These modules can operate independently or in combination, enabling input-dependent computation allocation without modifying model parameters. Evaluated on the SIMPLER benchmark with CogACT as the base model, AIA-VLA reduces floating-point operations by 50% and achieves a 1.54$\times$ inference speedup, while improving task success rate by 1.16 percentage points.

1 INTRODUCTION

Recent advances in large-scale pretrained vision-language models (VLMs) have enabled powerful visual perception, language understanding, and cross-modal integration. Building on these capabilities, vision-language-action (VLA) models transfer pretrained multimodal knowledge to robotic manipulation. Representative approaches, including RT-2 (Brohan et al., 2023), OpenVLA (Kim et al., 2024), CogACT (Li et al., 2024), π_0 (Black et al., 2026), and $\pi_{0.5}$ (Black et al., 2025), jointly model visual observations, language instructions, and robot actions. However, large vision-language backbones and iterative action generation introduce computational costs and inference latency, hindering deployment on resource-constrained robots that require low-latency closed-loop control.

Diffusion-based VLA architectures such as CogACT typically involve three computational stages: visual processing, multimodal integration, and iterative action generation. Each exhibits distinct forms of redundancy. Visual encoders produce numerous tokens, not all of which are relevant to the current instruction. Language models execute a predetermined number of Transformer layers, even when intermediate representations change little. Diffusion-based action models perform multiple denoising iterations, during which adjacent steps may produce similar features. Previous studies have explored visual token reduction (Zhang et al., 2025b;a; Chen et al., 2025), dynamic layer execution (Elhoushi et al., 2024; Corro et al., 2023), and diffusion feature caching (Ma et al., 2024; Liu et al., 2025a;b; Selvaraju et al., 2024). Nevertheless, similar intermediate representations do not necessarily imply that their associated computation can be omitted without affecting the output. Effective acceleration must therefore balance computational savings against action prediction.

Existing approaches improve VLA efficiency through lightweight architectures and inference acceleration. SmolVLA (Shukor et al., 2025) adopts a compact architecture, while SP-VLA (Li et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2026) combines model scheduling with visual token pruning to exploit temporal and spatial redundancy. Recent training-free approaches further exploit redundancy in pretrained VLA models. VLA-Cache (Xu et al., 2025) adaptively reuses visual representations across observations, whereas EfficientVLA (Yang et al., 2025b) combines task-aware visual token selection, language-model layer pruning, and diffusion feature caching. These advances motivate further investigation into how computation can be allocated according to intermediate signals observed during inference, rather than relying solely on fixed computational budgets that may not capture variations across inputs and inference stages.

We propose AIA-VLA (**A**daptive **I**nference **A**cceleration for **V**ision-**L**anguage-**A**ction models), a training-free framework that adaptively reduces computation across the three stages of diffusion-based VLA inference. The visual module uses cumulative task-relevant attention from the first language-model layer to determine an input-dependent token budget. The language-model module monitors cosine drift between recently executed layers and conditionally skips subsequent layers. The diffusion module measures MLP feature drift between consecutive fully computed denoising steps to determine whether cached attention and MLP outputs can be reused. These mechanisms make stage-specific decisions, operate independently or jointly, and require neither changes to pretrained parameters nor additional training.

We evaluate AIA-VLA on CogACT (Li et al., 2024) using the SIMPLER robotic manipulation benchmark (Li et al., 2025). Under the primary configuration, AIA-VLA reduces FLOPs by approximately 50% and achieves a $1.54\times$ inference speedup while maintaining a comparable task success rate, with an observed difference of 1.16 percentage points relative to unaccelerated CogACT. Experiments across acceleration settings and module combinations further characterize computational savings and their effects on task performance.

Our main contributions are summarized as follows:

- **A training-free, cross-stage adaptive acceleration framework.** We propose AIA-VLA, a unified framework that dynamically allocates computation across visual processing, language-model execution, and diffusion-based action generation based on intermediate inference signals. It adapts computational effort to the current inference state without modifying pretrained model parameters or requiring additional training.
- **Stage-specific adaptation driven by intermediate inference signals.** We develop adaptive visual token pruning, dynamic layer skipping, and conditional diffusion feature reuse, guided by cumulative task-relevant attention, inter-layer representation drift, and cross-step feature drift, respectively.
- **Systematic evaluation on robotic manipulation tasks.** We evaluate AIA-VLA on CogACT and SIMPLER, analyzing inference efficiency, task performance, and trade-offs across acceleration settings and module combinations.

2 PRELIMINARY

Diffusion-based VLA models integrate visual encoding, multimodal processing, and iterative action generation. Representative models, including RT-2, OpenVLA, CogACT, and π_0, connect pretrained vision-language representations with robotic control. Efficiency-oriented approaches address their computational demands through compact architectures and asynchronous inference, as in SmolVLA (Shukor et al., 2025), or action-aware scheduling and visual token pruning, as in SP-VLA (Li et al., 2026). Training-free methods include VLA-Cache (Xu et al., 2025), which reuses visual representations across observations, and EfficientVLA (Yang et al., 2025b), which jointly accelerates visual processing, language-model execution, and diffusion-based action generation. We next introduce the three computational stages underlying AIA-VLA and their associated acceleration techniques.

2.1 VISUAL TOKEN PROCESSING

Given an image, a visual encoder produces $V = \{v_i\}_{i=1}^{N}$. During multimodal processing, text queries attend to visual keys. For attention head h, the weight from text query q to visual token i is

expressed as

$$a_{q,i}^{(h)} = \frac{\exp\left(Q_q^{(h)}(K_i^{(h)})^\top/\sqrt{d_h}\right)}{\sum_j \exp\left(Q_q^{(h)}(K_j^{(h)})^\top/\sqrt{d_h}\right)}. \tag{1}$$

Visual token reduction exploits token importance and redundancy. Token Merging (ToMe) (Bolya et al., 2023) merges similar tokens, while LLaVA-PruMerge (Shang et al., 2025) combines visual attention with feature similarity. FastV (Chen et al., 2025) uses early-layer attention for subsequent token pruning. SparseVLM (Zhang et al., 2025b) introduces text-guided sparsification, whereas FasterVLM (Zhang et al., 2025a) estimates importance from visual-encoder attention. VisionZip (Yang et al., 2025a) selects informative tokens, while SparseVILA (Khaki et al., 2025) and ZipVL (He et al., 2025) investigate sparsity across inference stages. Building on these approaches, AIA-VLA uses first-layer text-to-visual attention and a cumulative relevance threshold to determine an input-dependent retention budget.

2.2 Transformer Layer Execution

A Transformer layer updates its hidden representations through attention and feed-forward computation. Abstracting these operations as a residual mapping F_l, its output is

$$H^{(l)} = H^{(l-1)} + F_l\left(H^{(l-1)}\right). \tag{2}$$

Executing every layer incurs a fixed computational cost despite variations in individual layer contributions. LayerSkip (Elhoushi et al., 2024) enables early exits through specialized training, ShortGPT (Men et al., 2025) identifies redundant layers using block influence, and SkipDecode (Corro et al., 2023) bypasses selected intermediate layers during autoregressive inference. AIA-VLA instead makes training-free, online skipping decisions from representation drift between recently executed layers while preserving designated layers.

2.3 Diffusion-Based Action Generation

Diffusion-based action models generate robot actions through iterative denoising. Let x_t denote the noisy action representation at step t and c the multimodal conditioning information. A denoising update is abstractly expressed as

$$x_{t-1} = \Phi_\theta(x_t, t, c), \tag{3}$$

where Φ_θ denotes the denoising update.

Feature caching reduces redundant computation across denoising steps. DeepCache (Ma et al., 2024) and block caching (Wimbauer et al., 2024) reuse intermediate features. FORA (Selvaraju et al., 2024) caches attention and MLP outputs in diffusion transformers, while TeaCache (Liu et al., 2025a) uses timestep-dependent signals to guide caching. TaylorSeer (Liu et al., 2025b) predicts future features, and FasterCache (Lv et al., 2025) exploits redundancy in video diffusion, including conditional and unconditional branches. AdaSpa (Xia et al., 2025) investigates adaptive sparse attention for diffusion transformers. Motivated by cross-step feature similarity, AIA-VLA measures MLP feature drift between consecutive fully computed steps to determine whether cached attention and MLP outputs can be reused.

3 Method

3.1 Overview of AIA-VLA

AIA-VLA is a training-free adaptive inference acceleration framework for diffusion-based VLA models. As shown in Figure 1, it uses intermediate signals to reduce computation across visual processing, language-model execution, and diffusion-based action generation without modifying pretrained parameters or requiring additional training.

The visual module uses first-layer text-to-visual attention to determine an input-dependent token budget. The language-model module monitors inter-layer representation drift to conditionally skip

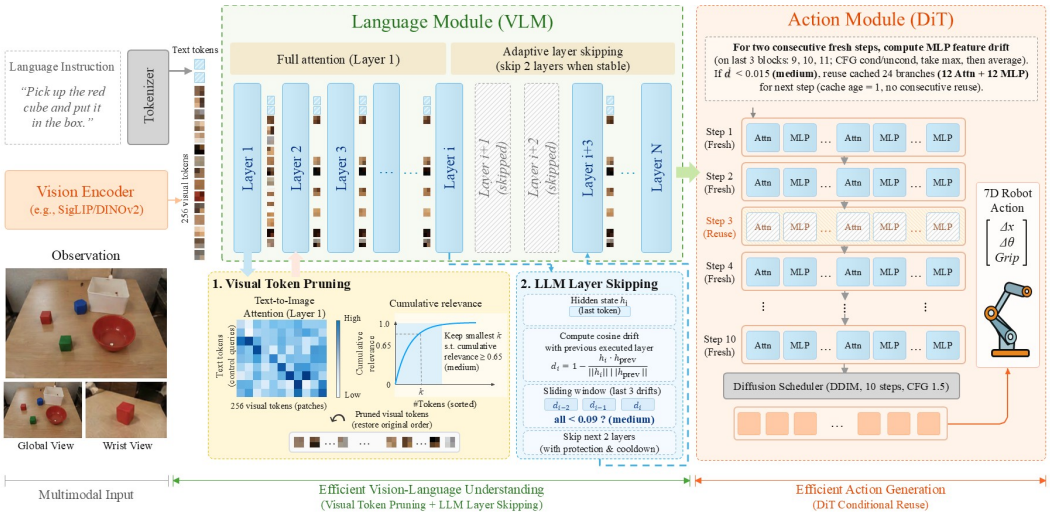

Figure 1: Overview of AIA-VLA. Three training-free modules adapt computation through cumulative task-relevant visual token pruning, representation-drift-based language-model layer skipping, and feature-drift-based diffusion branch reuse.

subsequent layers. The diffusion module measures MLP feature drift between consecutive fully computed denoising steps to decide whether cached attention and MLP outputs can be reused. The modules make independent decisions and can be enabled individually or in combination. We implement and evaluate AIA-VLA on CogACT.

3.2 Adaptive Visual Token Pruning

We estimate visual token relevance from first-layer text-to-visual attention and adaptively determine the retention budget within a single forward pass, without an additional scoring network or a second full inference pass.

Task relevance estimation. Let the visual encoder produce N tokens and the first language-model layer contain H attention heads. Denote by $\mathcal{Q}$ the valid text-context query indices and by $a_{q,i}^{(h)}$ the attention weight from text query q to visual token i in head h. Averaging across heads and summing over valid queries gives the relevance score

$$s_i = \sum_{q \in \mathcal{Q}} \frac{1}{H} \sum_{h=1}^{H} a_{q,i}^{(h)}. \tag{4}$$

The valid text context comprises the original prompt excluding the BOS token and final two suffix tokens, rather than an instruction-only span.

Adaptive token selection. We normalize relevance across visual tokens:

$$\hat{s}_i = \frac{s_i}{\sum_{j=1}^{N} s_j}. \tag{5}$$

Let $\hat{s}_{(1)} \geq \hat{s}_{(2)} \geq \cdots \geq \hat{s}_{(N)}$ be the normalized scores in descending order. Given a cumulative relevance threshold τ_v, the minimum retention budget is

$$k^* = \min \left\{ k : \sum_{i=1}^{k} \hat{s}_{(i)} \geq \tau_v \right\}. \tag{6}$$

Unlike a fixed retention ratio, τ_v specifies the fraction of total relevance covered, allowing the token budget to vary across inputs.

The cumulative criterion makes the retention budget sensitive to the distribution of attention rather than to a fixed pruning ratio. When relevance is concentrated on a small subset of visual tokens, fewer tokens are needed to satisfy the threshold. When relevance is distributed more broadly, the criterion retains a larger portion of the visual sequence. This provides an input-dependent budget without requiring a separately trained importance predictor or an additional scoring network.

Sequence compression. After fully executing the first language-model layer, we retain the top-k^* visual tokens, restore their original patch order, and preserve their original position IDs. At the entrance to the second layer, we compress the visual sequence while retaining all text tokens. Subsequent layers process the shorter multimodal sequence. If relevance scores are invalid or sum to zero, all visual tokens are retained. Importantly, visual token selection is performed after the first language-model layer rather than before visual encoding. This allows the relevance scores to reflect the interaction between the visual input and the current text context. Although the first layer processes the complete multimodal sequence, subsequent layers operate on the compressed sequence, reducing their attention and feed-forward computation. The selection is performed once per forward pass, without an additional inference pass.

3.3 Adaptive Language Model Layer Skipping

We dynamically skip language-model layers based on recent representation stability, subject to protected layers, cooldown, and history-reset constraints.

Inter-layer representation drift. For each executed layer i, let h_i be the raw hidden state of its final output token and h_p the corresponding state from the previously executed layer. Their cosine drift is

$$d_i = 1 - \frac{h_i^\top h_p}{\|h_i\|_2 \|h_p\|_2}. \tag{7}$$

Smaller drift indicates greater similarity between consecutive executed-layer representations.

Adaptive skipping decision. We maintain a sliding window $\mathcal{W}_i$ of the most recent w valid drift values. Once full, the window triggers a skip only if

$$\max_{d \in \mathcal{W}_i} d < \tau_l. \tag{8}$$

We use three drift values and skip the next two Transformer layers upon a successful trigger. During normal signal collection, the condition is checked after each executed layer, allowing multiple skip events within one forward pass.

Protection and cooldown. Layers 0, 1, and 31 are protected; any skip involving them is rejected. After a successful skip, we reset the drift history and execute the next three layers without further skipping. The first establishes a new reference, and the following two contribute drift measurements. Collection continues after cooldown, permitting another skip once the three-value window satisfies the threshold. The final layer and normalization are always executed. These constraints prevent the skipping decision from relying on stale representations. In particular, resetting the history after each skip ensures that subsequent decisions are based on newly executed layers rather than on hidden states collected before the skipped interval. Protecting the initial and final layers further preserves the complete initial processing and final representation transformation.

3.4 Adaptive Diffusion Feature Reuse

We conditionally reuse cached attention and MLP outputs according to MLP feature drift between consecutive fully computed denoising steps, preserving the original sampling schedule.

Cross-step feature drift. Let $t-1$ and t denote consecutive fully computed denoising steps. For DiT block b, let $m_{t,b}^{(c)}$ be its raw MLP branch output at step t, where $c \in \{\mathrm{cond}, \mathrm{uncond}\}$ denotes the classifier-free guidance (CFG) branch. We flatten each complete feature tensor and compute its

cross-step cosine drift:

$$d_{t,b}^{(c)} = 1 - \frac{\mathrm{vec}(m_{t,b}^{(c)})^\top \mathrm{vec}(m_{t-1,b}^{(c)})}{\|\mathrm{vec}(m_{t,b}^{(c)})\|_2 \|\mathrm{vec}(m_{t-1,b}^{(c)})\|_2}. \tag{9}$$

Here, $\mathrm{vec}(\cdot)$ flattens the entire tensor rather than averaging token-wise similarities.

We take the maximum drift across CFG branches for each block:

$$d_{t,b} = \max_{c \in \{\mathrm{cond}, \mathrm{uncond}\}} d_{t,b}^{(c)}. \tag{10}$$

The drift is then averaged over the monitored blocks $\mathcal{B}$:

$$D_t = \frac{1}{|\mathcal{B}|} \sum_{b \in \mathcal{B}} d_{t,b}. \tag{11}$$

For our 12-block DiT-B, $\mathcal{B} = \{9, 10, 11\}$ contains the final three blocks with zero-based indexing. We use the final three blocks as a compact monitoring set to avoid introducing additional decision overhead across the entire DiT. The resulting drift score serves as a shared trigger for feature reuse across all blocks, rather than requiring a separate caching decision for each block.

Conditional branch reuse. Given threshold τ_d, reuse is enabled at the next denoising step when

$$D_t < \tau_d. \tag{12}$$

Although the decision monitors only the final three blocks, a successful trigger reuses cached attention and MLP outputs from all 12 blocks, covering 24 branch outputs. Necessary computations and scheduler updates are still performed; no denoising step is skipped.

Cache validity and reuse constraints. We use a single-step cache with a fixed age of one: outputs may be reused only from the immediately preceding fully computed step. Consecutive reuse is prohibited. After each reuse step, we clear the history of fully computed steps, so subsequent reuse decisions require new drift measurements rather than following a fixed sampling schedule. This reduces attention and MLP computation while preserving the complete denoising process.

The reuse decision is deliberately constrained by the availability of freshly computed features. After a reuse event, the drift history is reset, and a new decision requires two consecutive fully computed steps. This prevents a sequence of reuse decisions from being triggered by the same previously observed feature similarity.

The mechanism changes the internal computation performed at selected denoising steps, but it does not remove any step from the original sampling schedule. The scheduler continues to update the action representation at every step, preserving the number and order of denoising iterations.

4 Experiments

4.1 Experimental Setup

Base model and benchmark. We evaluate AIA-VLA using CogACT as the base model on the SIMPLER benchmark. Our evaluation covers four robotic manipulation task categories: drawer, move near, pick coke can, and put in drawer, with 864 evaluation instances in total. All methods are evaluated using identical tasks, random seeds, and initial states. Model parameters, input conditions, and evaluation protocols remain unchanged across methods. We use 10 DDIM denoising steps and a classifier-free guidance (CFG) scale of 1.5 throughout the experiments.

All methods are evaluated on the same 864 instances, with paired random seeds and initial environmental states. This protocol keeps the evaluation conditions consistent across the unaccelerated model and the acceleration baselines, so that their observed differences are not confounded by different task instances.

Importantly, AIA-VLA preserves the original ten-step DDIM sampling schedule and CFG setting. Its computational savings arise from reducing redundant operations within the inference pipeline, rather than shortening the action-generation process or changing the sampling configuration.

Table 1: Overall comparison on SIMPLER. Relative FLOPs and speedup are measured against the original CogACT. The static baseline uses approximately matched computational cost.

Method	Success (%)	Rel. FLOPs	Latency (ms)	Speedup
CogACT	78.01	1.000	982.7	1.00×
Static	78.13	≈0.506	566.6	1.73×
VLA-Cache	78.82	0.792	818.5	1.21×
AIA-VLA (Ours)	**79.17**	0.506	635.4	1.54×

Baselines. We compare AIA-VLA against the original CogACT, VLA-Cache, and a static acceleration baseline with approximately matched computational costs. The static baseline adopts a fixed visual token budget corresponding to the average number of tokens retained by AIA-VLA under the default Medium configuration, combined with fixed strategies for language-model layer skipping and DiT feature reuse. Specifically, layer skipping and feature reuse follow predetermined schedules within the same execution constraints, rather than being dynamically triggered by intermediate inference signals.

The static configuration is selected to approximately match the average FLOPs of AIA-VLA, allowing us to disentangle the computational benefits of acceleration from those of adaptive computation allocation. By maintaining a comparable computational budget, this baseline provides a controlled reference for examining whether input-dependent decisions offer advantages over fixed acceleration schedules. As discussed in Section 4, AIA-VLA achieves a higher observed task success rate than the static baseline at approximately matched FLOPs, although its adaptive decisions introduce additional runtime overhead.

Acceleration configurations and metrics. We evaluate three acceleration configurations: Conservative $(\tau_v, \tau_l, \tau_d) = (0.80, 0.04, 0.006)$, Medium $(0.65, 0.09, 0.015)$, and Aggressive $(0.40, 0.14, 0.030)$. Unless otherwise specified, we use Medium as the default configuration.

We report task success rate, average FLOPs, inference latency, and speedup relative to the original CogACT. We also record the number of retained visual tokens, skipped language-model layers, and reused diffusion steps to characterize adaptive computation.

4.2 Main Results

Overall performance. Table 1 compares AIA-VLA with the original CogACT and two acceleration baselines. Under the default Medium configuration, AIA-VLA reduces average FLOPs by approximately 49.4% and decreases inference latency from 982.7 ms to 635.4 ms, achieving a $1.54\times$ speedup. Meanwhile, the observed task success rate changes from 78.01% to 79.17%, an increase of 1.16 percentage points.These results indicate that AIA-VLA substantially reduces inference computation while maintaining comparable task success.

Comparison with static acceleration. Comparisons with static acceleration schemes validate the effectiveness of adaptive computation. The average computational overhead of the static baseline was aligned with that of AIA-VLA to serve as a reference for equivalent FLOPs. Results show that AIA-VLA achieves a higher success rate than the static baseline (78.13%), demonstrating that dynamic resource scheduling matches input difficulty more precisely than uniform reduction, thereby overcoming static performance bottlenecks. Although AIA-VLA exhibits slightly higher latency—attributable to the overhead of adaptive decision-making—this increase reflects a shift in computational resource usage from "uniform consumption" to "on-demand allocation," representing a reasonable trade-off where a manageable time cost is exchanged for enhanced task reliability.

4.3 Ablation Studies

To investigate the individual and combined effects of our three modules, we evaluate all eight configurations of visual token pruning (V), language-model layer skipping (L), and diffusion feature reuse (D). M000 denotes the original CogACT, while M111 denotes the complete AIA-VLA framework.

Table 2: Ablation of the three acceleration modules under the default Medium configuration. V, L, and D denote visual token pruning, language-model layer skipping, and diffusion feature reuse, respectively.

Configuration	Success (%)	Rel. FLOPs	Latency (ms)
M000 (Baseline)	78.01	1.000	982.7
M100 (V)	80.09	0.595	760.5
M010 (L)	76.97	0.834	742.8
M001 (D)	79.05	0.996	753.5
M110 (V+L)	78.56	0.511	657.8
M101 (V+D)	78.59	0.591	684.7
M011 (L+D)	78.36	0.831	691.7
M111 (V+L+D)	79.17	0.506	635.4

Individual module contributions. As shown in Table 2,employing visual token pruning alone reduces FLOPs to 0.595 of the baseline value while achieving an observation success rate of 80.09%. This indicates that, under our evaluation setup, retaining highly relevant visual tokens not only reduces the computational load of the subsequent language model but also yields a certain improvement in task success rate. With the language model layer-skipping strategy, FLOPs drop to 0.834, latency decreases to 742.8 ms, and the success rate stands at 76.97%. Diffusion feature reuse results in a smaller reduction in FLOPs but lowers latency to 753.5 ms. These results demonstrate the distinct impacts of these three modules on computational cost and execution time.

Combined acceleration. The three two-module configurations achieve success rates between 78.36% and 78.59%. Combining all three modules reduces relative FLOPs to 0.506 and latency to 635.4 ms, while maintaining a success rate of 79.17%.

The computational savings of individual modules are not strictly additive. Visual pruning shortens the sequence processed by subsequent language-model layers, while layer skipping removes computation from some of those layers. Diffusion feature reuse operates separately during action generation. The full configuration demonstrates that these mechanisms can be integrated within a single inference pipeline.

4.4 Efficiency–Performance Trade-off and Adaptive Behavior

Efficiency–performance trade-off. To investigate the trade-off between inference efficiency and task performance, we evaluate AIA-VLA under three acceleration configurations: Conservative, Medium, and Aggressive. As shown in Figure 2, these configurations achieve speedups of $1.31\times$, $1.54\times$, and $1.90\times$, with corresponding task success rates of 80.09%, 79.17%, and 73.15%.

Compared with the original CogACT, which achieves a success rate of 78.01%, the Conservative and Medium configurations maintain comparable task performance while reducing inference latency.The Aggressive configuration achieves a higher speedup but reduces the success rate to 73.15%. These results illustrate the trade-off between computation reduction and task performance. We therefore adopt Medium as our default configuration, providing a balance between inference efficiency and task performance.

Adaptive computation behavior. To characterize the actual computation allocation of AIA-VLA, we analyze the decisions made by its three modules under the default Medium configuration. The visual module retains an average of 126.06 tokens, with observed counts ranging from 109 to 138. The language-model module skips an average of 6.02 layers, ranging from 4 to 8, while the diffusion module reuses features for an average of 2.685 denoising steps, ranging from 1 to 3.

These variations reflect the stage-specific nature of the proposed adaptive decisions. The visual module adjusts its retention budget according to the concentration of cumulative text-to-visual attention, whereas the language-model module responds to representation stability across recently executed layers. The diffusion module independently determines feature reuse from cross-step MLP drift.

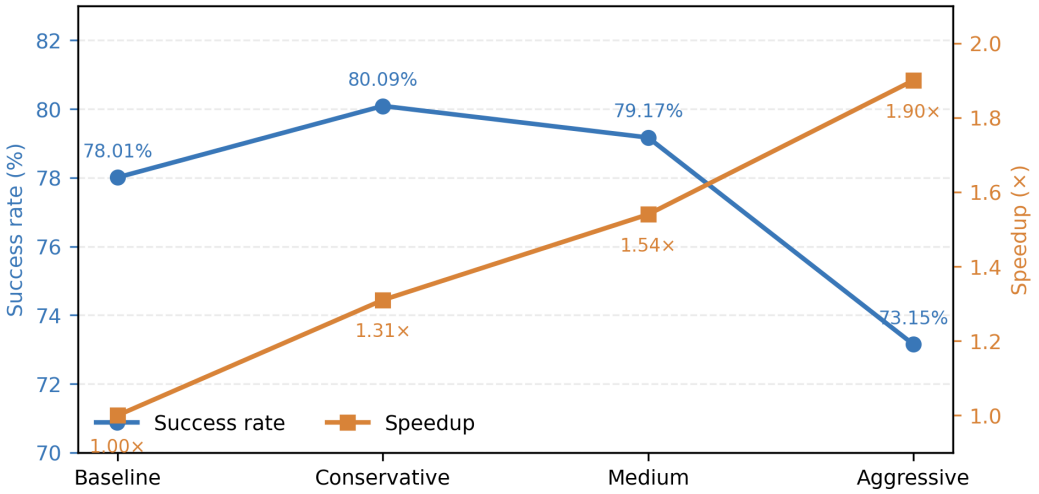

Figure 2: Efficiency–performance trade-off under different acceleration configurations. Task success rate is shown on the left axis, while inference speedup relative to the original CogACT is shown on the right axis.

Consequently, the three modules do not rely on a shared acceleration ratio or a predetermined global execution schedule. Each makes its own decision using signals available at the corresponding inference stage, allowing the overall computation budget to vary with the current input and intermediate model behavior.

5 Conclusion

In this paper, we propose AIA-VLA, a training-free adaptive inference acceleration framework for diffusion-based vision-language-action (VLA) models. Unlike acceleration strategies that employ fixed computation budgets, AIA-VLA accounts for variations in computational demands across inputs and inference stages by dynamically allocating computation based on intermediate signals from the model itself. Specifically, we introduce three coordinated adaptive modules that leverage cumulative task-relevant attention, inter-layer representation drift, and cross-step feature drift to dynamically adjust visual token retention, language-model layer execution, and feature reuse in diffusion-based action generation. The framework requires neither modifications to pretrained model parameters nor additional training.

Experiments on the SIMPLER benchmark demonstrate that AIA-VLA, implemented based on CogACT, reduces floating-point operations by approximately 49.4% and accelerates inference speed by 1.54 times under default configurations, while simultaneously improving the task success rate by 1.16 percentage points. When compared to static acceleration baselines under equivalent computational budgets, AIA-VLA achieves a higher task success rate, highlighting the potential of adaptive computational allocation in maintaining task performance. Ablation studies indicate that the three acceleration mechanisms work synergistically, while experiments involving various acceleration configurations further reveal the trade-off between inference efficiency and task performance.

Our approach still has several limitations. The additional runtime overhead introduced by adaptive decisions results in higher latency than the static baseline at comparable FLOPs. Moreover, our current evaluation primarily focuses on the CogACT architecture and simulated environments, leaving cross-model applicability and real-world robotic deployment for further investigation. Future work will explore lower-overhead adaptive decision mechanisms and evaluate their effectiveness across a broader range of VLA architectures and real-world robotic platforms. We hope that intermediate-signal-driven adaptive computation allocation will provide a practical approach to efficient VLA inference under constrained computational resources.

AI USE STATEMENT

The core research idea, methodological design, and manuscript structure were developed independently by the authors. We used generative AI tools to assist with literature retrieval and organization, code implementation and debugging, and discussions of experimental design and result analysis. Generative AI tools were also used for English translation, language polishing, and LaTeX formatting based on author-provided content.

All AI-assisted outputs were reviewed by the authors, who take full responsibility for the research methodology, implementation, experimental results, scientific claims, and final manuscript.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide detailed descriptions of the three adaptive modules, including their decision rules and implementation details, in Section 3. Section 4 describes the base model, evaluation benchmark, experimental settings, baselines, and evaluation metrics.

REFERENCES

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/black25a.html.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2026. URL https://arxiv.org/abs/2410.24164.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster, 2023. URL https://arxiv.org/abs/2210.09461.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023. URL https://arxiv.org/abs/2307.15818.

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 Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (eds.), *Computer Vision – ECCV 2024*, pp. 19–35, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-73004-7.

Luciano Del Corro, Allie Del Giorno, Sahaj Agarwal, Bin Yu, Ahmed Awadallah, and Subhabrata Mukherjee. Skipdecode: Autoregressive skip decoding with batching and caching for efficient llm inference, 2023. URL https://arxiv.org/abs/2307.02628.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed Aly, Beidi Chen, and Carole-Jean Wu. LayerSkip: Enabling early exit inference and self-speculative decoding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12622–12642, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.681. URL https://aclanthology.org/2024.acl-long.681/.

Yefei He, Feng Chen, Jing Liu, Wenqi Shao, Hong Zhou, Kaipeng Zhang, and Bohan Zhuang. Zipvl: Accelerating vision-language models through dynamic token sparsity. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 20477–20486, October 2025.

Samir Khaki, Junxian Guo, Jiaming Tang, Shang Yang, Yukang Chen, Konstantinos N. Plataniotis, Yao Lu, Song Han, and Zhijian Liu. Sparsevila: Decoupling visual sparsity for efficient vlm inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 23784–23794, October 2025.

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, 2024. URL https://arxiv.org/abs/2406.09246.

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. URL https://arxiv.org/abs/2411.19650.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Oier Mees, Karl Pertsch, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3705–3728. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/li25c.html.

Ye Li, Yuan Meng, Zewen Sun, Kangye Ji, Chen Tang, Jiajun Fan, Xinzhu Ma, Shu-Tao Xia, Zhi Wang, and Wenwu Zhu. Sp-vla: A joint model scheduling and token pruning approach for vla model acceleration. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 38155–38184, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/4072543747a14bbed76284cf2c04b9e9-Paper-Conference.pdf.

Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7353–7363, June 2025a.

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with taylorseers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15853–15863, October 2025b.

Zhengyao Lv, Chenyang Si, Junhao Song, Zhenyu Yang, Yu Qiao, Ziwei Liu, and Kwan-Yee K. Wong. Fastercache: Training-free video diffusion model acceleration with high quality, 2025. URL https://arxiv.org/abs/2410.19355.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15762–15772, June 2024.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. ShortGPT: Layers in large language models are more redundant than you expect. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20192–20204, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1035. URL https://aclanthology.org/2025.findings-acl.1035/.

Pratheba Selvaraju, Tianyu Ding, Tianyi Chen, Ilya Zharkov, and Luming Liang. Fora: Fast-forward caching in diffusion transformer acceleration, 2024. URL https://arxiv.org/abs/2407.01425.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22857–22867, October 2025.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. Smolvla: A vision-language-action model for affordable and efficient robotics, 2025. URL https://arxiv.org/abs/2506.01844.

Felix Wimbauer, Bichen Wu, Edgar Schoenfeld, Xiaoliang Dai, Ji Hou, Zijian He, Artsiom Sanakoyeu, Peizhao Zhang, Sam Tsai, Jonas Kohler, Christian Rupprecht, Daniel Cremers, Peter Vajda, and Jialiang Wang. Cache me if you can: Accelerating diffusion models through block caching, 2024. URL https://arxiv.org/abs/2312.03209.

Yifei Xia, Suhan Ling, Fangcheng Fu, Yujie Wang, Huixia Li, Xuefeng Xiao, and Bin Cui. Training-free and adaptive sparse attention for efficient long video generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15982–15993, October 2025.

Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. Vla-cache: Efficient vision-language-action manipulation via adaptive token caching. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 164448–164473. Curran Associates, Inc., 2025. doi: 10.52202/085713-5484. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/f062da1973ac9ac61fc6d44dd7fa309f-Paper-Conference.pdf.

Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. Visionzip: Longer is better but not necessary in vision language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19792–19802, June 2025a.

Yantai Yang, Yuhao Wang, Zichen Wen, Luo Zhongwei, Chang Zou, Zhipeng Zhang, Chuan Wen, and Linfeng Zhang. Efficientvla: Training-free acceleration and compression for vision-language-action models. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 40891–40914. Curran Associates, Inc., 2025b. doi: 10.52202/085713-1365. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/3a2ef31a1e45908901adc0ca853a8faf-Paper-Conference.pdf.

Qizhe Zhang, Aosong Cheng, Ming Lu, Renrui Zhang, Zhiyong Zhuo, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. Beyond text-visual attention: Exploiting visual cues for effective token pruning in vlms, 2025a. URL https://arxiv.org/abs/2412.01818.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. Sparsevlm: Visual token sparsification for efficient vision-language model inference, 2025b. URL https://arxiv.org/abs/2410.04417.

Table 3: Action MAE under different fixed visual token budgets. Lower is better.

Budget	High-K	Random-K	Low-K
64	**0.01741**	0.03524	0.11604
128	**0.00616**	0.01974	0.10291
192	**0.00221**	0.00994	0.06862

Table 4: Visual token retention under different cumulative relevance thresholds on the independent validation set.

τ_v	Mean K	Median K	Min–Max
0.40	63.56	64	51–73
0.65	124.37	125	112–136
0.80	171.01	171	159–181
1.00	256.00	256	256–256

A Validation of Intermediate Inference Signals

AIA-VLA uses intermediate signals to guide computation allocation at three inference stages. In this section, we provide additional analyses of these signals, focusing on visual token relevance, language-model layer sensitivity, and diffusion feature drift. These experiments complement the task-level evaluation in the main paper by examining the behavior of individual acceleration mechanisms.

A.1 Validation Protocol

For the intermediate-signal analyses, we construct a validation set of 100 observations from Fractal/RT-1. We select ten manipulation tasks, sample two episodes from each task, and extract observations at five temporal positions per episode: 10%, 30%, 50%, 70%, and 90% of the episode duration.

We evaluate action-output fidelity by comparing accelerated predictions with the corresponding predictions of the unaccelerated model. This protocol isolates the effects of individual computational interventions from the closed-loop task success measurements reported in the main paper.

A.2 Visual Relevance and Action Fidelity

We investigate whether the attention-based relevance scores used by AIA-VLA identify visual tokens that are important for preserving action predictions.

At fixed retention budgets of $K \in \{64, 128, 192\}$, we compare three strategies: selecting the highest-scoring visual tokens (High-K), randomly selecting tokens (Random-K), and selecting the lowest-scoring tokens (Low-K). All strategies retain the same number of tokens within each comparison.

As shown in Table 3, High-K achieves the lowest action MAE at all three retention budgets. Because the number of retained tokens is fixed within each comparison, the differences reflect the choice of tokens rather than differences in sequence length.

The results support the use of attention-based relevance for preserving action predictions. However, action fidelity alone does not establish whether the selected tokens correspond to semantically localized manipulation objects.

A.3 Visual Retention Budget Analysis

We further examine how the cumulative relevance threshold translates into the number of retained visual tokens. Unlike a fixed retention ratio, the cumulative criterion allows the budget to vary with the relevance distribution of each observation.

Table 5: Action-output L2 error under fixed language-model layer-skipping interventions. Lower values indicate closer agreement with the unaccelerated model.

Start	Skipped	Mean	Median	Min	Max
8	1	0.726	0.449	0.003	3.003
8	2	0.935	0.711	0.004	4.286
16	1	0.546	0.364	0.004	2.558
16	2	0.778	0.577	0.006	2.766
24	1	0.386	0.223	0.003	1.750
24	2	0.506	0.372	0.007	2.513

Table 6: Latent changes and action-output errors for selected diffusion feature-reuse interventions. The reported values are descriptive measurements for the indicated source and target steps.

Source $\rightarrow$ Target	Age	Mean D_z	Action MAE	MAE P95	Action L2
$1 \rightarrow 2$	1	0.003768	0.000262	0.000981	0.005511
$4 \rightarrow 5$	1	0.008332	0.000447	0.001307	0.009556
$7 \rightarrow 8$	1	0.040182	0.003977	0.006445	0.086288
$3 \rightarrow 8$	5	0.360725	0.007037	0.013027	0.156534
$0 \rightarrow 5$	5	0.121433	0.001361	0.004588	0.059444

Table 4 shows that increasing the threshold retains more visual tokens, while individual observations require different budgets at the same threshold. This behavior follows directly from the cumulative relevance criterion.

A.4 Language-Model Layer Sensitivity

We investigate how fixed layer omission affects the action predictions of CogACT's language-model backbone. Specifically, we intervene at three starting positions and skip either one or two consecutive layers. We measure the resulting action-output L2 error relative to the unaccelerated model on the validation set.

As shown in Table 5, the effect of layer omission depends on both the starting position and the number of skipped layers. At each evaluated starting position, skipping two layers produces a larger mean action-output error than skipping one. The observed mean errors also decrease for the later starting positions in this experiment.

Importantly, the error distributions exhibit substantial variation. For example, skipping two layers from position 8 produces a mean L2 error of 0.935, while the maximum error reaches 4.286. These results indicate that a fixed skip schedule can affect individual observations differently, even when its average error is moderate.Therefore, an adaptive layer-skipping strategy is necessary.

A.5 Diffusion Feature-Drift Analysis

We examine the effect of reusing diffusion features across denoising steps and investigate the relationship between changes in the denoising latent and intermediate DiT features.

All experiments in this subsection use ten DDIM denoising steps, a classifier-free guidance scale of 1.5, and BF16 inference. We report action-output deviations relative to the corresponding fully computed predictions.

Effect of cache age. We first compare feature reuse between adjacent denoising steps with reuse over a longer interval. Cache age denotes the number of denoising steps between the source computation and the target step.

Table 6 shows that the effect of feature reuse depends on the denoising interval and the specific source and target steps.

Table 7: Recorded associations between latent drift and final-block MLP feature drift at selected reuse targets. ρ denotes the recorded correlation coefficient. The final column reports the difference between the feature-drift and latent-drift correlations.

Target, age	Latent ρ	MLP ρ	Difference
$(2, 1)$	-0.0360	0.5683	0.6043
$(5, 1)$	0.0820	0.4382	0.3562
$(5, 5)$	0.0085	0.5010	0.4926
$(8, 1)$	-0.1006	0.6346	0.7352
$(8, 5)$	-0.0043	0.6535	0.6578

Table 8: Per-task paired outcomes of CogACT and AIA-VLA (Medium). "Only" indicates that the corresponding method succeeds while the other fails.

Task	Instances	Both succeed	CogACT only	AIA-VLA only	Both fail
drawer	216	132	28	17	39
move_near	240	185	15	19	21
pick_coke_can	300	242	21	30	7
put_in_drawer	108	31	20	28	29
Total	864	590	84	94	96

For example, adjacent-step reuse from step 4 to step 5 produces an action MAE of 0.000447, whereas reusing features from step 0 at step 5 produces an MAE of 0.001361. Their corresponding action L2 errors are 0.009556 and 0.059444, respectively.

However, cache age alone does not fully determine the error. Even among adjacent steps, the action MAE ranges from 0.000262 for steps 1 to 2 to 0.003977 for steps 7 to 8. This variation motivates using an inference-dependent signal rather than a fixed reuse schedule based solely on step distance.

Latent drift versus feature drift. We further compare the change in the denoising latent, D_z, with the mean cosine drift of MLP outputs in the final three DiT blocks.

The recorded correlations in Table 7 differ considerably between the whole-step latent signal and the MLP feature signal measured in the final three DiT blocks.

Across the five reported conditions, the MLP feature-drift correlations are positive, ranging from 0.4382 to 0.6535, whereas the latent-drift correlations remain close to zero. These observations support examining intermediate DiT features directly rather than relying exclusively on changes in the denoising latent.

This analysis is consistent with AIA-VLA's use of final-block MLP drift as its feature-reuse signal. Nevertheless, the reported correlations are descriptive; they do not establish that the selected threshold is optimal or that low drift guarantees negligible action error.

B Additional Experimental Results

This section provides additional analyses of the SIMPLER evaluation, including per-task paired outcomes, stage-wise computational costs, and the distribution of adaptive computation decisions.

B.1 Per-Task Paired Evaluation

To complement the aggregate task success rates reported in the main paper, we examine paired outcomes for CogACT and AIA-VLA under the default Medium configuration.

Both methods are evaluated on the same 864 instances, allowing us to identify cases in which both succeed, both fail, or only one method succeeds.

Table 9: Adaptive computation decisions under the default Medium configuration.

Indicator	Min	Median	Mean	Max
Visual tokens retained	109	127	126.06	138
LLM layers skipped	4	6	6.02	8
DiT steps reused	1	3	2.685	3

Table 8 provides a more detailed view of the aggregate evaluation. The two methods succeed on the same 590 instances and fail on the same 96 instances. CogACT alone succeeds on 84 instances, whereas AIA-VLA alone succeeds on 94.

The paired outcomes illustrate how differences in aggregate success rates arise from individual evaluation instances.

B.2 Stage-Wise Computational Costs

We decompose inference computation into visual encoding and projection, language-model execution, and diffusion-based action generation.

This decomposition is particularly relevant to AIA-VLA because visual token selection occurs after visual encoding and the first language-model layer. Therefore, reducing the number of visual tokens does not reduce the computational cost of the visual encoder itself.

Instead, visual token reduction primarily saves computation in subsequent language-model layers. Dynamic layer skipping provides additional savings within the language-model backbone, while diffusion feature reuse reduces selected attention and MLP computations during action generation.

The stage-wise breakdown helps distinguish where computation is reduced from where adaptive decisions are made.

B.3 Distribution of Adaptive Decisions

We further analyze the actual computation budgets selected by the three adaptive modules under the Medium configuration.

Table 9 summarizes the variation in the decisions made by each module. The visual retention budget, the number of skipped language-model layers, and the number of reused diffusion steps all vary across inference calls.

These results complement the aggregate efficiency measurements in the main paper by showing that the three modules allocate computation independently, rather than following a single predetermined schedule.