Coordinated Cross-Modal Token Reuse with a Unified Mask for Accurate and Efficient Inference in VLA Models

Anonymous ACL submission

Abstract

Vision-language-action (VLA) inference exhibits substantial redundancy across consecutive frames: large regions of the visual input remain unchanged, yet the model repeatedly re-encodes the entire scene. We present a training-free approach for efficient and accurate VLA inference based on *coordinated cross-modal token reuse*. Our method introduces a *unified mask* that identifies static and task-irrelevant visual patches using a two-stage criterion combining temporal appearance consistency and attention-based relevance. The unified mask drives reuse consistently in both the vision encoder and the language model: cached visual representations are reused for selected patches, while dynamic or task-critical regions are recomputed. This coordinated reuse preserves cross-modal consistency and enables end-to-end acceleration without modifying model architecture or requiring finetuning. Experiments on robotic manipulation tasks show that the proposed approach improves inference efficiency while maintaining or improving task success rates, validating the effectiveness of unified, cross-modal token reuse in VLA models.

1 Introduction

Vision-language-action (VLA) models map visual observations and natural language instructions to low-level control actions, enabling end-to-end robotic manipulation (Zitkovich et al., 2023; Octo Model Team et al., 2024; Kim et al., 2024; Black et al., 2024). Recent open-source VLA systems demonstrate strong generalization across tasks and environments, but their inference cost remains high due to repeated processing of redundant visual content (Liu et al., 2024; Wen et al., 2024; Yue et al., 2024). In closed-loop control, consecutive frames are highly correlated: most visual regions remain static, while only small areas around the end-effector or target objects change. Nevertheless, standard VLA inference recomputes all visual tokens and decoder states at every step, resulting in unnecessary computation and limiting control frequency.

In this work, we address efficient VLA inference from a cross-modal perspective. We propose a training-free approach that coordinates token reuse across the vision encoder and the language model using a single, unified decision variable. Specifically, we introduce a *unified mask* that selects reusable visual patches based on a two-stage criterion: temporal appearance consistency identifies static regions, while text-to-vision attention highlights task-critical regions (Xu et al., 2025). On the vision-encoder side, cached attention and MLP outputs are reused for selected patches, while dynamic regions are recomputed; a lightweight keyframe refresh mitigates error accumulation (Liu et al., 2025). On the language-model side, the same unified mask governs reuse of cached key–value states during the prefill stage (Xu et al., 2025). By enforcing consistent reuse decisions across modalities, our approach preserves visual–language alignment and improves inference efficiency without modifying model architectures or introducing additional training. Figure 1 provides an overview of the setting, and Figure 2 illustrates how the unified mask coordinates token reuse across the two modules.

Our contributions are threefold: (1) we introduce a coordinated cross-modal token reuse mechanism for VLA inference, in which reuse decisions are synchronized across the vision encoder and the language model via a unified mask; (2) we propose a training-free patch selection strategy that balances temporal stability and task relevance, supporting flexible appearance- and attention-based scoring; and (3) we demonstrate through extensive experiments that coordinated reuse improves inference efficiency while maintaining or improving task success rates across multiple benchmarks.

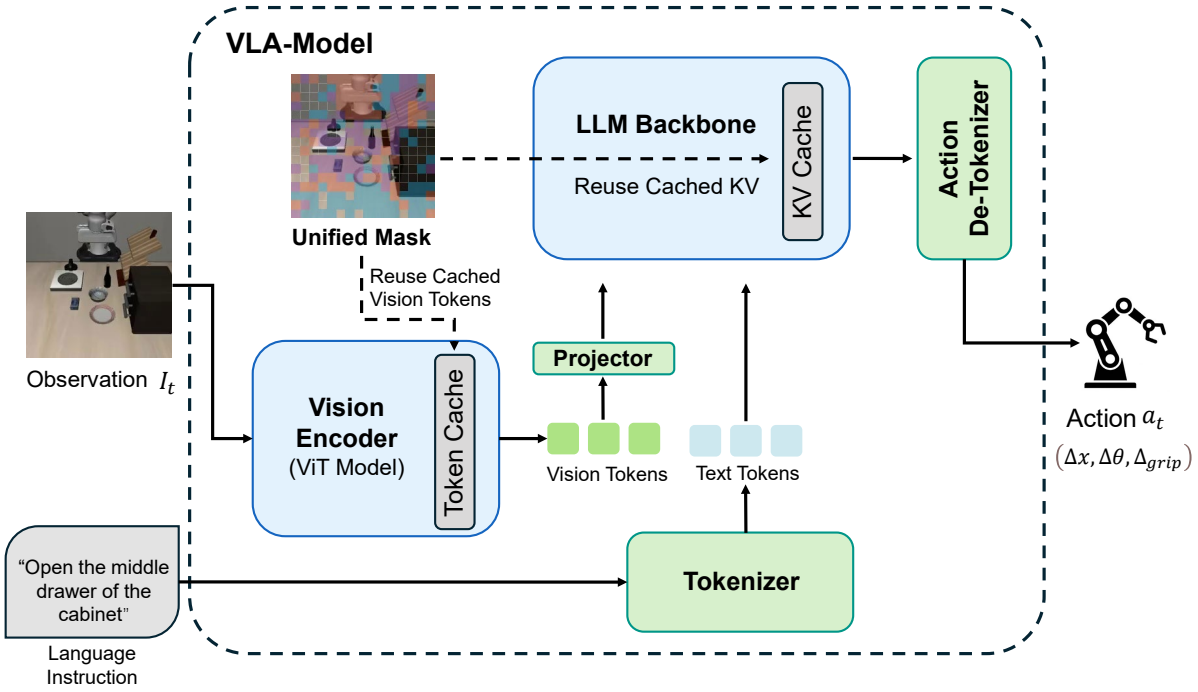

Figure 1: **Overview of a standard VLA inference pipeline and the target of optimization.** At each timestep, the model processes the current observation image through a ViT-based vision encoder to obtain visual tokens, which are projected and concatenated with language instruction tokens and fed into an autoregressive LLM to predict robot actions. In closed-loop control, this pipeline recomputes all visual tokens and decoder states at every step despite high frame-to-frame redundancy. Our method targets this redundancy by introducing synchronized token reuse across the vision encoder and the language model, while keeping the original inference structure and token ordering unchanged.

2 Related Work

VLA Inference Optimization. VLA models inherit large vision–language model backbones and benefit from strong multimodal understanding, yet their high inference cost limits real-time control and increases sensitivity to visual noise. Existing optimization approaches can be broadly categorized into training-aware and training-free methods. Training-aware methods modify model architectures, such as introducing lightweight vision encoders or specialized action heads, and require retraining to maintain performance (Wen et al., 2024; Yue et al., 2024; Zhang et al., 2025; Liu et al., 2024). In contrast, training-free methods exploit redundancy during inference, including pruning tokens or layers within a frame (Yang et al., 2025), reusing caches across frames (Xu et al., 2025), and reusing visual tokens to stabilize inference (Liu et al., 2025). Our work builds on this line by coordinating reuse across the vision encoder and the language model using a *unified mask* as a shared decision rule.

Token Processing Techniques. Efficient token processing in multimodal Transformers has been extensively studied through pruning, merging, and sparse attention mechanisms (Chen et al., 2024; Zhang et al., 2024; Bolya et al., 2022; Cao et al., 2023; Rao et al., 2021; Meng et al., 2022; Liang et al., 2022). These approaches primarily focus on reducing intra-frame computation and may degrade spatial fidelity or overlook temporal redundancy. Temporal reuse strategies, by contrast, are well suited to settings such as robotic control where visual content changes slowly over time. Existing methods typically target either the visual pathway or the language decoder in isolation. Our approach adopts a temporal perspective and emphasizes *cross-modal coordination*, ensuring that reuse decisions are consistent across both components.

3 Methodology

3.1 Problem Setting

We consider a VLA policy that maps an image I_t and a textual prompt to an action. The model consists of a vision encoder (implemented as a Vision Transformer, ViT), a projector, and a large

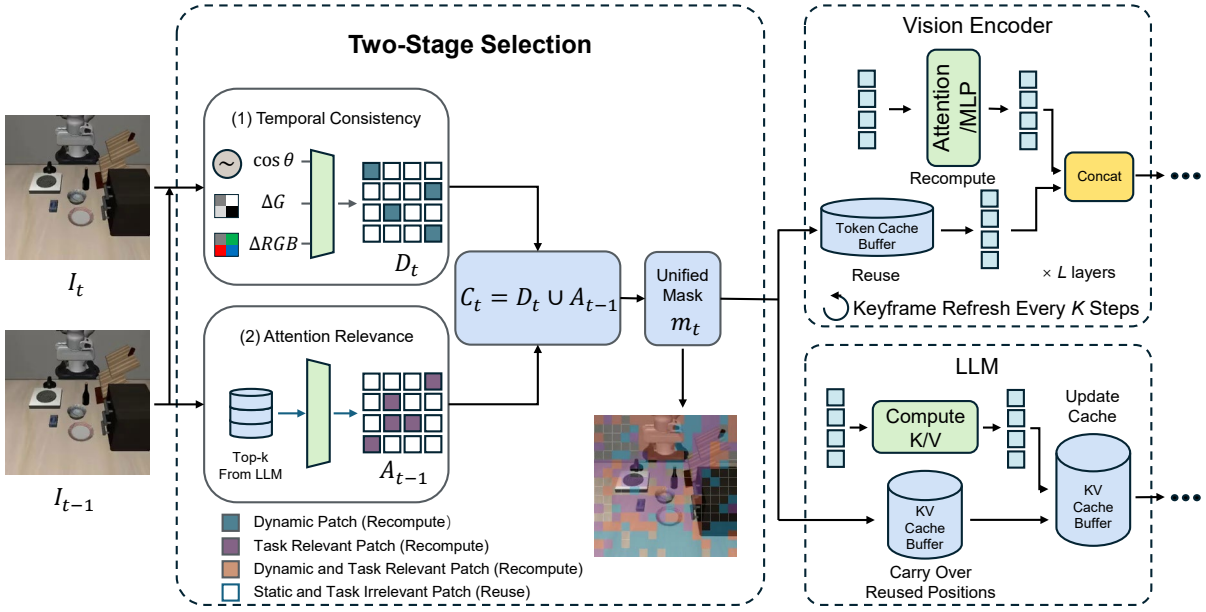

Figure 2: **Coordinated cross-modal token reuse driven by a unified mask.** Given consecutive frames, we construct a unified patch-level mask using a two-stage criterion: temporal appearance consistency identifies static regions, while text-to-vision attention highlights task-relevant patches. The same mask is applied consistently to both modules: in the vision encoder, selected patch representations are reused from cache while others are recomputed; in the language model, the corresponding visual token positions reuse cached key–value states during prefill. By synchronizing reuse decisions across modalities, the approach preserves visual–language alignment while reducing redundant computation.

language model (LLM) (Vaswani, 2017; Touvron et al., 2023; Bai et al., 2023; Team et al., 2024). The vision encoder partitions the image into P patches and outputs patch embeddings, which are projected and concatenated with text tokens before being processed by the language model. In closed-loop control, consecutive frames are highly redundant; recomputing all patches and decoder states at every step wastes significant computation. Our objective is to exploit temporal redundancy by reusing hidden representations across frames in both the vision encoder and the language model, without retraining.

3.2 Cross-Modal Reuse via a Unified Mask

We formulate dual-level reuse as a unified temporal caching strategy that couples the vision encoder and the language model through a single *unified mask*. Let $m_t \in \{0,1\}^P$ denote the unified mask at time t, where $m_t(p) = 1$ indicates that patch p is reusable from the previous frame. The same mask is applied consistently at two levels: (i) within the vision encoder, where selected patches reuse cached sublayer outputs and other patches are recomputed; and (ii) within the language model, where reusable visual token positions skip prefill recomputation and inherit cached key–value states from the previous frame. This design enforces cross-modal consistency, preventing semantic mismatch caused by independent reuse decisions. Token ordering and model architecture remain unchanged, enabling training-free acceleration under standard VLA inference. We refer to this mechanism as *cross-modal token reuse*: on the vision side it is implemented as reuse of cached sublayer outputs, and on the language side as reuse of cached key/value states.

3.3 Two-Stage Patch Selection

We determine the unified mask using a two-stage criterion that combines temporal consistency and task relevance. Let $\pi(\cdot)$ denote patchification into P non-overlapping patches. For each patch p, we first compute a temporal consistency score between I_{t-1} and I_t:

$$s_t(p) = \mathrm{Sim}(\pi(I_{t-1})_p,\ \pi(I_t)_p)\,, \qquad (1)$$

where Sim is either cosine similarity in pixel space or a pixel-difference metric. We mark a patch as static if $s_t(p)$ exceeds a similarity threshold (or falls below a difference threshold), yielding a static

candidate set $\mathcal{S}_t$. The dynamic set is its complement $\mathcal{D}_t = \mathcal{S}_t^c$.

Next, we use text-to-vision attention from the previous step to identify task-relevant patches; task relevance is always computed from $t-1$. Let $a_{t-1}(p)$ denote the aggregated attention score from text tokens to patch p at time $t-1$. We select the top-K patches,

$$\mathcal{A}_{t-1} = \mathrm{TopK}\big(\{a_{t-1}(p)\}_{p=1}^{P}\big), \tag{2}$$

and force them to be recomputed. The recompute set is therefore

$$\mathcal{C}_t = \mathcal{D}_t \cup \mathcal{A}_{t-1}, \tag{3}$$

and the reusable set is

$$\mathcal{R}_t = \mathcal{C}_t^c, \quad m_t(p) = \mathbb{1}[p \in \mathcal{R}_t]. \tag{4}$$

This design ensures that only patches that are temporally stable and not task-relevant are reused; dynamic or task-relevant patches are recomputed. The same m_t is then applied consistently to both the vision encoder and the language model, so the two modules operate on aligned visual content.

3.4 Vision-Encoder-Side Reuse

Let $H_t^{(\ell)} \in \mathbb{R}^{P \times d_v}$ denote the patch embeddings at layer ℓ of the vision encoder. We cache the attention and MLP sublayer outputs for each patch from time $t-1$ and reuse them when $m_t(p) = 1$. We denote the resulting representation after reuse as $\hat{H}_t^{(\ell)}$. Formally, for each patch p:

$$\hat{H}_t^{(\ell)}(p) = \begin{cases} \mathrm{Cache}_{t-1}^{(\ell)}(p), & m_t(p) = 1, \\ \mathrm{VE}^{(\ell)}\Big(H_t^{(\ell-1)}(p)\Big), & m_t(p) = 0, \end{cases} \tag{5}$$

where $\mathrm{VE}^{(\ell)}(\cdot)$ denotes the standard block computation used to obtain the representation for non-reused patches (with reused patches treated as constants). In practice, we treat reused patch representations as constants and only recompute the representations for non-reused patches, then merge them back into the full sequence at each Transformer block, caching the block outputs from the previous frame. We reuse the post-block patch representations (after the residual connection), which can be cached and merged without altering layer normalization or token ordering. This is an approximation because self-attention couples tokens; we do *not* shorten the sequence or modify the attention kernel. Concretely, for reused patches we carry over the post-block representations from $t-1$ and treat them as constants, while non-reused patches are recomputed using the full sequence that includes these constant tokens. Empirically, the approximation is stable under keyframe refresh while reducing the dominant per-layer computation on static patches. This approximation is most effective in settings where inter-patch interactions are dominated by local or low-frequency dynamics, as is common in closed-loop robotic control with slowly changing backgrounds. The full token sequence is preserved and the model architecture is unchanged.

Operationally, the reuse decision is applied at every Transformer block. For each block, we (i) form the full token sequence by combining cached (reused) patch representations with the recomputed patch representations, (ii) run the standard block computation only on the non-reused patches, and (iii) write the resulting post-block representations back into the cache. This yields a consistent per-layer cache of post-block patch embeddings that is used at the next timestep. The computational savings come from skipping the linear projections and MLP for reused patches while still allowing non-reused patches to attend to the full sequence.

Direct cross-frame key–value reuse inside ViT attention would require intrusive kernel changes and careful cache management. By reusing sublayer outputs instead, we keep the implementation minimal while still skipping substantial computation on static patches. To prevent error accumulation from repeated reuse, we apply a keyframe refresh: every K steps, all patches are recomputed and the cache is fully updated.

3.5 LLM-Side Reuse

On the language-model side, we build upon existing token-level cache reuse mechanisms for VLA inference (Xu et al., 2025), and extend them with a unified, cross-modal control signal. Specifically, instead of making reuse decisions solely based on language-model-internal heuristics, we apply the unified mask m_t—constructed from visual temporal consistency and task relevance—to the visual token positions in the decoder. This design aligns language-model-side reuse decisions with those made in the vision encoder, enforcing cross-modal consistency across timesteps.

During the prefill stage, visual tokens corresponding to reusable patches reuse their cached key–value states from the previous timestep, while

the remaining tokens are recomputed:

$$\mathrm{KV}_t(p) = \begin{cases} \mathrm{KV}_{t-1}(p), & p \in \mathcal{R}_t, \\ \mathrm{KV}_t^{\mathrm{new}}(p), & p \in \mathcal{D}_t, \end{cases} \tag{6}$$

where $\mathrm{KV}_t^{\mathrm{new}}(p)$ denotes the key/value states computed at time t for visual token p at the selected decoder layers. Unlike prior approaches that apply reuse independently within the language model, the reuse set $\mathcal{R}_t$ here is determined jointly with the vision encoder, ensuring that both modules operate on an aligned set of visual tokens.

Operationally, we prune the visual token positions in $\mathcal{R}_t$ at selected decoder layers so that these positions skip prefill recomputation and directly inherit cached K/V states from the previous timestep. Dynamic or task-relevant visual tokens remain in the prefill sequence and update the cache as usual. Pruning can be applied at a fixed subset of decoder layers (e.g., every r layers), and the reuse ratio may be modulated by attention entropy; we find our method to be insensitive to the exact choice of r within a reasonable range.

The reuse mask is applied only to visual token positions; text tokens are always recomputed. Token order and sequence layout are preserved, enabling a drop-in implementation compatible with standard autoregressive decoding. Cache updates are performed only for recomputed positions, while reused positions retain their cached states from the previous timestep.

By driving language-model-side reuse with the same unified mask used in the vision encoder, CTR elevates cache reuse from a modality-local optimization (Xu et al., 2025) to a coordinated cross-modal mechanism. This synchronization is critical for preserving visual–language alignment across timesteps and plays a key role in the stability and efficiency gains observed in our experiments.

4 Theoretical Analysis

We provide a detailed complexity analysis to characterize the scaling behavior of coordinated cross-modal token reuse. The analysis is intended to illustrate relative computational savings rather than exact wall-clock runtime.

Let p denote the number of visual patches, L_t the number of text tokens, and $L_v = p$ the number of visual tokens. Let D_v, M_v (D_l, M_l) denote the hidden size and MLP width of the vision encoder (language model), respectively.

Algorithm 1 Coordinated Token Reuse with a Unified Mask

Input: current frame I_t, previous frame I_{t-1}, previous attention A_{t-1}, caches $\mathcal{C}_{t-1}^{\mathrm{VE}}, \mathcal{C}_{t-1}^{\mathrm{LLM}}$
Output: action a_t, updated caches $\mathcal{C}_t^{\mathrm{VE}}, \mathcal{C}_t^{\mathrm{LLM}}$
if IsKeyframe(t) **then**
  $m_t \leftarrow \mathbf{0}$ {disable reuse}
  $V_t \leftarrow \mathrm{VisionEncoder}(I_t)$
else
  $\mathcal{S}_t \leftarrow \mathrm{StaticSelect}(I_{t-1}, I_t)$ {temporal consistency}
  $\mathcal{D}_t \leftarrow \mathcal{S}_t^c$ {dynamic patches}
  $\mathcal{A}_{t-1} \leftarrow \mathrm{TopKAttn}(A_{t-1})$ {task relevance}
  $\mathcal{C}_t \leftarrow \mathcal{D}_t \cup \mathcal{A}_{t-1}$ {recompute set}
  $\mathcal{R}_t \leftarrow \mathcal{C}_t^c$; build unified mask m_t
  $V_t \leftarrow \mathrm{VisionEncoder}(I_t)$ {reuse or recompute per m_t}
  for each patch p **do**
    if $m_t(i) = 1$ **then**
      reuse vision-encoder cache for patch p from $\mathcal{C}_{t-1}^{\mathrm{VE}}$
    end if
  end for
end if
apply LLM prefill pruning using $\mathcal{R}_t$; reuse cached K/V from $\mathcal{C}_{t-1}^{\mathrm{LLM}}$
$a_t \leftarrow \mathrm{LLM}(\mathrm{Project}(V_t))$
update $\mathcal{C}_t^{\mathrm{VE}}$ and $\mathcal{C}_t^{\mathrm{LLM}}$ with recomputed tokens

Selection overhead. Temporal consistency checking compares patch appearance between I_{t-1} and I_t. With patch size p_s and image resolution $H \times H$, the number of patches is $p = (H/p_s)^2$, yielding

$$C_{\mathrm{static}} = \mathcal{O}(pp_s^2) = \mathcal{O}(H^2). \tag{7}$$

Task relevance estimation aggregates text-to-vision attention, with cost

$$C_{\mathrm{attn}} = \mathcal{O}(L_t L_v D_l). \tag{8}$$

Both costs scale linearly with token count and are negligible compared to Transformer attention.

Vision-encoder-side savings. A standard Transformer block in the vision encoder incurs

$$C_{\mathrm{VE}}(p) \approx 4pD_v^2 + 2p^2 D_v + 2pD_v M_v, \tag{9}$$

where the quadratic term arises from self-attention. Our implementation preserves the full sequence length, so we do *not* claim a strict $p^2 \to (1-\rho_v)^2 p^2$

reduction. Instead, reused patches bypass the dominant linear projections and MLP computations, while attention is still evaluated over the full token set. With reuse ratio ρ_v, a conservative per-layer savings estimate is

$$\Delta C_{\text{VE}} \approx 4(\rho_v p) D_v^2 + 2(\rho_v p) D_v M_v, \quad (10)$$

reflecting the skipped projection and MLP costs for static patches. We do not include the quadratic attention score term in this lower-bound estimate, since it is still computed over the full sequence in our implementation. This asymmetry arises because visual-token reuse preserves the full attention graph in the vision encoder, whereas LLM-side pruning directly reduces the effective sequence length during prefill. This estimate aligns with the implementation and provides a lower bound on practical savings.

Language-model-side savings. The language model processes $L = L_t + L_v$ tokens during the prefill stage. With reuse ratio ρ_l, the recomputed sequence length becomes $L' = L_t + (1 - \rho_l)L_v$. Using the standard Transformer cost

$$C_{\text{LLM}}(L) \approx 4LD_l^2 + 2L^2 D_l + 2LD_l M_l, \quad (11)$$

the saved computation per layer is

$$\Delta C_{\text{LLM}} = 4(\rho_l L_v) D_l^2 + 2\big(L^2 - (L')^2\big) D_l + 2(\rho_l L_v) D_l M_l. \quad (12)$$

The quadratic term can be equivalently written as $2(L + L')(L - L')D_l$.

Total savings. Summing across layers and subtracting selection overhead yields

$$\Delta C_{\text{total}} = \sum_{\ell} \big(\Delta C_{\text{VE}}^{(\ell)} + \Delta C_{\text{LLM}}^{(\ell)}\big) - (C_{\text{static}} + C_{\text{attn}}). \quad (13)$$

Because the dominant term in ΔC_{LLM} (the attention score computation) scales quadratically with sequence length, moderate reuse ratios can yield substantial end-to-end reductions, while the selection overhead remains linear.

5 Experiments

We evaluate coordinated cross-modal token reuse (CTR) from three perspectives: (i) task performance across manipulation benchmarks, (ii) end-to-end inference efficiency, and (iii) the individual and combined contributions of vision-side and language-side reuse. Our evaluation spans multiple VLA backbones and environments, and we report both success rates and efficiency metrics.

5.1 Experimental Setup

Models. We evaluate CTR on two representative VLA architectures: OpenVLA and OpenVLA-OFT. OpenVLA is a standard open-source VLA model with a ViT-based vision encoder and an autoregressive language model, while OpenVLA-OFT is a higher-performing variant with additional fine-tuning. CTR is applied in a strictly training-free manner to both models, without modifying model weights.

Benchmarks. We evaluate CTR on two simulation benchmarks that cover diverse manipulation skills and temporal horizons. Our primary benchmark is LIBERO, where we report results on all four suites (Spatial, Object, Goal, and Long-horizon), spanning precise spatial placement, single-object interaction, goal-conditioned multi-step manipulation, and long-horizon sequential tasks. Unless otherwise specified, each suite contains 10 tasks and we evaluate 20 episodes per task (200 episodes per suite). To assess generalization beyond LIBERO without additional tuning, we further evaluate on SimplerEnv, which provides tabletop manipulation scenarios with different object layouts and visual statistics; we report three representative tasks (Move Near Object, Pick Coke Can, and Drawer) with hundreds of evaluation episodes per task (240/300/216, respectively).

Metrics. We report task success rate (percentage of successful episodes) as the primary performance metric. Inference efficiency is measured using model-only CUDA latency (milliseconds per control step) and total TFLOPs, computed as the sum of the vision encoder and language model costs. All efficiency measurements are obtained from the same evaluation runs as success rates and exclude environment/actuation overhead.

Implementation Details. Unless otherwise specified, CTR uses grayscale difference for temporal consistency checking with threshold 0.004, attention-based top-K selection with $K = 160$, and a fixed keyframe interval of $K = 5$, which yields a moderate reuse ratio in practice. SimplerEnv uses the same hyperparameters as LIBERO. All experiments are conducted under the same evaluation protocol and hardware configuration to ensure fair comparison.

Model / Setting	Spatial						Object					
	Succ.	ΔSucc(%)	Lat.(ms)	ΔLat(%)	TFLOPs	ΔTF(%)	Succ.	ΔSucc(%)	Lat.(ms)	ΔLat(%)	TFLOPs	ΔTF(%)
OpenVLA	0.745	–	57.588	–	2.070	–	0.665	–	57.277	–	2.045	–
OpenVLA + CTR	0.760	+2.0	55.172	-4.2	1.664	-19.6	0.710	+6.8	54.887	-4.2	1.657	-18.9
OpenVLA-OFT	0.950	–	87.248	–	4.499	–	0.980	–	87.693	–	4.466	–
OpenVLA-OFT + CTR	0.960	+1.1	85.895	-1.6	3.949	-12.2	0.980	+0.0	85.512	-2.5	3.915	-12.3
Model / Setting	**Goal**						**Long**					
	Succ.	**ΔSucc(%)**	**Lat.(ms)**	**ΔLat(%)**	**TFLOPs**	**ΔTF(%)**	**Succ.**	**ΔSucc(%)**	**Lat.(ms)**	**ΔLat(%)**	**TFLOPs**	**ΔTF(%)**
OpenVLA	0.725	–	59.260	–	2.023	–	0.465	–	67.118	–	2.061	–
OpenVLA + CTR	0.795	+9.7	54.604	-7.9	1.634	-19.2	0.480	+3.2	54.813	-18.3	1.675	-18.7
OpenVLA-OFT	0.965	–	87.693	–	4.444	–	0.945	–	86.455	–	4.487	–
OpenVLA-OFT + CTR	0.970	+0.5	83.815	-4.4	3.858	-13.2	0.950	+0.5	83.533	-3.4	3.847	-14.3

Table 1: Main results on LIBERO suites with per-task success rate and end-to-end efficiency. Latency is reported in ms and TFLOPs in trillions of FLOPs per step; totals are computed as the sum of the vision encoder and LLM measurements per task. Δ values are relative to the corresponding baseline for each model.

Setting	Move Near	Pick Coke	Drawer	Average
OpenVLA	49.6	17.3	32.4	33.1
OpenVLA + CTR	51.3	19.0	33.3	34.5

Table 2: SimplerEnv results (success rate, %) using the OpenVLA base checkpoint without task-specific fine-tuning. CTR uses the same hyperparameters as in LIBERO.

5.2 Results

5.2.1 Main Results on LIBERO

Table 1 presents the main results on the LIBERO benchmark. Across all four task suites, CTR improves inference efficiency while maintaining or improving task success rates.

Overall Performance Trends. On the OpenVLA backbone, CTR improves success rates across Spatial, Object, and Goal tasks, with the largest gain observed on the Goal suite (+9.7%). These gains are achieved while reducing total TFLOPs by approximately 18–20% and lowering end-to-end latency by up to 18.3%. This indicates that coordinated reuse can accelerate inference and improve policy stability in closed-loop control.

On OpenVLA-OFT, CTR preserves the already high success rates while still delivering efficiency gains (12–14% TFLOPs reduction). The smaller improvements are expected given the strong baseline. Importantly, CTR does not degrade performance in this regime.

Long-Horizon Tasks. Long-horizon tasks benefit particularly from CTR. On OpenVLA, CTR improves success rate from 0.465 to 0.480 while reducing latency by 18.3%. This suggests that coordinated cross-modal reuse stabilizes inference across consecutive frames, which is especially helpful when errors can accumulate over long action sequences.

Efficiency-Accuracy Trade-off. Across all suites, CTR achieves efficiency gains without a clear accuracy–efficiency trade-off in the reported results. In several cases (e.g., Goal and Object tasks), success rates improve alongside reduced computation, indicating that suppressing redundant visual updates can reduce noise and improve temporal consistency.

5.2.2 Cross-Environment Evaluation on SimplerEnv

To evaluate generalization beyond the LIBERO benchmark, we report results on SimplerEnv in Table 2. All experiments use the OpenVLA base checkpoint without task-specific fine-tuning, and CTR parameters are kept identical to those used on LIBERO.

CTR consistently improves success rates across all three tasks, yielding an average improvement of 1.4 percentage points. The largest relative gain is observed on the Pick Coke task, which involves precise grasping under varying object poses. These results demonstrate that CTR generalizes across environments with different visual statistics and task structures, without requiring parameter tuning or retraining.

The consistent gains on SimplerEnv suggest that CTR captures a general property of robotic manipulation: most visual regions remain temporally stable, and coordinated reuse of such regions improves robustness across environments.

LLMCache, VECache	Spatial					Object				
	Succ.	VE Lat. (ms)	VE TFLOPs	LLM Lat. (ms)	LLM TFLOPs	Succ.	VE Lat. (ms)	VE TFLOPs	LLM Lat. (ms)	LLM TFLOPs
(✗, ✗)	0.745	12.48	0.090	32.62	1.888	0.665	12.38	0.093	32.50	1.857
(✓, ✗)	0.750	12.51	0.091	31.50	1.475	0.705	12.42	0.094	31.42	1.468
(✗, ✓)	0.750	11.76	0.094	32.69	1.888	0.685	11.63	0.095	32.40	1.857
(✓, ✓)	0.760	11.90	0.092	31.36	1.480	0.710	11.77	0.093	31.32	1.470
LLMCache, VECache	**Goal**					**Long**				
	Succ.	**VE Lat. (ms)**	**VE TFLOPs**	**LLM Lat. (ms)**	**LLM TFLOPs**	**Succ.**	**VE Lat. (ms)**	**VE TFLOPs**	**LLM Lat. (ms)**	**LLM TFLOPs**
(✗, ✗)	0.725	13.43	0.094	32.40	1.834	0.465	17.32	0.091	32.47	1.878
(✓, ✗)	0.780	13.33	0.094	31.21	1.440	0.470	17.35	0.091	31.29	1.482
(✗, ✓)	0.755	11.98	0.095	32.27	1.834	0.465	11.67	0.095	32.55	1.878
(✓, ✓)	0.795	11.76	0.093	31.06	1.447	0.480	11.73	0.093	31.35	1.488

Table 3: Switch ablation on OpenVLA. We independently enable or disable LLM-side cache reuse and VE-side reuse, and report per-task success rate, VE/LLM latency (ms), and TFLOPs. Latency is measured with CUDA events and averaged after warmup.

Mask Variant	Spatial	Object	Goal	Long	Avg.
CTR (ours)	76.0	71.0	79.5	48.0	68.6
LLM mask high	73.5	66.5	74.5	39.0	63.4
LLM mask low	73.5	70.0	73.0	40.0	64.1
VE mask high	72.5	70.0	73.0	39.5	63.8
VE mask low	74.0	70.0	79.0	40.0	65.8

Table 4: Mask-consistency ablation on OpenVLA (success rate, %). We perturb the similarity threshold on only one side (LLM or VE) while keeping the other side fixed. CTR with a consistent mask performs best overall.

5.2.3 Switch Ablation: Vision-Side vs. LLM-Side Reuse

To isolate the contributions of vision-encoder-side reuse and language-model-side reuse, we conduct a switch ablation study on OpenVLA, reported in Table 3. We independently enable or disable reuse in the vision encoder (ViT) and the language model (VLA cache), resulting in four configurations.

Independent Effects. Enabling reuse only on the language-model side reduces LLM latency and TFLOPs while slightly improving success rates, confirming prior findings that reusing cached key–value states is effective for VLA inference. Enabling reuse only on the vision side reduces ViT latency and TFLOPs and yields moderate performance gains, indicating that reusing static visual patches does not harm—and can even help—policy execution.

Coordinated Reuse. The full CTR configuration, where both reuse mechanisms are enabled and driven by a unified mask, consistently outperforms either single-sided variant. This indicates that coordinating reuse across modalities helps preserve visual–language alignment.

Complementarity. The ablation results highlight the complementary nature of vision-side and LLM-side reuse. Vision-side reuse primarily reduces early-stage visual computation, while LLM-side reuse skips recomputation for reusable positions at selected layers, reducing prefill compute. CTR combines these benefits while maintaining consistent token semantics across modalities.

5.2.4 Mask Consistency Ablation

Table 4 reports success rates when the reuse mask is perturbed on only one side of the model. Compared to the consistent-mask baseline (OpenVLA + CTR), all mismatched variants reduce average success rate by 2.8–5.2 points, with the largest drops on the Long-horizon suite. This indicates that mask agreement across vision and language modules is beneficial for stable performance, and supports using the unified mask as the default configuration.

6 Conclusion

We presented CTR, a training-free approach for efficient and accurate inference in vision–language–action (VLA) models based on coordinated cross-modal token reuse. CTR employs a unified mask to synchronize reuse decisions across the vision encoder and the language model, selectively reusing static and task-irrelevant visual tokens while recomputing dynamic or critical regions. Experiments across multiple manipulation benchmarks and VLA backbones show that CTR reduces inference cost while maintaining or improving task success rates, highlighting cross-modal coordination as an effective way to exploit temporal redundancy in VLA inference.

Limitations

While CTR achieves consistent reductions in computation, the corresponding end-to-end latency improvement is smaller than the TFLOPs reduction, mainly due to engineering factors such as kernel launch overhead, memory access patterns, and limited overlap between computation and data movement; as a result, theoretical compute savings are not always fully reflected in wall-clock latency, especially at smaller batch sizes. In addition, although CTR reuses visual representations across frames, it does not transform the vision encoder into a shorter-sequence model: self-attention in the ViT is still computed over the full $P \times P$ token grid rather than a reduced sequence. This design choice keeps the method training-free and minimally invasive, but also limits the maximum speedup achievable compared to approaches that explicitly shorten the attention sequence.

References

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, and 1 others. 2024. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2022. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*.

Qingqing Cao, Bhargavi Paranjape, and Hannaneh Hajishirzi. 2023. Pumer: Pruning and merging tokens for efficient vision language models. *arXiv preprint arXiv:2305.17530*.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. 2024. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *Computer Vision – ECCV 2024*.

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. 2024. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*.

Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. 2022. Evit: Expediting vision transformers via token reorganizations. In *International Conference on Learning Representations*.

Chenghao Liu, Jiachen Zhang, Chengxuan Li, Zhimu Zhou, Shixin Wu, Songfang Huang, and Huiling Duan. 2025. Ttf-vla: Temporal token fusion via pixel-attention integration for vision-language-action models. *arXiv preprint arXiv:2508.19257*.

Jiaming Liu, Mengzhen Liu, Zhenyu Wang, Lily Lee, Kaichen Zhou, Pengju An, Senqiao Yang, Renrui Zhang, Yandong Guo, and Shanghang Zhang. 2024. Robomamba: Multimodal state space model for efficient robot reasoning and manipulation. *arXiv preprint arXiv:2406.04339*.

Lingchen Meng, Hengduo Li, Bor-Chun Chen, Shiyi Lan, Zuxuan Wu, Yu-Gang Jiang, and Ser-Nam Lim. 2022. Adavit: Adaptive vision transformers for efficient image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12309–12318.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. 2024. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. 2021. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In *Advances in Neural Information Processing Systems*, pages 13937–13949.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, and 1 others. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, and 1 others. 2024. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2409.12514*.

Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. 2025. Vla-cache: Efficient vision-language-action manipulation via adaptive token caching. *arXiv preprint arXiv:2502.02175*.

Yantai Yang, Yuhao Wang, Zichen Wen, Zhongwei Luo, Chang Zou, Zhipeng Zhang, Chuan Wen, and Linfeng Zhang. 2025. Efficientvla: Training-free acceleration and compression for vision-language-action models. *arXiv preprint arXiv:2506.10100*.

Yang Yue, Yulin Wang, Bingyi Kang, Yizeng Han, Shenzhi Wang, Shiji Song, Jiashi Feng, and Gao Huang. 2024. Deer-vla: Dynamic inference of multimodal large language models for efficient robot execution. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Rongyu Zhang, Menghang Dong, Yuan Zhang, Liang Heng, Xiaowei Chi, Gaole Dai, Li Du, Dan Wang, Yuan Du, and Shanghang Zhang. 2025. Mole-vla: Dynamic layer-skipping vision language action model via mixture-of-layers for efficient robot manipulation. *arXiv preprint arXiv:2503.20384*.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and 1 others. 2024. Sparsevlm: Visual token sparsification for efficient vision-language model inference. *arXiv preprint arXiv:2410.04417*.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, and 1 others. 2023. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR.

A Appendix

A.1 Experimental Environment

All experiments are conducted under a unified hardware and software configuration to ensure fair comparison across methods and benchmarks. Unless otherwise stated, the evaluation environment is as follows.

- **GPU:** NVIDIA RTX 5880 Ada GPU with 48 GB memory.
- **CPU:** Dual-socket AMD EPYC 9654 processors (96 cores per socket, 192 cores total, 384 threads).
- **CUDA / Driver:** CUDA 13.0 with NVIDIA driver version 580.65.06.

All latency measurements are obtained using CUDA events and exclude environment simulation and robot actuation overhead.

A.2 Vision Encoder Configuration

Unless otherwise specified, the visual observation at each timestep is resized to 224×224 and processed by a ViT-based vision encoder. Images are partitioned into non-overlapping patches of size 14×14, resulting in

$$P = (224/14)^2 = 256$$

visual patches per frame. We evaluate two widely used vision backbones in OpenVLA-style models: DINOv2 and SigLIP. These configurations are kept identical across baselines and CTR-enabled variants to isolate the effect of coordinated token reuse.

A.3 Benchmarks

We evaluate CTR on two complementary simulation benchmarks that cover a wide spectrum of manipulation behaviors.

LIBERO is used as the primary benchmark and includes four task suites: *Spatial* (precise object placement and spatial reasoning), *Object* (single-object manipulation with diverse geometries), *Goal* (goal-conditioned, multi-step manipulation), and *Long* (long-horizon tasks requiring temporal consistency). For LIBERO experiments, we use the official OpenVLA-7B checkpoint that has been fine-tuned on the corresponding LIBERO task suites.

To assess generalization beyond LIBERO, we additionally evaluate on **SimplerEnv**, a tabletop manipulation benchmark with different object layouts and visual statistics. We report results on three representative tasks: *Move Near Object*, *Pick Coke Can*, and *Drawer Operations*. For SimplerEnv, we use the original OpenVLA-7B checkpoint without any task-specific fine-tuning. All CTR hyperparameters are kept identical to those used in LIBERO experiments.

A.4 Model Variants and Checkpoints

We evaluate two VLA backbones: OpenVLA and OpenVLA-OFT. All experiments use officially released OpenVLA-7B checkpoints. CTR is applied strictly at inference time and does not modify model weights, training data, or optimization procedures. Both the baseline and CTR-enabled models share identical architectures and token layouts.

A.5 Qualitative Visualization of an Inference Trajectory

Figure 3 visualizes a complete closed-loop VLA inference trajectory from a single manipulation episode. The figure highlights that most visual regions remain unchanged across consecutive timesteps, which motivates exploiting temporal redundancy via token reuse.

Figure 3: A complete closed-loop VLA inference trajectory from a single manipulation episode. Consecutive frames exhibit strong temporal redundancy, with only localized regions changing over time.